A mathematical resolution to log transformations and the binning effect in applied processing of data in flow cytometry

Vidal J. Patricio
Florida International University

DOI: 10.25148/etd.FI15071600
Follow this and additional works at: https://digitalcommons.fiu.edu/etd
Part of the Electrical and Computer Engineering Commons

Recommended Citation
https://digitalcommons.fiu.edu/etd/2006

This work is brought to you for free and open access by the University Graduate School at FIU Digital Commons. It has been accepted for inclusion in FIU Electronic Theses and Dissertations by an authorized administrator of FIU Digital Commons. For more information, please contact dcc@fiu.edu.
A MATHEMATICAL RESOLUTION TO LOG TRANSFORMATIONS
AND THE BINNING EFFECT IN APPLIED PROCESSING OF DATA
IN FLOW CYTOMETRY

A dissertation submitted in partial fulfillment of the
requirements for the degree of
DOCTOR OF PHILOSOPHY
in
ELECTRICAL ENGINEERING

by
Patricio J. Vidal
2003
To: Dean Vish Prasad  
College of Engineering

This dissertation, written by Patricio J. Vidal, and entitled A Mathematical Resolution to Log Transformations and the Binning Effect in Applied Processing of Data in Flow Cytometry, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Malcolm Heimer

Armando Barreto

Malek Adjouadi, Major Professor

Date of Defense: March 17, 2003

The dissertation of Patricio J. Vidal is approved.

Dean Vish Prasad  
College of Engineering

Dean Douglas Wartzok  
University Graduate School

Florida International University, 2003
DEDICATION

I dedicate this dissertation to my wonderful family, which makes all the effort and sacrifice worthwhile.
ACKNOWLEDGMENTS

I wish to thank the members of the committee, Dr. Malcolm Heimer and Dr. Armando Barreto, for their interest, support, and enthusiasm through this work. Dr. Heimer and Dr. Barreto had the greatest disposition to help and made possible the oral examination and the dissertation defense even with short notice and sacrificing their free time.

I am also grateful for the support provided by the National Science Foundation Grant EIA-9906600 and the Office of Naval Research Grant N00014-99-1-0952, which have made this work possible.

I also would like to thanks Beckman Coulter for their continuous support and for making possible the filing of a patent application for this work.

Finally, I wish to extent my greatest appreciation to my major professor, Dr. Malek Adjouadi, who has helped, guided, motivated, and supported me since back 1994. Dr. Malek gave my family and me the opportunity of having a better future in this country, and we will be always grateful for that.
ABSTRACT OF THE DISSERTATION

A MATHEMATICAL RESOLUTION TO LOG TRANSFORMATIONS AND THE BINNING EFFECT IN APPLIED PROCESSING OF DATA IN FLOW CYTOMETRY

by

Patricio J. Vidal

Florida International University, 2003

Miami, Florida

Professor Malek Adjouadi, Major Professor

This dissertation develops a new mathematical approach that overcomes the effect of a data processing phenomenon known as “histogram binning” inherent to flow cytometry data. A real-time procedure is introduced to prove the effectiveness and fast implementation of such an approach on real-world data. The histogram binning effect is a dilemma posed by two seemingly antagonistic developments: (1) flow cytometry data in its histogram form is extended in its dynamic range to improve its analysis and interpretation, and (2) the inevitable dynamic range extension introduces an unwelcome side effect, the binning effect, which skews the statistics of the data, undermining as a consequence the accuracy of the analysis and the eventual interpretation of the data.

Researchers in the field contended with such a dilemma for many years, resorting either to hardware approaches that are rather costly with inherent calibration and noise effects; or have developed software techniques based on filtering the binning effect but without successfully preserving the statistical content of the original data.
The mathematical approach introduced in this dissertation is so appealing that a patent application has been filed. The contribution of this dissertation is an incremental scientific innovation based on a mathematical framework that will allow researchers in the field of flow cytometry to improve the interpretation of data knowing that its statistical meaning has been faithfully preserved for its optimized analysis. Furthermore, with the same mathematical foundation, proof of the origin of such an inherent artifact is provided.

These results are unique in that new mathematical derivations are established to define and solve the critical problem of the binning effect faced at the experimental assessment level, providing a data platform that preserves its statistical content.

In addition, a novel method for accumulating the log-transformed data was developed. This new method uses the properties of the transformation of statistical distributions to accumulate the output histogram in a non-integer and multi-channel fashion. Although the mathematics of this new mapping technique seem intricate, the concise nature of the derivations allow for an implementation procedure that lends itself to a real-time implementation using lookup tables, a task that is also introduced in this dissertation.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER 2 FLOW CYTOMETRY AND CELL POPULATIONS IN THE CONTEXT OF THE BINNING EFFECT</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Definition of Flow Cytometry</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Data Processing</td>
<td>8</td>
</tr>
<tr>
<td>CHAPTER 3 CURRENT LOG TRANSFORMATION APPROACHES</td>
<td>10</td>
</tr>
<tr>
<td>3.1 FIR Filtering</td>
<td>10</td>
</tr>
<tr>
<td>3.2 High-resolution A/D Converters</td>
<td>12</td>
</tr>
<tr>
<td>3.3 Electronic Log-Amplifiers</td>
<td>14</td>
</tr>
<tr>
<td>3.4 Summary</td>
<td>16</td>
</tr>
<tr>
<td>CHAPTER 4 SOLVING THE BINNING EFFECT</td>
<td>18</td>
</tr>
<tr>
<td>4.1 Correcting the Mapping Process of the Transformed Data</td>
<td>26</td>
</tr>
<tr>
<td>4.2 Uniform Approximation</td>
<td>30</td>
</tr>
<tr>
<td>4.3 Linear Approximation</td>
<td>33</td>
</tr>
<tr>
<td>4.4 Summary</td>
<td>37</td>
</tr>
<tr>
<td>CHAPTER 5 NON-INTEGER BIN CONTRIBUTIONS AND REAL-TIME IMPLEMENTATION</td>
<td>38</td>
</tr>
<tr>
<td>5.1 Non-integer Bin Contribution</td>
<td>38</td>
</tr>
<tr>
<td>5.2 Use of Lookup Tables for Real-Time Application</td>
<td>44</td>
</tr>
<tr>
<td>5.3 Results</td>
<td>47</td>
</tr>
<tr>
<td>5.4 Summary</td>
<td>48</td>
</tr>
<tr>
<td>CHAPTER 6 REAL DATA IMPLEMENTATIONS</td>
<td>50</td>
</tr>
<tr>
<td>6.1 Ten Bits Resolution Data</td>
<td>50</td>
</tr>
<tr>
<td>6.2 Twenty Bits Resolution Data</td>
<td>52</td>
</tr>
<tr>
<td>6.3 Summary</td>
<td>56</td>
</tr>
<tr>
<td>CHAPTER 7 MATHEMATICAL FRAMEWORK AND IMPLEMENTATION OF THE MULTI-DIMENSIONAL CASE</td>
<td>57</td>
</tr>
<tr>
<td>7.1 Introduction</td>
<td>57</td>
</tr>
<tr>
<td>7.2 Implementation</td>
<td>58</td>
</tr>
<tr>
<td>7.3 Mathematical Framework and Implications</td>
<td>61</td>
</tr>
<tr>
<td>7.4 Mathematical Extension to N-Dimensional</td>
<td>65</td>
</tr>
<tr>
<td>7.5 Summary</td>
<td>66</td>
</tr>
<tr>
<td>CHAPTER 8 CONCLUSION</td>
<td>67</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>72</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>74</td>
</tr>
<tr>
<td>VITA</td>
<td>88</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1. Simulated data of two overlapped Gaussian cell populations.</td>
<td>2</td>
</tr>
<tr>
<td>Figure 1.2. Logarithmic Transformation and the binning effect.</td>
<td>2</td>
</tr>
<tr>
<td>Figure 1.3. Advertising postal.</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2.1. Flow cytometer diagram.</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2.2. Peaks and valley in a 1-D histogram.</td>
<td>8</td>
</tr>
<tr>
<td>Figure 2.3. Two clusters in a 2-D histogram.</td>
<td>9</td>
</tr>
<tr>
<td>Figure 3.1. FIR filtering of binned histograms.</td>
<td>11</td>
</tr>
<tr>
<td>Figure 3.2. Electronic log-amplifier diagram.</td>
<td>16</td>
</tr>
<tr>
<td>Figure 4.1. Logarithmic Transformation and the binning effect.</td>
<td>19</td>
</tr>
<tr>
<td>Figure 4.2. Log Transformation function in log-scale.</td>
<td>20</td>
</tr>
<tr>
<td>Figure 4.3. Log Transformation function in linear-scale.</td>
<td>20</td>
</tr>
<tr>
<td>Figure 4.4. Details of the low range of the log-transformation curve.</td>
<td>21</td>
</tr>
<tr>
<td>Figure 4.5. Details of the “holes” in the transformed histogram.</td>
<td>22</td>
</tr>
<tr>
<td>Figure 4.6. Details of the middle range of the log transformation curve.</td>
<td>22</td>
</tr>
<tr>
<td>Figure 4.7. Details of the high range of the log transformation curve.</td>
<td>23</td>
</tr>
<tr>
<td>Figure 4.8. Log Transformation on Middle Range Channels.</td>
<td>24</td>
</tr>
<tr>
<td>Figure 4.9. Details of the “spikes” in the middle range channels.</td>
<td>25</td>
</tr>
<tr>
<td>Figure 4.10. Bin contribution for a low range channel.</td>
<td>26</td>
</tr>
<tr>
<td>Figure 4.11. Bin contribution for a high range channel.</td>
<td>26</td>
</tr>
<tr>
<td>Figure 4.12. Delta Approximation.</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4.13. Transformed delta approximation.</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4.14. Accumulated transformed data.</td>
<td>30</td>
</tr>
<tr>
<td>Figure 4.15. Details of Uniform Approximation.</td>
<td>31</td>
</tr>
<tr>
<td>Figure 4.16. Log Transformed Uniform Approximation.</td>
<td>32</td>
</tr>
<tr>
<td>Figure 4.17. Details of errors in Uniform Approximation.</td>
<td>32</td>
</tr>
<tr>
<td>Figure 4.18. Detail of Linear Approximation.</td>
<td>34</td>
</tr>
<tr>
<td>Figure 4.19. Log Transformed Linear Approximation.</td>
<td>35</td>
</tr>
<tr>
<td>Figure 4.20. Valley in the output histogram.</td>
<td>36</td>
</tr>
<tr>
<td>Figure 4.21. Valley mapped to the input domain.</td>
<td>36</td>
</tr>
</tbody>
</table>
Figure 5.1. Mapping for input data range [16, 17].................................................................40
Figure 5.2. Bin contributions for a low range channel..............................................................40
Figure 5.3. Mapping for input data range [151, 152]...............................................................41
Figure 5.4. Bin contribution for a high range channel.............................................................42
Figure 5.5. Mapping for input data range [161, 162]...............................................................43
Figure 5.6. Bin contribution for a high range channel that falls in a bin boundary.................43
Figure 5.7. Non-integer bin contribution example....................................................................45
Figure 5.8. Results of the new histogram accumulation method..............................................48
Figure 6.1. Real 10 bit flow cytometer data.............................................................................50
Figure 6.2. Standard log-transformation..................................................................................51
Figure 6.3. Results of the new technique................................................................................51
Figure 6.4. Four decades transformation................................................................................53
Figure 6.5. Real 20-bit flow cytometry data.............................................................................53
Figure 6.6. Transformed data using the standard log-transformation.......................................54
Figure 6.7. Whole range histogram (4 decades)......................................................................55
Figure 6.8. First decade detail..................................................................................................55
Figure 6.9. Second decade detail............................................................................................55
Figure 7.1. 3-D view of a simulated 2-D distribution...............................................................57
Figure 7.2. Top view of a simulated 2-D distribution...............................................................58
Figure 7.3. Log transformation for 2-D distribution...............................................................59
Figure 7.4. 3-D view of the ideal transformation.......................................................................59
Figure 7.5. Top view of the ideal transformation......................................................................60
Figure 7.6. Binning effect in two dimensions..........................................................................61
Figure 7.7. Detail of the uniform approximation of the data....................................................63
Figure 7.8. 3-D view of the results when using the uniform approximation............................64
Figure 7.9. Top view of the results of Figure 7.8....................................................................64
CHAPTER 1

Introduction

Current approaches used to analyze flow cytometry data, presented as 1-D or 2-D data distributions or histograms involve a preprocessing step known as the log-transformation, which is used to increase the dynamic range of such distributions in order to facilitate their analysis and henceforth enhance their interpretation. This first preprocessing step is most common in the realm of flow cytometry.

