TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

D I P L O M A R B E I T

Hardening the BMV Conjecture

ausgeführt am Institut für
Wirtschatfsmathematik
der Technischen Universität Wien

unter Anleitung von a.o.Univ.Prof. Josef Teichmann

durch
Georg Grafendorfer
Engerthstraße 234/8/26
Wien - Leopoldstadt

Datum Unterschrift

This work is dedicated to everybody who finds it to be useful, use it but don't abuse it.

Thank you Josef for your supervision and for being passionate about this work so much. Special thanks to my family for "inviting me to dinner" so often.

The entire work (book and software) was done on a machine equipped exclusively with free and open source software, therefore my special thanks goes to everybody who contributes to the wonderful world of open source, it's easy, just use it wherever it's possible and share your experience with other people, the following software components (and many others they depend upon) provided by Debian GNU/Linux *were used:*

EMACS*, the one and only "all in one" editor.*
AUCTEX*, a convenient LaTeX extension for Emacs.*
TEXLIVE*, a comprehensive LaTeX distribution for a professional looking document.*
GCC*, the GNU Compiler Collection.*
GLIBC*, the GNU C Library, nothing works without it!*
GMP*, the GNU Multiple Precision Library for arbitrary precision data types and functions.*
GTK+ (GLIB, PANGO)*, the Gimp Tool Kit for a fancy graphical user interface.*
CAIRO*, a 2D graphic library.*
TEXIFY*, a little tool which formats source code into a LaTeX document.*

Abstract

This work is based on the results of the paper "A hyper-geometric approach to the BMV-conjecture" by Michael Drmota, Walter Schachermayer and Josef Teichmann, in the following text referred to as [DST]. First I will introduce the conjecture itself and some basic facts about it followed by the results of the aforementioned paper, especially a formula to calculate the density function of a measure on $\mathbb{R}_{\geq 0}$. This measure respectively its density function plays the main role, because if it turns out to be positive, in this case it would prove the BMV conjecture in a special case. And that's actually what this is all about, mainly to calculate the density function based on the provided formula as a base or starting point for further investigations whether it's possible to find negative function values or not.

To do so I first provide some further simplifications of the given formula (mainly for faster computation), then some estimations needed by the following description of the chosen error strategy. In the last chapter the results of the computation are presented. In the first part of the appendix the source code of the program, written in C is listed, in the second part the results are verified.

Contents

1 Introduction

1.1 The BMV conjecture

Conjecture. *Let A, B be complex hermitian $d \times d$ matrices ($d \in \mathbb{N}^{\times}$) and $B \geq 0$. The trace function*

$$\mathbb{C} \to \mathbb{C} \qquad z \mapsto \Phi^{A,B}(z) := \operatorname{tr}\left(e^{A-zB}\right)$$

is the Laplace transform of a positive measure $\nu^{A,B}$ supported by $[0,\infty[$, i. e.

$$\operatorname{tr}\left(e^{A-zB}\right) = \int_0^\infty e^{-zx} d\nu^{A,B}(x).$$

This is the so called BMV conjecture, to date neither a proof nor a counterexample has been found. Originally it was formulated more generally in 1975 by Bessis, Mousa and Villani (BMV), namely with the functions $z \to \langle v, e^{A-zB}v\rangle$ for each eigenvector v of B instead of the trace function above, but it turned out to fail soon, a counterexample is provided in the appendix of [DST]. In the following we will discuss some well known facts about the BMV conjecture, first to mention is that under the assumptions of the conjecture there is indeed a possibly signed measure, which we will from now on denote by $\nu^{A,B}$, so that

$$\operatorname{tr}\left(e^{A-zB}\right) = \int_0^\infty e^{-zx} d\nu^{A,B}(x) \tag{1}$$

holds true. Therefore it "only" lacks to prove that $\nu^{A,B}$ is positive. Essential to further investigations is the following theorem concerning $\nu^{A,B}$.

Theorem 1. *Let $A = (a_{ij})$ be a hermitian $d \times d$ matrix and $B = \operatorname{diag}(b_1, \ldots, b_d)$ where $b_1, \ldots, b_d \in \mathbb{R}_+$ are mutually different. Then there exists a piecewise continuous function $\psi^{A,B}$ with possible discontinuities at b_i and support in $[\min_i b_i, \max_i b_i]$ so that*

$$\nu^{A,B} = \sum_{i=1}^{d} e^{a_{ii}} \delta_{b_i} + \psi^{A,B}\beta \tag{2}$$

i. e. $\nu^{A,B}$ is a signed measure decomposing into an singular and absolutely continuous part with density function $\psi^{A,B}$ with respect to the Lebesgue measure β on $\mathbb{R}$.

We will see that the assumptions of this theorem do not mean any restrictions to the case of dimension 3. To show this we first recall that a complex $d \times d$ matrix C is hermitian if and only if C is unitary similar to a real diagonal matrix, i. e. if and only if there is a unitary matrix U ($U^H U = UU^H = I := \operatorname{diag}(1, \ldots, 1)$) so that $\widetilde{C} := U^H C U$ is a real diagonal matrix , $\widetilde{C}$ is unique except for the order of the elements (eigenvalues of C) which can be chosen arbitrary since this is just another unitary transformation, in this case $C \geq 0$ means that all eigenvalues of C ($\widetilde{C}$) are non negative. Furthermore we have to know that for an arbitrary matrix A and a regular matrix R there is $e^{R^{-1}AR} = R^{-1}e^A R$, now let's choose a unitary matrix U so that $\widetilde{B} := U^H B U$ is a diagonal matrix with real non negative entries, together with the fact that traces of similar matrices coincide, we conclude

$$\operatorname{tr}\left(e^{A-zB}\right) = \operatorname{tr}\left(U^{-1}e^{A-zB}U\right) = \operatorname{tr}\left(e^{U^H(A-zB)U}\right) = \operatorname{tr}\left(e^{U^H AU - z(U^H BU)}\right) = \operatorname{tr}\left(e^{\widetilde{A}-z\widetilde{B}}\right)$$

where $\widetilde{A} := U^H A U$ is again a hermitian matrix. For the next equation note that $e^{\operatorname{diag}(z_1,\ldots,z_n)} = \operatorname{diag}(e^{z_1},\ldots,e^{z_n})$ $(z_1,\ldots,z_n \in \mathbb{C})$, $z\operatorname{tr}(E) = \operatorname{tr}(zE) = \operatorname{tr}(E\operatorname{diag}(z,\ldots,z))$ $(E$ arbitrary, $z \in \mathbb{C})$ and the valid relation $e^{F+G} = e^F + e^G$ for arbitrary commuting matrices F and G, we set $b := \min_i b_i$ where b_i $(i = 1,\ldots,d)$ are the diagonal entries of $\widetilde{B}$ and conclude

$$\begin{aligned} e^{zb}\operatorname{tr}\left(e^{\widetilde{A}-z\widetilde{B}}\right) &= \operatorname{tr}\left(e^{\widetilde{A}-z\widetilde{B}}\operatorname{diag}(e^{zb},\ldots,e^{zb})\right) \\ &= \operatorname{tr}\left(e^{\widetilde{A}-z\widetilde{B}}e^{\operatorname{diag}(zb,\ldots,zb)}\right) = \operatorname{tr}\left(e^{\widetilde{A}-z(\widetilde{B}-\operatorname{diag}(b,\ldots,b))}\right). \end{aligned} \tag{3}$$

Summarizing all this together means that in the general case we can assume without loss of generality $B = \operatorname{diag}(0,b_2,\ldots,b_d)$ $(b_2,\ldots,b_d \geq 0)$. The last well known fact we mention here is the following:

- The BMV conjecture holds true if
  - B has at most 2 different eigenvalues, i. e. especially in the cases $d = 1,2$.
  - $AB = BA$, for example if both A and B are diagonal matrices.

This means especially in the case of dimension 3 that due to the first statement we can assume that all 3 eigenvalues are different, hence together with the former considerations in this case we can apply Theorem 1. In particular this means that for $d = 3$ to prove the BMV conjecture it remains to prove the non-negativity of $\psi^{A,B}$ under the assumption $B = \operatorname{diag}(0,b_2,b_3)$ $(b_2,b_3 > 0, b_2 \neq b_3)$. And that's exactly what the rest of this work is all about, namely to investigate the non-negativity of $\psi^{A,B}$ for the real case in dimension 3. To do so we will first have a look at some further results about the BMV conjecture in the mentioned case which are specific to [DST], mainly a formula to calculate $\psi^{A,B}$, and then the fun begins.

1.2 Further Results

In this section the main results of [DST], which build the starting point for this work, will be presented. We restrict our self to the real case in Dimension 3 where we can assume without loss of generality $B = \operatorname{diag}(0,b_2,b_3)$ with the eigenvalues of B in arbitrary order. The fact that the case of at most 2 different eigenvalues already is proved leads us to the following global commitment:

> For the rest of this document we restrict our self to the real case in Dimension 3:
> $A = (a_{ij})$ is a real symmetric 3×3 matrix and $B = \operatorname{diag}(0,b_2,b_3)$ with $0 < b_3 < b_2$.

i. e. the case which remains to be proved, to show that the BMV conjecture holds true in the case of two real symmetric 3×3 matrices A and B. The following notations are adopted from [DST], for $x \in]0,b_3[$ we define:

$$\begin{aligned} x_2 &:= x/b_2 & x_3 &:= x/b_3 \\ A_{12} &:= a_{12}\sqrt{x_2} & A_{13} &:= a_{13}\sqrt{x_3} \end{aligned}$$

$$\lambda := a_{11}(x_2 - x_3) - a_{22}x_2 + a_{33}x_3$$

$$
\begin{aligned}
\mu &:= a_{11}(1-x_2)+a_{22}x_2 \\
\upsilon &:= \frac{2A_{12}A_{13}}{A_{12}^2+A_{13}^2} \qquad (\Rightarrow |\upsilon| \le 1) \\
\xi &:= a_{23}\sqrt{x_2x_3} \\
\omega_1 &:= \frac{1-x_3}{2}(A_{12}^2+A_{13}^2) \qquad (\Rightarrow \omega_1 \ge 0) \\
\omega_2 &:= \frac{x_3-x_2}{2(1-x_3)} \qquad (\Rightarrow \omega_2 > 0) \\
\omega_3 &:= -\frac{A_{13}}{A_{12}}\upsilon \qquad (\Rightarrow |\omega_3| \le 2).
\end{aligned}
$$

Sufficient for our needs let $n \in \mathbb{N}$, $x > -n$ and define the *rising factorial* by $(x)_n := x(x+1)\cdots(x+n-1)$ if $n \ge 1$ and $(x)_0 := 1$ for $n = 0$ (More generally this is $\frac{\Gamma(x+n)}{\Gamma(x)}$ where $\Gamma(x)$ as usual denotes the Gamma function defined on $\mathbb{C}\backslash\{-1,-2,\dots\}$. For the following we set $\frac{1}{n!} := \lim_{x\to n}\frac{1}{\Gamma(x+1)} = 0$ $(n = -1,-2,\dots)$:

$$
T(k,r,\rho;\upsilon,\xi) := \sum_{m\ge 0}\sum_{j=0}^{k}\binom{k}{j}\frac{2^{k+r-\rho-1}(\frac{m-j+2}{2})_{k+r-\rho-1}}{(m+1)_{k+r-1}}\frac{(k-j)!}{(k-j-\rho)!}\upsilon^j\frac{\xi^m}{m!} \tag{4}
$$

$$
\widetilde{T}(k,r,\rho;\upsilon,\xi) := \frac{1}{2}(T(k,r,\rho;\upsilon,\xi)+T(k,r,\rho;-\upsilon,-\xi)).
$$

Result 1. *For $|\upsilon| \le 1$ there is*

$$
\widetilde{T}(k,r,\rho;\upsilon,\xi) \ge 0.
$$

Result 2. *For $x \in]0,b_3[$ there is*

$$
\begin{aligned}
\psi^{A,B}(x) = \frac{2e^{\lambda+\mu}}{x(1-x_3)}\sum_{k\ge 1}\frac{\omega_1^k}{k!(k-1)!}\sum_{r=0}^{k-1}\binom{k-1}{r}\omega_2^r\sum_{L\ge 0}\frac{(-\lambda)^L}{L!} \\
\times\sum_{\rho=0}^{r+L}\binom{r+L}{\rho}\widetilde{T}(k,r+L,\rho;\upsilon,\xi)\ \omega_3^\rho.
\end{aligned} \tag{5}
$$

For $x \in]b_3,b_2[$ we have to perform a transformation to apply result 2 to the transformed matrices. For these purposes we first apply equation (1) together with Theorem 1 to our matrices A and B which results in

$$
\operatorname{tr}\left(e^{A-zB}\right) = \int_0^{b_2} e^{-zx}\psi^{A,B}(x)dx + e^{a_{11}} + e^{a_{22}}e^{-zb_2} + e^{a_{33}}e^{-zb_3}
$$

multiplying with e^{zb_2} yields (remember equation (3))

$$
\operatorname{tr}\left(e^{A-z(B-\operatorname{diag}(b_2,b_2,b_2))}\right) = \int_0^{b_2} e^{-z(x-b_2)}\psi^{A,B}(x)dx + e^{a_{11}}e^{zb_2} + e^{a_{22}} + e^{a_{33}}e^{-z(b_3-b_2)}.
$$

As this equation is valid for $z \in \mathbb{C}$ we can substitute z by $-z$, additionally we perform a variable transformation in the integral term by setting $x = b_2 - y$, together we have

$$
\operatorname{tr}\left(e^{A-z(\operatorname{diag}(b_2,b_2,b_2)-B)}\right) = \int_0^{b_2} e^{-zy}\psi^{A,B}(b_2-y)dy + e^{a_{11}}e^{-zb_2} + e^{a_{22}} + e^{a_{33}}e^{-z(b_2-b_3)}. \tag{6}
$$

Let us now perform the same orthogonal transformation to both, the matrices A and $\mathrm{diag}(b_2,b_2,b_2)-B$ by changing indices 1 and 2 and denote the result by $\widetilde{A}$ and $\widetilde{B}$ respectively, thus

$$\widetilde{A}=\begin{pmatrix} a_{22} & a_{12} & a_{23} \\ a_{12} & a_{11} & a_{13} \\ a_{23} & a_{13} & a_{33} \end{pmatrix} \qquad \widetilde{B}=\begin{pmatrix} 0 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & b_2-b_3 \end{pmatrix}.$$

If we now apply Theorem 1 and equation (1) to the matrices $\widetilde{A}$ and $\widetilde{B}$ (the same way as we did to A and B above) we obtain

$$\mathrm{tr}\left(e^{\widetilde{A}-z\widetilde{B}}\right)=\int_0^{b_2} e^{-zx}\psi^{\widetilde{A},\widetilde{B}}(x)dx+e^{a_{22}}+e^{a_{11}}e^{-zb_2}+e^{a_{33}}e^{-z(b_2-b_3)}.$$

Together with $\mathrm{tr}\left(e^{\widetilde{A}-z\widetilde{B}}\right)=\mathrm{tr}\left(e^{A-z(\mathrm{diag}(b_2,\ldots,b_2)-B)}\right)$ and equation (6) we derive

$$\int_0^{b_2} e^{-zx}\psi^{A,B}(b_2-x)dx=\int_0^{b_2} e^{-zx}\psi^{\widetilde{A},\widetilde{B}}(x)dx \quad (z\in\mathbb{C})$$

which finally yields

$$\psi^{A,B}(b_2-x)=\psi^{\widetilde{A},\widetilde{B}}(x) \quad \text{for} \quad x\in]0,b_2[.$$

Note that b_2-x lies within $]0,b_2-b_3[=]0,\tilde{b}_3[$ if $x\in]b_3,b_2[$. Thus for $x\in]b_3,b_2[$ we can apply result 2 to the matrices $\widetilde{A}$ and $\widetilde{B}$ by means of

$$\psi^{A,B}(x)=\psi^{\widetilde{A},\widetilde{B}}(b_2-x).$$

Here we are, we are now able to calculate the density function $\psi^{A,B}$ of Theorem 1 for $x\in]0,b_3[\cup]b_3,b_2[$. Non negativity of this function for all matrices A and B of the form assumed above would prove the BMV conjecture in the real case of dimension 3. That's why the rest of this work is dedicated to the calculation of $\psi^{A,B}$.

Another result from [DST] is mentioned here without a derivation.

Result 3. *$\psi^{A,B}$ is non-negative if $a_{12}a_{13}a_{23}\geq 0$.*

2 Computation

Let's split $\psi^{A,B}(x)$ into 3 factors, for a complete representation of $\psi_3^{A,B}(x)$ look at Figure 1 on page 6:

$$\psi^{A,B}(x) = \underbrace{\frac{2}{x(1-x_3)}}_{=:\psi_1^{A,B}(x)} \quad \underbrace{e^{\lambda+\mu}}_{=:\psi_2^{A,B}(x)} \quad \underbrace{\sum_{k=1}^{\infty}\dots}_{=:\psi_3^{A,B}(x)} .$$

It's easy to see that the third factor, the so called main factor, is the most difficult to handle, so we will put our focus on it, the first two factors are quite easy to handle as we will see later. We begin with some simplifications on the main factor merely concerning the index areas of the some finite sums to speed up calculation, unfortunately despite of some effort I could not find a way to profit from this simplifications with respect to the estimations.

2.1 Simplification

Let's define for $n \in \mathbb{N}$ and $\frac{x}{2} > -n$:

$$[x]_n := 2^n \left(\frac{x}{2}\right)_n \quad \text{thus for } n \in \mathbb{N}^\times \;\; [x]_n = x(x+2)\cdots(x+2(n-1)).$$

Inserting into (4) together with $\binom{k}{j}(k-j)! = \frac{k!}{j!}$ yields

$$\begin{aligned}\widetilde{T}(k,r,\rho;\upsilon,\xi) &= k!\,\frac{1}{2}\left(\sum_{m\geq 0}\frac{1}{(m+1)_{k+r-1}}\frac{\xi^m}{m!}\sum_{j=0}^{k}\frac{[m-j+2]_{k+r-\rho-1}}{(k-j-\rho)!}\frac{\upsilon^j}{j!}\right.\\ &\qquad \left. + \sum_{m\geq 0}\frac{1}{(m+1)_{k+r-1}}\frac{(-\xi)^m}{m!}\sum_{j=0}^{k}\frac{[m-j+2]_{k+r-\rho-1}}{(k-j-\rho)!}\frac{(-\upsilon)^j}{j!}\right)\\ &= k!\sum_{m\geq 0}\frac{1}{(m+1)_{k+r-1}}\frac{\xi^m}{m!}\sum_{\substack{0\leq j\leq k\\ j\equiv m \bmod 2}}\frac{[m-j+2]_{k+r-\rho-1}}{(k-j-\rho)!}\frac{\upsilon^j}{j!}.\end{aligned}$$

For the purpose of the numerical computation we will keep as many factors as possible (with some expectations) outside the loops, namely those which do not depend upon the running index, the factor $k!$ for example we are going to cancel together with the factor $\frac{1}{k!}$ in (5), as opposed to the estimation where it's somewhat better to keep this factor inside the most inner sum, as we will see later.

Remark. For $1 \leq k$ and $0 \leq \rho \leq r$ (3^{rd} argument of $\widetilde{T} \leq 2^{nd}$ argument of $\widetilde{T}$) there is

$$\widetilde{T}(k,r,\rho;\upsilon,\xi) = k!\sum_{m\geq 0}\frac{1}{(m+1)_{k+r-1}}\frac{\xi^m}{m!}\sum_{\substack{0\leq j\leq \min(m,k-\rho)\\ j\equiv m \bmod 2}}\frac{[m-j+2]_{k+r-\rho-1}}{(k-j-\rho)!}\frac{\upsilon^j}{j!} \quad (\rho \leq k)$$

$$\widetilde{T}(k,r,\rho;\upsilon,\xi) = 0 \quad (\rho > k).$$

$$\sum_{k\geq 1}\frac{\omega_1^k}{(k-1)!}\underbrace{\sum_{r=0}^{k-1}\binom{k-1}{r}\omega_2^r\underbrace{\sum_{L\geq 0}\frac{(-\lambda)^L}{L!}\underbrace{\sum_{\rho=0}^{\min(r+L,k)}\binom{r+L}{\rho}\omega_3^\rho\overbrace{\underbrace{\sum_{m\geq 0}\frac{1}{(m+1)_{k+r+L-1}}\frac{\xi^m}{m!}\underbrace{\sum_{\substack{0\leq j\leq \min(m,k-\rho)\\ j\equiv m \bmod 2}}\frac{[m-j+2]_{k+r+L-\rho-1}}{(k-j-\rho)!}\frac{\upsilon^j}{j!}}_{=:E_j(k,r+L,\rho,m;\upsilon)}}_{=:E_m(k,r+L,\rho;\upsilon,\xi)}}^{=\frac{1}{k!}\widetilde{T}(k,r+L,\rho;\upsilon,\xi)}}_{=:E_\rho(k,r+L;\upsilon,\xi,\omega_3)}}_{=:E_L(k,r;\upsilon,\xi,\omega_3,\lambda)}}_{=:E_r(k;\upsilon,\xi,\omega_3,\lambda,\omega_2)}}_{=:E_k(\upsilon,\xi,\omega_3,\lambda,\omega_2,\omega_1)}$$

$$\sum_{k\geq 1}\sum_{r=0}^{k-1}\sum_{L\geq 0}\sum_{\rho=0}^{\min(r+L,k)}\sum_{m\geq 0}\sum_{\substack{0\leq j\leq \min(m,k-\rho)\\ j\equiv m \bmod 2}}\underbrace{\frac{\omega_1^k}{(k-1)!}}_{=:\hat{E}_k}\underbrace{\binom{k-1}{r}\omega_2^r}_{=:\hat{E}_r}\underbrace{\frac{(-\lambda)^L}{L!}}_{=:\hat{E}_L}\underbrace{\binom{r+L}{\rho}\omega_3^\rho}_{=:\hat{E}_\rho}\frac{1}{(m+1)_{k+r+L-1}}\frac{\xi^m}{m!}\frac{[m-j+2]_{k+r+L-\rho-1}}{(k-j-\rho)!}\frac{\upsilon^j}{j!}$$

Figure 1: *Road Map, two different representations of* $\psi_3^{A,B}(x)$.

Proof. Let's check out first if $[m-j+2]_{k+r-\rho-1}$ is well defined, i. e. if there is $\frac{m-j+2}{2} > -(k+r-\rho-1)$, due to $\rho \leq r$ and $j \leq k$ it follows that $m-j+2k+2r-2\rho \geq m+k \geq 1$, so we have

$$[m-j+2]_{k+r-\rho-1} = (m-j+2)(m-j+2+2)\cdots(m-j+2+2(k+r-\rho-2)).$$

The last factor is already proved to be greater than 0, also note that $m-j$ is even, so the whole term equals 0 if $j \geq m+2$, in this case there is $k+r-\rho-1 > 0$ so that we can conclude $[m-j+2]_{k+r-\rho-1} = 0$ for $j \geq m+2$. From this and $\frac{1}{(k-j-\rho)!} = 0$ for $j > k-\rho$ we follow that it's sufficient for the index j to run from 0 to $\min(m, k-\rho)$ instead from 0 to k. The second statement is a consequence of the first one. ☯

Corollary. *The index ρ (look at equation (5)) can run from* 0 *to* $\min(r+L,k)$ *instead.* ☯

2.2 Estimation

We keep on focusing at the main factor and going to estimate step by step from the inside to the most outer sum to afterward provide a strategy for computing the sum within some desired error margins. Note that from [DST] we already know $0 \leq \widetilde{T}(k,r,\rho;\upsilon,\xi)$ for $|\upsilon| \leq 1$. Due to better estimation results (in some cases) we do not simplify $1+|\upsilon| \leq 2$ or $2+|\omega_3| \leq 4$. A kind of clearer representation especially for nested terms provides the following notation:

$$\begin{Bmatrix} x \\ y \end{Bmatrix} := \min(x,y)$$

given two (ore more) terms x and y. The following remark will be used frequently in the forthcoming text.

Remark. Let $n \geq 0$ and $x \in \mathbb{R}$. Then we have

$$\sum_{k=n+1}^{\infty} \frac{x^k}{k!} \leq \frac{|x|^{n+1}}{(n+1)!} e^{|x|}. \tag{7}$$

Proof.

$$\frac{|x|^{n+1}}{(n+1)!} e^{|x|} = \frac{|x|^{n+1}}{(n+1)!} \sum_{k=0}^{\infty} \frac{|x|^k}{k!} = \sum_{k=0}^{\infty} \frac{|x|^{n+1+k}}{(n+1)!k!} \geq \sum_{k=0}^{\infty} \frac{x^{n+1+k}}{(n+1+k)!} = \sum_{k=n+1}^{\infty} \frac{x^k}{k!}.$$ ☯

The next remark will be used in the proof of the first step.

Remark.

$$\frac{(\frac{m}{2}+1)_{N-d}}{(m+1)_N} \leq \frac{1}{(m+N-d+1)^d} \qquad 0 \leq m \quad 0 \leq d \leq N \tag{i}$$

$$\frac{(N-d_1)!}{(N-d_1-d_2)!} \leq \min(N!, N^{d_2}) \qquad 1 \leq N \quad 0 \leq d_1 \leq N \quad 0 \leq d_2. \tag{ii}$$

Proof. (i) For $N = 0$ it's obvious, for $N > 0$ we have

$$\frac{(\frac{m}{2}+1)_{N-d}}{(m+1)_N} \leq \frac{(m+1)_{N-d}}{(m+1)_N} = \frac{1}{(m+N-d+1)\cdots(m+N)} \leq \frac{1}{(m+N-d+1)^d}.$$

(ii) $\frac{(N-d_1)!}{(N-d_1-d_2)!} \le N!$ is clear, it remains to prove $\frac{(N-d_1)!}{(N-d_1-d_2)!} \le N^{d_2}$: for both cases $d_2 = 0$ or $d_1+d_2 > N$ it's obvious, so let's assume $d_2 > 0$ and $d_1+d_2 \le N$, thus we have $\frac{(N-d_1)!}{(N-d_1-d_2)!} = (N-d_1)\cdots(N-d_1-d_2+1) \le (N-d_1)^{d_2} \le N^{d_2}$. ☯

Here the estimation starts, the definitions of the symbols used in the following can be found in the road map, Figure 1 on page 6.

