Manual of MCKR
version 0.1.0

October 13, 2020

Contents

1 Getting started: What is MCKR and how to use it?

MCKR is the abbreviation of "Monte Carlo Kac-Rice". This Julia app is an implementation of the algorithms in [2], and stems from the code in the GitHub repository [3]. With the app one can compute a Kac-Rice integral on an arbitrary parameter box using Monte Carlo integration for the examples included in the example bank. The user can create their own input files by adapting the example files.

Using the tasks provided in the app, the user can address the questions on the number of solutions of parametrized polynomial equations indicated in [2].

The app is a Julia program written using Julia version 1.4.2. Julia must be installed on the system, and it is assumed that it is added to the system's path (see Julia's webpage for instructions [1]).

In order to obtain the app, follow the link https://github.com/Hovakhshatra/MCKR_App/releases, from where you download the zipped folder. After unzipping it, the folder MCKR_app is created. The app is now ready to use.

In the terminal, navigate to the MCKR_app folder and type

```
julia MCKR.jl input-file
```

2 Input files

The input file is read line by line. A typical input file has the following structure. A 'header', a 'body' and an 'optional system-info request ending'.

Header	Task: Save_location: Save_name: MCKR_file_name:
Body	<arguments>: <options>
System-info request	Include cpu information Include memory information

Each line has one of the two following shapes.

1. <keyword>: <value>
2. Include <something>

Note that there must be one space after colon and before the value in the first case.

Every additional line not starting with a keyword (see below) will be ignored. This can be used to write some comments, and to have an input template and commenting out with '#' optional parts at the beginning of their corresponding lines.

If a keyword is repeated, only the value in the first occurrence is considered.

In what follows we detail the possible tasks, keywords, and give information on the examples in the Example bank. The app is under constant development and new tasks and options will be included.

3 Tasks

3.1 `MCKR integrate`

This task computes a single Monte Carlo integration.

- Body arguments: `Box`, `MC_method`, `Sample_size`.
- Body options: `Include standard error`, `Include integration time`.

3.2 `MCKR integrate parallelized`

This task computes a single Monte Carlo integration using parallelization.

- Body arguments: `Box`, `MC_method`, `Sample_size`, `Workers_number`.
- Body options: `Include standard error`, `Include integration time`.

3.3 `MCKR parallel compare`

This task compares speed of parallelization by computing a single Monte Carlo integral once with and once without parallelization.

- Body arguments: `Box`, `MC_method`, `Sample_size`, `Workers_number`.

3.4 `MCKR method comparison table parallelized`

This task computes an integral using both Simple and Antithetic Monte Carlo integration and for different sample sizes. The information in the output file is similar to that of Table 1 in [2].

- Body arguments: `Box`, `Sample_size`, `Workers_number`.

3.5 `MCKR method comparison table parallelized - include tracking`

This task computes an integral using both Simple and Antithetic Monte Carlo integration and for different sample sizes. The information in the output file is similar to the table 1 in [2]. This task also creates a text file called "tracker.txt" in the same directory, which gets updated after each integration is completed.

- Body arguments: `Box`, `Sample_size`, `Workers_number`.

3.6 `MCKR bisect search`

This task runs a bisect search to find a sub-box of the initial parameter box inside the multistationarity region or states that it is not possible.

- Body arguments: `Box`, `Stop_criterion_subbox_size`, `Stop_criterion_bisect_step`, `MC_method`, `Sample_size`.
- Body options: `First_axis`, `Include last box integration info`, `Include searching time`.

3.7 `MCKR bisect search parallelized`

Same as `MCKR bisect search`, but with parallelization.

- Body arguments: `Box`, `Stop_criterion_subbox_size`, `Stop_criterion_bisect_step`, `MC_method`, `Sample_size`, `Workers_number`.
- Body options: `First_axis`, `Include last box integration info`, `Include searching time`.

3.8 MCKR bisect search parallelized - include tracking

Same as MCKR bisect search parallelized, but also creates a text file "tracker.txt" in the same directory, which gets updated live after each integration is completed.

- Body arguments: Box, Stop_criterion_subbox_size, Stop_criterion_bisect_step, MC_method, Sample_size, Workers_number.
- Body options: First_axis, Include last box integration info, Include searching time.

3.9 MCKR generate example

This task lets the user to generate a new example in the example bank folder of the app. To do so the user needs to give necessary information asked by the following arguments.

- Body arguments: Example_name, Number_of_variables, Number_of_parameters, Determinant_expression, g_expression, summand_expression, Distribution.

3.10 MCKR generate example - conserved and positive orthant

This task is similar to MCKR generate example task with the difference that the values of the variables are restricted to the positive orthant and are assumed to be bounded above by values of some parameters.

- Body arguments: Example_name, Number_of_variables, Number_of_parameters, Determinant_expression, g_expression, Distribution, Variables_upper_bound.

4 Index of keywords

Ordered alphabetically.