Ironically, as this step attempts to broaden the dynamic range of the data distributions, it introduces a well know artifact called “picked fencing” or “binning effect” that undermines the main aspect of the solution it is intended to reinforce, that of enhancing the analysis of the data. At the source of this problem is the lack of preserving the essence of the data by distorting irreversibly its statistical properties. The binning effect can be best described as a combination of the following two undesired outcomes:

- Low histogram channels are not accumulated although the input data is continuous; and
- High histogram channels reveal a non-uniform distribution as data is unduly compressed.

Figure 1.1 shows the distribution of simulated data representing two overlapped Gaussian cell populations. This input data represents a typical example of flow cytometry data that requires to be dynamically expanded in order for it to be properly analyzed.
Figure 1.1. Simulated data of two overlapped Gaussian cell populations.

Figure 1.2. Logarithmic Transformation and the binning effect.
Figure 1.2 shows in black the result of the log-transformation and accumulation of the digitized input data depicted in Figure 1.1. Figure 1.2 also shows in gray what would be the ideal continuous transformation sought of the same input data.

This transformation data is beset with contentious aspects: even if dynamically expanded, introduces a significant binning artifact, and furthermore, the result from the log-transformation is not suitable for data analysis or any direct and consequential statistical analysis. Figure 1.3 shows advertising postal distributed by a manufacturer of flow cytometers that uses analog log-transformation modules to avoid the binning effect.

---

**Figure 1.3. Advertising postal.**
This dissertation introduces a new mathematical approach to address and redress the effect of binning. By redress we mean not attenuating the effect of binning but eliminating it altogether.

Chapter 2 provides an introduction to flow cytometry in the context of the binning effect, knowing that the field of flow cytometry is broad in both its theoretical context as well as in its experimental developments. Within the framework of this dissertation, the focus in this Chapter is placed on the origin of the binning effect in relation to the logarithmic principles used in flow cytometry.

Chapter 3 assesses the problematic issues of the current approaches with their respective advantages and limitations brought to light. The emphasis is placed on those studies that contended with the binning phenomenon, from the hardware point of view as well as through well-known software methods.

Chapter 4 provides the mathematical foundation that was established in this dissertation to solve the binning effect. Different approximations that can be used to generate a practical solution are detailed. The soundness of this mathematical approach is fundamentally revealed through two different but inextricably linked issues.

The mathematical foundation established eliminates the binning effect in log-transformed data.

The same mathematical foundation establishes the origin of such an effect and derives the mathematical expression that describes it.

Consequently, Chapter IV is the centerpiece contribution of this dissertation.
Chapter 5 describes a novel method for accumulating the log-transformed data and its applicability to real-time implementation. The objective at this stage is to prove that although the mathematical foundation established is intricate in principle, but it does lend itself to real-time implementation. Real-time is defined in this case as being less than 200 milliseconds (ms), which is experimentally analogous to the rate of accumulating a histogram free of the binning effect as data is being collected. This is considered as another major contribution of this dissertation.

Chapter 6 demonstrates the binning effect and the results of the new technique in real flow cytometry data. Different data bit resolutions, 10-bit and 20-bit are used in the 1-D implementation. In both resolution cases, results are shown to address effectively the binning effect in contrast to existing techniques.

Chapter 7 presents the binning effect and the results of the new technique in two-dimensional histograms. The two-dimensional case is detailed both mathematically and experimentally in view that higher dimensions are perceived as predicable extensions of the 2-D case.

Finally, Chapter 8 provides the conclusions, highlighting the nature of the binning effect and the results accomplished. In addition, recommendations for future work are presented in the context of refining implementation methods that will integrate higher dimensions (>3-D) while maintaining real-time computational requirements.
CHAPTER 2

Flow Cytometry and Cell Populations in the Context of the Binning Effect

To construct a general understanding of the binning effect from its physical meaning and the relevance to the logarithmic principles, this Chapter provides the key aspects of flow cytometry in direct relation to the problems associated with the log transformation and the resulting consequence of the binning effect.

2.1 Definition of Flow Cytometry

Flow cytometry derives from the quantitative measurement (meter) of structural features of cells (cyto) traveling in a controlled flow through a series of primarily optical detectors. Figure 2.1 shows a simplified diagram of a typical flow cytometer:

![Flow Cytometer Diagram](https://via.placeholder.com/150)

**Figure 2.1. Flow cytometer diagram.**

In this diagram representation, cells in suspension flow one at a time at rates of over 100 cells per second in a capillary tube 50 – 150 μm in diameter [3, 4]. A laser beam is focused in the stream and intercepts each cell as they travel one by one. Optical sensors are located in such a way that they can measure the forward scatter intensity (proportional to cell properties...
diameter), the orthogonal scatter intensity (proportional to cell granularity), and fluorescence intensities at various wavelengths [3] to assess textural and other biological properties. Each cell measurement is digitized and transmitted to a computer for further processing.

Because different cell types can be differentiated by the statistical properties of their measurements, flow cytometry can be used to separate and count different cell populations in a mixture as in blood samples for example. In addition, the cells can be stained with fluorescent reagents or dyes that bind to specific biochemical receptors of certain cells, allowing the measurement of biological and biochemical properties.

The binning effect manifests itself especially in fluorescence because this phenomenon introduces a wide dynamic range, where the use of a log-transformation is necessary in order to properly analyze the data.

Flow cytometers have been commercially available since the early 1970's, and their use has been increasing since then. There are about 7,000 flow cytometers in use worldwide. The most numerous flow cytometers are those used for complete blood cell counts in clinical laboratories. Flow cytometers are found in all major biological research institutions and they are numerous in medical centers, where they are used for diagnosis as well as research. Their major diagnostic use is in chromosome count and cell cycle analysis of cancers. Most importantly, flow cytometry has been the method of choice for monitoring AIDS patients [3], and they are used to intensively study lymphomas and leukemia for surface markers of diagnostic and prognostic value.
2.2 Data Processing
As mentioned in the previous section, the goal of flow cytometry is to separate and enumerate cell populations. This goal, in the field of flow cytometry, is usually attained by processing 1-D or 2-D histograms. These are statistical events which define a frequency of occurrence over any given channel.

In 1-D histograms, cell populations can be separated by finding peaks and valleys. Figure 2.2 shows a 1-D histogram example with its peaks and valleys identified.

![1-D Histogram Example](image)

**Figure 2.2. Peaks and valley in a 1-D histogram.**

In 2-D histograms, cell populations can be separated by locating clusters through cluster seeking methods. After separating the different cell populations into clusters, each cell is enumerated and tagged. It is noted that in using either the 1-D or the 2-D case, researches continue to contend with the elusive problem of data overlap. Although this problem of
overlap is beyond the scope of this dissertation, it is asserted that a resolution to the binning effect will improve the prospects of assessing more accurately the extent of the overlap.

Figure 2.3 shows a 2-D histogram with two located clusters with a non-linear decision function separating them. Each cluster represents a distinct cell population. The curved black line represents the boundary generated by a cluster detection algorithm applied to the 2-D histogram data. After this boundary is generated, each data event is then enumerated and classified using the dividing boundary as a discriminating criteria.

![Figure 2.3. Two clusters in a 2-D histogram.](image)
CHAPTER 3

Current Log Transformation Approaches

As means to counter the undesired artifacts associated to the binning effect, which undoubtedly skew the analysis/interpretation of the results, experts in the field have come to rely on key filtering techniques, such as FIR (finite impulse response) filtering, that attempt to attenuate such effect without introducing changes that might undermine the statistical properties of the original data. As we shall elicit this dilemma, these filtering approaches are fraught with an irreconcilable issue: that of relating the degree at which we need to filter the artifact with the point at which we still believe that the filtered data is retaining statistics similar of the original data.

Other experts have, for a same goal, taken a hardware approach, particularly that of making use of analog to digital (A/D) converters or log amplifiers in order to electronically transform the input signal in its analog form as an enhancement step to its eventual digitization. These notable approaches, as we will explain, have to contend with the issues of cost effectiveness, noise and instability inherent to hardware approaches.

Both software and hardware methods are summarized in the following sections.

3.1 FIR Filtering

Typically, to resolve the binning effect issue, averaging schemes based on finite impulse response (FIR) filters are used [1]. Most effective FIR filtering schemes are achieved using the traditional 1-2-1 3-tap FIR filter. These filters have positive attributes provided that the following constraints are met: (1) to maintain the area under the curve requires that the sum of the FIR filter coefficients must equal to 1, and that the appropriate boundary conditions
have been met, (2) to prevent the data from being skewed, the filter coefficients must be symmetric around the center (this requires that the number of filter taps in the FIR filter be odd), and the individual distributions need to be symmetrical and not overlapping, which is not always true when analyzing log-transformed data from cell populations.

![FIR filtering scheme after 20 passes](image1)

![FIR filtering scheme after 100 passes](image2)

![FIR filtering scheme after 200 passes](image3)

![FIR filtering scheme after 500 passes](image4)

**Figure 3.1. FIR filtering of binned histograms.**

Figure 3.1 illustrates the effects of the FIR filtering scheme after 20, 100, 200 and 500 passes of the binned histogram of Figure 1.2 through the traditional 1-2-1 FIR filter. The black curve shows the filtered histogram, and the gray curve shows the ideal transformed data.
Qualitative analyses of the results of the FIR smoothing may suggest that there is an optimal number of passes required for the closest approximation to the actual distribution. However, FIR filtering cannot be optimal because the log transformation function has been applied to initially Gaussian populations, and the resultant Log-Normal populations will be skewed. Clearly, after 500 passes, this can be observed.

Another factor that makes the use of FIR filtering techniques inappropriate is the fact that in real life cell populations tend to be overlapped and therefore any excessive filtering will skew their statistical properties and merge the populations together.

