# Analysis and Simulation Tools for Probabilistic Real-Time Systems

Dorin Maxim
Loria - University of Lorraine
dorin.maxim@loria.fr

Antoine Bertout
Inria de Paris
antoine.bertout@inria.fr

***Abstract*—In this paper we present two tools meant to simulate and analyze probabilistic real-time task sets. That is, tasks sets which have their timing parameters represented by discrete probabilistic distributions. We describe the main features of each tool and provide configuration details necessary to use them. The two tools are compared, pointing out the advantages and disadvantages of each one, so that interested users can make an informed choice regarding which tool best fits their needs. Both tools are open source and freely available. One of the main objectives of this paper is to make these tools available to the real-time systems research community, which is also invited to participate in their improvements, by giving feedback and even extending the implementations.**

***Keywords*—Simulation, Probabilistic Real-Time Systems, Probabilistic Analysis, pWCET, pMIT**

## I. Introduction

In the real-time community, numerous theoretical analyses have been developed to assess the capability of a system to respect its timing constraint, in other words, its schedulability. In that context, a system is generally described as a set of tasks that are successively released, each described by their worst-case parameters (execution time, arrivals pattern, offset, etc). Analyses have been proposed to compute if the system is schedulable or not. Another way of checking the schedulability of the systems, and at the same time having more details about the execution of various tasks in a given interval (job response time, starting executions date, etc.) is by simulating its run-time execution. This is generally counterbalanced by an accepted (empirical) margin of error introducing a large amount of pessimism to compensate for the potential optimism of the initial results.

Recently, a new trend has emerged for analyzing real-time systems, in the form of probabilistic analysis [1]–[4]. That is, task parameters are described using probability distributions (as opposed to single worst-case values) and the analysis computes response time probability distributions from which it is possible to extract the probability that a job may miss its deadline.

Probabilistic analyses of real-time systems provide levels of confidence on the response time distributions, helping to assess how safe its use is in practice. Nevertheless, the analysis of some systems might be excessively hard to compute especially considering multiple probabilistic parameters (as probabilistic worst-case execution time and probabilistic minimal inter-arrival time) and asynchronous arrivals of jobs due to the complexity as well as the amount of convolution operations required. An alternative to analyzing a probabilistic task set is to simulate it and gather the relevant information from the traces observed, such as frequency of response times. Although there exist currently several analysis methods for probabilistic real-time systems, very few simulation tools have been proposed by the community. In this paper, we present two tools able to simulate probabilistic task set. The tools are freely available for the community, with great potential to being extended to enhance their capabilities.

*Related works:* Various simulators exist to evaluate deterministic task sets (by deterministic we mean that they are not probabilistic). For instance, MAST [5] and Cheddar [6] both provide analysis and simulation. SimSo [7] is focused only on the simulation of multiprocessor real-time system with consideration of overhead. Also written in Python, pyCPA [8] is an implementation of the Compositional Performance Analysis approach to analyze worst-case timing behavior of real-time distributed systems. CPAL [9] is meant to be both an analysis and simulation tool, as well as programming language for real-time systems, in the sense that the simulated model can directly be deployed on the hardware and the system obtained has the same (timing) properties as the simulated one. An existing simulator that can handle probabilistic execution times and probabilistic inter-arrival times of tasks is RTSim [10]. We note that RTSim is not specifically targeted towards probabilistic simulations, but towards deterministic ones.

*Organization of the paper:* In Section II we describe our system model composed of tasks with probabilistic parameters. Section III is dedicated to the presentation of PAnSim and of an extension of SimSo, two tools able to simulate probabilistic task sets. We provide a comparison of the two tools in Section IV before concluding in Section V.

## II. Model

In this section, we present the task set model used for the simulation and analysis of probabilistic real-time systems.

