



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Signal and Information
Processing Laboratory

Institut für Signal- und
Informationsverarbeitung



Adaption of A Priori Models in Emg Signal Decomposition

Semester Thesis

Georg Böcherer

July 7, 2005

Professor: H-A. Loeliger

Advisor : Volker Koch

Danksagung

Ich möchte mich bei allen Personen vom Institut für Signal- und Informationsverarbeitung für die interessante Zeit während meiner Semesterarbeit bedanken. Mein besonderer Dank gilt meinem Betreuer Volker Koch, der mich die gesamte Zeit über selbstständig arbeiten ließ, sich gleichzeitig immer für meine Fragen und Ideen interessierte und sich stets die Zeit nahm mit mir über meine Arbeit zu sprechen. Er gab mir entscheidende Anregungen, ohne die meine Semesterarbeit in dieser Form nicht vorliegen könnte. Ein weiterer Dank geht an meine Kollegen aus meinem Arbeitszimmer für die abwechslungsreichen und interessanten Gespräche während unserer Arbeit: Wenji, Johan, Musa und Tunc.

Meiner Mutter wünsche ich von ganzem Herzen gute Besserung.

Zürich, den 1. Juli 2005

Georg Böcherer

Contents

1. Introduction	11
1.1. Abstract	11
1.2. Definitions	11
1.3. A Priori Models in Emg Decomposition	12
1.3.1. Definition	12
1.3.2. Some Examples of A Priori Models	12
1.4. Muaps and Firing Time Statistics	13
2. Adapting the Muaps	15
2.1. Problem Formulation	15
2.2. The Implementation	16
2.3. Results	17
2.3.1. Results from Simple Adaption	17
2.3.2. Results from Averaging	17
2.3.3. Results from Averaging with Aging	17
2.4. Conclusion and Outlook	18
3. Modeling Firing Time Statistics	19
3.1. Modeling the Statistics with Markov Chains	19
3.1.1. The Mu States	19
3.1.2. Characteristics of the Model	21
3.1.3. Proof of the Convergence	22
3.2. Overview: The Different Statistics Models	25
3.2.1. The Geometric Distribution	25
3.2.2. The Truncated Poisson-like Distribution	27
3.2.3. The Approximated Truncated Poisson-like Distribution	29
3.3. Conclusion and Outlook	32
4. Implementing Firing Time Statistics	35
4.1. Introduction	35
4.1.1. The Source Node	35
4.1.2. Messages towards the Coefficient Node	36
4.1.3. Messages towards the Fire Node	37
4.1.4. Messages towards the next Source Node	37
4.2. Class <code>EmgFnSource</code>	37
4.3. Class <code>EmgFnSourcePoisson</code>	37

Contents

4.3.1. Messages from Transition Node <code>FnModelX</code>	38
4.3.2. Messages from Transition Node <code>FnModelXI</code>	39
4.3.3. Messages from Equal Node <code>FnTest4Eq</code>	39
4.3.4. Messages from Conversion Node <code>FnConv</code>	40
4.3.5. The Update Schedule for the Internal Network	42
4.4. Verifying <code>EmgFnSourcePoisson</code> by its Convergence	43
4.5. Results	45
4.6. Conclusion and Outlook	46
5. Adapting A Priori Models by Iteration	49
5.1. The Expectation Maximization Algorithm	49
5.1.1. The General Case	49
5.1.2. An Example	50
5.2. The Estimation Minimization Algorithm	53
5.2.1. The General Formulation	53
5.2.2. Formulation for the Adaption of the Firing Time Statistics	53
5.3. Results From Adapting the Firing Time Statistics by Iteration	54
5.4. Conclusion and Outlook	55
A. Implementations	57
A.1. The class <code>AprioriModelParameter</code>	57
A.2. The class <code>AdaptionParameter</code>	59

List of Figures

1.1.	An Emg signal, Muaps and Muapts	12
2.1.	Simple adaption	17
2.2.	Adaption by averaging	18
3.1.	States of a Mu	20
3.2.	The PMF of the geometric distribution	25
3.3.	The geometric FSM	26
3.4.	The stationary PMF of the geometric FSM	27
3.5.	The Poisson problem	28
3.6.	The truncated Poisson-like FSM	29
3.7.	The stationary PMF of the truncated Poisson-like FSM	30
3.8.	The PMF of the aproximated truncated Poisson-like distribution	32
3.9.	The approximated truncated Poisson-like FSM	33
3.10.	Stationary PMF's of the truncated Poisson-like FSM and its approximation	33
4.1.	The factor graph	36
4.2.	The class <code>EmgNodeSource</code>	38
4.3.	The class <code>EmgFnSourcePoisson</code>	39
4.4.	The stationary PMF of <code>ModelXI</code>	44
4.5.	Results from the truncated Poisson-like model with wrong parameters	45
4.6.	Results from the geometric model	46
4.7.	Results from the approximated truncated Poisson-like model	47
4.8.	The PMF of the truncated Poisson-like distribution	48
5.1.	The effect of the EM algorithm on the PMF	52
5.2.	The expectation minimization algorithm: iteration 1	55
5.3.	The expectation minimization algorithm: iteration 2	56
5.4.	The expectation minimization algorithm: iteration 3	56

List of Figures

List of Tables

2.1. Notation in chapter 2	15
3.1. Notation in chapter 3	19
3.2. The geometric distribution	25
3.3. The Poisson distribution	27
3.4. The truncated Poisson-like distribution	28
3.5. The approximated truncated Poisson-like distribution	31
4.1. Notation in chapter 4	35
5.1. Notation in section 5.2	53
A.1. Example configuration of the statistics model	58
A.2. Example configuration of the adaption	60

List of Tables

1. Introduction

1.1. Abstract

My semester thesis is about adapting a priori models used in the decomposition of Emg signals. I elaborated theoretically how firing time statistics can be modeled as Markov Chains on finite state machines and how the properties of the models can be examined. Inspired by the expectation maximization algorithm, I formulated the estimation minimization algorithm which made it possible to adapt the parameters of the statistics model by iteration. Additionally I implemented the adaption of Muaps to motor unit action potentials changing over time.

1.2. Definitions

In my report, I use some terms which you probably don't know if you are not familiar with Emg signal decomposition. I will define these terms here. However, I will not give you an introduction into the domain of Emg signals and their decomposition in general. If you are interested in this, you should read [12] or [10]. My work makes part of the Emg research project on the ISI. I will not give you a general introduction into this project either. Informations about the approach in this project you find for example in [5], [2] and [10].

To understand this report, you need to know some basic terms of Emg signal decomposition. I will introduce them by explaining figure 1.1.

The blue curve on the top is an **electromyographic (Emg)** signal measured by inserting an electrode into a muscle. The Emg signal is the result of the activity of several **motor units (Mu)**. Each Mu can be identified by its motor unit action potential which has a specific shape. We want to represent the Emg signal as a superposition of the activities of the different Mus. We do this by trying to detect when each Mu is **active**. It is active when we can find the shape of the motor unit action potential in the Emg signal.

We therefore estimate the shapes of the motor unit action potentials. Such an estimation I call a **Muap**. The two plots in the second line of the figure show the two Muaps which were estimated. The moment when a Mu starts to be active is called a **firing time**. When a Mu is not active it is said to be **idle**. The firing times of a Mu can be represented by a discrete impulse train. The result from convoluting such an impulse train with a Muap is called a **Muap train (Muapt)** and can be interpreted as our estimation of the activity of one Mu. The third and the fourth line of the figure show two Muapts. The last line of the figure shows the superposition of the two Muapts. As you can see, this

1. Introduction

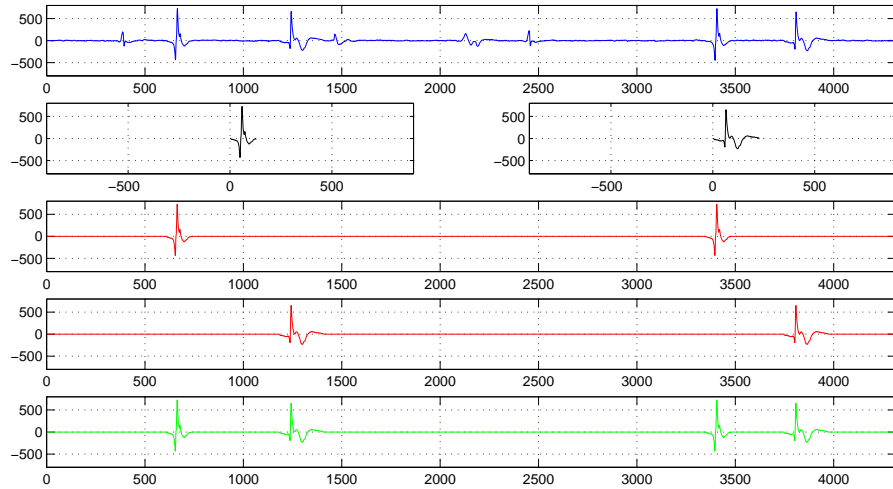


Figure 1.1.: From top to bottom: A measured Emg signal, two Muaps, Muapt 1, Muapt 2, the superposition of the Muaps.

superposition is quite similar to the measured Emg signal shown on the top.

1.3. A Priori Models in Emg Decomposition

1.3.1. Definition

Compared to the definition of an a priori model used in [6], I will give a more general meaning to it:

Definition 1. *When decomposing an Emg signal, the information the decomposition is based on additionally to the signal itself forms an **a priori model**.*

1.3.2. Some Examples of A Priori Models

I will give some examples of a priori models to illustrate this definition. First, the a priori model used for the segmentation of an Emg signal as described in [15]:

- The two states active and idle of a Mu can be modeled by two Gaussian distributions with two different standard deviations σ_{active} and σ_{idle} .
- The transitions active \rightarrow idle and active \leftarrow idle occur with the probabilities ϵ_0 and ϵ_1 respectively.

Secondly, the a priori model used for the initial estimation of the Muaps as described in [10]:

- The results from the segmentation as described above.
- All Mus have to fire alone (their Muaps are not in superposition with a Muap from another Mu) at least once in the Emg signal.
- The Emg signal is the superposition of at least 2 Muaps.
- The segmentation provided at least 3 (active) segments.

Finally, the a priori model used for the decomposition of the Emg signal:

- The results from the estimation of the Muaps.
- There are no self-superpositions of a Muap to itself in the Emg signal.
- The firing times follow a certain statistics.

1.4. Muaps and Firing Time Statistics

In my semester thesis, I mainly worked on the Muaps and the firing time statistics. The main tasks were the following:

For the Muaps:

- So far, we assumed that the Muaps remain constant over time. In real signals however, the motor unit action potentials change over time. How can we track these changes by adapting the Muaps?

For the firing time statistics:

- So far, the a priori model we use for the firing time statistics implements a geometric distribution, in contrast to the statistics we have can observe in practice, which is rather a Poisson distribution of the distances between firings. In [16], we have a proposition how we could implement a Poisson distribution as an a priori model. This has to be implemented and tested.
- How can we find the right parameters for the statistics model?

1. Introduction

2. Adapting the Muaps

In this chapter, I explain how I implemented the adaption of the Muaps to changing motor unit action potentials in the signal and which results I obtained with using Muap adaption when decomposing Emg signals. In table 2.1, you find the notation I will use in this chapter.

Counter	i, j
Firing time	$x_s^{(i)}$
Impulse train of firing times	$x_s[\cdot]$
Mu	s
Muap	c_s
Muap length	L_s
Muapt	t_s
New Muap	c'_s
Number of Mus	S
Number of Realisations	R_s
Realisation	$r_s^{(i)}$
Residual	$\mathbf{r}[\cdot]$
Sample	k

Table 2.1.: Notation in chapter 2

2.1. Problem Formulation

So far, we mainly worked with the assumption that the Motor unit action potentials remain constant over time. In reality, this is not the case, so we should somehow adapt our Muaps to these changes.

First, we correct our assumption to the following:

Muaps remain constant over a certain interval of time which I will call the blocklength.

Now we can process the Emg signal block by block and at the end of each block, we adapt the Muaps to the realisations found during the detection.

2. Adapting the Muaps

2.2. The Implementation

I decided to implement the adaption in a separate class `Adaption`. I will first give an overview over the implemented functionality and the idea behind it and then illustrate some results from the practice.

At first, I will give the definition of two terms that I will use throughout this section.

Definition 2. The *residual* $\tau[\cdot]$ of a signal $y[\cdot]$ is that part of the signal which remains when we subtract all detected Muaps $t_s[\cdot]$ from it. The detected Muaps $t_s[\cdot]$ are equal to the convolution of the detected impulse train $x_s[\cdot]$ indicating the firing times with the current Muaps $c_s[\cdot]$. The residual becomes

$$\tau[\cdot] = y[\cdot] - \sum_{s=1}^S t_s[\cdot] = y[\cdot] - \sum_{s=1}^S (x_s * c_s)[\cdot]$$

where S is the number of Mus.

Definition 3. A *realisation* $r_s^{(i)}[\cdot]$ of a Muap c_s at firing time $x_s^{(i)}$ is the sum of the residual τ left shifted by $x_s^{(i)}$ and the current Muap c_s of length L_s , all multiplied by a unit gain window of length L_s :

$$r_s^{(i)}[k] = \begin{cases} \tau[k + x_s^{(i)}] + c_s[k], & 0 \leq k < L_s \\ 0, & \text{otherwise} \end{cases}$$

Assume that we found R_s realisations of Muap c_s in the current block. There are several ways to calculate a new Muap c'_s for the next block:

Simple Adaption:

$$c'_s = r_s^{(R_s)}$$

Averaging:

$$c'_s = \frac{1}{R_s} \sum_{i=1}^{R_s} r_s^{(i)}$$

Averaging with aging:

$$c'_s = \frac{1}{\sum_{j=1}^{R_s} w_j} \sum_{i=1}^{R_s} w_i r_s^{(i)}$$

where w_i defines the weight of realisation $r_s^{(i)}$ (it can be reasonable to give a higher weight to newer realisations than to older ones).

The rest of the implemented functionality you find shortly described in appendix A.2. In the next section, I explain results I could obtain with the class `Adaption`.

2.3. Results

2.3.1. Results from Simple Adaption

To test the simple adaption, I generated a signal with `emgsynth` which was developed in [11]. The signal has one Mu and a strong trend variability which both results in an increasing amplitude and heavy shape changes. As you can see in figure 2.1, the Muap could be adapted quite well. This was reached by using simple adaption and a blocklength of 1000. Additionally, the adapted Muaps were smoothed by a Gaussian filter of length 25 to remove the noise. In the upper plot, you see the Muaps which were used in each block to decompose the signal shown in the plot below. You can see the similarity of the Muap in one block with the signal in the previous block.