Step 1. For $0 \le \rho \le r$ and $1 \le k$ there is

$$\sum_{\substack{0\le j\le \min(m,k-\rho)\\ j\equiv m \bmod 2}} |E_j(k,r,\rho,m;\upsilon)| \le [m+2]_{k+r-\rho-1} \left\{ \begin{matrix} (1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|} \end{matrix} \right\}.$$

Proof. There are two inequalities to prove, we will prove one more inequality (just to mention it here) which has at most theoretical importance, the first two are proved by

$$\sum_{\substack{0\le j\le \min(m,k-\rho)\\ j\equiv m \bmod 2}} |E_j(k,r,\rho,m;\upsilon)| = \sum_{\substack{0\le j\le \min(m,k-\rho)\\ j\equiv m \bmod 2}} \frac{[m-j+2]_{k+r-\rho-1}}{(k-j-\rho)!} \frac{|\upsilon|^j}{j!}$$
$$= \frac{1}{k!} \sum_{\substack{0\le j\le \min(m,k-\rho)\\ j\equiv m \bmod 2}} \binom{k}{j} [m-j+2]_{k+r-\rho-1} \frac{(k-j)!}{(k-j-\rho)!} |\upsilon|^j.$$

From the remark follows

$$\le \frac{1}{k!} [m+2]_{k+r-\rho-1} \min(k!, k^\rho) \sum_{0\le j\le k} \binom{k}{j} |\upsilon|^j$$
$$= [m+2]_{k+r-\rho-1} \min\left(1, \frac{k^\rho}{k!}\right) (1+|\upsilon|)^k.$$

The third inequality follows from

$$\sum_{\substack{0\le j\le \min(m,k-\rho)\\ j\equiv m \bmod 2}} \frac{[m-j+2]_{k+r-\rho-1}}{(k-j-\rho)!} \frac{|\upsilon|^j}{j!} \le [m+2]_{k+r-\rho-1} e^{|\upsilon|}$$

All together we have

$$\sum_{\substack{0\le j\le \min(m,k-\rho)\\ j\equiv m \bmod 2}} |E_j(k,r,\rho,m;\upsilon)| \le [m+2]_{k+r-\rho-1} \left\{ \begin{matrix} (1+|\upsilon|)^k \left\{ \begin{matrix} 1 \\ \frac{k^\rho}{k!} \end{matrix} \right\} \\ e^{|\upsilon|} \end{matrix} \right\}$$ ☯

Before we continue lets take a look at the first inequality to see that it has no practical importance, hence is just mentioned here for the sake of theoretical completeness. Let's compare the first inequality with the third one: for $k=1$ we have $1+|\upsilon| \le e^{|\upsilon|}$ and thus the first one beats the third one, but already for $k=2$ there is (remember $|\upsilon| \le 1$) $e^{|\upsilon|} \le 1+|\upsilon|+\frac{|\upsilon|^2}{2!}+\frac{|\upsilon|^3}{3!}e \le 1+|\upsilon|+\frac{|\upsilon|^2}{2}+\frac{|\upsilon|^3}{2} \le 1+2|\upsilon|+|\upsilon|^2 = (1+|\upsilon|)^2$, thus for $k=2$ and more than ever for $k \ge 2$ the third inequality beats the first one. Therefore we will not carry on with the first inequality, because just for the case $k=1$ it is not worth the effort, moreover it isn't a much better estimation.

Step 2. For $0 \le \rho \le r$, $1 \le k$ and $0 \le m_{\max}$ there is

$$\sum_{m=m_{\max}+1}^{\infty} |E_m(k,r,\rho;\upsilon,\xi)| \le \left\{ \begin{matrix} (1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|} \end{matrix} \right\} \frac{2^{k+r-\rho-1} e^{|\xi|}}{(m_{\max}+1+k+r-\rho)^\rho} \frac{|\xi|^{m_{\max}+1}}{(m_{\max}+1)!}$$

$$\sum_{m\ge 0} |E_m(k,r,\rho;\upsilon,\xi)| \le \left\{ \begin{matrix} (1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|} \end{matrix} \right\} \frac{2^{k+r-\rho-1}}{(k+r-\rho)^\rho} e^{|\xi|}.$$

Proof.

$$\sum_{m=m_{\max}+1}^{\infty} |E_m(k,r,\rho;\upsilon,\xi)| = \sum_{m=m_{\max}+1}^{\infty} \left| \frac{1}{(m+1)_{k+r+-1}} \frac{\xi^m}{m!} \sum_{\substack{0\le j\le \min(m,k-\rho) \\ j\equiv m \mod 2}} E_j(k,r,\rho,m;\upsilon) \right|$$

$$\le \sum_{m=m_{\max}+1}^{\infty} \frac{1}{(m+1)_{k+r-1}} \frac{|\xi|^m}{m!} \sum_{\substack{0\le j\le \min(m,k-\rho) \\ j\equiv m \mod 2}} |E_j(k,r,\rho,m;\upsilon)|$$

$$\le \left\{ \begin{matrix} (1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|} \end{matrix} \right\} \sum_{m=m_{\max}+1}^{\infty} \frac{[m+2]_{k+r-\rho-1}}{(m+1)_{k+r-1}} \frac{|\xi|^m}{m!}.$$

From the remark follows (note that $\rho \le k+r-1$)

$$\le 2^{k+r-\rho-1} \left\{ \begin{matrix} (1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|} \end{matrix} \right\} \sum_{m=m_{\max}+1}^{\infty} \frac{1}{(m+k+r-\rho)^\rho} \frac{|\xi|^m}{m!}$$

$$\le \left\{ \begin{matrix} (1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|} \end{matrix} \right\} \frac{2^{k+r-\rho-1}}{(m_{\max}+1+k+r-\rho)^\rho} \frac{|\xi|^{m_{\max}+1}}{(m_{\max}+1)!} e^{|\xi|}.$$

For the proof of the second part we just change $\sum_{m=m_{\max}+1}^{\infty}$ to $\sum_{m\ge 0}$ and instead the last inequality we write

$$2^{k+r-\rho-1} \left\{ \begin{matrix} (1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|} \end{matrix} \right\} \sum_{m\ge 0} \frac{1}{(m+k+r-\rho)^\rho} \frac{|\xi|^m}{m!}$$

$$\le \left\{ \begin{matrix} (1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|} \end{matrix} \right\} \frac{2^{k+r-\rho-1}}{(k+r-\rho)^\rho} e^{|\xi|}.$$

☯

Step 3. For $1 \le k$ there is

$$\sum_{\rho=0}^{\min(r,k)} |E_\rho(k,r;\upsilon,\xi,\omega_3)| \le 2^{k-1} e^{|\xi|} \left\{ \begin{matrix} \frac{1}{k!}(1+|\upsilon|)^k \left(2+\frac{|\omega_3|k}{\max(k,r)}\right)^r \\ e^{|\upsilon|}\left(2+\frac{|\omega_3|}{\max(r,k)}\right)^r \end{matrix} \right\}.$$

Proof.

$$\sum_{\rho=0}^{\min(r,k)} |E_\rho(k,r;\upsilon,\xi,\omega_3)| = \sum_{\rho=0}^{\min(r,k)} \left| \binom{r}{\rho} \omega_3^\rho \sum_{m\ge 0} E_m(k,r,\rho;\upsilon,\xi) \right|$$

$$\leq \sum_{\rho=0}^{\min(r,k)} \binom{r}{\rho} |\omega_3|^\rho \sum_{m\geq 0} |E_m(k,r,\rho;\upsilon,\xi)|$$
$$\leq \sum_{\rho=0}^{r} \binom{r}{\rho} |\omega_3|^\rho \left\{ \begin{matrix} (1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|} \end{matrix} \right\} \frac{2^{k+r-\rho-1}}{(k+r-\rho)^\rho} e^{|\xi|}.$$

Note that $\frac{1}{k+r-\rho} \leq \frac{1}{k+r-\min(r,k)} = \frac{1}{\max(k,r)}$.

$$\leq e^{|\xi|} 2^{k-1} \sum_{\rho=0}^{r} \binom{r}{\rho} |\omega_3|^\rho \left\{ \begin{matrix} (1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|} \end{matrix} \right\} \frac{2^{r-\rho}}{\max(k,r)^\rho}$$

Continuing with the first inequality we have

$$\leq e^{|\xi|} 2^{k-1} (1+|\upsilon|)^k \frac{1}{k!} \sum_{\rho=0}^{r} \binom{r}{\rho} |\omega_3|^\rho k^\rho \frac{2^{r-\rho}}{\max(k,r)^\rho}$$
$$= e^{|\xi|} 2^{k-1} (1+|\upsilon|)^k \frac{1}{k!} \left(2 + \frac{|\omega_3| k}{\max(k,r)}\right)^r.$$

For the second inequality we have

$$\leq e^{|\xi|} 2^{k-1} e^{|\upsilon|} \sum_{\rho=0}^{r} \binom{r}{\rho} |\omega_3|^\rho \frac{2^{r-\rho}}{\max(k,r)^\rho}$$
$$= e^{|\xi|} 2^{k-1} e^{|\upsilon|} \left(2 + \frac{|\omega_3|}{\max(k,r)}\right)^r. \qquad \text{☯}$$

Now let's check out which estimation is the better one, we divide the upper term by the lower term (note that $|\upsilon| \leq 1$):

$$\frac{\frac{1}{k!}(1+|\upsilon|)^k \left(2+\frac{|\omega_3|k}{\max(k,r)}\right)^r}{e^{|\upsilon|}\left(2+\frac{|\omega_3|}{\max(r,k)}\right)^r} \leq \frac{\frac{1}{k!}(1+|\upsilon|)^k (2+|\omega_3|))^r}{e^{|\upsilon|}\left(2+\frac{|\omega_3|}{\max(r,k)}\right)^r} \leq \frac{\frac{1}{k!}(1+|\upsilon|)^k 4^r}{e^{|\upsilon|} 2^r}$$
$$= \frac{\frac{1}{k!}(1+|\upsilon|)^k 2^r}{e^{|\upsilon|}} \leq \frac{1}{k!} 2^{k+r},$$

in the case $k << r$ we may set $\frac{|\omega_3|k}{\max(k,r)} \approx \frac{|\omega_3|}{\max(k,r)} = \frac{|\omega_3|}{r} \approx 0$ (note that $|\omega_3| \leq 2$) and derive

$$\frac{\frac{1}{k!}(1+|\upsilon|)^k \left(2+\frac{|\omega_3|k}{\max(k,r)}\right)^r}{e^{|\upsilon|}\left(2+\frac{|\omega_3|}{\max(r,k)}\right)^r} \approx \frac{\frac{1}{k!}(1+|\upsilon|)^k}{e^{|\upsilon|}} \leq \frac{1}{k!} 2^k.$$

In both cases we see that with rising k the upper term is becoming very soon a much better approximation, therefore we will not carry on with the second inequality but concentrate on the first one.

Step 4. For $1 \le k$, $0 \le L_{\max}$, $\gamma := 2 + \frac{|\omega_3| k}{\max(k, r+L_{\max}+1)}$ and $\delta := 2 + \frac{|\omega_3| k}{\max(k,r)}$ there is

$$\sum_{L=L_{\max}+1}^{\infty} |E_L(k,r;\upsilon,\xi,\omega_3,\lambda)| \le e^{|\xi|+|\lambda|\gamma}\, 2^{k-1} \frac{1}{k!}(1+|\upsilon|)^k \gamma^r \frac{(|\lambda|\gamma)^{L_{\max}+1}}{(L_{\max}+1)!}$$

$$\sum_{L\ge 0} |E_L(k,r;\upsilon,\xi,\omega_3,\lambda)| \le e^{|\xi|+|\lambda|\delta}\, 2^{k-1} \frac{1}{k!}(1+|\upsilon|)^k \delta^r.$$

Proof.

$$\begin{aligned}
\sum_{L=L_{\max}+1}^{\infty} |E_L(k,r;\upsilon,\xi,\omega_3,\lambda)| &= \sum_{L=L_{\max}+1}^{\infty} \left| \frac{(-\lambda)^L}{L!} \sum_{\rho=0}^{\min(r+L,k)} E_\rho(k,r+L;\upsilon,\xi,\omega_3) \right| \\
&\le \sum_{L=L_{\max}+1}^{\infty} \frac{|\lambda|^L}{L!} \sum_{\rho=0}^{\min(r+L,k)} |E_\rho(k,r+L;\upsilon,\xi,\omega_3)| \\
&\le \sum_{L=L_{\max}+1}^{\infty} \frac{|\lambda|^L}{L!} 2^{k-1} e^{|\xi|} \frac{1}{k!}(1+|\upsilon|)^k \left(2 + \frac{|\omega_3| k}{\max(k, r+L)}\right)^{r+L} \\
&\le 2^{k-1} e^{|\xi|} \frac{1}{k!}(1+|\upsilon|)^k \sum_{L=L_{\max}+1}^{\infty} \frac{|\lambda|^L}{L!} \left(2 + \frac{|\omega_3| k}{\max(k, r+L_{\max}+1)}\right)^{r+L} \\
&= 2^{k-1} e^{|\xi|} \frac{1}{k!}(1+|\upsilon|)^k \gamma^r \underbrace{\sum_{L=L_{\max}+1}^{\infty} \frac{(|\lambda|\gamma)^L}{L!}}_{\le \frac{(|\lambda|\gamma)^{L_{\max}+1}}{(L_{\max}+1)!} e^{|\lambda|\gamma}}.
\end{aligned}$$

For the proof of the second part where L goes from 0 to ∞ we just take δ instead of γ and consider the Taylor expansion of e^x (actually the definition of e^x). ☯

Step 5. For $1 \le k$ there is

$$\sum_{r=0}^{k-1} |E_r(k;\upsilon,\xi,\omega_3,\lambda,\omega_2)| \le e^{|\xi|+|\lambda|(2+|\omega_3|)} \frac{1}{k!}(1+|\upsilon|)^k\, 2^{k-1} \left(1+\omega_2\,(2+|\omega_3|)\right)^{k-1}.$$

Proof.

$$\begin{aligned}
\sum_{r=0}^{k-1} |E_r(k;\upsilon,\xi,\omega_3,\lambda,\omega_2)| &= \sum_{r=0}^{k-1} \left| \binom{k-1}{r} {\omega_2}^r \sum_{L\ge O} E_L(k,r;\upsilon,\xi,\omega_3,\lambda) \right| \\
&\le \sum_{r=0}^{k-1} \binom{k-1}{r} {\omega_2}^r \sum_{L\ge O} |E_L(k,r;\upsilon,\xi,\omega_3,\lambda)| \\
&\le \sum_{r=0}^{k-1} \binom{k-1}{r} {\omega_2}^r e^{|\xi|} 2^{k-1} \frac{1}{k!}(1+|\upsilon|)^k \delta^r e^{|\lambda|\delta} \\
&\le e^{|\xi|} 2^{k-1} e^{|\lambda|(2+|\omega_3|)} \frac{1}{k!}(1+|\upsilon|)^k \underbrace{\sum_{r=0}^{k-1} \binom{k-1}{r} {\omega_2}^r (2+|\omega_3|)^r}_{=(1+\omega_2(2+|\omega_3|))^{k-1}}.
\end{aligned}$$

☯

Step 6. For $1 \leq k_{\max}$ and $\beta := 2\omega_1(1+|\upsilon|)(1+\omega_2(2+|\omega_3|))$ there is

$$\sum_{k=k_{\max}+1}^{\infty} |E_k(\upsilon,\xi,\omega_3,\lambda,\omega_1,\omega_2)| \leq \frac{e^{|\xi|+|\lambda|(2+|\omega_3|)+\beta}}{2(1+\omega_2(2+|\omega_3|))} \frac{\beta^{k_{\max}+1}}{k_{\max}!(k_{\max}+1)!}$$

$$\sum_{k=1}^{\infty} |E_k(\upsilon,\xi,\omega_3,\lambda,\omega_1,\omega_2)| \leq \frac{e^{|\xi|+|\lambda|(2+|\omega_3|)}}{2(1+\omega_2(2+|\omega_3|))}(e^{\beta}-1).$$

Proof.

$$\begin{aligned}
&\sum_{k=k_{\max}+1}^{\infty} |E_k(\upsilon,\xi,\omega_3,\lambda,\omega_1,\omega_2)| = \sum_{k=k_{\max}+1}^{\infty} \left| \frac{{\omega_1}^k}{(k-1)!} \sum_{r=0}^{k-1} E_r(k;\upsilon,\xi,\omega_3,\lambda,\omega_2) \right| \\
&\leq \sum_{k=k_{\max}+1}^{\infty} \frac{{\omega_1}^k}{(k-1)!} \sum_{r=0}^{k-1} |E_r(k;\upsilon,\xi,\omega_3,\lambda,\omega_2)| \\
&\leq \sum_{k=k_{\max}+1}^{\infty} \frac{{\omega_1}^k}{(k-1)!} e^{|\xi|+|\lambda|(2+|\omega_3|)} \frac{1}{k!} (1+|\upsilon|)^k \, 2^{k-1} \, (1+\omega_2\,(2+|\omega_3|))^{k-1} \\
&= e^{|\xi|+|\lambda|(2+|\omega_3|)} \frac{1}{2\,(1+\omega_2\,(2+|\omega_3|))} \sum_{k=k_{\max}+1}^{\infty} \frac{1}{(k-1)!} \frac{(2\omega_1(1+|\upsilon|)\,(1+\omega_2\,(2+|\omega_3|)))^k}{k!} \\
&= e^{|\xi|+|\lambda|(2+|\omega_3|)} \frac{1}{2(1+\omega_2(2+|\omega_3|))} \underbrace{\sum_{k=k_{\max}+1}^{\infty} \frac{\beta^k}{(k-1)!k!}}_{\leq \frac{1}{k_{\max}!} \frac{\beta^{k_{\max}+1}}{(k_{\max}+1)!} e^{\beta}}.
\end{aligned}$$

For $k_{\max} = 0$ we have $\sum_{k=1}^{\infty} \frac{\beta^k}{(k-1)!k!} \leq \sum_{k=1}^{\infty} \frac{\beta^k}{k!} = e^{\beta} - 1$. ☯

2.3 Error Strategy

Actually our main goal is to decide whether $\psi^{A,B}(x)$ is negative or not, to do so we have to perform calculation with an absolute error less than the absolute value of the calculated value $\widetilde{\psi^{A,B}}(x)$, i. e. $|\widetilde{\psi^{A,B}}(x)| > \mathit{abserr}$. As we will see later we are able to provide a calculation which meets $|\psi^{A,B}(x) - \widetilde{\psi^{A,B}}(x)| \leq \mathit{abserr}$ for some arbitrary positive real number *abserr*, but of course we neither know the calculated nor the exact value in advance. The only clue for the value of $\widetilde{\psi^{A,B}}(x)$ or $\psi^{A,B}(x)$ are just some possible calculated adjacent values or the estimated boundary of $|\psi^{A,B}(x)|$ from above. As we are able to provide a upper bound for $|\psi^{A,B}(x)|$ we will use this as a starting point and choose *abserr* less than this upper bound, calculate $\widetilde{\psi^{A,B}}(x)$ and repeat calculation by decreasing *abserr* until it meets the required inequality $|\widetilde{\psi^{A,B}}(x)| > \mathit{abserr}$, it's easy to see that for $\psi^{A,B}(x) \neq 0$ this method succeeds.

So let's step further to calculate $\psi^{A,B}(x)$, the basic idea is to only allow rational inputs to $\psi^{A,B}(x)$ which allows us to compute the first factor exactly, the second one is easy to estimate, for the third factor, since it only contains rational operations, we can at least compute the elements of the sum exact. This way we don't have to bother about additionally error considerations but only have to worry about the infinite sums. This means that we must choose our basic input

variables, i. e. A, B, x, so that all the variables defined in section 1.2 are rational. Since there are only rationals and square root operations this is easy to do and finally without any real restrictions as it's always possible to find such numbers arbitrary close to real inputs A, B and x.

For the second factor we will use a quite simple method based on inequality (7), certainly there are much faster methods, but we focus here on the calculation of the main factor which will take much more time.

In general this looks as follows: We have 3 factors a, b, c with the corresponding errors $\Delta a(=0), \Delta b, \Delta c$ respectively, for $|b|$ and $|c|$ we can provide upper bounds which we will denote by $b_{\max}$ and $c_{\max}$ respectively. Thus we have (note that $a > 0$):

$$\begin{aligned}|\Delta(abc)| &= |abc - a(b+\Delta b)(c+\Delta c)| = |abc - abc - ab\Delta c - a\Delta bc - a\Delta b\Delta c| \\ &\leq a(b_{\max}|\Delta c| + c_{\max}|\Delta b| + |\Delta b||\Delta c|) \leq abserr,\end{aligned}$$

i. e. the following inequality must hold.

$$|\Delta c| + |\Delta b|\frac{(c_{\max} + |\Delta c|)}{b_{\max}} \leq \frac{abserr}{ab_{\max}}.$$

Hence we have to choose $|\Delta b|$ and $|\Delta c|$ appropriate. We keep in mind that the third factor is the one with the most effort to compute, thus we should choose $|\Delta c|$ as big as possible, i. e. close to $\frac{abserr}{ab_{\max}}$. So let's set $|\Delta c| = \frac{9}{10}\frac{abserr}{ab_{\max}}$ which results in $|\Delta b| = \frac{1}{10}\frac{abserr}{a(c_{\max}+|\Delta c|)}$ to perfectly meet the required inequality above.

So let's manage to compute the main factor $\psi_3^{A,B}(x) = \sum_{k=1}^{\infty} E_k(\upsilon, \xi, \omega_3, \lambda, \omega_2, \omega_1)$ with an absolute error less equal $|\Delta c|$, i. e.

$$|\Delta\psi_3^{A,B}(x)| \leq |\Delta c|. \tag{8}$$

To achieve this goal we will walk through a chain of sufficient conditions until we come to an end. Like sometimes usual in error analysis we denote the approximate value (the calculated one) with a tilde (˜) which in general differs from the exact value, the difference is denoted with a Δ, for example $|\Delta x| = |x - \tilde{x}|$. So let's consider

$$\begin{aligned}|\Delta\psi_3^{A,B}(x)| &= \left|\sum_{k=1}^{\infty} E_k(\upsilon, \xi, \omega_3, \lambda, \omega_2, \omega_1) - \sum_{k=1}^{k_{\max}} \widetilde{E}_k(\upsilon, \xi, \omega_3, \lambda, \omega_2, \omega_1)\right| \\ &\leq \sum_{k=1}^{k_{\max}} |\Delta E_k(\upsilon, \xi, \omega_3, \lambda, \omega_2, \omega_1)| + \sum_{k=k_{\max}+1}^{\infty} |E_k(\upsilon, \xi, \omega_3, \lambda, \omega_2, \omega_1)|.\end{aligned}$$

We recall from step 6

$$\sum_{k=k_{\max}+1}^{\infty} |E_k(\upsilon, \xi, \omega_3, \lambda, \omega_2, \omega_1)| \leq \frac{e^{|\xi|+|\lambda|(2+|\omega_3|)+\beta}}{2(1+\omega_2(2+|\omega_3|))}\frac{\beta^{k_{\max}+1}}{k_{\max}!(k_{\max}+1)!} =: \Delta_{k_{\max}},$$

so if we choose $k_{\max}$ to meet

$$\Delta_{k_{\max}} \leq \frac{1}{2}|\Delta c|, \tag{9}$$

we see that, using the rest of our available error tolerance for the $k_{\max}$ parts of the first sum, to satisfy (8) it suffices to meet

$$|\Delta E_k(\upsilon,\xi,\omega_3,\lambda,\omega_2,\omega_1)| \le \frac{|\Delta c| - \Delta_{k_{\max}}}{k_{\max}} =: \Delta_k(k_{\max}) \qquad k = 1,\dots,k_{\max}. \tag{10}$$

Therefore we consider

$$|\Delta E_k(\upsilon,\xi,\omega_3,\lambda,\omega_2,\omega_1)| = |\hat{E}_k \sum_{r=0}^{k-1} (E_r(k;\upsilon,\xi,\omega_3,\lambda,\omega_2) - \widetilde{E}_r(k;\upsilon,\xi,\omega_3,\lambda,\omega_2))|$$
$$\le |\hat{E}_k| \sum_{r=0}^{k-1} |\Delta E_r(k;\upsilon,\xi,\omega_3,\lambda,\omega_2)|,$$

thus to satisfy (10) it suffices to meet

$$|\Delta E_r(k;\upsilon,\xi,\omega_3,\lambda,\omega_2)| \le \frac{\Delta_k(k_{\max})}{k\hat{E}_k} =: \Delta_r(k,k_{\max}) \qquad r = 0,\dots,k-1. \tag{11}$$

We continue with

$$|\Delta E_r(k;\upsilon,\xi,\omega_3,\lambda,\omega_2)| = \left|\hat{E}_r \left(\sum_{L\ge 0} E_L(k,r;\upsilon,\xi,\omega_3,\lambda) - \sum_{L=0}^{L_{\max}} \widetilde{E}_L(k,r;\upsilon,\xi,\omega_3,\lambda) \right)\right|$$
$$\le \hat{E}_r \sum_{L=0}^{L_{\max}} |\Delta E_L(k,r;\upsilon,\xi,\omega_3,\lambda)| + \hat{E}_r \sum_{L=L_{\max}+1}^{\infty} |E_L(k,r;\upsilon,\xi,\omega_3,\lambda)|,$$

from step 4 we know

$$\sum_{L=L_{\max}+1}^{\infty} |E_L(k,r;\upsilon,\xi,\omega_3,\lambda)| \le e^{|\xi|+|\lambda|\gamma} 2^{k-1} \frac{1}{k!} (1+|\upsilon|)^k \gamma^r \frac{(|\lambda|\gamma)^{L_{\max}+1}}{(L_{\max}+1)!} =: \Delta_{L_{\max}}(k,r),$$

so we first choose $L_{\max}$ to satisfy

$$\Delta_{L_{\max}}(k,r) \le \frac{1}{2} \frac{\Delta_r(k,k_{\max})}{\hat{E}_r}, \tag{12}$$

so that it suffices to meet

$$|\Delta E_L(k,r;\upsilon,\xi,\omega_3,\lambda)| \le \frac{\frac{\Delta_r(k,k_{\max})}{\hat{E}_r} - \Delta_{L_{\max}}(k,r)}{L_{\max}+1} =: \Delta_L(k,k_{\max},r,L_{\max}) \qquad L = 0,\dots,L_{\max} \tag{13}$$

to satisfy (11). Little surprisingly we continue with

$$|\Delta E_L(k,r;\upsilon,\xi,\omega_3,\lambda)| = \left|\hat{E}_L \sum_{\rho=0}^{\min(r+L,k)} \Big(E_\rho(k,r+L;\upsilon,\xi,\omega_3) - \widetilde{E}_\rho(k,r+L;\upsilon,\xi,\omega_3) \Big)\right|$$
$$\le |\hat{E}_L| \sum_{\rho=0}^{\min(r+L,k)} |\Delta E_\rho(k,r+L;\upsilon,\xi,\omega_3)|,$$

thus to satisfy (13) we have to meet for $\rho = 0,\ldots,\min(r+L,k)$

$$|\Delta E_\rho(k,r+L;\upsilon,\xi,\omega_3)| \leq \frac{\Delta_L(k,k_{\max},r,L_{\max})}{|\hat{E}_L|(\min(r+L,k)+1)} =: \Delta\rho(k,k_{\max},r,L,L_{\max}). \tag{14}$$