We model the system as a set of $n$ tasks $\{\tau_1, \tau_2, ..., \tau_n\}$ scheduled by a preemptive algorithm. Each task $\tau_i$ generates an infinite number of successive jobs $\tau_{i,j}$, with $j = 1, \ldots, \infty$. All jobs are assumed to be independent of other jobs of the same task and those of other tasks. We assume that a job

may miss its deadline and postpone the execution of the next job of the same task.

Each task $\tau_i$ is characterized by a probabilistic worst-case execution time (pWCET) denoted by $\mathcal{C}_i$ and by a probabilistic minimal inter-arrival time (pMIT) denoted by $\mathcal{T}_i$.

Each parameter is represented by a random variable $\mathcal{X}$ having a probability distribution $f_{\mathcal{X}}(\cdot)$ with $f_{\mathcal{X}}(x) = P(\mathcal{X} = x)$. The possible values of $\mathcal{X}_i$ belong to the interval $[X^{\min}, X^{\max}]$. In this paper we associate the probabilities to the possible values of a random variable using the following notation:

$$\mathcal{X} = \begin{pmatrix} X^0 = X^{min} & X^1 & \cdots & X^k = X^{max} \\ f_{\mathcal{X}}(X^{min}) & f_{\mathcal{X}}(X^1) & \cdots & f_{\mathcal{X}}(X^{max}) \end{pmatrix}$$

where $\sum_{j=0}^{k} f_{\mathcal{X}}(X^j) = 1$.

The notions of pWCET and pMIT are defined as follows. For more details about these concepts, the reader can refer to [1].

**Definition 1** (From [1])**.** The probabilistic execution time (pET) of a job of a task describes the probability that the execution time of the job is equal to a given value.

**Definition 2** (From [1])**.** The probabilistic worst-case execution time (pWCET) of a task describes the probability that the worst-case execution time of that task is equal to a given value.

A safe pWCET $\mathcal{C}_i$ is an upper bound on the pETs $\mathcal{C}_i^j$, $\forall j$ and it may be described by the relation $\succeq$ as $\mathcal{C}_i \succeq \mathcal{C}_i^j$, $\forall j$. Graphically this means that the CDF of $\mathcal{C}_i$ stays under the CDF of $\mathcal{C}_i^j$, $\forall j$.

Following the same reasoning the probabilistic minimal inter-arrival time (pMIT) denoted by $\mathcal{T}_i$ describes the probabilistic minimal inter-arrival times of all jobs.

**Definition 3** (From [1])**.** The probabilistic inter-arrival time (pIT) of a job of a task describes the probability that the job arrival time occurs at a given value.

**Definition 4** (From [1])**.** The probabilistic minimal inter-arrival time (pMIT) of a task describes the probability that the minimal inter-arrival time of that task is equal to a given value.

A safe pMIT $\mathcal{T}_i$ is a lower-bound on the pITs $\mathcal{T}_i^j$, $\forall j$ and it may be described by the relation $\succeq$ as $\mathcal{T}_i^j \succeq \mathcal{T}_i$, $\forall j$. Graphically this means that the CDF of $\mathcal{T}_i$ stays above the CDF of $\mathcal{T}_i^j$, $\forall j$.

A job of a task must finish its execution before its relative deadline $D_i$ with $D_i = \mathcal{T}_i^{min}$, i.e. the minimal value of the $\mathcal{T}_i$ distribution. Hence, a task $\tau_i$ is represented by a tuple $(\mathcal{C}_i, \mathcal{T}_i, D_i)$.

We assume that the pWCET and pMIT distributions of two tasks $\tau_i$ and $\tau_j$ are independent [11]. In this paper, we use the terminology of the probabilistic worst-case response time analysis (pMIT and pWCET). The reader interested in the probabilistic (non worst-case) response time distributions can indistinctly use the same reasoning with pIT and pET parameters. The simulators later presented are agnostic of the chosen reasoning.

Existing probabilistic schedulability analyses [1], [3] produce task worst-case response time distribution pWCRT, counterpart of the deterministic worst-case response time (WCRT) [12]). From the pWCRT, the deadline miss probability (DMP) of a task can be deduced.