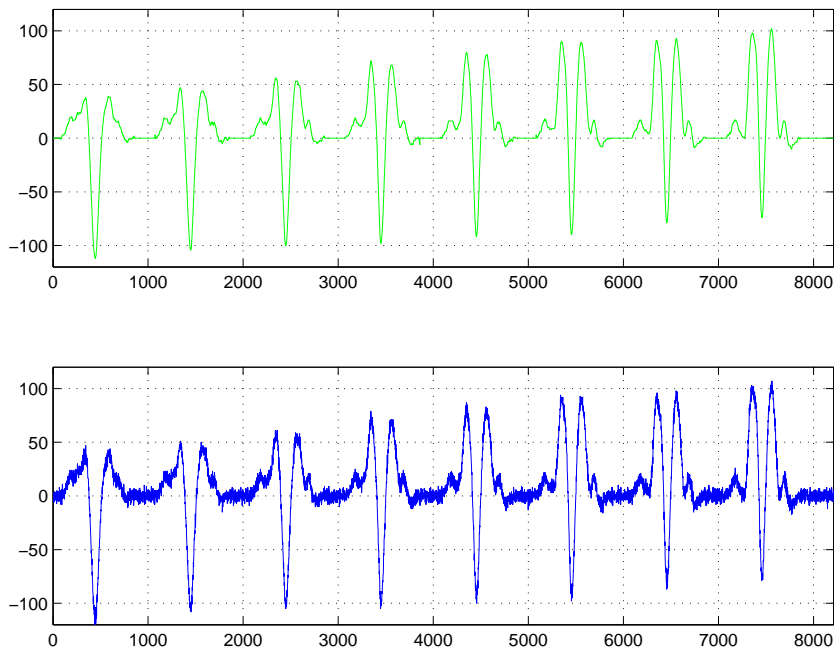


Figure 2.1.: Adaption of the model by simple adaption and block length 1000.

2.3.2. Results from Averaging

In figure 2.2, you can see how we can profit from the adaption by averaging. The Muap on the left was estimated by the algorithm developed in [10]. The realisations of this Muap however didn't show this hill on the right and by averaging over the estimated Muap and the realisations, the hill disappeared and the Muap could be cut.

2.3.3. Results from Averaging with Aging

I couldn't find examples where averaging with aging lead to better results yet.

2. Adapting the Muaps

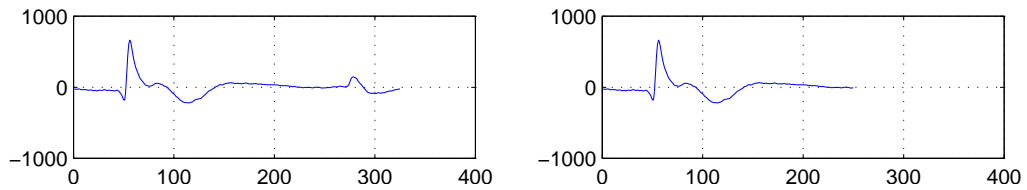


Figure 2.2.: Adaption of the model by averaging.

2.4. Conclusion and Outlook

I developed the adaption to be able to adapt the Muaps to motor unit action potentials in real signals which change over time. It has to be tested extensively on real signals because the remaining question is:

- Do we get better decomposition results when we adapt the Muaps by the simple methods developed in this chapter?

From my observations, I believe that the assumption that the motor unit action potentials remain constant over a certain interval of time holds, so the answer to the question should be yes, at least when we can find realisations of the Muaps without superpositions. The adaption then should work perfectly.

However when we try to decompose signals with heavy superpositions, my simple methods will not work anymore, hence we should somehow adapt our Muaps in a more sophisticated way. Some ideas about how this could be done:

- We could extend our channel model: To generate a Muapt, we currently simply use an impulse train representing the firing times and convolve it with the Muaps. If we look at the setup used to measure Emg signals, it is however obvious that there is a channel between the muscle and the monitor which effects the measured Muaps all in the same way. This should be investigated.
- We could correlate the information we get from different channels when measuring the Emg signal by several electrodes at the same time. Currently we mainly work with one channel.
- We could redefine the Muaps: Currently they work as FIR filters which we convolve with the impulse trains representing the firing times. This model is far away from reality: A motor unit action potential is in reality not the result of passing an impulse through a frequency selective channel but it is caused by a chemical process which takes time and therefore it has a length larger than zero.
- The sophisticated model could perhaps be adapted by methods as described in chapter 5.

3. Modeling Firing Time Statistics

In this chapter, I talk about how we can model the firing time statistics. I will first give an theoretic introduction and then give an overview of the models which can be used when decomposing Emg signals. The basics of probability theory which I use in my explanations can be found in [8]. In table 3.1, you find the notation I use in this chapter.

Average PMF	$\bar{\mathbf{p}}, \mathbf{p}_1$
Counter	i, j, m, n, x
Distribution parameter	μ, σ, λ, c
Impulse train of firing times	$X[\cdot]$
Muap length	L
Number of idle states	N
Path	w_i
Period	ξ
PMF	\mathbf{p}
Prime number	α, β
Probability	$P(\dots)$
Probability of a state	p_i
Probability of a transition	$\epsilon_0, \epsilon_1, t_{ij}$
Sample	k
Set of states	S
States	S_i
Transition matrix	$\mathbf{T}, \mathbf{T}_g, \mathbf{T}_p, \mathbf{T}_a$

Table 3.1.: Notation in chapter 3

3.1. Modeling the Statistics with Markov Chains

3.1.1. The Mu States

At one sample k , a Mu can be either in an active or in an idle state. The active state can be divided into L different states where L is the length of the Muap corresponding to the Mu. The active states I will denote by $S_1..S_L$ and the idle state I will denote by S_{L+1} . The Mu fires in state S_1 .

The state of the Mu at sample k of the signal is a random variable $X[k]$ taking values in $S = \{S_1, S_2, \dots, S_{L+1}\}$. A realisation of $X[\cdot]$ is a **random walk** through the states in S .

3. Modeling Firing Time Statistics

During this walk, the next step depends only on the present state and not on the past, so we have

$$P(X[k+1] = x_{k+1} | x_k, x_{k-1}, \dots, x_0) = P(X[k+1] = x_{k+1} | x_k)$$

where $x_i \in S$ denotes the realisation of $X[i]$. The equation implies that the steps of our walk form a **Markov Chain**. Since we assumed that a Muap cannot be superposed to itself, not all transitions are allowed. In figure 3.1, you can see the **finite state machine (FSM)** illustrating the possible transitions with the corresponding probabilities.

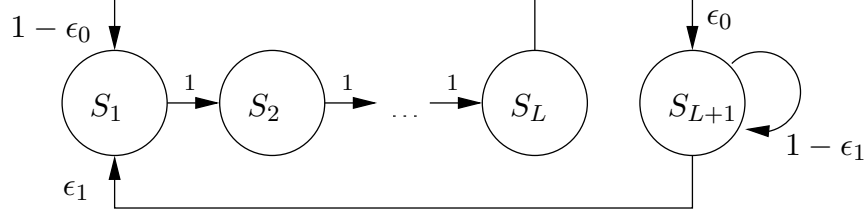


Figure 3.1.: The different states of a Mu and the possible transitions.

We have for example

$$P(X[k+1] = S_2 | X[k] = S_1) = 1$$

and

$$P(X[k+1] = S_1 | X[k] = S_{L+1}) = \epsilon_1$$

We will now examine the statistics of the firing times defined by ϵ_0 and ϵ_1 .

Let $\mathbf{p}[k] \in \{0..1\}^{L+1}$ be the **probability mass function (PMF)** of $X[k]$ such that

$$p_i[k] = P(X[k] = S_i) \quad \forall i \in \{1..L+1\}$$

$$\sum_{i=1}^{L+1} p_i[k] = 1$$

where $p_i[k]$ is the i -th entry of $\mathbf{p}[k]$. Let t_{ij} denote the probability that we go from state S_i to state S_j :

$$t_{ij} = P(X[k+1] = S_j | X[k] = S_i)$$

We can calculate the entries p_j of the PMF of $X[k+1]$ given the PMF of $X[k]$ in the following way:

$$p_j[k+1] = \sum_{i=1}^{L+1} p_i[k] t_{ij}$$

I define the transition matrix $\mathbf{T} \in \{0..1\}^{(L+1) \times (L+1)}$ as follows:

$$(\mathbf{T})_{ij} = t_{ij} \quad \forall i, j \in \{1, \dots, L+1\}$$

The transition from $\mathbf{p}[k]$ to $\mathbf{p}[k+1]$ can now be written in the compact form

$$\mathbf{p}[k+1] = \mathbf{p}[k] \mathbf{T}$$

where the matrix \mathbf{T} corresponding to the FSM in figure 3.1 is of the form

$$\mathbf{T} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & & 0 & 1 & 0 \\ 1 - \epsilon_0 & 0 & \dots & 0 & \epsilon_0 \\ \epsilon_1 & 0 & \dots & 0 & 1 - \epsilon_1 \end{pmatrix}$$

3.1.2. Characteristics of the Model

As derived in the previous section, we can represent our model for the statistics of a Mu by a transition matrix \mathbf{T} . With this matrix, we can calculate the PMF $\mathbf{p}[k+1]$ from the PMF $\mathbf{p}[k]$. It would be nice if we could say something about the average PMF of the states over all samples k . Let us consider an example. With $L = 2$, $\epsilon_0 = 1$ and $\epsilon_1 = 0.1$ \mathbf{T} becomes

$$\mathbf{T} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.1 & 0 & 0.9 \end{pmatrix}$$

For \mathbf{p}_0 I choose

$$\mathbf{p}_0 = (1 \ 0 \ 0)$$

From matlab, we get

$$\lim_{k \rightarrow \infty} \mathbf{p}[k] = \lim_{k \rightarrow \infty} \mathbf{p}_0 \mathbf{T}^k \approx \begin{pmatrix} 0.0833 \\ 0.0833 \\ 0.8333 \end{pmatrix}$$

If we choose a different initial distribution \mathbf{p}_0 , for example

$$\mathbf{p}_0 = (0.5 \ 0.25 \ 0.25)$$

we get

$$\lim_{k \rightarrow \infty} \mathbf{p}_0 \mathbf{T}^k \approx \begin{pmatrix} 0.0833 \\ 0.0833 \\ 0.8333 \end{pmatrix}$$

which is exactly the same result. There seems to be a stationary PMF $\bar{\mathbf{p}}$ such that

$$\bar{\mathbf{p}} \mathbf{T} = \bar{\mathbf{p}}$$

which implies that $\bar{\mathbf{p}}$ is the average PMF of the states which we are looking for. For any initial PMF \mathbf{p}_0 , $\mathbf{p}[k]$ seems to converge to $\bar{\mathbf{p}}$ for $k \rightarrow \infty$.

If we can calculate $\bar{\mathbf{p}}$ by iteration, we can easily use it

- to characterize our model,
- to test if our implementation fulfills the specifications by checking its convergence by iteration.

3. Modeling Firing Time Statistics

For this we have to prove that

- $\bar{\mathbf{p}}$ is unique
- $\lim_{k \rightarrow \infty} \mathbf{p}_0 \mathbf{T}^k = \bar{\mathbf{p}}$ for any \mathbf{p}_0 .

3.1.3. Proof of the Convergence

For the proof of the unique convergence of our Markov Chain represented by the transition matrix \mathbf{T} I first have to introduce some mathematical terms. All definitions and theorems listed in this section I found in [14].

Definition 4. The *period* of a state $S_i \in S$ is the largest number $\xi \in \mathbb{N}$ such that

$$\{n \in \mathbb{N} | p_{ii}^{(n)} > 0\} \subseteq \{j \cdot \xi | j \in \mathbb{N}\}$$

where $p_{ii}^{(n)}$ denotes the probability of going in n steps from state S_i to state S_i .

Definition 5. A state with period $\xi = 1$ is called *aperiodic*. A Markov Chain is aperiodic when every state is aperiodic.

Theorem 1. A state S_i is aperiodic if it has the following property:

$$\exists n, m \in \mathbb{N} : p_{ii}^{(n)} > 0, p_{ii}^{(m)} > 0 \text{ and } \text{gcd}(n, m) = 1$$

where $\text{gcd}(n, m)$ is the greatest common divisor of n and m .

Definition 6. A Markov Chain is called *irreducible* if for any states S_i, S_j there is a $n \in \mathbb{N}$ such that

$$p_{ij}^{(n)} > 0$$

In other words, it is irreducible if there is a path from S_i to S_j for any $S_i, S_j \in S$.

Definition 7. A Markov Chain is called *ergodic* if it is both aperiodic and irreducible.

With these definitions I can formulate the fundamental theorem for ergodic Markov Chains:

Theorem 2. For any ergodic Markov Chain defined by its transition matrix \mathbf{T} , we have

$$\lim_{k \rightarrow \infty} \mathbf{p}_0 \mathbf{T}^k = \bar{\mathbf{p}}$$

where \mathbf{p}_0 is an arbitrary initial distribution of the states and $\bar{\mathbf{p}}$ the unique stationary PMF of the Markov Chain.

Remark 1. $\bar{\mathbf{p}}$ is the eigenvector of \mathbf{T} corresponding to the eigenvalue 1. I will therefore denote it by \mathbf{p}_1 in the following.

If we can show that the Markov Chain of our statistic model represented by the matrix \mathbf{T} is ergodic, we can use the convergence property without any problems.

3.1. Modeling the Statistics with Markov Chains

Proposition 1. *The Markov Chain represented by the matrix \mathbf{T} introduced in the previous section is ergodic **if and only if (iff)** $\epsilon_0 > 0$, $\epsilon_1 > 0$ and $\epsilon_0 + \epsilon_1 < 2$.*

Proof. I will prove proposition 1 in two steps: In the first step I will show that \mathbf{T} is irreducible iff both ϵ_0 and ϵ_1 are larger then zero, and in the second step I will show that if both ϵ_0 and ϵ_1 are larger then zero, then \mathbf{T} is aperiodic iff $\epsilon_0 + \epsilon_1 < 2$.

1st step. As stated in definition 6, \mathbf{T} is irreducible when you can find for any pair $S_i, S_j \in S$ a path from S_i to S_j in the corresponding FSM. If $\epsilon_0 = 0$, we cannot go from the active states to the idle state, and if $\epsilon_1 = 0$, we cannot go from the idle state to the active states. So in these cases, \mathbf{T} is not irreducible. Let now both ϵ_0 and ϵ_1 be larger then 0. In this case, we can go the following path w_0 through the FSM:

$$w_0 = (S_2, \dots, S_L, S_{L+1}, S_1)$$

Where the state we start from (e.g. S_1) is not part of the path.

We can go from S_1 to S_1 along path w_0 and since every $S_i \in S$ lies on w_0 , there is a path from S_i to S_j for any pair $S_i, S_j \in S$, which implies that \mathbf{T} is irreducible.