Therefore we continue with (note that we can compute $E_m(\ldots)$ exactly, i. e. $E_m(\ldots) = \widetilde{E}_m(\ldots)$)

$$\begin{aligned}|\Delta E_\rho(k,r+L;\upsilon,\xi,\omega_3)| &= \left|\hat{E}_\rho \sum_{m=m_{\max}+1}^{\infty} E_m(k,r+L,\rho;\upsilon,\xi)\right| \\ &\leq |\hat{E}_\rho| \sum_{m=m_{\max}+1}^{\infty} |E_m(k,r+L,\rho;\upsilon,\xi)|,\end{aligned}$$

thus to satisfy (14) we have to meet

$$\sum_{m=m_{\max}+1}^{\infty} |E_m(k,r+L,\rho;\upsilon,\xi)| \leq \frac{\Delta\rho(k,k_{\max},r,L,L_{\max})}{|\hat{E}_\rho|}. \tag{15}$$

From step 2 we know

$$\sum_{m=m_{\max}+1}^{\infty} |E_m(k,r+L,\rho;\upsilon,\xi)| \leq \left\{\begin{matrix}(1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|}\end{matrix}\right\} \frac{2^{k+r+L-\rho-1}e^{|\xi|}}{(m_{\max}+1+k+r+L-\rho)^\rho} \frac{|\xi|^{m_{\max}+1}}{(m_{\max}+1)!},$$

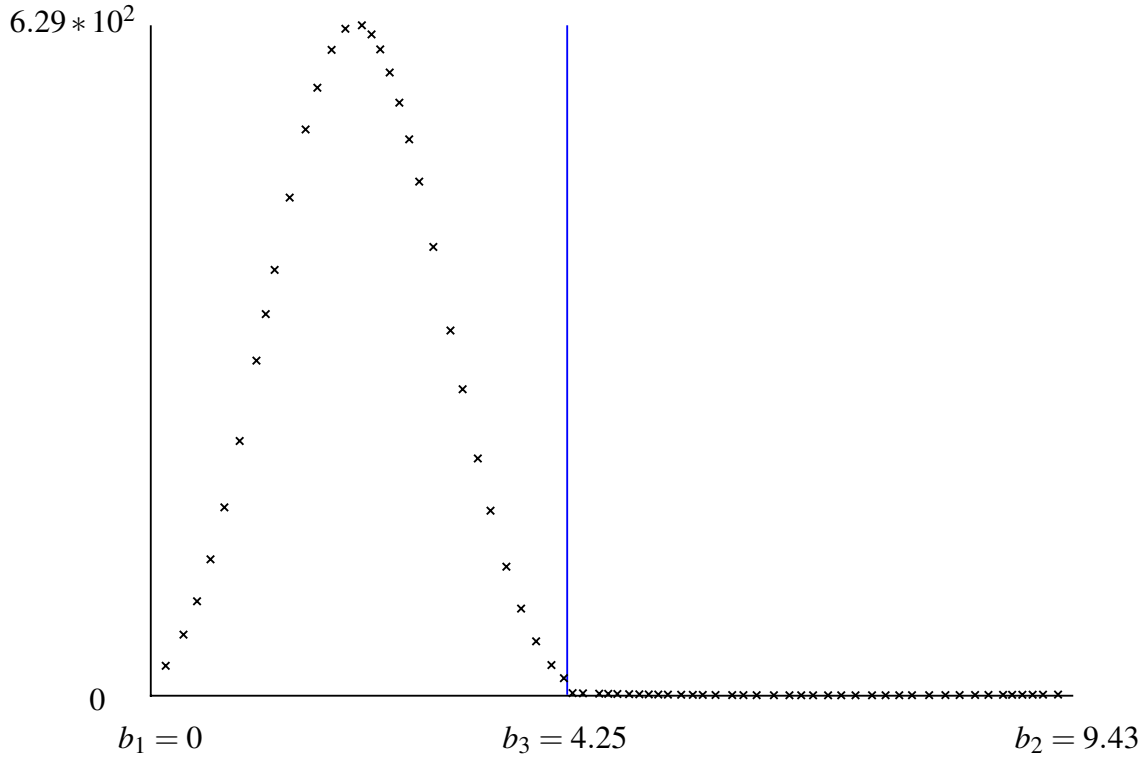
thus to achieve (15) we have to choose $m_{\max}$ to meet

$$\left\{\begin{matrix}(1+|\upsilon|)^k \frac{k^\rho}{k!} \\ e^{|\upsilon|}\end{matrix}\right\} \frac{2^{k+r+L-\rho-1}e^{|\xi|}}{(m_{\max}+1+k+r+L-\rho)^\rho} \frac{|\xi|^{m_{\max}+1}}{(m_{\max}+1)!} \leq \frac{\Delta\rho(k,k_{\max},r,L,L_{\max})}{|\hat{E}_\rho|}. \tag{16}$$

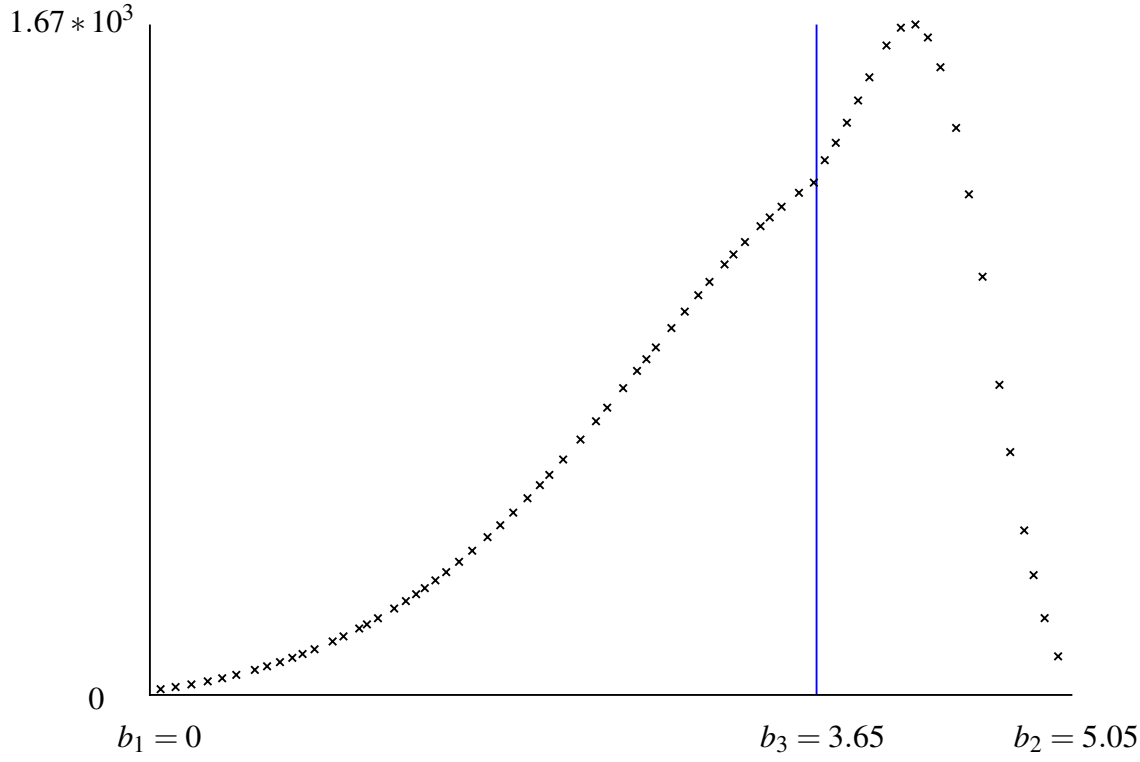
Here we are, since there is no additionally condition left we finally reached the end of the chain. So if we come across an infinite sum during computation and choose the relevant index k_{max}, L_{max} or m_{max} according to (9), (12) or (16) respectively, it's assured that we are going to achieve our original goal, namely to satisfy $|\Delta\psi_3^{A,B}(x)| \leq |\Delta c|$.

3 Results

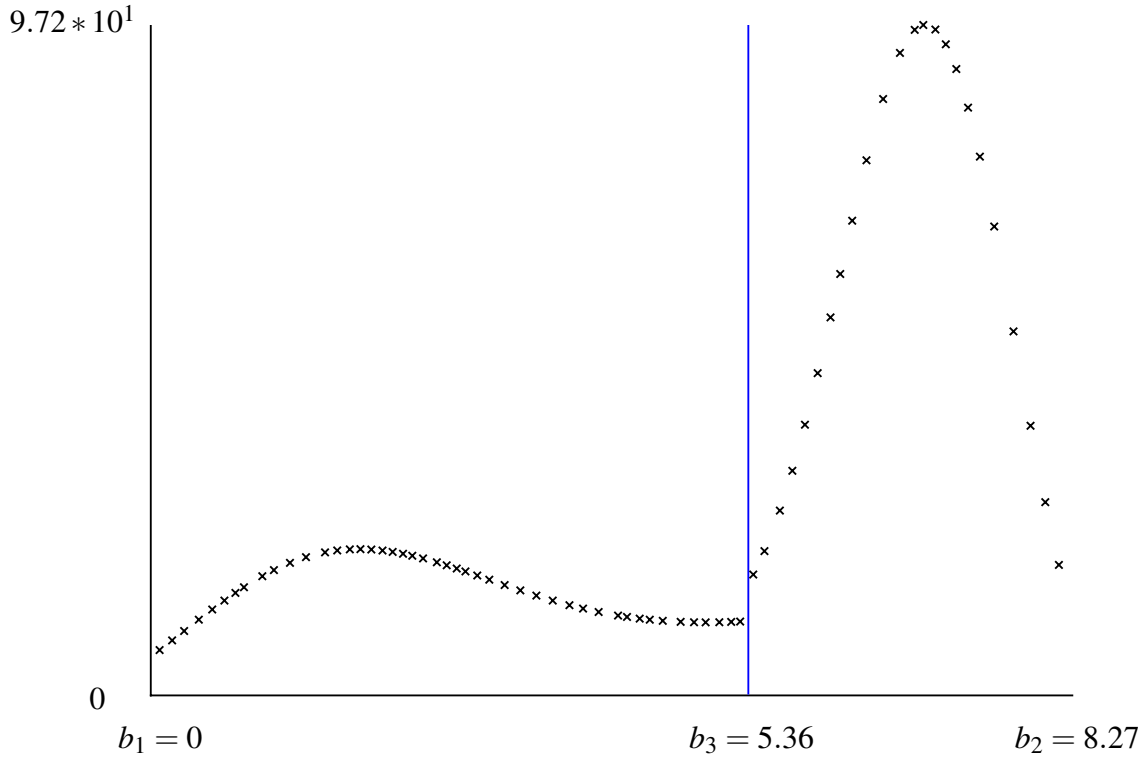
The following diagrams show $\psi^{A,B}(x)$, computed by means of the introduced method above, for some random matrices with diagonal elements equal zero, i. e. the type of matrices the formula originally was developed for. As expected no negative values were found so far.



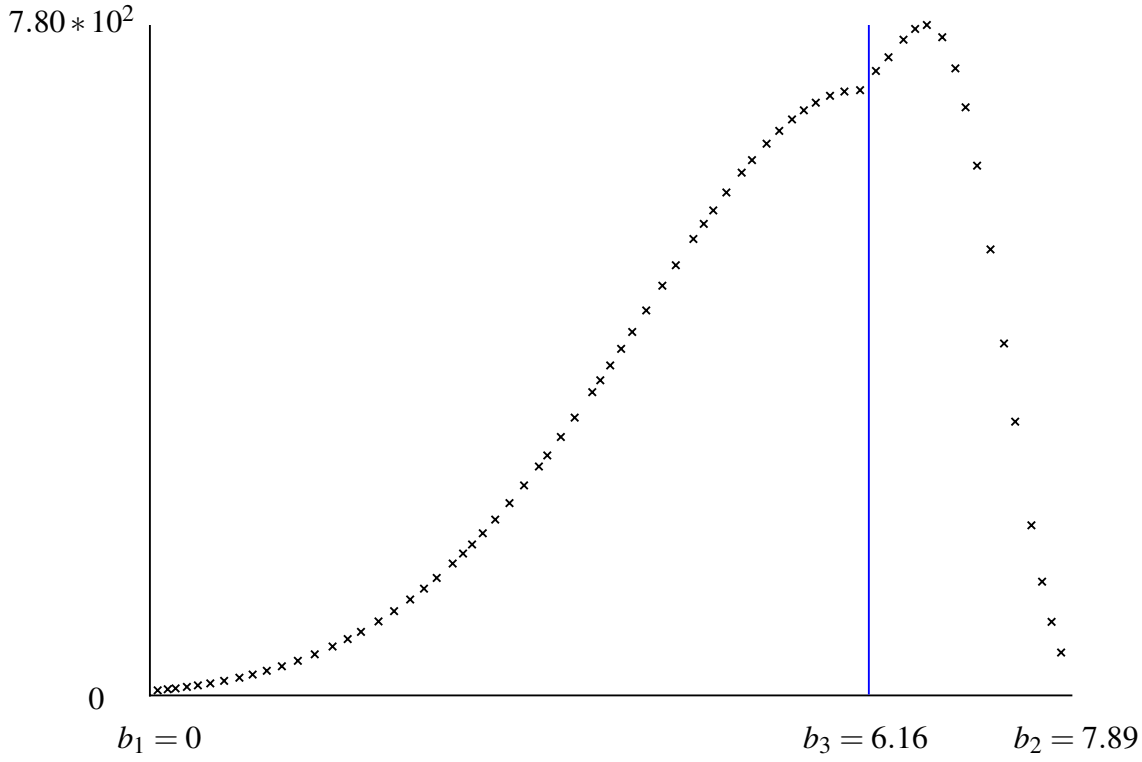
$$A = \begin{pmatrix} 0 & 2.52 & 7.13 \\ 2.52 & 0 & -1.56 \\ 7.13 & -1.56 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 9.43 & 0 \\ 0 & 0 & 4.25 \end{pmatrix}$$



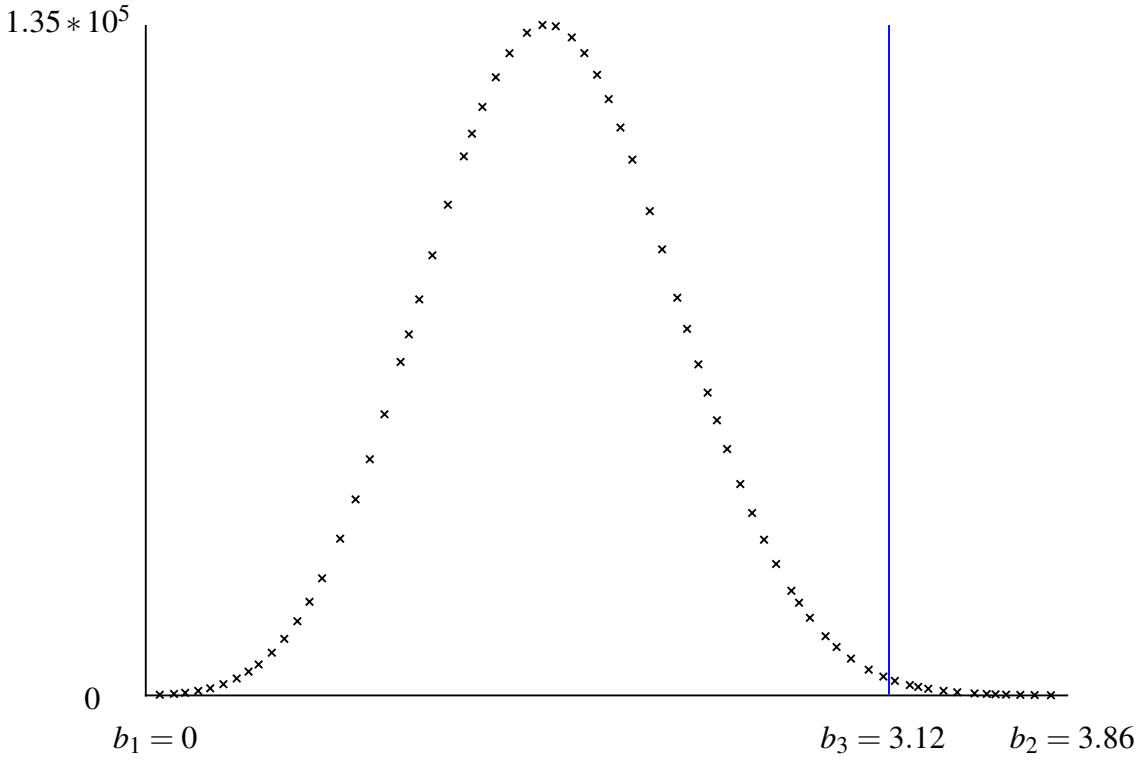
$$A = \begin{pmatrix} 0 & -2.25 & 6.10 \\ -2.25 & 0 & 6.84 \\ 6.10 & 6.84 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 5.05 & 0 \\ 0 & 0 & 3.65 \end{pmatrix}$$



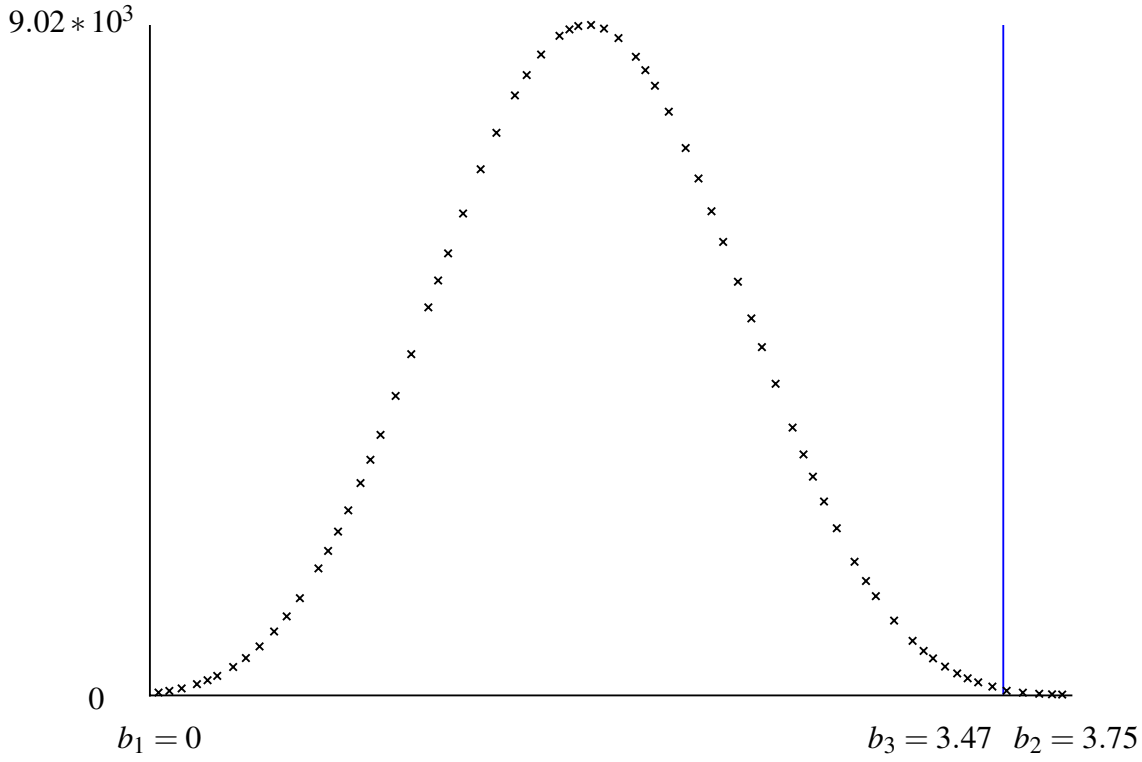
$$A = \begin{pmatrix} 0 & 5.04 & -3.71 \\ 5.04 & 0 & 4.84 \\ -3.71 & 4.84 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 8.27 & 0 \\ 0 & 0 & 5.36 \end{pmatrix}$$



$$A = \begin{pmatrix} 0 & 6.30 & 0.75 \\ 6.30 & 0 & -5.34 \\ 0.75 & -5.34 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 7.89 & 0 \\ 0 & 0 & 6.16 \end{pmatrix}$$



$$A = \begin{pmatrix} 0 & 8.96 & -8.61 \\ 8.96 & 0 & 0.70 \\ -8.61 & 0.70 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3.86 & 0 \\ 0 & 0 & 3.12 \end{pmatrix}$$



$$A = \begin{pmatrix} 0 & 5.94 & -7.48 \\ 5.94 & 0 & 0.06 \\ -7.48 & 0.06 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3.75 & 0 \\ 0 & 0 & 3.47 \end{pmatrix}$$

A Verification

We are now going to verify the formula for $\psi^{A,B}(x)$ as well as its numerical implementation by means of a comparison, mainly by computing the trace function

$$z \mapsto \Phi^{A,B}(z) := \operatorname{tr}\left(e^{A-zB}\right) \tag{17}$$

in two different ways. First the conventional way by means of formula (17) and then due to the Laplace transform

$$z \mapsto \overline{\Phi}^{A,B}(z) := \int_0^\infty e^{-zx} d\nu^{A,B}(x)$$

of the measure $\nu^{A,B}(x)$ represented by its dense function $\psi^{A,B}(x)$ together with the singular parts (see (2)) where $\psi^{A,B}(x)$ was estimated by a rectangular function based on exactly 70 values for which it was calculated (section 3). By increasing this number the calculated values can still be improved in some cases, i. e. $\frac{\Phi^{A,B}(z)}{\overline{\Phi}^{A,B}(z)}$ goes towards 1. Of course the quotient can not get arbitrary close to 1 as the values of $\Psi^{A,B}(x)$ are not exact. Altogether these results give us a good reason to believe that the formula for $\Psi^{A,B}(x)$ as well as the method to calculate it are correct.

