MŰEGYETEM 1782

BuTools

Program packages for computations with PH, ME distributions and MAP, RAP processes

Levente Bodrog,
Peter Buchholz,
Armin Heindl,
András Horváth,
Gábor Horváth,
István Kolossváry,
András Mészáros,
Zoltán Németh,
János Papp
Philipp Reinecke,
Miklós Telek,
Miklós Vécsei

Sept. 30, 2014

Contents

Introduction

BuTools is a collection of Mathematica, Matlab/Octave functions related to recent research results on the field of phase type (PH) and matrix exponential (ME) distributions and Markov arrival processes (MAPs) and rational arrival processes (RAPs).

This document lists the elements and the use of the currently available (see the date on the cover page) BuTools functions. One of the main goals of this document is to relate the functions and the papers where the related algorithms are published. The readers are referred to the original publications (available links are provided at the list of references) for detailed descriptions of the procedures.

1 Usage of the BuTools functions

1.1 The test files

The BuTools package includes a set of test files which demonstrate the usage of the BuTools functions. There is an associated test function for all main parts (utilities, PH, MAP, special processes, fluid). The name of the test files are:

test_utils_functions.nb, test_ph_functions.nb and test_map_functions.nb in the Mathematica package and

test_utils_functions.m, test_ph_functions.m and test_map_functions.m in the Matlab/Octave package.

To obtain a quick impression about the input, the output and the behavior of the available functions run the related test file. Doing so one gets the description of each function with some related examples on the output.

1.2 Matlab/Octave

In Matlab/Octave you have to specify the path of the BuTools subpackages.
For example, in Windows base operation systems if the Utilities package is in the *C:\Work\Butools\Utilities* directory the path has to be specified by the **path(path, 'C:\Work\Butools\Utilities')** command in the related test file.

If the package is located in the home/Work/Butools/Utilities directory then the syntax of the command is **path(path, '/home/[your username]/Work/Butools/Utilities')** in Unix/Linux based operation systems.

1.3 Mathematica

In order to use the BuTools packages in Mathematica one has to locate and load them.

The first step is to specify the directory of the BuTools Mathematica packages (*.m files). It can be done with assigning the name of the directory to a variable (e.g., "dir"). The syntax is different in Windows and in Linux systems.

In Windows systems if the BuTools packages (*.m files) are on the *C:* drive in the *Work\BuTools* directory then the command to locate the packages is:
dir = "C:\\Work\\BuTools"

In Linux systems if the BuTools packages (*.m files) are in the *home/Work/BuTools* directory then the command to locate the packages is:
dir = "/home/[your username]/Work/BuTools"

After the path is specified in variable "dir" correctly, type the following command:
AppendTo[$Path, dir]

Now the path of Mathematica is extended with the directory containing the BuTools packages.

To use a package it has to be loaded by typing <<*[package name]'*
For example to load the Utilities package type:
<<"Utilities'"

After these steps you can use all the avaliable functions of the Utilities package.

A complete Mathematica notebook file which locates and loads the package looks like this:

dir = "[the directory]"
AppendTo[$Path, Dir]
<<"[package name]'"

The test files also contain the commands for locating and loading the packages.

1.4 Error handling

In most of the functions to use the methods we have assumptions about the input (e.g: the input of MatginalMomentsFromMRAP is a real MRAP). We check the inputs to ensure these assumptions, and if these checks fail then an error is occured. To handle the errors we throw an exception in Mathematica and use the built in *error* function in Matlab.

If you simply call a function in Mathematica and an exception is thrown, you get the exception twice. Once in a built in Mathematica warning (unhandled exception) and once as the output. You can avoid it if you call your function inside the built in *Catch* function, e.g Catch[YourFunction[...]]. In this case you don't get the unhandled exception warning and the output is a little bit nicer.

In Matlab you don't have to do anything when you call only one function. If an error occurs then you get it as the output. Although if you're running a script and an error occurs, then your script will stop running. To avoid this, use the try-catch statement and handle the error. You can see examples in the test files and can get more information about try-catch in the Matlab help.

2 Available functions

The program package is divided into the following 4 main parts:

- BuToolsUtilities
- BuToolsPH
- BuToolsMAP
- BuToolsSpecialProcesses
- BuToolsFluid

Some functions has optional parameters. We give their default value of these parameters in []. You can give a numerical precision ε (which is 10^{-14} by default) to almost every function. If a check or a function fail, it could be a numerical inaccuracy. In these cases try to give a bit larger ε to the functions.

2.1 BuToolsUtilities

The BuToolsUtilities package contains the following functions

BuToolsVerbose
A flag (global variable) to switch between verbose and silent modes.

CRPSolve
Gives the steady state distribution of the continuous time rational process (CRP). It is the same as CTMCSolve but without checking if the input matrix is a proper generator matrix.

Input: matrix

Output: vector

DRPSolve
Gives the steady state distribution of a discrete time rational process (DRP). It is the same as DTMCSolve but without checking if the input matrix is a proper stochastic matrix.

Input: matrix

Output: vector

CTMCSolve
Gives the steady state distribution (π) of the CTMC with generator matrix $\boldsymbol{Q}$. I.e., the solution of the linear system $\pi\boldsymbol{Q} = 0, \pi\mathbb{1} = 1$.

Input: matrix, $\varepsilon[10^{-14}]$

Output: vector

DTMCSolve

Gives the steady state distribution (π) of the DTMC with transition probability matrix $\boldsymbol{P}$. I.e., the solution of the linear system $\pi \boldsymbol{P} = \pi, \pi \mathbb{1} = 1$.

Input: matrix, $\varepsilon[10^{-14}]$

Output: vector

CheckGenerator

Checks if the matrix is a valid generator matrix. I.e., the matrix is a square matrix, the matrix has non-negative off-diagonal elements, the diagonal of the matrix is negative, the row sum of the matrix is 0.
If the transient flag is True it checks if the matrix is a valid transient generator matrix. I.e., the matrix is a square matrix, the diagonal of the matrix is negative, the matrix has non-negative off-diagonal elements, the real part of the maximum absolute eigenvalue is less than zero.

Input: matrix, transient flag [False], $\varepsilon[10^{-14}]$

Output: flag

CheckProbMatrix

Checks if the matrix is a valid probability matrix. I.e., the matrix is a square matrix, the matrix has positive or zero elements, the row sum of the matrix is 1.
If the transient flag is True it checks if the matrix is a valid transient probability matrix. I.e., the matrix is a square matrix, the matrix has positive or zero elements, the row sum of the matrix is less equal than 1, the maximum of the absolute values of the eigenvalues is less than 1.