FIR filtering in the literature has constituted the predominant approach used through software development. A different research direction has addressed the problem from a hardware point of view, which included the use of either A/D converters or log amplifiers. These hardware method are summarized below:

### 3.2 High-resolution A/D Converters

There is a vast variety of A/D converters (ADCs) on the market today. ADCs vary in resolutions, speeds, accuracies, and architectures, covering a broad range of performance needs. ADCs are found in a variety of applications in data-acquisition, communications, imaging, instrumentation, consumer electronics, and interfacing for signal processing [12, 13, 14].

The most popular ADC architectures available today are flash, pipelined, successive approximations, and sigma-delta. All ADCs require one or more steps involving comparison of an input signal with a reference [15].
The flash architecture uses a set of $2^n-1$ comparators to directly measure an analog signal to a resolution of n bits. The flash architecture has the advantage of being very fast, because the conversion occurs in a single ADC cycle. The disadvantage of this approach is that it requires a large number of comparators [15].

The pipelined architecture effectively overcomes the limitations of the flash architecture. A pipelined converter divides the conversion task into several consecutive stages. Each of these stages consists of a sample and hold circuit, an $m$-bit ADC, and an $m$-bit D/A converter (DAC). First the sample and hold circuit of the first stage acquires the signal. The $m$-bit flash converter then converts the sampled signal to digital data. The conversion result forms the most significant bits of the digital output. This same digital output is fed into an $m$-bit digital-to-analog converter, and its output is subtracted from the original sampled signal. The residual analog signal is then amplified and sent on to the next stage in the pipeline to be sampled and converted as it was in the first stage. This process is repeated through as many stages as are necessary to achieve the desired resolution [15].

The successive-approximations architecture works like an old-fashioned balance scale. On one side of the scale, we place the sampled unknown quantity. On the other side, we place a weight that has the value of $\frac{1}{2}$ of full-scale and compare the two values. This first weight represents the most significant bit. If the unknown quantity is larger, the $\frac{1}{2}$-scale weight is retained; if the unknown quantity is smaller, it is removed. This series of steps is repeated n times, using successively smaller weights in binary progression (e.g., $1/4$, $1/8$, $1/16$, $1/32$, ... $1/2^n$ of full scale) until the desired resolution, $n$, is attained [15].
The sigma-delta architecture takes a fundamentally different approach from those outlined above. In its most basic form, a sigma-delta converter consists of an integrator, a comparator, and a single-bit DAC. The output of the DAC is subtracted from the input signal. The resulting signal is then integrated, and the integrator output voltage is converted to a single-bit digital output (1 or 0) by the comparator. The resulting bit becomes the input to the DAC, and the DAC's output is subtracted from the ADC input signal, etc. This closed-loop process is carried out at a very high "oversampled" rate.

A common hardware approach designed to overcome the binning effect is one that makes use of high resolution ADCs so that the log transformation will not have discontinuities in the lower range of histogram channels.

This approach, which provides a perceivable improvement over the software approach, still contends with the following constraints and limitations:

- High-resolution ADCs are expensive.
- The amount of data the instruments need to send and store will increase proportionally with the increase in bit resolution of the A/D converter. Another expensive proposition.
- No matter what the resolution of the A/D converter used, the binning effect, although minimized, will still be present in the output histogram.

### 3.3 Electronic Log-Amplifiers

Current technology offers a broad range of logarithmic amplifiers which include Ultrafast wideband high-frequency logarithmic amplifiers, high-performance logarithmic amplifiers.
Their applications include both for both RF and IF applications, with the ability to measure signals up to 100dB dynamic range, from dc to 2.7GHz, and accuracies of +/-0.2dB. Log amplifiers provide a linear output voltage that is the decibel (dB) equivalent of the input signal. The designs are based on a multistage demodulating principle, a cascaded number of amplifier cells with a fixed gain of typically 10dB. The output of each gain stage is subsequently summed to provide the linear output voltage and in some devices a limited output is provided that represents the relative phase of the input signal.

Many researchers have experimented with logarithmic amplifiers, and the challenge has always been to expand the dynamic range of data recording, whether it be in biomedicine, seismic recordings, wireless communications, radar / military applications, low power instrumentation, high-accuracy optical power measurement in optical networks, to name a few.

There are new attempts at bypassing the use of log amplifiers. For example, at Beckman-Coulter, digital log transformation applications have replaced log amplifiers in applications that require fluorescence-base measurements. The Coulter Epics XL flow cytometer is now producing more linear and reproducible results with digital log transformation.

Research in the field of log amplification in general remains a challenging endeavor in addressing the following desired outcomes:

- Low output noise for increased accuracy
- Low Noise / Low Distortion Amplifiers
- Improved sensitivity
The approach considered in overcoming the histogram binning effect is to use Log-Amplifiers to electronically transform the signals before digitizing them. Figure 3.2 shows the diagram of an electronic log-amplifier setup which within a typical A/D converter after the log-amplification process is accomplished.

![Figure 3.2. Electronic log-amplifier diagram.](image)

This approach on the other hand contends with the following drawbacks:

- Requires the use of additional and expensive hardware.

- Logarithmic amplifiers are notoriously noisy and unstable.

- When linear data is also required, the instruments need to send and store twice the amount of data: the log-transformed data and the linear data.

### 3.4 Summary

This Chapter introduced the current and most applicable techniques designed to overcome the binning effect. It was shown that FIR filtering alone is not sufficient to accommodate for this phenomenon, as excessive filtering can dramatically skew the data and affect or even undermine the statistical properties of the original data. A contentious dilemma faced in this instance is one that seeks a practical balance between the level of filtering and the
preservation state of the statistical properties of the original data. Any solution will have, within this balance, compromised either on skewing the statistics of the data and hence skewing its analysis as a consequence, or maintaining some of the artifacts of the binning effect and hence limiting the accuracy in data interpretation.
CHAPTER 4

Solving the Binning Effect

A solution to the binning effect is introduced in this Chapter. Fundamentally, when logarithmically transformed data is accumulated into a frequency histogram with a finite number of channels (or bins), the binning effect produces three major, and very visible, artifacts as shown in the black curve in Figure 4.1:

1. "Holes": some of the low histogram channels are not accumulated although the input data is continuous.

2. "Spikes": data is compressed in the high histogram channels in a non-uniform manner, generating abrupt changes in amplitude.

3. "False Higher Peaks": the peak amplitude on the low histogram channels is artificially enlarged.

Equation (4.1) shows a standard logarithmic transformation used in flow cytometry:

\[
t(x) = \frac{\log(x/K)}{S}
\]

(4.1)

where \( t(x) \) is the output transformed value of input value \( x \), \( S \) and \( K \) are scaling constants defined as \( S = D/(B - 1) \) and \( K = 2^R/10^D \), with \( B \) being the number of bins in the output histogram, \( D \) is the number of desired decades, and \( R \) is the bit resolution of the A/D converter. This Equation maps the input data \( x \) from the full resolution of the A/D converter to the desired number of channels and decades in the output histogram.
Figure 4.1. Logarithmic Transformation and the binning effect.

Figure 4.2 illustrates in a log-scale the realization of the log-transformation shown in Equation (4.1) for an input bit resolution of 10 bits, with three decades, and with an output histogram of 256 channels. The log-scale used for the plot shows how the log-transformation covers the three desired decades in a linear fashion.

Figure 4.3 illustrates in a linear-scale the realization of the same log-transformation shown in Equation (4.1) for the same input bit resolution of 10 bits, with three decades, and with an output histogram of 256 channels.

The linear-scale used for the plot shows how the log-transformation expands the lower range channels and compresses the upper range channels.
Figure 4.2. Log Transformation function in log-scale.

Figure 4.3. Log Transformation function in linear-scale.
In the lower range of the logarithmic transformation, the input data is expanded so that consecutive input data values are mapped to non-consecutive channels as shown in Figure 4.4. In this figure consecutive input data values 1 and 2 are mapped to channels 0 and 25 respectively, and consecutive input data values 2 and 3 are mapped to channels 25 and 40 respectively, and likewise for all subsequent data points in the lower range. The “holes” are created at all those channels in the output histogram where the log-transformation did not provide a direct mapping.

Figure 4.4. Details of the low range of the log-transformation curve.

Figure 4.5 shows the “holes” in the low-range channels of the transformed data from Figure 1.1.
Figure 4.5. Details of the “holes” in the transformed histogram.

Figure 4.6. Details of the middle range of the log transformation curve.
Figure 4.7. Details of the high range of the log transformation curve.

In the middle and upper range of the histogram the logarithmic transformation actually compresses the data so that consecutive channels may be mapped to the same channel or bin. Figure 4.6 and Figure 4.7 illustrates the mapping of the middle and high range channels respectively. Input data range from 45 to 47 is mapped to channel 140, and similarly input data range from 958 to 984 is mapped to channel 253, and so forth.

It is noted that a “spike” is formed whenever there is a change in the number of consecutive input channels that are being mapped into the same bin. The size of the “spike” will depend on the relation between the change (usually one channel) and the number of channels mapped to the bin.
Figure 4.8 shows the log-transformation in the middle-range channels. In this range, the number of consecutives input channels that are being mapped into the same bin changes from one to two and to one again.

For example, one input channel maps to bin 141, two input channels map to bin 142, and one input channel to bin 143. When this transformed data is accumulated in the correspondent bins, the amplitude in bin 142 will be about twice of that in bin 141 and 143 generating a prominent “spike”.

![Graph showing log transformation on middle range channels](image)

**Figure 4.8. Log Transformation on Middle Range Channels.**

Figure 4.9 shows details of the “spikes” at the middle-range channels 142, 144, and 147. In the high-range channels, the change could be one channel over 20 to 30 channels mapped into the same bin, generating smaller “spikes”.

24
The binning effect is thus produced by the inconsistent assumption that the input range 
\([x_0, x_0 + 1)\) uniformly maps to the output range \([t(x_0), t(x_0) + 1)\), where \(t(x)\) is the transformation function of \(x\). In other words, it is assumed that the contribution of an input data value \(x_0\) is one unit in the output histogram bin \(t(x_0)\). This inconsistency has always been overlooked because it provides a simple mechanism to generate the output histogram: that is, for each input data value, one unit will be accumulated in the output histogram channel correspondent to the log-transformed value of the input data.

![Figure 4.9. Details of the “spikes” in the middle range channels.](image)

Figure 4.10 and Figure 4.11 show the bin contribution for a low and a high range channel respectively under this inconsistent assumption. Although the input values are within the ranges where the transformation expands and compresses the data, the bin contribution to the output histogram has been assumed to be the same.
4.1 Correcting the Mapping Process of the Transformed Data

To correctly map the input data from the A/D converter to the transformed domain, the A/D discretization process needs to be revisited. In mathematical terms, the discretization process of the A/D converter implements a floor function. This means that for the analog data range \([x_0, x_0 + 1)\), the output will be the integer digital value \(x_0\). Without loss of generality, we shall assume a one-to-one correspondence between the analog input metric
units and the digital output of the A/D converter. If the probability distribution of the input data in the range \([x_0, x_0 + 1]\) is known, then it is possible to determine the probability distribution in the transformed domain.