4.1 Box

This keyword determines the parameter box (a Cartesian product of intervals). Therefore if the parameter space is of dimension m, then a parameter box B can be written as

$$B = B_1 \times B_2 \times \cdots \times B_m$$

where each B_i is an interval $[a_i, b_i] \subset \mathbb{R}$. The value of this keyword depends on the choice of the random distribution: uniform or normal. If the random distribution is uniform, then the value for Box is of the form:

- [a_1,b_1],[a_2,b_2],...,[a_m,b_m]

where a_i and b_i are real numbers.

If the random distribution is normal, then the value of Box includes the mean (μ_i) and the variance (σ_i^2) of the normal distribution for each parameter, given as follows:

- [a_1,b_1,mu_1,v_1],[a_2,b_2,mu_2,v_2],...,[a_m,b_m,mu_m,v_m]

where a_i, b_i and mu_i are real numbers and v_i is a non-negative real number.

4.2 Determinant_expression

This keyword receives the mathematical expression of "det $\left(J_{g_{\bar{\kappa}}}(t)\right)$" in 1D math input format. The basic operations are denoted by "+" (addition), "-" (subtraction), "*" (multiplication), "/" (division), "^" (exponentiation). Please refer to Theorem 1.1 and equation (8) of [2] to see how to compute this expression. The variables of the system are denoted by x1, ..., xn where n is an integer equal to the number of variables of the system. Parameters are denoted by k1, ..., km where m is an integer equal to the number of parameters of the system. The n linearly isolated parameters should be indexed first.

4.3 Distribution

This keyword determines the distribution on the parameters in a example generating task. The current possible values for this keyword are:

- `Uniform`
- `Normal`

4.4 Example_name

This keyword determines the name of the new example file to be generated in the example bank folder of the app. The default value is `User_made_example`.

4.5 First_axis

This keyword determines the first parameter which the bisect search should start bisecting along its axis. The default value is 1.

4.6 g_expression

This keyword receives the mathematical expression of "$g_{\bar{\kappa},i}(t)$"'s where $i = 1, \cdots, n$ and n is the number of variables of the system. Each expression should be written in a separate line with this keyword at their beginning. The index, i, is determined by the order of lines. These expressions should be given in 1D math input format. The basic operations are denoted by "`+`" (addition), "`-`" (subtraction), "`*`" (multiplication), "`/`" (division), "`^`" (exponentiation). Please refer to [2, Theorem 1.1] to see how to compute these expressions. The variables of the system are denoted by `x1`, ..., `xn` where n is an integer equal to the number of variables of the system. Parameters are denoted by `k1`, ..., `km` where m is an integer equal to the number of parameters of the system.

4.7 Include cpu information

Using this keyword adds a section at the end of the report file containing the number of cpus and their characteristics.

4.8 Include integration time

Using this keyword the report includes the computation time.

4.9 Include last box integration info

Using this keyword with a search task, the I-hat and e-hat of the last sub-box of the search algorithm are computed again.

4.10 Include memory information

Using this keyword adds a section at the end of the report file containing the total memory of the system and the free memory before the computations start.

4.11 Include searching time

Using this keyword in a search task adds the total computation time for the search algorithm in the report file.

4.12 Include standard error

Using this keyword includes the standard error of the Monte Carlo integration.

4.13 MC_method

This keyword determines the method of Monte Carlo integration. The current possible values for this keyword are:

- Simple
- Antithetic

4.14 MCKR_file_name

This keyword determines the parametric system of polynomial equations for the computations. From the Example bank we offer:

- MCKR_example1_uniform_2p
- MCKR_example1_uniform_8p
- MCKR_example2_normal
- MCKR_example2_uniform
- MCKR_example3_uniform
- MCKR_example4_uniform
- MCKR_example5_uniform

The choice of the random distribution on the parameters is as indicated in the file name.

4.15 Number_of_parameters

This keyword determines number of parameters in the parametric system of equations that the user wants to create a new example file for it in the example bank of the app.

4.16 Number_of_variables

This keyword determines number of variables in the parametric system of equations that the user wants to create a new example file for it in the example bank of the app.

4.17 Sample_size

This keyword determines the sample size or a set of sample sizes depending on the task. Sample size is the number of randomly chosen points from the parameter box. If the task requires one value for the sample size, then one positive integer should be given. If the task requires a set of sample sizes, then three positive integers are given in the following format:

- `Sample_size_start,Sample_size_step_size,Sample_size_end`

This triplet creates a set of sample sizes starting from `Sample_size_start` and increasing with '`Sample_size_step_size`' until reaching the largest value less than or equal to `Sample_size_end`.

4.18 Save_location

This keyword determines the directory (folder) in which the report file will be saved. The default directory is the folder `MCKR_app` (that is, `./`).

4.19 Save_name

This keyword determines the name of the report file. The default value is `Output.txt`.

4.20 `Stop_criterion_bisect_step`

This keyword determines the stop criterion for the bisect search with respect to the number of bisecting steps. The value is a finite positive integer.