Input: matrix, transient flag [False], $\varepsilon[10^{-14}]$

Output: flag

CheckProbVector

Checks if the vector is a valid probability vector.
I.e., the vector has only non-negative elements, the sum of the vector elements is 1. If the sub flag is True it checks if the vector is a valid sub-probability vector.
I.e., the vector has only non-negative elements, the sum of the vector elements is less equal than 1.

Input: vector, sub flag [False], $\varepsilon[10^{-14}]$

Output: flag

CheckMERepresentation

Checks some matrix exponential conditions of a vector-matrix pair. I.e., the matrix is a square matrix, the vector and the matrix have the same size, the dominant eigenvalue (one with maximal real part) of the matrix is negative and real.

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: flag

CheckMGRepresentation

Checks some matrix geometric conditions of a vector-matrix pair. I.e., the matrix is a square matrix, the vector and the matrix have the same size, the dominant eigenvalue (one with maximal absolute value) of the matrix is real, positive and less than 1.

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: flag

CheckPHRepresentation

Checks the phase type conditions on a vector - matrix pair. I.e, the vector is a probability vector, the matrix is a transient generator and they have the same size.

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: flag

CheckDPHRepresentation

Checks the discrete phase type conditions on a vector - matrix pair. I.e, the vector is a probability vector, the matrix is a transient probability matrix and they have the same size.

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: flag

CheckRAPRepresentation

Checks the rational arrival process conditions on 2 matrices.
I.e., matrix0, matrix1 are square matrices, they have the same size, the dominant eigenvalue of matrix0 is negative and real.The rowsums of matrix0+matrix1 are 0.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: flag

CheckDRAPRepresentation

Checks the discrete rational arrival process conditions on 2 matrices.
I.e., matrix0, matrix1 are square matrices, they have the same size, the dominant eigenvalue of matrix0 is real and less than 1. The rowsums of matrix0+matrix1 are 1.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: flag

CheckMRAPRepresentation

Checks the marked rational arrival process (MRAP: RAP with arrivals of different types) conditions on input matrices.
I.e. matrix0 and $\sum_{i=1}^{M} matrix_i$ is a RAP.

Input: vector of matrix0, matrix1, ... matrixM, $\varepsilon[10^{-14}]$

Output: flag

CheckDMRAPRepresentation

Checks the discrete marked rational arrival process (DMRAP: DRAP with arrivals of different types) conditions on input matrices.
I.e. matrix0 and $\sum_{i=1}^{M} matrix_i$ is a DRAP.

Input: vector of matrix0, matrix1, ... matrixM, $\varepsilon[10^{-14}]$

Output: flag

CheckMAPRepresentation

Checks if the input matrixes define a continuous time MAP. I.e., matrix0 and matrix1 are squrare matrixes of identical size, matrix0 is a transient generator matrix, matrix1 has only non-negative elements, and the rowsums of matrix0+matrix1 are 0.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: flag

CheckDMAPRepresentation

Checks if the input matrixes define a discrete time MAP. I.e., matrix0 and matrix1 are squrare matrixes of identical size, matrix0 is a transient generator matrix, the matrices have only non-negative elements, and the rowsums of matrix0+matrix1 are 1.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: flag

CheckMMAPRepresentation

Checks the marked markovian arrival process (MMAP: MAP with arrivals of different types) conditions on input matrices. I.e. matrix0, $\sum_{i=1}^{k} matrix_i$ is a MAP and $matrix_i$ has only non-negative elements.

Input: vector of matrix0, matrix1, ... matrixK, $\varepsilon[10^{-14}]$

Output: flag

CheckDMMAPRepresentation

Checks the discrete marked markovian arrival process (DMMAP: DMAP with arrivals of different types) conditions on input matrices. I.e. matrix0, $\sum_{i=1}^{M} matrix_i$ is a DMAP and $matrix_i$ has only non-negative elements.

Input: vector of matrix0, matrix1, ... matrixM, $\varepsilon[10^{-14}]$

Output: flag

NormmomsFromMoms

Computes normalized moments (m_i) from moments (μ_i): $m_i = \frac{\mu_i}{\mu_{i-1}\mu_1}$

Input: moments

Output: moments

MomsFromNormmoms

Computes the moments (μ_i) from normalized moments (m_i) based on $\mu_i = m_i \mu_{i-1} \mu_1$ assuming $\mu_1 = 1$.

Input: moments

Output: moments

ReducedmomsFromMoms

Computes reduced moments (r_i) from moments (μ_i): $r_i = \mu_i / i!$

Input: moments

Output: moments

MomsFromReducedmoms

Computes moments (μ_i) from reduced moments (r_i): $\mu_i = r_i i!$

Input: moments

Output: moments

FactorialmomsFromMoms

Computes factorial moments (f_i) from moments (μ_i), where $\mu_i = E(X^i)$ and $f_i = E(X(X-1)(X-2)\ldots(X-i+1))$

Input: moments

Output: moments

MomsFromFactorialmoms

Computes moments (μ_i) from factorial moments (f_i), where $\mu_i = E(X^i)$ and $f_i = E(X(X-1)(X-2)\ldots(X-i+1))$

Input: moments

Output: moments

JFactorialmomsFromJMoms

Computes factorial joint moments (f_{ij}) from joint moments (μ_{ij}), where $\mu_{ij} = E(X^iY^j)$ and $f_{ij} = E(X(X-1)(X-2)\ldots(X-i+1)Y(Y-1)(Y-2)\ldots(Y-j+1))$.

Input: moments

Output: moments

JMomsFromJFactorialmoms

Computes joint moments (μ_{ij}) from factorial joint moments (f_{ij}), where $\mu_{ij} = E(X^iY^j)$ and $f_{ij} = E(X(X-1)(X-2)\ldots(X-i+1)Y(Y-1)(Y-2)\ldots(Y-j+1))$.

Input: moments

Output: moments

KroneckerProduct

Gives the Kronecker product of the two matrices

Input: matrix0, matrix1

Output: matrix

KroneckerSum

Gives the Kronecker sum of the two matrices

Input: matrix0, matrix1

Output: matrix

2.2 BuToolsPH

The BuToolsPH package contains the following functions

RandomPH

Generates a random PH of the given order, that contains *zeroEntries* zeros. The mean of the generated PH can be set. The function stops after *maxTrials* failed try. A try can fail, if a row contain too many zeros, or the results order is less than the given one.

Input: order, zeroEntries, mean [1], $\varepsilon[10^{-14}]$, maxTrials [25000]

Output: vector-matrix pair

RandomDPH

Generates a random DPH of the given order, that contains *zeroEntries* zeros. The function stops after *maxTrials* failed try. A try can fail, if a row contain too many zeros, or the result's order is less than the given one.

Input: order, zeroEntries, $\varepsilon[10^{-14}]$, maxTrials [25000]

Output: vector-matrix pair

PHFromME

Converts a non-Markovian representation (vector-matrix pair) to Markovian representation of a phase type distribution if possible using the procedure from [20].

