```
================================================================
GMPR Generator v1.0
================================================================
```
These MATLAB routines provide functionalities to generate GMPR and MPR interfaces of a real-time tasks set. The implemented algorithms are described in the paper "The Generalized Multiprocessor Periodic Resource Interface Model for Hierarchical Multiprocessor Scheduling" by A. Burmyakov, E. Bini, E. Tovar, submitted to the 20th International Conference on Real-Time and Network Systems (RTNS) 2012.

This software package includes the functionality to generate GMPR and MPR interfaces as well as an implementation of a few simulation scenarios. The description of the main functions and the usage instructions are specified below.

This software is developed by Artem Burmyakov and Enrico Bini and is freely available at https://sites. google.com/site/artemburmyakov/home/papers.

```
================================================================
MAIN FUNCTIONALITIES:
================================================================
```

```
gmprInterfaces = compute_gmpr(Pi, m, taskSet, schedPolicy)
```

Computes a list of GMPR interfaces for a task set taskSet locally scheduled by a scheduling policy schedPolicy. The period and the maximum parallelism of interfaces are fixed: Pi and m. Computation is performed based on the PSF-based schedulability test, proposed by Bini [1].

Input data:
- Pi - interface period, a positive integer;
- m - maximum parallelism of an interface, a positive integer;
- taskSet - a set of real-time tasks; to generate a random task set, use generate_taskset(U, Umax, Tmaxmin) function; to define a particular task set, use sample_taskset() function;
- schedPolicy - local scheduling policy of a task set, which is "GEDF" or "FP".

Output data:
- gmprInterfaces - an array of GMPR interfaces, where each interface is characterized by two fields:
  - .Pi - the period,
  - .thetas - an array of supplies {\Theta_k}_{k=1}^m.

```
mprInterfaces = compute_mpr(Pi, m, taskSet, schedPolicy)
```

Computes MPR interfaces for a task set taskSet locally scheduled by a scheduling policy schedPolicy. The period and the maximum parallelism of interfaces are fixed: Pi and m. Computation is performed based on the PSF-based schedulability test, proposed by Bini [1].

Input data:
- Pi - interface period, a positive integer;
- m - maximum parallelism of an interface, a positive integer;
- taskSet - a set of real-time tasks; to generate a random task set, use generate_taskset(U, Umax, Tmaxmin) function; to define a particular task set, use sample_taskset() function;
- schedPolicy - local scheduling policy of a task set, which is "GEDF" or "FP".

Output data:
- mprInterfaces - an array of MPR interfaces, where each interface is characterized by three fields:

.Pi - the period,
.theta - the resource value,
.m - the maximum parallelism.

tasks = generate_taskset(U, Umax, Tmaxmin)

Randomly generates a real-time periodic task set (D=T) with an overall utilization U. An individual task utilization does not exceed Umax; a ratio Tmax/Tmin does not exceed Tmaxmin, where Tmax and Tmin are the maximum and minimum individual task periods within a task set (see Note).

Note: Tmin is randomly extracted from the range [20;40], and Tmax is computed as ceil (Tmin*Tmaxmin).

Input data:
	U - an overall task set utilization, a real value;
	Umax - a maximum individual task utilization within a task set, a real value not greater than 1;
	Tmaxmin - a ratio Tmax/Tmin, where Tmax and Tmin are the maximum and minimum individual task periods within a task set (see Note).

Output data:
	tasks - a set of real-time tasks. The set is represented as a Matlab structured array of the following format: struct('C', 'T', 'D', 'Wedf', 'Wfp', 'P'), where:
		C - execution time of a task;
		T - period of a task;
		D - deadline of a task (D = T);
		Wedf - worst-case interfering workload on a task according to Bertogna [2] in case of GEDF;
		Wfp - worst-case interfering workload on a task according to Bertogna [2] in case of FP.

tasks = sample_taskset()

Specification of a sample task set. Specify the desired parameters of a task set directly in the m-file following a predefined pattern.

Input data:
None

Output data:
	tasks - a set of real-time tasks. The set is represented as a Matlab structured array of the following format: struct('C', 'T', 'D', 'Wedf', 'Wfp', 'P'), where:
		C - execution time of a task;
		T - period of a task;
		D - deadline of a task (D = T);
		Wedf - worst-case interfering workload on a task according to Bertogna [2] in case of GEDF;
		Wfp - worst-case interfering workload on a task according to Bertogna [2] in case of FP;
		P - a fixed priority of a task (FP).

tasks = get_interfering_workloads(tasks)

Computes interfering workloads for each task from a task set tasks for both GEDF and FP scheduling policies.