## III. The proposed simulators

In this section, we describe the two probabilistic simulators and provide an example of use by applying them on the task set depicted in Table I scheduled by a fixed-task priority policy. This task set has 5 tasks, and each task is represented by a pWCET distribution and a pMIT distribution, with 10 values per distribution. For the sake of simplicity all probabilities of appearance are equal to 0.1 and we omit them from the table. Task indexes give their priorities, with task $\tau_1$ having the highest priority and task $\tau_5$ having the lowest one. Throughout the rest of the paper we are interested in analyzing and simulating task $\tau_5$, unless otherwise specified.

### A. PAnSim

PAnSim (short for Probabilistic Analysis and Simulation) is a MATLAB implementation (released under the CeCILL licence, compatible with GNU GPL) consisting of a simulator and an analysis tool. In this regard, PanSim is similar to MAST [5] or Cheddar [6]. The simulator takes as input real-time task sets with parameters given as discrete random variables: worst-case execution time distributions and minimal inter-arrival time distributions. Any of these distributions can be given as a single values distribution, in which case the parameter is deterministic. If all distributions are single values then the entire task set is deterministic, which is a special case of the probabilistic representation of real-time task sets. PAnSim supports fixed priority preemptive scheduling (FPPS) policies, but the simulator can easily be extended to dynamic scheduling policies as well as non-preemptive execution. The simulator also has two deadline miss policies: at the instance when a job misses its deadline it can either be allowed to continue or it can be stopped. The decision of stopping of continuing execution past deadlines is system-wide, meaning that the same policy is applied to all jobs that miss their deadlines and to all tasks amd this can not be changed during the simulation.

Listing 1 details how to define, simulate and analyze a probabilistic task set using a preemptive fixed-task priority scheduling policy with PAnSim, and the main functions of the tool are as follows:

*Declaring tasks and task sets:* In PAnSim a task is represented as a cell containing two arrays. The first array is the pWCET distribution and the second array is the pMIT distribution. Each of the two array has two lines of equal size, the first line being the possible values that the respective parameter can exhibit and the second line being the occurrence probabilities of these values. The minimal value of the pMIT distribution is consider to be the deadline of the task. This can be extended to the case of arbitrary

Table I. Probabilistic task set

| task | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\tau_1$ | pWCET | 134 | 137 | 140 | 143 | 146 | 149 | 152 | 155 | 158 | 161 |
| | pMIT | 3565 | 3637 | 3709 | 3781 | 3853 | 3925 | 3997 | 4069 | 4141 | 4213 |
| $\tau_2$ | pWCET | 311 | 318 | 325 | 332 | 339 | 346 | 353 | 360 | 367 | 374 |
| | pMIT | 7784 | 7940 | 8096 | 8252 | 8408 | 8564 | 8720 | 8876 | 9032 | 9188 |
| $\tau_3$ | pWCET | 2879 | 2949 | 3019 | 3089 | 3159 | 3229 | 3299 | 3369 | 3439 | 3509 |
| | pMIT | 26226 | 26751 | 27276 | 27801 | 28326 | 28851 | 29376 | 29901 | 30426 | 30951 |
| $\tau_4$ | pWCET | 5540 | 5675 | 5810 | 5945 | 6080 | 6215 | 6350 | 6485 | 6620 | 6755 |
| | pMIT | 19617 | 20010 | 20403 | 20796 | 21189 | 21582 | 21975 | 22368 | 22761 | 23154 |
| $\tau_5$ | pWCET | 3403 | 3486 | 3569 | 3652 | 3735 | 3818 | 3901 | 3984 | 4067 | 4150 |
| | pMIT | 32313 | 32960 | 33607 | 34254 | 34901 | 35548 | 36195 | 36842 | 37489 | 38136 |

deadlines or even probabilistic deadlines, but for the sake of simplicity we only present the case of implicit deadline here.