Input: vector-matrix pair

Output: vector-matrix pair

MomentsFromME

Calculates the first k moments of a ME given with a vector-matrix pair $(\alpha, \boldsymbol{A})$: $i!\alpha(-\boldsymbol{A})^{-i}\mathbb{1}$ $(i = 1, 2, \ldots, k)$ [14]. It fails if the input isn't a valid ME representation.

Input: vector, matrix, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MomentsFromMG

Calculates the first k moments of a MG given with a vector-matrix pair $(\alpha, \boldsymbol{A})$: where the factorial moments are $f_i = i!\alpha(\boldsymbol{I} - \boldsymbol{A})^{-i}\boldsymbol{A}^{i-1}\mathbb{1}$ $(i = 1, 2, \ldots, k)$ and it transforms the factorial moments to ordinary moments. It fails if the input isn't a valid MG representation.

Input: vector, matrix, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MomentsFromPH

Checks is the input is a PH and calculates the first k moments of a PH given with a vector-matrix pair $(\alpha, \boldsymbol{A})$: $i!\alpha(-\boldsymbol{A})^{-i}\mathbb{1}$ $(i = 1, 2, \ldots, k)$ [14]. It fails if the input isn't a valid PH representation.

Input: vector, matrix, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MomentsFromDPH

Checks is the input is a DPH and calculates the first k moments of a DPH given with a vector-matrix pair $(\alpha, \boldsymbol{A})$: where the factorial moments are $f_i = i!\alpha(\boldsymbol{I} - \boldsymbol{A})^{-i}\boldsymbol{A}^{i-1}\mathbb{1}$

$(i = 1, 2, \ldots, k)$ and it transforms the factorial moments to ordinary moments. It fails if the input isn't a valid DPH representation.

Input: vector, matrix, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MEFromMoments

Based on a set of moments μ_i $(i = 1, 2, \ldots, k)$ it calculates a vector-matrix pair $(\alpha, \boldsymbol{A})$ such that $\mu_i = i!\alpha(-\boldsymbol{A})^{-i}\mathbb{1}$ $(i = 1, 2, \ldots, k)$ [20, 21]. Check the size of the obtained representation because the procedure might stop before fitting all moments.

Input: moments

Output: vector, matrix

MGFromMoments

Based on a set of moments μ_i $(i = 1, 2, \ldots, k)$ it calculates a vector-matrix pair $(\alpha, \boldsymbol{A})$ such that μ_i $(i = 1, 2, \ldots, k)$ are the moments of the returned MG distribution.

Input: moments

Output: vector, matrix

PH2Canonical

Calculates the order 2 canonical representation from any order 2 vector-matrix representation, if exists based on [19]. It fails if the input isn't a valid ME representation.

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

DPH2Canonical

Calculates the order 2 canonical representation from any order 2 vector-matrix representation, if exists based on [17]. It fails if the input isn't a valid MG representation.

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

PH3Canonical

Calculates the order 3 canonical representation from any order 3 vector-matrix representation, if exists based on [11]. It gives a warning, if the input isn't a valid PH representation, and fails if the input isn't a valid ME representation.

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

DPH3Canonical

Calculates the order 3 canonical representation from any order 3 vector-matrix representation, if exists based on [17]. It fails if the input isn't a valid DPH representation.

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

APHRepresentation

Calculates the APH (CF1) representation from any order n vector-matrix representation, if exists. The procedure is similar to the one in [8], but after computing the eigenvalues it computes a similarity matrix by solving a system of linear equations. It fails if the input isn't a valid ME representation.

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

ADPHRepresentation

Calculates the ADPH representation from any order n vector-matrix representation, if exists. The procedure is similar to the one in [8], but after computing the eigenvalues it computes a similarity matrix by solving a system of linear equations. It fails if the input isn't a valid MG representation.

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

MonocyclicRepresentation

Calculates the representation of the input ME distribution with Markovian monocyclic generator defined in [16].

Input: vector-matrix pair, exit prob [0]

Output: vector-matrix pair

RepTrafo

Finds the transformation matrix from matrix1 to matrix2, if it exists, and then it applies that transformation to the input vector, such that {vector1, matrix1} and {vector2, matrix2} are two representations of the same ME distributions with potentially different sizes. The sizes of the distributions are determined by the size of matrix1 and matrix2.

Input: vector1, matrix1, matrix2

Output: vector2

APHFrom3Moments

Calculates the smallest APH with the given first 3 moments based on [2].

Input: mom1, mom2, mom3

Output: vector-matrix pair

MEOrderFromMoments

Calculates the order of the ME distribution based on its moments using the determinant of the Hankel matrix [4].

Input: moments, $\varepsilon[10^{-14}]$

Output: order

ME3member

Checks if the vector-matrix pair of size 3 defines an ME(3) distribution [10, 13]

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: flag

MEContOrder

Controllability (closing vector) order of the vector-matrix pair [5]. Assuming $\boldsymbol{A}$=matrix and $\mathbb{1}$ is the column vector of ones the controllability order is Rank $\begin{pmatrix} \mathbb{1} & \boldsymbol{A}\mathbb{1} & \boldsymbol{A}^2\mathbb{1} & \cdots \end{pmatrix}$

Input: vector-matrix pair

Output: order

MEObsOrder

Observability (initial vector) order of the vector-matrix pair [5]. Assuming α=vector and $\boldsymbol{A}$=matrix the observability order is Rank $\begin{pmatrix} \alpha \\ \alpha\boldsymbol{A} \\ \alpha\boldsymbol{A}^2 \\ \vdots \end{pmatrix}$

Input: vector-matrix pair

Output: order

CheckMEPositiveDensity

Checks if the vector-matrix pair results in a positive density

Input: vector-matrix pair, $\varepsilon[10^{-14}]$

Output: flag

MEDensity

Gives back the value of the density function of a vector-matrix pair $(\alpha, \boldsymbol{A})$ at point x: $f(x) = -\alpha e^{\boldsymbol{A}x}\boldsymbol{A}\mathbb{1}$.

Input: vector, matrix, x

Output: density value

2.3 BuToolsMAP

The BuToolsMAP package contains the following functions

RandomMAP

Generates a random MAP of the given order. You could specify the number of zero entries in the representation. If it can't generate a MAP after *maxTrials* trials, then the function fails. The obtained MAP's first moment is *mean*.

Input: order, zeroEntries [0], mean [1], $\varepsilon[10^{-14}]$, maxTrials [100]

Output: matrix0, matrix1

RandomDMAP

Generates a random DMAP of the given order. You could specify the number of zero entries in the representation. If it can't generate a DMAP after *maxTrials* trials, then the function fails.

