C.F. van Oosten

c.f.van.oosten@umail.leidenuniv.nl

The EM-algorithm for Poisson data

Bachelor thesis

Thesis supervisor: Dr. J. Schmidt-Hieber

Date of publishment: August 14, 2014

Mathematical Institute of Leiden University

Contents

1 Problem formulation

The goal is to find stable estimators for the parameters $\lambda = (\lambda_j)_j$ in the following problem: We know a matrix A of known weights given as

$$A = (a_{ij})_{\substack{i=1,...,n \\ j=1,...,m}}$$

$$\sum_{j=1}^{m} a_{ij} = 1, a_{ij} \geq 0, \quad \forall i = 1, ..., n, j = 1, ..., m.$$

These conditions are necessary for identifiability of the parameter λ. The distribution of N_{ij} is given by

$$N_{ij} \sim \mathcal{P}(a_{ij}\lambda_j), i = 1, ..., n, j = 1, ..., m$$

where $\mathcal{P}$ denotes the Poisson distribution. While we do not know the observations of these N_{ij}, we do have the observations $\{Y_i\}_{i=1,...n}$ which have the following distribution:

$$Y_i = \sum_{j=1}^{m} N_{ij} \sim \mathcal{P}(\sum_{j=1}^{m} a_{ij}\lambda_j) \tag{1}$$

We would like to find stable estimators for λ using the data $\{Y_i\}_{i=1,...,n}$.

2 Summary

Using the EM-algorithm we calculate iterative formulas to estimate λ. Because the iteration formula for Poisson data is difficult to analyse, we study a related problem. We study the normal distribution with parameters $a_{ij}\lambda_j, a_{ij}$ and another normal distribution with parameters $a_{ij}\lambda_j, a_{ij}\lambda_j$ which approximates the Poisson distribution for large $a_{ij}\lambda_j$. We find the Landweber iteration formula for the first normal distribution which has an explicit solution. We compare the second normal distribution with the Poisson case to find the similarities. In the end we run some simulations to see the convergent behaviour of the iteration formula's.

3 Application

This problem occurs in positron emission tomography, molecular microscopy and various problems in astrophysics as mentioned in [4]. In positron emission tomography (PET), the Poisson data models the emission density. The model fits the distribution of the amount of photons in double-slit interference of light. This works as follows: After the light goes through the double-slit it diverges, where a bigger diversion results in a lower light intensity.

4 EM-algorithm

The EM-algoritm is an iterative method used to find the maximum likelihood for parameters and can be used when some of the data is not available. We will first define

the EM-algorithm as in [2, p. 134]. Let $p(X, \theta)$ be the probability function of X with parameter θ and $S(X)$ is a linear function of X. Let

$$J(\theta|\theta_0) \equiv E_{\theta_0}\left(\log \frac{p(X,\theta)}{p(X,\theta_0)}|S(X) = s\right). \tag{2}$$

Then do the following steps:
Step 1: Initialize the parameter $\theta_{old} = \theta_0$.
Step 2: Compute $J(\theta|\theta_{old})$ for as many values as needed.
Step 3: Maximize $J(\theta|\theta_{old})$ as a function of θ.
Step 4: Define $\theta_{new} = \arg\max J(\theta|\theta_{old})$, set $\theta_{old} = \theta_{new}$ and continue with step 2

5 EM-algorithm for Poisson data

In model (1) we can easily compute the likelihood function, so we can use the EM-algorithm to estimate λ. We find for the likelihood function

$$\mathcal{L}_{(N_{ij})_{ij}}(\lambda) = \prod_{i=1}^{n}\prod_{j=1}^{m} e^{a_{ij}\lambda_j}\frac{(\lambda_j a_{ij})^{N_{ij}}}{N_{ij}!}$$

and for the log-likelihood function we find

$$l_{(N_{ij})_{ij}}(\lambda) = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(-\lambda_j a_{ij} + N_{ij}\log(\lambda_j a_{ij}) - log(N_{ij}!)\right).$$

Looking at the derivative of the log-likelihood with resprect to λ_j we obtain

$$\frac{d}{d\lambda_j}E[l_{(N_{ij})_{ij}}|(Y_i)_i] = \sum_{i=1}^{n} -a_{ij} + \frac{1}{\lambda_j}E[(N_{ij}|(Y_i)_i] \quad \forall j = 1, ..., m.$$

Note that $N_{ij}|(Y_i)_i$ has the same distribution as $N_{ij}|Y_i$ because N_{ij} is independent of all Y_k with $k \neq i$. We use the following claim to determine the distribution of $N_{ij}|Y_i$, the proof can be found in the technical appendix:

Lemma 5.1. *Let X_1, X_2 be independent Poisson distributions with*

$$X_1 \sim \mathcal{P}(\lambda_1),$$
$$X_2 \sim \mathcal{P}(\lambda_2).$$

Then $X_1|(X_1 + X_2) \sim Bin\left(X_1 + X_2, \frac{\lambda_1}{\lambda_1+\lambda_2}\right)$.

□

By taking $X_1 = N_{ij}$ and $X_2 = Y_i - N_{ij}$ we find $N_{ij}|Y_i \sim \text{Bin}\left(Y_i, \frac{a_{ij}\lambda_j}{\sum_{k=1}^{m} a_{ik}\lambda_k}\right)$.

Because the expectation of a Binomial distribution with parameters n, p is np we have

$$E[N_{ij}|Y_i] = \frac{Y_i a_{ij}\lambda_j}{\sum_{k=1}^{m} a_{ik}\lambda_k}.$$

Therefore

$$\frac{d}{d\lambda_j}E[l_{(N_{ij})_{ij}}|(Y_i)_i] = \sum_{i=1}^{n} -a_{ij} + \frac{1}{\lambda_j}\frac{Y_i a_{ij}\lambda_j^{old}}{\sum_{k=1}^{m} a_{ik}\lambda_k^{old}} \; \forall j = 1, ..., m. \tag{3}$$

Setting the derivative to 0 to find a possible maximum gives

$$0 = \sum_{i=1}^{n} -a_{ij} + \frac{1}{\lambda_j}\frac{Y_i a_{ij}\lambda_j^{old}}{\sum_{k=1}^{m} a_{ik}\lambda_k^{old}}.$$

Solving this for all λ_j gives

$$\lambda_j = \frac{\lambda_j^{old}}{\sum_{i=1}^{n} a_{ij}} \sum_{i=1}^{n} \frac{Y_i a_{ij}}{\sum_{k=1}^{m} a_{ik}\lambda_k^{old}}. \tag{4}$$

Looking back at (3) we see that the derivative function is a decreasing function of λ_j, so the value we found is a maximum value.

6 EM-algorithm for normal data

Since the Poisson problem issues a dilemma in finding an exact solution, we first work in a toy model. We find that for larger parameters λ of the Poisson distribution that it approximates a normal distribution with mean λ and variance λ. So we look at the similar problem with normal distributions which we formulate as:

$$N_{ij} \sim \mathcal{N}(a_{ij}\lambda_j, a_{ij})$$
$$A = (a_{ij})_{\substack{i=1,...,n \\ j=1,...,m}}$$

With the following conditions on the matrix A:

$$\sum_{j=1}^{m} a_{ij} = 1, a_{ij} \geq 0, \forall i = 1, ..., n, j = 1, ..., m$$

Here we observe the data $\{Y_i\}_{i=1,...,n}$ which has the following distribution:

$$Y_i = \sum_{j=1}^{m} N_{ij} \sim \mathcal{N}(\sum_{j=1}^{m} a_{ij}\lambda_j, 1) \tag{5}$$

In this model we can easily compute the likelihood function of normal distributions, so we use the EM-algorithm to estimate λ. We find the likelihood function for N_{ij} to be

$$\mathcal{L}_{(N_{ij})_{ij}}(\lambda) = \prod_{i=1}^{n}\prod_{j=1}^{m}\left(\frac{1}{\sqrt{a_{ij}}\sqrt{2\pi}} e^{-\frac{1}{2a_{ij}}(N_{ij}-a_{ij}\lambda_j)^2}\right)$$

and the log-likelihood function of N_{ij} is

$$l_{(N_{ij})_{ij}}(\lambda) = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(-\log(\sqrt{a_{ij}}\sqrt{2\pi}) - \frac{1}{2a_{ij}}(N_{ij} - a_{ij}\lambda_j)^2\right).$$

Thus the derivative of the log-likelihood with respect to λ_j is

$$\frac{d}{d\lambda_j}E[l_{(N_{ij})_{ij}}|Y_i] = \sum_{i=1}^{n} -\frac{1}{2a_{ij}}\frac{d}{d\lambda_j}E[(N_{ij} - a_{ij}\lambda_j)^2|Y_i] \quad \forall j = 1, ..., m.$$

To find a maximum for the log-likelihood we have to compute $\frac{d}{d\lambda_j}E[(N_{ij}-a_{ij}\lambda_j)^2|Y_i]$. Note that $(N_{ij} - a_{ij}\lambda_j)^2$ is a continuous function and so is the derivative $\frac{d}{d\lambda_j}(N_{ij} - a_{ij}\lambda_j)^2 = -2a_{ij}(N_{ij} - \lambda_j a_{ij})$. Since $N_{ij} - a_{ij}\lambda_j$ is a normal distribution function it is also bounded. Therefore $(N_{ij} - a_{ij}\lambda_j)^2$ and $-2a_{ij}(N_{ij} - \lambda_j a_{ij})$ are also bounded. This satisfies the conditions of theorem 10.3 in [3] so we can switch the order of differentation and expectation. By switching them we find

$$\frac{d}{d\lambda_j}E[(N_{ij} - a_{ij}\lambda_j)^2|Y_i] = -2a_{ij}E[N_{ij} - \lambda_j a_{ij}|Y_i]$$

which gives us

$$\frac{d}{d\lambda_j}E[l_{(N_{ij})_{ij}}|Y_i] = \sum_{i=1}^{n} E[N_{ij} - \lambda_j a_{ij}|Y_i] \; \forall j = 1, ..., m.$$

To further simplify this we have to find an explicit formula for the conditional expectation of two normal distributions looks like. We use the following lemma of which the proof can be found in the technical appendix.