Here are the results for the random matrices of section 3:

$$A = \begin{pmatrix} 0 & 2.52 & 7.13 \\ 2.52 & 0 & -1.56 \\ 7.13 & -1.56 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 9.43 & 0 \\ 0 & 0 & 4.25 \end{pmatrix}$$

z	$\Phi^{A,B}(z)$	$\overline{\Phi}^{A,B}(z)$	$\frac{\Phi^{A,B}(z)}{\overline{\Phi}^{A,B}(z)}$
0	$0,13336e4$	$0,13351e4$	$0,99888e0$
1	$0,21938e3$	$0,22015e3$	$0,99651e0$
2	$0,62378e2$	$0,62881e2$	$0,992e0$
3	$0,2636e2$	$0,2666e2$	$0,98875e0$
4	$0,14482e2$	$0,14615e2$	$0,99093e0$
5	$0,94278e1$	$0,94235e1$	$0,10005e1$
6	$0,68598e1$	$0,67439e1$	$0,10172e1$
7	$0,53805e1$	$0,51736e1$	$0,104e1$
8	$0,44468e1$	$0,4166e1$	$0,10674e1$
9	$0,38158e1$	$0,3475e1$	$0,10981e1$

$$A = \begin{pmatrix} 0 & -2.25 & 6.10 \\ -2.25 & 0 & 6.84 \\ 6.10 & 6.84 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 5.05 & 0 \\ 0 & 0 & 3.65 \end{pmatrix}$$

z	$\Phi^{A,B}(z)$	$\overline{\Phi}^{A,B}(z)$	$\frac{\Phi^{A,B}(z)}{\overline{\Phi}^{A,B}(z)}$
0	$0,33602e4$	$0,33616e4$	$0,99957e0$
1	$0,19988e3$	$0,19993e3$	$0,99975e0$
2	$0,38526e2$	$0,38554e2$	$0,99929e0$
3	$0,15731e2$	$0,15743e2$	$0,99925e0$
4	$0,91489e1$	$0,91476e1$	$0,10001e1$
5	$0,63641e1$	$0,63507e1$	$0,10021e1$
6	$0,49045e1$	$0,48798e1$	$0,10051e1$
7	$0,40304e1$	$0,39949e1$	$0,10089e1$
8	$0,34577e1$	$0,34121e1$	$0,10134e1$
9	$0,30576e1$	$0,30023e1$	$0,10184e1$

$$A = \begin{pmatrix} 0 & 5.04 & -3.71 \\ 5.04 & 0 & 4.84 \\ -3.71 & 4.84 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 8.27 & 0 \\ 0 & 0 & 5.36 \end{pmatrix}$$

z	$\Phi^{A,B}(z)$	$\overline{\Phi}^{A,B}(z)$	$\frac{\Phi^{A,B}(z)}{\overline{\Phi}^{A,B}(z)}$
0	$0,25704e3$	$0,25758e3$	$0,9979e0$
1	$0,14914e2$	$0,14924e2$	$0,99931e0$
2	$0,65657e1$	$0,65656e1$	$0,1e1$
3	$0,41933e1$	$0,41824e1$	$0,10026e1$
4	$0,3166e1$	$0,31446e1$	$0,10068e1$
5	$0,26156e1$	$0,25844e1$	$0,10121e1$
6	$0,22794e1$	$0,22392e1$	$0,10179e1$
7	$0,2055e1$	$0,20067e1$	$0,10241e1$
8	$0,18957e1$	$0,18398e1$	$0,10304e1$
9	$0,17772e1$	$0,17143e1$	$0,10367e1$

$$A = \begin{pmatrix} 0 & 6.30 & 0.75 \\ 6.30 & 0 & -5.34 \\ 0.75 & -5.34 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 7.89 & 0 \\ 0 & 0 & 6.16 \end{pmatrix}$$

z	$\Phi^{A,B}(z)$	$\overline{\Phi}^{A,B}(z)$	$\frac{\Phi^{A,B}(z)}{\overline{\Phi}^{A,B}(z)}$
0	$0,26961e4$	$0,26975e4$	$0,99949e0$
1	$0,51068e2$	$0,51064e2$	$0,10001e1$
2	$0,99062e1$	$0,99003e1$	$0,10006e1$
3	$0,49071e1$	$0,48967e1$	$0,10021e1$
4	$0,33711e1$	$0,33544e1$	$0,1005e1$
5	$0,26719e1$	$0,26486e1$	$0,10088e1$
6	$0,22817e1$	$0,22519e1$	$0,10132e1$
7	$0,20354e1$	$0,19996e1$	$0,10179e1$
8	$0,18668e1$	$0,18253e1$	$0,10227e1$
9	$0,17446e1$	$0,16979e1$	$0,10275e1$

$$A = \begin{pmatrix} 0 & 8.96 & -8.61 \\ 8.96 & 0 & 0.70 \\ -8.61 & 0.70 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3.86 & 0 \\ 0 & 0 & 3.12 \end{pmatrix}$$

z	$\Phi^{A,B}(z)$	$\overline{\Phi}^{A,B}(z)$	$\frac{\Phi^{A,B}(z)}{\overline{\Phi}^{A,B}(z)}$
0	$0,17543e6$	$0,17542e6$	$0,1e1$
1	$0,3625e5$	$0,36248e5$	$0,1e1$
2	$0,95257e4$	$0,95265e4$	$0,99992e0$
3	$0,31028e4$	$0,31042e4$	$0,99957e0$
4	$0,12164e4$	$0,12179e4$	$0,99881e0$
5	$0,55711e3$	$0,5585e3$	$0,99751e0$
6	$0,29003e3$	$0,29129e3$	$0,99566e0$
7	$0,16761e3$	$0,16873e3$	$0,99338e0$
8	$0,10543e3$	$0,1064e3$	$0,99091e0$
9	$0,71047e2$	$0,71873e2$	$0,98851e0$

$$A = \begin{pmatrix} 0 & 5.94 & -7.48 \\ 5.94 & 0 & 0.06 \\ -7.48 & 0.06 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3.75 & 0 \\ 0 & 0 & 3.47 \end{pmatrix}$$

z	$\Phi^{A,B}(z)$	$\overline{\Phi}^{A,B}(z)$	$\frac{\Phi^{A,B}(z)}{\overline{\Phi}^{A,B}(z)}$
0	$0,13674e5$	$0,13673e5$	$0,1e1$
1	$0,27121e4$	$0,27121e4$	$0,1e1$
2	$0,73804e3$	$0,73812e3$	$0,99989e0$
3	$0,26403e3$	$0,26413e3$	$0,99961e0$
4	$0,11784e3$	$0,11793e3$	$0,99921e0$
5	$0,62438e2$	$0,62518e2$	$0,99872e0$
6	$0,3769e2$	$0,37759e2$	$0,99819e0$
7	$0,25109e2$	$0,25167e2$	$0,9977e0$
8	$0,18028e2$	$0,18075e2$	$0,99738e0$
9	$0,13709e2$	$0,13746e2$	$0,99732e0$