Input: order, zeroEntries [0], $\varepsilon[10^{-14}]$, maxTrials [100]

Output: matrix0, matrix1

RandomMMAP

Generates a random MMAP of the given order and the given number of types. You could specify the number of zero entries in the representation. If it can't generate a MMAP after *maxTrials* trials, then the function fails. The obtained MMAP's first moment is *mean*.

Input: order, types, zeroEntries [0], mean [1], $\varepsilon[10^{-14}]$, maxTrials [100]

Output: vector of matrix0, matrix1, ... matrixM

RandomDMMAP

Generates a random DMMAP of the given order and the given number of types. You could specify the number of zero entries in the representation. If it can't generate a MMAP after *maxTrials* trials, then the function fails.

Input: order, types, zeroEntries [0], $\varepsilon[10^{-14}]$, maxTrials [100]

Output: vector of matrix0, matrix1, ... matrixM

MAPFromRAP

Similarity transforms the (matrix0, matrix1) non-Markovian representation of a RAP to the Markovian representation (outputmx0, outputmx1), if possible using the method from [20]. It fails if the input isn't a valid RAP representation.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: outputmx0, outputmx1

MMAPFromMRAP

Similarity transforms the non-Markovian representation of a marked RAP (matrix0 $(n \times n)$, ..., matrixM $(n \times n)$) to the Markovian representation (outputmx0, ..., outputmxM) of the same size, if possible using the method from [20].

Input: vector of matrix0, ..., matrixM

Output: vector of outputmx0, ..., outputmxM

MarginalDistributionFromRAP

Computes the matrix exponential representation of the marginal distribution of a rational arrival process. It fails if the input isn't a valid RAP representation.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

MarginalDistributionFromDRAP

Computes the matrix geometric representation of the marginal distribution of a discrete rational arrival process. It fails if the input isn't a valid DRAP representation.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

MarginalDistributionFromMRAP

Computes the matrix exponential representation of the marginal distribution of a marked rational arrival process. It fails if the input isn't a valid MRAP representation.

Input: vector of matrix0, matrix1, ... matrixM, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

MarginalDistributionFromDMRAP

Computes the matrix geometric representation of the marginal distribution of a discrete marked rational arrival process. It fails if the input isn't a valid DMRAP representation.

Input: vector of matrix0, matrix1, ... matrixM, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

MarginalDistributionFromMAP

Computes the phase type representation of the marginal distribution of a Markovian arrival process. It fails if the input isn't a valid MAP representation.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

MarginalDistributionFromDMAP

Computes the discrete phase type representation of the marginal distribution of a discrete Markovian arrival process. It fails if the input isn't a valid DMAP representation.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

MarginalDistributionFromMMAP

Computes the phase type representation of marginal distribution of a marked Markovian arrival process. It fails if the input isn't a valid MMAP representation.

Input: vector of matrix0, matrix1, ... matrixM, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

MarginalDistributionFromDMMAP

Computes the discrete phase type representation of marginal distribution of a discrete marked Markovian arrival process. It fails if the input isn't a valid DMMAP representation.

Input: vector of matrix0, matrix1, ... matrixM, $\varepsilon[10^{-14}]$

Output: vector-matrix pair

MarginalMomentsFromRAP

Calculates the first k marginal moments of the RAP with representation $\boldsymbol{D_0}, \boldsymbol{D_1}$: $\mu_i = i!\pi(-\boldsymbol{D_0})^{-i}\mathbb{1}$ $(i = 1, 2, \ldots, k)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-1}\boldsymbol{D_1} = \pi, \pi\mathbb{1} = 1$ [14]. It fails if the input isn't a valid RAP representation.

Input: matrix0, matrix1, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MarginalMomentsFromDRAP

Calculates the first k marginal moments of the DRAP with representation $\boldsymbol{D_0}, \boldsymbol{D_1}$: $f_i = i!\pi(\boldsymbol{I} - \boldsymbol{D_0})^{-i}\boldsymbol{D_0}^{i-1}\mathbb{1}$ $(i = 1, 2, \dots, k)$ are the factorial moments (and they are transformed into raw moments), where π is the solution of $\pi(\boldsymbol{I}-\boldsymbol{D_0})^{-1}\boldsymbol{D_1} = \pi, \pi\mathbb{1} = 1$. It fails if the input isn't a valid DRAP representation.

Input: matrix0, matrix1, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MarginalMomentsFromMRAP

Calculates the first k marginal moments of the MRAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}, \dots, \boldsymbol{D_M}$: $\mu_i = i!\pi(-\boldsymbol{D_0})^{-i}\mathbb{1}$ $(i = 1, 2, \dots, k)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-1}\sum_{k=1}^{M}\boldsymbol{D_k} = \pi, \pi\mathbb{1} = 1$ [5]. It fails if the input isn't a valid MRAP representation.

Input: vector of matrix0, matrix1, ... matrixM, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MarginalMomentsFromDMRAP

Calculates the first k marginal moments of the DMRAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}, \dots, \boldsymbol{D_M}$: $f_i = i!\pi(\boldsymbol{I} - \boldsymbol{D_0})^{-i}\boldsymbol{D_0}^{i-1}\mathbb{1}$ $(i = 1, 2, \dots, k)$ are the factorial moments (and they are transformed into raw moments), where π is the solution of $\pi(\boldsymbol{I} - \boldsymbol{D_0})^{-1}\sum_{k=1}^{M}\boldsymbol{D_k} = \pi, \pi\mathbb{1} = 1$. It fails if the input isn't a valid DMRAP representation.

Input: vector of matrix0, matrix1, ... matrixM, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MarginalMomentsFromMAP

Checks is the input is a MAP representation and calculates the first k marginal moments of the MAP with representation $(\boldsymbol{D_0}, \boldsymbol{D_1})$: $\mu_i = i!\pi(-\boldsymbol{D_0})^{-i}\mathbb{1}$ $(i = 1, 2, \dots, k)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-1}\boldsymbol{D_1} = \pi, \pi\mathbb{1} = 1$ [14]. It fails if the input isn't a valid MAP representation.

Input: matrix0, matrix1, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MarginalMomentsFromDMAP

Calculates the first k marginal moments of the DMAP with representation $\boldsymbol{D_0}, \boldsymbol{D_1}$: $f_i = i!\pi(\boldsymbol{D_0})^{-i}\boldsymbol{D_0}^{i-1}\mathbb{1}$ $(i = 1, 2, \dots, k)$ are the factorial moments (and they are transformed into raw moments), where π is the solution of $\pi(\boldsymbol{I} - \boldsymbol{D_0})^{-1}\boldsymbol{D_1} = \pi, \pi\mathbb{1} = 1$. It fails if the input isn't a valid DMAP representation.