Lemma 6.1. *Let X_1, X_2 be non-degenerate normal distributions with*

$$X_1 \sim \mathcal{N}(\mu_1, \sigma_1),$$
$$X_2 \sim \mathcal{N}(\mu_2, \sigma_2).$$

Then $X_1|X_2 \sim \mathcal{N}\left(\mu_1 + \frac{\sigma_1}{\sigma_2}\rho(X_2 - \mu_2), (1 - \rho^2)\sigma_1^2\right)$, with ρ the correlation coefficient between X_1 and X_2.

□

For N_{ij} and Y_i we find that $\mathrm{Cov}(N_{ij}, Y_i) = \mathrm{Cov}(N_{ij}, \sum_{j=1}^{m} N_{ij}) = \mathrm{Cov}(N_{ij}, N_{ij}) = \mathrm{Var}(N_{ij}) = a_{ij}$. So the corrolation coefficient between N_{ij} and Y_i is

$$\rho = \frac{\mathrm{Cov}(N_{ij}, Y_i)}{\sqrt{\mathrm{Var}(N_{ij})}\sqrt{\mathrm{Var}(Y_i)}} = \sqrt{a_{ij}}.$$

Using the previous lemma we obtain

$$N_{ij}|Y_i \sim \mathcal{N}\left(a_{ij}\lambda_j^{old} + a_{ij}(Y_i - \sum_{l=1}^{m} a_{il}\lambda_l^{old}), (1 - a_{ij})a_{ij}\right).$$

This means $E[N_{ij} - \lambda_j a_{ij}|Y_i] = a_{ij}\lambda_j^{old} + a_{ij}(Y_i - \sum_{l=1}^m a_{il}\lambda_l^{old})$ which gives

$$\frac{d}{d\lambda_j}E[l_{(N_{ij})_{ij}}(\lambda)|Y_i] = \sum_{i=1}^n \left(a_{ij}(\lambda_j^{old} - \lambda_j) + a_{ij}(Y_i - \sum_{l=1}^m a_{il}\lambda_l^{old}) \right) \ \forall j = 1, ..., m.$$

To maximize the likelihood we have to solve $\frac{d}{d\lambda_j}E[l_{(N_{ij})_{ij}}(\lambda)|Y_i] = 0$, or equivalently

$$[\sum_{i=1}^n a_{ij}](\lambda_j^{old} - \lambda_j) + \sum_{i=1}^n [a_{ij}(Y_i - \sum_{l=1}^m a_{il}\lambda_l^{old})] = 0 \ \forall j = 1, ..., m.$$

Bringing all terms other than λ to the other side gives

$$\lambda_j = \lambda_j^{old} + \frac{1}{\sum_{i=1}^n a_{ij}} \sum_{i=1}^n a_{ij}(Y_i - \sum_{l=1}^m a_{il}\lambda_l^{old}) \ \forall j = 1, ..., m. \tag{6}$$

Define $w = (w_j)_{j=1,...,m}$ with terms $w_j = \frac{1}{\sum_{i=1}^n a_{ij}}$ for all $j = 1, ..., m$ and take W the square matrix with elements w_j on the diagonal.

Then we can rewrite equation (6) into matrix notation. This gives the following updating formula:

$$\lambda = \lambda^{old} + WA^T(Y - A\lambda^{old}) \tag{7}$$

This iterative method is also known as Landweber iteration and will be studied in more detail in the next section.

7 Landweber iteration

We have found an iterative method to estimate our parameters now, but can we find an explicit solution for each iteration step without having to go through all steps before that?

Fortunately there is such a solution if we initialize the EM-algorithm with $\lambda^{(0)} = 0$, as shown by the following lemma:

Lemma 7.1. *Let W in (7) be constant on the diagonal and write*

$$\lambda^{(k)} = p_k(WA^TA)WA^TY \qquad \text{for } k = 0, 1, 2, ...$$

Then $\lambda^{(k)}$ is an explicit solution to the Landweber iteration if

$$p_k(x) = -\frac{1}{x}(1 - x)^k + \frac{1}{x}$$

Proof: Define $x = WA^TA$. Substituting $\lambda^{(k)}$ into equation (7) gives

$$\begin{aligned} p_{k+1}(x)WA^TY &= p_k(x)WA^TY + WA^TY - xp_k(x)WA^TY \\ p_{k+1}(x) &= p_k(x) + 1 - xp_k(x). \end{aligned}$$

Substituting p_k we have in the lemma gives

$$\begin{aligned} -\frac{1}{x}(1-x)^{k+1} + \frac{1}{x} &= (1-x)\left(-\frac{1}{x}(1-x)^k + \frac{1}{x}\right) + 1 \\ &= -\frac{1}{x}(1-x)^{k+1} + \frac{1-x}{x} + 1 \\ &= -\frac{1}{x}(1-x)^{k+1} + \frac{1}{x}. \end{aligned}$$

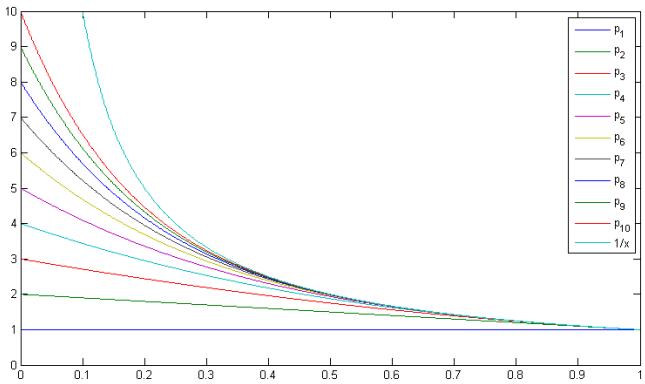
This goes for all $x \neq 0$, so $\lambda^{(k)}$ suffices the iteration method.

We find $\lambda^{(0)} = p_0(x)WA^TY = 0 \cdot WA^TY = 0$. Thus $\lambda^{(k)} = p_k(WA^TA)WA^TY$ is a solution to the Landweber iteration if $p_k(x) = -\frac{1}{x}(1-x)^k + \frac{1}{x}$. □

Since A^TA is a symmetric matrix there exists an orthogonal matrix D and a diagonal matrix Δ such that $A^TA = D^T\Delta D$. Then we can easily rewrite

$$p_k(WA^TA) = D^T p_k(W\Delta)D.$$

For k large, $p_k(x)$ approaches $\frac{1}{x}$ on $0 < x < 1$. Moreover, the supremum over p_k is bounded, that is $\sup_{x\in[0,1]} p_k(x) \leq k$. This is shown in the following figure:



8 EM-algorithm for normal data revisited

Now that we found an explicit solution for the normal model with variance independent of λ (5) we can improve it further. We expand the toy model by making the variance of N_{ij} dependent on λ_j. This means the problem is formulated as

$$M_{ij} \sim \mathcal{N}(a_{ij}\lambda_j, a_{ij}\lambda_j).$$

We are interested in this normal distribution because for large parameters $a_{ij}\lambda_j$ this distribution approximates a Poisson distribution according to the Central Limit Theorem. We have the same weight matrix A:

$$A = (a_{ij})_{\substack{i=1,\dots,n \\ j=1,\dots,m}}$$

with the following conditions on the matrix A:

$$\sum_{j=1}^{m} a_{ij} = 1, a_{ij} \geq 0 \; \forall i = 1, ..., n, j = 1, ..., m.$$

We know the data $\{Y_i\}_{i=1,\dots,n}$ which has the following distribution:

$$Y_i = \sum_{j=1}^{m} M_{ij} \sim \mathcal{N}(\sum_{j=1}^{m} a_{ij}\lambda_j, \sum_{j=1}^{m} a_{ij}\lambda_j) \tag{8}$$

We find that the likelihood fuction of M_{ij} is

$$\mathcal{L}_{M_{ij}}(\lambda) = \prod_{i=1}^{n}\prod_{j=1}^{m}\left[\frac{1}{\sqrt{2\pi}\sqrt{a_{ij}\lambda_j}} + \exp\left(-\frac{1}{2a_{ij}\lambda_j}(M_{ij} - a_{ij}\lambda_j)^2\right)\right].$$

Thus the log-likelihood becomes

$$l_{M_{ij}}(\lambda) = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(-\frac{1}{2}\log(2\pi a_{ij}\lambda_j) - \frac{1}{2a_{ij}\lambda_j}(M_{ij} - a_{ij}\lambda_j)^2\right).$$

Looking at the derivative of the log-likelihood with respect to λ_j we obtain

$$\frac{d}{d\lambda_j}E[l_{M_{ij}}(\lambda)] = \sum_{i=1}^{n}\left(-\frac{1}{2\lambda_j} - \frac{d}{d\lambda_j}E\left[\frac{1}{2a_{ij}\lambda_j}(M_{ij} - a_{ij}\lambda_j)^2\right]\right).$$