When the input data is transformed from the input domain, \(X\), to the output domain, \(Y\), using a given transformation \(Y = t(X)\), the input probability distribution, \(f_X(x)\), is transformed into \(f_Y(y)\). Because the transformation does not affect the probability of each event, the infinitesimal probability of any point \(x\) is the same at the output value \(y = t(x)\).

In mathematical terms, this can be expressed as follows:

\[
f_Y(y) \cdot dy = f_X(x) \cdot dx \tag{4.2}
\]

By simple substitution of \(x\) by \(t^{-1}(y)\), the relationship between the resulting probability distribution, \(f_Y(y)\), and the input probability distribution, \(f_X(x)\), will take the new form:

\[
f_Y(y) = f_X(t^{-1}(y)) \cdot \frac{d}{dy} t^{-1}(y) \tag{4.3}
\]

where \(X\) and \(Y\) are two random variables related by a transformation \(Y = t(X)\). The transformation \(t(x)\) is a monotone increasing function whose inverse \(t^{-1}(y)\) has a continuous derivative on \(Y\) [2]. Equation (4.2) also guarantees that the areas under the input and output probability distributions are maintained, which is an important requisite when analyzing cell populations.

For the particular case of the standard log-transformation presented earlier in Equation (4.1), the transformed density distribution is given by:
where $t(x)$ is the output transformed value of input $x$, and where $S$ and $K$ are as defined earlier while explaining Equation (4.1).

To use new log-transformation in Equation (4.4) we need to know the probability distribution of the input histogram, which is unknown but it can be approximated, as we will see later.

The binning effect can be thought as the result of assuming that the input probability distribution at each A/D interval are delta functions positioned at each A/D output value, which is the least probably case when dealing with biological events as in cell populations [7].

Figure 4.12 shows the lower 80 channels of the delta approximation for the input distribution of Figure 1.1.

Figure 4.13 shows the results of transforming the delta approximation of the input data using the standard log-transformation in Equation (4.1). Note how the log-transformed values are separated in the lower-range where the log-transformation function is expanding the dynamic range, and note how histogram values are overlapped in the mid and upper-range where the log-transformation function is compressing the dynamic range.

Figure 4.14 shows the result of accumulating the transformed data into the output histogram. Note that the result is identical to the binning effect shown in Figure 4.1.
Figure 4.12. Delta Approximation.

Figure 4.13. Transformed delta approximation.
Assuming a more realistic probability distribution at each A/D interval can make a better log-transformation. This assumption could be accomplished through: (1) uniform approximation, (2) linear approximation and any higher order approximation.

The following sections describe the uniform and linear approximation approaches to address the binning effect.

4.2 Uniform Approximation
The simplest approximation, and a far better than the original delta approximation, is the uniform approximation. In this case, the probability distribution of the input data in each A/D step is assumed uniform. The black curve in Figure 4.15 shows details of the uniform approximation of the input data distribution of Figure 1.1 (shown in gray).
Figure 4.15. Details of Uniform Approximation.

For this type of approximation, a uniform distribution at each A/D step is assumed, meaning that the analog data range \([x_0, x_0 + 1)\) will be assigned to the value \(x_0\). It can be observed that when the input data distribution is increasing, the uniform approximation will underestimate the real distribution. On the other hand, when the input data distribution is decreasing, the uniform approximation will over-estimate the real distribution.

When the input data of Figure 4.15 is log-transformed using the new transformation in Equation (4.4), the resulting histogram resembles more closely the ideal transformation as show in black in Figure 4.16. The ideal transformation is shown in gray.
Figure 4.16. Log Transformed Uniform Approximation.

Figure 4.17. Details of errors in Uniform Approximation.
In this case, artifacts of the binning effect are eliminated because the input domain is now assumed continuous. Even though the uniform approximation provides results similar to the ideal transformation, these results are not perfect. The imperfections are more visible in the lower range in the location of the “holes” of the binned histogram. These imperfections are the direct result of the difference between the actual input distribution and the uniform approximation made. Figure 4.17 shows in the black curve the details of these imperfections where the actual input distribution differs most from the uniform approximation. As in Figure 4.16, the ideal transformation is shown in gray.

Although the uniform approximation does not provide an ideal transformation, the area under the curve is maintained, and it will be shown in the next Chapter that the uniform approximation can be implemented in real-time by using lookup-up tables.

To improve the results obtained with the uniform approximation, it is necessary to employ a higher order approximation like the linear approximation, which is explained in the next section.

4.3 Linear Approximation

After the uniform approximation, the next increasingly complex approximation that can be used is the linear approximation. In this approximation, the probability distribution of the input data in each A/D step is assumed linear. The black curve in Figure 4.18 shows a detail of the linear approximation of the input data distribution of Figure 1.1 (shown in gray).
Figure 4.18. Detail of Linear Approximation.

The linear approximation resembles very well the original distribution so that it is expected that results will be close to the ideal.

Figure 4.19 shows the results of using the new log-transformation in Equation (4.4) under the assumption that the input data distribution is linear at each A/D interval. In this case, the results are practically identical to the ideal log-transformation. As in the uniform approximation, the area under the curve is also maintained.

Even though the results can be improved using higher approximations, it does not make practical sense to move forward with this process, since the natural variability of biological populations is likely to be far greater than the error of the approximation made.
Figure 4.19. Log Transformed Linear Approximation.

It is important to note at this juncture that the amplitudes of the output histogram are non-integer; therefore, it is impossible to create a sequence of data points that can generate the same histogram. This fact makes it impossible to directly enumerate cell populations based on the analysis results of the output histogram, but this can be overcome by doing an inverse transformation of the analysis results into the input domain and enumerating the cell populations using the input events.

For example, Figure 4.20 shows the valley obtained by processing the output histogram. This valley can be mapped to the input domain by using the inverse transformation of $t(x)$.

Figure 4.21 shows the valley of Figure 4.20 mapped back to the input histogram. Once this valley is in the input domain scale, it can be used to enumerate the cell populations on an event-by-event basis.
Figure 4.20. Valley in the output histogram.

Figure 4.21. Valley mapped to the input domain.
4.4 Summary

This Chapter provided a mathematical foundation based on uniform and linear approximation with the objective to solve for the binning effect. Examples were provided to demonstrate consequences of the ubiquitous binning effect that occurs when the traditional approach to a logarithmic transformation is taken.

The results of these new mapping techniques demonstrate that the outcome is a quasi-optimal approximation of the actual logarithmically transformed data. With this new approach to data mapping, the need for filtering is a non-issue since the binning effect is countered early in the process, therefore preserving the statistical properties of the populations under study.
CHAPTER 5

Non-integer Bin Contributions and Real-time implementation

Feasibility studies of the new histogram accumulation method introduced in this dissertation call for real-time implementations in flow cytometer instruments. In this context, the term real-time means the ability to process each data point as it is being acquired. This is equivalent to processing the whole data almost instantaneously after its acquisition, estimated in this case at less than 50 milliseconds (ms). The 50 ms measure is used in view of the fact that 50 ms is the accepted standard for instantaneous perception in user interaction [10, 11].

To reach real-time processing it is necessary to study the nature of the contribution to the output histogram for each acquired data point. The next section develops a novel technique to accumulate data points as they are being acquired in the resulting log-transformed histogram.

5.1 Non-integer Bin Contribution

To process input data points in real-time it is necessary to process them one-by-one in a sequential fashion. Our goal is to accumulate the histogram as the data points are being acquired. As already mentioned in the previous Chapter, it is impossible to generate a data sequence that can be accumulated to generate the desired log-transformed histogram, therefore it is necessary to develop a new histogram accumulation technique capable of constructing the desired histogram as the input data points are gathered in real time.
In the next paragraphs, we will study how each individual input data point contributes to the final log-transformed histogram, allowing us to visualize how it is possible to implement the histogram accumulation in real-time.

Using the new log-transformation in Equation (4.4) and assuming that the probability distribution for any given input data range \([x_0, x_0 + 1]\) is uniform, a non-integer bin contribution in the output histogram can be computed by applying Equation (4.4) to each input data point. These bin contributions are added together to form the output histogram.

In the assessments that follows, the non-integer bin contributions will be shown for both low and high range channels respectively, using the new mapping technique developed in Chapter 4.

For example, Figure 5.1 shows how the input range \([16, 17)\) maps to the output channels between 101 and 103.

In this lower range, the log-transformation expands the dynamic range of the input data.

The gray curve in Figure 5.2 represents the exact transformation using the new log-transformation in Equation (4.4), for the uniform unit distribution in the range \([16, 17)\). The area under the gray curve is equal to 1 as expected, since the probability area is maintained. Because the output histogram bins have integer boundaries, the area under the gray curve in each bin is computed and assigned to that bin. The resulting non-integer bins are shown in dark in Figure 5.2. The total area of the bin contributions is maintained to be equal to one.
Figure 5.1. Mapping for input data range [16, 17).

Figure 5.2. Bin contributions for a low range channel.
The next range [17, 18) will generate an overlapping contribution starting from channel 106, therefore “filling” the area not covered by the upper part of the [16, 17) interval. The “spreading” of one input data point into several non-integer bins and the overlapping contributions between contiguous input data points is what fills the discontinuities or “holes” and eliminates the “spikes” observed in the typical binned histogram as illustrated earlier in Figure 1.2.

On the other hand, Figure 5.3 shows how the input range [151, 152) maps to the output channel 184. In this mid-range, the log-transformation compresses the dynamic range of the input data.

![Figure 5.3. Mapping for input data range [151, 152).](image)
The gray curve in Figure 5.4 represents the exact transformation, using the new log-transformation in Equation (4.4), for the uniform unit distribution in the range \([151, 152)\). In this case, because the input range maps inside a single bin, this resulting bin in the output histogram will have a unit area regardless of the shape of the probability distribution in the \([151, 152)\) interval.

Figure 5.5 shows how the input range \([162, 163)\) maps to the output channel 187 and 189. The gray curve in Figure 5.6 represents the exact transformation, using the new log-transformation in Equation (4.4), for the uniform unit distribution in the range \([162, 163)\). In this case, even though the input range maps to a range smaller than a single bin, this resulting contribution falls in the boundary between bins 186 and 187, generating a non-integer bin contribution that spreads into two bins.
Figure 5.5. Mapping for input data range [161, 162).

Figure 5.6. Bin contribution for a high range channel that falls in a bin boundary.
These different types of non-integer bin contributions for the log-transformation are computed and accumulated for each point that needs to be mapped from the original domain to the transformed domain. Each set of bin contributions is different and is a function of the position/index of the histogram channel. The next section shows how to implement this new histogram transformation technique using lookup tables.

5.2 Use of Lookup Tables for Real-Time Application

A fast way to accumulate the histogram is to have pre-computed in-memory all the needed non-integer bin contributions indexed by the input channel. This pre-computed values could be used as a lookup table allowing fast access to the information. The non-integer contributions will have to be described in such a way that it is efficient to retrieve them from memory and add them to the output histogram.