Input: matrix0, matrix1, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MarginalMomentsFromMMAP

Calculates the first k marginal moments of the MMAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}, \dots, \boldsymbol{D_M}$: $\mu_i = i!\pi(-\boldsymbol{D_0})^{-i}\mathbb{1}$ $(i = 1, 2, \dots, k)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-1}\sum_{k=1}^{M}\boldsymbol{D_k} = \pi, \pi\mathbb{1} = 1$ [5]. It fails if the input isn't a valid MMAP representation.

Input: vector of matrix0, matrix1, ... matrixM, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

MarginalMomentsFromDMMAP

Calculates the first k marginal moments of the DMMAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}, \ldots, \boldsymbol{D_M}$: $f_i = i!\pi(\boldsymbol{I} - \boldsymbol{D_0})^{-i}\boldsymbol{D_0}^{i-1}\mathbb{1}$ $(i = 1, 2, \ldots, k)$ are the factorial moments (and they are transformed into raw moments), where π is the solution of $\pi(\boldsymbol{I} - \boldsymbol{D_0})^{-1}\sum_{k=1}^{M}\boldsymbol{D_k} = \pi, \pi\mathbb{1} = 1$. It fails if the input isn't a valid DMMAP representation.

Input: vector of matrix0, matrix1, ... matrixM, k $[2n-1]$, $\varepsilon[10^{-14}]$

Output: moments

LagkJointMomentsFromRAP

Calculates the matrix of the $E(X_0^i, X_{lag}^j)$ moments of the RAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}$: $E(X_0^i, X_{lag}^j) = i!j!\pi(-\boldsymbol{D_0})^{-i}\left((-\boldsymbol{D_0})^{-1}\boldsymbol{D_1}\right)^{lag}(-\boldsymbol{D_0})^{-j}\mathbb{1}$ $(i, j = 0, 1, \ldots, K)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-i}\boldsymbol{D_1} = \pi, \pi\mathbb{1} = 1$ [14]. It fails if the input isn't a valid RAP representation.

Input: matrix0, matrix1, K $[n]$, lag [1], $\varepsilon[10^{-14}]$

Output: moments

LagkJointMomentsFromDRAP

Calculates the matrix of the $E(X_0^i, X_{lag}^j)$ moments of the DRAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}, \ldots \boldsymbol{D_M}$. Factorial joint moments are:
$E(X_0^i, X_{lag}^j) = i!j!\pi(\boldsymbol{I} - \boldsymbol{D_0})^{-i}\boldsymbol{D_0}^{i-1}\left((\boldsymbol{I} - \boldsymbol{D_0})^{-1}\boldsymbol{D_1}\right)^{lag}(\boldsymbol{I} - \boldsymbol{D_0})^{-j-2}\boldsymbol{D_0}^{j-1}\mathbb{1}$ $(i, j = 0, 1, \ldots, K)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-i}\boldsymbol{D_1} = \pi, \pi\mathbb{1} = 1$. It fails if the input isn't a valid DRAP representation.

Input: vector of matrix0, matrix1, K $[n]$, lag [1], $\varepsilon[10^{-14}]$

Output: matrix of joint moments

LagkJointMomentsFromMRAP

Calculates the matrix of the $E(X_0^i, X_{lag}^j)$ moments for every arrival types of the MRAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}, \ldots \boldsymbol{D_M}$. Joint moments of type m are:
$E(X_{m,0}^i, X_{lag}^j) = i!j!\pi(-\boldsymbol{D_0})^{-i-1}\boldsymbol{D_m}\left((-\boldsymbol{D_0})^{-1}\sum_{s=1}^{M}\boldsymbol{D_s}\right)^{lag-1}(-\boldsymbol{D_0})^{-j}\mathbb{1}$ $(i, j = 0, 1, \ldots, K)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-i}\sum_{s=1}^{M}\boldsymbol{D_s} = \pi, \pi\mathbb{1} = 1$. It fails if the input isn't a valid MRAP representation.

Input: vector of matrix0, matrix1,... matrixM, K $[n]$, lag [1], $\varepsilon[10^{-14}]$

Output: matrix of joint moments

LagkJointMomentsFromDMRAP

Calculates the matrix of the $E(X_0^i, X_{lag}^j)$ moments for every arrival types of the DMRAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}, \ldots \boldsymbol{D_M}$. Factorial joint moments of type m are:
$E(X_{m,0}^i, X_{lag}^j) = i!j!\pi(\boldsymbol{I} - \boldsymbol{D_0})^{-i-1}\boldsymbol{D_0}^{i-1}\boldsymbol{D_m}\left((\boldsymbol{I} - \boldsymbol{D_0})^{-1}\sum_{s=1}^{M}\boldsymbol{D_s}\right)^{lag-1}(\boldsymbol{I} -$

$\boldsymbol{D_0})^{-j-2}\boldsymbol{D_0}^{j-1}\mathbb{1}$ $(i,j=0,1,\dots,K)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-i}\sum_{s=1}^{M}\boldsymbol{D_s}=\pi, \pi\mathbb{1}=1$. It fails if the input isn't a valid DMRAP representation.

Input: vector of matrix0, matrix1,... matrixM, K [n], lag [1], $\varepsilon[10^{-14}]$

Output: matrix of joint moments

LagkJointMomentsFromMAP

Calculates the matrix of the $E(X_0^i, X_{lag}^j)$ moments of the MAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}$: $E(X_0^i, X_{lag}^j) = i!j!\pi(-\boldsymbol{D_0})^{-i}\left((-\boldsymbol{D_0})^{-1}\boldsymbol{D_1}\right)^{lag}(-\boldsymbol{D_0})^{-j}\mathbb{1}$ $(i,j=0,1,\dots,K)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-i}\boldsymbol{D_1}=\pi, \pi\mathbb{1}=1$ [14]. It fails if the input isn't a valid MAP representation.

Input: matrix0, matrix1, K [n], lag [1], $\varepsilon[10^{-14}]$

Output: moments

LagkJointMomentsFromDMAP

Calculates the matrix of the $E(X_0^i, X_{lag}^j)$ moments of the DMAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}, \dots \boldsymbol{D_M}$. Factorial joint moments are:
$E(X_0^i, X_{lag}^j) = i!j!\pi(\boldsymbol{I}-\boldsymbol{D_0})^{-i}\boldsymbol{D_0}^{i-1}\left((\boldsymbol{I}-\boldsymbol{D_0})^{-1}\boldsymbol{D_1}\right)^{lag}(\boldsymbol{I}-\boldsymbol{D_0})^{-j-2}\boldsymbol{D_0}^{j-1}\mathbb{1}$ $(i,j=0,1,\dots,K)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-i}\boldsymbol{D_1}=\pi, \pi\mathbb{1}=1$. It fails if the input isn't a valid DMAP representation.