2nd step. Let both ϵ_0 and ϵ_1 be larger then 0. We have to show that \mathbf{T} is in this case aperiodic iff $\epsilon_0 + \epsilon_1 < 2$. The condition is fulfilled if either $\epsilon_0 < 1$ or $\epsilon_1 < 1$ and it is not if both $\epsilon_0 = 1$ and $\epsilon_1 = 1$, so we have to treat these 3 cases separately.

case 1: $\epsilon_0 < 1$. I define the following two paths:

$$\begin{aligned} w_1 &= (S_1, S_2, S_3, \dots, S_L) \\ w_2 &= (S_{L+1}, S_1, S_2, \dots, S_L) \end{aligned}$$

Again, the state we start from is not part of the path. Note that any concatenations of w_1 and w_2 are also valid paths through the FSM.

Let $\alpha > L^2$ be a prime number. We are looking for a path from S_L to S_L of length α . With $x \equiv \alpha \pmod{L}$ and $\mathbb{N} \ni n = \frac{\alpha - x}{L}$, α can be written as

$$\begin{aligned} \alpha &= n \cdot L + x \\ &= \underbrace{(n - x)}_{>0 \text{ since } \alpha > L^2} \cdot L + L \cdot x + x \\ &= (n - x)L + (L + 1)x \end{aligned}$$

Since w_1 is of length L and w_2 is of length $L + 1$, we start at S_L and go $(n - x)$ times path w_1 and x times path w_2 which is equivalent to a path from S_L to S_L of length α . Since this path is closed and since it goes through all states $S_i \in S$, we have for any state $S_i \in S$ a path from S_i to S_i of length α , which is equivalent to $p_{ii}^{(\alpha)} > 0 \forall S_i \in S$. Hence for any prime number $\alpha > L^2$, we can construct a closed path of length α going through all states in S . Let $\alpha, \beta > L^2$, $\alpha \neq \beta$ be two prime numbers. We have shown that

$$p_{ii}^{(\alpha)} > 0 \text{ and } p_{ii}^{(\beta)} > 0 \forall S_i \in S$$

and since α and β are prime numbers we have

$$\gcd(\alpha, \beta) = 1$$

3. Modeling Firing Time Statistics

so by invoking theorem 1, we have shown that \mathbf{T} is aperiodic¹ for $\epsilon_0 < 1$.

case 2: $\epsilon_1 < 1$. The proof for this case is quite similar. We define the two paths

$$\begin{aligned} w_3 &= (S_1, S_2, S_3, \dots, S_L, S_{L+1}) \\ w_4 &= (S_1, S_2, \dots, S_L, S_{L+1}, S_{L+1}) \end{aligned}$$

and let $\alpha = n \cdot (L + 1) + x > (L + 1)^2$, $x \equiv \text{mod}(\alpha, L + 1)$, $n \in \mathbb{N}$ be a prime number. α can be written as:

$$\begin{aligned} \alpha &= n(L + 1) + x \\ &= \underbrace{(n - x)}_{>0 \text{ since } \alpha > (L+1)^2} \cdot (L + 1) + (L + 1)x + x \\ &= (n - x)(L + 1) + (L + 2)x \end{aligned}$$

We now start at S_{L+1} , we go $(n - x)$ times path w_3 which is of length $(L + 1)$ and x times path w_4 which is of length $(L + 2)$. So with the same argumentation as in case 1, we can show that \mathbf{T} is aperiodic for $\epsilon_1 < 1$.

case 3: $\epsilon_0 = \epsilon_1 = 1$. In this case we have only one path through all states of the FSM:

$$w_5 = (S_1, S_2, \dots, S_L, S_{L+1})$$

Let us consider state S_{L+1} : It has the period of $\xi = L + 1 > 1$, since the only way to go from S_{L+1} to S_{L+1} is along path w_5 and its multiples, so S_{L+1} is not aperiodic which implies that \mathbf{T} is not aperiodic if $\epsilon_0 = \epsilon_1 = 1$. □

¹This was quite complicated. With the properties of the paths w_1 and w_2 you can with definition 4 prove this directly. How?

3.2. Overview: The Different Statistics Models

In this section, I give an overview over the different statistics models which have been used to model the firing time statistics so far. For every model, I give the distribution of the distance between two firings, the FSM used to implement the distribution, the matrix which defines the probabilities of the transitions between the different states and the stationary PMF of the states.

3.2.1. The Geometric Distribution

The first model of the firing time statistics was introduced in [2] and implemented the geometric distribution. The way I define the FSM for this distribution differs slightly from the definition which was given in [2]. With my definition it is however easier to compare the different statistics models.

The Distribution

The distribution of the distance between the firing times used in this approach is the **geometric distribution** which has the following properties:

PMF	$(1 - \epsilon_1)^{n-1} \epsilon_1, n \geq 0$
Mean	$\frac{1-\epsilon_1}{\epsilon_1}$
Variance	$\frac{1-\epsilon_1}{\epsilon_1^2}$

Table 3.2.: The geometric distribution

The PMF is plotted in Figure 3.2.

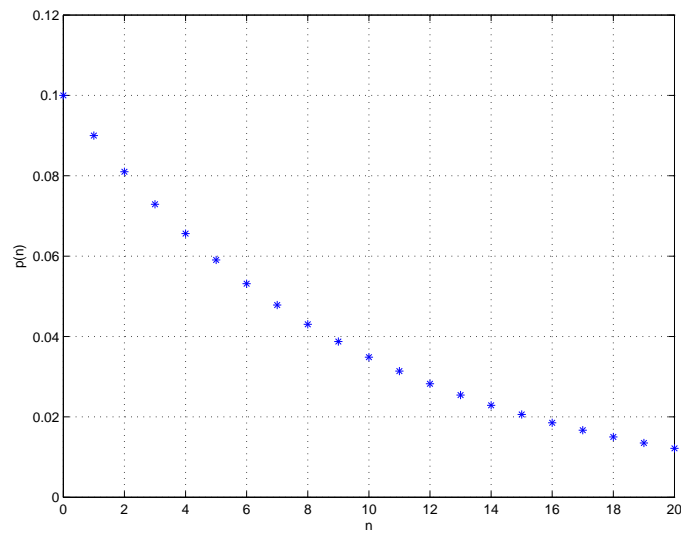


Figure 3.2.: The PMF of the geometric distribution for $\epsilon_1 = 0.1$.

3. Modeling Firing Time Statistics

The State Machine

The FSM used to implement this distribution can be seen in figure 3.3. It is important to notify that the transition from the last active state S_L to the first active state S_1 has probability 0, which doesn't make sense from the theoretical point of view. In practice however this was never problematic.

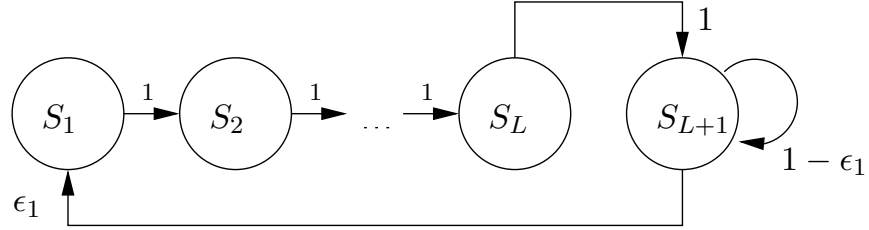


Figure 3.3.: The FSM which implements the geometric distribution

The Transition Matrix

The Transition matrix \mathbf{T}_g is the following:

$$\mathbf{T}_g = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 1 \\ \epsilon_1 & 0 & \dots & 0 & 1 - \epsilon_1 \end{pmatrix}$$

\mathbf{T}_g is equal to the matrix \mathbf{T} defined in section 3.1 when you set ϵ_0 equal 1.

The Stationary PMF

With $0 < \epsilon_1 < 1$, the PMF of the states converges because of proposition 1. We can calculate the stationary PMF \mathbf{p}_1 by invoking remark 1. We have to solve

$$\begin{aligned} \mathbf{p}\mathbf{T} &= \mathbf{p} \\ \iff \\ \mathbf{p}(\mathbf{T} - \mathbf{I}) &= \mathbf{0} \end{aligned}$$

with the additional condition $\sum_{i=1}^{L+1} (\mathbf{p})_i = 1$. The analytic solution is

$$\mathbf{p}_1 = \left(\frac{\epsilon_1}{L\epsilon_1+1} \quad \dots \quad \frac{\epsilon_1}{L\epsilon_1+1} \quad \frac{1}{L\epsilon_1+1} \right)$$

An example of the stationary PMF \mathbf{p}_1 is plotted in figure 3.4.

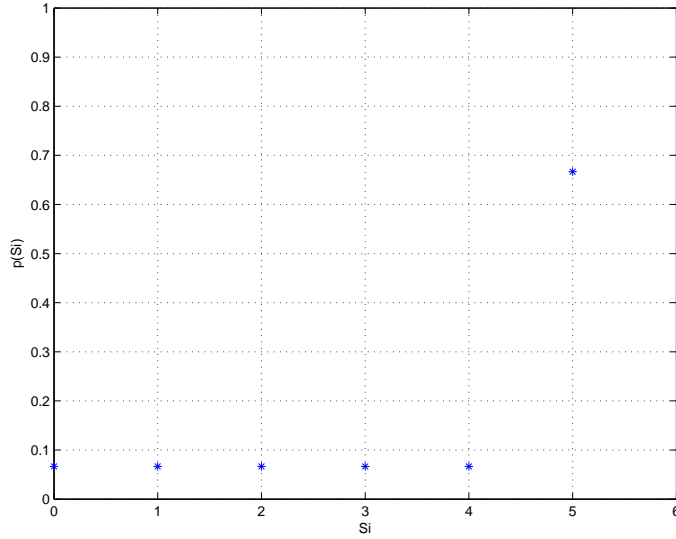


Figure 3.4.: The stationary PMF of the geometric model for $L = 5$ and $\epsilon_1 = 0.1$

3.2.2. The Truncated Poisson-like Distribution

Since the distances of the firing times in reality follow rather a Poisson distribution than a geometric distribution, Christian Zimmermann elaborated in [16] a way how to implement a Poisson-like distribution.

The Distribution

The Poisson distribution has the properties listed in table 3.3. As you can see, the

PMF	$\frac{\lambda e^{-\lambda}}{n!}, n \geq 0$
Mean	λ
Variance	λ

Table 3.3.: The Poisson distribution

Poisson distribution is discrete but the reality is not. For different sampling times we get completely different distributions if we simply try to adapt our model by changing the mean value λ . Figure 3.5 illustrates this problem. The reason is that for the Poisson distribution, the mean value is equal to the standard deviation. Christian Zimmermann solved this problem by interpolating a discrete Poisson PMF with the sinc function, resampling this continuous function and normalizing it. With this, he constructed a distribution which I call the Poisson-like distribution, where it is possible to specify the mean value μ and the standard deviation σ separately.

When implementing this distribution in a FSM, the PMF has additionally to be truncated at N (The PMF is set equal 0 for $n > N$) and then normalized. I will denote

3. Modeling Firing Time Statistics

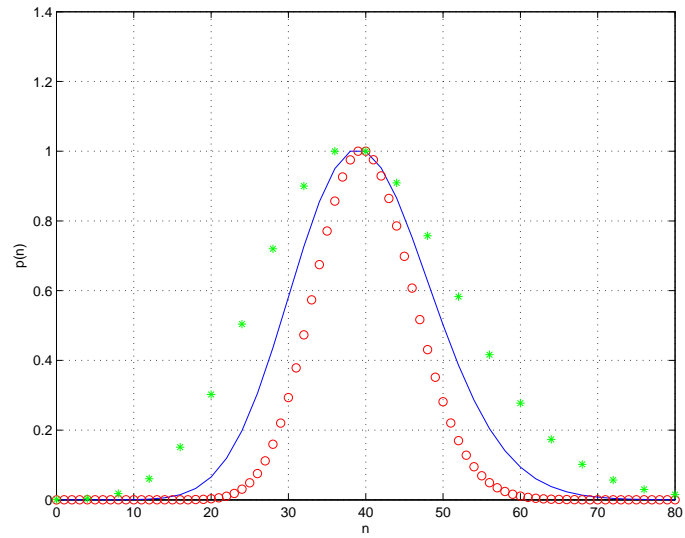


Figure 3.5.: Problems when describing a continuous Poisson PMF by a discrete one: Oversampling (\circ) and undersampling ($*$).

the PMF of this distribution by $p_{\mu,\sigma}^{(z)}$. It can be calculated by the java class `Distribution.java` which is part of the `jsim` package.

PMF	$p_{\mu,\sigma}^{(z)}(n), 0 \leq n \leq N$
Mean	$\approx \mu$
Variance	$\approx \sigma^2$

Table 3.4.: The truncated Poisson-like distribution

μ and σ are the correct values for the mean and the standard deviation of the Poisson-like distribution. But since $p_{\mu,\sigma}^{(z)}$ is truncated at N , these values are only approximations for the mean and the standard deviation of the truncated Poisson-like distribution.

The State Machine

In figure 3.6, you can see the FSM which Christian Zimmermann proposed in [16] to implement $p_{\mu,\sigma}^{(z)}$. There is one fundamental problem with this FSM: The probability $P(n \leq N)$ is equal one, which comes from the fact that we only use the finite number $N + L$ of states in the FSM.

Remark 2. Any distribution over the finite set $\{0, 1, \dots, N\}$ can be used to define the transition probabilities in the FSM shown in figure 3.6.

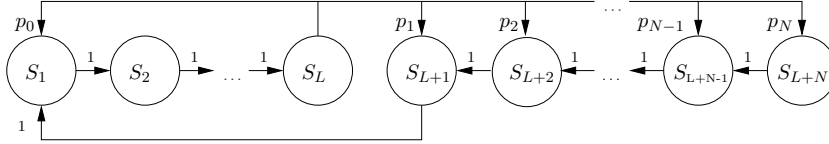


Figure 3.6.: The FSM implementing the Poisson-like distribution with $p_i = p_{\mu, \sigma}^{(z)}(i)$ and the number of idle states equal N .

The Transition Matrix

The transition matrix \mathbf{T}_p is the following:

$$\mathbf{T}_p = \begin{pmatrix} 0 & 1 & 0 & \dots & & & & 0 \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ 0 & \dots & 0 & 1 & 0 & \dots & & 0 \\ p_0 & 0 & \dots & 0 & p_1 & \dots & p_{N-1} & p_N \\ 1 & 0 & \dots & & & & & 0 \\ 0 & \dots & & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & & & & & 1 & 0 \end{pmatrix}$$

The Stationary PMF

With $p_i > 0 \forall i \in \{0, 1, \dots, N\}$, the PMF of the states converges, which is a consequence of the proof I gave for proposition 1 (You only have to change some indices). This allows us to calculate the eigenvector of \mathbf{T}_p for the eigenvalue 1 by iteration which gives us the stationary PMF \mathbf{p}_1 .