We know $M_ij - a_ij\lambda_j$ is a normal probabilty function, so it is continuous and bounded. Thus $\frac{1}{2a_{ij}\lambda_j}(M_{ij} - a_{ij}\lambda_j)^2$ is continous and bounded for $\lambda_j \neq 0$. We find the same for the derivative $\frac{d}{d\lambda_j}\left(\frac{1}{2a_{ij}\lambda_j}(M_{ij} - a_{ij}\lambda_j)^2\right) = -\frac{1}{2a_{ij}\lambda_j^2}(M_{ij} - a_{ij}\lambda_j)^2 - \frac{1}{\lambda_j}(M_{ij} - a_{ij}\lambda_j)$. This satisfies the conditions of theorem 10.3 in [3] so we can switch the order of differentation and expectation. By switching them we obtain

$$\frac{d}{d\lambda_j}E[l_{M_{ij}}(\lambda)|Y_i] = \sum_{i=1}^{n}\left(-\frac{1}{2\lambda_j} + \frac{1}{2a_{ij}\lambda_j^2}E[(M_{ij} - a_{ij}\lambda_j)^2|Y_i] + \frac{1}{\lambda_j}E[M_{ij} - a_{ij}\lambda_j|Y_i]\right). \tag{9}$$

To find these expectations we look into the following claim of which the proof can be found in the technical appendix:

Claim 8.1. *Let $M'_{ij} = \alpha Y_i + \xi_{ij}$. Then M_{ij} and M'_{ij} have the same distribution when*

$$\xi_{ij} \sim \mathcal{N}\left(0, a_{ij}\lambda_j - \frac{(a_{ij}\lambda_j)^2}{\sum_{j=1}^m a_{ij}\lambda_j}\right),$$

$$\alpha = \frac{a_{ij}\lambda_j}{\sum_{j=1}^m a_{ij}\lambda_j}.$$

With this claim we can find the expectations in (9). We have

$$\begin{aligned}
E[(M'_{ij})^2|Y_i] &= E[\alpha^2 Y_i^2 - 2\alpha Y_i \xi_{ij} + \xi_{ij}^2|Y_i] \\
&= \frac{(a_{ij}\lambda_j^{old})^2}{(\sum_{j=1}^m a_{ij}\lambda_j^{old})^2} Y_i^2 + 0 + a_{ij}\lambda_j^{old} - \frac{(a_{ij}\lambda_j^{old})^2}{\sum_{j=1}^m a_{ij}\lambda_j^{old}} \\
E[M'_{ij}|Y_i] &= \frac{a_{ij}\lambda_j^{old}}{\sum_{j=1}^m a_{ij}\lambda_j^{old}} Y_i.
\end{aligned}$$

Therefore we obtain

$$\begin{aligned}
E[(M'_{ij} - a_{ij}\lambda_j)^2|Y_i] &= E[M'^2_{ij} - a_{ij}\lambda_j M'_{ij} + a_{ij}^2\lambda_j^2|Y_i] \\
&= \frac{(a_{ij}\lambda_j^{old})^2}{(\sum_{j=1}^m a_{ij}\lambda_j^{old})^2} Y_i^2 + 0 + a_{ij}\lambda_j^{old} - \frac{(a_{ij}\lambda_j^{old})^2}{\sum_{j=1}^m a_{ij}\lambda_j^{old}} \\
&\quad - 2a_{ij}\lambda_j \frac{a_{ij}\lambda_j^{old}}{\sum_{j=1}^m a_{ij}\lambda_j^{old}} Y_i + a_{ij}^2\lambda_j^2.
\end{aligned}$$

Substituting these in equation (9) results in

$$\begin{aligned}
\frac{d}{d\lambda_j} E[l_{M_{ij}}(\lambda)|Y_i] &= \sum_{i=1}^n \left(-\frac{1}{2\lambda_j} + \frac{1}{2a_{ij}\lambda_j^2} E[(M'_{ij} - a_{ij}\lambda_j)^2|Y_i] + \frac{1}{\lambda_j} E[M'_{ij} - a_{ij}\lambda_j|Y_i] \right) \\
&= \sum_{i=1}^n \left(-\frac{1}{2\lambda_j} + \frac{1}{2a_{ij}\lambda_j^2} \Big[\frac{(a_{ij}\lambda_j^{old})^2}{(\sum_{j=1}^m a_{ij}\lambda_j^{old})^2} Y_i^2 + 0 + a_{ij}\lambda_j^{old} - \frac{(a_{ij}\lambda_j^{old})^2}{\sum_{j=1}^m a_{ij}\lambda_j^{old}} \right. \right. \\
&\quad \left. \left. -2a_{ij}\lambda_j \frac{a_{ij}\lambda_j^{old}}{\sum_{j=1}^m a_{ij}\lambda_j^{old}} Y_i + a_{ij}^2\lambda_j^2 \Big] + \frac{1}{\lambda_j} \left[\frac{a_{ij}\lambda_j^{old}}{\sum_{j=1}^m a_{ij}\lambda_j^{old}} Y_i - a_{ij}\lambda_j \right] \right) \\
&= \sum_{i=1}^n \left(-\frac{1}{2\lambda_j} + \frac{1}{2a_{ij}\lambda_j^2} \left[\frac{(a_{ij}\lambda_j^{old})^2}{(\sum_{j=1}^m a_{ij}\lambda_j^{old})^2} Y_i^2 + a_{ij}\lambda_j^{old} - \frac{(a_{ij}\lambda_j^{old})^2}{\sum_{j=1}^m a_{ij}\lambda_j^{old}} \right] - \frac{1}{2} a_{ij} \right)
\end{aligned}$$

$$\frac{d}{d\lambda_j} E[l_{M_{ij}}(\lambda)|Y_i] = \sum_{i=1}^n \left(-\frac{1}{2\lambda_j} + \frac{1}{2\lambda_j^2} \left[\lambda_j^{old} + \frac{a_{ij}(\lambda_j^{old})^2}{(\sum_{j=1}^m a_{ij}\lambda_j^{old})^2} Y_i^2 - \frac{a_{ij}(\lambda_j^{old})^2}{\sum_{j=1}^m a_{ij}\lambda_j^{old}} \right] - \frac{1}{2} a_{ij} \right). \tag{10}$$

Setting the derivative to 0 and multiplying with $2\lambda_j^2$ on both sides gives

$$0 = \sum_{i=1}^n \left(-\lambda_j + \left[\lambda_j^{old} + \frac{a_{ij}(\lambda_j^{old})^2}{(\sum_{j=1}^m a_{ij}\lambda_j^{old})^2} Y_i^2 - \frac{a_{ij}(\lambda_j^{old})^2}{\sum_{j=1}^m a_{ij}\lambda_j^{old}} \right] - a_{ij}\lambda_j^2 \right).$$

Or equivalently,

$$0 = \frac{n}{\sum_{i=1}^{n} a_{ij}}\lambda_j - \frac{\sum_{i=1}^{n}\left[\lambda_j^{old} + \frac{a_{ij}(\lambda_j^{old})^2}{(\sum_{j=1}^{m} a_{ij}\lambda_j^{old})^2}Y_i^2 - \frac{a_{ij}(\lambda_j^{old})^2}{\sum_{j=1}^{m} a_{ij}\lambda_j^{old}}\right]}{\sum_{i=1}^{n} a_{ij}} + \lambda_j^2. \tag{11}$$

So we have a quadratic equation. We can look at the behaviour when λ_j is either large or small. When λ_j is large then the linear term will be of a lower order compared to the quadratic term. When λ_j is small however the quadratic term will be of a lower order compared to the linear term.

Because we have a quadratic equation of the form $x^2 + px + q = 0$ we can solve this using the quadratic formula which gives solutions $x_\pm = -\frac{p}{2} + \sqrt{\frac{p^2}{4} - q}$. Let

$$p = \frac{n}{\sum_{i=1}^{n} a_{ij}},$$

$$q = -\frac{\sum_{i=1}^{n}\left[\lambda_j^{old} + \frac{a_{ij}(\lambda_j^{old})^2}{(\sum_{j=1}^{m} a_{ij}\lambda_j^{old})^2}Y_i^2 - \frac{a_{ij}(\lambda_j^{old})^2}{\sum_{j=1}^{m} a_{ij}\lambda_j^{old}}\right]}{\sum_{i=1}^{n} a_{ij}}.$$

This equation has two solutions, namely $\lambda_{j+,j-} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}$. Note that $p > 0$ since $n > 0$, so $\lambda_{j-} < 0$. This doesn't fit in our problem, so we should look at the other solution. We find that $\lambda_j^{old} + \frac{a_{ij}(\lambda_j^{old})^2}{(\sum_{j=1}^{m} a_{ij}\lambda_j^{old})^2}Y_i^2 - \frac{a_{ij}(\lambda_j^{old})^2}{\sum_{j=1}^{m} a_{ij}\lambda_j^{old}} > \lambda_j^{old} + \frac{a_{ij}(\lambda_j^{old})^2}{(\sum_{j=1}^{m} a_{ij}\lambda_j^{old})^2}Y_i^2 - \lambda_j^{old} = \frac{a_{ij}(\lambda_j^{old})^2}{(\sum_{j=1}^{m} a_{ij}\lambda_j^{old})^2}Y_i^2 \geq 0$ for all j. This means that $q \leq 0$, thus

$$\lambda_{j+} = -\frac{p}{2} + \sqrt{\frac{p^2}{4} - q} \tag{12}$$

is a positive solution.