Once all tasks have been declared, the task set can be formed by adding all tasks to a cell array. The order in which the tasks appear in the cell array gives the priority ordering of the set, i.e. $taskSet(1)$ has a higher priority than $taskSet(2)$ which has a higher priority than $taskSet(3)$ and so on.

*1) Simulation tool:*

*Simulating the task set:* The functions *simulatePastDeadline* and *simulateStopAtDeadline* are used to perform the simulation of the given task set with two different strategies for deadline misses, i.e. continue the execution of jobs even past their deadlines, respectively abort jobs when they arrive at their deadlines without finishing execution. The simulation returns three outputs. The first output is a cell array containing vectors of observed response times for each task in the set (one vector per task). The other two outputs are used to plot the gantt chart of the run-time execution that was simulated.

The parameter $numberOfJobsToSimulate$ specifies how many consecutive jobs of the task on the lowest priority should be simulated. This is done in order to guarantee that the desired number of traces are obtained. An alternative way is to specify the length of the time interval to be simulated, but this does not guarantee an exact number of traces due to the probabilistic nature of the periods.

*Plotting the results:* The gantt chart of the simulated jobs can be plotted by calling the function *plotExecutionSchedule*. This plotting is meant for small number of jobs in order to obtain intuition from the visual representation of the run-time execution and it is recommended that the function be deactivated (by commenting it) when a large number of jobs is simulated as the gantt chart becomes difficult to read.

*2) Analysis tool:* In order to analyze the task set, the function $probabilisticWorstCaseResponseTime$ needs to be called. This function takes as input four parameters. The first parameter is the task set itself. The second parameters specifies if the analysis should stop once there are no more preemptions that can modify the deadline miss probability (i.e. include only preemptions which arrive before the deadline of the task). The final two parameters specify if re-sampling is performed on the pWET and pMIT respectively. The re-sampling technique used is 'Uniform Spacing' as described in [13] and [1]. When the re-sampling parameter is zero then re-sampling is not performed, otherwise the parameter needs to be an integer $k$ and in this case re-sampling will be performed every time the result of a convolution is larger than $k$.

The output of the analysis is the response time probability distribution of the last task in the input task set. In order to analyze a different task, then the input needs to be given in the form $taskSet(1:k)$ to analyze, e.g., task $k$ in the set. Note that this function assumes that jobs are allowed to continue execution past their respective deadlines, and this is an upper-bound on the case when jobs are dropped at deadline. Also, computing response time distributions in the case when jobs are stopped at deadline is still an open problem.

The obtained theoretical response time distribution can be plotted alongside the empirical distribution of response times observed during simulation in order to compare the two results, as it is exemplified in the final lines of Listing 1. Also, from the theoretical distribution it is possible to extract the deadline miss probability of the task in order to check if it is lower than a given threshold.

PAnSim was used in the experimental section of [1] for the analytical evaluation and in [2] for doing the simulations necessary in the experimental evaluations. PAnSim is freely available online[1] for the benefit of the real-time systems research community.