B Source Listing

B.1 BMV.c

```
#include <stdio.h>
#include <math.h>  /* for rint() */
#include <gtk/gtk.h>
#include <glib/gprintf.h>
#include <glib/gstdio.h>  /* for g_mkdir () */
#include <gmp.h>
#include <cairo.h>
#include <cairo-pdf.h>
#include <time.h>

#include <unistd.h>
#include <stdlib.h>  /* for testing */

#include "gg_gmp_cairo.h"
#include "BMV.h"

int main (int argc, char *argv[])
{
  guint i;

  gchar *config_dir;
  gchar *data_dir;
  gchar *cache_dir;

  FileData *filedata;

  GtkLabelGString *add_ops;
  GtkLabelGString *indices;

  GtkButton *button_generate_matrices;
  InputMatrices *generated_matrices;

  Display *display;

  PangoFontDescription *font_description;

  GtkEntry *entry_data_file;

  /* initializing memory */
  filedata = FileData_new ();

  /* user configuration */
  config_dir = g_strconcat (g_get_user_config_dir (), "/bmv/", NULL);
  if (!g_file_test (config_dir, G_FILE_TEST_EXISTS)) g_mkdir_with_parents (config_dir, 0700);

  data_dir = g_strconcat (g_get_user_data_dir (), "/bmv/", NULL);
  if (!g_file_test (data_dir, G_FILE_TEST_EXISTS)) g_mkdir_with_parents (data_dir, 0700);

  cache_dir = g_strconcat (g_get_user_cache_dir (), "/bmv/", NULL);
  if (!g_file_test (cache_dir, G_FILE_TEST_EXISTS)) g_mkdir_with_parents (cache_dir, 0700);


  /* initialization, g_thread_init must be called exactly once */
  if (!g_thread_supported ()) g_thread_init (NULL);
  gdk_threads_init();

  gtk_init(&argc, &argv);


  /* GUI configuration */
  window = g_object_new(GTK_TYPE_WINDOW,
                        "title", "BMV",
                        "resizable", FALSE,  /* if user can resize window, default=TRUE */
```

```
                        "window-position", GTK_WIN_POS_CENTER,
                        "border-width", 5,
                        NULL);


hbox = g_object_new(GTK_TYPE_HBOX,
                    "spacing", 5,
                    NULL);


vbox_main = g_object_new(GTK_TYPE_VBOX,
                         "homogeneous", FALSE,
                         "spacing", 5,
                         NULL);


vbox_1 = g_object_new(GTK_TYPE_VBOX,
                      "homogeneous", FALSE,
                      "spacing", 5,
                      NULL);


vbox_2 = g_object_new(GTK_TYPE_VBOX,
                      "homogeneous", TRUE,
                      "spacing", 5,
                      NULL);


table_1 = g_object_new (GTK_TYPE_TABLE,
                        "homogeneous", TRUE,
                        "column-spacing", 5,
                        "row-spacing", 5,
                        "n-columns", 3,
                        "n-rows", 3,
                        NULL);


table_2 = g_object_new (GTK_TYPE_TABLE,
                        "homogeneous", TRUE,
                        "column-spacing", 5,
                        "row-spacing", 5,
                        "n-columns", 3,
                        "n-rows", 3,
                        NULL);


entry_data_file = g_object_new (GTK_TYPE_ENTRY,
                                "activates-default", TRUE,
                                "text", data_dir,
                                "width-chars", 20,
                                NULL);


for (i=0; i<=5; i++)
  {
    adjustment_matrices[i] = g_object_new (GTK_TYPE_ADJUSTMENT,
                                           "lower", -20.,
                                           "upper", +20.,
                                           "step-increment", 0.05,
                                           "value", 0.,
                                           NULL);
  }


for (i=6; i<=7; i++)
  {
    adjustment_matrices[i] = g_object_new (GTK_TYPE_ADJUSTMENT,
                                           "lower", -0.,
```

```
                                        "upper", +20.,
                                        "step-increment", 0.05,
                                        "value", 0.,
                                        NULL);
  }

for (i=0; i<=7; i++)
  {
    spinbutton_matrices[i] = g_object_new (GTK_TYPE_SPIN_BUTTON,
                                        "adjustment", adjustment_matrices[i],
                                        "digits", DISPLAYED_DIGITS,
                                        "climb-rate", 200.,
                                        "numeric", TRUE,
                                        NULL);
  }

adjustment_x = g_object_new (GTK_TYPE_ADJUSTMENT,
                             "lower", -0.,
                             "upper", +20.,
                             "step-increment", 0.01,
                             "value", 0.,
                             NULL);

spinbutton_x = g_object_new (GTK_TYPE_SPIN_BUTTON,
                             "adjustment", adjustment_x,
                             "digits", 5,
                             "climb-rate", 200.,
                             "numeric", TRUE,
                             NULL);

button_generate_matrices = g_object_new (GTK_TYPE_BUTTON,
                                         "label", "generate matrices",
                                         NULL);
generated_matrices = InputMatrices_with_gmp_variables_new ();

button_computate = g_object_new (GTK_TYPE_BUTTON,
                                 "label", "start",
                                 NULL);

adjustment_input_precision = g_object_new (GTK_TYPE_ADJUSTMENT,
                                        "step-increment", 1.,
                                        "lower", 0.,
                                        "upper", (gdouble) PRECISION,
                                        NULL);

spinbutton_input_precision = g_object_new (GTK_TYPE_SPIN_BUTTON,
                                        "adjustment", adjustment_input_precision,
                                        "value", 6.,
                                        "digits", 0,
                                        "numeric", TRUE,
                                        NULL);

adjustment_result_accuracy = g_object_new (GTK_TYPE_ADJUSTMENT,
                                        "step-increment", 1.,
                                        "lower", -200.,
                                        "upper", +200.,
                                        NULL);

spinbutton_result_accuracy = g_object_new (GTK_TYPE_SPIN_BUTTON,
```

```
                                        "adjustment", adjustment_result_accuracy,
                                        "value", -0.,
                                        "digits", 0,
                                        "numeric", TRUE,
                                        NULL);



display = Display_new ();


fileselection = g_object_new (GTK_TYPE_FILE_SELECTION,
                                    "filename", data_dir,
                                    "title", "select file",
                                    "show-fileops", FALSE,
                                    NULL);


indices = g_new (GtkLabelGString, 1);
indices->label = g_object_new (GTK_TYPE_LABEL,
                                    NULL);
indices->gstring = g_string_new ("");
gtk_widget_set_size_request (GTK_WIDGET (indices->label), 100, 60);
font_description = pango_font_description_from_string ("Monospace 12");
gtk_widget_modify_font (GTK_WIDGET (indices->label), font_description);
pango_font_description_free (font_description);


add_ops = g_new (GtkLabelGString, 1);
add_ops->label = g_object_new (GTK_TYPE_LABEL,
                                    NULL);
add_ops->gstring = g_string_new ("");


toolbar = g_object_new (GTK_TYPE_TOOLBAR,
                                NULL);


toolbutton_openfile = g_object_new (GTK_TYPE_TOOL_BUTTON,
                                        "stock-id", "gtk-open",
                                        NULL);


toolbutton_verify = g_object_new (GTK_TYPE_TOOL_BUTTON,
                                        "label", "verify",
                                        NULL);


gtk_toolbar_insert (toolbar, GTK_TOOL_ITEM (toolbutton_openfile), 0);
gtk_toolbar_insert (toolbar, GTK_TOOL_ITEM (toolbutton_verify), 1);



gtk_container_add (GTK_CONTAINER (window), GTK_WIDGET (vbox_main));
gtk_box_pack_start_defaults (GTK_BOX (vbox_main), GTK_WIDGET (toolbar));
gtk_box_pack_start_defaults (GTK_BOX (vbox_main), GTK_WIDGET (entry_data_file));
gtk_box_pack_start_defaults (GTK_BOX (vbox_main), GTK_WIDGET (hbox));
gtk_box_pack_start_defaults (GTK_BOX (vbox_main), GTK_WIDGET (display->drawingarea));
gtk_box_pack_start_defaults (GTK_BOX (hbox), GTK_WIDGET (vbox_1));
gtk_box_pack_start (GTK_BOX (hbox), GTK_WIDGET (vbox_2), FALSE, TRUE, 0);
gtk_box_pack_start_defaults (GTK_BOX (vbox_1), GTK_WIDGET (table_1));
gtk_box_pack_start_defaults (GTK_BOX (vbox_1), GTK_WIDGET (table_2));
gtk_box_pack_start_defaults (GTK_BOX (vbox_1), GTK_WIDGET (indices->label));
gtk_box_pack_start_defaults (GTK_BOX (vbox_2), GTK_WIDGET (spinbutton_x));
gtk_box_pack_start_defaults (GTK_BOX (vbox_2), GTK_WIDGET (button_generate_matrices));
gtk_box_pack_start_defaults (GTK_BOX (vbox_2), GTK_WIDGET (spinbutton_input_precision));
gtk_box_pack_start_defaults (GTK_BOX (vbox_2), GTK_WIDGET (button_computate));
```

```
gtk_box_pack_start_defaults (GTK_BOX (vbox_2), GTK_WIDGET (spinbutton_result_accuracy));
gtk_box_pack_start_defaults (GTK_BOX (vbox_2), GTK_WIDGET (add_ops->label));
gtk_table_attach_defaults (table_1, GTK_WIDGET (spinbutton_matrices[0]), 0, 1, 0, 1);
gtk_table_attach_defaults (table_1, GTK_WIDGET (spinbutton_matrices[1]), 1, 2, 1, 2);
gtk_table_attach_defaults (table_1, GTK_WIDGET (spinbutton_matrices[2]), 2, 3, 2, 3);
gtk_table_attach_defaults (table_1, GTK_WIDGET (spinbutton_matrices[3]), 1, 2, 0, 1);
gtk_table_attach_defaults (table_1, GTK_WIDGET (spinbutton_matrices[4]), 2, 3, 0, 1);
gtk_table_attach_defaults (table_1, GTK_WIDGET (spinbutton_matrices[5]), 2, 3, 1, 2);
gtk_table_attach_defaults (table_2, GTK_WIDGET (spinbutton_matrices[6]), 1, 2, 1, 2);
gtk_table_attach_defaults (table_2, GTK_WIDGET (spinbutton_matrices[7]), 2, 3, 2, 3);




gtk_widget_add_events (GTK_WIDGET (display->drawingarea),
                       GDK_POINTER_MOTION_HINT_MASK |
                       GDK_BUTTON_MOTION_MASK |
                       GDK_BUTTON_PRESS_MASK |
                       GDK_BUTTON_RELEASE_MASK);

/* assigning pointer structure */
pointerbundle all = {& computation, thread, & stop, add_ops, indices,
                     filedata, display, generated_matrices, entry_data_file, NULL};


g_signal_connect_after (G_OBJECT (display->drawingarea),
                        "realize",
                        G_CALLBACK (create_cairo_context_for_window_of_drawingarea),
                        & display->cd);


g_signal_connect (G_OBJECT (display->drawingarea),
                  "motion_notify_event",
                  G_CALLBACK (button_motion),
                  display);

g_signal_connect (G_OBJECT (display->drawingarea),
                  "button_press_event",
                  G_CALLBACK (button_press),
                  display->focus);

g_signal_connect (G_OBJECT (window),
                  "delete_event",
                  G_CALLBACK (delete_event_window),
                  NULL);

g_signal_connect (G_OBJECT (window),
                  "destroy",
                  G_CALLBACK (destroy),
                  NULL);

g_signal_connect_swapped (G_OBJECT (fileselection->cancel_button),
                          "clicked",
                          G_CALLBACK (gtk_widget_hide),
                          GTK_WIDGET (fileselection));

g_signal_connect_swapped (G_OBJECT (toolbutton_openfile),
                          "clicked",
                          G_CALLBACK (gtk_widget_show),
                          GTK_WIDGET (fileselection));

g_signal_connect_swapped (G_OBJECT (fileselection),
                          "delete_event",
                          G_CALLBACK (gtk_widget_hide),
                          GTK_WIDGET (fileselection));

g_signal_connect (G_OBJECT (fileselection->ok_button),
```

```
                        "clicked",
                        G_CALLBACK (fileselection_ok_button),
                        & all);

    g_signal_connect (G_OBJECT (entry_data_file),
                        "activate",
                        G_CALLBACK (activate_entry_data_file),
                        & all);

    g_signal_connect (G_OBJECT (toolbutton_verify),
                        "clicked",
                        G_CALLBACK (verify),
                        filedata);

    g_signal_connect (G_OBJECT (button_generate_matrices),
                        "clicked",
                        G_CALLBACK (generate_matrices_sh),
                        generated_matrices);

    g_signal_connect (G_OBJECT (button_computate),
                        "clicked",
                        G_CALLBACK (computation_on_off),
                        & all);


    gtk_widget_show_all (GTK_WIDGET (window));


    gdk_threads_enter();
    gtk_main();
    gdk_threads_leave();


    return EXIT_SUCCESS;
}



/* A signal handler ("realize") connected with the widget drawingarea of the display structure, just */
/* because connecting a cairo context to a GdkDrawable (the window of drawingarea) can not be done */
/* before the widget is realized. */
void create_cairo_context_for_window_of_drawingarea (GtkDrawingArea *drawingarea, cairo_t **cd)
{
    *cd = gdk_cairo_create (GDK_DRAWABLE (GTK_WIDGET (drawingarea)->window));
}



gboolean button_press (GtkDrawingArea   *drawingarea, GdkEventButton *event, GdkRectangle *focus)
{
    gdk_window_clear (GTK_WIDGET (drawingarea)->window);
    focus->x = (gint) event->x;
    focus->y = (gint) event->y;
    focus->width = 0;
    focus->height = 0;

    return FALSE;
}



gboolean button_motion (GtkDrawingArea   *drawingarea, GdkEventMotion *event, Display *display)
{
    gint x;
```

```
  gdk_window_clear (GTK_WIDGET (drawingarea)->window);

  cairo_set_source_rgba (display->cd, 1., 0.1, 0., 1.);

  cairo_set_line_width (display->cd, 1.);

  if (event->x < 0)     display->focus->width = (gint) (0 - display->focus->x);
  else if (event->x < WIDTH)     display->focus->width = (gint) (event->x - display->focus->x);
  else display->focus->width = (gint) (WIDTH-1 - display->focus->x);

  if (event->y < 0)     display->focus->height = (gint) (0 - display->focus->y);
  else if (event->y < HEIGHT)     display->focus->height = (gint) (event->y - display->focus->y);
  else display->focus->height = (gint) (HEIGHT-1 - display->focus->y);

  cairo_rectangle (display->cd, (gdouble) (display->focus->x) + 0.5, (gdouble) (display->focus->y) + 0.5,
                   (gdouble) (display->focus->width), (gdouble) (display->focus->height));

  cairo_stroke (display->cd);

  /* We set the GDK_POINTER_MOTION_HINT_MASK to suppress the following Button Motion Event until the */
  /* current one is not handled yet, to release it again we need the gtk_widget_get_pointer instruction */
  gtk_widget_get_pointer (GTK_WIDGET (drawingarea), & x, & x);

  return FALSE; /* to allow other handlers also connected to this signal to be invoked */
}



int delete_event_window (GtkWidget *widget, GdkEvent *event, gpointer unused)
{
  g_print ("Delete Event occured \n");

  return FALSE;  /* emit a destroy signal */
}



void destroy (GtkWidget *widget, gpointer unused)
{
  gtk_main_quit();
}



/* This is a signal handler to stop or create the computate thread: computation is a gboolean variable */
/* initialized with FALSE, it's turned to TRUE only after creating the computate thread which on the */
/* other hand can only be created if computate is FALSE, i.e. there is at most one computate thread */
/* running.  It's turned to FALSE only if the computate thread was asked to stop (by setting the global */
/* gboolean variable stop to TRUE) and then was joined or if the computate thread reaches the end (the */
/* last instructions (except return) whitin the computate thread is computate = FALSE).  As the pointer */
/* thread is only turned to NULL directly after joining the thread, in the second case the pointer */
/* thread is not equal NULL and therefore has to be joined (just to free memory), everything is done */
/* within an idle function because any signal handler is embeeded whitin a gdk_thread environment (if */
/* the gtk_main loop is embeeded in such an environment), so it's not possible to join a thread there */
/* if the thread has another gdk_thread environment to run through, the whole application will be */
/* blocked then!!Idle functions are not whitin a gdk_thread environment and therefore appropriated to */
/* join threads, but don't forget to consider that both, signal handler and idle function running */
/* within the main loop (but just the first also within the gtk+ lock produced by embeeding the */
/* gtk_main loop whitin an gdk_threads environment) and therefore preventing the whole GUI to response */
/* while they are running.  Do not allow another button_pressed signal the associated signal is blocked */
/* from the signal handler and will be unblocked after starting the idle function, or at the end, no */
/* matter, any signal handler anyway has to wait for the idle function to finish, as both of them are */
/* running within the gtk_main loop. */
void computation_on_off (GtkButton *button_computate, pointerbundle *p)
{
```

```
  gtk_widget_set_sensitive (GTK_WIDGET (button_computate), FALSE);
  g_idle_add ((GSourceFunc) create_stop_thread, p);
}



/* A idle function to tell the computate thread to stop and join him, to do so the */
/* boolean variable stop is set to TRUE, the computate threads looks within his loop frequently for */
/* this variable and exits immediately if it's set to TRUE, so actually it won't take a long time from */
/* setting stop to TRUE until it exits, depends upon the current computation, currently the querying is */
/* inside the L loop */
gboolean create_stop_thread (pointerbundle *p)  /* global: computation, thread, stop, button_computate */
{
  if (computation == FALSE)  /* if there is a thread running at all it must be very close to finish */
    {
      if (!(thread == NULL))  /* previous thread (already finished or close to finish) was not yet joined */
        {
          g_thread_join (thread);  /* free old GThread structure */
          thread = NULL;    /* actually not necessary, new one will be created soon */
        }
      stop = FALSE;
      if ((thread = g_thread_create ((GThreadFunc) computate, p, TRUE, NULL)))
        {
          computation = TRUE;
          gdk_threads_enter ();
          gtk_button_set_label (button_computate, "stop");
          gdk_threads_leave ();
          g_print ("new thread created\n");
        }
      else g_printerr ("thread could not be created\n");
    }
  else
    {
      stop = TRUE;
      g_thread_join (thread);
      thread = NULL;
      computation = FALSE;
      g_print ("thread stoped\n");
      gdk_threads_enter ();
      gtk_button_set_label (button_computate, "start");
      gdk_threads_leave ();
    }

  gdk_threads_enter ();
  gtk_widget_set_sensitive (GTK_WIDGET (button_computate), TRUE);
  gdk_threads_leave ();


  return FALSE;
}



/* A timeout function to print out the number of add operations to the assigned label widget */
gboolean print_num_of_add_ops_to_label (pointerbundle *p)
{
  g_string_printf (p->add_ops->gstring, "%lu", num_of_add_ops);

  gdk_threads_enter ();
  gtk_label_set_text (p->add_ops->label, p->add_ops->gstring->str);
  gdk_threads_leave ();

  if (* p->computation == TRUE) return TRUE;
  else return FALSE;
}
```

```
/* global variables: entry_data_file, spinbutton_matrices */
void load_file_into_gui (const gchar *fname, FileData *fdata, Display *display, GtkEntry *entry)
{
  guint i;

  if (check_and_load_file_into_memory (fdata, fname))
    {
      gtk_entry_set_text (entry, fname);

      /* set range_rectangle */
      mpq_set_ui (display->range->x, 0UL, 1UL);
      mpq_set (display->range->width, fdata->matrices->b2);
      mpq_set_ui (display->range->y, 0UL, 1UL);

      if (fdata->list != NULL)
        {
          mpq_set (display->range->height, ((ListItem *) fdata->list->data)->fv);
          g_list_foreach (fdata->list, (GFunc) maximum_functionvalue, display->range->height);
          if (mpq_cmp_ui (display->range->height, 0UL, 1UL) == 0)
            mpq_set_ui (display->range->height, 1UL, 1UL);
        }
      else
        mpq_set_ui (display->range->height, 1UL, 1UL);

      /* display matrices */
      for (i=0; i<=7; i++)
        gtk_spin_button_set_value (spinbutton_matrices[i], fdata->matrices->d[i]);

      draw_on_display (fdata, display);
    }
}



/* A signal handler (activate, if return or enter was invoked) connected to entry_data_file of type */
/* GtkEntry. Note that the function gtk_entry_get_text returns a pointer to a string associated to the */
/* entry widget and thus must not be freed */
void activate_entry_data_file (GtkEntry *entry, pointerbundle *p)
{
  load_file_into_gui (gtk_entry_get_text (entry), p->filedata, p->display, entry);
}



/* A signal handler (clicked) connected to the OK button of the fileselection of type GtkFileSelection. */
/* Note that the function gtk_fileselection_get_filename returns a pointer to a string associated to */
/* the fileselection widget and thus must not be freed */
void fileselection_ok_button (GtkButton *button, pointerbundle *p)  /* global: fileselection */
{
  load_file_into_gui (gtk_file_selection_get_filename (fileselection), p->filedata,
                      p->display, p->entry_data_file);


  gtk_widget_hide (GTK_WIDGET (fileselection));
}



double user_x (mpq_t x, mpqRectangle *range)
{
  mpq_sub (range->auqx, x, range->x);
```

```
  mpq_div (range->auqx, range->auqx, range->width);
  gg_mpq_mul_ui (range->auqx, range->auqx, DIAGRAMWIDTH);

  return  mpq_get_d (range->auqx);
}


double user_y (mpq_t y, mpqRectangle *range)
{
  mpq_sub (range->auqy, y, range->y);
  mpq_div (range->auqy, range->auqy, range->height);
  gg_mpq_mul_ui (range->auqy, range->auqy, DIAGRAMHEIGHT);

  return  mpq_get_d (range->auqy);
}



gboolean is_within_range_rectangle (ListItem *item, mpqRectangle *range)
{
  mpq_add (range->auqx, range->x, range->width);
  if (!(mpq_cmp (range->x, item->x) <= 0 && mpq_cmp (item->x, range->auqx) <= 0)) return FALSE;
  mpq_add (range->auqy, range->y, range->height);
  if (!(mpq_cmp (range->y, item->fv) <= 0 && mpq_cmp (item->fv, range->auqy) <= 0)) return FALSE;
  return TRUE;
}



/* Bring the argument to normal form (x right, y up, width and height nonnegativ) */
void normalize_focus_rectangle (GdkRectangle *focus)
{
  focus->x = (focus->x + MIN (0, focus->width));
  focus->y = (HEIGHT-1) - (focus->y + MAX (0, focus->height));
  focus->width = ABS (focus->width);
  focus->height = ABS (focus->height);
}



void adopt_range_rectangle (mpqRectangle *range, GdkRectangle *focus)
{
  mpq_t auq1;

  mpq_init (auq1);


  if (MIN (focus->width, focus->height) != 0)
    {
      normalize_focus_rectangle (focus);

      /* new->x = old->x + old->width * focus->x / WIDTH    */
      gg_mpq_mul_ui (auq1, range->width, (unsigned long) focus->x);
      gg_mpq_div_ui (auq1, auq1, (unsigned long) WIDTH);
      mpq_add (range->x, range->x, auq1);    /* auq1 nil */

      /* the same with y */
      gg_mpq_mul_ui (auq1, range->height, (unsigned long) focus->y);
      gg_mpq_div_ui (auq1, auq1, (unsigned long) HEIGHT);
      mpq_add (range->y, range->y, auq1);  /* auq1 nil */

      /* new->width = old->width * focus->width / WIDTH    */
      gg_mpq_mul_ui (auq1, range->width, (unsigned long) focus->width);
      gg_mpq_div_ui (range->width, auq1, (unsigned long) WIDTH);  /* auq1 nil */

      /* the same with new->height    */
```

```
      gg_mpq_mul_ui (auq1, range->height, (unsigned long) focus->height);
      gg_mpq_div_ui (range->height, auq1, (unsigned long) HEIGHT);

      /* after adopting the range rectangle set focus rectangle back to zero */
      focus->x = 0;
      focus->y = 0;
      focus->width = 0;
      focus->height = 0;
    }

  mpq_clear (auq1);
}



/* This function performs all drawing operations (without labeling) on the desired surface given as an */
/* argument */
void draw_on_surface (cairo_surface_t *cs, FileData *fdata, mpqRectangle *range)
{
  cairo_t *cc;
  cairo_matrix_t *cmat;

  cmat = g_new (cairo_matrix_t, 1);
  cairo_matrix_init_identity (cmat);
  cairo_matrix_translate (cmat,  (double) LEFTMARGIN, (double) HEIGHT-1 - BOTTOMMARGIN);
  cairo_matrix_scale (cmat, 1., -1.);

  cc = cairo_create (cs);
  cairo_set_matrix (cc, cmat);

  /* paint surface white */
  cairo_set_source_rgb (cc, 1., 1., 1.);
  cairo_paint (cc);

  /* axes of coordinates */
  cairo_set_source_rgba (cc, 0., 0., 0., 1.);
  cairo_set_line_width (cc, LINEWIDTH);
  cairo_move_to (cc, - (double) LINEWIDTH/2, 0.);
  cairo_rel_line_to (cc, (double) (DIAGRAMWIDTH + LINEWIDTH/2), 0.);
  cairo_move_to (cc, 0., - (double) LINEWIDTH/2);
  cairo_rel_line_to (cc, 0., (double) (DIAGRAMHEIGHT + LINEWIDTH/2));
  cairo_stroke (cc);

  /* line at b3 */
  cairo_set_source_rgba (cc, 0., 0., 1., 1.);
  cairo_move_to (cc, rint (user_x (fdata->matrices->b3, range)), + (double) LINEWIDTH/2);
  cairo_rel_line_to (cc, 0., (double) (DIAGRAMHEIGHT - LINEWIDTH/2));
  cairo_stroke (cc);

  /* values */
  cairo_set_source_rgba (cc, 0., 0., 0., 1.);
  cairo_set_line_width (cc, LINE_WIDTH_FOR_CROSS);
  {
    gpointer gp[2];

    gp[0] = cc;
    gp[1] = range;
    g_list_foreach (fdata->list, (GFunc) make_cross_foreach, gp);
  }
  cairo_stroke (cc);

  /* Emits and clears the current page for backends that support multiple pages. Use cairo_copy_page() if */
  /* you don't want to clear the page.  I guess for surfaces (backends) which do not support multiple */
  /* pages this instruction has no affect. */
  cairo_show_page (cc);

  cairo_destroy (cc);
```

```
}



void draw_on_display (FileData *fdata, Display *display)
{
  draw_on_surface (display->mi, fdata, display->range);

  /* labels */
  cairo_set_font_size (display->cmi, FONTSIZE);
  cairo_set_source_rgba (display->cmi, 0.05, 0.55, 0.15, 1.);
  gg_mpq_draw_as_mpf (display->cmi, display->range->x,
                      45., (double) (HEIGHT - BOTTOMMARGIN/2 + FONTSIZE/2));
  gg_mpq_draw_as_mpf (display->cmi, display->range->y,
                      2., (double) (HEIGHT - 3 - BOTTOMMARGIN + FONTSIZE/2));
  gg_mpq_draw_as_mpf (display->cmi, display->range->width,
                      WIDTH/2., (double) (HEIGHT - BOTTOMMARGIN/2 + FONTSIZE/2));
  gg_mpq_draw_as_mpf (display->cmi, display->range->height,
                      2., (double) (TOPMARGIN/2 + (HEIGHT - BOTTOMMARGIN)/2 + FONTSIZE/2));
  cairo_stroke (display->cmi);

  /* labels for b1, b2, b3 */
  {
    gdouble x, y;
    cairo_surface_t *cs;

    cairo_identity_matrix (display->cmi);

    if (g_file_test ("b1_25.png", G_FILE_TEST_EXISTS))
      {
        x = 3.5;
        y = 27.5;
        cairo_matrix_transform_point (display->matrix, & x, & y);
        cs = cairo_image_surface_create_from_png ("b1_25.png");
        cairo_set_source_surface (display->cmi, cs, x, y);
        cairo_paint (display->cmi);
        cairo_surface_destroy (cs);
      }
    if (g_file_test ("b3_25.png", G_FILE_TEST_EXISTS))
      {
        x = user_x (fdata->matrices->b3, display->range);
        x = x + 2.5;
        y = 27.5;
        cairo_matrix_transform_point (display->matrix, & x, & y);
        cs = cairo_image_surface_create_from_png ("b3_25.png");
        cairo_set_source_surface (display->cmi, cs, x, y);
        cairo_paint (display->cmi);
        cairo_surface_destroy (cs);
      }
    if (g_file_test ("b2_25.png", G_FILE_TEST_EXISTS))
      {
        x = DIAGRAMWIDTH + 3.5;
        y = 27.5;
        cairo_matrix_transform_point (display->matrix, & x, & y);
        cs = cairo_image_surface_create_from_png ("b2_25.png");
        cairo_set_source_surface (display->cmi, cs, x, y);
        cairo_paint (display->cmi);
        cairo_surface_destroy (cs);
      }
  }

  /* copy memory image to pixmap */
  cairo_set_source_surface (display->cp, display->mi, 0., 0.);
  cairo_paint (display->cp);

  /* copy pixmap to the background of drawingarea */
  gdk_window_set_back_pixmap (GTK_WIDGET (display->drawingarea)->window, display->pixmap, FALSE);
```

```
  /* show the background */
  gdk_window_clear (GTK_WIDGET (display->drawingarea)->window);
}



void make_cross_for_listitem (cairo_t *cc, ListItem *item, mpqRectangle *range)
{
  if (is_within_range_rectangle (item, range))
    gg_cairo_cross (cc, user_x (item->x, range), user_y (item->fv, range));
}


/* A list foreach handler */
void make_cross_foreach (ListItem *item, gpointer *gp)
{
  make_cross_for_listitem ((cairo_t *) (gp[0]), item, (mpqRectangle *) (gp[1]));
}



/* Expects a pointer to the beginning of a mpq array with at least 8 elements, prints this elements, */
/* converts them to mpf to print them again as mpf. */
void print_matrices_as_mpf (InputMatrices *mat)
{
  guint i;

  for (i=0; i<=7; i++)
    gg_mpq_print_as_mpf ( * mat->p[i]);
}



void message_file_dialog (MessageDialogeType type)
{
  GtkMessageDialog *dialog;

  dialog = g_object_new (GTK_TYPE_MESSAGE_DIALOG,
                         "buttons", GTK_BUTTONS_OK,
                         "message-type", GTK_MESSAGE_WARNING,
                         NULL);

  switch (type)
    {
    case DIALOG_FILE_NOT_FOUND:
      gtk_message_dialog_set_markup (dialog, "file not found\nfile may not exist"); break;
    case DIALOG_FAILED_TO_OPEN_FILE:
      gtk_message_dialog_set_markup (dialog, "failed to open file\nfile may not exist"); break;
    case DIALOG_FILE_IS_A_DIRECTORY:
      gtk_message_dialog_set_markup (dialog, "file is a directory"); break;
    case DIALOG_FILE_IS_NOT_A_REGULAR_BMV_DATA_FILE:
      gtk_message_dialog_set_markup (dialog, "file is not a regular bmv data file"); break;
    case DIALOG_FAILED_TO_GENERATE_MATRICES:
      gtk_message_dialog_set_markup (dialog, "failed to generate matrices after "QTRIES" tries"); break;
    case DIALOG_FAILED_TO_GENERATE_X_VALUE:
      gtk_message_dialog_set_markup (dialog, "failed to generate x value after "QTRIES" tries"); break;
    case DIALOG_FILE_ALREADY_EXISTS:
      gtk_message_dialog_set_markup (dialog, "file already exists"); break;
    }

  gtk_window_set_title (GTK_WINDOW (dialog), "Warning");
  gtk_dialog_run (GTK_DIALOG (dialog));
  gtk_widget_destroy (GTK_WIDGET (dialog));
}
```

```
void write_listitem_to_file (ListItem *item, FILE *stream)
{
  g_fprintf (stream, "%ld\n", item->lberror);
  mpq_out_str (stream, 16, item->x);
  g_fprintf (stream, "\n");
  mpq_out_str (stream, 16, item->fv);
  g_fprintf (stream, "\n\n");
}



/* foreach function */
void maximum_functionvalue (ListItem *item, mpq_t op)
{
  gg_mpq_max (op, op, item->fv);
}



/* foreach function */
void minimum_functionvalue (ListItem *item, mpq_t op)
{
  gg_mpq_min (op, op, item->fv);
}



/* 1.Argument: pointer to structure to which to write filedata */
/* 2.Argument: file to check and read from */
gboolean check_and_load_file_into_memory (FileData *fdata, const gchar *fname)
{
  guint i;
  long l;
  FILE *stream;
  ListItem *item;
  gchar *first_line = "bmvdatafile=TRUE\n";


  if (!g_file_test (fname, G_FILE_TEST_EXISTS))
    {
      message_file_dialog (DIALOG_FILE_NOT_FOUND);
      return FALSE;
    }
  if (g_file_test (fname, G_FILE_TEST_IS_DIR))
    {
      message_file_dialog (DIALOG_FILE_IS_A_DIRECTORY);
      return FALSE;
    }
  if ((stream = g_fopen (fname, "r")) == NULL)
    {
      message_file_dialog (DIALOG_FAILED_TO_OPEN_FILE);
      return FALSE;
    }
  for (i=0; i<=16; i++)
    if (!(first_line[i]==fgetc (stream)))
      {
        message_file_dialog (DIALOG_FILE_IS_NOT_A_REGULAR_BMV_DATA_FILE);
        return FALSE;
      }
```

```
  /* clear old list */
  g_list_foreach (fdata->list, (GFunc) ListItem_with_gmp_variables_free, NULL);
  g_list_free (fdata->list);
  fdata->list = NULL;  /* is necessary because g_list_free () does not care about this pointer */

  for (i=0; i<=7; i++) mpq_inp_str (* fdata->matrices->p[i], stream, 16);
  adopt_matrices_from_q (fdata->matrices);

  while (EOF != fscanf (stream, "%ld", & l))
    {
      item = ListItem_with_gmp_variables_new ();

      item->lberror = l;
      mpq_inp_str (item->x, stream, 16);
      mpq_inp_str (item->fv, stream, 16);

      fdata->list = g_list_append (fdata->list, item);
    }

  fclose (stream);

  g_string_assign (fdata->name, fname);

  return TRUE;
}



/* struct FileData consist of a name and the associated data which is written to disk, a file with the */
/* same name will be overwritten */
gboolean write_filedata_to_file (FileData *fdata)
{
  guint i;
  FILE *stream;


  if ((stream = g_fopen (fdata->name->str, "w")) == NULL)
    {
      message_file_dialog (DIALOG_FAILED_TO_OPEN_FILE);
      return FALSE;
    }

  g_fprintf (stream, "bmvdatafile=TRUE\n\n");

  for (i=0; i<=7; i++)
    {
      mpq_out_str (stream, 16, * fdata->matrices->p[i]);
      g_fprintf (stream, "\n");
    }

  g_fprintf (stream, "\n\n");

  g_list_foreach (fdata->list, (GFunc)write_listitem_to_file, stream);

  fclose (stream);

  return TRUE;
}



/* global: spinbutton_input_precision, spinbutton_matrices */
gboolean compare_matrices (InputMatrices *mat)
{
  gint i;
```

```
  for (i=0; i<=7; i++)
    if (mat->d[i] != gtk_spin_button_get_value (spinbutton_matrices[i])) return FALSE;

  return TRUE;
}



void adopt_matrices_from_q (InputMatrices *mat)
{
  gint i;
  for (i=0; i<=7; i++)
    {
      mpf_set_q (mat->f[i], * mat->p[i]);
      mat->d[i] = mpq_get_d (* mat->p[i]);
    }
}



/* A signal handler (clicked) for the button button_generate_matrices */
/* global: spinbutton_input_precision, spinbutton_matrices */
void generate_matrices_sh (GtkButton *unused1, InputMatrices *mat)
{
  guint i;


  generate_matrices (mat, (unsigned long) DISPLAYED_DIGITS*5, 10.);

  for (i=0; i<=7; i++)
    gtk_spin_button_set_value (spinbutton_matrices[i], (gdouble) mat->d[i]);

  print_matrices_as_mpf (mat);
}



void generate_matrices (InputMatrices *mat, unsigned long precision, double range)
{
  guint i, j=0;
  InputMatrices *auxmat = InputMatrices_with_gmp_variables_new ();

  do
    {
      j++;
      generate_matrices_as_double (auxmat->d, -range, range);
      for (i=0; i<=7; i++) mpf_set_d (auxmat->f[i], auxmat->d[i]);

      for (i=0; i<=5; i++) from_mpf_to_mpq (* auxmat->p[i], auxmat->f[i], precision);
      for (i=6; i<=7; i++) from_mpf_to_mpq_as_perfect_square (* auxmat->p[i], auxmat->f[i], precision);
    }
  while (j < TRIES && !proof_generated_matrices (auxmat));

  if (j == TRIES && !proof_generated_matrices (auxmat))
    message_file_dialog (DIALOG_FAILED_TO_GENERATE_MATRICES);
  else
    {
      g_print ("tries: %d\n", j);
      for (i=0; i<=7; i++)
          mpq_set (* mat->p[i], * auxmat->p[i]);
      adopt_matrices_from_q (mat);
    }

  InputMatrices_with_gmp_variables_free (auxmat);
```

```
}


/* This function expects as argument a pointer to the beginning of a double array which contains at */
/* least 8 elements, otherwise it will leed to a memory error, and writes to the first 6 elements */
/* random numbers uniformly distributed between the arguments lower_bound and upper_bound, to the last */
/* 2 elements uniformly distributed random numbers between 0 and upper_bound. */
void generate_matrices_as_double (double *array, double lower_bound, double upper_bound)
{
  GRand* random_generator;
  gint i;

  random_generator = g_rand_new ();

  for (i=0; i<=5; i++)
    {
      array[i] = (double) g_rand_double_range (random_generator, lower_bound, upper_bound);
    }
  for (i=6; i<=7; i++)
    {
      array[i] = (double) g_rand_double_range (random_generator, 0., upper_bound);
    }

  g_rand_free (random_generator);
}



/* Expects the beginning of a mpq array with at least 8 elements as argument and returns true if they fit */
/* some rules. */
gboolean proof_generated_matrices (InputMatrices *mat)
{
  mpq_t auq1, auq2;

  mpq_init (auq1); mpq_init (auq2);


  if (!(
        mpq_cmp_ui (mat->b3, 0UL, 1UL) > 0
        &&
        mpq_cmp (mat->b2, mat->b3) > 0
        ))
    {
      mpq_clear (auq1); mpq_clear (auq2);
      return FALSE;
    }

  mpq_mul (auq1, mat->a12, mat->a13);
  mpq_mul (auq1, auq1, mat->a23);
  if (mpq_cmp_ui (auq1, 0UL, 1UL) >= 0)
    {
      mpq_clear (auq1); mpq_clear (auq2);
      return FALSE;
    }