Note that for $x \downarrow 0$ the derivative (10) goes to $+\infty$ because $\lambda_j^{old} + \frac{a_{ij}(\lambda_j^{old})^2}{(\sum_{j=1}^{m} a_{ij}\lambda_j^{old})^2}Y_i^2 - \frac{a_{ij}(\lambda_j^{old})^2}{\sum_{j=1}^{m} a_{ij}\lambda_j^{old}} \geq 0$. Since λ_{j+} is the only positive zero point this means for x near λ_j that $\frac{d}{d\lambda_j}E[l_{M_{ij}}(\lambda)|Y_i] > 0$ if $x < \lambda_{j+}$ and $\frac{d}{d\lambda_j}E[l_{M_{ij}}(\lambda)|Y_i] < 0$ if $x > \lambda_{j+}$. Thus λ_{j+} is a maximum value.

9 Linking the formulas of the Normal and Poisson problems

Let us look further into equation (11) when λ_j is large. We are looking for the stable value of the iteration formula, so we require $\lambda_j^{old} = \lambda_j$. We noticed that the linear terms of (11) are of a smaller order than the quadratic terms when λ_j is large. Since $\lambda_j^{old} = \lambda_j$ the λ_j^{old} terms are also of smaller order when λ_j is large. This means we can rewrite (11) into the following equation:

$$0 = \sum_{i=1}^{n} \frac{a_{ij}(\lambda_j^{old})^2}{(\sum_{j=1}^{m} a_{ij}\lambda_j^{old})^2} Y_i^2 - a_{ij}\lambda_j^2$$

$$\lambda_j^2 = \frac{1}{\sum\limits_{i=1}^{n} a_{ij}} \sum_{i=1}^{n} \frac{a_{ij}(\lambda_j^{old})^2}{(\sum_{j=1}^{m} a_{ij}\lambda_j^{old})^2} Y_i^2 \tag{13}$$

We know that $Y_i \sim \mathcal{N}(\sum_{j=1}^{m} a_{ij}\lambda_j, \sum_{j=1}^{m} a_{ij}\lambda_j)$, so we can say

$$Y_i = \sum_{j=1}^{m} a_{ij}\lambda_j + \sqrt{\sum_{j=1}^{m} a_{ij}\lambda_j} \cdot \xi_i$$

where $\xi_i \sim \mathcal{N}(0,1)$ and independent for all i. This means

$$\begin{aligned} Y_i^2 &= Y_i \left(\sum_{j=1}^{m} a_{ij}\lambda_j + \sqrt{\sum_{j=1}^{m} a_{ij}\lambda_j} \cdot \xi_i \right) \\ &= Y_i \sum_{j=1}^{m} a_{ij}\lambda_j + \text{ lower order terms of } \lambda_j \end{aligned}$$

Substituting this in (13) and taking $(\lambda_j^{old})^2$ out of the sum we find

$$\lambda_j^2 = \frac{(\lambda_j^{old})^2}{\sum\limits_{i=1}^{n} a_{ij}} \sum_{i=1}^{n} \frac{a_{ij}}{\sum_{k=1}^{m} a_{ik}\lambda_k^{old}} Y_i. \tag{14}$$

We can compare this equation with the iteration formula of the Poisson problem (4),

$$\lambda_j = \frac{\lambda_j^{old}}{\sum\limits_{i=1}^{n} a_{ij}} \sum_{i=1}^{n} \frac{\overline{Y_i} a_{ij}}{\sum\limits_{k=1}^{m} a_{ik}\lambda_k^{old}}.$$

Note that Y_i and $\overline{Y_i}$ have approximately the same distribution when $\sum_{j=1}^{m} a_{ij}\lambda_j$ is large. Let

$$D_j = \frac{1}{\sum\limits_{i=1}^{n} a_{ij}} \sum_{i=1}^{n} \frac{Y_i a_{ij}}{\sum_{k=1}^{m} a_{ik}\lambda_k^{old}}$$

Then we can rewrite the iterative formula (14) into

$$\lambda_j^2 = (\lambda_j^{old})^2 D_j$$

and for the Poisson formula (4) we obtain

$$\lambda_j = \lambda_j^{old} \overline{D_j}.$$

Since $\lambda_j^{old} \to \lambda_j$ for any convergent solution we find that $D_j \to 1$ for all $j = 1, 2, ...$ and so does $\overline{D_j}$. So there are similarities between the two, and if it can be proven that $D_j = 1$ has a unique solution then both formulas have the same stable values.

10 Simulations

We now have an iteration formula for the normal problem (8), but it is not easy to find an explicit solution. That is why we run some simulations and check if the iterative formula (12) has the same behaviour as the original formula for the Poisson data (4). We will generate the independent weights a_{ij} in two different ways: an exponential distribution and a fixed distribution. Generating weights a^*_{ij} exponentially with parameter $\frac{1}{m}$ we find $a^*_{ij} \geq 0$ and

$$E[a^*_{ij}] = \frac{1}{m} \text{ and } E[\sum_{j=1}^{m} a^*_{ij}] = 1$$

Then we normalize the weights a^*_{ij},

$$a_{ij} = \frac{a^*_{ij}}{\sum_{j=1}^{m} a^*_{ij}}$$

to make sure that $\sum_{j=1}^{m} a_{ij} = 1$.

We can deduce a fixed distribution for the weights from the dual-slit interference problem. Since the weights a_{ij} represent the amount of photons moving from point i to j we can assume that amount is only dependent on the distance between i and j. This result in A having the same elements on every subdiagonal. We can take the following values for weights on every subdiagonal:

$$b_k = \begin{cases} \frac{1}{(k+1)^z} & k \geq 0 \\ b_{-k} & k < 0 \end{cases}$$

$$a^*_{ij} = b_{(i-j)}$$

Where we choose $z \in (0, 2)$ as the diversion factor. We see that the main diagonal has the largest value because most photons will stay at the same point. Note that $\sum_{j=1}^{m} a^*_{ij} \neq 1$, so we have to normalize the weights:

$$a_{ij} = \frac{a^*_{ij}}{\sum_{j=1}^{m} a^*_{ij}}$$

With this in mind we look into five scenarios with our simulations.

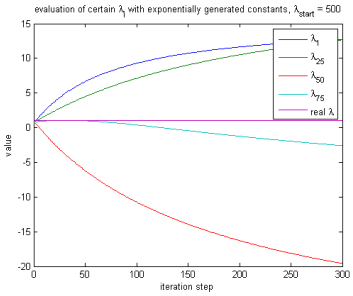
1. The weights are exponentially generated and $\lambda_{real} = 1$.
2. The weights are exponentially generated and $\lambda_{real} = 10000$.
3. The weights are fixed and $\lambda_{real} = 1, z = 2$.
4. The weights are fixed and $\lambda_{real} = 10000, z = 2$.
5. The weights are fixed and $\lambda_{real} = 10000, z = 1/100$.

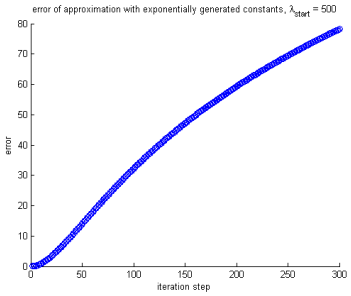
In the following simulations we use $m = n = 100$ and take the starting value for our approximation at 500.

10.1 Normal model with variance independent of λ (First model)

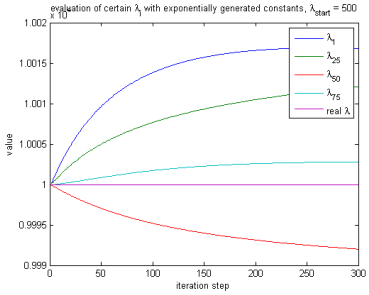
We simulate the first model which resulted in the Landweber iteration (7). This gives the following results:

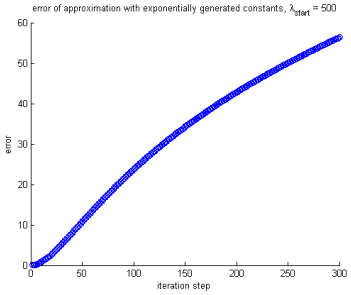
Scenario 1) The weights are exponentially generated and $\lambda_{real} = 1$:



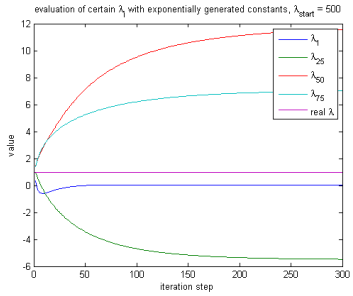


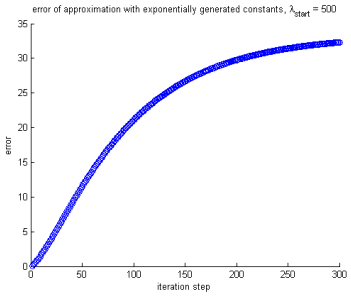
Scenario 2) The weights are exponentially generated and $\lambda_{real} = 10000$:



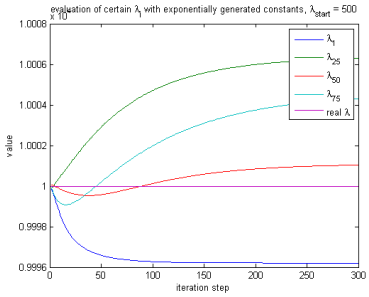


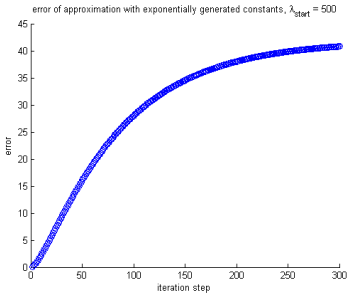
Scenario 3) The weights are fixed and $\lambda_{real} = 1, z = 2$:





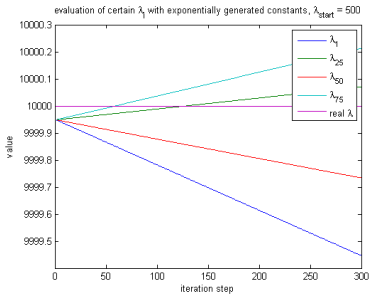
Scenario 4) The weights are fixed and $\lambda_{real} = 10000, z = 2$:

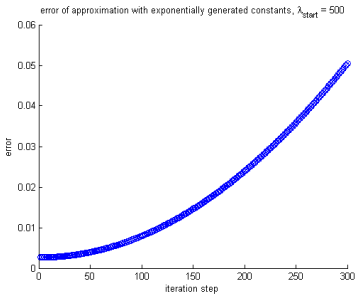




For scenarios 1-4 we notice that the minimal value was always attained in the second iteration step.