On the left half of figure 3.7, you see the PMF $p_{\mu, \sigma}^{(z)}(n)$ and $1 - \text{cdf}_{\mu, \sigma}^{(z)}(n)$ where $\text{cdf}_{\mu, \sigma}^{(z)}(n)$ is the **cumulative distribution function (CDF)** of $p_{\mu, \sigma}^{(z)}$. On the right side, you see on the top the stationary PMF \mathbf{p}_1 of the FSM and on the bottom \mathbf{p}_1 and $1 - \text{cdf}_{\mu, \sigma}^{(z)}(n)$ shifted to the right by the number of active states and scaled by the stationary probability of one active state. As we would expect, this fits perfectly.

Comparing 1 minus the CDF of the implemented PMF to the stationary PMF of the FSM calculated by iteration is another possibility to check if the FSM was implemented correctly.

3.2.3. The Approximated Truncated Poisson-like Distribution

As I said in section 3.2.1 and section 3.2.2, the FSM's introduced so far have still some problems. To solve these, I propose a combination of the two former models by approximating the truncated Poisson-like distribution for $n > \mu$ by a geometric distribution. This approximation has two fundamental advantages:

3. Modeling Firing Time Statistics

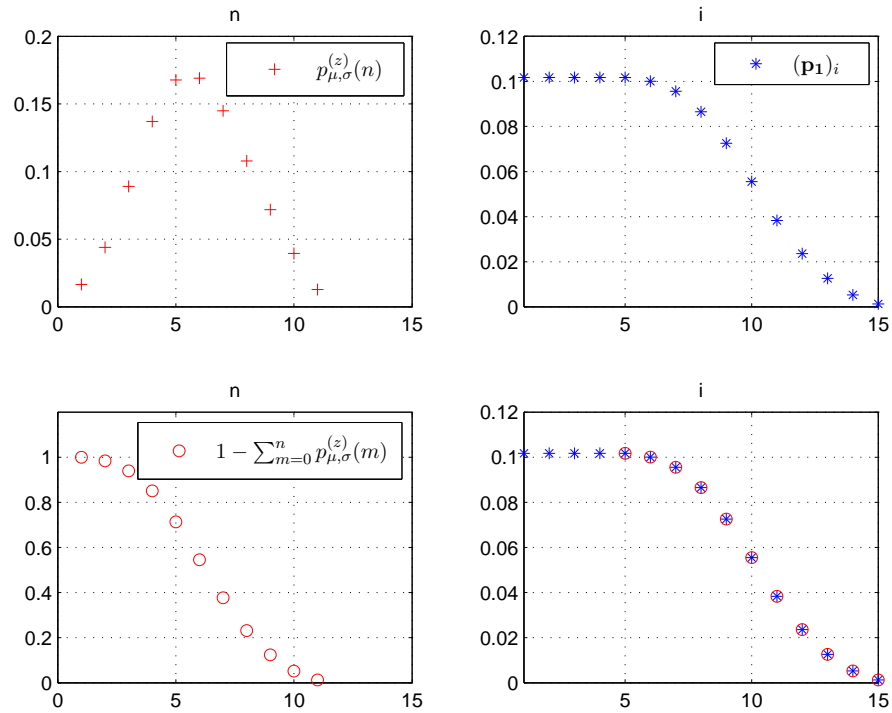


Figure 3.7.: The stationary PMF of the FSM from figure 3.6. The transition probabilities were set to $p_i = p_{\mu,\sigma}^{(z)}(i)$ with $\mu = 5$, $\sigma = \sqrt{5}$ and $N = 10$.

- The FSM needs less states than the Poisson-like distribution for the same values of μ and σ .
- It defines a PMF over an infinite set. In practice, this turns out to be the main advantage as I will show in the next section.

The Distribution

It is not clear if the mean and the variance are approximated correctly by the corresponding values of the Poisson-like distribution. In figure 3.8 the PMF of the truncated Poisson-like distribution and its approximation are plotted for different values of σ . As you can see, the curves fit quite well, which implies that the approximations made for the mean and the variance are reasonable.

At least, the PMF is defined correctly. Let $p_{\mu,\sigma}^{(a)}(n)$ be the approximated PMF of the truncated Poisson-like distribution. For $N > \mu + 1$, we have $\epsilon > 0$, $p_{\mu,\sigma}^{(a)}(n) \geq 0 \forall n \in \mathbb{Z}$

3.2. Overview: The Different Statistics Models

$$\begin{array}{l}
 \text{PMF} \quad \left\{ \begin{array}{l} p_{\mu,\sigma}^{(z)}(n), \quad 0 \leq n \leq \mu \\ c \cdot \epsilon_1 (1 - \epsilon_1)^{n-\mu-1}, \quad n \geq \mu + 1 \end{array} \right. \\
 \text{Mean} \quad \approx \mu \\
 \text{Variance} \quad \approx \sigma^2
 \end{array}$$

$$c = 1 - \sum_{n=0}^{n=\mu} p_{\mu,\sigma}^{(z)}(n)$$

$$\epsilon_1 = p_{\mu,\sigma}^{(z)}(\mu + 1)/c$$

Table 3.5.: The approximated truncated Poisson-like distribution

and

$$\begin{aligned}
 \sum_{n=-\infty}^{\infty} p_{\mu,\sigma}^{(a)}(n) &= \sum_{n=0}^{\infty} p_{\mu,\sigma}^{(a)}(n) \\
 &= \sum_{n=0}^{\mu} p_{\mu,\sigma}^{(z)}(n) + c \cdot \sum_{n=0}^{\infty} \epsilon_1 (1 - \epsilon_1)^n \\
 &= 1 - c + c \cdot 1 \\
 &= 1
 \end{aligned}$$

The State Machine

The FSM to implement the approximation of the truncated Poisson-like distribution is quite similar to the one from figure 3.6. I simply added a transition from state S_{L+N} to itself and inserted the parameter ϵ_1 for the probabilities of the transitions from state S_{L+N} . The FSM is shown in figure 3.9.

The Transition Matrix

The transition matrix \mathbf{T}_a is the following:

$$\mathbf{T}_a = \begin{pmatrix} 0 & 1 & 0 & \dots & & & & 0 \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ 0 & \dots & 0 & 1 & 0 & \dots & & 0 \\ p_0 & 0 & \dots & 0 & p_1 & \dots & p_{N-1} & p_N \\ 1 & 0 & \dots & & & & & 0 \\ 0 & \dots & & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & & & & & \epsilon_1 & 1 - \epsilon_1 \end{pmatrix}$$

The Stationary PMF

Again, for $p_i > 0 \forall i \in \{0, 1, \dots, N\}$ and $\epsilon_1 > 0$, the PMF over the states converges to the stationary PMF \mathbf{p}_1 . In figure 3.10, you can see a comparison between the stationary PMF of the truncated Poisson-like distribution and the stationary PMF of its approximation.

3. Modeling Firing Time Statistics

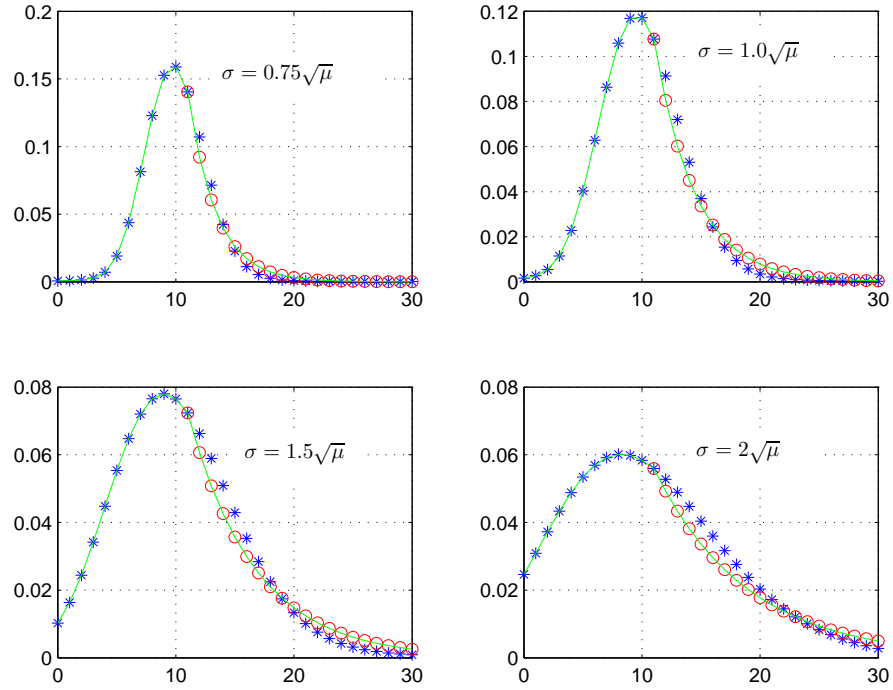


Figure 3.8.: The approximated PMF of the truncated Poisson-like distribution for $\mu = 10$, $N = 30$ and different values of σ . The PMF of the truncated Poisson-like distribution is plotted by $*$, the PMF of the approximation is plotted by $-$ and the part which is approximated by a geometric PMF is indicated by \circ .

3.3. Conclusion and Outlook

I hope that I could give a theoretical fundament for the modeling of firing time statistics in this chapter. The big challenge remains:

- How can we construct a FSM which implements a certain distribution with as few states as possible?

I could do a first step by approximating the PMF of a Poisson distribution by the PMF of a geometric distribution in the decreasing part. For increasing parts of PMF's I have the following conjecture:

Conjecture 1. *It is impossible to approximate an increasing part of length L of a certain PMF by a FSM with less than L states without allowing the approximating PMF to decrease locally.*

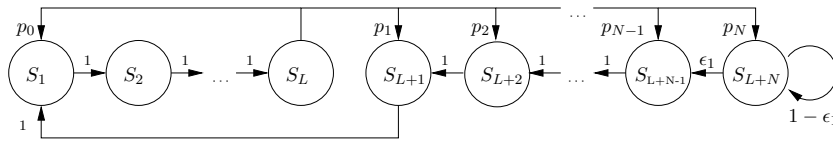


Figure 3.9.: The FSM for the approximation of the truncated Poisson-like distribution.

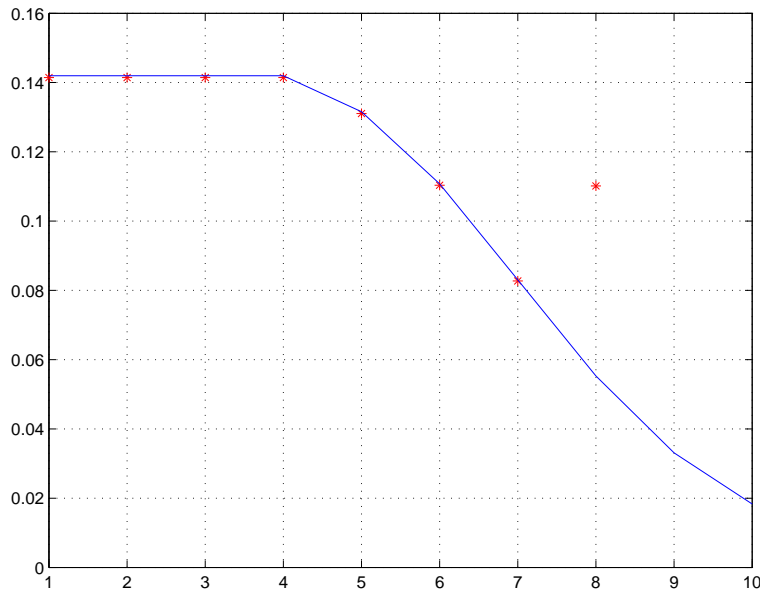


Figure 3.10.: The stationary PMF's of the FSM's from figure 3.6 and 3.9. - is from the FSM which implements the truncated Poisson-like distribution and * is from the FSM which implements the approximation of the former. As you can see, the part of - on the right of S_7 is cumulated in the value of * at S_8 .

3. Modeling *Firing Time Statistics*

4. Implementing Firing Time Statistics

In this section, I will talk about the implementation of the models developed in the previous chapter into the factor graph.

In table 4.1, I listed the notation I use in this section. In my description, I will try to stay as local in the factor graph as possible to avoid too much indices. For example, I notate the actual sample k only when going in horizontal direction through the factor graph and I don't say which Mu I am talking about since the following applies to all Mus in the same way.

Auxiliary random variables	U, R
Channel	c
Coefficient node of the Mu in channel c	cff_c
Constants	A
Counter	i
Emg signal	$y[\cdot]$
Fire node	Fire
Length of the Muap as seen in channel c	L_c
Maximum length of the Muap	L
Messages	$\mu \rightarrow$
Number of channels	n
Number of idle states in the used model	N
Random variable for the states of the model at sample k	$S[k]$
Random variable for the states of the Mu	V
Random variable for the states of the Mu in channel c	$V^{(c)}$
Source node	Src
Transition matrices	$\mathbf{T}, \mathbf{T}_p, \mathbf{T}_a$

Table 4.1.: Notation in chapter 4

4.1. Introduction

If you are not familiar with factor graphs, you should read [7] first. My notation of for example messages is taken from there.

4.1.1. The Source Node

In figure 4.1, you can see the factor graph for sample k with 2 Mu's and one channel. The source node is labeled Src. It has the following tasks:

4. Implementing Firing Time Statistics

- From port 1 to port 2, it applies the transition from the PMF of the states in sample $k - 1$ to the PMF of the states in sample k according to the model it was initialized with.
- At the ports $3..3 + n$ it tells the coefficient node how probable which state is.
- At port 0, it provides a PMF over the possible states of the Mu at sample k . This PMF is in the end of the decomposition algorithm used to estimate the firing times. How this can be done is described in [3].