Input: vector of matrix0, matrix1, K [n], lag [1], $\varepsilon[10^{-14}]$

Output: matrix of joint moments

LagkJointMomentsFromMMAP

Calculates the matrix of the $E(X_0^i, X_{lag}^j)$ moments for every arrival types of the MMAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}, \dots \boldsymbol{D_M}$. Joint moments of type m are:
$E(X_{m,0}^i, X_{lag}^j) = i!j!\pi(-\boldsymbol{D_0})^{-i-1}\boldsymbol{D_m}\left((-\boldsymbol{D_0})^{-1}\sum_{s=1}^{M}\boldsymbol{D_s}\right)^{lag-1}(-\boldsymbol{D_0})^{-j}\mathbb{1}$ $(i,j=0,1,\dots,K)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-i}\sum_{s=1}^{M}\boldsymbol{D_s}=\pi, \pi\mathbb{1}=1$. It fails if the input isn't a valid MMAP representation.

Input: vector of matrix0, matrix1,... matrixM, K [n], lag [1], $\varepsilon[10^{-14}]$

Output: matrix of joint moments

LagkJointMomentsFromDMMAP

Calculates the matrix of the $E(X_0^i, X_{lag}^j)$ moments for every arrival types of the DMMAP of size n with representation $\boldsymbol{D_0}, \boldsymbol{D_1}, \dots \boldsymbol{D_M}$. Factorial joint moments of type m are:
$E(X_{m,0}^i, X_{lag}^j) = i!j!\pi(\boldsymbol{I}-\boldsymbol{D_0})^{-i-1}\boldsymbol{D_0}^{i-1}\boldsymbol{D_m}\left((\boldsymbol{I}-\boldsymbol{D_0})^{-1}\sum_{s=1}^{M}\boldsymbol{D_s}\right)^{lag-1}(\boldsymbol{I}-\boldsymbol{D_0})^{-j-2}\boldsymbol{D_0}^{j-1}\mathbb{1}$ $(i,j=0,1,\dots,K)$, where π is the solution of $\pi(-\boldsymbol{D_0})^{-i}\sum_{s=1}^{M}\boldsymbol{D_s}=\pi, \pi\mathbb{1}=1$. It fails if the input isn't a valid DMMAP representation.

Input: vector of matrix0, matrix1,... matrixM, K [n], lag [1], $\varepsilon[10^{-14}]$

Output: matrix of joint moments

LagCorrelationsFromRAP

Calculates the lag correlations of the RAP with representation $\boldsymbol{D_0}, \boldsymbol{D_1}$ of size n from lag 1 to lag L: $c_k = \dfrac{E(X_0, X_k) - \mu_1^2}{\mu_2 - \mu_1^2}$ $(k = 1, 2, \ldots, L)$ [14]. It fails if the input isn't a valid RAP representation.

Input: matrix0, matrix1, L [1], $\varepsilon[10^{-14}]$

Output: lagcorrelations

LagCorrelationsFromDRAP

Calculates the lag correlations of the RAP with representation $\boldsymbol{D_0}, \boldsymbol{D_1}$ of size n from lag 1 to lag L: $c_k = \dfrac{E(X_0, X_k) - \mu_1^2}{\mu_2 - \mu_1^2}$ $(k = 1, 2, \ldots, L)$ [14]. It fails if the input isn't a valid DRAP representation.

Input: matrix0, matrix1, L [1], $\varepsilon[10^{-14}]$

Output: lagcorrelations

LagCorrelationsFromMAP

Calculates the lag correlations of the MAP with representation $\boldsymbol{D_0}, \boldsymbol{D_1}$ of size n from lag 1 to lag L: $c_k = \dfrac{E(X_0, X_k) - \mu_1^2}{\mu_2 - \mu_1^2}$ $(k = 1, 2, \ldots, L)$ [14]. It fails if the input isn't a valid MAP representation.

Input: matrix0, matrix1, L [1], $\varepsilon[10^{-14}]$

Output: lagcorrelations

LagCorrelationsFromDMAP

Calculates the lag correlations of the MAP with representation $\boldsymbol{D_0}, \boldsymbol{D_1}$ of size n from lag 1 to lag L: $c_k = \dfrac{E(X_0, X_k) - \mu_1^2}{\mu_2 - \mu_1^2}$ $(k = 1, 2, \ldots, L)$ [14]. It fails if the input isn't a valid MAP representation.

Input: matrix0, matrix1, L [1], $\varepsilon[10^{-14}]$

Output: lagcorrelations

RAPFromMoments

Calculates a RAP representation based on 2n-1 marginal moments and the $n \times n$ matrix of the lag 1 joint moments based on [20].

Input: marginal moments, joint moments

Output: outputmx0, outputmx1

DRAPFromMoments

Calculates a DRAP representation based on 2n-1 marginal moments and the $n \times n$ matrix of the lag 1 joint moments.

Input: marginal moments, joint moments

Output: outputmx0, outputmx1

RAPFromMomentsAndCorrelations

Calculates a RAP representation based on the first $2n - 1$ marginal moments and first $2n - 3$ lag correlation parameters based on [15].

Input: marginal moments, lag correlations

Output: outputmx0, outputmx1

MRAPFromMoments
Calculates an MRAP representation based on 2n-1 marginal moments and the $n \times n$ matrices of the lag 1 joint moments based on [9].

Input: marginal moments, vector of jointmomentsmatrix1, ... jointmomentsmatrixM

Output: vector of matrix0, matrix1, ... matrixM,

MAP2Canonical
Calculates the canonical representation of the input MAP of size 2 if possible based on [3]. It fails if the input isn't a valid MAP representation.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: matrix0, matrix1

DMAP2Canonical
Calculates the canonical representation of the input DMAP of size 2 if possible based on ... . It fails if the input isn't a valid DMAP representation.

Input: matrix0, matrix1, $\varepsilon[10^{-14}]$

Output: matrix0, matrix1

StairCase
Computes a smaller representation of a RAP if possible using the staircase algorithm [5]. If $\boldsymbol{D_0}$=matrix0, $\boldsymbol{D_1}$=matrix1, and $\boldsymbol{B}$=similarity matrix then the small representation is the upper-left non-zero block of $(\boldsymbol{B}^{-1}\boldsymbol{D_0}\boldsymbol{B}, \boldsymbol{B}^{-1}\boldsymbol{D_1}\boldsymbol{B})$. The outputs in Mathematica and Matlab can be different for the same input due to that the built in singular value decomposition gives different results, but both are sufficient. For more details see the test examples.

Input: matrix0, matrix1, closing vector, $\varepsilon[10^{-14}]$

Output: size, similarity matrix

MStairCase
Computes a smaller representation of an MRAP using staircase algorithm [5].