Scenario 5) The weights are fixed and $\lambda_{real} = 10000, z = 1/100$:



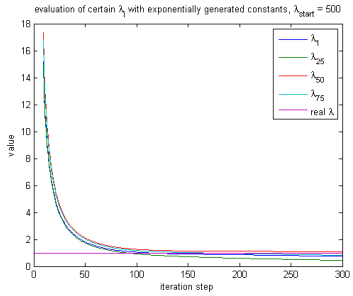


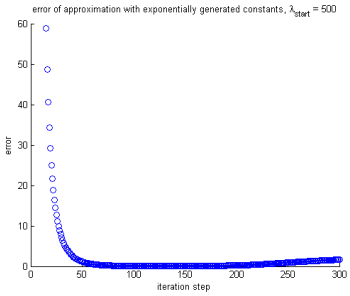
We see that the iteration formula gives a linear result.

10.2 Normal model with variance dependent of λ (Second model)

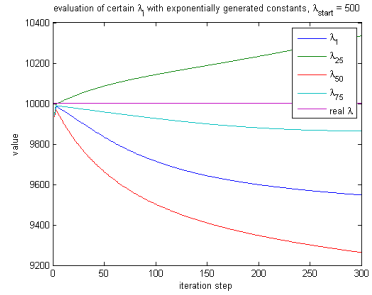
We use the iterative formula (12) to find the following simulations:

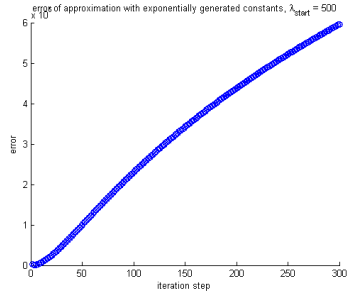
Scenario 1) The weights are exponentially generated and $\lambda_{real} = 1$:



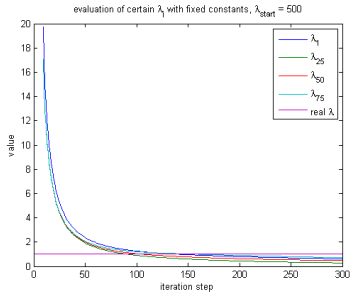


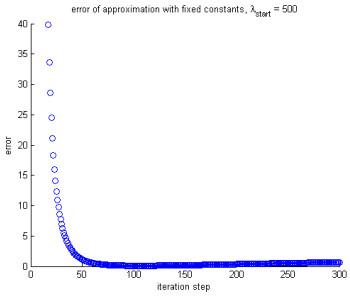
Scenario 2) The weights are exponentially generated and $\lambda_{real} = 10000$:



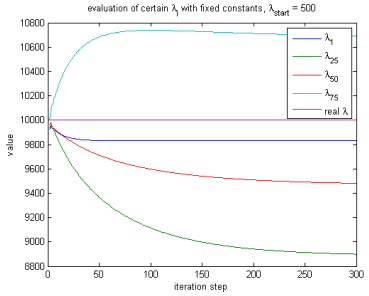


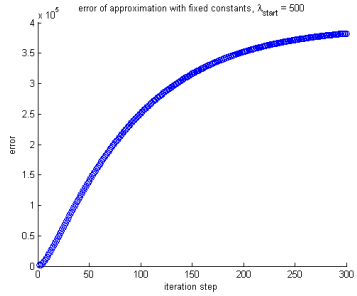
Scenario 3) The weights are fixed and $\lambda_{real} = 1, z = 2$:



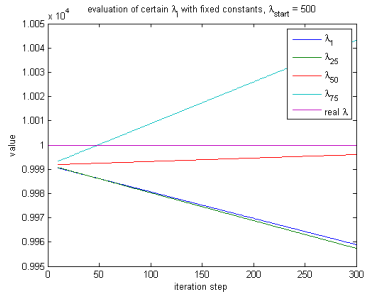


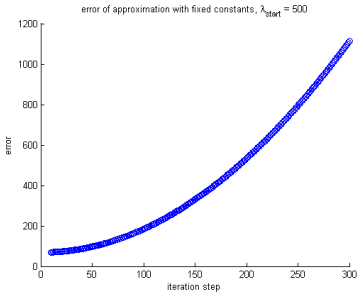
Scenario 4) The weights are fixed and $\lambda_{real} = 10000, z = 2$:





Scenario 5) The weights are fixed and $\lambda_{real} = 10000, z = 1/100$:

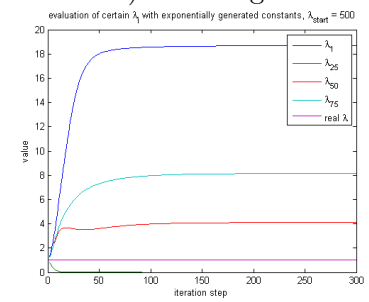


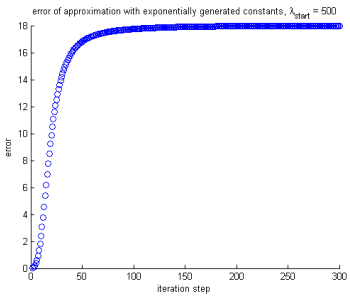


10.3 Poisson Model

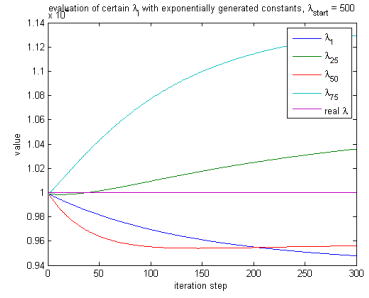
We can compare this with simulations done using the iterative formula for the Poisson model (4) we found:

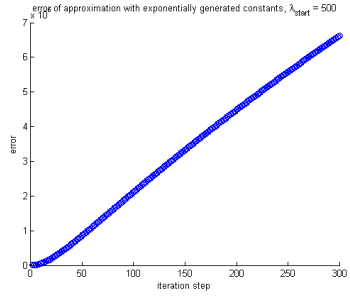
Scenario 1) The weights are exponentially generated and $\lambda_{real} = 1$:



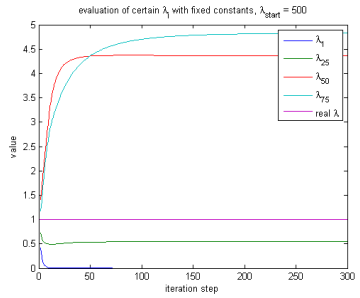


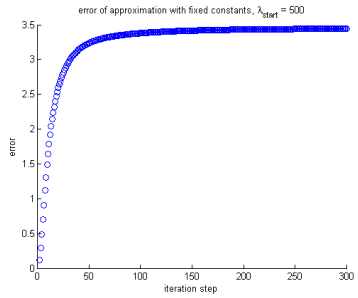
Scenario 2) The weights are exponentially generated and $\lambda_{real} = 10000$:



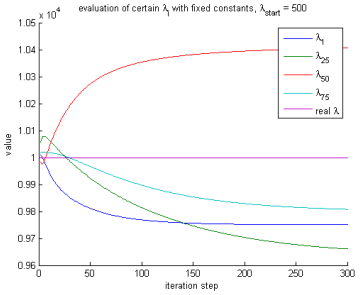


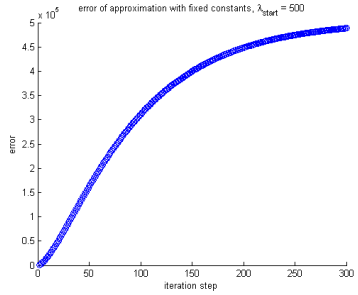
Scenario 3) The weights are fixed and $\lambda_{real} = 1, z = 2$:



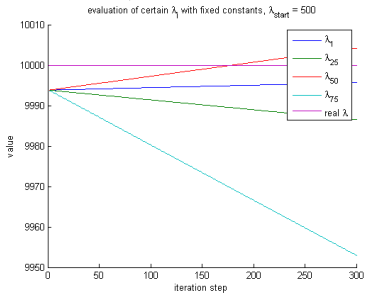


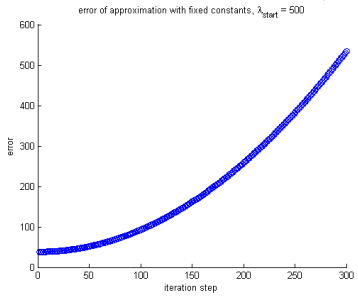
Scenario 4) The weights are fixed and $\lambda_{real} = 10000, z = 2$:





Scenario 5) The weights are fixed and $\lambda_{real} = 10000, z = 1/100$:





10.4 Discussion of simulation results

By comparing the models with large values for λ_{real} we see that the results are very similar for the normal model with variance dependent on λ and the Poisson model. This seems to stave our assumption that the EM-algorithm for the normal model we used is comparable with the EM-algorithm of the Poisson model. For the lower values of λ_{real} we see very different behaviour for each of the models. We can not base any conclusions off of them.