4.21 `Stop_critetion_subbox_size`

This keyword determines the stop criterion for the bisect search with respect to the length of the edges of the sub-boxes. If the parameter space is of dimension m, then the value for this keyword has the following format:

- `d_1,d_2,...,d_m`

where `d_i` are finite positive real numbers. The bisection does not continue along an axis where the edge of the sub-box has length less than d_i, and the algorithm stops if there is no direction left to bisect along it. If `Stop_criterion_bisect_step` is also given, then the most restrictive condition is considered.

4.22 `summand_expression`

This keyword is used for the task `MCKR generate example`. The user must convert the part of integration domain of the Kac-Rice formula which is on the variable space to the unit cube. To do so one needs to use the following formulas.

$$\begin{aligned}
\int_a^b s(t)dt &= \int_0^1 \Big((b-a)s\big((b-a)t+a\big)\Big)dt \\
\int_a^\infty s(t)dt &= \int_0^1 \Big((1-a)s\big((1-a)t+a\big)+\frac{1}{t^2}s(\frac{1}{t})\Big)dt \\
\int_{-\infty}^b s(t)dt &= \int_0^1 \Big(\frac{1}{t^2}s(\frac{-1}{t})+(b+1)s\big((b+1)t-1\big)\Big)dt \\
\int_{-\infty}^\infty s(t)dt &= \int_0^1 \Big(\frac{1}{t^2}s(\frac{-1}{t})+2s(2t-1)+\frac{1}{t^2}s(\frac{1}{t})\Big)dt
\end{aligned}$$

What the user need to do for this keyword is to expand its Kac-Rice integral along each variable according to the above formulas and at the end write each term using "`s`" as the integrand function name and include one extra argument "`B`". You can see an example below.

Consider the example of [2, Subsection 2.3]. The Kac-Rice integral is of the following form.

$$\hat{r}(B) = \int_0^\infty \int_{a_2}^{b_2} s(x_1,\kappa_2)d\kappa_2 dx_1.$$

Therefore its transformed version will be as the following.

$$\int_0^1 \int_{a_2}^{b_2} \Big(s(x_1,\kappa_2)+\frac{1}{x_1^2}s(\frac{1}{x_1},\kappa_2)\Big)d\kappa_2 dx_1.$$

Hence the value of the `summand_expression` is the following.

```
s(x1,k2,B)+s(1/x1,k2,B)/x1^2
```

Note: The basic operations are denoted by "`+`" (addition), "`-`" (subtraction), "`*`" (multiplication), "`/`" (division), "`^`" (exponentiation). The variables of the system are denoted by `x1`, ..., `xn` where n is an integer equal to the number of variables of the system. Parameters are denoted by `k1`, ..., `km` where m is an integer equal to the number of parameters of the system. The n linearly isolated parameters should be indexed first.

4.23 `Task`

This keyword determines the computational task for the app. The current possible values for this keyword are listed below (and explained in Section 3):

- `MCKR integrate`

- `MCKR integrate parallelized`
- `MCKR parallel compare`
- `MCKR method comparison parallelized`
- `MCKR method comparison parallellized - include tracking`
- `MCKR bisect search`
- `MCKR bisect search parallelized`
- `MCKR bisect search parallelized - include tracking`
- `MCKR generate example`
- `MCKR generate example - conserved and positive orthant`

4.24 Variables_upper_bound

This keyword is used in the task `MCKR generate example - conserved and positive orthant`. If the system has n variables, then the value for this keyword has the following format:

- `d_1,d_2,...,d_n`

where `d_i` is index of the parameter that its value is an upper bound for the value of the variable x_i.

4.25 `Workers_number`

This keyword determines how many workers are to be used in parallel. The possible values are listed below.

- A positive integer, e.g. `2`.
- `cpu_number` (as many as the number of cpus of the system).
- `half_cpu_number` (as many as half of the number of cpus of the system).

5 List of examples in the example bank

5.1 `MCKR_example1_uniform_2p`

Equation (23), subsection 2.1 of [2], parameters equipped with uniform distribution.

5.2 `MCKR_example1_uniform_8p`

Equation after equation (20), subsection 2.1 of [2], parameters equipped with uniform distribution.

5.3 `MCKR_example2_normal`

Equation (27), subsection 2.2 of [2], parameters equipped with normal distribution.

5.4 `MCKR_example2_uniform`

Equation (27), subsection 2.2 of [2], parameters equipped with uniform distribution.

5.5 `MCKR_example3_uniform`

Equation (30), subsection 2.3 of [2], parameters equipped with uniform distribution.

5.6 `MCKR_example4_uniform`

Subsection 2.4 of [2], parameters equipped with uniform distribution.

5.7 `MCKR_example5_uniform`

Subsection 2.5 of [2], parameters equipped with uniform distribution.

References

[1] *The Julia programming language.* `https://julialang.org/`.

[2] E. Feliu and A. Sadeghimanesh. *Kac-Rice formulas and the number of solutions of parametrized systems of polynomial equations.* arXiv:2010.00804, 2020.

[3] A. Sadeghimanesh and E. Feliu. *MCKR project, Version 1.0.0.* Available online at `https://doi.org/10.5281/zenodo.4026954`, 2020.