[1] https://github.com/dorinmaxim/PAnSim-Tool

**Listing 1** Simulation and analysis of a probabilistic task set with PAnSim

```
% Main file to run the PAnSim tool

%% declare task-set.
% Priorities are given in the order in which tasks are placed in the
 ↪   set, e.g. taskSet{1} has the highest priority and taskSet{end}
 ↪   has the lowest priority. The sheduling policy is FP (fixed
 ↪   priority preemptive scheduling)

c5=[3403, 3486, 3569, 3652, 3735, 3818, 3901, 3984, 4067, 4150; 0.1,
 ↪   0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1];
t5=[32313, 32960, 33607, 34254, 34901, 35548, 36195, 36842, 37489,
 ↪   38136 ; 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1];
tau5={c5, t5};

%% Similar way of defining other tasks not repeated here for the
 ↪   sake of conciseness

taskSet={tau1, tau2, tau3, tau4, tau5};

%% perform the simulation
%number of jobs of the lowest priority task to simulate
numberOfJobsToSimulate = 10;

% there are two possible approaches regarding what happens when a
 ↪   job misses a deadline: to continue the job or to abort it. The
 ↪   appropriate function below should be activated while the unused
 ↪   one should be commented

%[responseTimes, grafic, jobs] = simulateContinuePastDeadline(
 ↪   taskSet, numberOfJobsToSimulate );
[responseTimes, grafic, jobs] = simulateStopAtDeadline( taskSet,
 ↪   numberOfJobsToSimulate);


%% plot the execution schedule; comment the next line to disable the
 ↪   gantt chart
plotExecutionSchedule(jobs, grafic)

%% analyze the task-set
stopAnalysisWhenDeadlineIsReached=1;
pWCETresampling = 0;
pMITresampling = 0;
pWCRT = probabilisticWorstCaseResponseTime(taskSet,
 ↪   stopAnalysisWhenDeadlineIsReached, pWCETresampling,
 ↪   pMITresampling);

%% plot the results for the least prioritary task in the set
% plot theoretical distribution in 1-CDF form
uCDF=unuMinusCDF(pWCRT);
figure; hold all;
plot(uCDF(1,:),uCDF(2,:));

% plot empirical distribution observed during simulation
plotEmpiricalDistribution(responseTimes{end})
```

### B. Probabilistic SimSo

We propose here a probabilistic extension for the real-time simulator SimSo [7] (short for Simulation of Multiprocessor Scheduling with Overheads).

*1) SimSo:* is an open-source multiprocessor simulator of real-time scheduling written in Python released under the CeCILL licence. It provides the main task generators, uniprocessor and multiprocessor scheduling policies and a convenient and concise way to specify new scheduling policies. SimSo also integrates different cache models to evaluate the overhead due to cache misses. Jobs that miss their deadlines can be stopped or allowed to continue (with the abort_on_miss boolean option in Listing 2) their execution and then eventually postpone further jobs. SimSo relies on a discrete-event simulation which allows it to deal with long intervals of simulation. SimSo is divided between the core and the graphical user interface (GUI). The GUI adorns the core with a display of the gantt chart and scheduling results of the simulation [2]. A scheduling configuration (task set, scheduler, etc.) can be described through an XML file, but it can also be directly written in Python using the core component. The separation of the core from the graphical user interface allows SimSo to be called and run as a Python module in a script which is very useful to process the scheduling results. In this regard, SimSo interestingly keeps record of a very complete set of scheduling metrics, such as job response times, number of context-switches, number of preemptions, job migration, etc. that are meaningful information for a later evaluation.

Beyond its very practical usage, SimSo relies on a modular architecture which makes it possible to provide extensions for it, such as the probabilistic extensions that we propose in this paper. In the following, we detail the major additions that have been made to extend the core component of SimSo from a deterministic to a probabilistic behaviour.

*2) Probabilistic extension:*

*a) pWCET:* We added the possibility to use pWCET by using the concept of SimSo called Execution Time Models (ETM). An ETM is a class that determines the duration of the jobs during the simulation. By default, it can deal with WCET value but it can also dynamically change the execution time of a task during the simulation, taking into account some time overhead (e.g timing penalties coming from cache misses). We implemented a new ETM that draws at each execution the WCET from the pWCET distribution defined.

*b) pMIT:* SimSo allows to define periodic but also sporadic behaviour of tasks by describing the arrival time instants in the configuration files in the interval of simulation chosen. We used this feature to add a step to the parser to generate a set of arrivals drawn from the pMIT defined in the configuration files.

Listing 2 details how to describe and simulate a probabilistic task set using a preemptive fixed-task priority scheduling policy (here Deadline Monotonic). The probabilistic parameters are defined by a vector of tuples corresponding to each random variable and its probability of appearance. The other parameters of the simulation are similar to those of the original SimSo tool and their use is straightforward and commented in the Listing 2. In particular, the logs object contains a set of scheduling metrics as the job response times. The complete set of the available metrics can be found on the SimSo documentation. The whole configuration file can be simply executed as a Python script. The probabilistic extension is fully compatible with the features (scheduling policies, performance metrics, etc.) of the standard core version of SimSo. In this respect, the extension benefits from the documentation of SimSo.