  mpq_mul (auq1, mat->a12, mat->a12);
  mpq_mul (auq1, auq1, mat->b3);
  mpq_mul (auq2, mat->a13, mat->a13);
  mpq_mul (auq2, auq2, mat->b2);
  if (!(mpq_cmp (auq1, auq2) >= 0))
    {
      mpq_clear (auq1); mpq_clear (auq2);
      return TRUE;
    }
```

```
  mpq_sub (auq1, mat->b2, mat->b3);
  mpq_mul (auq1, auq1, mat->a12);
  mpq_mul (auq1, auq1, mat->a12);
  mpq_mul (auq2, mat->a23, mat->a23);
  mpq_mul (auq2, auq2, mat->b3);
  if (!(mpq_cmp (auq1, auq2) >= 0))
    {
      mpq_clear (auq1); mpq_clear (auq2);
      return TRUE;
    }

  mpq_sub (auq1, mat->b2, mat->b3);
  mpq_mul (auq1, auq1, mat->a11);
  mpq_mul (auq2, mat->a22, mat->b3);
  mpq_add (auq1, auq1, auq2);
  mpq_mul (auq2, mat->a33, mat->b2);
  mpq_sub (auq1, auq1, auq2);
  if (!(mpq_cmp_ui (auq1, 0UL, 1UL) >= 0))
    {
      mpq_clear (auq1); mpq_clear (auq2);
      return TRUE;
    }


  mpq_clear (auq1); mpq_clear (auq2);

  return FALSE;
}



/* Assigns an mpq to the first argument which differs not more than 2^{3.Argument} from the 2.Argument */
void from_mpf_to_mpq (mpq_t q, mpf_t f, unsigned long precision)
{
  mpf_mul_2exp (f, f, precision);
  mpz_set_f (mpq_numref (q), f);
  mpz_set_ui (mpq_denref (q), 1UL);
  mpz_mul_2exp (mpq_denref (q), mpq_denref (q), precision);
  mpq_canonicalize (q);
}



/* Assigns an mpq as a perfect quare to the first argument which differs not more than 2^{3.Argument} */
/* from the 2.Argument */
void from_mpf_to_mpq_as_perfect_square (mpq_t q, mpf_t f, unsigned long precision)
{
  unsigned long j=0, k=0;
  mpq_t upper_bound_of_sqrt, auq1, auq2;
  mpf_t auf;

  mpq_init (upper_bound_of_sqrt); mpq_init (auq1); mpq_init (auq2);
  mpf_init2 (auf, PRECISION);


  mpf_set (auf, f);

  while (mpf_integer_p (auf) == 0)
    {
      k++; k++;
      mpf_mul_2exp (auf, auf, 2UL);
    }

  mpf_sqrt (auf, auf);
  mpf_ceil (auf, auf);
  mpz_set_f (mpq_numref (upper_bound_of_sqrt), auf);
```

```
  mpz_set_ui (mpq_denref (upper_bound_of_sqrt), 1UL);
  mpz_mul_2exp (mpq_denref (upper_bound_of_sqrt), mpq_denref (upper_bound_of_sqrt), k/2);
  mpq_canonicalize (upper_bound_of_sqrt);

  do
    {
      mpq_set_ui (auq1, 1UL, 1UL);
      mpq_mul_2exp (auq1, auq1, j);
      mpq_inv (auq1, auq1);

      mpq_set (auq2, upper_bound_of_sqrt);
      mpz_mul_ui (mpq_numref (auq2), mpq_numref (auq2), 2UL);
      mpq_canonicalize (auq2);
      mpq_add (auq2, auq2, auq1);
      mpq_mul (auq2, auq2, auq1);

      mpq_set_ui (auq1, 1UL, 1UL);
      mpq_mul_2exp (auq1, auq1, precision);
      mpq_inv (auq1, auq1);
      j++;
    }
  while (mpq_cmp (auq2, auq1)>0);
  j--;

  mpf_set (auf, f);
  mpf_sqrt (auf, auf);
  mpf_mul_2exp (auf, auf, j);
  mpz_set_f (mpq_numref (q), auf);
  mpz_set_ui (mpq_denref (q), 1UL);
  mpz_mul_2exp (mpq_denref (q), mpq_denref (q), j);
  mpq_canonicalize (q);
  mpq_mul (q, q, q);


  mpq_clear (upper_bound_of_sqrt); mpq_clear (auq1); mpq_clear (auq2);
  mpf_clear (auf);
}




void sort_list_x (GtkToolButton *unused1, GList **list)
{
  sleep (2);
  gtk_widget_set_sensitive (GTK_WIDGET (button_computate), FALSE);
  gtk_widget_set_sensitive (GTK_WIDGET (button_computate), TRUE);
}




gboolean write_generated_matrices_to_file (InputMatrices *im, const gchar *fname)
{
  guint i;
  FILE *stream;


  if (g_file_test (fname, G_FILE_TEST_EXISTS))
    {
      message_file_dialog (DIALOG_FILE_ALREADY_EXISTS);
      return FALSE;
    }

  if ((stream = g_fopen (fname, "w")) == NULL)
    {
      message_file_dialog (DIALOG_FAILED_TO_OPEN_FILE);
      return FALSE;
    }
```

```
  g_fprintf (stream, "bmvdatafile=TRUE\n\n");

  for (i=0; i<=7; i++)
    {
      mpq_out_str (stream, 16, * im->p[i]);
      g_fprintf (stream, "\n");
    }

  g_fprintf (stream, "\n\n");

  fclose (stream);

  return TRUE;
}



/* A function of type GCompareFunc */
gint compare_list_x (ListItem *itema, ListItem *itemb)
{
  return mpq_cmp (itema->x, itemb->x);
}



gboolean max_distance (mpq_t lower_x, mpq_t upper_x, GList *list, mpqRectangle *range)
{
  mpq_t current_lower_x, current_width, range_upper_x, previous_x, auq;

  mpq_init (current_lower_x); mpq_init (current_width); mpq_init (range_upper_x);
  mpq_init (previous_x); mpq_init (auq);


  mpq_add (range_upper_x, range->x, range->width);

  /* search for the first element of list inside range concerning only the x coordinate */
  while (list != NULL && mpq_cmp (((ListItem*) list->data)->x, range->x) < 0)
    {
      list = g_list_next (list);
    }

  /* is there a listitem inside range concerning the x coordinate */
  if (list == NULL || mpq_cmp (range_upper_x, ((ListItem*) list->data)->x) < 0)
    {
      mpq_set (lower_x, range->x);
      mpq_add (upper_x, range->x, range->width);

      mpq_clear (current_lower_x); mpq_clear (current_width); mpq_clear (range_upper_x);
      mpq_clear (previous_x); mpq_clear (auq);
      return TRUE;
    }

  mpq_set (previous_x, range->x);
  mpq_set_ui (current_width, 0UL, 1UL);
  while (list != NULL && mpq_cmp (((ListItem*) list->data)->x, range_upper_x) < 0)
    {
      mpq_sub (auq, ((ListItem*) list->data)->x, previous_x);
      if (mpq_cmp (current_width, auq) < 0)
        {
          mpq_set (current_lower_x, previous_x);
          mpq_set (current_width, auq);
        }

      mpq_set (previous_x, ((ListItem*) list->data)->x);
      list = g_list_next (list);
```

```
    }

  mpq_sub (auq, range_upper_x, previous_x);
  if (mpq_cmp (current_width, auq) < 0)
    {
      mpq_set (current_lower_x, previous_x);
      mpq_set (current_width, auq);
    }

  mpq_set (lower_x, current_lower_x);
  mpq_add (upper_x, lower_x, current_width);


  mpq_clear (current_lower_x); mpq_clear (current_width); mpq_clear (range_upper_x);
  mpq_clear (previous_x); mpq_clear (auq);

  return TRUE;
}



unsigned long necessary_input_precision (mpq_t distance)
{
  unsigned long i=0;
  mpq_t auq1;

  mpq_init (auq1);

  do
    {
      i++;
      mpq_set_ui (auq1, 1UL, 1UL);
      mpq_div_2exp (auq1, auq1, i);
    }
  while (mpq_cmp (distance, auq1) <= 0);

 i+=2;


 mpq_clear (auq1);
 return i;
}



/* Takes argument of type Filedate instead of GList just because b3 is needed to avoid x = b3, this */
/* could also be done outside, also the function max_distance could be applied outside, in this case */
/* fdata and range would not be necessary inside */
void generate_x (mpq_t x, FileData *fdata, mpqRectangle *range)
{
  guint i=0;
  GRand* random;
  double r;
  mpq_t auq1, auq2, auq3, auq4;
  mpf_t f;

  mpq_init (auq1); mpq_init (auq2); mpq_init (auq3); mpq_init (auq4);
  mpf_init2 (f, PRECISION);

  max_distance (auq1, auq2, fdata->list, range);
  mpq_sub (auq2, auq2, auq1);
  gg_mpq_div_ui (auq2, auq2, 3UL);
  mpq_add (auq1, auq1, auq2);
  mpq_add (auq2, auq1, auq2);
  mpq_sub (auq3, auq2, auq1);
  do
    {
```

```
      i++;
      random = g_rand_new ();
      r = (double) g_rand_double_range (random, 0.3, 0.7);
      g_rand_free (random);
      mpq_set_d (auq4, r);
      mpq_mul (auq4, auq3, auq4);
      mpq_add (auq4, auq1, auq4);
      mpf_set_q (f, auq4);  /* auq4 nil */
      from_mpf_to_mpq_as_perfect_square (auq4, f, necessary_input_precision (auq3));
    }
  while (i < TRIES &&
         !(mpq_cmp (auq4, auq1) > 0 &&
           mpq_cmp (auq2, auq4) > 0 &&
           mpq_cmp (auq4, fdata->matrices->b3) != 0
           )
         );

  if (i == TRIES &&
      !(mpq_cmp (auq4, auq1) > 0 &&
        mpq_cmp (auq2, auq4) > 0 &&
        mpq_cmp (auq4, fdata->matrices->b3) != 0
        )
      )
    message_file_dialog (DIALOG_FAILED_TO_GENERATE_X_VALUE);
  else mpq_set (x, auq4);

  mpq_clear (auq1); mpq_clear (auq2); mpq_clear (auq3); mpq_clear (auq4);
  mpf_clear (f);
}



/* By means of the x value together with the global defined input matrices all the input variables to */
/* the function psi and some extra variables are evaluated */
void evaluate_input_definitions (InputDefinitions *id, InputMatrices *mat, mpq_t x)
{
  mpq_t auq1, auq2;

  mpq_init (auq1); mpq_init (auq2);

  /* x2=x/b2 */
  mpq_div (id->x2, x, mat->b2);

  /* x3=x/b3 */
  mpq_div (id->x3, x, mat->b3);

  /* A12=a12*sqrt(x2) */
  mpz_sqrt (mpq_numref (auq1), mpq_numref (id->x2));
  mpz_sqrt (mpq_denref (auq1), mpq_denref (id->x2));
  mpq_canonicalize (auq1);
  mpq_mul (id->A12, mat->a12, auq1);

  /* A13=a13*sqrt(x3) */
  mpz_sqrt (mpq_numref (auq1), mpq_numref (id->x3));
  mpz_sqrt (mpq_denref (auq1), mpq_denref (id->x3));
  mpq_canonicalize (auq1);
  mpq_mul (id->A13, mat->a13, auq1);

  /* lambda=a1*(x2-x3)-a2*x2+a3*x3 */
  mpq_sub (auq1, id->x2, id->x3);
  mpq_mul (auq1, auq1, mat->a11);
  mpq_mul (auq2, mat->a22, id->x2);
  mpq_sub (auq1, auq1, auq2);
  mpq_mul (auq2, mat->a33, id->x3);
  mpq_add (id->lambda, auq1, auq2);  /* auq1, auq2 nil */

  /* mu=a1*(1-x2)+a2*x2 */
```

```
mpq_set_ui (auq1, 1UL, 1UL);
mpq_sub (auq1, auq1, id->x2);
mpq_mul (auq1, auq1, mat->a11);
mpq_mul (auq2, mat->a22, id->x2);
mpq_add (id->mu, auq1, auq2);

/* upsilon=(2*A12*A13)/(A12*A12+A13*A13) */
mpq_mul (auq1, id->A12, id->A12); mpq_mul (auq2, id->A13, id->A13);
mpq_add (auq2, auq1, auq2);
mpq_mul (auq1, id->A12, id->A13);
gg_mpq_mul_ui (auq1, auq1, 2UL);
mpq_div (id->upsilon, auq1, auq2);

/* xi=a23*sqrt(x2*x3) */
mpq_mul (auq1, id->x2, id->x3);
mpz_sqrt (mpq_numref (auq1), mpq_numref (auq1));
mpz_sqrt (mpq_denref (auq1), mpq_denref (auq1));
mpq_canonicalize (auq1);
mpq_mul (id->xi, mat->a23, auq1);

/* omega1=(1-x3)*(A12*A12+A13*A13)/2 */
mpq_mul (auq1, id->A12, id->A12); mpq_mul (auq2, id->A13, id->A13);
mpq_add (auq2, auq1, auq2);
gg_mpq_div_ui (auq2, auq2, 2UL);
mpq_set_ui (auq1, 1UL, 1UL);
mpq_sub (auq1, auq1, id->x3);
mpq_mul (id->omega1, auq1, auq2);

/* omega2=(x3-x2)/((1-x3)*2) */
/* note that auq1 still equals (1-x3) */
gg_mpq_mul_ui (auq1, auq1, 2UL);
mpq_sub (auq2, id->x3, id->x2);
mpq_div (id->omega2, auq2, auq1);

/* omega3=-A13*upsilon/A12 */
mpq_mul (auq2, id->A13, id->upsilon);
mpq_div (auq2, auq2, id->A12);
mpq_neg (id->omega3, auq2);    /* auq1, auq2 nil */

/* omega=1-A13*upsilon/A12 */
mpq_set_ui (auq1, 1UL, 1UL);
mpq_mul (auq2, id->A13, id->upsilon);
mpq_div (auq2, auq2, id->A12);
mpq_sub (id->omega, auq1, auq2);    /* auq1, auq2 nil */

/* tomega=A13*upsilon/A12 */
mpq_mul (id->tomega, id->A13, id->upsilon);
mpq_div (id->tomega, id->tomega, id->A12);

/* a_omega3_p_2 */
mpq_abs (id->a_omega3_p_2, id->omega3);
gg_mpq_add_ui (id->a_omega3_p_2, id->a_omega3_p_2, 2UL);

/* a_omega3_p_2_t_a_lambda */
mpq_mul(id->a_omega3_p_2_t_a_lambda, id->a_omega3_p_2, id->lambda);
mpq_abs(id->a_omega3_p_2_t_a_lambda, id->a_omega3_p_2_t_a_lambda);

/* a_omega3_p_2_t_a_lambda_p_a_xi */
mpq_abs (auq1, id->xi);
mpq_add (id->a_omega3_p_2_t_a_lambda_p_a_xi, id->a_omega3_p_2_t_a_lambda, auq1);

/* beta */
mpq_mul (auq1, id->a_omega3_p_2, id->omega2);
gg_mpq_increment (auq1);

mpq_abs (auq2, id->upsilon);
gg_mpq_increment (auq2);
```

```
  mpq_mul (auq1, auq1, auq2);    /* auq2 nil */
  mpq_mul (auq1, auq1, id->omega1);
  gg_mpq_mul_ui (id->beta, auq1, 2UL);


  mpq_clear (auq1); mpq_clear (auq2);
}



#ifndef MERGE
#ifndef MODUL
void evaluate_factor3 (mpq_t rop, mpq_t error, InputDefinitions *id, pointerbundle *p)
{
  long Lmax, mmax;
  unsigned long k, r, L, rho, m, j, kmax;

  mpz_t m_num,m_den,j_num,j_den;
  mpz_t auz1;

  mpq_t E_k, E_r, E_L, E_rho, E_m, E_j;
  mpq_t E_k_hat, E_r_hat, E_L_hat, E_rho_hat;
  mpq_t auq1, auq2;

  mpq_t g_part_of_comp_Lmax, k_part_of_comp_Lmax, r_part_of_comp_Lmax, comp_Lmax;
  mpq_t g_part_of_comp_mmax, k_part_of_comp_mmax, r_part_of_comp_mmax, L_part_of_comp_mmax;
  mpq_t rho_part_of_comp_mmax, tau_part_of_comp_mmax, comp_mmax;
  mpq_t k_part_of_rho_part_1_of_comp_mmax;

  mpq_t au1_q; mpf_t au1;

  mpq_init (au1_q); mpf_init2 (au1, PRECISION);

  mpz_init (m_num); mpz_init (m_den);
  mpz_init (j_num); mpz_init (j_den);
  mpz_init (auz1);

  mpq_init (E_k); mpq_init (E_r); mpq_init (E_L); mpq_init (E_rho);
  mpq_init (E_m); mpq_init (E_j);
  mpq_init (E_k_hat); mpq_init (E_r_hat); mpq_init (E_L_hat);
  mpq_init (E_rho_hat);
  mpq_init (auq1); mpq_init (auq2);

  mpq_init (g_part_of_comp_Lmax); mpq_init (k_part_of_comp_Lmax); mpq_init (r_part_of_comp_Lmax);
  mpq_init (comp_Lmax); mpq_init (g_part_of_comp_mmax); mpq_init (k_part_of_comp_mmax);
  mpq_init (r_part_of_comp_mmax); mpq_init (L_part_of_comp_mmax); mpq_init (rho_part_of_comp_mmax);
  mpq_init (tau_part_of_comp_mmax); mpq_init (comp_mmax);
  mpq_init (k_part_of_rho_part_1_of_comp_mmax);


  kmax = eval_kmax (error, id->a_omega3_p_2_t_a_lambda_p_a_xi, id->beta, id->a_omega3_p_2, id->omega2);

  /* g_part_of_comp_Lmax */
  gg_mpq_mul_ui (g_part_of_comp_Lmax, id->lambda, 4UL);
  mpq_abs (g_part_of_comp_Lmax, g_part_of_comp_Lmax);
  mpq_abs (auq1, id->xi);
  mpq_add (g_part_of_comp_Lmax, g_part_of_comp_Lmax, auq1);  /* auq1 nil */
  gg_mpq_upper_bound_for_exp (g_part_of_comp_Lmax, g_part_of_comp_Lmax);
  gg_mpq_mul_ui (g_part_of_comp_Lmax, g_part_of_comp_Lmax, kmax);
  mpq_inv (g_part_of_comp_Lmax, g_part_of_comp_Lmax);
  mpq_mul (g_part_of_comp_Lmax, g_part_of_comp_Lmax, error);

  /* g_part_of_comp_mmax */
  mpq_abs (g_part_of_comp_mmax, id->xi);
  gg_mpq_upper_bound_for_exp (g_part_of_comp_mmax, g_part_of_comp_mmax);
  gg_mpq_mul_ui (g_part_of_comp_mmax, g_part_of_comp_mmax, kmax);
  mpq_inv (g_part_of_comp_mmax, g_part_of_comp_mmax);
  mpq_mul (g_part_of_comp_mmax, g_part_of_comp_mmax, error);
```

```
num_of_add_ops = 0;
mpq_set_ui (rop, 0UL, 1UL);
for (k=1;k <= kmax+KMAXEXTRA ;k++)
  {
    /* E_k_hat */
    gg_mpq_pow_ui (E_k_hat, id->omega1, k);
    mpz_fac_ui (auz1, k-1);
    mpz_mul (mpq_denref (E_k_hat), mpq_denref (E_k_hat), auz1);
    mpq_canonicalize (E_k_hat);

    /* k_part_of_comp_Lmax without with help from auz1 */
    mpq_abs (auq1, id->upsilon);
    gg_mpq_increment (auq1);
    gg_mpq_pow_ui (auq1, auq1, k);
    mpq_mul_2exp (k_part_of_comp_Lmax, auq1, k+1);
    mpq_mul (k_part_of_comp_Lmax, k_part_of_comp_Lmax, E_k_hat);
    mpq_inv (k_part_of_comp_Lmax, k_part_of_comp_Lmax);
    mpz_mul (mpq_numref (k_part_of_comp_Lmax), mpq_numref (k_part_of_comp_Lmax), auz1);
    mpq_canonicalize (k_part_of_comp_Lmax);
    mpq_mul (k_part_of_comp_Lmax, k_part_of_comp_Lmax, g_part_of_comp_Lmax);

    /* k_part_of_comp_mmax */
    mpq_set_ui (k_part_of_comp_mmax, 1UL, k);
    mpq_div (k_part_of_comp_mmax, k_part_of_comp_mmax, E_k_hat);
    mpq_mul (k_part_of_comp_mmax, k_part_of_comp_mmax, g_part_of_comp_mmax);

    /* k_part_of_rho_part_1_of_comp_mmax with help from auz1 and auq1 */
    mpq_inv (k_part_of_rho_part_1_of_comp_mmax, auq1);  /* auq1 nil */
    mpz_mul_ui (auz1, auz1, k);  /* auz1= k! */
    mpz_mul (mpq_numref (k_part_of_rho_part_1_of_comp_mmax),
             mpq_numref (k_part_of_rho_part_1_of_comp_mmax), auz1);  /* auz1 nil */
    mpq_canonicalize (k_part_of_rho_part_1_of_comp_mmax);


    mpq_set_ui (E_k, 0UL, 1UL);
    for (r=0; r<=k-1; r++)
      {
        /* E_r_hat */
        gg_mpq_pow_ui (E_r_hat, id->omega2, r);
        mpz_bin_uiui (auz1, k-1, r);
        mpz_mul (mpq_numref (E_r_hat), mpq_numref (E_r_hat), auz1);  /* auz1 nil */
        mpq_canonicalize (E_r_hat);

        /* r_part_of_comp_Lmax */
        mpq_mul_2exp (r_part_of_comp_Lmax, E_r_hat, 2*r);
        mpq_inv (r_part_of_comp_Lmax, r_part_of_comp_Lmax);
        mpq_mul (r_part_of_comp_Lmax, r_part_of_comp_Lmax, k_part_of_comp_Lmax);

        /* Lmax */
        mpq_set_ui (auq1, 1UL, 1UL);
        if(mpq_cmp (auq1, r_part_of_comp_Lmax) <= 0) Lmax=-2;
        else
          {
            gg_mpq_mul_ui (auq2, id->lambda, 4UL);
            mpq_abs (auq2, auq2);
            Lmax=-1;
            do
              {
                Lmax++;
                mpq_mul (auq1, auq1, auq2);
                gg_mpq_div_ui (auq1, auq1, Lmax+1);
              }
            while(!(mpq_cmp (auq1, r_part_of_comp_Lmax) <= 0));
          }

        /* r_part_of_comp_mmax */
        mpq_set_ui (r_part_of_comp_mmax, 1UL, Lmax+1);
```

```
mpq_div (r_part_of_comp_mmax, r_part_of_comp_mmax, E_r_hat);
mpq_mul (r_part_of_comp_mmax, r_part_of_comp_mmax, k_part_of_comp_mmax);

mpq_set_ui (E_r, 0UL, 1UL);
for (L=0; (long) L <= Lmax+LMAXEXTRA; L++)
  {
    if (stop == TRUE) {g_print ("ciao\n"); g_thread_exit (NULL);}

    /* indices output */
    g_string_printf (p->indices->gstring, "kmax = %5ld\n        %5lu   %5lu   %5ld\n"
                     "                              %5lu", kmax, k, r, Lmax, L);
    gdk_threads_enter ();
    gtk_label_set_text (p->indices->label, p->indices->gstring->str);
    gdk_flush ();
    gdk_threads_leave ();

    /* E_L_hat */
    mpq_neg (auq1, id->lambda);
    gg_mpq_pow_ui (E_L_hat, auq1, L);
    mpz_fac_ui (auz1, L);
    mpz_mul (mpq_denref (E_L_hat), mpq_denref (E_L_hat), auz1);
    mpq_canonicalize (E_L_hat);

    /* L_part_of_comp_mmax */
    mpq_set_ui (L_part_of_comp_mmax, 1UL, MIN(r+L, k)+1);
    mpq_div (L_part_of_comp_mmax, L_part_of_comp_mmax, E_L_hat);
    mpq_abs (L_part_of_comp_mmax, L_part_of_comp_mmax);
    mpq_mul (L_part_of_comp_mmax, L_part_of_comp_mmax, r_part_of_comp_mmax);

    mpq_set_ui (E_L, 0UL, 1UL);
    for (rho=0; rho <= MIN(r+L, k); rho++)
      {
        /* E_rho_hat */
        gg_mpq_pow_ui (E_rho_hat, id->omega3, rho);
        mpz_bin_uiui (auz1, r+L, rho);
        mpz_mul (mpq_numref (E_rho_hat), mpq_numref (E_rho_hat), auz1);
        mpq_canonicalize (E_rho_hat);

        /* rho_part_of_comp_mmax */
        mpq_set_ui (rho_part_of_comp_mmax, 1UL, 1UL);
        mpq_mul_2exp (rho_part_of_comp_mmax, rho_part_of_comp_mmax, k+r+L-rho+1);
        mpq_mul (rho_part_of_comp_mmax, rho_part_of_comp_mmax, E_rho_hat);
        mpq_inv (rho_part_of_comp_mmax, rho_part_of_comp_mmax);
        mpq_abs (rho_part_of_comp_mmax, rho_part_of_comp_mmax);
        mpq_mul (rho_part_of_comp_mmax, rho_part_of_comp_mmax, L_part_of_comp_mmax);

        mpz_ui_pow_ui (auz1, k, rho);
        mpq_set_z (auq1, auz1);
        mpq_div (k_part_of_rho_part_1_of_comp_mmax,
                 k_part_of_rho_part_1_of_comp_mmax, auq1);  /* auq1 nil */
        mpq_set_ui (auq1, 100UL, 272UL);
        gg_mpq_max (auq1, auq1, k_part_of_rho_part_1_of_comp_mmax);

        mpq_mul (rho_part_of_comp_mmax, rho_part_of_comp_mmax, auq1);

        /* mmax */
        mpq_set_ui (auq1, 1UL, 1UL);
        mpz_ui_pow_ui (auz1, k+r+L-rho, rho);
        mpq_set_z (auq2, auz1); /*  auz1 nil */
        mpq_mul (auq2, auq2, rho_part_of_comp_mmax);
        if (mpq_cmp (auq1, auq2) <= 0) mmax=-2;  /* auq1, auq2 nil */
        else
          {
            mmax=-1;
            mpq_set_ui (auq1, 1UL, 1UL);
            do
              {
                mmax++;
```

```
            mpq_mul (auq1, auq1, id->xi);
            mpq_abs (auq1, auq1);
            mpz_mul_ui (mpq_denref(auq1), mpq_denref(auq1), mmax+1);
            mpq_canonicalize (auq1);

            mpq_set (auq2, auq1);
            mpz_ui_pow_ui (auz1, ((unsigned long) mmax)+1+k+r+L-rho, rho);
            mpz_mul (mpq_denref (auq2), mpq_denref (auq2), auz1);
            mpq_canonicalize (auq2);
          }
        while(!(mpq_cmp(auq2, rho_part_of_comp_mmax) <= 0));
      }

    mpq_set_ui (E_rho, 0UL, 1UL);
    for (m=0; (long) m <= mmax+MMAXEXTRA; m++)
      {
        /* numerator */
        mpz_pow_ui (m_num, mpq_numref(id->xi), m);
        /* denumerator */
        mpz_pow_ui (m_den, mpq_denref(id->xi), m);
        mpz_fac_ui (auz1, m);
        mpz_mul (m_den, m_den, auz1);
        gg_mpz_rf (auz1, m+1, k+r+L-1);
        mpz_mul (m_den, m_den, auz1);

        mpq_set_ui (E_m, 0UL, 1UL);
        for (j=m%2; j+rho <= MIN(m+rho, k); j=j+2)
          {
            if (j<2)
              {
                /* numerator */
                gg_mpz_rf2 (auz1, m-j+2, k+r+L-rho-1);
                mpz_pow_ui (j_num, mpq_numref (id->upsilon), j);
                mpz_mul (j_num, j_num, auz1);
                /* denumerator */
                mpz_pow_ui (j_den, mpq_denref (id->upsilon), j);
                mpz_fac_ui (auz1, j);
                mpz_mul (j_den, j_den, auz1);
                mpz_fac_ui (auz1, k-j-rho);
                mpz_mul (j_den, j_den, auz1);
              }
            else
              {
                if (k+r+L-rho-1>0)
                  {
                    mpz_mul_ui (j_num, j_num, m-j+2);
                    mpz_mul_ui (j_den, j_den, m-j+4+2*(k+r+L-rho-2));
                  }
                mpz_mul (j_num, j_num, mpq_numref (id->upsilon));
                mpz_mul (j_num, j_num, mpq_numref (id->upsilon));
                mpz_mul (j_den, j_den, mpq_denref (id->upsilon));
                mpz_mul (j_den, j_den, mpq_denref (id->upsilon));
                mpz_mul_ui (j_den, j_den, j-1);
                mpz_mul_ui (j_den, j_den, j);
                mpz_mul_ui (j_num, j_num, k-j-rho+1);
                mpz_mul_ui (j_num, j_num, k-j-rho+2);
              }

            mpz_set (mpq_numref (E_j), j_num);
            mpz_set (mpq_denref (E_j), j_den);
            mpq_canonicalize (E_j);
            mpq_add (E_m, E_m, E_j); num_of_add_ops++;
          }
        mpz_mul (mpq_numref (E_m), mpq_numref (E_m), m_num);
        mpz_mul (mpq_denref (E_m), mpq_denref (E_m), m_den);
        mpq_canonicalize (E_m);
        mpq_add (E_rho, E_rho, E_m); num_of_add_ops++;
```

```
                    }
                  mpq_mul (E_rho, E_rho, E_rho_hat);
                  mpq_add (E_L, E_L, E_rho); num_of_add_ops++;
                }
              mpq_mul (E_L, E_L, E_L_hat);
              mpq_add (E_r, E_r, E_L); num_of_add_ops++;
            }
          mpq_mul (E_r, E_r, E_r_hat);
          mpq_add (E_k, E_k, E_r); num_of_add_ops++;
        }
      mpq_mul (E_k, E_k, E_k_hat);
      mpq_add (rop, rop, E_k); num_of_add_ops++;
    }

  mpq_clear (au1_q); mpf_clear (au1);

  mpz_clear (m_num); mpz_clear (m_den);
  mpz_clear (j_num); mpz_clear (j_den);
  mpz_clear (auz1);

  mpq_clear (E_k); mpq_clear (E_r); mpq_clear (E_L); mpq_clear (E_rho);
  mpq_clear (E_m); mpq_clear (E_j);
  mpq_clear (E_k_hat); mpq_clear (E_r_hat); mpq_clear (E_L_hat);
  mpq_clear (E_rho_hat);
  mpq_clear (auq1); mpq_clear (auq2);

  mpq_clear (g_part_of_comp_Lmax); mpq_clear (k_part_of_comp_Lmax); mpq_clear (r_part_of_comp_Lmax);
  mpq_clear (comp_Lmax); mpq_clear (g_part_of_comp_mmax); mpq_clear (k_part_of_comp_mmax);
  mpq_clear (r_part_of_comp_mmax); mpq_clear (L_part_of_comp_mmax); mpq_clear (rho_part_of_comp_mmax);
  mpq_clear (tau_part_of_comp_mmax); mpq_clear (comp_mmax);
  mpq_clear (k_part_of_rho_part_1_of_comp_mmax);
}
#endif
#endif



void evaluate_ubound_of_abs_psi (mpq_t ubound, InputMatrices *mat, mpq_t x)
{
  mpq_t factor1;
  mpq_t ubound2, ubound3;
  mpq_t auq1;
  InputDefinitions *id = InputDefinitions_with_gmp_variables_new ();

  mpq_init (factor1);
  mpq_init (ubound2); mpq_init (ubound3);
  mpq_init (auq1);


  evaluate_input_definitions (id, mat, x);

  /* factor1 = 2/(x(1-x3)) */
  mpq_set_ui (factor1, 1UL, 1UL);
  mpq_sub (factor1, factor1, id->x3);
  mpq_mul (factor1, factor1, x);
  mpq_inv (factor1, factor1);
  gg_mpq_mul_ui (factor1, factor1, 2UL);

  /* ubound2  */
  mpq_add (ubound2, id->lambda, id->mu);
  gg_mpq_upper_bound_for_exp (ubound2, ubound2);

  /* ubound3 */
  gg_mpq_upper_bound_for_exp (ubound3, id->a_omega3_p_2_t_a_lambda_p_a_xi);

  gg_mpq_upper_bound_for_exp (auq1, id->beta);
  gg_mpq_decrement (auq1);
```

```
  mpq_mul (ubound3, ubound3, auq1);  /* auq1 nil */

  mpq_mul (auq1, id->a_omega3_p_2, id->omega2);
  gg_mpq_increment (auq1);
  gg_mpq_mul_ui (auq1, auq1, 2UL);
  mpq_div (ubound3, ubound3, auq1);  /* auq1 nil */

  /* auq1 = upper bound for abs psi */
  mpq_mul (auq1, factor1, ubound2);
  mpq_mul (ubound, auq1, ubound3);

  mpq_clear (factor1);
  mpq_clear (ubound2); mpq_clear (ubound3);
  mpq_clear (auq1);
  InputDefinitions_with_gmp_variables_free (id);
}



void evaluate_psi_with_error (mpq_t psi, InputMatrices *im, mpq_t x, long lberror, pointerbundle *p)
{
  mpq_t factor1, factor2, factor3;
  mpq_t ubound2, ubound3;
  mpq_t error2, error3;
  mpq_t auq1, auq2;
  InputDefinitions *id;

  id = InputDefinitions_with_gmp_variables_new ();

  mpq_init (factor1); mpq_init (factor2); mpq_init (factor3);
  mpq_init (ubound2); mpq_init (ubound3);
  mpq_init (error2); mpq_init (error3);
  mpq_init (auq1); mpq_init (auq2);

  evaluate_input_definitions (id, im, x);

  /* factor1 = 2/(x(1-x3)) */
  mpq_set_ui (factor1, 1UL, 1UL);
  mpq_sub (factor1, factor1, id->x3);
  mpq_mul (factor1, factor1, x);
  mpq_inv (factor1, factor1);
  gg_mpq_mul_ui (factor1, factor1, 2UL);

  /* ubound2 (upper bound for abs of factor2) */
  mpq_add (ubound2, id->lambda, id->mu);
  gg_mpq_upper_bound_for_exp (ubound2, ubound2);

  /* ubound3 */
  gg_mpq_upper_bound_for_exp (ubound3, id->a_omega3_p_2_t_a_lambda_p_a_xi);

  gg_mpq_upper_bound_for_exp (auq1, id->beta);
  gg_mpq_decrement (auq1);
  mpq_mul (ubound3, ubound3, auq1);  /* auq1 nil */

  mpq_mul (auq1, id->a_omega3_p_2, id->omega2);
  gg_mpq_increment (auq1);
  gg_mpq_mul_ui (auq1, auq1, 2UL);
  mpq_inv (auq1, auq1);
  mpq_mul (ubound3, ubound3, auq1);  /* auq1 nil */

  /* error3 */
  gg_mpq_2exp_si (auq1, lberror);

  mpq_div (auq1, auq1, factor1);

  mpq_div (error3, auq1, ubound2);
  mpz_mul_ui (mpq_numref (error3), mpq_numref (error3), 9UL);
```

```
  mpz_mul_ui (mpq_denref (error3), mpq_denref (error3), 10UL);
  mpq_canonicalize (error3);

  /* error2 (with a little help from auq1) */
  gg_mpq_div_ui (auq1, auq1, 10UL);
  mpq_add (auq2, ubound3, error3);
  mpq_div (error2, auq1, auq2);  /* auq1, auq2 nil */

  /* factor2 */
  mpq_add (factor2, id->lambda, id->mu);
  gg_mpq_exp (factor2, factor2, error2);  if (mpq_cmp_ui (factor2, 0UL, 1UL) < 0) g_printf ("sch...\n");

  /* factor3 */
  evaluate_factor3 (factor3, error3, id, p);

  mpq_mul (psi, factor3, factor2);
  mpq_mul (psi, psi, factor1);


  mpq_clear (factor1); mpq_clear (factor2); mpq_clear (factor3);
  mpq_clear (ubound2); mpq_clear (ubound3);
  mpq_clear (error2); mpq_clear (error3);
  mpq_clear (auq1); mpq_clear (auq2);
  InputDefinitions_with_gmp_variables_free (id);
}



glong evaluate_psi (mpq_t psi, InputMatrices *mat, mpq_t x, pointerbundle *p)
{
  glong lberror;
  mpq_t auq1, auq2;

  mpq_init (auq1); mpq_init (auq2);

  evaluate_ubound_of_abs_psi (auq1, mat, x);

  /* lberror */
  lberror=0;
  mpq_set_ui (auq2, 1UL, 1UL);
  while (mpq_cmp (auq2, auq1) < 0)  /* if ubound > 1 */
    {
      lberror++;
      gg_mpq_mul_ui (auq2, auq2, 2UL);
    }
  while (mpq_cmp (auq2, auq1) > 0)  /* if ubound < 1 */
    {
      lberror--;
      gg_mpq_div_ui (auq2, auq2, 2UL);
    }
  /* Here should be 2^lberror \le ubound \le 2^{lberror+1}, hence a starting point to further  */
  /* decrease lberror, in the next do-while-loop lberror will be decreased until computation */
  /* delivers a psi not= 0 what then will be the first richtwert to estimate the final */
  /* result and hence the required error. */

#ifdef MODUL
      evaluate_psi_with_error (psi, mat, x, lberror, p);
#else
  do
    {
      lberror = lberror-5;

      gdk_threads_enter();
      gtk_spin_button_set_value (spinbutton_result_accuracy, (gdouble) lberror);
      gdk_threads_leave();

      evaluate_psi_with_error (psi, mat, x, lberror, p);
```

```
      mpq_set_ui (auq2, 0UL, 1UL);
    }
  while (mpq_cmp (psi, auq2) == 0);
  /* Here lberror has the biggest value so that psi not equals zero */

  gg_mpq_2exp_si (auq2, lberror);
  if (!(mpq_cmp (auq2, psi) < 0))
    do
      {
        mpq_abs (auq1, psi);
        gg_mpq_div_ui (auq1, auq1, BMVERROR);
        while (mpq_cmp (auq2, auq1) >= 0)
          {
            lberror--;
            gg_mpq_div_ui (auq2, auq2, 2UL);
          }

        gdk_threads_enter();
        gtk_spin_button_set_value (spinbutton_result_accuracy, (gdouble) lberror);
        gdk_threads_leave();

        evaluate_psi_with_error (psi, mat, x, lberror, p);

        mpq_abs (auq1, psi);
        gg_mpq_div_ui (auq1, auq1, BMVERROR);
      }
    while (!(mpq_cmp (auq2, auq1) < 0));
#endif

  mpq_clear (auq1); mpq_clear (auq2);

  return lberror;
}



/* Note that in the following the function evaluate_psi_with_error ist applied frequently with */
/* different values of lberror, because of this some components of this functions witch do not depend on */
/* lberror are calculated more often than necessary */
gpointer computate (pointerbundle *p)
{
  guint i;
  glong lberror;
  ListItem *item;
  mpq_t x;
  mpq_t result;
  mpf_t f;

  mpq_init (x);
  mpq_init (result);

  mpf_init2 (f, PRECISION);

  g_timeout_add (200, (GSourceFunc) print_num_of_add_ops_to_label, p);

  for (i=1; i <= COMPUTATIONS; i++)
    {
      /* if (! compare_matrices (p->filedata->matrices)) g_print ("Change!!\n"); */
      gdk_threads_enter();
      if (!g_file_test (gtk_entry_get_text (p->entry_data_file), G_FILE_TEST_EXISTS))
        {
          gint i;

          for (i=0; i<=7; i++) mpf_set_d (p->filedata->matrices->f[i],
                                          gtk_spin_button_get_value (spinbutton_matrices[i]));
          for (i=0; i<=5; i++) from_mpf_to_mpq (* p->filedata->matrices->p[i],
```

```
                                                p->filedata->matrices->f[i],
                                                DISPLAYED_DIGITS*5);
      for (i=6; i<=7; i++) from_mpf_to_mpq_as_perfect_square (* p->filedata->matrices->p[i],
                                                              p->filedata->matrices->f[i],
                                                              DISPLAYED_DIGITS*5);
      adopt_matrices_from_q (p->filedata->matrices);
      for (i=0; i<=7; i++)
        gtk_spin_button_set_value (spinbutton_matrices[i], p->filedata->matrices->d[i]);

      p->filedata->name = g_string_assign (p->filedata->name, gtk_entry_get_text (p->entry_data_file));
      p->filedata->list = NULL;
      write_filedata_to_file (p->filedata);
      load_file_into_gui (p->filedata->name->str, p->filedata, p->display, p->entry_data_file);
    }
gdk_threads_leave();

generate_x (x, p->filedata, p->display->range);

gdk_threads_enter();
gtk_spin_button_set_value (spinbutton_x, (gdouble) mpq_get_d (x));
/* lberror = (glong) gtk_spin_button_get_value_as_int (spinbutton_result_accuracy); */
gdk_threads_leave();

if (mpq_cmp (x, p->filedata->matrices->b3) < 0)
  lberror = evaluate_psi (result, p->filedata->matrices, x, p);

if (mpq_cmp (p->filedata->matrices->b3, x) < 0)
  {
    InputMatrices *mat;
    mpq_t auq;
    mpf_t auf;

    mat = InputMatrices_with_gmp_variables_new ();
    mpq_init (auq);
    mpf_init2 (auf, PRECISION);

    mpq_set (mat->a11, p->filedata->matrices->a22);
    mpq_set (mat->a22, p->filedata->matrices->a11);
    mpq_set (mat->a33, p->filedata->matrices->a33);
    mpq_set (mat->a12, p->filedata->matrices->a12);
    mpq_set (mat->a13, p->filedata->matrices->a23);
    mpq_set (mat->a23, p->filedata->matrices->a13);
    mpq_set (mat->b2, p->filedata->matrices->b2);

    mpq_sub (mat->b3, p->filedata->matrices->b2, p->filedata->matrices->b3);
    mpf_set_q (auf, mat->b3);
    from_mpf_to_mpq_as_perfect_square (mat->b3, auf, INPUT_PRECISION);

    mpq_sub (auq, p->filedata->matrices->b2, x);
    mpf_set_q (auf, auq);
    from_mpf_to_mpq_as_perfect_square (auq, auf, INPUT_PRECISION);

    lberror = evaluate_psi (result, mat, auq, p);

    InputMatrices_with_gmp_variables_free (mat);
    mpq_clear (auq);
    mpf_clear (auf);
  }

item = ListItem_with_gmp_variables_new ();
item->lberror = lberror;
mpq_set (item->x, x);
mpq_set (item->fv, result);
p->filedata->list = g_list_insert_sorted (p->filedata->list, item, (GCompareFunc) compare_list_x);
write_filedata_to_file (p->filedata);

mpf_set_q (f, result);
mpf_out_str (stdout, 10, 30, f);
```

```
        g_print ("\n");

        gdk_threads_enter();
        load_file_into_gui (p->filedata->name->str, p->filedata, p->display, p->entry_data_file);
        gdk_threads_leave();

        g_print ("computation ready\n\n");
      }

    mpq_clear (x);
    mpq_clear (result);

    mpf_clear (f);

    gdk_threads_enter ();
    gtk_button_set_label (button_computate, "start");
    gdk_flush();  /* to update display, worked also without this, look at gtk faq */
    gdk_threads_leave ();

    computation = FALSE;
    /* it's not over yet, but !(thread == NULL), so there won't be created  */
    /* a new one before this one is not over (look at the computation_on_off signal handler) */
    /* it's also important to notice that this instruction has to be after the one before  */
    /* with much bad luck (everthing what can happen will happen) computation is turned to FALSE */
    /* and the computation_on_off signal handler is joining this thread before the instruction */
    /* gdk_threads_enter!!, can not happen any more since joining is now done by a timeout function */

    return NULL;
}



gboolean min_max_functionvalue_within_intervall (mpq_t min, mpq_t max, GList *list, mpqIntervall *in)
{
    GList *llist = list;
    ListItem *item = (ListItem*) llist->data;

    if (llist == NULL) return FALSE;

    mpq_set_ui (min, 0UL, 1UL);
    mpq_set_ui (max, 0UL, 1UL);

    while (llist != NULL && (mpq_cmp (in->a, item->x) >= 0))
      {
        llist = llist->next;
        if (llist != NULL) item = (ListItem*) llist->data;
      }

    while (llist != NULL && mpq_cmp (in->a, item->x) < 0 && mpq_cmp (item->x, in->b) < 0)
      {
        gg_mpq_min (min, min, item->fv);
        gg_mpq_max (max, max, item->fv);
        llist = llist->next;
        if (llist != NULL) item = (ListItem*) llist->data;
      }

    return TRUE;
}



void trace_function (FileData *fdata)
{
    short i;
    GList *llist;
```

```
mpq_t zq, auq1, auq2, auq3, auq4, expp;
mpq_t mat[3][3];

mpq_init (zq); mpq_init (auq1); mpq_init (auq2); mpq_init (auq3); mpq_init (auq4); mpq_init (expp);
gg_mpq_matrix_init (mat);


gg_mpq_2exp_si (expp, LB_EXP_PRECISION);

for (i=1; i<=10; i++)
  {
    llist = fdata->list;

    if (i==1) mpq_set_si (zq, 0, 1UL);
    if (i==2) mpq_set_si (zq, 1, 1UL);
    if (i==3) mpq_set_si (zq, 2, 1UL);
    if (i==4) mpq_set_si (zq, 3, 1UL);
    if (i==5) mpq_set_si (zq, 4, 1UL);
    if (i==6) mpq_set_si (zq, 5, 1UL);
    if (i==7) mpq_set_si (zq, 6, 1UL);
    if (i==8) mpq_set_si (zq, 7, 1UL);
    if (i==9) mpq_set_si (zq, 8, 1UL);
    if (i==10) mpq_set_si (zq, 9, 1UL);

    gg_mpq_matrix_set_A_minus_zB (mat, fdata->matrices, zq);

    gg_mpq_matrix_exp (mat, mat);

    gg_mpq_matrix_trace (auq4, mat);


    gg_mpq_exp (auq1, fdata->matrices->a11, expp);


    gg_mpq_exp (auq2, fdata->matrices->a22, expp);
    mpq_mul (auq3, fdata->matrices->b2, zq);
    mpq_neg (auq3, auq3);
    gg_mpq_exp (auq3, auq3, expp);
    mpq_mul (auq2, auq2, auq3);

    mpq_add (auq1, auq1, auq2);


    gg_mpq_exp (auq2, fdata->matrices->a33, expp);
    mpq_mul (auq3, fdata->matrices->b3, zq);
    mpq_neg (auq3, auq3);
    gg_mpq_exp (auq3, auq3, expp);
    mpq_mul (auq2, auq2, auq3);

    mpq_add (auq1, auq1, auq2);  /* auq2, auq3 nil */

    do
      {
        if (llist->next != NULL)
          {
            mpq_sub (auq3, ((ListItem*) llist->next->data)->x, ((ListItem*) llist->data)->x);
            gg_mpq_div_ui (auq3, auq3, 2UL);
            mpq_add (auq3, ((ListItem*) llist->data)->x, auq3);
          }
        else mpq_set (auq3, fdata->matrices->b2);

        if (llist->prev != NULL)
          {
            mpq_sub (auq2, ((ListItem*) llist->data)->x, ((ListItem*) llist->prev->data)->x);
            gg_mpq_div_ui (auq2, auq2, 2UL);
            mpq_sub (auq2, ((ListItem*) llist->data)->x, auq2);
          }
        else mpq_set_ui (auq2, 0UL, 1UL);
```

```
            mpq_sub (auq2, auq3, auq2);  /* auq3 nil */

            mpq_mul (auq3, ((ListItem*) llist->data)->x, zq);
            mpq_neg (auq3, auq3);
            gg_mpq_exp (auq3, auq3, expp);

            mpq_mul (auq3, auq3, ((ListItem*) llist->data)->fv);
            mpq_mul (auq3, auq3, auq2);

            mpq_add (auq1, auq1, auq3);

            llist = llist->next;
          }
        while (llist != NULL);

        gg_mpq_print_as_mpf (zq);
        gg_mpq_print_as_mpf (auq4);
        gg_mpq_print_as_mpf (auq1);
        mpq_div (auq1, auq4, auq1);
        gg_mpq_print_as_mpf (auq1);

        g_printf ("\n");
      }


    mpq_clear (zq); mpq_clear (auq1); mpq_clear (auq2); mpq_clear (auq3); mpq_clear (auq4); mpq_clear (expp);
    gg_mpq_matrix_clear (mat);
}



gboolean verify (GtkWidget *widget, FileData *fdata)
{
    trace_function (fdata);

    return TRUE;
}
```