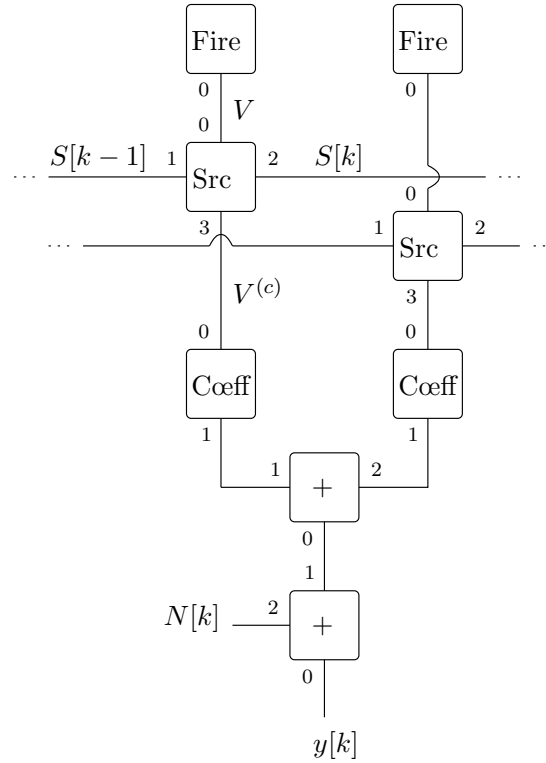


Figure 4.1.: The factor graph for sample k with 2 MUs and one channel. The node called Src is implemented in the class `EmgNodeSource`

4.1.2. Messages towards the Coefficient Node

Each coefficient node cff_c maintains a PMF over L_c active states and 1 idle state. The dimension of the messages towards the coefficient node is therefore

$$\dim(\mu_{\text{Src} \rightarrow V^{(c)}}(V^{(c)})) = L_c + 1$$

Note that L_c can be different for the same Mu but different channels c . The calculation of the messages depends on the model the source node was initialized with. I will discuss this later in this section.

4.1.3. Messages towards the Fire Node

A Mu is active when it is active in at least one channel and it is idle when it is idle in every channel. So the number of states of the Mu is equal to the number of states it has in the channel where it has the longest Muap. With this, the dimension of the messages toward the fire node becomes

$$\dim(\mu_{\text{Src} \rightarrow V}(V)) = L + 1 = \underset{c}{\operatorname{argmax}} L_c + 1$$

4.1.4. Messages towards the next Source Node

The step from $\text{Src}[k - 1]$ to $\text{Src}[k]$ corresponds to the calculation of the PMF $\mathbf{p}[k]$ of the states of the model given the PMF $\mathbf{p}[k - 1]$. Hence the dimension of the messages between the source nodes depends on the number of states used in the model the source nodes were initialized with:

$$\dim(\mu_{\text{Src} \rightarrow S[k]}(S[k])) = L + N$$

4.2. Class *EmgFnSource*

Our factor graph is implemented using the `jsim` package which was developed in [13]. Internally, a `DataNode` is initialized with a `DataNodeType` which itself is initialized with a `NodeFunction`. I will only talk about the implementation of the function which works on the data of the source node `EmgNodeSource`.

This function is implemented by the abstract superclass `EmgFnSource` with the subclasses `EmgFnSourceExponential` and `EmgFnSourcePoisson`. The former was introduced in [2] and implements the geometric distribution as described in section 3.2.1.

The latter was proposed in [16] and based on that, I implemented the class `EmgFnSourcePoisson` which realizes the truncated Poisson-like distribution and its approximation as defined in section 3.2. The class itself has an internal network, but is not assigned exclusively to a single `EmgNodeSource`, as is normally done in the `jsim` package. I did this to save memory.

I tried in figure 4.2 to visualize how the components work together. In the next section, I will explain in detail how the class `EmgFnSourcePoisson` works on the data.

4.3. Class *EmgFnSourcePoisson*

I will now give the update rules for the messages of the internal network according to the summary propagation algorithm as described in [7]. I believe that it is best to represent the calculation of messages by matrix notation, so my notation can differ from the one you are used to.

The internal nodes I will identify by their numbers as defined in figure 4.3. The random variable represented by the edge between 4 and 5 I will call U and the random variable represented by the edge between 4 and 3 I will call R . For the other edges I take the variable names of the external edges as defined in table 4.1.

4. Implementing Firing Time Statistics

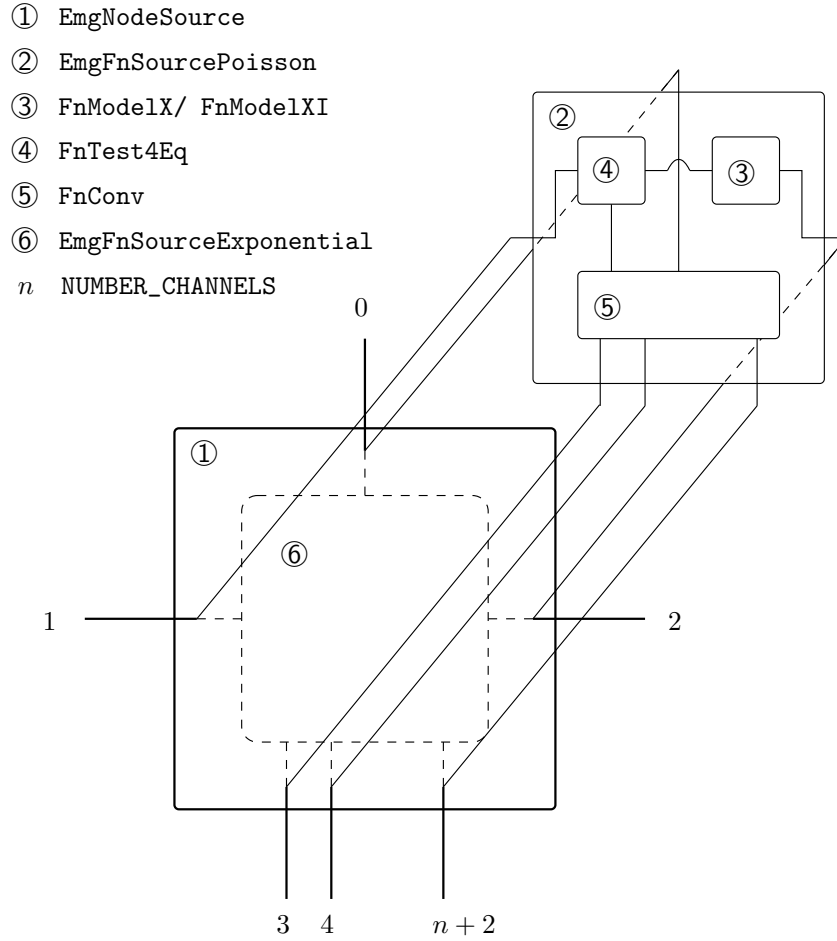


Figure 4.2.: How EmgFnSourcePoisson and EmgFnSourceExponential work on EmgNodeSource

4.3.1. Messages from Transition Node FnModelX

In this node, the truncated Poisson-like distribution is implemented as a statistics model. It has L active states and N idle states, which implies, as I mentioned before, that the messages depending on either $S[k-1]$, $S[k]$, U or R are all of dimension $L+N$. The message rules are as follows:

port 0

$$\mu_{3 \rightarrow R}(R) = \mu_{S[k] \rightarrow 3}(S[k]) \mathbf{T}_p^T$$

where \mathbf{T}_p^T is the transposed of the transition matrix \mathbf{T}_p as defined in 3.2.2 (If you are surprised by the transposed, one hint: When you go from step k to step $k-1$ in a FSM, the possible transitions and their direction stay the same, but you are going back in **time**: The probability that I'm here is the probability that I will

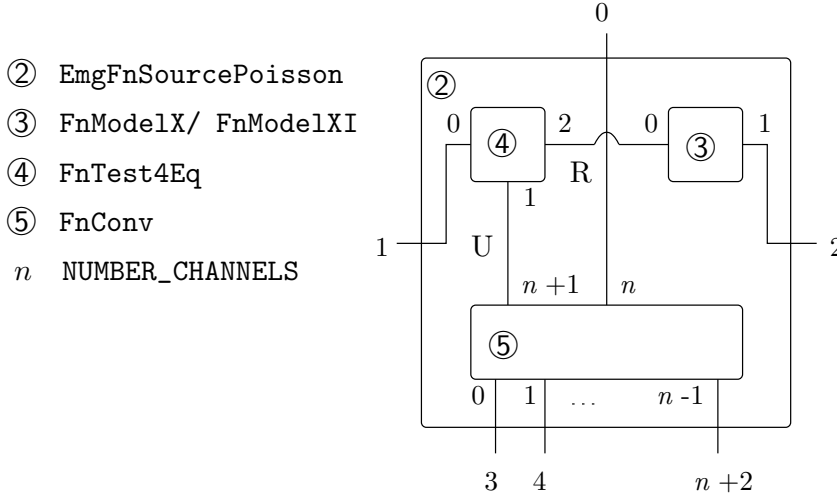


Figure 4.3.: The internal network of *EmgFnSourcePoisson*. You can see the mapping from the internal port numbers to the external port numbers.

go to a certain state times the probability that I will be there added up over all possible states I can go).

port 1

$$\mu_{3 \rightarrow S[k]}(S[k]) = \mu_{R \rightarrow 3}(R) \mathbf{T}_p$$

where \mathbf{T}_p is the transition matrix as defined in 3.2.2.

4.3.2. Messages from Transition Node *FnModelXI*

The message calculation is almost the same as for node *FnModelX*, only the transition matrix changes to \mathbf{T}_a as defined in 3.2.3.

port 0

$$\mu_{3 \rightarrow R}(R) = \mu_{S[k] \rightarrow 3}(S[k]) \mathbf{T}_a^T$$

port 1

$$\mu_{3 \rightarrow S[k]}(S[k]) = \mu_{R \rightarrow 3}(R) \mathbf{T}_a$$

4.3.3. Messages from Equal Node *FnTest4Eq*

FnTest4Eq is a standard equal node with 3 ports. The message rules are as follows, where \bullet denotes elementwise multiplication of two vectors:

port 0

$$\mu_{4 \rightarrow S[k-1]}(S[k-1]) = \mu_{U \rightarrow 4}(U) \bullet \mu_{R \rightarrow 4}(R)$$

4. Implementing Firing Time Statistics

port 1

$$\mu_{4 \rightarrow U}(U) = \mu_{S[k-1] \rightarrow 4}(S[k-1]) \bullet \mu_{R \rightarrow 4}(R)$$

port 2

$$\mu_{4 \rightarrow R}(R) = \mu_{S[k-1] \rightarrow 4}(S[k-1]) \bullet \mu_{U \rightarrow 4}(U)$$

4.3.4. Messages from Conversion Node FnConv

As I stated before, the messages depending on $V^{(c)}$ are of dimension $L_c + 1$. It is important to notify that for some “historical” reasons, the state $V_1^{(c)}$ corresponds to the idle state, whereas the states $V_2^{(c)} \dots V_{L_c+1}^{(c)}$ correspond to the active states, which differs from the way I defined it in section 3.2. I will now give the message rules.

port $n + 1$

$$\mu_{5 \rightarrow U}(U) = \left(\prod_{c=1}^n \mu_{V^{(1)} \rightarrow 5}(V^{(i)}) \mathbf{T}_c \right) \bullet (\mu_{V \rightarrow 5}(V) \mathbf{T}_V)$$

where \prod means elementwise vector multiplication. \mathbf{T}_c and \mathbf{T}_V are matrices defined in the following way:

$$\mathbf{T}_c = \begin{pmatrix} 0 & \dots & & 0 & A_c & \dots & A_c \\ 1 & 0 & \dots & & & & 0 \\ 0 & \ddots & \ddots & & & & \vdots \\ \vdots & \ddots & & & & & \\ 0 & & 0 & 1 & 0 & \dots & 0 \end{pmatrix}$$

$\underbrace{\hspace{10em}}_{L_c} \quad \underbrace{\hspace{10em}}_{L+N-L_c}$

This matrix maps the active states of $V_2^{(c)} \dots V_{L_c+1}^{(c)}$ to the active states $U_1 \dots U_{L_c}$ and the idle state $V_1^{(c)}$ to the active states $U_{L_c+1} \dots U_L$ and the idle states $U_{L+1} \dots U_{L+N}$. In [16] Christian Zimmermann chose $A_c = 1$, but I would rather propose $A_c = \frac{1}{N}$ and I would argue in the following way: Consider the case where we have only one channel, so $L_1 = L$. From the message $\mu_{5 \rightarrow U}(U)$, the probability that the Mu is active is

$$P(\text{Mu is active}) = \sum_{i=1}^L \mu_{5 \rightarrow U}(U_i)$$

And the probability that the Mu is idle is

$$P(\text{Mu is idle}) = \sum_{i=L+1}^{L+N} \mu_{5 \rightarrow U}(U_i)$$

From the message $\mu_{V(c) \rightarrow 5}(V^{(1)})$ the probability that the Mu is active is

$$P(\text{Mu is active}) = \sum_{i=2}^{L+1} \mu_{V^{(1)} \rightarrow 5}(V_i^{(1)})$$

and the probability that the Mu is idle is

$$P(\text{Mu is idle}) = \mu_{V^{(1)} \rightarrow 5}(V_1^{(1)})$$

If we want the conversion node to maintain this relation, we should have

$$\sum_{i=2}^{L+1} \mu_{V^{(1)} \rightarrow 5}(V_i^{(1)}) = \sum_{i=1}^L \mu_{5 \rightarrow U}(U_i)$$

which is fulfilled for $L_1 = L$ because of the definition of \mathbf{T}_c . For $N > 0$, we can develop the condition for $P(\text{Mu is idle})$ in the following way:

$$\begin{aligned} \sum_{i=L+1}^{L+N} \mu_{5 \rightarrow U}(U_i) &= \mu_{V^{(1)} \rightarrow 5}(V_1^{(1)}) \\ &\iff \\ \sum_{i=L+1}^{L+N} A_c \cdot \mu_{V^{(1)} \rightarrow 5}(V_1^{(1)}) &= \mu_{V^{(1)} \rightarrow 5}(V_1^{(1)}) \\ &\iff \\ N A_c \cdot \mu_{V^{(1)} \rightarrow 5}(V_1^{(1)}) &= \mu_{V^{(1)} \rightarrow 5}(V_1^{(1)}) \\ &\iff \\ A_c &= \frac{1}{N} \end{aligned}$$

□

When decomposing real signals, N can be around values of 4000 for the truncated Poisson-like distribution and around 2000 for its approximation, so the impact of $\frac{1}{N}$ is fundamental. This should be fixed. I didn't because all my results base on the implementation with $A_c = 1$. The problem is that when you change A , you have to be carefull when you want to initialize a node with neutral messages. Currently, the conversion node maps messages from the lower part of the factor graph with entries all set to the same value to messages in the upper part of the factor graph with all entries being of equal value too, though the two messages represent in fact completely different distributions if you only differ between active and idle.

In the following, we will implicitly have similar problems, but they are not fundamental since the factor is only around the maximum difference of the lengths of two Muaps of the same Mu in different channels.