11 Technical Appendix

In this appendix, we give the proofs of lemma 5.1, lemma 6.1 and claim 8.1. In the second part, we provide the Matlab-code used for the simulations in section 10.

Proof lemma 5.1: Since X_1 and X_2 are independent we know that $X_1 + X_2$ has a Poisson distribution with parameter $\lambda_1 + \lambda_2$ [1]. Using the definition of the conditional probability we find (with the independence of X_1 and X_2)

$$\begin{aligned}
p(X_1 = x | X_1 + X_2 = y) &= \frac{p(X_1 = x, X_1 + X_2 = y)}{p(X_1 + X_2 = y)} \\
&= \frac{p(X_1 = x, X_2 = y - x)}{p(X_1 + X_2 = y)} \\
&= \frac{\frac{\lambda_1^x}{x!} \cdot^{-\lambda_1} \cdot \frac{\lambda_2^{y-x}}{(y-x)!} \cdot e^{-\lambda_2}}{\frac{(\lambda_1+\lambda_2)^y}{y!} \cdot e^{-\lambda_1-\lambda_2}} \\
&= \frac{\frac{\lambda_1^x}{x!} \cdot \frac{\lambda_2^{y-x}}{(y-x)!}}{\frac{(\lambda_1+\lambda_2)^y}{y!}} \\
&= \frac{y!}{x!(y-x)!} \cdot \frac{\lambda_1^x \cdot \lambda_2^{y-x}}{(\lambda_1+\lambda_2)^y} \\
&= \binom{y}{x} \cdot \frac{\lambda_1^x}{(\lambda_1+\lambda_2)^x} \cdot \frac{\lambda_2^{y-x}}{(\lambda_1+\lambda_2)^{y-x}} \\
&= \binom{y}{x} \cdot \left(\frac{\lambda_1}{\lambda_1+\lambda_2}\right)^x \cdot \left(\frac{\lambda_2}{\lambda_1+\lambda_2}\right)^{y-x} \\
&= \binom{y}{x} \cdot \left(\frac{\lambda_1}{\lambda_1+\lambda_2}\right)^x \cdot \left(1 - \frac{\lambda_1}{\lambda_1+\lambda_2}\right)^{y-x}
\end{aligned}$$

This is the probability function of Bin $\left(y, \frac{\lambda_1}{\lambda_1+\lambda_2}\right)$

Proof lemma 6.1: Using the definition of the conditional probability we find

$$p(X_1 = x | X_2 = y) = \frac{p(X_1 = x, X_2 = y)}{p(X_2 = y)}$$

We know that

$$\begin{aligned}
p(X_1 = x, X_2 = y) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{(1-\rho^2)}} \exp\Bigg(-\frac{1}{2(1-\rho^2)}\Bigg[\frac{(x-\mu_1)^2}{\sigma_1^2} \\
-\frac{2\rho}{\sigma_1\sigma_2}(x-\mu_1)(y-\mu_2) + \frac{(y-\mu_2)^2}{\sigma_2^2}\Bigg]\Bigg)
\end{aligned}$$

This gives us the following result

$$\begin{aligned} p(X_1 = x, X_2 = y) &= \frac{1}{\sigma_1\sqrt{(1-\rho^2)}\sqrt{2\pi}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_1)^2}{\sigma_1^2}\right.\right. \\ &\quad \left.\left. -\frac{2\rho}{\sigma_1\sigma_2}(x-\mu_1)(y-\mu_2) + (1-(1-\rho^2))\frac{(y-\mu_2)^2}{\sigma_2^2}\right]\right) \\ &= \frac{1}{\sigma_1\sqrt{(1-\rho^2)}\sqrt{2\pi}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_1)}{\sigma_1} - \rho\frac{(y-\mu_2)}{\sigma_2}\right]^2\right) \end{aligned}$$

Proof claim 8.1: We can rewrite our distribution of M_{ij} and Y_i as

$$\begin{pmatrix} M_{ij} \\ Y_i \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} a_{ij}\lambda_j \\ \sum_{j=1}^m a_{ij}\lambda_j \end{pmatrix}, \begin{pmatrix} a_{ij}\lambda_j & a_{ij}\lambda_j \\ a_{ij}\lambda_j & \sum_{j=1}^m a_{ij}\lambda_j \end{pmatrix}\right)$$

We can look at $M'_{ij} = \alpha Y_i + \xi_{ij}$. We have

$$E[M'_{ij}] = \alpha E[Y_i] + E[\xi_{ij}]$$

$$\operatorname{Var}(M'_{ij}) = \alpha^2 \operatorname{Var}(Y_i) + \operatorname{Var}(\xi_{ij})$$

Thus we find that

$$\begin{pmatrix} M'_{ij} \\ Y_i \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \alpha\sum_{j=1}^m a_{ij}\lambda_j + E[\xi_{ij}] \\ \sum_{j=1}^m a_{ij}\lambda_j \end{pmatrix}, \begin{pmatrix} \alpha^2\sum_{j=1}^m a_{ij}\lambda_j + \operatorname{Var}(\xi_{ij}) & B_{ij} \\ B_{ij} & \sum_{j=1}^m a_{ij}\lambda_j \end{pmatrix}\right)$$

Where $B_{ij} = \operatorname{Cov}(M'_{ij}, Y_i)$. Note that ξ_{ij} and Y_i are independent, so we find

$$\operatorname{Cov}(M'_{ij}, Y_i) = \alpha \operatorname{Var}(Y_i) = \alpha \sum_{j=1}^m a_{ij}\lambda_j$$

Substituting our α and ξ as in the claim we find

$$\begin{pmatrix} M'_{ij} \\ Y_i \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} a_{ij}\lambda_j \\ \sum_{j=1}^m a_{ij}\lambda_j \end{pmatrix}, \begin{pmatrix} a_{ij}\lambda_j & a_{ij}\lambda_j \\ a_{ij}\lambda_j & \sum_{j=1}^m a_{ij}\lambda_j \end{pmatrix}\right)$$

So M_{ij} and M'_{ij} have the same distribution.

Code for normal problem with variation independent of λ, exponential weights

```
%This program runs the iterative formula found for the
%Normal a_ij lambda_j, a_ij lambda_j distribution
%with exponentially distributed weights
%for simulations

%%%%%%%%%%%%%%%%%%%  PARAMETERS  %%%%%%%%%%%%%%%%%%%

m = 100; %Rows of matrix A
n = 100; %Columns of matrix A

true = 10000;
lambdareal = true*ones(1,m); %true lambda

k = 300; %nr of iteration steps


%%%%%%%%%%%% Objects for computations %%%%%%%%%%%%%%%

A = zeros(n,m); %weight matrix
A_init = zeros(n,m); %for construction of weight matrix

%lambda = 0.001*ones(k,m);
lambda = 500*ones(k,m); %starting value
N = zeros(n,m); %matrix of Poisson variables N_ij
u = zeros(k,1); %for approx. error in each iteration step


%%%%%%%%% COMPUTATION OF MATRIX A %%%%%%%%%%%%%%%%%%%%

A_init = -log(rand(n,m))./m; %generate exponential values
S = sum(A_init,2);
for i = 1:n; %normalize the values
    for j = 1:m;
        A(i,j) = A_init(i,j)/S(i);
    end
end

%%%%%%%%% COMPUTATION OF VARIABLES N %%%%%%%%%%%%%%%%%%

for i = 1:n;
    for j = 1:m;
        N(i,j) = A(i,j)*lambdareal(1,j) + sqrt(A(i,j))*randn(1);
    end
end
Sumi = sum(A); %sum over the i :   sum_i A_ij
Y = sum(N,2); %sum over the j :   sum_j N_ij
```

```
u(1) = (lambda(1,1)-lambdareal(1,1))^2;
%Average error of starting value

%%%%%%%%%%%%%% ITERATIONS %%%%%%%%%%%%%%%%%%%

temp = zeros(n,1); %for constructing lambda
for t=1:k-1;
    Q = A*lambda';
    for j =1:m;
        weight = 1/Sumi(j);
        for i=1:n;
            temp(i) = A(i,j)*(Y(i)-Q(i,t));
        end
        lambda(t+1,j) = lambda(t,j)+weight*sum(temp);
    end
    M=(lambda(t+1,:)-lambdareal).^2;
    u(t+1) = 1/m * sum(M);%average squared error
end

%%%%%%%%%%%%%% PLOTTING %%%%%%%%%%%%%%%%%%%

x = 1:k;
z=2;
w=2;
c(x)=lambdareal(1);
plot(x(z:end), lambda(x(z:end),1),x(z:end),lambda(x(z:end),25),...
    x(z:end), lambda(x(z:end),50),x(z:end),lambda(x(z:end),75),...
    x(1:end),c(x(1:end)))
xlabel('iteration step')
ylabel('value')
title(['evaluation of certain \lambda_j with ',...
    'exponentially generated constants, \lambda_{start} = 500'])
legend('\lambda_1','\lambda_{25}','\lambda_{50}',...
    '\lambda_{75}','real \lambda')
figure;
scatter(x(w:end),u(x(w:end)))
xlabel('iteration step')
ylabel('error')
title(['error of approximation with exponentially ',...
    'generated constants, \lambda_{start} = 500'])
```

