# ARTIFACT EVALUATION INSTRUCTIONS
# IMPACT: INTERVAL MDP PARALLEL CONSTRUCTION FOR CONTROLLER SYNTHESIS OF LARGE-SCALE STOCHASTIC SYSTEMS

**BEN WOODING AND ABOLFAZL LAVAEI**[1]

[1]SCHOOL OF COMPUTING, NEWCASTLE UNIVERSITY, UK

{BEN.WOODING,ABOLFAZL.LAVAEI}@NEWCASTLE.AC.UK

Welcome to the IMPaCT repeatability instructions. This document explains how to reproduce the results presented in the paper "IMPaCT: **I**nterval **M**DP **Pa**rallel Construction for **C**ontroller Synthesis of Large-Scale S**T**ochastic Systems". The artifact to be evaluated can be found at:

https://github.com/Kiguli/IMPaCT

## 1. System Requirements

We recommend evaluating the artifact either on Linux using the shell script we provided, or by using the Docker image found in the packages of the repository. It is possible to also run the tool on *older* MacOS with an *Intel CPU*, but this installation is more challenging and GPU is not supported inside of the AdaptiveCPP dependency.

Table 1 and Table 2 from the paper requires the repeatability evaluation. If a member of the committee wishes to reproduce the figures from the Appendix, MATLAB was used to visualize a trace of the controlled system, thus requiring it to be installed as well.

*Considering the significant size of some case studies and relying on the hardware available to the AE committee, some examples might not be replicable due to considerable memory requirements and synthesis times, if the committee's hardware capacity is restricted. For example, we used a powerful high performance computer for Table* 1 *with* 2 *AMD EPYC* 7702 64-*Core along with* 2*TB RAM. The larger* 3-*dimension vehicle model required nearly* 250 *GB of memory to store all the matrix information, and required* 2.62 *hours of computation time for the construction of the matrices and then running the sorted approach "b" in Table* 1.

*Many of the examples should be completely feasible on even a less powerful personal computers such as the examples* `ex_2Drobot-R` *which requires in total only* 368 *MB and taking less than* 15 *seconds to run the construction and synthesis via sorting, "b" in Table* 1. *We recommend to the committee to use this* `ex_2Drobot-R` *example as a benchmark to compare the performance of their machine against the HPC we used to see how many of the case studies they can replicate.*

## 2. Installation

2.1. **Manual Installation.** We have personally got the tool installed and working on both Linux (Ubuntu 22.04) and MacOS (Intel CPU). We have provided a shell script for ease of installation on Ubuntu (tested on Ubuntu 22.04) that can be run from the terminal using the following commands:

```
chmod +x install_ubuntu22.sh
sudo ./install_ubuntu22.sh
```

*We have also provided a YouTube tutorial video for this installation and running a simple example:* here.

To the best of our understanding, there are compatibility issues with ARM architecture and AdaptiveCPP (the parallelism dependency used by IMPaCT). This means newer Apple M1 and M2 Macs are unsupported. We are unsure about support for using our tool on Windows as it is described by AdaptiveCPP as experimental here. For the AE committee using these systems, we hope the Docker package can facilitate the evaluation of the artifact.

2.2. **Docker Installation.** We also offer a Dockerfile with IMPaCT pre-installed, available for download from the packages section. This allows users to utilize the tool on different operating systems. We assume the user already has Docker installed, if not follow the instructions here.

Extensive details for the installation and running of the examples on the Docker image can be found in the repository markdown file: `Docker_instructions.md`. These instructions also include details of mounting a volume to pass files between the PC and the Docker image, including how to retrieve the results from the Docker image to the host machine (*e.g.,* for plotting the examples using MATLAB).