4. Implementing Firing Time Statistics

port 0..n-1 The message from 5 towards $V^{(c)}$, $c \in \{1, \dots, n\}$ can be written in the following way:

$$\mu_{5 \rightarrow V^{(c)}}(V^{(c)}) = \left(\prod_{\substack{1 \leq i \leq n \\ i \neq c}} \mu_{V^{(i)} \rightarrow 5}(V^{(i)}) \mathbf{T}_{ic} \right) \bullet (\mu_{U \rightarrow 5}(U) \mathbf{T}_c^T) \bullet (\mu_{V \rightarrow 5}(V) \mathbf{T}_{vc})$$

\mathbf{T}_c^T is the transposed of the matrix \mathbf{T}_c defined before and \mathbf{T}_{ic} can have three shapes:

case 1: $L_c > L_i$.

$$\mathbf{T}_{ic} = \left(\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & 1 & \dots & 1 & \\ 0 & \ddots & \ddots & & 0 & \dots & 0 & \\ \vdots & \ddots & & & \vdots & & \vdots & \\ 0 & & & 0 & 1 & 0 & \dots & 0 \end{array} \right)$$

$\underbrace{\hspace{10em}}_{L_i} \quad \underbrace{\hspace{10em}}_{L_c - L_i}$

case 2: $L_c < L_i$.

$$\mathbf{T}_{ic} = \left(\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & 1 & \dots & 1 & \\ 0 & \ddots & \ddots & & 0 & \dots & 0 & \\ \vdots & \ddots & & & \vdots & & \vdots & \\ 0 & & & 0 & 1 & 0 & \dots & 0 \end{array} \right)^T$$

$\underbrace{\hspace{10em}}_{L_c} \quad \underbrace{\hspace{10em}}_{L_i - L_c}$

case 3: $L_c = L_i$.

$$\mathbf{T}_{ic} = \left(\begin{array}{cc} 1 & 0 \\ & \ddots \\ 0 & 1 \end{array} \right)$$

$\underbrace{\hspace{10em}}_{L_c}$

port n The message $\mu_{5 \rightarrow V}(V)$ can be defined in exactly the same way as the messages $\mu_{5 \rightarrow V^{(c)}}(V^{(c)})$.

4.3.5. The Update Schedule for the Internal Network

When the external ports of the `EmgNodeSource` have to be updated, the `EmgFnSourcePoisson` node connects its internal network to the `EmgNodeSource`. It then has to update its internal network, which is done automatically. The update schedules are as follows:

port 1 of node EmgNodeSource

1. Update port 0 of node `FnModelIX/ FnModelIXI`

4.4. Verifying *EmgFnSourcePoisson* by its Convergence

2. Update port $n + 1$ of node `FnConv`
3. Update port 0 of node `FnTest4Eq`

port 2 of node `EmgNodeSource`

1. Update port $n + 1$ of node `FnConv`
2. Update port 2 of node `FnTest4Eq`
3. Update port 1 of node `FnModelX/ FnModelXI`

port 0 of `EmgNodeSource`

1. Update port 0 of node `FnModelX/ FnModelXI`
2. Update port 1 of node `FnTest4Eq`
3. Update port n of node `FnConv`

port $i \in \{3, \dots, n + 2\}$ of `EmgNodeSource`

1. Update port 0 of node `FnModelX/ FnModelXI`
2. Update port 1 of node `FnTest4Eq`
3. Update port $i - 3$ of node `FnConv`

4.4. Verifying *EmgFnSourcePoisson* by its Convergence

To check if I implemented the node function in the right way, I initialized a `EmgNodeSource` with it. The ports 0 and 3 of the `EmgNodeSource` I connected to constant messages and the ports 1 and 2 I connected to each other. To test the transition which the function node `EmgFnSourcePoisson` performs, I updated port 2 around 100000 times and checked then the out message on port 2.

One comment on the high number of iterations: When you have a 2000×2000 transition matrix, it can need almost 2000 iteration until an information gets from the left of the vector representing the message to the right. So don't be surprised by strange results from iterating 100 times to calculate the stationary PMF of a 2000×2000 transition matrix.

The result is displayed in figure 4.4. As you can see, the results are as we expected, compare with the theoretical results we obtained in section 3.2. As indicated in the figure, the probabilities of the states 1101 to 2000 from the FSM of the truncated Poisson-like distribution are cumulated on state 1101 of the FSM of the approximation. The values are slightly different because the mean and the standard deviation of the truncated Poisson-like distribution and its approximation are not exactly the same, as stated before in section 3.2.3.

4. Implementing Firing Time Statistics

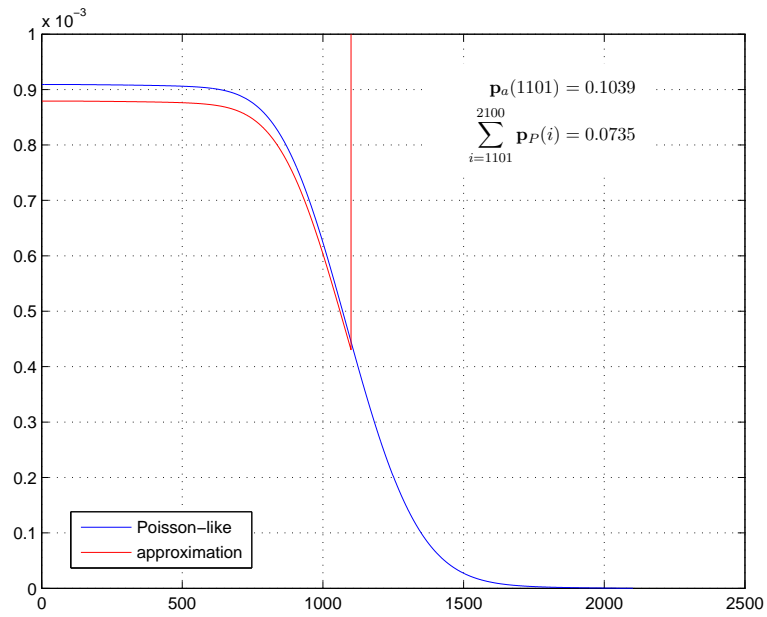


Figure 4.4.: The stationary PMF obtained for $\mu = 1000$, $\sigma = 6 \cdot \sqrt{\mu}$ and `numberOfIdleStates` equal 2000 for the truncated Poisson-like distribution (blue) and `numberOfIdleStates` equal 1001 for the approximation. The length of the Muap was set equal 100.

4.5. Results

The Motivation for the implementation of a better statistic model was originally of theoretical kind: It wasn't acceptable to model the firing time statistics with an exponentially decreasing curve, since this is far away from reality. However, we never had really problems with the former model, and it is very difficult to find examples where the new model performs better than the old one.

The model with the truncated Poisson-like distribution produces rather problems. Initialized with the wrong values, it has a bad impact onto the results, as can be seen in figure 4.5. It is interesting to note that the approximation of the truncated Poisson-like distribution doesn't make us these problems, because its PMF is larger than zero for all possible distances between two firings, ∞ included.

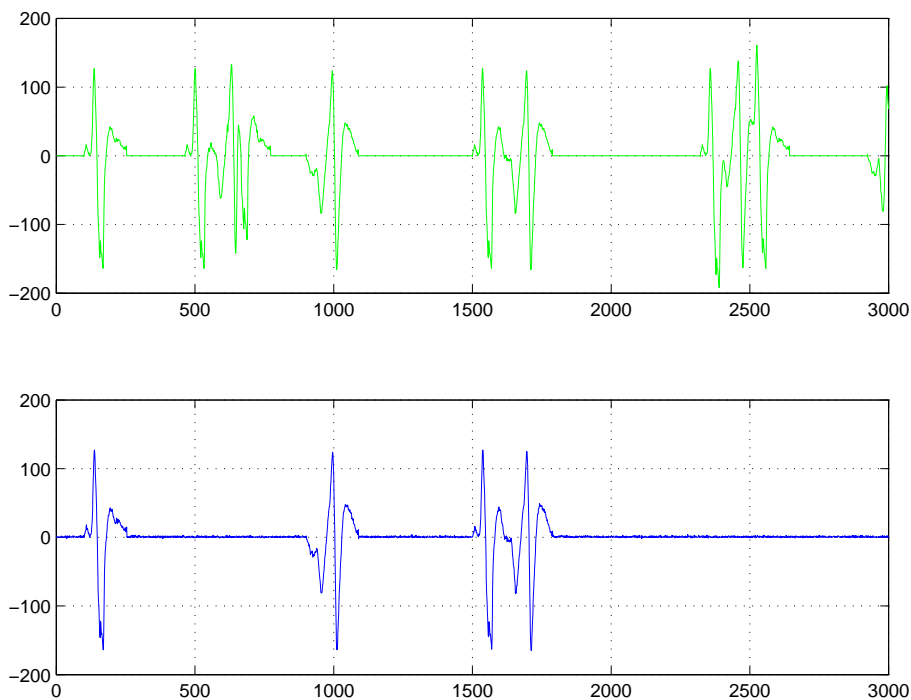


Figure 4.5.: Decomposition results when using the truncated Poisson-like distribution with wrong parameters.

When trying to decompose a noisy signal with lots of artefacts, the impact of the statistics model can help. In the figures 4.6 and 4.7 you see the results from trying to decompose a real signal with artefacts. Both results are quite poor, but the wrong firing of the upper μ between 1800 and 2000 could be avoided by using the approximated truncated Poisson-like distribution with the parameters $\mu = 1850$ and $\sigma = 12 \cdot \sqrt{\mu}$. The corresponding PMF is shown in figure 4.8 to illustrate that the variance was high enough to allow firings at any place in the signal.

4. Implementing Firing Time Statistics

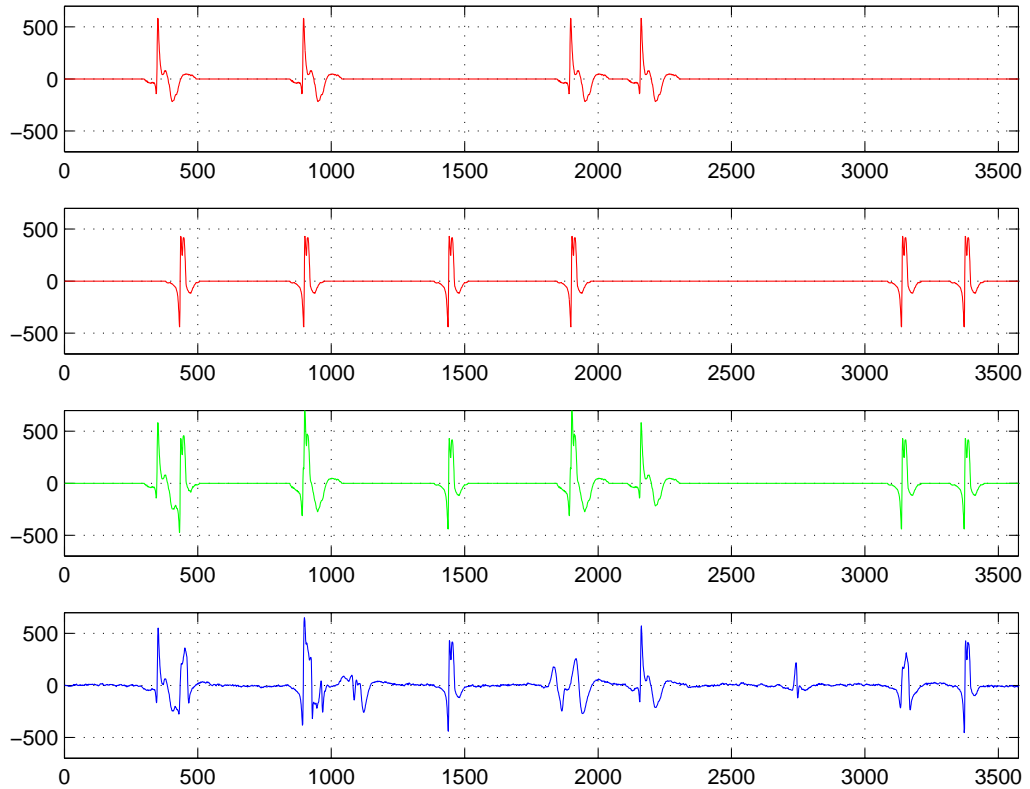


Figure 4.6.: Decomposition results using the geometric distribution.

For the second Mu however, there is no difference between the geometric and the approximated truncated Poisson-like distribution. Decomposition with lower values for the variance of the distribution didn't improve the results, so probably the firings of the second Mu come from the fact that its Muap is very similar to the artefacts (other Muaps?) which can be found in the signal.

4.6. Conclusion and Outlook

- The geometric distribution worked well so far and needs only one idle state.
- The truncated Poisson-like distribution has a too strong impact onto the decomposition results when initialized with a realizable number of idle states.
It needs a lot of memory because of the large number of states, a problem which is fundamental and not solvable by code optimization.
- The approximation of the truncated Poisson-like distribution leads to the best results so far. It needs however still a lot of memory.
- The main goal should be to minimize the number of states without losing the

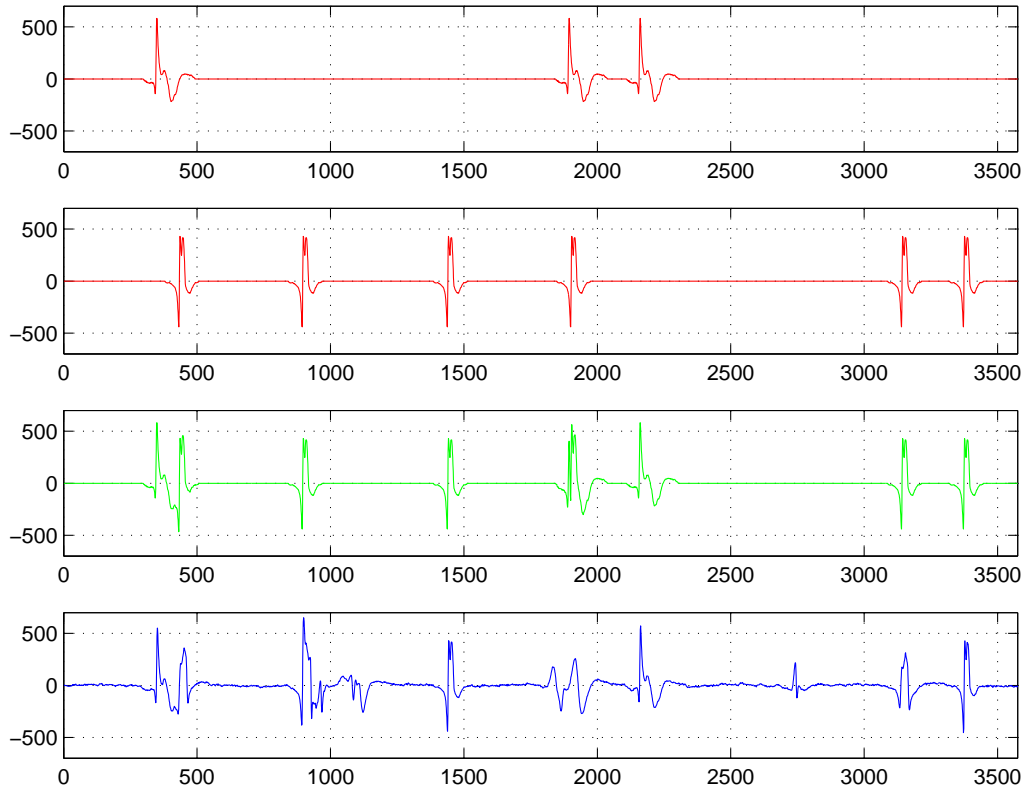


Figure 4.7.: Decomposition results using the approximation of the truncated Poisson-like distribution.

profits we have from modeling the reality in a correct way. This is a challenging task, especially when the conjecture I made in 3.3 turns out to be true.