Input data:

```
        tasks - a set of real-time tasks; to generate a random task set, use
                generate_taskset(U, Umax, Tmaxmin) function; to define a particular task set,
                use sample_taskset() function.

Output data:
        tasks - a set of real-time tasks; resulted interfering workloads are assigned to .Wedf
                (for EDF scheduler) and .Wfp (for FP scheduler) fields.



isSchedulable = check_gmpr_candidate_Bini2009(gmprInterface, taskSet, schedPolicy)

Implements the PSF-based schedulability test, proposed by Bini [1]. Checks the schedulability
of a task set taskSet under the scheduling policy schedPolicy over a GMPR interface
gmprInterface according.

Input data:
        gmprInterface - GMPR interface; to compute a GMPR interface, use
                        compute_gmpr(Pi, m, taskSet, schedPolicy);
        taskSet - a set of real-time tasks; to generate a random task set, use
                generate_taskset(U, Umax, Tmaxmin) function; to define a particular task set,
                use sample_taskset() function;
        schedPolicy - local scheduling policy of a task set, which is "GEDF" or "FP".

Output data:
        isSchedulable - the result of the test, where "1" means "schedulable" and "0" is
                        "unschedulable".



S_thetas = get_thetas_search_space(taskSet, Pi, m, schedPolicy)

Computes a search space for feasible GMPR interfaces S_thetas.

Input data:
        taskSet - a set of real-time tasks; to generate a random task set, use
                generate_taskset(U, Umax, Tmaxmin) function; to define a particular task set,
                use sample_taskset() function;
        Pi - interface period, a positive integer;
        m - maximum parallelism of an interface, a positive integer;
        schedPolicy - local scheduling policy of a task set, which is "GEDF" or "FP".

Output data:
        S_thetas - a set of vectors, lower-bounding the search space for GMPR interfaces. The
                data format is the Matlab cell array.


inSearchSpace = are_thetas_in_search_space(thetas, searchSpace)

Checks if the GMPR interface with specified supplies {\Theta_k}_{k=1}^m is in the search space
searchSpace.

Input data:
        thetas - an array of supplies {\Theta_k}_{k=1}^m of a GMPR interface;
        searchSpace - the search space for GMPR interfaces; to compute searchSpace, use
                get_thetas_search_space(taskSet, Pi, m, schedPolicy) function;

Output data:
        inSearchSpace - the results of the test, where "1" means thetas are in the search
                space, and "0" means that thetas are outside the search space and can be
                ignored.
```

psfKValue = gmpr_psf_k(k, thetas, deltaT, Pi)

Computes the value of the PSF function Y_k(deltaT) for a GMPR interface with supplies thetas and a period Pi.

Input data:
	k - level of the PSF function, integer;
	thetas - supplies of a GMPR interface, an array of integer values;
	deltaT - the length of a time interval to compute the PSF function;

Output data:
	psfKValue - the value of the PSF function Y_k(deltaT).

==
SIMULATION SCENARIOS:
==

simulations_varing_Pi_and_m(tasks, PiMin, PiMax, mMax, schedPolicy)

For a task set tasks the function generates GMPR and MPR interfaces for each Pi in the range [PiMin;PiMax]. The maximum parallelism of interfaces is limited to mMax. The results are displayed on a plot as a dependency of interface utilizations on period Pi.

Input data:
	tasks - a set of real-time tasks; to generate a random task set, use
		generate_taskset(U, Umax, Tmaxmin) function; to define a particular task set,
		use sample_taskset() function;
	PiMin - minimum value of a period Pi, integer;
	PiMax - maximum value of a period Pi, integer;
	mMax - maximum parallelism of interfaces, integer;
	schedPolicy - the local scheduling policy for a task set tasks, which is "GEDF" or
		"FP"; if schedPolicy is not specified, then computations are performed for
		both.

Output data:
The results are displayed on a plot as a dependency of interface utilizations on period Pi.

simulations_fixed_m_varing_Pi(PiMin, PiMax, U, Umax, Tmaxmin, mFixed, simulationsNumber, schedPolicy)

Randomly generates periodic task sets using generate_taskset(U, Umax, Tmaxmin). For a generated task set computes GMPR and MPR interfaces for each Pi in the range [PiMin;PiMax] with a fixed parallelism mFixed. The number of simulations is simulationsNumber. The local scheduling policy schedPolicy for task sets is "GEDF" or "FP"; in case no scheduling policy is specified, computations are performed for both cases. The results are displayed on a plot as a dependency of interface utilizations on period Pi.

Note: If no interfaces exist for a generated task set with a parallelism mFixed, then this task set is regenerated.

Input data:
	PiMin - minimum period Pi, integer;
	PiMax - maximum period Pi, integer;
	U - an overall utilization of a task set to be generated, real;
	Umax - a maximum utilization of an individual task, real;
	mFixed - parallelism of interfaces (fixed), integer;
	simulationsNumber - number of different task sets to be generated;
	schedPolicy - the local scheduling policy for a task set tasks, which is "GEDF" or
		"FP"; if schedPolicy is not specified, then computations are performed for
		both.

Output data:
        The results are displayed on a plot as a dependency of interface utilizations on
        period Pi.

[1] Enrico Bini, Marko Bertogna, and Sanjoy Baruah. Virtual multiprocessor platforms:
        Specification and use. In Proceed- ings of the 30th IEEE Real-Time Systems Symposium,
        pages 437–446, Washinghton, DC, USA, December 2009.

[2] Marko Bertogna, Michele Cirinei, and Giuseppe Lipari. Schedulability analysis of global
        scheduling algorithms on multiprocessor platforms. IEEE Transactions on Parallel and
        Distributed Systems, 20(4):553–566, April 2009.

END