One way to describe a non-integer contribution is with the Initial Bin, the Final Bin, and the Area of each non-integer bin contribution. Given the previous structure, the following pseudo-code shows the procedure to accumulate each input data point as is being acquired:

```plaintext
for each input data point
{
    retrieve the non-integer contribution from the lookup table
    for each area in the non-integer contribution
    {
        add the area at the appropriate bin in the output histogram
    }
}
```

To generate the lookup table in memory we need to compute each one of the non-integer contribution parameters described in the previous paragraph. The initial bin, $Bin_{ini}$, is the first integer bin position that is smaller or equal than the transformed input value $x$.
\[ Bin_{in}(x) = \text{floor}(t(x)) \]  

(5.1)

The final bin, \( Bin_{end} \), is the first integer bin position that is smaller or equal than the transformed input value \( x + 1 \):

\[ Bin_{end}(x) = \text{floor}(t(x + 1)) \]  

(5.2)

The center bins are all the bins between the initial and final bins.

---

**Figure 5.7. Non-integer bin contribution example.**

The last step is to compute the areas associated to each bin. Figure 5.7 shows again an example of how a unit input at channel 16 is mapped to its non-integer contribution.

To compute each non-integer contribution area we need to integrate Equation (4.3) inside each integer bin range. Given a function \( A(y) \) that defines the integral from 0 to \( y \), the area
inside each bin can be computed as the difference in $A(y)$ evaluated at the boundaries inside each bin.

Equation (5.3) shows the mathematical expression of $A(y)$ in terms of the input probability distribution, $f_X(x)$, and the inverse transformation function, $t^{-1}(y)$:

$$A(y) = \int_0^y f_X(t^{-1}(y)) \cdot \frac{d}{dy} t^{-1}(y)$$  \hspace{1cm} (5.3)

Because $f_X(x) = 1$ when a uniform distribution is assumed, Equation (5.3) can be simplified to yield the expressions provided by Equations (5.4) and (5.5):

$$A(y) = \int_0^y \frac{d}{dy} t^{-1}(y)$$  \hspace{1cm} (5.4)

$$A(y) = t^{-1}(y)$$  \hspace{1cm} (5.5)

The calculation of the bin area can be divided in three cases:

1) The initial area, $A_{ini}$.

2) The center bins area, $A_{center}$, and

3) The final area, $A_{end}$.

The initial area, $A_{ini}$, is the area between the end of the initial bin, $Bin_{ini} + 1$, and the transformed data point $x$, as described in Equation (5.6):

$$A_{ini} = A(Bin_{ini} + 1) - A(t(x)) = A(Bin_{ini} + 1) - x$$  \hspace{1cm} (5.6)
The center bins area, $A_{\text{center}}$, is the area between the end of each center bin, $Bin_{\text{center}} + 1$, and the beginning of that bin, $Bin_{\text{center}}$, as given in Equation (5.7):

$$A_{\text{center}} = A(Bin_{\text{center}} + 1) - A(Bin_{\text{center}})$$

(5.7)

The end bin area, $A_{\text{end}}$, is the area between the transformed data point $x + 1$ and the beginning of the final bin, $Bin_{\text{end}}$, as given in Equation (5.8):

$$A_{\text{end}} = A(t(x + 1)) - A(Bin_{\text{end}}) = (x + 1) - A(Bin_{\text{end}})$$

(5.8)

There is a special case when $Bin_{\text{end}}$ is equal to $Bin_{\text{ini}}$, and therefore there is no center bins. This case represents the high range transformation, and the area to be accumulated is set to one (refer back to Figure 5.3.)

Using the previous Equations (5.1), (5.2), (5.6), (5.7), and (5.8), it is possible to generate the lookup table with the non-integers bins contributions and use it to accumulate the output histogram in real-time, i.e., as each input data point is acquired.

### 5.3 Results

The black curve in Figure 5.8 is the resulting histogram using this new technique.

The resulting histogram has the same area as that of the initial distribution, confirming that the number of events is maintained. Furthermore, the resulting histogram is very close to the ideal transformation shown with the gray curve.
Figure 5.8. Results of the new histogram accumulation method.

The use of pre-computed lookup tables makes the computation time negligible. The implementation using C++ programming language took few milliseconds to process 8,000 data points in a 900 MHz PC, therefore allowing the use of this new technique in real-time applications.

Note that the results shown in Figure 5.8 shows some small differences with the results obtained earlier in Figure 4.16. These differences are not because the technique introduces errors, but because the input distribution needs to be quantized to integer values and therefore generates rounding errors.

5.4 Summary
A novel method for accumulating the log-transformed data was developed. This new method uses the properties of the transformation of statistical distributions to accumulate the
output histogram in a non-integer and multi-channel fashion. That is, per each input data point, several histogram bins are accumulated in non-integer amounts instead of the traditional method of accumulating one unit in only one histogram bin.

This new technique will have utility in replacing electronic log-amplifiers and offer simpler electronics that are inherently more stable and more reliable. Furthermore, although the computation of this new mapping technique is involved, it does lend itself to a real-time implementation using lookup tables.
CHAPTER 6

Real Data Implementations

This Chapter shows the application of the newly developed technique in real flow cytometer data. Two implementation cases are studied: ten bits and twenty bits resolution data.

6.1 Ten Bits Resolution Data

Ten bits resolution, which is equivalent to 1024 channels, is a common resolution of ADCs used in past and current technology. Figure 6.1 shows a sample of 10 bits flow cytometry data:

![Real 10 bit flow cytomer data](image)

Figure 6.1. Real 10 bit flow cytomer data.
Figure 6.2. Standard log-transformation.

Figure 6.3. Results of the new technique.
Figure 6.2 shows the result of applying the log-transformation given by the standard log-transformation in Equation (4.1) to the input data shown in Figure 6.1. As expected, artifacts of the binning effect distort the output result making it inappropriate for accurate statistical analysis.

Figure 6.3 shows the results of applying the new technique on the same input data shown in Figure 6.1. The artifacts disappeared generating a smooth approximation with results that are most appropriate for enhanced statistical analysis.

6.2 Twenty Bits Resolution Data
Twenty bits resolution data is considered high-resolution data in the current technology. New instruments are using 20-bits resolution ADCs as a mean of avoiding the binning effect. It will be shown that the use of high-resolution data diminishes the binning effect but does not eliminate it altogether.

Figure 6.4 shows the realization of the standard log transformation in Equation (4.1) for an input of 20-bit resolution, four decades, and an output histogram of 1024 channels.

Figure 6.5 shows the histogram of real 20-bit flow cytometry data.

Figure 6.6 shows the results of applying the standard log transformation of Equation (4.1) to the real 20-bit flow cytometry data shown in Figure 6.5.
Figure 6.4. Four decades transformation.

Figure 6.5. Real 20-bit flow cytometry data.
Figure 6.6. Transformed data using the standard log-transformation.

Note how the “holes” artifacts from the binning effect are not present. The elimination of the “holes” in the high-resolution data is what led people in the field to believe that the artifacts of the binning effect disappeared, but the reality is that the “spikes” artifacts can be seen especially in the detailed data given in Figure 6.8(a) and Figure 6.9(a).

Figure 6.7(b) shows the results of using the new technique to compute the output histogram. The “spikes” artifacts of the binning effect are eliminated leaving only the original irregularities of the data. It is interesting to note that when dealing with real data the artifacts of the binning effect can be obscured by the natural data irregularities.

Figure 6.8 and Figure 6.9 show the details of the first and second decade for the standard log transformation and the new technique in the 20-bit real flow cytometry data. These Figures show the hidden “spikes” in the standard log-transformed data.
a.) Standard transformation.  

b.) Proposed method.

Figure 6.7. Whole range histogram (4 decades).

a.) Standard transformation  

b.) Proposed method

Figure 6.8. First decade detail.

a.) Standard transformation  

b.) Proposed method

Figure 6.9. Second decade detail.
6.3 Summary

This Chapter showed the merit of the new technique applied to real flow cytometry data. It was shown that the new approach eliminates the artifacts of the binning effect in low as well as in high-resolution data. It is noted that when seeking higher resolution A/D converters, the binning effect was alleviated but it is still omnipresent. This situation highlights the importance of the method introduced in this dissertation in preserving the statistical properties of the original data.
CHAPTER 7

Mathematical Framework and Implementation of the Multi-Dimensional Case

7.1 Introduction

The new technique developed can be extended to any numbers of dimensions. This Chapter shows the 2-D case, which can be easily extended to more dimensions.

Figure 7.1 shows a 3-D view of the lower-left quadrant of a simulated 2-D distribution:

![3-D view of a simulated 2-D distribution.](image)

Figure 7.1. 3-D view of a simulated 2-D distribution.

This simulated distribution shows two overlapped Gaussian distributions in the low-range of the histograms. Figure 7.2 shows the top view of this distribution.
7.2 Implementation

As in the 1-D case, the input data is transformed with a logarithmic transformation in each dimension. Due to memory constrains, the number of output histograms channels is less than the number used in the 1-D case. Figure 7.3 shows the results of the standard log-transformation as described in Equation (4.1) for an input with a 10-bit resolution, with three decades, and with an output histogram of 64 channels per dimension.

Figure 7.4 shows the 3-D view of the ideal log transformation of the simulated distribution of Figure 7.1 using the log transformation shown in Figure 7.3:
Figure 7.3. Log transformation for 2-D distribution.

Figure 7.4. 3-D view of the ideal transformation.
Figure 7.5. Top view of the ideal transformation.

Figure 7.5 shows the top view of the ideal log transformation of the simulated distribution of Figure 7.1 using the log transformation shown in Figure 7.3.

The ideal transformation can be computed only by applying the log transformation to the continuous input data. When the digitized input data is log transformed, the binning effect is observed in two dimensions. Figure 7.6 shows the binning effect when the log-transformation of Figure 7.3 is applied to the digitized input data of Figure 7.1.

The white spaces in the lower-left quadrant are the equivalent of the “holes” of the 1-D case illustrated earlier in Figure 4.4, and the black dots in the lower-left quadrant are the equivalent of the “spikes” of the 1-D case illustrated earlier in both Figure 4.6 and Figure 4.8. The cause for the binning effect in two dimensions is the same as in the 1-D case: the non-uniform mapping of the input range into the output range.
7.3 Mathematical Framework and Implications

When the 2-D input data is transformed from the input domain, \((X, Y)\), to the output domain, \((V, W)\), using a given transformation \(V = t(X)\) and \(W = t(Y)\), the input probability distribution, \(f_{x,y}(x, y)\), gets transformed into \(f_{v,w}(v, w)\). Because the transformation does not affects the probability of each event, the infinitesimal probability of any point \((x, y)\) is the same at the output value \(v = t(x)\) and \(w = t(y)\).
In mathematical terms, this outcome can be expressed as follows:

\[
  f_{V,W}(v, w) \cdot dv \cdot dw = f_{X,Y}(x, y) \cdot dx \cdot dy \tag{7.1}
\]

By substituting \((x, y)\) with \((t^{-1}(v), t^{-1}(w))\), the relationship between the resulting probability distribution, \(f_{V,W}(v, w)\), and the input probability distribution, \(f_{X,Y}(x, y)\), is given by:

\[
  f_{V,W}(v, w) = f_{X,Y}(t^{-1}(v), t^{-1}(w)) \cdot \frac{d}{dv} t^{-1}(v) \cdot \frac{d}{dw} t^{-1}(w) \tag{7.2}
\]

where \(X\) and \(Y\) are two random variables related by a transformation \(V = t(X), W = t(Y)\), and \(t(x)\) is a monotone increasing transformation function whose inverse \(t^{-1}(y)\) has a continuous derivative in \(Y\) [2]. As in the 1-D case, Equation (7.2) guarantees that the areas under the input and output probability distributions are maintained.