Code for normal problem with variation independent of λ, fixed weights

```
%This program runs the iterative formula found for the
%Normal a_ij lambda_j, a_ij lambda_j distribution
%with fixed weights
%for simulations

%%%%%%%%%%%%%%%%%%%  PARAMETERS  %%%%%%%%%%%%%%%%%%%
```

```
m = 100;%Rows of matrix A
n = 100;%Columns of matrix A

true = 10000;
lambdareal = true*ones(1,m); %true lambda

k=300; %nr of iteration steps


%%%%%%%%%%%% Objects for computations %%%%%%%%%%%%%%%%

A = zeros(n,m); %weight matrix
A_init = zeros(n,m); %for construction of weight matrix

%lambda = 0.001*ones(k,m);
lambda = 500*ones(k,m); %starting value
N = zeros(n,m); %matrix of Poisson variables N_ij
u = zeros(k,1); %for approx. error in each iteration step


%%%%%%%%% COMPUTATION OF MATRX A %%%%%%%%%%%%%%%%%%%%%%

b = zeros(n,1);
for l = 1:n
    b(l) = 1/(l+1)^2;
    %b(l) = 1/(l+1)^1;
end
for z=1:n/m %assumes n is dividable by m
    for i = 1:m;
        for j = 1:m;
            A_init((z-1)*m+i,j) = b(abs(i-j)+1);
        end
    end
end
for i = 1:n; %normalize the values
    for j = 1:m;
        A(i,j) = A_init(i,j)/sum(A_init(i,:));
    end
end

%%%%%%%%% COMPUTATION OF VARIABLES N %%%%%%%%%%%%%%%%%%%

for i = 1:n;
    for j = 1:m;
        N(i,j) = A(i,j)*lambdareal(1,j) + sqrt(A(i,j))*randn(1);
    end
end
```

```
Sumi = sum(A); %sum over the i :   sum_i A_ij
Y = sum(N,2); %sum over the j :   sum_j N_ij
u(1) = (lambda(1,1)-lambdareal(1,1))^2;
%Average error of starting value

%%%%%%%%%%%%%%% ITERATIONS %%%%%%%%%%%%%%%%%%%%

temp = zeros(n,1); %for constructing lambda
for t=1:k-1;
    Q = A*transpose(lambda);
    for j =1:m;
        weight = 1/Sumi(j);
        for i=1:n;
            temp(i) = A(i,j)*(Y(i)-Q(i,t));
        end
        lambda(t+1,j) = lambda(t,j)+weight*sum(temp);
    end
    M=(lambda(t+1,:)-lambdareal).^2;
    u(t+1) = 1/m * sum(M);%average squared error
end

%%%%%%%%%%%%%%% PLOTTING %%%%%%%%%%%%%%%%%%%%

x = 1:k;
z=2;
w=2;
c(x)=lambdareal(1);
plot(x(z:end), lambda(x(z:end),1),x(z:end),lambda(x(z:end),25),...
    x(z:end), lambda(x(z:end),50),x(z:end),lambda(x(z:end),75),...
    x(1:end),c(x(1:end)))
xlabel('iteration step')
ylabel('value')
title(['evaluation of certain \lambda_j with exponentially ',...
    'generated constants, \lambda_{start} = 500'])
legend('\lambda_1','\lambda_{25}','\lambda_{50}',...
    '\lambda_{75}','real \lambda')
figure;
scatter(x(w:end),u(x(w:end)))
xlabel('iteration step')
ylabel('error')
title(['error of approximation with exponentially ',...
    'generated constants, \lambda_{start} = 500'])
```

Code for normal problem with variation dependent of λ, exponential weights

```
%This program runs the iterative formula found for the
%Normal a_ij lambda_j, a_ij lambda_j distribution
%with exponentially distributed weights
```

```
%for simulations

%%%%%%%%%%%%%%%%%%%%  PARAMETERS  %%%%%%%%%%%%%%%%%%%%

m = 100; %Rows of matrix A
n = 100; %Columns of matrix A

true = 10000;
lambdareal = true*ones(1,m); %true lambda

k = 300; %nr of iteration steps


%%%%%%%%%%%% Objects for computations %%%%%%%%%%%%%%%%

A = zeros(n,m); %weight matrix
A_init = zeros(n,m); %for construction of weight matrix

%lambda = 0.001*ones(k,m);
lambda = 500*ones(k,m); %starting value
N = zeros(n,m); %matrix of Poisson variables N_ij
u = zeros(k,1); %for approx. error in each iteration step


%%%%%%%%% COMPUTATION OF MATRIX A %%%%%%%%%%%%%%%%%%%%%

A_init = -log(rand(n,m))./m; %generate exponential values
S = sum(A_init,2);
for i = 1:n; %normalize the values
    for j = 1:m;
        A(i,j) = A_init(i,j)/S(i);
    end
end

%%%%%%%%% COMPUTATION OF VARIABLES N %%%%%%%%%%%%%%%%%%%

for i = 1:n; %generate normal data
    for j = 1:m;
        par = A(i,j)*lambdareal(1,j);
        N(i,j) = par + sqrt(par)*randn(1);
    end
end

Sumi = sum(A); %sum over the i :   sum_i A_ij
Y = sum(N,2); %sum over the j :   sum_j N_ij
u(1) = (lambda(1,1)-lambdareal(1,1))^2;
%Average error of starting value

%%%%%%%%%%%%%%%% ITERATIONS %%%%%%%%%%%%%%%%%%%%%%%%
```

```
temp = zeros(n,1); %for constructing lambda
for t=1:k-1;
    Q = A*lambda';
    for j =1:m;
        weight = 1/Sumi(j);
        for i=1:n;
            temp(i) = -lambda(t,j) - A(i,j)*lambda(t,j)^2/Q(i,t)^2*Y(i)^2+...
                +A(i,j)*lambda(t,j)^2/Q(i,t);
        end
        q = sum(temp)*weight;
        p = n * weight;
        lambda(t+1,j) = -p/2 + (p^2/4-q)^(1/2);
    end
    M = (lambda(t+1,:)-lambdareal).^2;
    u(t+1) = 1/m * sum(M); %average squared error
end

%%%%%%%%%%%%%% PLOTTING %%%%%%%%%%%%%%%%%%%%%

x = 1:k;
z=2;
w=2;
c(x)=lambdareal(1);
plot(x(z:end), lambda(x(z:end),1),x(z:end),lambda(x(z:end),25),...
    x(z:end), lambda(x(z:end),50),x(z:end),lambda(x(z:end),75),...
    x(1:end),c(x(1:end)))
xlabel('iteration step')
ylabel('value')
title(['evaluation of certain \lambda_j with ',...
    'exponentially generated constants, \lambda_{start} = 500'])
legend('\lambda_1','\lambda_{25}',...
    '\lambda_{50}','\lambda_{75}','real \lambda')
figure;
scatter(x(w:end),u(x(w:end)))
xlabel('iteration step')
ylabel('error')
title(['error of approximation with exponentially ', ...
'generated constants, \lambda_{start} = 500'])
```

Code for normal problem with variation dependent of λ, fixed weights

```
%This program runs the iterative formula found for the
%Normal a_ij lambda_j, a_ij lambda_j distribution
%with fixed weights
%for simulations

%%%%%%%%%%%%%%%%%%%  PARAMETERS  %%%%%%%%%%%%%%%%%%%
```

```
m = 100;%Rows of matrix A
n = 100;%Columns of matrix A

true = 10000;
lambdareal = true*ones(1,m); %true lambda

k=300; %nr of iteration steps


%%%%%%%%%%%% Objects for computations %%%%%%%%%%%%%%%%

A = zeros(n,m); %weight matrix
A_init = zeros(n,m); %for construction of weight matrix

%lambda = 0.001*ones(k,m);
lambda = 500*ones(k,m); %starting value
N = zeros(n,m); %matrix of Poisson variables N_ij
u = zeros(k,1); %for approx. error in each iteration step


%%%%%%%% COMPUTATION OF MATRX A %%%%%%%%%%%%%%%%%%%%%

b = zeros(n,1);
for l = 1:n
    b(l) = 1/(l+1)^2;
    %b(l) = 1/(l+1)^1;
end
for z=1:n/m %assumes n is dividable by m
    for i = 1:m;
        for j = 1:m;
            A_init((z-1)*m+i,j) = b(abs(i-j)+1);
        end
    end
end
for i = 1:n; %normalize the values
    for j = 1:m;
        A(i,j) = A_init(i,j)/sum(A_init(i,:));
    end
end

%%%%%%%% COMPUTATION OF VARIABLES N %%%%%%%%%%%%%%%%%%%

for i = 1:n; %generate normal data
    for j = 1:m;
        par = A(i,j)*lambdareal(1,j);
        N(i,j) = par + sqrt(par)*randn(1);
    end
end
```

```
Sumi = sum(A); %sum over the i :   sum_i A_ij
Y = sum(N,2); %sum over the j :   sum_j N_ij
u(1) = (lambda(1,1)-lambdareal(1,1))^2; %Average error of starting value


%%%%%%%%%%%%%%% ITERATIONS %%%%%%%%%%%%%%%%%%%%

temp = zeros(n,1); %for constructing lambda
for t=1:k-1;
    Q = A*lambda'; %Q(i,t) = sum_j a_ij lambda_j^t
    for j =1:m;
        weight = 1/Sumi(j);
        for i=1:n;
            temp(i) = -lambda(t,j) - A(i,j)*lambda(t,j)^2/Q(i,t)^2*Y(i)^2+...
            +A(i,j)*lambda(t,j)^2/Q(i,t);
        end
        q = sum(temp)*weight;
        p = n * weight;
        lambda(t+1,j) = -p/2 + (p^2/4-q)^(1/2);
    end
    M = (lambda(t+1,:)-lambdareal).^2;
    u(t+1) = 1/m * sum(M); %average squared error
end

%%%%%%%%%%%%%%% PLOTTING %%%%%%%%%%%%%%%%%%%%

x = 1:k;
z=9;
w=17;
c(x)=lambdareal(1);
plot(x(z:end), lambda(x(z:end),1),x(z:end),lambda(x(z:end),25),...
    x(z:end), lambda(x(z:end),50),x(z:end),lambda(x(z:end),75),...
    x(1:end),c(x(1:end)))
xlabel('iteration step')
ylabel('value')
title(['evaluation of certain \lambda_j ',...
    'with fixed constants, \lambda_{start} = 500'])
legend('\lambda_1','\lambda_{25}','\lambda_{50}',...
    '\lambda_{75}','real \lambda')
figure;
scatter(x(w:end),u(x(w:end)))
xlabel('iteration step')
ylabel('error')
title(['error of approximation with ', ...
    'fixed constants, \lambda_{start} = 500'])
```