- I believe that the approximation of the truncated Poisson-like distribution is still much too accurate. Approximating the increasing part from the left to the right successively by geometrical distributions will lead to an approximation looking like a sawtooth, but it should still work fine in our decomposition algorithm.
- When designing statistics models, remember that the lower part of the factor graph sees only the sum of the probabilities of the idle states. If you can guarantee that this value is always reasonable, you can do almost whatever you want in the upper part of the factor graph. The sawtooth mentioned above should therefore not be of any problem.

4. Implementing Firing Time Statistics

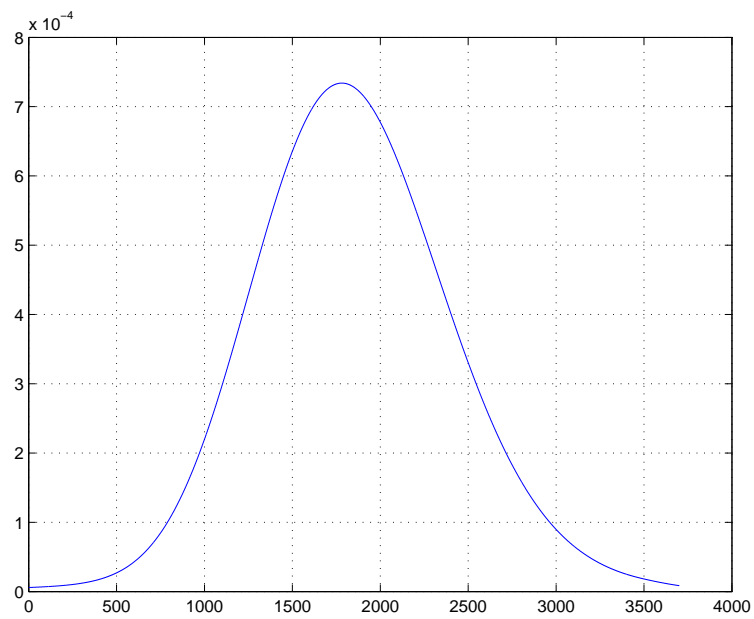


Figure 4.8.: The PMF of the truncated Poisson-like distribution with $\mu = 1850$ and $\sigma = 12 \cdot \sqrt{\mu}$.

5. Adapting A Priori Models by Iteration

In this chapter, I will develop the idea how we can find the correct values for the parameters of our a priori models by iteration. The basic idea is an iteration between a decomposition based on the current model and an adaption of the model by evaluating the results from this decomposition.

5.1. The Expectation Maximization Algorithm

There are a lot of different formulations of the EM algorithm for different classes of probability distributions. I will formulate it as was done in [4], since it is the most general formulation I could find.

5.1.1. The General Case

Suppose we have a function $f(x, \theta) \geq 0 \forall x, \theta$. We assume that the marginalization¹ $\int_x f(x, \theta)$ exists $\forall x, \theta$. We now want to find

$$\theta_{\max} = \operatorname{argmax}_{\theta} \int_x f(x, \theta) \quad (5.1)$$

To be able to formulate the algorithm in terms of probability theory, we define the probability distribution

$$p(x|\hat{\theta}) = \frac{f(x, \hat{\theta})}{\int_x f(x, \hat{\theta})}$$

Since we do not know x , we try to maximize the expectation value of $f(x, \theta)$. \log is strictly monotonic, so we can use the identity

$$\operatorname{argmax}_{\theta} f(x, \theta) = \operatorname{argmax}_{\theta} \log f(x, \theta)$$

and maximize the expectation value of $\log f(x, \theta)$ instead. The algorithm then becomes

$$E_p^{(n)}[\log f(x, \theta)] = \int_x \log f(x, \theta) p(x|\hat{\theta}_n) \quad (5.2)$$

$$\theta_{n+1} = \operatorname{argmax}_{\theta} E_p^{(n)}[\log f(x, \theta)] \quad (5.3)$$

We initialize this procedure with a suitable guess value θ_0 and continue until convergence. The convergence of the EM can actually be proven. You can find a proof in [12] and [9].

¹This might be an integral or a sum, depending on x taking continuous or discrete values

5. Adapting A Priori Models by Iteration

When you look at the general formulation of the EM algorithm, it is not clear why it should be useful. If you compare the equations 5.1 and 5.2, it is not obvious why the second one should be more easy to calculate than the first one. Let us therefore consider an example.

5.1.2. An Example

There exist a lot of variations of this problem. It was probably used for the first time in [1]. You can also find it in [9] as an introducing example. I reformulate it in terms of signal processing:

Let $Y[k]$ be an iid random process with the following probabilities:

$$\begin{aligned} P(Y[k] = -1) &= \frac{1}{4} \\ P(Y[k] = 0) &= \left(\frac{1}{4} + \frac{\theta}{4}\right) \\ P(Y[k] = 1) &= \left(\frac{1}{2} - \frac{\theta}{4}\right) \end{aligned} \tag{5.4}$$

The symbols sent by a transmitter can be modelled by this process. Assume that our receiver can't distinguish between -1 and 1 when observing the process, so he sees $|Y[k]|$. We now have a set of observations of N samples of our random process. We know that m_1 samples were either -1 or 1 and we know that m_2 samples were 0 . We introduce the random variable X which denotes the number of sent -1 's. We get the PMF

$$p(x|\theta) = \binom{N}{m_2} \left(\frac{1}{4} + \frac{\theta}{4}\right)^{m_2} \binom{m_1}{x} \left(\frac{1}{4}\right)^x \left(\frac{1}{2} - \frac{\theta}{4}\right)^{m_1-x} \tag{5.5}$$

As I said before, the example is not new. However in literature, it is always used as an introducing example with the EM algorithm formulated in a way adapted to this class of problems. I couldn't find a derivation from the general formulation to the one adapted to the problem, so I derived it by myself.

5.1. The Expectation Maximization Algorithm

I develop (5.3):

$$\begin{aligned}
\hat{\theta}_{n+1} &= \operatorname{argmax}_{\theta} \int_x \log p(x, \theta) \frac{p(x, \hat{\theta}_n)}{\int_x p(x, \hat{\theta}_n)} \\
&= \operatorname{argmax}_{\theta} \int_x \log p(x, \theta) p(x, \hat{\theta}_n) \\
&= \operatorname{argmax}_{\theta} \sum_{x=0}^{m_1} \log \left[\binom{N}{m_2} \left(\frac{1}{4} + \frac{\theta}{4}\right)^{m_2} \binom{m_1}{x} \left(\frac{1}{4}\right)^x \left(\frac{1}{2} - \frac{\theta}{4}\right)^{m_1-x} \right] p(x, \hat{\theta}_n) \\
&= \operatorname{argmax}_{\theta} \sum_{x=0}^{m_1} \log \left[\left(\frac{1}{4} + \frac{\theta}{4}\right)^{m_2} \left(\frac{1}{2} - \frac{\theta}{4}\right)^{m_1-x} \right] p(x, \hat{\theta}_n) \\
&= \operatorname{argmax}_{\theta} \sum_{x=0}^{m_1} \left(m_2 \log \left(\frac{1}{4} + \frac{\theta}{4}\right) + (m_1 - x) \log \left(\frac{1}{2} - \frac{\theta}{4}\right) \right) p(x, \hat{\theta}_n) \\
&= \operatorname{argmax}_{\theta} \sum_{x=0}^{m_1} m_2 \log \left(\frac{1}{4} + \frac{\theta}{4}\right) p(x, \hat{\theta}_n) \\
&\quad + \sum_{x=0}^{m_1} m_1 \log \left(\frac{1}{2} - \frac{\theta}{4}\right) p(x, \hat{\theta}_n) \\
&\quad - \sum_{x=0}^{m_1} x \log \left(\frac{1}{2} - \frac{\theta}{4}\right) p(x, \hat{\theta}_n) \\
&= \operatorname{argmax}_{\theta} m_2 \log \left(\frac{1}{4} + \frac{\theta}{4}\right) + m_1 \log \left(\frac{1}{2} - \frac{\theta}{4}\right) - \mathbb{E}[X|\hat{\theta}_n] \log \left(\frac{1}{2} - \frac{\theta}{4}\right) \\
&= \operatorname{argmax}_{\theta} \left(\frac{1}{4} + \frac{\theta}{4}\right)^{m_2} \left(\frac{1}{2} - \frac{\theta}{4}\right)^{m_1 - \mathbb{E}[X|\hat{\theta}_n]} \\
&= \operatorname{argmax}_{\theta} p(\mathbb{E}[X|\hat{\theta}_n]|\theta)
\end{aligned}$$

□

With this result, we can for this example formulate the EM Algorithm in a much more intuitive way:

$$\begin{aligned}
\mathbb{E}[X|\hat{\theta}_n] &= \sum_X xp(x|\hat{\theta}_n) \\
\theta_{n+1} &= \operatorname{argmax}_{\theta} p(\mathbb{E}[X|\hat{\theta}_n]|\theta)
\end{aligned} \tag{5.6}$$

The effect of the EM Algorithm on p can be seen in figure 5.1. The algorithm was initialized with $\hat{\theta}_1 = 0$, and the curves represent $p(x|\hat{\theta}_n)$ for $n = 1, 2, 3, 20$, with the curves moving to the right for increasing n .

5. Adapting A Priori Models by Iteration

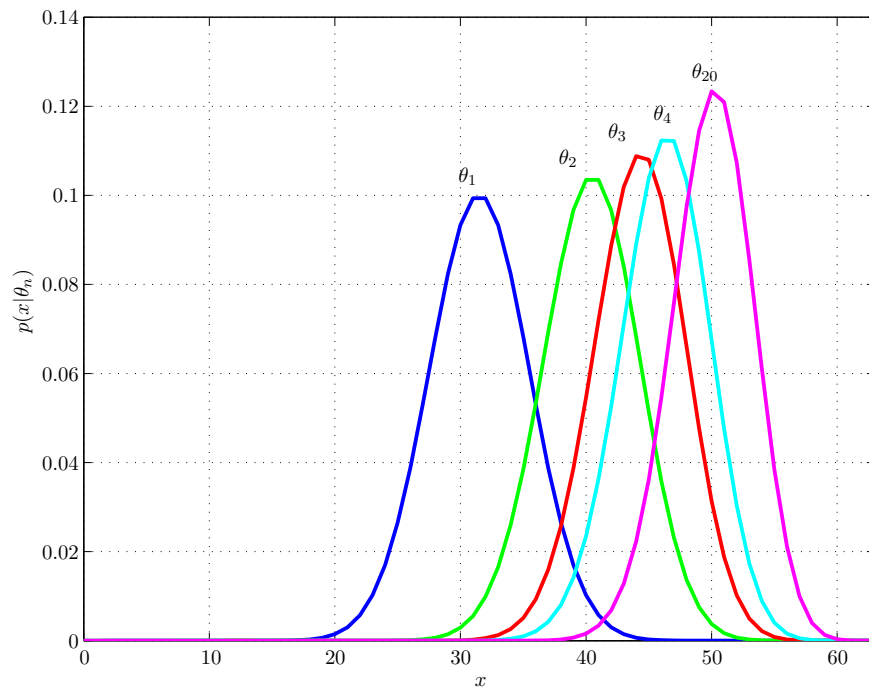


Figure 5.1.: EM algorithm: $N = 200$, $m_1=63$, number of iterations equal 20.

5.2. The Estimation Minimization Algorithm

Inspired by the EM algorithm I tried to formulate it for the decomposition of Emg signals, which leads to the adaption of the a priori model by iteration. Since the expectation maximization algorithm is formulated in strictly defined mathematical terms, it is difficult to use it directly in our decomposition: We make approximations, guesses and estimations at several places.

I tried therefore to find a more general formulation, which I will introduce in this section. In table 5.1 you find the notation I use in this section.

A priori model	θ
A priori model of firing time statistics	ϵ
Cost function	κ
Impulse train of firing times from results	\hat{X}
Impulse train of firing times from model	$X(\epsilon)$
Iteration step	n
Mu	s
Number of Mus	S
Parameters of ϵ	μ, σ
Signal length	K

Table 5.1.: Notation in section 5.2

5.2.1. The General Formulation

I propose the following algorithm, which I call the **estimation minimization** algorithm:

$$\hat{X} = \text{results from decomposition based on } \theta_n \quad (5.7)$$

$$\theta_{n+1} = \underset{\theta}{\operatorname{argmin}} \kappa(\hat{X}, \theta) \quad (5.8)$$

Equation (5.7) defines the estimation step and equation (5.8) defines the minimization step. κ denotes some cost function.

5.2.2. Formulation for the Adaption of the Firing Time Statistics

Suppose we have S different Mus. Let \hat{X}_s , $s \in \{1, \dots, S\}$ be the estimated impulse train representing the firing times of Mu s . The model we use for the firing time statistics is denoted by ϵ . I define the following cost function κ_s :

$$\kappa_s = \|X_s(\epsilon)\|^2 - \|\hat{X}_s\|^2$$

Note that since \hat{X}_s is a discrete impulse train with ones where the Mu fires and zeros otherwise, $\|\hat{X}_s\|^2$ is equal to the number of detected firings. $\|X_s(\epsilon)\|^2$ is an estimation of the number of firings we expect when using model ϵ . If the distribution of our model has

5. Adapting A Priori Models by Iteration

the expectation value μ , and if the length of our signal is K , we can estimate $\|X_s(\epsilon)\|^2$ by $\frac{K}{\mu}$. If we now can characterize our statistics model completely by the expectation value μ , we can formulate the minimization step as follows:

$$\begin{aligned}\mu_{n+1} &= \underset{\mu}{\operatorname{argmin}} \kappa(\hat{X}_s, \mu) \\ &= \underset{\mu}{\operatorname{argmin}} \|X_s(\mu)\|^2 - \|\hat{X}_s\|^2 \\ &\approx \underset{\mu}{\operatorname{argmin}} \frac{K}{\mu} - \|\hat{X}_s\|^2 \\ &= \frac{K}{\|\hat{X}_s\|^2}\end{aligned}$$

This is a very simple update rule for our model. In the next section I will give some results which I obtained by using this update rule to adapt the model of the firing time statistics by iteration.