Input: vector of matrix0, matrix1, ... matrixK, closing vector, $\varepsilon[10^{-14}]$

Output: size, similarity matrix

MinimalRepFromRAP
Computes a minimal representation of a RAP using the staircase method once for eliminating the redundancy caused by the initial vector and once for the closing vector [5].

Input: matrix0, matrix1

Output: smaller representation if exists

MinimalRepFromMRAP

Computes a minimal representation of an MRAP using the staircase method once for eliminating the redundancy caused by the initial vector and once for the closing vector [5]. It fails if the input isn't a valid MRAP representation.

Input: vector of matrix0, matrix1, ... matrixK

Output: vector of matrix0, matrix1, ... matrixK

MRAPContMinimize

Computes a minimal representation of an MRAP using the staircase method by eliminating the redundancy caused by closing vector [5].

Input: vector of matrix0, matrix1, ... matrixK

Output: vector of matrix0, matrix1, ... matrixK

MRAPObsMinimize

Computes a minimal representation of an MRAP using the staircase method by eliminating the redundancy caused by initial vector [5].

Input: vector of matrix0, matrix1, ... matrixK

Output: vector of matrix0, matrix1, ... matrixK

2.4 BuToolsSpecialProcesses

The BuToolsSpecialProcesses package contains functions associated with transient MAP/RAP (TMAP/TRAP), Markovian/rational binary trees (MBT/RBT). These functions are associated with the computation of characterizing set of moments of these processes and the computation of a representation based on a characterizing moments set.

MomentsFromTRAP

Calculates the order $0, 1, \ldots, 2n-1$ marginal moments ($\mu_i = E(X_0^i I_{X_0<\infty})$) of the TRAP of size n with representation $\alpha, \boldsymbol{D_0}, \boldsymbol{D_1}$: $\mu_i = i!\alpha(-\boldsymbol{D_0})^{-i-1}\boldsymbol{D_1}\mathbb{1}$ ($i = 1, 2, \ldots, 2n-1$) [7].

Input: vector, matrix0, matrix1

Output: moments

Lag1JointMomentsFromTRAP

Calculates the matrix of the $E(X_0^i X_1^j I_{X_0<\infty, X_1<\infty})$ moments of the transient RAP with representation $\alpha, \boldsymbol{D_0}, \boldsymbol{D_1}$: $E(X_0^i, X_1^j I_{X_0<\infty, X_1<\infty}) = i!j!\alpha(-\boldsymbol{D_0})^{-i-1}\boldsymbol{D_1}(-\boldsymbol{D_0})^{-j-1}\boldsymbol{D_1}\mathbb{1}$ ($i = 1, 2, \ldots, n$) [7].

Input: vector, matrix0, matrix1

Output: jointmoments

TRAPFromMoments

Calculates an TRAP representation based on the marginal moments and the lag 1 joint moments based on [7].

Input: marginal moments, jointmoments

Output: vector, matrix0, matrix1

TRAPContMinimize
Computes a minimal representation of an TRAP using the staircase method by eliminating the redundancy caused by closing vector [7].

Input: vector, matrix0, matrix1,

Output: vector, matrix0, matrix1,

TRAPObsMinimize
Computes a minimal representation of an TRAP using the staircase method by eliminating the redundancy caused by initial vector [7].

Input: vector, matrix0, matrix1,

Output: vector, matrix0, matrix1,

TRAPMinimize
Computes a minimal representation of an TRAP using the staircase method once for eliminating the redundancy caused by the initial vector and once for the closing vector [7].

Input: vector, matrix0, matrix1,

Output: vector, matrix0, matrix1,

MomentsFromRBT
Calculates the order $0, 1, \ldots, 2n-1$ marginal moments $(E(X_0^i I_{X_0<\infty}))$ of the RBT of size n with representation $\alpha, \boldsymbol{D_0}, \boldsymbol{B}$: $\mu_i = i!\alpha(-\boldsymbol{D_0})^{-i-1}\boldsymbol{B}\mathbb{1}$ $(i = 1, 2, \ldots, 2n-1)$ [7].

Input: vector, matrix0 $(n \times n)$, matrix1 $(n \times n^2)$,

Output: moments

GammaijkMomentsFromRBT
Calculates the matrix of the $\gamma_{ijk} = E(X_0^i X_1^j Y_0^k I_{X_0<\infty, X_1<\infty, Y_0<\infty})$ moments [7] of the RBT with representation $\alpha, \boldsymbol{D_0}, \boldsymbol{B}$:

$$E((X_0^k)^i, X_1^j) = i!j!k!\alpha(-\boldsymbol{D_0})^{-i-1}\boldsymbol{B}\left((-\boldsymbol{D_0})^{-j-1}\boldsymbol{B}\mathbb{1} \otimes (-\boldsymbol{D_0})^{-k-1}\boldsymbol{B}\mathbb{1}\right).$$

Input: vector, matrix0, matrix1

Output: jointmoments

RBTFromMoments
Calculates an RBT representation based on the marginal moments and the γ_{ijk} moments based on [7].

Input: marginal moments, jointmoments

Output: vector, matrix0 $(n \times n)$, matrix1 $(n \times n^2)$,

RBTContMinimize

Computes a minimal representation of an RBT using the staircase method by eliminating the redundancy caused by closing vector [7].

Input: vector, matrix0 ($n \times n$), matrix1 ($n \times n^2$),

Output: vector, matrix0 ($n \times n$), matrix1 ($n \times n^2$),

RBTObsMinimize

Computes a minimal representation of an RBT using the staircase method by eliminating the redundancy caused by initial vector [7].

Input: vector, matrix0 ($n \times n$), matrix1 ($n \times n^2$),

Output: vector, matrix0 ($n \times n$), matrix1 ($n \times n^2$),

RBTMinimize

Computes a minimal representation of an RBT using the staircase method once for eliminating the redundancy caused by the initial vector and once for the closing vector [7].

Input: vector, matrix0 ($n \times n$), matrix1 ($n \times n^2$),

Output: vector, matrix0 ($n \times n$), matrix1 ($n \times n^2$),

TRAPToTMAP

Similarity transforms the non-Markovian representation of a transient rational arrival process (vector ($1 \times n$), matrix0 ($n \times n$), matrix1 ($n \times n$)) to the Markovian representation (outputvec, outputmx0, outputmx1) of the same size, if possible using the method from [20].

Input: vector, matrix0, matrix1

Output: outputvec, outputmx0, outputmx1

RBTToMBT

Similarity transforms the non-Markovian representation of a rational binary tree process (vector ($1 \times n$), matrix0 ($n \times n$), matrix1 ($n \times n^2$)) to the Markovian representation (outputvec, outputmx0, outputmx1) of the same size, if possible using the method from [20].

