ARTIFACT EVALUATION INSTRUCTIONS
PROTECT: <u>P</u>ARALLELIZED CONST<u>R</u>UCTION <u>O</u>F SAFE<u>T</u>Y BARRI<u>E</u>R <u>C</u>ERTIFICATES FOR NONLINEAR POLYNOMIAL SYS<u>T</u>EMS

BEN WOODING, VIACHESLAV HORBANOV, AND ABOLFAZL LAVAEI[1]

[1]SCHOOL OF COMPUTING, NEWCASTLE UNIVERSITY, UNITED KINGDOM

{BEN.WOODING,V.HORBANOV2,ABOLFAZL.LAVAEI}@NEWCASTLE.AC.UK

Welcome to the PRoTECT artifact evaluation instructions. This document explains how to reproduce the results presented in the paper "**PRoTECT**: **<u>P</u>**arallelized Const**<u>R</u>**uction **<u>o</u>**f Safe**<u>T</u>**y Barri**<u>E</u>**r **<u>C</u>**ertificates for Nonlinear Polynomial Sys**<u>T</u>**ems". The artifact to be evaluated can be found at:

https://github.com/Kiguli/PRoTECT

This artifact is also assigned a permanent DOI, which can be accessed here on Zenodo. We have also provided some detailed Youtube tutorial videos which explain how to install the tool on Linux (the tool is also viable on MacOS and Windows), how to use the GUI for different case studies of the four types of dynamical systems, and how to edit the example configuration python scripts to use the tool as an application programming interface (API). We expect the reviewers only needs to read Sections $1-4$ of this guide, Sections $5-6$ include manual instructions if any issues occur, and Section 7 contains reference Tables 1 and 2 but using the CVXOPT solver (although the Mosek solver is strongly advised).

1. System Requirements

Since our tool offers a GUI, we recommend using the Linux Virtual Machine (VM), based on Ubuntu 22.04 LTS, provided by the Artifact Evaluation committee that can be downloaded here on Zenodo. We got this to run easily on VirtualBox installed on the host PC and double clicking the `Artifact_VM_Ubuntu_22.04.ova` file from the download which auto-loaded itself into VirtualBox. We ran all the examples on a machine using an Intel i9-12900 with 24 CPUs, 33GB memory and 8.6GB of swap memory. Machines with less CPUs and memory should still be able to run PRoTECT smoothly, in general each parallel run will use one CPU per degree value, the memory required grows exponentially with the degree value and number of dimensions. We estimate the largest example ($\texttt{hi\_ord}_8$) requires up to 5GB RAM if using Mosek.

Our tool is installable on Linux, Windows, and MacOS, and we have personally installed it on each of these operating systems. For the purposes of this artifact evaluation, we assume the reviewer is using the above mentioned VM, and provide detailed instructions especially for this machine. The repository contains more general instructions for how to install the tool on any operating system. We have also included a Dockerfile, although it does not provide GUI support and so is likely unnecessary for this artifact evaluation.

Table 1 and Table 2 from the paper require the artifact evaluation. We will provide the instructions for how to install our tool PRoTECT, as well as how to install the tool we compare against called FOSSIL. We anticipate that running all the case studies may take up to 2 hours. We strongly advise the reviewers to download the Mosek license used as the solver for the results in Table 1 and 2. Although an open source solver CVXOPT is also available, it is noticeably slower and requires significantly more memory. If it is really not possible to use the Mosek license, we provide the majority of the results using the CVXOPT solver in Section 7.

Table 1 requires utilizing the tool in both serial and parallel modes, and we will furnish instructions detailing the minor adjustments needed in the files to enable execution in each setting, as well as shell scripts which automatically adjust these parameters for all the examples. Similarly, Table 2 demonstrates running PRoTECT with and without optimization, and we offer instructions for adjusting the settings accordingly in each case, and a shell script for automatically changing the examples.

We present two methods for installation, an automatic version via shell script in Section 2 and a manual installation in Section 5. The FOSSIL release 2.0 has been included in the repository in a folder named fossil-main which was used for comparison with our tool.

2. Automatic Installation

We assume the user is using the VM mentioned previously, it has username `artifact` and password (required for sudo commands) `artifact`.

Download the Zenodo zip file for PRoTECT v1.2 to the home directory, then unzip it and change the folder name to PRoTECT, or alternatively clone the v1.2 branch of the PRoTECT GitHub page which is linked to this same Zenodo DOI from the home directory in a terminal using the commands:

```
cd ~
sudo apt-get install git
git clone https://github.com/Kiguli/PRoTECT --branch v1.2
```

We now navigate to the folder bash-scripts inside the folder PRoTECT we just added and run the shell script that will install PRoTECT and FOSSIL onto the VM, other steps will be completed automatically:

```
cd ~/PRoTECT/bash-scripts
./install_ubuntu22_PRoTECT_and_FOSSIL.sh
```

It is important not to run the last command with sudo, the commands requiring sudo are defined inside this script and it will still request the sudo password from you running it this way.