To make proper use of Equation (7.2) it is necessary to make assumptions about the probability distribution. The simplest approximation, as indicated earlier, is the uniform approximation, where the probability distribution in each A/D step is assumed uniform. Figure 7.7 shows the detail of the uniform approximation of the original 2-D distribution shown in Figure 7.1.
Figure 7.7. Detail of the uniform approximation of the data.

Figure 7.8 and Figure 7.9 shows the results of using Equation (7.2) in the uniform approximation of Figure 7.7.

Note how artifacts of the binning effect of Figure 7.6 are eliminated. The smaller remaining artifacts are the direct result of the difference between the actual input distribution and the uniform approximation made.

As in the 1-D case, better results can be obtained by using higher order approximations. Even though higher order approximations produce near ideal results, the results obtained using the uniform approximation are good enough for all practical purposes.
Figure 7.8. 3-D view of the results when using the uniform approximation.

Figure 7.9. Top view of the results of Figure 7.8.
7.4 Mathematical Extension to N-Dimensional

The 2-D mathematics derived can be easily extended to multidimensional \((N > 2)\) space.

What follows are extensions by simple deduction of Equation (7.2):

\[
\begin{align*}
    f_{i[\text{var}_i, t[\text{var}_i], ..., t[\text{var}_{i+k}]]}(t[\text{var}_1], t[\text{var}_{i+1}], ..., t[\text{var}_{i+k}]) \\
    = f_{\text{VAR}_i, \text{VAR}_i+1, ..., \text{VAR}_i+k}(t^{-1}[\text{var}_1], t^{-1}[\text{var}_{i+1}], ..., t^{-1}[\text{var}_{i+k}]) \cdot \prod_{k=0}^{N-1} d \frac{d}{d \text{var}_{i+k}} \cdot t^{-1}[\text{var}_{i+k}]
\end{align*}
\]  

Note that when \(k = 0\), the 1-D case is generated as:

\[
f_{i[\text{VAR}_i]}(t[\text{var}_i]) = f_{\text{VAR}_i}(t^{-1}[\text{var}_i]) \cdot \frac{d}{d \text{var}_{i}} \cdot t^{-1}[\text{var}_i]  
\]  

with \(i\) used to simply identify which first variable of the alphabet is used, e.g., for \(i = 0\), \(\text{var}_i = x\), \(\text{VAR}_i = X\), \(t[\text{var}_i] = \nu\), and \(t[\text{VAR}_i] = \nu\).

In the 3-D case \((N = 3)\) as another example, the general Equation (7.3) yields for a given \(i\) the following transform equation:

\[
\begin{align*}
    f_{i[\text{VAR}_i, \text{VAR}_i+1, \text{VAR}_i+2]}(t[\text{var}_1], t[\text{var}_{i+1}], t[\text{var}_{i+2}]) \\
    = f_{\text{VAR}_i, \text{VAR}_i+1, \text{VAR}_i+2}(t^{-1}[\text{var}_1], t^{-1}[\text{var}_{i+1}], t^{-1}[\text{var}_{i+2}]) \cdot \prod_{k=0}^{2} d \frac{d}{d \text{var}_{i+k}} \cdot t^{-1}[\text{var}_{i+k}]
\end{align*}
\]

choosing \(\text{var}_0 = x\), for \(i = 0\) then \(\text{var}_{i+k}(k = 0, 1, 2) = x, y, z\), \(\text{VAR}_{i+k}(k = 0, 1, 2) = X, Y, Z\), \(t(\text{var}_{i+k})(k = 0, 1, 2) = u, v, w\), and \(t(\text{VAR}_{i+k})(k = 0, 1, 2) = U, V, W\). Equation (7.5) will thus yield for the 3-D case the expression:

\[
f_{U, V, W}(u, v, w) \\
    = f_{X, Y, Z}(t^{-1}(u), t^{-1}(v), t^{-1}(w)) \cdot \frac{d}{du} t^{-1}(u) \cdot \frac{d}{dv} t^{-1}(v) \cdot \frac{d}{dw} t^{-1}(w)
\]
Equation (7.6) is descriptive of the 3-D case. Other higher dimensions can be derived in the same manner using the general Equation (7.3).

7.5 Summary

This Chapter introduced the mathematics and the practical aspects of the binning effect problem focusing in the two-dimensional case. It was shown how the 2-D approach to solving the binning effect could be extended to higher dimensions with rather simple mathematical extensions.
CHAPTER 8

Conclusion

The mathematical framework and the supporting method for real-time implementation presented in this dissertation relate in general to statistical analysis of data sample distributions, such as a histogram of photoluminescence data produced by a blood flow cytometer system, and are particularly directed to the realistic construction of dynamically-extended accumulated data histograms. The focal operative issue is to preserve statistical probability distribution characteristics for a quantized data set that is subjected to a dynamic range expansion transform, such as a logarithmic operator, which, without the statistical probability preservation mechanism, would introduce undesirable binning artifacts into the transformed data.

The importance of the research presented in this study is based upon the practical implications related to flow cytometry. Flow cytometry derives from the quantitative measurement (meter) of structural features of biological cells (cyto) transported by a carrier in a controlled flow through a series of primarily optical detectors. Flow cytometers have been commercially available since the early 1970's, and their use has been continuously increasing. The most numerous flow cytometers are those employed for complete blood cell counts in clinical laboratories. Flow cytometers are found in all major biological research institutions. They are also numerous in medical centers, where they are used for diagnosis as well as research. There are currently on the order of 7,000 flow cytometers in use worldwide. Among its critical applications, chromosome count and cell cycle analysis of cancers remains one of the major diagnostic use of flow cytometers. Lymphomas and
leukemia are intensively studied for surface markers of diagnostic and prognostic value. Furthermore, flow cytometry has been the method of choice for monitoring AIDS patients. It is for such critical applications with serious real-world implications and societal impact, that the contributions of this dissertation are most eminent.

In the current approaches for analyzing flow cytometric histograms, a pre-processing step known as the log-transformation, is employed to increase the dynamic range of the data distribution, in order to facilitate analysis and henceforth enhance interpretation. The objective of such an important task is to separate and quantify cell populations accurately. Typically, the acquired data is accumulated into one or two dimensional data distributions so that the morphological variability of the distributions can be interpreted to distinguish the cellular populations. Unfortunately, although this step serves to broaden the dynamic range of the data distributions, it introduces undesirable artifacts associated with a phenomenon known as the “picket fence” or “binning effect”, that undermines the one aspect of the solution it is intended to reinforce.

In an effort to counter the binning effect’s undesired artifacts, which skew the analysis/interpretation of the results, a number of practitioners have relied upon filtering techniques, that attempt to attenuate this undesired effect without introducing changes that might undermine the statistical values of the original data. However, these filtering approaches are fraught with an irreconcilable issue: that of relating the degree to which the artifact must be filtered with the point at which the filtered data is still considered to retain statistics similar to those of the original data.
More particularly, to resolve the binning effect issue, averaging schemes based on finite impulse response (FIR) filters are typically used. Most effective FIR filtering schemes are achieved using a traditional (1-2-1) 3-tap FIR filter. Such a filter has positive attributes, provided that the following constraints are met: (1) maintaining the area under the curve requires that the sum of the FIR filter coefficients equals 1, and that the appropriate boundary conditions have been met; (2) to prevent the data from being skewed, the filter coefficients must be symmetric around the center (this requires that the number of filter taps in the FIR filter be odd), and the individual distributions must be symmetrical and not overlapping, which is not typically true when analyzing log-transformed data from cell populations.

Other practitioners have attempted to ameliorate the binning effect by making use of log-amplifiers to electronically transform the input signal in its analog form before digitalization. This approach which has its own merit inherits however the following drawbacks:

- It requires the use of additional and expensive hardware;
- Logarithmic amplifiers are notoriously noisy and unstable; and
- When linear data is required, the instruments must send and store twice the amount of data, which could be overwhelming.

Another approach used to overcome the binning effect is to use high-resolution ADCs in an effort to prevent the log transformation from exhibiting discontinuities in the lower range histogram channels. It is revealed that this approach is not yet perfected since it contends with the following problems:
• High-resolution ADCs converters are expensive;

• The amount of data the instruments must send and store will increase proportionally with the increase in bit resolution of the ADC - another expensive proposition; and

• No matter what the resolution of the ADC converter used, the binning effect, although minimized, will still be present in the output histogram. Of course, there is a point where the natural irregularities of the real data will be greater than the binning effect.

In accordance with the current technological limitations, the premise of this dissertation was to overcome such limitations by tackling the problem at the source of its origin. As a consequence of the work introduced in this dissertation, the above drawbacks of conventional signal processing mechanisms used to process histograms of data, especially data associated with biological cell populations, including human blood cells that have been subjected to flow cytometry processing, are effectively obviated by a new and improved, statistical probability distribution-preserving histogram transform mechanism. This inventive mechanism effectively preserves the statistical probability distribution characteristics of the original data that would otherwise be removed or lost in the course of expanding the dynamic range of a quantized histogram data set through the use of a conventional log transformation. With this new approach to data mapping, the need for filtering becomes a non-issue, since the binning effect is countered early in the process, thus preserving the statistical properties of populations under study.
The implications of the contributions of this research take on different directions:

(1) On the mathematical front, this dissertation introduces for the first time mathematical derivations that are genuine not only in the solution they yield, but also in the concise manner in which they inherently characterize the nature of the binning effect.

(2) On the analysis front it was determined that the binning effect results from the erroneous premise that an input data range \([x_0, x_0 + 1)\) uniformly maps or is transformed to the output data range \([t(x_0), t(x_0) + 1)\), where \(t(x)\) is the transformation function. In actuality, however, when mapping digitized input data to the transformed domain, the quantization or discretization process carried out by an analog-to-digital converter implements a floor function, so that for the analog data range \([x_0, x_0 + 1)\), the output will be the integer digital value \(x_0\). In addition, when using A/D converters, it was determined that the assumption that the binning effect may be considered to be the result of assuming that the input probability distribution at each ADC interval is a delta function positioned at each ADC output value is an erroneous assumption. In actuality, this assumption is the least probable case when dealing with biological events, such as biological cell populations. To improve the accuracy of the log-transformation, a more realistic probability distribution at each ADC interval, such as uniform approximation, linear approximation, and any higher order approximation, should be used instead.