The probabilistic extension of SimSo is freely available online[3] and was recently used in [14].

[2] A simpler and intuitive web version of SimSo is also available online http://projects.laas.fr/simso/simso-web which is especially useful for educational purposes

[3] https://github.com/abertout/simso

**Listing 2** Simulation of a probabilistic task set with the extension of SimSo

```
import sys
from simso.core import Model
from simso.configuration import Configuration


def main(argv):

    configuration = Configuration()

    configuration.cycles_per_ms = 1

    configuration.etm = "pwcet"

    configuration.duration = 381360 * configuration.cycles_per_ms

    # Add tasks:

    configuration.add_task(name="t5", identifier=4,
     ↪   activation_date=0,
     ↪   pwcet=[(3403,0.1),(3486,0.1),(3569,0.1),(3652,0.1),
(3735,0.1),(3818,0.1),(3901,0.1),(3984,0.1),(4067,0.1),(4150,0.1)],
pmit=[(32313,0.1),(32960,0.1),(33607,0.1),(34254,0.1),(34901,0.1),
(35548,0.1),(36195,0.1),(36842,0.1),(37489,0.1),(38136,0.1)],
deadline=32313,
task_type = "Probabilistic",
abort_on_miss=False)

    # Similar way of defining other tasks not repeated here for the
     ↪   sake of conciseness

    # Add a processor:
    configuration.add_processor(name="CPU 1", identifier=1)

    # Add a scheduler:
    configuration.scheduler_info.filename =
     ↪   "../simso/schedulers/DM_mono.py"

    # Check the config before trying to run it.
    configuration.check_all()

    # Init a model from the configuration.
    model = Model(configuration)

    # Execute the simulation.
    model.run_model()

    # Print logs.
    for log in model.logs:
        print(log)

main(sys.argv)
```

## IV. Comparison between the two simulators

In this section, we compare both tools over their modularity and extensibility, their environment and their functionality.

*Modularity:* The extension of SimSo is based upon a powerful and modular simulator. For example, the scheduler is an option in the configuration file and there is no need to modify any line of code related to other functionality of SimSo to implement a new one. Adding a scheduling policy (PanSim implements fixed-task priority preemptive scheduling) in PAnSim is not a burden but requires to modify the core source code.

*Environment:* PanSim is written in MATLAB language and take advantages of the MATLAB environment. The latter proposes an interactive system that comes with the ability of plotting functions and data. This is particularly useful to compare probabilistic distribution. SimSo benefits from the large support of Python third party libraries (e.g numpy, matplotlib, etc.) but representing data is less straightforward.

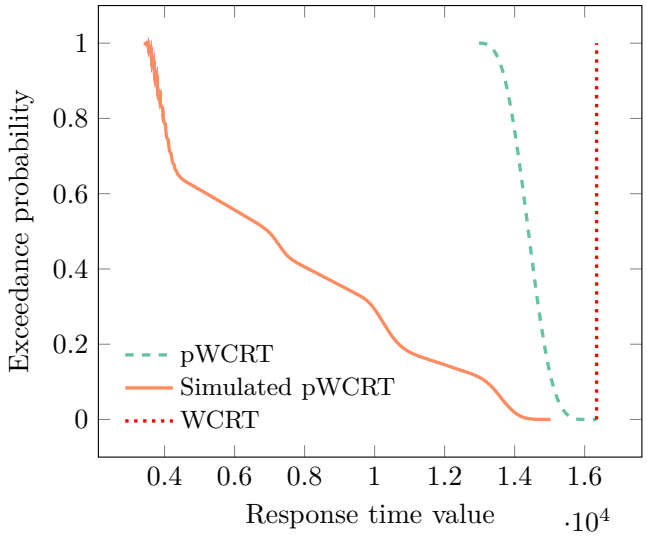


Figure 1. Response time distributions of $\tau_5$ obtained in three different ways