B.2 BMV.h

```
#define FACTOR 1
#define PRECISION 500UL
#define WIDTH 600*FACTOR
#define HEIGHT 400*FACTOR
#define TOPMARGIN 10.5*FACTOR
#define BOTTOMMARGIN 30.5*FACTOR
#define LEFTMARGIN 75.5*FACTOR
#define RIGHTMARGIN 30.5*FACTOR
#define DIAGRAMWIDTH WIDTH-LEFTMARGIN-RIGHTMARGIN
#define DIAGRAMHEIGHT HEIGHT-BOTTOMMARGIN-TOPMARGIN
#define FONTSIZE 14*FACTOR
#define LINE_WIDTH_FOR_CROSS 1.*FACTOR  /* cross looks ugly whithout the point, or cast to double */
#define LINEWIDTH 1.*FACTOR
#define TRIES 1000
#define QTRIES "1000"  /* quoted tries: should be synchronized with TRIES */
#define BMVERROR 1024UL  /* the error will be < 1/BMVERROR * ABS (result) */
#define INPUT_PRECISION 100UL
#define DISPLAYED_DIGITS 3
#define LB_EXP_PRECISION -100
#define MATEXP 300
#define COMPUTATIONS 70
#define KMAXEXTRA 0
#define LMAXEXTRA 0
#define MMAXEXTRA 0
```

```
typedef enum
  {
    DIALOG_FILE_NOT_FOUND,
    DIALOG_FAILED_TO_OPEN_FILE,
    DIALOG_FILE_IS_A_DIRECTORY,
    DIALOG_FILE_IS_NOT_A_REGULAR_BMV_DATA_FILE,
    DIALOG_FAILED_TO_GENERATE_MATRICES,
    DIALOG_FAILED_TO_GENERATE_X_VALUE,
    DIALOG_FILE_ALREADY_EXISTS
  }
MessageDialogeType;


typedef enum
  {
    INIT,
    CLEAR
  }
InitOrClear;


typedef enum
  {
    PAIR,
    TRIPLE,
    MPQ_INTERVALL,
    MPQ_RECTANGLE,
    QUINTUPLE,
    INPUT_DEFINITIONS,
    INPUT_MATRICES,
    MPQ_MATRIX
  }
mpqBundleType;


typedef struct
{
  long lberror;
  mpq_t x;
  mpq_t fv;
}
ListItem;


typedef struct
{
  mpq_t a, b;
}
mpqPair;


typedef struct
{
  mpq_t a, b, c;
}
mpqTriple;


typedef struct
{
  mpq_t a, b, length;
}
mpqIntervall;


typedef struct
{
  mpq_t x, y, width, height, auqx, auqy;
```

```
}
mpqRectangle;


typedef struct
{
  mpq_t a, b, c, d, e;
}
mpqQuintuple;


typedef struct
{
  mpq_t a11, a22, a33, a12, a13, a23, b2, b3;
  mpq_t *p[8];
  mpf_t f[8];
  double d[8];
}
InputMatrices;


typedef struct
{
  GString *name;
  InputMatrices *matrices;
  GList *list;
}
FileData;


typedef struct
{
  mpq_t x2, x3, A12, A13, lambda, mu, upsilon, xi, omega1, omega2, omega, omega3;
  mpq_t tomega;
  mpq_t a_omega3_p_2;
  mpq_t a_omega3_p_2_t_a_lambda;
  mpq_t a_omega3_p_2_t_a_lambda_p_a_xi;
  mpq_t beta;
}
InputDefinitions;


typedef struct
{
  GtkLabel *label;
  GString *gstring;
}
GtkLabelGString;


typedef struct
{
  cairo_surface_t *mi;
  cairo_t *cmi;
  GtkDrawingArea *drawingarea;
  /* cairo context which connects to drawingarea after realization of drawingarea */
  cairo_t *cd;

  mpqRectangle *range;
  GdkRectangle *focus;


  /* auxilliary pixmap (GdkDrawable) serves as intermediate station to copy the drawed memory image to */
  /* the background of drawingarea */
  GdkPixmap *pixmap;
  cairo_t *cp;

  /* auxilliary variables */
```

```
  cairo_matrix_t *matrix;
  mpq_t auq;
  gdouble x, y;
}
Display;


/* structure passed as argument to signal handler functions */
typedef struct
{
  gboolean *computation;
  GThread *thread;
  gboolean *stop;
  GtkLabelGString *add_ops;
  GtkLabelGString *indices;
  FileData *filedata;
  Display *display;
  InputMatrices *generated_matrices;
  GtkEntry *entry_data_file;
  gpointer misc;
}
pointerbundle;



gint delete_event_window (GtkWidget *, GdkEvent *, gpointer);
void destroy (GtkWidget *, gpointer);
void fileselection_ok_button (GtkButton *, pointerbundle *);
void draw_on_display (FileData *, Display *);
void print_matrices_as_mpf (InputMatrices *);
void failed_to_open_file_dialog (void);
void activate_entry_data_file (GtkEntry *, pointerbundle *);
void write_listitem_to_file (ListItem *, FILE *);
void maximum_functionvalue (ListItem *, mpq_t);
void minimum_functionvalue (ListItem *, mpq_t);
void file_not_found_dialog (void);
gboolean verify (GtkWidget *, FileData *);
void message_file_dialog (MessageDialogeType);
void generate_matrices_sh (GtkButton *, InputMatrices *);
void generate_matrices (InputMatrices *, unsigned long, double);
void generate_matrices_as_double (double *, double, double);
gboolean proof_generated_matrices (InputMatrices *);
void from_mpf_to_mpq (mpq_t, mpf_t, unsigned long);
void from_mpf_to_mpq_as_perfect_square (mpq_t, mpf_t, unsigned long);
gboolean write_filedata_to_file (FileData *);
gboolean check_and_load_file_into_memory (FileData *, const gchar *);
void sort_list_x (GtkToolButton *, GList **);
gint compare_list_x (ListItem *, ListItem *);
gboolean max_distance (mpq_t, mpq_t, GList *, mpqRectangle *);
void computation_on_off (GtkButton *, pointerbundle *);
gpointer computate (pointerbundle *);
unsigned long necessary_input_precision (mpq_t);
void evaluate_factor3 (mpq_t, mpq_t, InputDefinitions *, pointerbundle *);
void textbuffer_print_handler (const gchar *);
gboolean create_stop_thread (pointerbundle *);
gboolean print_num_of_add_ops_to_label (pointerbundle *);
void make_cross (ListItem *, Display *);
void create_cairo_context_for_window_of_drawingarea (GtkDrawingArea *, cairo_t **);
gboolean button_motion (GtkDrawingArea  *, GdkEventMotion *, Display *);
gboolean button_press (GtkDrawingArea  *, GdkEventButton *, GdkRectangle *);
gboolean is_within_range_rectangle (ListItem *, mpqRectangle *);
void get_user_coordinates (gdouble *, gdouble *, mpq_t, mpq_t, mpqRectangle *, mpq_t);
void get_new_range_rectangle (mpqRectangle *, mpqRectangle *, GdkRectangle *);
void normalize_focus_rectangle (GdkRectangle *);
void gg_mpq_draw_as_mpf (cairo_t *, mpq_t, double, double);
void evaluate_input_definitions (InputDefinitions *, InputMatrices *, mpq_t);
void load_file_into_gui (const gchar *, FileData *, Display *, GtkEntry *);
void generate_x (mpq_t, FileData *, mpqRectangle *);
```

```
void evaluate_psi_with_error (mpq_t, InputMatrices *, mpq_t, long, pointerbundle *);
void evaluate_ubound_of_abs_psi (mpq_t, InputMatrices *, mpq_t);
void gg_mpq_init_clear (InitOrClear , mpqBundleType, gpointer);
ListItem* ListItem_with_gmp_variables_new (void);
void ListItem_with_gmp_variables_free (ListItem *, gpointer);
InputDefinitions* InputDefinitions_with_gmp_variables_new (void);
void InputDefinitions_with_gmp_variables_free (InputDefinitions *);
InputMatrices* InputMatrices_with_gmp_variables_new (void);
void InputMatrices_with_gmp_variables_free (InputMatrices *);
mpqRectangle* mpqRectangle_with_gmp_variables_new ();
void mpqRectangle_with_gmp_variables_free (mpqRectangle *);
FileData* FileData_new (void);
void FileData_free (FileData *);
mpqIntervall* mpqIntervall_with_gmp_variables_new (void);
void mpqIntervall_with_gmp_variables_free (mpqIntervall *);
void mpqIntervall_set_a_length (mpqIntervall *, mpq_t, mpq_t);
Display* Display_new (void);
void Display_free (Display *);
void gg_mpq_matrix_init (mpq_t (*) [3]);
void gg_mpq_matrix_clear (mpq_t (*) [3]);
void gg_mpq_matrix_add (mpq_t (*) [3], mpq_t (*) [3], mpq_t (*) [3]);
void gg_mpq_matrix_div_ui (mpq_t (*) [3], mpq_t (*) [3], unsigned long op2);
void gg_mpq_matrix_set (mpq_t (*) [3], mpq_t (*) [3]);
gboolean write_generated_matrices_to_file (InputMatrices *, const gchar *);
gboolean write_image_to_file (GtkWidget *, pointerbundle *);
double user_x (mpq_t, mpqRectangle *);
double user_y (mpq_t, mpqRectangle *);
void make_cross_for_listitem (cairo_t *, ListItem *, mpqRectangle *);
void make_cross_foreach (ListItem *, gpointer *);
gboolean compare_matrices (InputMatrices *);
void adopt_matrices_from_q (InputMatrices *);
void draw_on_surface (cairo_surface_t *, FileData *, mpqRectangle *);
```

```
GThread *thread = NULL;
GtkWindow *window;
GtkHBox *hbox;
GtkVBox *vbox_main, *vbox_1, *vbox_2;
GtkTable *table_1, *table_2;
GtkAdjustment *adjustment_matrices[8];
GtkAdjustment *adjustment_x;
GtkAdjustment *adjustment_input_precision;
GtkAdjustment *adjustment_result_accuracy;
GtkSpinButton *spinbutton_matrices[8];
GtkSpinButton *spinbutton_x;
GtkSpinButton *spinbutton_input_precision;
GtkSpinButton *spinbutton_result_accuracy;
GtkToolbar *toolbar;
GtkToolButton *toolbutton_openfile;
GtkToolButton *toolbutton_draw;
GtkToolButton *toolbutton_verify;
GtkSeparatorToolItem *separator;
GtkFileSelection *fileselection;
GtkButton *button_computate;
GtkDrawingArea *drawingarea;
GdkPixmap *pixmap;


double x_double;
unsigned long input_precision;

gulong num_of_add_ops;
gboolean stop = FALSE;
gboolean computation = FALSE;
```

```
unsigned long eval_kmax (mpq_t error, mpq_t a_omega3_p_2_t_a_lambda_p_a_xi, mpq_t beta, mpq_t a_omega3_p_2, mpq_t omega2)
{
  unsigned long ui;
  mpz_t auz1;
  mpq_t comp, auq1;

  mpz_init (auz1);
  mpq_init (comp); mpq_init (auq1);


  mpq_add (comp, a_omega3_p_2_t_a_lambda_p_a_xi, beta);

  gg_mpq_upper_bound_for_exp (comp, comp);

  /* 1+omega2(|omega|+2) */
  mpq_mul (auq1, a_omega3_p_2, omega2);
  gg_mpq_increment (auq1);

  mpq_div (comp, auq1, comp);

  mpq_mul (comp, comp, error);

  ui=0;
  mpq_set (auq1, beta);
  do
    {
      ui++;
      mpq_mul (auq1, auq1, beta);
      gg_mpq_div_ui (auq1, auq1, (unsigned long)((ui+1)*ui));
    }
  while(!(mpq_cmp (auq1, comp) <= 0));

  mpz_clear (auz1);
  mpq_clear (comp); mpq_clear (auq1);

  return ui;
}




void gg_mpq_matrix_set_A_minus_zB (mpq_t (*mat) [3], InputMatrices *im, mpq_t z)
{
  mpq_set (mat[0][0], im->a11);
  mpq_set (mat[0][1], im->a12);
  mpq_set (mat[0][2], im->a13);


  mpq_set (mat[1][0], im->a12);

  mpq_mul (mat[1][1], im->b2, z);
  mpq_neg (mat[1][1], mat[1][1]);
  mpq_add (mat[1][1], mat[1][1], im->a22);

  mpq_set (mat[1][2], im->a23);


  mpq_set (mat[2][0], im->a13);
  mpq_set (mat[2][1], im->a23);

  mpq_mul (mat[2][2], im->b3, z);
```

```
  mpq_neg (mat[2][2], mat[2][2]);
  mpq_add (mat[2][2], mat[2][2], im->a33);
}



void gg_mpq_matrix_set_unit (mpq_t (*mat) [3])
{
  short i, j;

  for (i=0; i<=2; i++)
    for (j=0; j<=2; j++)
      if (i==j) mpq_set_ui (mat[i][j], 1UL, 1UL);
      else mpq_set_ui (mat[i][j], 0UL, 1UL);
}



void gg_mpq_matrix_set (mpq_t (*rop) [3], mpq_t (*op) [3])
{
  short i, j;

  for (i=0; i<=2; i++)
    for (j=0; j<=2; j++)
      mpq_set (rop[i][j], op[i][j]);
}



void gg_mpq_matrix_add (mpq_t (*rop) [3], mpq_t (*op1) [3], mpq_t (*op2) [3])
{
  short i, j;


  for (i=0; i<=2; i++)
    for (j=0; j<=2; j++)
      mpq_add (rop[i][j], op1[i][j], op2[i][j]);
}



void gg_mpq_matrix_div_ui (mpq_t (*rop) [3], mpq_t (*op1) [3], unsigned long op2)
{
  short i, j;

  for (i=0; i<=2; i++)
    for (j=0; j<=2; j++)
      gg_mpq_div_ui (rop[i][j], op1[i][j], op2);
}



void gg_mpq_matrix_mul (mpq_t (*rop)[3], mpq_t (*op1)[3], mpq_t (*op2)[3]) /* equal arguments possible */
{
  short i, j, k;
  mpq_t auq;
  mpq_t mat[3][3];

  mpq_init (auq);
  gg_mpq_matrix_init (mat);

  for (i=0; i<=2; i++)
    for (j=0; j<=2; j++)
```

```
        {
          mpq_set_ui (mat[i][j], 0UL, 1UL);
          for (k=0; k<=2; k++)
            {
              mpq_mul (auq, op1[i][k], op2[k][j]);
              mpq_add (mat[i][j], mat[i][j], auq);
            }
        }

  gg_mpq_matrix_set (rop, mat);

  mpq_clear (auq);
  gg_mpq_matrix_clear (mat);
}



void gg_mpq_matrix_exp (mpq_t (*rop) [3], mpq_t (*exp) [3])
{
  unsigned long i;
  mpq_t auq1[3][3];
  mpq_t auq2[3][3];

  gg_mpq_matrix_init (auq1);
  gg_mpq_matrix_init (auq2);

  gg_mpq_matrix_set_unit (auq1);
  gg_mpq_matrix_set_unit (auq2);
  for (i=1; i<=MATEXP; i++)
    {
      gg_mpq_matrix_mul (auq1, auq1, exp);
      gg_mpq_matrix_div_ui (auq1, auq1, i);

      gg_mpq_matrix_add (auq2, auq2, auq1);
    }

  gg_mpq_matrix_set (rop, auq2);

  gg_mpq_matrix_clear (auq1);
  gg_mpq_matrix_clear (auq2);
}



void gg_mpq_matrix_trace (mpq_t rop, mpq_t (*mat) [3])
{
  mpq_add (rop, mat[0][0], mat[1][1]);
  mpq_add (rop, rop, mat[2][2]);
}



void gg_mpq_init_clear (InitOrClear init_clear, mpqBundleType type, gpointer bundle)
{
  short i;

  switch (init_clear)
    {
    case INIT:
      switch (type)
        {
        case PAIR:
          mpq_init (((mpqPair*) bundle)->a);
          mpq_init (((mpqPair*) bundle)->b);
          break;
```

```
      case TRIPLE:
        mpq_init (((mpqTriple*) bundle)->a);
        mpq_init (((mpqTriple*) bundle)->b);
        mpq_init (((mpqTriple*) bundle)->c);
        break;
      case MPQ_INTERVALL:
        mpq_init (((mpqIntervall*) bundle)->a);
        mpq_init (((mpqIntervall*) bundle)->b);
        mpq_init (((mpqIntervall*) bundle)->length);
        break;
      case MPQ_RECTANGLE:
        mpq_init (((mpqRectangle*) bundle)->x);
        mpq_init (((mpqRectangle*) bundle)->y);
        mpq_init (((mpqRectangle*) bundle)->width);
        mpq_init (((mpqRectangle*) bundle)->height);
        mpq_init (((mpqRectangle*) bundle)->auqx);
        mpq_init (((mpqRectangle*) bundle)->auqy);
        break;
      case QUINTUPLE:
        mpq_init (((mpqQuintuple*) bundle)->a);
        mpq_init (((mpqQuintuple*) bundle)->b);
        mpq_init (((mpqQuintuple*) bundle)->c);
        mpq_init (((mpqQuintuple*) bundle)->d);
        mpq_init (((mpqQuintuple*) bundle)->e);
        break;
      case INPUT_DEFINITIONS:
        mpq_init (((InputDefinitions*) bundle)->x2);
        mpq_init (((InputDefinitions*) bundle)->x3);
        mpq_init (((InputDefinitions*) bundle)->A12);
        mpq_init (((InputDefinitions*) bundle)->A13);
        mpq_init (((InputDefinitions*) bundle)->lambda);
        mpq_init (((InputDefinitions*) bundle)->mu);
        mpq_init (((InputDefinitions*) bundle)->upsilon);
        mpq_init (((InputDefinitions*) bundle)->xi);
        mpq_init (((InputDefinitions*) bundle)->omega1);
        mpq_init (((InputDefinitions*) bundle)->omega2);
        mpq_init (((InputDefinitions*) bundle)->omega);
        mpq_init (((InputDefinitions*) bundle)->omega3);
        mpq_init (((InputDefinitions*) bundle)->tomega);

        mpq_init (((InputDefinitions*) bundle)->a_omega3_p_2);
        mpq_init (((InputDefinitions*) bundle)->a_omega3_p_2_t_a_lambda);
        mpq_init (((InputDefinitions*) bundle)->a_omega3_p_2_t_a_lambda_p_a_xi);
        mpq_init (((InputDefinitions*) bundle)->beta);
        break;
      case INPUT_MATRICES:
        ((InputMatrices*) bundle)->p[0] = & ((InputMatrices*) bundle)->a11;
        ((InputMatrices*) bundle)->p[1] = & ((InputMatrices*) bundle)->a22;
        ((InputMatrices*) bundle)->p[2] = & ((InputMatrices*) bundle)->a33;
        ((InputMatrices*) bundle)->p[3] = & ((InputMatrices*) bundle)->a12;
        ((InputMatrices*) bundle)->p[4] = & ((InputMatrices*) bundle)->a13;
        ((InputMatrices*) bundle)->p[5] = & ((InputMatrices*) bundle)->a23;
        ((InputMatrices*) bundle)->p[6] = & ((InputMatrices*) bundle)->b2;
        ((InputMatrices*) bundle)->p[7] = & ((InputMatrices*) bundle)->b3;

        for (i=0; i<=7; i++) mpq_init (* ((InputMatrices*) bundle)->p[i]);
        break;
      case MPQ_MATRIX:
        break;
      }
    break;
  case CLEAR:
    switch (type)
      {
      case PAIR:
        mpq_clear (((mpqPair*) bundle)->a);
        mpq_clear (((mpqPair*) bundle)->b);
        break;
```

```
        case TRIPLE:
          mpq_clear (((mpqTriple*) bundle)->a);
          mpq_clear (((mpqTriple*) bundle)->b);
          mpq_clear (((mpqTriple*) bundle)->c);
          break;
        case MPQ_INTERVALL:
          mpq_clear (((mpqIntervall*) bundle)->a);
          mpq_clear (((mpqIntervall*) bundle)->b);
          mpq_clear (((mpqIntervall*) bundle)->length);
          break;
        case MPQ_RECTANGLE:
          mpq_clear (((mpqRectangle*) bundle)->x);
          mpq_clear (((mpqRectangle*) bundle)->y);
          mpq_clear (((mpqRectangle*) bundle)->width);
          mpq_clear (((mpqRectangle*) bundle)->height);
          mpq_clear (((mpqRectangle*) bundle)->auqx);
          mpq_clear (((mpqRectangle*) bundle)->auqy);
          break;
        case QUINTUPLE:
          mpq_clear (((mpqQuintuple*) bundle)->a);
          mpq_clear (((mpqQuintuple*) bundle)->b);
          mpq_clear (((mpqQuintuple*) bundle)->c);
          mpq_clear (((mpqQuintuple*) bundle)->d);
          mpq_clear (((mpqQuintuple*) bundle)->e);
          break;
        case INPUT_DEFINITIONS:
          mpq_clear (((InputDefinitions*) bundle)->x2);
          mpq_clear (((InputDefinitions*) bundle)->x3);
          mpq_clear (((InputDefinitions*) bundle)->A12);
          mpq_clear (((InputDefinitions*) bundle)->A13);
          mpq_clear (((InputDefinitions*) bundle)->lambda);
          mpq_clear (((InputDefinitions*) bundle)->mu);
          mpq_clear (((InputDefinitions*) bundle)->upsilon);
          mpq_clear (((InputDefinitions*) bundle)->xi);
          mpq_clear (((InputDefinitions*) bundle)->omega1);
          mpq_clear (((InputDefinitions*) bundle)->omega2);
          mpq_clear (((InputDefinitions*) bundle)->omega);
          mpq_clear (((InputDefinitions*) bundle)->omega3);
          mpq_clear (((InputDefinitions*) bundle)->tomega);

          mpq_clear (((InputDefinitions*) bundle)->a_omega3_p_2);
          mpq_clear (((InputDefinitions*) bundle)->a_omega3_p_2_t_a_lambda);
          mpq_clear (((InputDefinitions*) bundle)->a_omega3_p_2_t_a_lambda_p_a_xi);
          mpq_clear (((InputDefinitions*) bundle)->beta);
          break;
        case INPUT_MATRICES:
          for (i=0; i<=7; i++) mpq_init (* ((InputMatrices*) bundle)->p[i]);
          break;
        case MPQ_MATRIX:
          break;
        }
    }
}



ListItem* ListItem_with_gmp_variables_new (void)
{
  ListItem *item;

  item = g_new (ListItem, 1);
  mpq_init (item->x);
  mpq_init (item->fv);

  return item;
}
```

```
/* foreach function */
void ListItem_with_gmp_variables_free (ListItem *item, gpointer unused)
{
  mpq_clear (item->x);
  mpq_clear (item->fv);
  g_free (item);
}



InputDefinitions* InputDefinitions_with_gmp_variables_new (void)
{
  InputDefinitions *struc;

  struc = g_new (InputDefinitions, 1);
  gg_mpq_init_clear (INIT, INPUT_DEFINITIONS, struc);

  return struc;
}



void InputDefinitions_with_gmp_variables_free (InputDefinitions *struc)
{
  gg_mpq_init_clear (CLEAR, INPUT_DEFINITIONS, struc);
  g_free (struc);
}



InputMatrices* InputMatrices_with_gmp_variables_new (void)
{
  gint i;
  InputMatrices *struc;

  struc = g_new (InputMatrices, 1);
  gg_mpq_init_clear (INIT, INPUT_MATRICES, struc);

  for (i=0; i<=7; i++) mpf_init2 (((InputMatrices*) struc)->f[i], PRECISION);


  return struc;
}



void InputMatrices_with_gmp_variables_free (InputMatrices *struc)
{
  gint i;

  gg_mpq_init_clear (CLEAR, INPUT_MATRICES, struc);
  for (i=0; i<=7; i++) mpf_clear (((InputMatrices*) struc)->f[i]);

  g_free (struc);
}



mpqIntervall* mpqIntervall_with_gmp_variables_new (void)
{
```

```
  mpqIntervall *struc;

  struc = g_new (mpqIntervall, 1);
  gg_mpq_init_clear (INIT, MPQ_INTERVALL, struc);

  return struc;
}



void mpqIntervall_with_gmp_variables_free (mpqIntervall *struc)
{
  gg_mpq_init_clear (CLEAR, MPQ_INTERVALL, struc);
  g_free (struc);
}



void mpqIntervall_set_a_length (mpqIntervall *intervall, mpq_t a, mpq_t length)
{
  mpq_set (intervall->a, a);
  mpq_add (intervall->b, a, length);
  mpq_set (intervall->length, length);
}



void mpqIntervall_set_a_b (mpqIntervall *intervall, mpq_t a, mpq_t b)
{
  mpq_set (intervall->a, a);
  mpq_set (intervall->b, b);
  mpq_sub (intervall->length, b, a);
}



mpqRectangle* mpqRectangle_with_gmp_variables_new (void)
{
  mpqRectangle *struc;

  struc = g_new (mpqRectangle, 1);
  gg_mpq_init_clear (INIT, MPQ_RECTANGLE, struc);

  return struc;
}



void mpqRectangle_with_gmp_variables_free (mpqRectangle *struc)
{
  gg_mpq_init_clear (CLEAR, MPQ_RECTANGLE, struc);
  g_free (struc);
}



FileData* FileData_new (void)
{
  FileData *struc;

  struc = g_new (FileData, 1);
  struc->name = g_string_new ("");
```

```
  struc->matrices = InputMatrices_with_gmp_variables_new ();
  struc->list = NULL;

  return struc;
}



void FileData_free (FileData *struc)
{
  g_string_free (struc->name, TRUE);
  InputMatrices_with_gmp_variables_free (struc->matrices);
  g_free (struc);
}



Display* Display_new (void)
{
  Display *struc;

  struc = g_new (Display, 1);
  struc->mi = cairo_image_surface_create (CAIRO_FORMAT_ARGB32, WIDTH, HEIGHT);
  struc->cmi = cairo_create (struc->mi);
  struc->drawingarea = g_object_new (GTK_TYPE_DRAWING_AREA,
                                     NULL);
  gtk_widget_set_size_request (GTK_WIDGET (struc->drawingarea), WIDTH, HEIGHT);
  struc->range = mpqRectangle_with_gmp_variables_new ();
  struc->focus = g_new (GdkRectangle, 1);
  struc->pixmap = gdk_pixmap_new (NULL, WIDTH, HEIGHT, 24);
  struc->cp = gdk_cairo_create (GDK_DRAWABLE (struc->pixmap));

  struc->matrix = g_new (cairo_matrix_t, 1);
  cairo_matrix_init_identity (struc->matrix);
  cairo_matrix_translate (struc->matrix,  (double) LEFTMARGIN, (double) HEIGHT-1 - BOTTOMMARGIN);
  cairo_matrix_scale (struc->matrix, 1., -1.);

  mpq_init (struc->auq);

  return struc;
}



void Display_free (Display *struc)
{
  cairo_destroy (struc->cmi);
  cairo_surface_destroy (struc->mi);
/*    if (struc->drawingarea != NULL) g_printf ("null\n"); */
/*    gtk_object_sink (GTK_OBJECT (struc->drawingarea)); */
/*    g_object_unref (G_OBJECT (struc->drawingarea)); */
  cairo_destroy (struc->cd);
  cairo_destroy (struc->cp);
  g_object_unref (struc->pixmap);
  mpqRectangle_with_gmp_variables_free (struc->range);
  g_free (struc->focus);
  /* g_free (struc->matrix); /\* not yet tested! *\/ */
  mpq_clear (struc->auq);

  g_free (struc);
}



/* unused matrix operations */
```

```
void gg_mpq_matrix_change_rows (mpq_t (*mat)[3], short k, short l)
{
  short j;

  for (j=0; j<=2; j++)
    mpq_swap (mat[k-1][j], mat[l-1][j]);
}



void gg_mpq_matrix_change_colums (mpq_t (*mat)[3], short k, short l)
{
  short i;

  for (i=0; i<=2; i++)
    mpq_swap (mat[i][k-1], mat[i][l-1]);
}



void gg_mpq_matrix_change_both (mpq_t (*mat)[3], short k, short l)
{
  gg_mpq_matrix_change_rows (mat, k, l);
  gg_mpq_matrix_change_colums (mat, k, l);
}



void gg_mpq_matrix_multiply_row (mpq_t (*mat)[3], mpq_t factor, short k)
{
  short j;

  for (j=0; j<=2; j++)
    mpq_mul (mat[k-1][j], mat[k-1][j], factor);
}



void gg_mpq_matrix_multiply_colum (mpq_t (*mat)[3], mpq_t factor, short k)
{
  short i;

  for (i=0; i<=2; i++)
    mpq_mul (mat[i][k-1], mat[i][k-1], factor);
}



void gg_mpq_matrix_multiply_both (mpq_t (*mat)[3], mpq_t factor, short k)
{
  gg_mpq_matrix_multiply_row (mat, factor, k);
  gg_mpq_matrix_multiply_colum (mat, factor, k);
}



void gg_mpq_matrix_add_multiplied_row (mpq_t (*mat)[3], mpq_t factor, short k, short l)
{
  short j;
  mpq_t auq;
```

```
  mpq_init (auq);

  for (j=0; j<=2; j++)
    {
      mpq_mul (auq, mat[k-1][j], factor);
      mpq_add (mat[l-1][j], mat[l-1][j], auq);
    }

  mpq_clear (auq);
}



void gg_mpq_matrix_add_multiplied_colum (mpq_t (*mat)[3], mpq_t factor, short k, short l)
{
  short i;
  mpq_t auq;

  mpq_init (auq);

  /* Hallo Meysi */
  for (i=0; i<=2; i++)
    {
      mpq_mul (auq, mat[i][k-1], factor);
      mpq_add (mat[i][l-1], mat[i][l-1], auq);
    }

  mpq_clear (auq);
}



void gg_mpq_matrix_add_multiplied_both (mpq_t (*mat)[3], mpq_t factor, short k, short l)
{
  gg_mpq_matrix_add_multiplied_row (mat, factor, k, l);
  gg_mpq_matrix_add_multiplied_colum (mat, factor, k, l);
}



void gg_mpq_matrix_diagonalize (mpq_t (*mat)[3], mpq_t (*erow)[3], mpq_t (*ecolum)[3])
{
  mpq_t auq;

  mpq_init (auq);

  mpq_div (auq, mat[0][1], mat[0][0]);
  mpq_neg (auq, auq);
  gg_mpq_matrix_add_multiplied_row (mat, auq, 1, 2);
  gg_mpq_matrix_add_multiplied_row (erow, auq, 1, 2);
  gg_mpq_matrix_add_multiplied_colum (mat, auq, 1, 2);
  gg_mpq_matrix_add_multiplied_colum (ecolum, auq, 1, 2);

  mpq_div (auq, mat[0][2], mat[0][0]);
  mpq_neg (auq, auq);
  gg_mpq_matrix_add_multiplied_row (mat, auq, 1, 3);
  gg_mpq_matrix_add_multiplied_row (erow, auq, 1, 3);
  gg_mpq_matrix_add_multiplied_colum (mat, auq, 1, 3);
  gg_mpq_matrix_add_multiplied_colum (ecolum, auq, 1, 3);

  mpq_div (auq, mat[1][2], mat[1][1]);
  mpq_neg (auq, auq);
  gg_mpq_matrix_add_multiplied_row (mat, auq, 2, 3);
  gg_mpq_matrix_add_multiplied_row (erow, auq, 2, 3);
  gg_mpq_matrix_add_multiplied_colum (mat, auq, 2, 3);
```

```
  gg_mpq_matrix_add_multiplied_colum (ecolum, auq, 2, 3);

  mpq_clear (auq);
}
```