Input: vector, matrix0 ($n \times n$), matrix1 ($n \times n^2$),

Output: outputvec, outputmx0 ($n \times n$), outputmx1 ($n \times n^2$),

2.5 BuToolsFluid

The BuToolsFluid package contains the functions for solving the multi regime (with piecewise continuous fluid rates) Markov fluid models. These models are defined by the vector defining the regions of the fluid buffer by their boundaries, vector thres, the generator matrix of the background continuous time Markov chain, matrix Q, and the diagonal matrix of the fluid rates, matrix R, both, for all regions of the fluid buffer. The vector of the boundaries contains also the lower and the upper buffer limit. Q and R are three dimensional: $Q(:,:,i)$ and $R(:,:,i)$ are the generator and the rate matrix for the i-th region. The functions computes the probability masses at the boundaries, matrix pmatrix (number of boundaries $\times$ number of states), and the fluid density values for M (default= 10^5) equidistance buffer levels, matrix f ($M\times$ number of states).

additive_decomposition

This function is based on the additive decomposition method [12]. The program uses the function $X = lyapunov(A, B, C)$ to solve the following equation: $A \cdot X + X \cdot B = -C$. (MATLAB's Control System Toolbox contains a program called lyap.m which solves the same problem, but with a different algorithm.) Furthermore it needs the function $X = ordereigs(A)$, which returns X, the ordered eigenvalues of the upper quasi-triangular matrix A. (For newer MATLAB distributions it can be replaced with $ordeig()$.)

At this moment this function can not be used for Octave, as it uses $ordschur()$ a program only available in MATLAB.

Input: matrices Q, R and the vector thres

Output: matrices pmatrix, f

matrix_analytic

The function is based on the matrix-analytic method proposed in [6]. It calculates the characterizing matrix Ψ by the function matrix_analytic_psi. This computation is based on the numerical solution of the quadratic matrix equation $G = A0 + A \cdot G + A2 \cdot G^2$ by the cyclic reduction algorithm ($[G, R, U] = QBD\_CR(A0, A1, A2)$). (Other solvers are available in the SMCSolver package [1].

Input: matrices Q, R and the vector thres

Output: matrices pmatrix, f

3 Other related libraries, utilities

3.1 The `libphprng` library

The `libphprng` library is a pseudo-random number generation library for omnet++ and ns2. It implements several efficient algorithms to obtain PH distributed random numbers.

Requirements to compile the library:

- an existing installation of omnet++
- cmake build system
- gnu C++ compiler
- the Eigen3 linear algebra package for c++

More details on the library (the algorithms included and the usage) can be found in [18].

References

[1] D. A. Bini, B. Meini, S. Steffé, and B. Van Houdt. Structured markov chains solver: software tools. In *Proceeding from the 2006 workshop on Tools for solving structured Markov chains*, SMCtools '06, New York, NY, USA, 2006. ACM.

[2] A. Bobbio, A. Horváth, and M. Telek. Matching three moments with minimal acyclic phase type distributions. *Stochastic models*, pages 303–326, 2005. Link.

[3] L. Bodrog, A. Heindl, G. Horváth, and M. Telek. A markovian canonical form of second-order matrix-exponential processes. *European Journal of Operation Research*, 190:459–477, 2008. Link.

[4] L. Bodrog, A. Horváth, and M. Telek. Moment characterization of matrix exponential and Markovian arrival processes. *Annals of Operations Research*, 160:51–68, 2008. Link.

[5] Peter Buchholz and Miklós Telek. On minimal representation of rational arrival processes. *Annals of Operations Research, ANOR-1588*, 2010. Link.

[6] Ana da Silva Soares and Guy Latouche. Fluid queues with level dependent evolution. *European Journal of Operational Research*, 196:1041–1048, 2009.

[7] Sophie Hautphenne and Miklós Telek. Extension of some MAP results to transient MAPs and markovian binary trees. *Performance Evaluation*, 70(9):607 – 622, 2013. Link.

[8] Qi-Ming He and Hanqin Zhang. Spectral polynomial algorithms for computing bi-diagonal representations for phase type distributions and matrix-exponential distributions. *Stochastic Models*, 22:289–317, 2006. Link.

[9] András Horváth, Gábor Horváth, and Miklós Telek. A traffic based decomposition of two-class queueing network with priority service. *Computer networks*, 53(8):1235–1248, 2009. Link.

[10] András Horváth, Sándor Rácz, and Miklós Telek. Moments characterization of the class of order 3 matrix exponential distributions. In *16th Int. Conf. on Analytical and Stochastic Modeling Techniques and Applications (ASMTA)*, volume 5513 of *LNCS*, pages 174 – 188, Madrid, Spain, June 2009. Springer. Link.

[11] Gábor Horváth and Miklós Telek. On the canonical representation of phase type distributions. *Performance Evaluation*, 66(8):396 – 409, 2009. Link.

[12] H. E. Kankaya and N. Akar. Solving multi-regime feedback fluid queues. *Stochastic Models*, 24:425–450, 2008.

[13] I. Kolossváry and M. Telek. Explicit evaluation of me(3) membership. In *8th International Conference on Quantitative Evaluation of SysTems (QEST), fast abstract*, pages 11–13, Aachen, Germany, sept. 2011. Centre for Telematics and Information Technology (CTIT), University of Twente. Link.

[14] G. Latouche and V. Ramaswami. *Introduction to matrix analytic methods in stochastic modeling*. SIAM, Philadelphia, 1999.

[15] K. Mitchell and A. van de Liefvoort. Approximation models of feed-forward g/g/1/n queueing networks with correlated arrivals. *Performance Evaluation*, 51(2-4):137 – 152, 2003. Link.

[16] S. Mocanu and C. Commault C. Sparse representations of phase-type distributions. *Stochastic Models*, 15:759–778, 1999.

[17] J. Papp and M. Telek. Canonical representation of discrete phase type distributions of order 2 and 3. In *In Proc. of UK Performance Evaluation Workshop, UKPEW 2013*, 2013. Link.

[18] P. Reinecke and G. Horvath. Phase-type distributions for realistic modelling in discrete-event simulation. In *SIMUTools '12: Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, 2012. Link.

[19] M. Telek and A. Heindl. Matching moments for acyclic discrete and continuous phase-type distributions of second order. *International Journal of Simulation Systems, Science & Technology*, 3(3-4):47–57, dec. 2002. Link.

[20] M. Telek and G. Horváth. A minimal representation of markov arrival processes and a moments matching method. *Performance Evaluation*, 64(9-12):1153–1168, Aug. 2007. Link.

[21] A. van de Liefvoort. The moment problem for continuous distributions. Technical report, University of Missouri, WP-CM-1990-02, Kansas City, 1990. Link.