*Functionality:* While SimSo is a general purpose simulator PanSim goes along with an analysis tool. The first has the advantage to provide a rich set of scheduling metrics of the simulations but the latter provides analytical distributions which may be useful for comparison purpose. PAnSim also does the analysis of the input task set. When the pMITs are single-values, then the analysis is reduced to the Diaz analysis presented in [3]. If both pWCET and pMIT are deterministic the analysis returns the same result as the state of the art deterministic worst-case response time analysis introduced by [12]. With both tools, defining and simulating a probabilistic task set is concise as shown in Listings 1 and Listing 2 for respectively PAnSim and SimSo.

The main differences between the two proposed simulators are summarized in Table II and in the rest of the paper we show the type of results that the tools can produce.

Table II. Main differences between the proposed tools

| | **PAnSim** | **pSimSo** |
|---|---|---|
| *Language* | MATLAB | Python |
| *Scheduling policies* | FPPS | Several priority schemes |
| *Performs analysis* | Yes | No |

*Joint usage:* We illustrate the possibilities offered by the joint usage of SimSo and PAnSim in Figure 1 and in Figure 2.

Figure 1 depicts the response times of task $\tau_5$ (the task on the lowest priority in Table I) obtained in three different ways:

- the dashed curve represents in the form of inverse cumulative distribution function (1-CDF) the probabilistic worst-case response times of the task in the synchronous case (i.e. worst-case conditions)

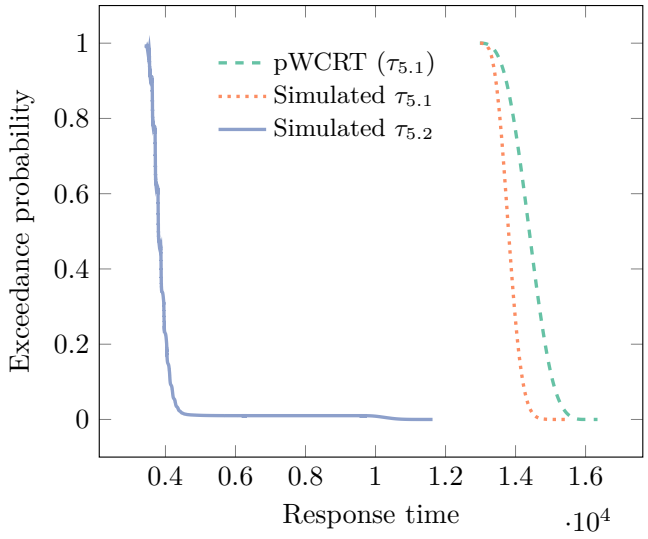


Figure 2. Worst-case response time distributions of $\tau_5$

computed using the probabilistic response time analysis of Maxim and Cucu [1] implemented in PAnSim.

- the solid line is an empirical 1-CDF distribution obtained by simulating the probabilistic task set and recording the observed response times for $5 \cdot 10^5$ consecutive jobs of $\tau_5$ using the probabilistic extension of SimSo.
- for completeness, we have also added the deterministic worst-case response time obtained using the analysis in [12]. We may note that this value is the largest of the probabilistic response time distribution and represented as a dotted vertical line.

We used the probabilistic extension of SimSo in a different way in Figure 2. We simulated $10^6$ times the execution of the two first instances of $\tau_5$, i.e $\tau_{5.1}$ and $\tau_{5.2}$ to obtain their empirical worst-case response time distribution respectively represented as a solid curve and as a dotted curve. For comparison purpose, we also depicted the analytical probabilistic worst-case response times of the task in the synchronous case (first job). As expected, we can observe that the pWCRT computed with the analysis is an upper bound on the simulated pWCRT of the two first jobs of $\tau_5$. Interestingly, the simulation provides an approximation of any job pWCRT whose no theoretical analysis yet exists [14].