2.3. **Artifact Evaluation via GPU.** IMPaCT supports the parallel implementation using GPU as well as CPU. However, the installation is dependent on the user's specific hardware. We therefore refer the committee to the AdaptiveCPP installation instructions to find the appropriate tags to include in their installation instructions: here. *AdaptiveCPP does try to automatically enable all backends that it finds on the host PC.* If not, the installation lines for AdaptiveCPP (based on installing AdaptiveCPP from the zip file included in the repository) are as follows:

```
unzip AdaptiveCpp-develop.zip
cd AdaptiveCpp-develop
mkdir build && cd build
```

```
cmake <tags> ..
make install
```

should be updated so `<tags>` is replaced by *e.g.* `-DCUDA_TOOLKIT_ROOT_DIR=/path/to/cuda`. Additional dependencies such as LLVM, CUDA, or ROCm may also be required depending on the host PC architecture, again we refer to their installation instructions.

For Docker, the GPU may need to be passed into the Docker image; we do not provide the details for how to do this but refer to links such as: here.

## 3. Running the Examples

*As previously mentioned, we demonstrated in a YouTube tutorial video here (from 4:00), how to run the example* `ex_2Drobot-R-U`.

### 3.1. **Examples over CPU.**

In general, to run the examples simply navigate to respective folder of the example and run the following commands in the terminal:

```
cd ./examples/<example>/
make
./<example>
```

To then get a single trace used in the figures, run the MATLAB file `plot.m`.

There may be lots of debug warnings that appear; these do not affect the outcome of the results, but can clutter the terminal and may be removed before running the example with the command:

```
export ACPP_DEBUG_LEVEL=0
```

### 3.2. **Alternate Synthesis Algorithm.**

In Table 1 of the paper, there are two different synthesis algorithm times recorded. By default, all the examples in the folder `examples` use the GLPK algorithm. We provide also a folder `sorted-examples` which uses the sorted algorithm. The difference in implementation between these is as follows. Inside of the respective example configuration file `example.cpp`; the GLPK controller synthesis command *e.g.,* `mdp.infiniteHorizonReachController(true)` is appended with the word `Sorted` *e.g.* `mdp.infiniteHorizonReachControllerSorted(true)` for the sorted algorithm. When reproducing the examples for the examples marked with an asterisk for Table 2 where a finite horizon was used, the example file should be edited to change the word `infinite` to `finite`, and a second argument for the number of time intervals should be added e.g. `mdp.finiteHorizonReachController(true,10)` for 10 time steps.

*We demonstrate this and other ways to edit the configuration files in another tutorial YouTube video: here.*

3.3. **Dealing with Absorbing States.** For some of the safety verification case studies marked with an asterisk in Table 1, the tool detects the presence of absorbing states (see Remark 2 in the paper). To obtain the synthesis times for these case studies, we use the larger of the two outputted time steps, denoted as `n`, that the tool recommends when running `infiniteHorizonSafeController(true)`, we then run the command `finiteHorizonSafeController(true, n)` using those `n` steps. These synthesis times are summed together in Table 1.

3.4. **Examples over GPU.** To run examples over the GPU, some changes need to be made to both the configuration file and the Makefile. In particular, this is due to GPU not being able to handle external function commands, see Remark 4 in the paper.

In particular, the functions for constructing the matrices should be removed from the configuration file, and the matrices should be loaded from the HDF5 files, additionally only the sorting approach algorithm should be used, see `ex_GPU` for an example. The Makefile also needs to be modified to ensure that `--acpp-targets` flag specifies the GPU architecture, *e.g.,* `cuda:sm_80`. Furthermore, in the Makefile, `IMDP.cpp` should be substituted with `GPU_synthesis.cpp`.

A suggested way to run the examples over GPU is to copy the respective `*.h5` files of the example you wish to run into the folder `ex_GPU` and then adjust the `GPU.cpp` file inside of the folder with the correct number of state dimensions `dim_x = ...;`, input dimensions `dim_u = ...;` and disturbance dimensions `dim_w = ...;`. These can be found in the original example folder's `*.cpp` file.

*We have provided two YouTube tutorial videos explaining how to adjust the configuration files and the Makefile here and here, respectively.*

## 4. Extending IMPaCT with new examples

Writing new examples for IMPaCT is fairly straightforward, simply by editing a similar example file. It is also straightforward to create any brand new examples.

*We have provided two YouTube tutorial videos explaining how to adjust the configuration files and the Makefile here and here, respectively. We have also provided the markdown file* `setup.md` *which explains the same details.*