Code for Poisson problem with parameter $a_{ij}\lambda_j$, exponential weights

```
%This program runs the iterative formula found for the
```

```
%Poisson a_ij lambda_j distribution
%with exponentially distributed weights
%for simulations

%%%%%%%%%%%%%%%%%%%  PARAMETERS  %%%%%%%%%%%%%%%%%%%

m = 100; %Rows of matrix A
n = 100; %Columns of matrix A

true = 10000;
lambdareal = true*ones(1,m); %true lambda

k = 300; %nr of iteration steps


%%%%%%%%%%%% Objects for computations %%%%%%%%%%%%%%%

A = zeros(n,m); %weight matrix
A_init = zeros(n,m); %for construction of weight matrix

%lambda = 0.001*ones(k,m);
lambda = 500*ones(k,m); %starting value
N = zeros(n,m); %matrix of Poisson variables N_ij
u = zeros(k,1); %for approx. error in each iteration step


%%%%%%%%% COMPUTATION OF MATRIX A %%%%%%%%%%%%%%%%%%%%%

A_init = -log(rand(n,m))./m; %generate exponential values
S = sum(A_init,2);
for i = 1:n; %normalize the values
    for j = 1:m;
        A(i,j) = A_init(i,j)/S(i);
    end
end

%%%%%%%%% COMPUTATION OF VARIABLES N %%%%%%%%%%%%%%%%%%%%%

for i = 1:n;
    for j = 1:m;
        L = exp(-A(i,j)*lambdareal(1,j)); %Knuth Poisson Generator
        t = 0;
        p = 1;
        while p > L
            p = p * rand(1);
            t = t+1;
        end
        N(i,j) = t-1;
        %N(i,j)=poissrnd(A(i,j)*lambdareal(1,j));
```

```
    end
end

sumi = sum(A); %sum over the i :   sum_i A_ij
Y = sum(N,2); %sum over the j :   sum_j N_ij
u(1) = (lambda(1,1)-lambdareal(1,1))^2; %Average error of starting value


%%%%%%%%%%%%%% ITERATIONS %%%%%%%%%%%%%%%%%%%%

temp = zeros(n,1); %for constructing lambda
for t=1:k-1;
    Q = A*lambda';
    for j =1:m;
        for i=1:n;
            temp(i) = Y(i)*A(i,j)/Q(i,t);
        end
        lambda(t+1,j) = lambda(t,j)/sumi(j)*sum(temp);
    end
    M=(lambda(t+1,:)-lambdareal).^2;
    u(t+1) = 1/m * sum(M);
end

%%%%%%%%%%%%%% PLOTTING %%%%%%%%%%%%%%%%%%%%

x = 1:k;
z=2;
w=2;
c(x)=lambdareal(1);
plot(x(z:end), lambda(x(z:end),1),x(z:end),lambda(x(z:end),25),...
    x(z:end), lambda(x(z:end),50),x(z:end),lambda(x(z:end),75),...
    x(1:end),c(x(1:end)))
xlabel('iteration step')
ylabel('value')
title(['evaluation of certain \lambda_j ', ...
    'with exponentially generated constants, \lambda_{start} = 500'])
legend('\lambda_1','\lambda_{25}','\lambda_{50}',...
    '\lambda_{75}','real \lambda')
figure;
scatter(x(2:end),u(x(2:end)))
xlabel('iteration step')
ylabel('error')
title(['error of approximation with ',...
    'exponentially generated constants, \lambda_{start} = 500'])
```

Code for Poisson problem with parameter $a_{ij}\lambda_j$, fixed weights

```
%This program runs the iterative formula found for the
```

```
%Poisson a_ij lambda_j distribution
%with fixed weights
%for simulations

%%%%%%%%%%%%%%%%%%%  PARAMETERS  %%%%%%%%%%%%%%%%%%%

m = 100;%Rows of matrix A
n = 100;%Columns of matrix A

true = 10000;
lambdareal = true*ones(1,m); %true lambda

k = 300; %nr of iteration steps


%%%%%%%%%%%% Objects for computations %%%%%%%%%%%%%%%

A = zeros(n,m); %weight matrix
A_init = zeros(n,m); %for construction of weight matrix

%lambda = 0.001*ones(k,m);
lambda = 500*ones(k,m); %starting value
N = zeros(n,m); %matrix of Poisson variables N_ij
u = zeros(k,1); %for approx. error in each iteration step

is_const_vec = zeros(1,k);


%%%%%%%%% COMPUTATION OF MATRX A %%%%%%%%%%%%%%%%%%%%%

b = zeros(n,1);
for l = 1:n
    b(l) = 1/(l+1)^2;
    %b(l) = 1/(l+1)^1;
end
for z=1:n/m %assumes n is dividable by m
    for i = 1:m;
        for j = 1:m;
            A_init((z-1)*m+i,j) = b(abs(i-j)+1);
        end
    end
end
for i = 1:n; %normalize the values
    for j = 1:m;
        A(i,j) = A_init(i,j)/sum(A_init(i,:));
    end
end

%%%%%%%%% COMPUTATION OF VARIABLES N %%%%%%%%%%%%%%%%%%
```

```
for i = 1:n;
    for j = 1:m;
        %L = exp(-A(i,j)*lambdareal(1,j)); %Knuth, poisson generator
        %t = 0;
        %p = 1;
        %while p > L
        %   p = p * rand(1);
        %   t = t+1;
        %end
        %N(i,j) = t-1;
        N(i,j)=poissrnd(A(i,j)*lambdareal(1,j));
    end
end

sumi = sum(A); %sum over the i :   sum_i A_ij
Y = sum(N,2); %sum over the j :   sum_j N_ij
u(1) = (lambda(1,1)-lambdareal(1,1))^2;
%u(1) = abs(lambda(1,1)-lambdareal(1,1));
is_const_vec(1)=lambda(1,1);


%%%%%%%%%%%%%% ITERATIONS %%%%%%%%%%%%%%%%%%%

temp = zeros(n,1); %for constructing lambda
for t=1:k-1;
    Q = A*lambda';
    for j =1:m;
        for i=1:n;
            temp(i) = Y(i)*A(i,j)/Q(i,t);
        end
        lambda(t+1,j) = lambda(t,j)/sumi(j)*sum(temp);
    end
    M=(lambda(t+1,:)-lambdareal).^2;
    %M=abs(lambda(t+1,:)-lambdareal);
    u(t+1) = 1/m * sum(M);
    is_const_vec(t+1) = mean(lambda(t+1,:));
end

%%%%%%%%%%%%%% PLOTTING %%%%%%%%%%%%%%%%%%%%

x = 1:k;
z=2;
w=2;
c(x)=lambdareal(1);
figure(1)
hold on
plot(x(z:end), lambda(x(z:end),1),x(z:end),lambda(x(z:end),25),...
    x(z:end), lambda(x(z:end),50),x(z:end),lambda(x(z:end),75),...
```

```
    x(1:end),c(x(1:end)))
xlabel('iteration step')
ylabel('value')
title(['evaluation of certain \lambda_j ',...
    'with fixed constants, \lambda_{start} = 500'])
legend('\lambda_1','\lambda_{25}','\lambda_{50}',...
    '\lambda_{75}','real \lambda')
figure;
scatter(x(w:end),u(x(w:end)))
xlabel('iteration step')
ylabel('error')
title(['error of approximation with ', ...
    'fixed constants, \lambda_{start} = 500'])
```

References

[1] Sums of independent random variables. `http://www.math.uconn.edu/~kconrad/blurbs/analysis/diffunderint.pdf`. Accessed: 2014-08-13.

[2] P.J. Bickel and K.A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics*. Number v. 1 in Holden-Day series in probability and statistics. Prentice Hall, 2001.

[3] Keith Conrad. Differentiating under the integral sign. `http://www.math.uconn.edu/~kconrad/blurbs/analysis/diffunderint.pdf`. Accessed: 2014-08-13.

[4] Axel Munk and Mihaela Pricop. On the self-regularization property of the em algorithm for poisson inverse problems. In Thomas Kneib and Gerhard Tutz, editors, *Statistical Modelling and Regression Structures*, pages 431–448. Physica-Verlag HD, 2010.