5.3. Results From Adapting the Firing Time Statistics by Iteration

To test the estimation minimization algorithm, I initialized the model ϵ with the truncated Poisson-like distribution and tried to decompose signal 80. I initialized the model with $\mu = 300$, $\sigma = 70$ and `numberOfIdleStates` = 800 and used the following update rules for μ with signal length $K = 3000$ and number of Mus $S = 2$

$$\begin{aligned}\mu_{n+1} &= \frac{S \cdot K}{\sum_{s=1}^S \|\hat{X}_{s,n}\|^2} \\ \sigma_{n+1} &= 6.3 \cdot \sqrt{\mu_{n+1}}\end{aligned}\tag{5.9}$$

$$\text{numberOfIdleStates}_{n+1} = 2 \cdot \mu_{n+1}$$

The factor 6.3 is a value which worked well as a relation between μ and $\sqrt{\sigma}$. It is not fix and probably not the best. My update rule should perhaps be extended such that σ is adapted separately.

The figures 5.2 to 5.4 show the results. From one iteration to the next, the parameters of the Poisson-like distribution were adapted with the update rule from equation 5.9. If you look at the signal (blue curve) and compare it with the initial parameters of the distribution, you see that the parameters represent the statistics of the signal very poorly. As we would expect, this leads to wrong decomposition results as shown in figure 5.2. The update rule 5.9 takes these wrong results to adapt the statistic model. It is interesting that though the adaption is based on wrong results, it leads to a better model of the statistics, which in fact leads to a correct decomposition which can be seen in figure 5.3

(We neglect the wrong firing at the right side as usual). This effect can be explained in the following way: The decomposition is based on the a priori model of the firing time

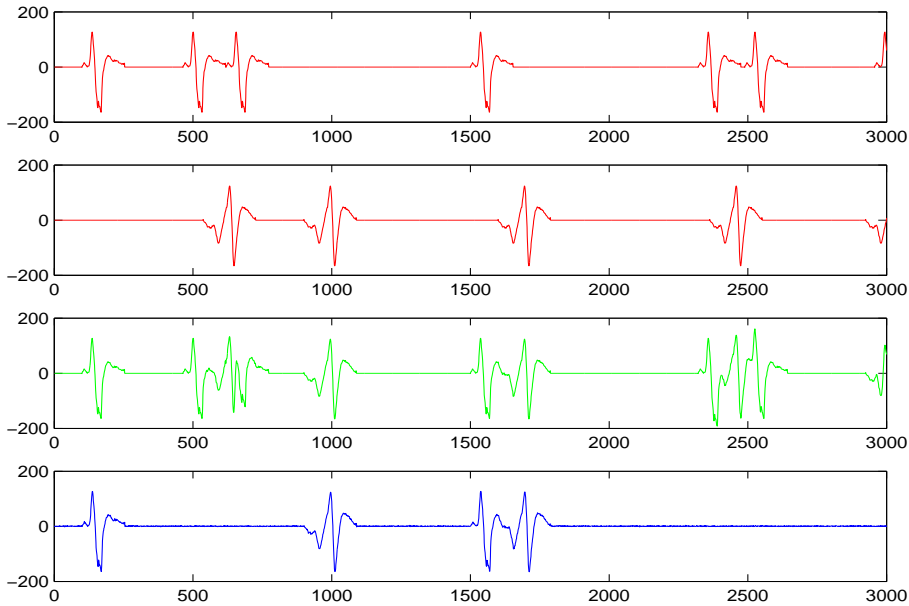


Figure 5.2.: **Iteration 1.** Results from decomposing Emg signal 80 (blue curve). The red curves are the detected Muapt's and the green curve is the superposition of the Muapt's. The truncated Poisson-like distribution was initialized with $\mu = 300$, $\sigma = 70$ and `numberOfIdleStates = 800`.

statistics ϵ and the measured signal y . The factor graph tries to satisfy both ϵ and y , which leads to a compromise: In the result, we have less firings than we would expect from our model but more than we actually can find in our signal. Adapted to this result, the gap between model and reality gets smaller and the result from the decomposition based on the new model gets better.

We get the same results in iteration 2 and iteration 3, so our algorithm converged since applying update rule 5.9 will not lead to new parameters for the distribution of our model.

5.4. Conclusion and Outlook

In [6], Volker Koch discusses the pro's and con's of a priori models for the firing time statistics. A priori models are problematic when they have a wrong impact onto the results which is the case when there is a difference between the model and the modeled part of the reality.

In this chapter, I elaborated a way how we can come from a poor model to a good model by iterating between decomposing a signal with the current model and then adapting the model by evaluating the obtained results. This procedure closes iteratively the gap between model and reality. I succeeded to apply this principle to the adaption of the firing time statistics.

5. Adapting A Priori Models by Iteration

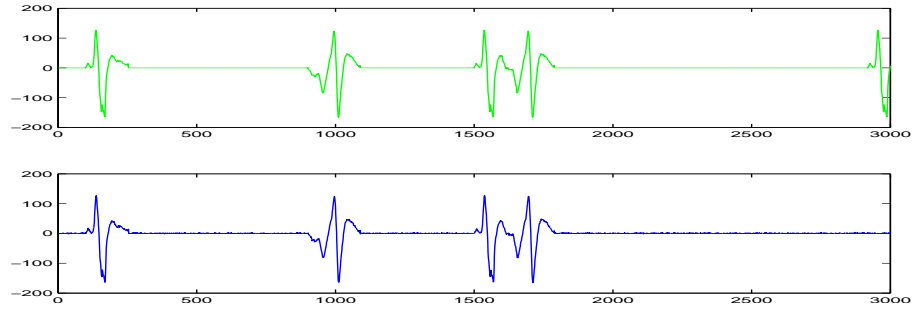


Figure 5.3.: **Iteration 2.** The distribution was adapted to $\mu = 666$, $\sigma = 162$, `numberOfIdleStates = 1332`

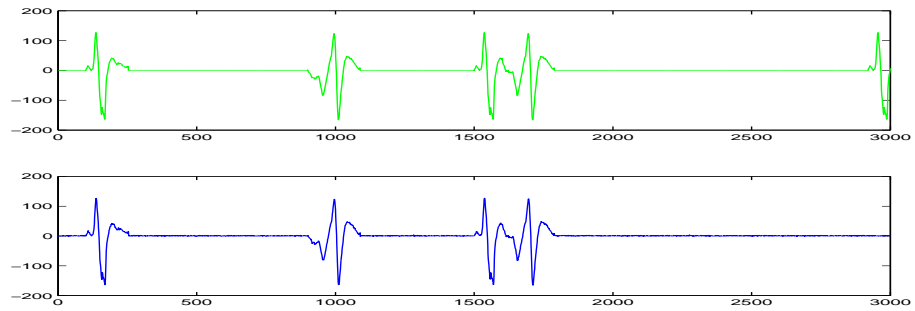


Figure 5.4.: **Iteration 3.** The distribution was adapted to $\mu = 1200$, $\sigma = 218$, `numberOfIdleStates = 2400`

The results are very encouraging and it should be tried to extend this principle to the adaption of other a priori models and parameters the decomposition is base on. The noise node introduced in [10] to model noise and artefacts for example is a good candidate to be adapted by the estimation minimization algorithm.

A. Implementations

A.1. The class `AprioriModelParameter`

Parameters in `Emg`

`aprioriModel` This reference has always to be initialized.

Parameters in `Emg.aprioriModel`

`type` With this parameter you define which a priori model should be used for the firing time statistics.

- `type = EXPONENTIAL` In this case the former model is used, which implements a geometric distribution. The other parameters are of no mean.
- `type = POISSON` In this case, the truncated Poisson-like distribution or its approximation is used, depending on `poissonApproximation`.

`mu` This parameter is the expectation value of the Poisson-like distribution. Remember that this it only an estimation of the expectation value of the truncated Poisson-like distribution or its approximation. You can find a reasonable value by looking at the signal you want to decompose and estimating the mean distance between 2 firings.

`sigma` This parameter is the standard deviation of the Poisson-distribution. In the formal Poisson-distribution, σ is equal $\sqrt{\mu}$. For μ around 2000, I made good experiences with $\sigma = 6.3\sqrt{\mu}$.

`poissonApproximation = false` In this case, you use the truncated Poisson-like distribution for the a priori model. Note that this model is very strong, since there is only a finite number of idle states. After some time, the Mu has to fire, which can lead to wrong results.

- `numberOfIdleStates` You should set this to around $2 \cdot \mu$ when you use the truncated Poisson-like distribution.

`poissonApproximation = true` In this case, you use the approximation of the truncated Poisson-like distribution. The main advantage is that you allow an infinite number of idle states, so you don't force the Mu to fire.

- `numberOfIdleStates` You should set this to at least $\mu + 1$ when you use the approximation.

A. Implementations

```
aprioriModel = new ParametersAprioriModel ();  
aprioriModel.type = POISSON;  
aprioriModel.mu = 1000;  
aprioriModel.sigma = 6.3*Math.sqrt(aprioriModel.mu);  
aprioriModel.poissonApproximation = false;  
aprioriModel.numberOfIdleStates = 2*aprioriModel.mu;
```

Table A.1.: The part of my current `Eng_0080.java` concerning the firing time statistics.

A.2. The class `AdaptionParameter`

I will explain the current features of the implemented adaption by giving some comments on the available parameters. In this way, you get an idea how you can profit best from the adaption by optimizing the parameters for a specific signal.

Parameters in `Emg`

`adaption` Set this to `true` to activate the adaption. If you do so, you also have to initialize `Emg.adaptionParameter`.

Parameters in `Emg.adaptionParameter`

`weighting` This is the most important parameter since it defines the type of adaption you apply. There are mainly three types of adaption:

- `weighting = {1};`
You take the latest realisation as a new Muap. This is reasonable when trend variability of the motor unit action potentials is dominant.
- `weighting = {1,1,1};`
You average over the last realisations. This is reasonable when random variability of the motor unit action potentials is dominant.
- `weighting = {3,2,1};`
You calculate the average with aging. Reasonable when there is a local random variability of the motor unit action potentials and global trend variability.

`smooth` Set this to `true` if you want the new Muaps to be filtered with `smoothFilter`. When there is significant noise present in the signal, you really should do it. You can specify your filter as follows:

- `smoothFilter = {0.1,...}`
You can set any filter of your choice to smooth the new Muaps. It has to be of odd length. The length of your filter should correspond to the sampling rate of your signal. For a Muap length of about 600 samples, I made good experiences with a gaussian filter generated in `matlab` with the command `gausswin(25)`.

`avoidSuperpositions` When you set this to `true`, only isolated firings (no superpositions) of the corresponding Mu are taken into account. This is reasonable when you don't want the new Muap of one Mu to be influenced by the changes of motor unit action potentials of other Mus.

`avoidNoiseSource` When you set this to `true`, those parts of the signal where the `noiseSourceNode` is active are not taken into account. This is reasonable when your detection is partly bad.

A. Implementations

maxDistanceOfRealisations When you set this to a positive value, you can avoid that much to old realisations are taken into account when you for example calculate new Muaps by averaging. A reasonable value is the blocklength (`Emg.MAX_WINDOW_LENGTH`). It can specifically be helpful when you exclude realisations by `avoidSuperpositions` or `avoidNoiseSource` because this increases the probability that you don't have new (valid) realisations over a long period of time (so those you have get older and older...). A negative value disables this feature.

capacityIsLimited When you set this to `true`, you can avoid the adaption from taking to much memory. This is reasonable when you process very long signals. You can specify two numbers:

- **maxNumberOfFirings** Only the given number of the latest firings are kept in memory.
- **maxNumberOfRealisations** Only the given number of the latest realisations are kept in memory.

setFILENAME_muapsFromAdaption(path/filename)

With this function you can set the folder and first part of the filename to which the calculated muaps for each block should be saved.

```
adaption = true;
adaptionParameter = new AdaptionParameter();

adaptionParameter.weighting = new double [] {1,1};

adaptionParameter.smooth = true;
adaptionParameter.smoothFilter = new double [] {
    0.1353, 0.6065, 1.0000, 0.6065, 0.1353};

adaptionParameter.avoidNoiseSource = false;
adaptionParameter.avoidSuperpositions = true;

adaptionParameter.maxDistanceOfRealisations = -1;
//disabled

adaptionParameter.capacityIsLimited = true;
adaptionParameter.maxNumberOfFirings = 5;
adaptionParameter.maxNumberOfRealisations = 5;
setFILENAME_muapsFromAdaption(getBasePath()
    + "results/resultsMuapsFromAdaption");
```

Table A.2.: The part of my current `Emg_0160.java` concerning the adaption.

Bibliography

- [1] D. B. Rubin A. P. Dempster, N. M. Laird. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1977.
- [2] Benjamin Amsler and Samuel Bruhin. Zerlegung von EMG-Signalen mittels graphischen Modellen und iterativen Algorithmen. Master's thesis, ETH Zurich, 2004.
- [3] Martin Byröd. Segmentation and decomposition of EMG signals. Semester thesis, ETH Zurich, 2004.
- [4] Loeliger H.-A. Dauwels J., Korl S. Expectation maximization as message passing. Proc. 2005 IEEE Int. Symp. on Inform. Theory, Sept. 2005.
- [5] V.M. Koch and H.-A. Loeliger. Decomposition of electromyographic signals by iterative message passing in a graphical model. In *Proc. of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE EMBC 2004)*, pages 65–68, 2004.
- [6] Volker Koch. Decomposition of electromyographic signals. a big-picture view. Talk, 2005.
- [7] Hans-Andrea Loeliger. An introduction to factor graphs. *IEEE Signal Processing Mag.*, Jan. 2004.
- [8] Hans-Andrea Loeliger. Stochastische Modelle und Signalverarbeitung, 2004/ 2005.
- [9] T.K. Moon. The expectation-maximization algorithm. *IEEE Trans Signal Processing*, 1996.
- [10] Johan Peterson. Estimation of filters for electromyographic (EMG) signal decomposition. Master's thesis, ETH Zurich, 2004/ 2005.
- [11] Christian Rijke. Simulation of EMG signals to test decomposition algorithms. Semester thesis, ETH Zurich, 2004.
- [12] Dan Stashuk. EMG signal decomposition: how can it be accomplished and used? *Journal of Electromyography & Kinesiology*, 2001.
- [13] Jürg Treichler. Simulation of analog decoders. Semester thesis, ETH Zurich, 2002.
- [14] Roger Wattenhofer. Discrete event systems. lecture notes ETH Zurich, 2004/ 2005.

Bibliography

- [15] Markus Wenk. Segmentierung von EMG-Signalen. Semester thesis, ETH Zurich, 2005.
- [16] Christian Zimmermann. Decomposition of an EMG-Signal using an a-priori model. Semester thesis, ETH Zurich, 2005.