## V. Conclusion

In this paper, we presented an extension of SimSo, a well known academic tool, and PAnSim, an in-house MATLAB implementation. These two tools are dedicated to the simulation and analysis of probabilistic task sets. We presented their characteristics and how they can be used, with strengths and weaknesses for each one of them. Another objective of this paper is to make the tools available to the real-time systems research community. Many advancements and improvements can be made on each of the tools, such as adding new scheduling policies to PAnSim or adding the probabilistic interface to the graphical version of SimSo. The community is invited to participate in the improvement of these tools, either directly, e.g. by adding extensions and new implementations, or indirectly, by giving feedback to the authors of the tools.

## Acknowledgements

This work was partially supported by the RETINA Eurostars Project E10171.

## References

[1] D. Maxim and L. Cucu-Grosjean, "Response time analysis for fixed-priority tasks with multiple probabilistic parameters," in *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*. IEEE, 2013, pp. 224–235.

[2] D. Maxim, F. Soboczenski, I. Bate, and E. Tovar, "Study of the reliability of statistical timing analysis for real-time systems," in *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, ser. RTNS '15. New York, NY, USA: ACM, 2015, pp. 55–64. [Online]. Available: http://doi.acm.org/10.1145/2834848.2834878

[3] J. Diaz, D. Garcia, K. Kim, C.-G. Lee, L. Lo Bello, J. Lopez, S. L. Min, and O. Mirabella, "Stochastic analysis of periodic real-time systems," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02), Austin, Texas, USA, December 3-5, 2002*, 2002, pp. 289–300.

[4] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quinones, and F. J. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs," in *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*. IEEE, 2012, pp. 91–101.

[5] M. G. Harbour, J. G. García, J. P. Gutiérrez, and J. D. Moyano, "Mast: Modeling and analysis suite for real time applications," in *Real-Time Systems, 13th Euromicro Conference on, 2001.* IEEE, 2001, pp. 125–134.

[6] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: a flexible real time scheduling framework," in *ACM SIGAda Ada Letters*, vol. 24, no. 4. ACM, 2004, pp. 1–8.

[7] M. Chéramy, P.-E. Hladik, and A.-M. Déplanche, "Simso: A simulation tool to evaluate real-time multiprocessor scheduling algorithms," in *Proc. of the 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, ser. WATERS, 2014.

[8] J. Diemer, P. Axer, and R. Ernst, "Compositional performance analysis in python with pycpa," *Proc. of WATERS*, 2012.

[9] "Cpal," https://www.designcps.com/, accessed: 2017-04-14.

[10] L. Palopoli, G. Lipari, G. Lamastra, L. Abeni, G. Bolognini, and P. Ancilotti, "An object-oriented tool for simulating distributed real-time control systems," *Software: Practice and Experience*, vol. 32, no. 9, pp. 907–932, 2002.

[11] L. Cucu-Grosjean, "Independence-a misunderstood property of and for probabilistic real-time systems," *In Real-Time Systems: the past, the present and the future*, pp. 29–37, 2013.

[12] M. Joseph and P. K. Pandya, "Finding response times in a real-time system," *Comput. J.*, vol. 29, no. 5, pp. 390–395, 1986.

[13] D. Maxim, M. Houston, L. Santinelli, G. Bernat, R. I. Davis, and L. Cucu-Grosjean, "Re-sampling for statistical timing analysis of real-time systems," in *Proceedings of the 20th International Conference on Real-Time and Network Systems*, ser. RTNS '12. New York, NY, USA: ACM, 2012, pp. 111–120. [Online]. Available: http://doi.acm.org/10.1145/2392987.2393001

[14] A. Bertout, D. Maxim, and L. Cucu-Grosjean, "Average probabilistic response time analysis of tasks with multiple probabilistic parameters," in *Real-Time Systems Symposium (RTSS), 2016 IEEE*. IEEE, 2016, pp. 367–367.