(3) On the implementation front, a real-time procedure was developed based on this new mathematical framework by using look tables so that well-accumulated histograms are dynamically extended as the statistical properties of the data are preserved.
REFERENCES


APPENDIX

This appendix lists some of the MatLab scripts written in support of the mathematical framework developed in this dissertation. The scripts provided implement the following tasks:

- The standard log-transformation.
- The ideal 1-D case.
- The binning effect in 1-D.
- The FIR approach.
- The 1-D delta approximation.
- The 1-D uniform approximation.
- The 1-D linear approximation.
- The 1-D solution.
- The ideal 2-D case.
- The binning effect in 2-D.
- The 2-D uniform approximation.
% LogTransformation Script

Decades = 3;
InputRange = 2^BitResolution;
a = (Bins - 1)/ Decades;
b = 10^Decades / InputRange;
syms x;
Transform = a * log10(x * b);
InvTransform = (10^(x / a)) / b;

% Transformation curve plot
xAxis = logspace(0, 3, 200);
yAxis = round(subs(Transform, xAxis));
figure
line(xAxis, yAxis, 'color', 'k', 'LineWidth', 1);
set(gca, 'XScale', 'log');
axis([1, InputRange - 1, 0, Bins - 1]);
set(gcf, 'Name', 'Transformation curve.');
xlabel('x');
ylabel('t(x)');
grid on;
box on;

% Transformation details
xAxis = 1:InputRange - 1;
loLimits = xAxis;
hiLimits = xAxis + 1;
xExtended = zeros(2, length(xAxis));
yExtended = zeros(2, length(xAxis));

xExtended(1, :) = loLimits;
xExtended(2, :) = hiLimits;
yAxis = round(subs(Transform, xAxis));
yExtended(1, :) = yAxis;
yExtended(2, :) = yAxis;
figure
line(xExtended, yExtended, 'color', 'k', 'LineWidth', 1);
axis([0, InputRange - 1, 0, Bins - 1]);
set(gcf, 'Name', 'Transformation curve.');
xlabel('x');
ylabel('t(x)');
grid on;
figure
line(xExtended, yExtended, 'color', 'k', 'LineWidth', 2);
axis([0 10 0 100]);
set(gcf, 'Name', 'Detail of low range expansion');
xlabel('x');
ylabel('t(x)');
grid on;

figure
line(xExtended, yExtended, 'color', 'k', 'LineWidth', 2);
XRange = [42:54];
axis([XRange(1) XRange(end) 138 146]);
set(gca, 'XTick', XRange);
set(gcf, 'Name', 'Detail of middle range expansion');
xlabel('x');
ylabel('t(x)');
grid on;

figure
line(xExtended, yExtended, 'color', 'k', 'LineWidth', 2);
axis([InputRange - 100, InputRange - 1, Bins - 10, Bins]);
set(gcf, 'Name', 'Detail of high range compression');
xlabel('x');
ylabel('t(x)');
grid on;

**IdealCase Script**

```matlab
%***************************************************************************
% Author: Patricio J. Vidal.
% Ph.D. Dissertation.
%***************************************************************************
if ~-exist('InputRange', 'var'), error('A InputRange variable is not defined.'); end
if ~-exist('InvTransform', 'var'), error('A InvTransform function is not defined.'); end
if ~-exist('Transform', 'var'), error('A Transform function is not defined.'); end
if ~-exist('LineOpt', 'var'), error('A LineOpt variable is not defined.'); end

Gaussian1 = Gaussian1D(100, 5, 5);
Gaussian2 = Gaussian1D(25, 30, 20);
OriginalDistribution = Gaussian1(x) + Gaussian2(x);

figure
xAxis = 0:0.1:InputRange - 1;
plot(xAxis, subs(OriginalDistribution, xAxis), LineOpt{:});
set(gcf, 'Name', 'Original distribution.');
set(gca, 'YLimMode', 'auto');
xlabel('Value');
ylabel('Frequency')
TightXAxis;
```
IdealTransformedDistribution = subs(OriginalDistribution, InvTransform) * diff(InvTransform);

figure
yAxis = 0:Bins;
plot(yAxis, subs(IdealTransformedDistribution, yAxis), LineOpt{:});
set(gcf, 'Name', 'Ideal Transformed distribution.');
set(gca, 'YLimMode', 'auto');
xlabel('Value');
ylabel('Frequency')
TightXAxis;

BinningProblem Script

%*****************************************************************************
% Author: Patricio J. Vidal.
% Ph.D. Dissertation.
%*****************************************************************************

if ~exist('Transform', 'var'), error('A Transform function is not defined.'); end
if ~exist('InvTransform', 'var'), error('A InvTransform function is not defined.'); end
if ~exist('OriginalDistribution', 'var'), error('A OriginalDistribution variable is not defined.'); end
if ~exist('LineOpt', 'var'), error('A LineOpt variable is not defined.'); end

% Offset to avoid negative values after the transformation.
offset = 2;
histogramData = ArrayID(round(subs(OriginalDistribution, offset:InputRange - 1)));

rawData = ArrayID(zeros(1, sum(histogramData(:))));
index = 0;
for i = offset:InputRange - 1
    for y = 1:histogramData(i - offset)
        rawData(index) = i;
        index = index + 1;
    end
end

histogram = Histogram1D('Original Data', InputRange, 'Channel');
histogram(:) = subs(OriginalDistribution, histogram.Bins(:));
figure, plot(histogram, 'line', LineOpt{:});
transformData = subs(Transform, rawData(:));
TransformedHistogram = Histogram1D('Transformed Data', Bins, 'Channel');
TransformedHistogram = Accumulate(TransformedHistogram, transformData);
transformedDistribution = subs(OriginalDistribution, InvTransform) * diff(InvTransform);

figure;
plot(TransformedHistogram, 'line', LineOpt{:});
hold on;
xAxis = TransformedHistogram.Bins(:);
plot( xAxis, subs(transformedDistribution, xAxis), 'color', .7 * [1 1 1], 'LineWidth', 2);
hold off;

SmoothingApproach Script

%******************************************************************************
% Author: Patricio J. Vidal.
% Ph.D. Dissertation.
%******************************************************************************

if ~exist('NumberOfPasses', 'var'), error('A NumberOfPasses variable is not defined.'); end
if ~exist('TransformedHistogram', 'var'), error('A TransformedHistogram variable is not defined.'); end
if ~exist('IdealTransformedDistribution', 'var'), error('A IdealTransformedDistribution function is not defined.'); end
if ~exist('LineOpt', 'var'), error('A LineOpt variable is not defined.'); end

% Let's smooth several times...
smoothedHistogram = Smooth(TransformedHistogram, NumberOfPasses);
figure;
hold on;

xAxis = TransformedHistogram.Bins(:);
plot(xAxis, subs(IdealTransformedDistribution, xAxis),'color', .7 * [1 1 1], 'LineWidth', 2);
plot(smoothedHistogram, 'line', LineOpt{:});
box on;
axis([0, 255, 0, 25]);
set(gcf, 'Name', sprintf('%d passes smoothed histogram.', NumberOfPasses));

DeltaApproximation Script

%******************************************************************************
% Author: Patricio J. Vidal.
% Ph.D. Dissertation.
%******************************************************************************

if ~exist('Transform', 'var'), error('A Transform function is not defined.'); end
if ~exist('InvTransform', 'var'), error('A InvTransform function is not defined.'); end
if ~exist('OriginalDistribution', 'var'), error('A OriginalDistribution variable is not defined.'); end
if ~exist('LineOpt', 'var'), error('A LineOpt variable is not defined.'); end

xAxis = zeros(InputRange * 3, 1);

% Delta
% Create Approximation
for x = 0:InputRange - 1
    idx = 3 * x + 1;
    xAxis(idx:idx + 2) = x;
end

deltaApproximation = zeros(size(xAxis));
original = subs(OriginalDistribution, 0:InputRange - 1);
for x = 0:InputRange - 1
    idx = 3 * x + 1;
    deltaApproximation(idx + 1) = original(x + 1);
end

figure
plot(xAxis, deltaApproximation, LineOpt{:});
set(gcf, 'Name', 'Delta Approximation distribution.');
xlabel('Value');
ylabel('Frequency');
set(gca, 'YLimMode', 'auto');
axis([0 80 0 120]);
drawnow;

% Transform
figure
yAxis = subs(Transform, xAxis);
plot(yAxis, deltaApproximation, LineOpt{:});
xlabel('Value');
ylabel('Frequency');
set(gcf, 'Name', 'Delta Transform.');
axis([0 Bins 0 120]);

figure
plot(yAxis, deltaApproximation, LineOpt{:});
set(gcf, 'Name', 'Low Range Channels of Delta Approximation.');
xlabel('Channel');
ylabel('Frequency');
xRange = 0:100;
axis([0 Bins 0 120]);

figure
plot(yAxis, deltaApproximation, LineOpt{:});
set(gcf, 'Name', 'Delta Transform.');
xlabel('Channel');
ylabel('Frequency');
xRange = 140:148;
axis([xRange(1) xRange(end) 0 40]);
set(gca, 'XTick', xRange);

figure;
plot(TransformedHistogram, 'line', 'k', 'LineWidth', 2);
set(gcf, 'Name', 'Details of "spikes"');
xlabel('Channel');
ylabel('Frequency');
axis([xRange(1), xRange(end), 0, 40]);

figure;
plot(TransformedHistogram,'line', 'k', 'LineWidth', 2);

---

**UniformApproximation Script**

```matlab
%**********************************************************************
% Author: Patricio J. Vidal.
% Ph.D. Dissertation.
%**********************************************************************

xAxis = 0:0.1:InputRange - 1;
yAxis = subs(Transform, xAxis);

% Uniform
%
% Create Approximation
uniformApproximation = subs(OriginalDistribution, floor(xAxis));

figure
plot( xAxis, subs(OriginalDistribution, xAxis), 'color', .7 *[1 1 1],
     'LineWidth', 2);
hold on
plot(xAxis, uniformApproximation, LineOpt{:});
set(gcf, 'Name', 'Uniform Approximation distribution.');
xlabel('Value');
ylabel('Frequency');
set(gca, 'YLimMode','auto');
axis([0 12 90 116]);

% Transform
diffTransform = subs(diff(InvTransform), subs(Transform, xAxis));
linearTransformedDistribution = uniformApproximation .* diffTransform;

figure
plot( xAxis, subs(IdealTransformedDistribution, xAxis), 'color', .7 *[1 1 1], 'LineWidth', 2);
hold on
plot(yAxis, linearTransformedDistribution, LineOpt{:});
set(gcf, 'Name', 'Uniform Transform.');
xlabel('Value');
ylabel('Frequency');
```

80
axis([0 256 0 30]);

% Details of Transform
diffTransform = subs(diff(InvTransform), subs(Transform, xAxis));
linearTransformedDistribution = uniformApproximation .* diffTransform;

figure
plot( xAxis, subs(IdealTransformedDistribution, xAxis), 'color', .7 * [1 1 1], 'LineWidth', 2);
hold on
plot(yAxis, linearTransformedDistribution, LineOpt{:});
set(gca, 'Name', 'Details Uniform Transform.');
xlabel('Value');
ylabel('Frequency');
axis([50 120 14 28]);

LinearApproximation Script

%******************************************************************************
% Author: Patricio J. Vidal.
% Ph.D. Dissertation.
%******************************************************************************

xAxis = 0:0.1:InputRange - 1;
yAxis = subs(Transform, xAxis);