B.3 gg-gmp-cairo.h

```
void gg_mpq_increment (mpq_t rop)
{
  mpz_add (mpq_numref (rop), mpq_numref (rop), mpq_denref (rop));
  mpq_canonicalize (rop);
}
```

```
void gg_mpq_decrement (mpq_t rop)
{
  mpz_sub (mpq_numref (rop), mpq_numref (rop), mpq_denref (rop));
  mpq_canonicalize (rop);
}
```

```
void gg_mpq_add_ui (mpq_t rop, mpq_t op1, unsigned long op2) /* equal arguments possible */
{
  mpq_set (rop, op1);
  mpz_addmul_ui (mpq_numref (rop), mpq_denref (rop), op2);
  mpq_canonicalize (rop);
}
```

```
void gg_mpq_mul_ui (mpq_t rop, mpq_t op1, unsigned long op2) /* equal arguments possible */
{
  mpq_set (rop, op1);
  mpz_mul_ui (mpq_numref(rop), mpq_numref(rop), op2);
  mpq_canonicalize(rop);
}
```

```
void gg_mpq_div_ui (mpq_t rop, mpq_t op1, unsigned long op2) /* equal arguments possible */
{
  mpq_set (rop, op1);
  mpz_mul_ui (mpq_denref(rop), mpq_denref(rop), op2);
  mpq_canonicalize(rop);
}
```

```
void gg_mpq_mul_mpz (mpq_t rop, mpq_t op1, mpz_t op2)
{
  mpq_set (rop, op1);
  mpz_mul (mpq_numref (rop), mpq_numref (op1), op2);
  mpq_canonicalize (rop);
}
```

```
void gg_mpq_div_mpz (mpq_t rop, mpq_t op1, mpz_t op2)
{
  mpq_set (rop, op1);
  mpz_mul (mpq_denref (rop), mpq_denref (op1), op2);
  mpq_canonicalize (rop);
}



void gg_mpq_max (mpq_t rop, mpq_t op1, mpq_t op2)  /* equal arguments possible */
{
  if (mpq_cmp (op1, op2) > 0) mpq_set (rop, op1);
  else mpq_set (rop, op2);
}



void gg_mpq_min (mpq_t rop, mpq_t op1, mpq_t op2)  /* equal arguments possible */
{
  if (mpq_cmp (op1, op2) < 0) mpq_set (rop, op1);
  else mpq_set (rop, op2);
}



void gg_mpq_pow_ui(mpq_t rop, mpq_t op, unsigned long e) /* equal arguments possible */
{
  mpz_pow_ui (mpq_numref (rop), mpq_numref (op), e);
  mpz_pow_ui (mpq_denref (rop), mpq_denref (op), e);
  mpq_canonicalize(rop);
}



void gg_mpq_abs_pow_ui(mpq_t rop, mpq_t op, unsigned long e) /* equal arguments possible */
{
  mpq_abs (rop, op);
  mpz_pow_ui (mpq_numref (rop), mpq_numref (rop), e);
  mpz_pow_ui (mpq_denref (rop), mpq_denref (rop), e);
  mpq_canonicalize(rop);
}



/* rising factorial (x)_n */
void gg_mpz_rf (mpz_t rop, unsigned long x, unsigned long n)
{
  unsigned long i;
  mpz_set_ui(rop, 1UL);
  for (i=0; i<n; i++) mpz_mul_ui(rop, rop, x+i);
}



/* [x]_n */
void gg_mpz_rf2 (mpz_t rop, unsigned long x, unsigned long n)
{
  unsigned long i;
  mpz_set_ui(rop, 1UL);
  for(i=0; i<n; i++) mpz_mul_ui(rop, rop, x+2*i);
}
```

```
void gg_mpq_2exp_si (mpq_t rop, long n)
{
  mpq_t auq1;
  mpq_init (auq1);

  mpq_set_ui (auq1, 1UL, 1UL);
  mpq_mul_2exp (rop, auq1, (unsigned long) ABS(n));
  if (n<0) mpq_inv(rop, rop);

  mpq_clear (auq1);
}



#define PRINT_DIGITS 5
/* very unefficient but convenient function */
void gg_mpz_print_as_mpf (mpz_t op)
{
  mpf_t auf;

  mpf_init2 (auf, PRINT_DIGITS*4);

  mpf_set_z (auf, op);
  mpf_out_str (stdout, 10, PRINT_DIGITS, auf);
  g_print ("\n");

  mpf_clear (auf);
}



/* very unefficient but convenient function */
void gg_mpq_print_as_mpf (mpq_t op)
{
  mpf_t auf;

  mpf_init2 (auf, PRINT_DIGITS*4);

  mpf_set_q (auf, op);
  mpf_out_str (stdout, 10, PRINT_DIGITS, auf);
  g_print ("\n");

  mpf_clear (auf);
}



void gg_mpq_matrix_print_as_mpf (mpq_t (*mat) [3])
{
  short i, j;

  for (i=0; i<=2; i++)
    for (j=0; j<=2; j++)
      gg_mpq_print_as_mpf (mat[i][j]);
}



/* draws a number given as an mpq-Type on the drawing surface connected with the drawing context cr at */
/* the given coordinates, the input coordinates must be in original gtk form (x right, y down) */
void gg_mpq_draw_as_mpf (cairo_t *cr, mpq_t q, double x, double y)
```

```
{
  GString *gstring;
  mpf_t auf;

  gstring = g_string_new ("");
  mpf_init2 (auf, PRINT_DIGITS*4);


  mpf_set_q (auf, q);
  gmp_asprintf (& gstring->str, "%.2Fe", auf);

  cairo_move_to (cr, x, y);
  cairo_show_text (cr, gstring->str);


  g_string_free (gstring, TRUE);
  mpf_clear (auf);
}



/* provides an upper bound for e^q */
gboolean gg_mpq_upper_bound_for_exp (mpq_t rop, mpq_t q)  /* equal arguments possible */
{
  unsigned long rounded_exponent;
  mpz_t auz1;
  mpq_t auq1;

  mpz_init (auz1);
  mpq_init (auq1);

  mpq_set_ui (auq1, 0UL, 1UL);
  if (mpq_cmp (q, auq1) >= 0)
    {
      mpz_cdiv_q (auz1, mpq_numref (q), mpq_denref (q));  // last use of q
      if (mpz_fits_ulong_p (auz1)) rounded_exponent = mpz_get_ui (auz1);
      else
        {
          g_printf ("Error: Exponent to high\n");
          mpz_clear (auz1);
          mpq_clear (auq1);
          return FALSE;
        }
      mpq_set_ui(auq1, 272UL, 100UL);
      mpq_canonicalize (auq1);
      gg_mpq_pow_ui (rop, auq1, rounded_exponent);  // first assignment to rop
    }
  else
    {
      mpq_abs (q, q);
      mpz_fdiv_q (auz1, mpq_numref (q), mpq_denref (q));  // last use of q
      if (mpz_fits_ulong_p (auz1)) rounded_exponent = mpz_get_ui (auz1);
      else
        {
          g_printf ("Error: Exponent to high\n");
          mpz_clear (auz1);
          mpq_clear (auq1);
          return FALSE;
        }
      mpq_set_ui(auq1, 271UL, 100UL);
      mpq_canonicalize (auq1);
      gg_mpq_pow_ui (rop, auq1, rounded_exponent);  // first assignment to rop
      mpq_inv (rop, rop);
    }


  mpz_clear (auz1);
```

```
  mpq_clear (auq1);

  return TRUE;
}



/* calculates e^q with an absolute error less equal than err */
void gg_mpq_exp (mpq_t rop, mpq_t q, mpq_t err) /* equal arguments possible */
{
  unsigned long i, imax;
  mpq_t auq1, auq2;

  mpq_init (auq1); mpq_init (auq2);


  mpq_set_ui (auq2, 0UL, 1UL);
  if (mpq_cmp (q, auq2)<0) mpq_set_ui (auq1, 1UL, 1UL);
  else gg_mpq_upper_bound_for_exp (auq1, q);

  mpq_abs (auq2, q);
  mpq_mul (auq1, auq1, auq2);
  imax=0;
  do
    {
      imax++;
      mpq_mul (auq1, auq1, auq2);
      gg_mpq_div_ui (auq1, auq1, (unsigned long) (imax+1));
    }
  while (mpq_cmp (auq1, err) > 0);


  mpq_set_ui(auq1, 1UL, 1UL);
  mpq_set_ui(auq2, 1UL, 1UL);
  for (i=1; i<=imax; i++)
    {
      mpq_mul(auq1, auq1, q);
      gg_mpq_div_ui (auq1, auq1, i);

      mpq_add (auq2, auq2, auq1);
    }

  mpq_set (rop, auq2); // first assignment to rop


  mpq_clear (auq1); mpq_clear (auq2);
}



#define FACTOR 1
#define CROSS_SIZE 4.*FACTOR //cross looks ugly whithout the point, or cast to double
void gg_cairo_cross (cairo_t *cr, double x, double y)
{
  cairo_move_to (cr, x - CROSS_SIZE/2, y - CROSS_SIZE/2);
  cairo_rel_line_to (cr, CROSS_SIZE, CROSS_SIZE);
  cairo_move_to (cr, x + CROSS_SIZE/2, y - CROSS_SIZE/2);
  cairo_rel_line_to (cr, - CROSS_SIZE, CROSS_SIZE);
}



void gg_mpq_matrix_init (mpq_t (*mat) [3])
{
  short i, j;
  for (i=0; i<=2; i++)
```

```
    for (j=0; j<=2; j++)
      mpq_init (mat[i][j]);
}



void gg_mpq_matrix_clear (mpq_t (*mat) [3])
{
  short i, j;
  for (i=0; i<=2; i++)
    for (j=0; j<=2; j++)
      mpq_clear (mat[i][j]);
}
```

References

[DST] Michael Drmota, Walter Schachermayer, Josef Teichmann: A hyper-geometric approach to the BMV-conjecture, Monatshefte für Mathematik, 146, 179-201, 2005.