% % Linear %

% % Create Approximation
linearApproximation = zeros(size(xAxis));
nNumberOfPoints = length(xAxis);
maxXAxis = max(xAxis);
idxScale = numberOfPoints / maxXAxis;
G = subs(OriginalDistribution, 1:maxXAxis - 1) - subs(OriginalDistribution, 0:maxXAxis - 2);
Yi = subs(OriginalDistribution, 0:maxXAxis - 2);
for x = 0:maxXAxis - 2
    g = G(x+1);
    yi = Yi(x+1);
    for xx = x:0.1:x + 1
        linearApproximation(round(xx * idxScale) + 1) = yi + g * (xx - x);
    end
end

figure
plot( xAxis, subs(OriginalDistribution, xAxis), 'color', .7 * [1 1 1], 'LineWidth', 2);
hold on
plot(xAxis, linearApproximation, LineOpt{:});
set(gca, 'Name', 'Linear Approximation distribution.');
xlabel('Value');
ylabel('Frequency');
set(gca, 'YLimMode', 'auto');
axis([0 12 90 116]);
drawnow;

% Transform
diffTransform = subs(diff(InvTransform), subs(Transform, xAxi));
linearTransformedDistribution = linearApproximation .* diffTransform;

figure
plot(yAxis, linearTransformedDistribution, LineOpt{:});
set(gcf, 'Name', 'Linear Transform.');
xlabel('Value');
ylabel('Frequency');
axis([0 Bins 0 25]);

BinningSolution Script

%***************************************************************************
% Author: Patricio Vidal.
% Ph.D. Dissertation.
%***************************************************************************

if ~exist('Transform', 'var'), error('A Transform function is not defined.'); end
if ~exist('InvTransform', 'var'), error('A InvTransform function is not defined.'); end
if ~exist('OriginalDistribution', 'var'), error('A OriginalDistribution variable is not defined.'); end
if ~exist('LineOpt', 'var'), error('A LineOpt variable is not defined.'); end

% Offset to avoid negative values after the transformation.
offset = 1;
histogramData = ArrayID(round(subs(OriginalDistribution, offset:InputRange - 1)));

rawData = ArrayID(zeros(1, sum(histogramData(:))));
index = 0;
for i = offset:InputRange - 1
    for y = 1:histogramData(i - offset)
        rawData(index) = i;
        index = index + 1;
    end
end

histogram = HistogramID('Original Data', InputRange, 'Channel');
histogram = Accumulate(histogram, rawData(:));

figure
plot(histogram, 'line', LineOpt{:});
drawnow;

LoLimits = histogram.LoLimits;
HiLimits = histogram.HiLimits;

% Build vectors
InitialBins = Array1D(floor(subs(Transform, 0:InputRange)));
LoBound = Array1D(subs(Transform, 0:InputRange));
HiBound = Array1D(subs(Transform, 1:InputRange));

BinIntegral = Array1D(subs(InvTransform, 0:Bins));
LoBoundIntegral = Array1D(subs(InvTransform, LoBound(:)));
HiBoundIntegral = Array1D(subs(InvTransform, HiBound(:)));

UpperIntegral = Array1D(subs(InvTransform, 1:Bins));
LowerIntegral = Array1D(subs(InvTransform, 0:Bins - 1));
Areas = Array1D(UpperIntegral(:) - LowerIntegral(:));

% Accumulate in new histogram
NewHistogram = Histogram1D('Optimal', Bins,'');

for i = 0:length(rawData)-1
    Value = rawData(i);
    LoBin = InitialBins(Value);
    HiBin = ceil(HiBound(Value)) - 1;

    if LoBin == HiBin, % The area falls in one bin.
        NewHistogram(LoBin) = NewHistogram(LoBin) + 1;
    else
        % Accumulate low bin
        if LoBin >= 0
            NewHistogram(LoBin) = NewHistogram(LoBin) +
            BinIntegral(LoBin + 1) - LoBoundIntegral(Value);
        end
        % Accumulate center bins
        for CenterBin = (LoBin + 1):(HiBin - 1)
            if CenterBin >= 0
                NewHistogram(CenterBin) = NewHistogram(CenterBin) +
                BinIntegral(CenterBin + 1) - BinIntegral(CenterBin);
            end
        end
        % Accumulate high bin
        if HiBin >= 0
            NewHistogram(HiBin) = NewHistogram(HiBin) +
            HiBoundIntegral(Value) - BinIntegral(HiBin);
        end
    end
end

figure;
XAxis = 0:0.1:Bins - 1;
plot(xAxis, subs(IdealTransformedDistribution, xAxis), 'color', .7 * [1 1 1], 'LineWidth', 2);
hold on;
plot(NewHistogram, 'line', 'k', 'LineWidth', 2);
xlabel('Channel');
hold off;
AreaOfHistogram = sum(NewHistogram(:))

EventsInRawData = length(RawData)

**IdealCase2D Script**

```matlab
%*****************************************************************************
% Author: Patricio Vidal.
% Ph.D. Dissertation.
%*****************************************************************************

if ~exist('InputRange', 'var'), error('A InputRange variable is not defined.'); end
if ~exist('InvTransform', 'var'), error('A InvTransform function is not defined.'); end
if ~exist('Transform', 'var'), error('A Transform function is not defined.'); end

gaussian1 = Gaussian2D(100, Coordinate2D(5, 3), Coordinate2D(5, 3), 0.5);
gaussian2 = Gaussian2D(20, Coordinate2D(17, 12), Coordinate2D(10, 5), 0.2);

syms x;
syms y;
OriginalDistribution2D = gaussian1(x, y) + gaussian2(x, y);

xAxis = 0:199;

% 3D plot
figure
[xI, yI] = meshgrid(xAxis);
surfl(xI, yI, subs(OriginalDistribution2D, {'x', 'y'}, {xI, yI}));
shading interp
colormap gray
set(gcf, 'Name', 'Original distribution.');</ref>
view([42, 30]);
axis('tight');
xlabel('X Value');
ylabel('Y Value');
zlabel('Frequency');

% Histogram plot
originalHistogram2D = Histogram2D('Original distribution', [200 200], 'X Channel', 'Y Channel');
originalHistogram2D(:,:) = subs(OriginalDistribution2D, {'x', 'y'}, {xI, yI});

figure;
plot(originalHistogram2D);
xlabel('X Value');
ylabel('Y Value');
```

84
sum(originalHistogram2D(:))

% Ideal transformation.
InvTransformY = subs(InvTransform, y);
IdealTransformedDistribution2D = subs(OriginalDistribution2D, {'x',
'y'}, (InvTransform, InvTransformY)) .* diff(InvTransform) .*
diff(InvTransformY);

% 3D plot
figure
yAxis = linspace(0, Bins, 200);
[xI, yI] = meshgrid(yAxis);
surfl(xI, yI, subs(IdealTransformedDistribution2D, {'x', 'y'}, {xI,
yI}));
shading interp
colormap gray
set(gcf, 'Name', 'Ideal distribution.);
view([42, 30]);
axis('tight');
xlabel('X Value');
ylabel('Y Value');
zlabel('Frequency');

% Histogram plot
idealHistogram = Histogram2D('Ideal Transformed Data', [Bins Bins], 'X
Channel', 'Y Channel');
[xI, yI] = meshgrid(1:Bins);
idealHistogram(:, :) = subs(IdealTransformedDistribution2D, {'x', 'y'},
{xI, yI});

figure;
plot(idealHistogram, 100);
xlabel('X Value');
ylabel('Y Value');

BinningProblem2D Script

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patricio Vidal.
% Ph.D. Dissertation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xAxis = 0:199;
[xI, yI] = meshgrid(xAxis);
histogramData = Array2D(round(subs(OriginalDistribution2D, {'x', 'y'},
{xI, yI})));

% Original Histogram plot
originalHistogram2D = Histogram2D('Original distribution', [200 200],
'X Channel', 'Y Channel');
originalHistogram2D(:, :) = histogramData(:, :);
sprintf('Original area = %d', sum(originalHistogram2D(:)))

85
figure;
plot(originalHistogram2D);

% Binning effect
transformedXBins = floor(subs(Transform, xI));
transformedYBins = floor(subs(Transform, yI));
transformedHistogram = Histogram2D('Transformed Data', Bins, 'X Channel', 'Y Channel');

waitBarHandle = waitbar(0, 'Please wait...', 'CreateCancelButton', 'Cancel');
numElements = length(xAxis) ^ 2;
n = 0;
for i = 1:199
    for j = 1:199
        outputXBin = transformedYBins(i, j);
        outputYBin = transformedXBins(i, j);
        if((outputXBin >= 0) & (outputYBin >= 0))
            transformedHistogram(outputXBin, outputYBin) =
            transformedHistogram(outputXBin, outputYBin) + histogramData(i - 1, j - 1);
        end
        n = n + 1;
    end
    if(~rem(n, 100))
        waitbar(n / numElements, waitBarHandle);
    end
end
delete(waitBarHandle);

sprintf('Binning area = %d', sum(transformedHistogram(:)))

figure;
plot(transformedHistogram, 100);

UniformApproximation2D Script

% Author: Patricio Vidal.
% Ph.D. Dissertation.

% Uniform

% Create Approximation
uniformApproximation = subs(OriginalDistribution2D, {'x', 'y'},
    {maple('map', 'floor', x), maple('map', 'floor', y)});

% 3D plot
figure
xAxis = 0:0.1:50;
[xI, yI] = meshgrid(xAxis);
surfl(xI, yI, subs(uniformApproximation, {'x', 'y'}, {xI, yI}));
shading interp
colormap gray
set(gcf, 'Name', 'Uniform distribution. ');
view([42, 30]);
axis('tight');
xlabel('X Value'); ylabel('Y Value'); zlabel('Frequency');

% Transform

InvTransformY = subs(InvTransform, y);
transformedUniformApproximation = subs(uniformApproximation, {'x', 'y'}, {InvTransform, InvTransformY}).* diff(InvTransform).* diff(InvTransformY);

% 3D plot
figure
yAxis = linspace(0, Bins, 200);
[xI, yI] = meshgrid(yAxis);
surfl(xI, yI, subs(transformedUniformApproximation, {'x', 'y'}, {xI, yI}));
shading interp
colormap gray
set(gcf, 'Name', 'Uniform Transformed Data. ');
view([42, 30]);
axis('tight');
xlabel('X Channel'); ylabel('Y Channel'); zlabel('Frequency');

% Histogram plot
transformedHistogram = Histogram2D('Uniform Transformed Data',
size(xI), 'X Channel', 'Y Channel');
transformedHistogram(:, :) = subs(transformedUniformApproximation, {'x', 'y'}, {xI, yI});
figure;
plot(transformedHistogram, 100);
VITA

PATRICIO J. VIDAL

1968
Born, Santiago, Chile

1984
4th Place in National Mathematics Olympics
Caracas, Venezuela

1990
B.S., Electronic Engineering
First in Major
Simon Bolivar University
Caracas, Venezuela

1990-1996
R&D Professional
Engineering Institute,
Caracas, Venezuela

1996
M.S., Electronic Engineering
First in Major
Simon Bolivar University
Caracas, Venezuela

1996-2000
CATE Lab Manager
Florida International University
Miami, Florida

1998-1999
Alfred Estrada Scholar
Florida International University
Miami, Florida

1999
Internship
IBM
San Jose, California.

1999
Internship
Motorola, Inc.
Miami, Florida

2000-present
Software Engineer
Beckman Coulter, Inc.
Miami, Florida
PUBLICATIONS