If you only want to install PRoTECT and not FOSSIL, instead of the last line run:

```
./install_ubuntu22_PRoTECT.sh
```

2.1. **Get Mosek licence.** For the results in Tables 1 and 2 the solver Mosek was used, therefore we now install the licence for Mosek (for academics this is free, and there is also a free trial if necessary).

Fill in your details to get a Mosek licence at https://www.mosek.com/license/request/?i=acp. The licence file will be emailed to you with instructions of where to place the file in your home directory. On the VM this will be adding a folder `mosek` to the home directory and add the licence file `mosek.lic` into this directory.

If for whatever reason, the Mosek licence is not available for the reviewer to get, there is an open-source solver CVXOPT which can also be used, in all the example files the line `'solver': "mosek",` in the dictionary `fixed_params` should be changed to `'solver': "cvxopt"`. The table results might be a little different to the solutions that used Mosek, see Section 7 for the CVXOPT tables. The following script run from inside the folder `bash-scripts` can be used to change all examples from the default `mosek` to `cvxopt`:

```
./all_examples_to_cvxopt.sh
```

Similarly, they can be changed back to `"mosek"` using:

```
./all_examples_to_mosek.sh
```

2.2. **Docker.** We have included a Dockerfile inside the repository which can be used to run Table 1 and Table 2 via the terminal. However, the Docker image will not be able to support the GUI. The Docker image is also defaulted to the CVXOPT solver, and the Mosek license would need to be copied into the Docker image in order to use the Mosek solver. To build and run the Docker image, navigate to the folder containing `Dockerfile` and then run:

```
docker build -t protect .
docker run --rm -it --name protect protect
```

To copy the Mosek license into the Docker image you can adapt the following command:

```
docker cp <license-file-on-host-machine> protect:<location-for-license-on-Docker-image>
```

3. Running the Examples

After the previous installation instructions have been setup correctly, it should be straightforward to run all the examples that make up Tables 1 and 2. The results for these tables were acquired by running the python scripts in the folder `ex` and not from using the GUI. This is in part because the GUI provides an overhead that somewhat reduces the efficiency of the functions being called, and secondly it provides a fairer comparison against FOSSIL which does not use a GUI.

3.1. **Smoke Test.**

To run the smoke test to check that PRoTECT and FOSSIL are both installed correctly navigate first to the folder:

```
cd ~/PRoTECT/ex/benchmarks-deterministic/PRoTECT-versions/
```

and run in the terminal:

```
python3 ex1_dt_DS.py
```

You should see output similar to:

```
Sympy variables:  (x1,)
elapsed time:  0.09335684776306152
{'b_degree':  2, 'barrier':  3.651*(1 - 0.016*x1)**2, 'gamma':  3.8977794359919664, 'lambda':
4.0398245551064065}
```

To check FOSSIL is installed and working correctly, navigate first to the folder:

```
cd ~/PRoTECT/ex/benchmarks-deterministic/FOSSIL-versions/
```

and run in the terminal:

```
python3 ex1_dt_DS.py
```

You should see output similar to:

```
Found a valid BARRIERALT certificate
Elapsed Time:  0.3446953239999857
```

If you receive both of these, you can consider the smoke test to be passed.

To test the GUI, return to the PRoTECT folder and start the GUI via:

```
cd ~/PRoTECT
python3 main.py
```

3.2. **Full Evaluation.** The folder `ex` contains three subfolders; `GUI_config_files`, `benchmarks-stochastic` and `benchmarks-deterministic`. The subfolder `GUI_config_files` can be ignored until testing the GUI since it is not necessary for evaluating Table 1 and Table 2. It contains all the examples as JSON files which can be imported by the GUI to run the examples via that interface. The subfolder `benchmarks-stochastic` contains the examples used to construct Table 2 for the stochastic cases of both discrete-time stochastic systems (dt-SS) and continuous-time stochastic systems (ct-SS). The final subfolder `benchmarks-deterministic` contains the examples for discrete-time deterministic systems (dt-DS) and continuous-time deterministic systems (ct-DS). This folder also contains two subfolders, the first has the examples in the form that uses PRoTECT called `PRoTECT-versions`, and the second has the examples in the form that uses FOSSIL called `FOSSIL-versions`.

The files themselves are python scripts *e.g.* `ex2_jet_engine_ct_DS.py`. The deterministic case studies are by default set to run in the mode with the heading named serial from Table 1. The stochastic case studies are by default set to run in the mode with heading named Standard from Table 2. Each example can be run from the terminal in the correct directory with:

```
python3 example.py
```

where `example.py` should be replaced by the exact name of the example file to be run. This includes both the examples that use PRoTECT and the examples that use FOSSIL. The output of the examples will always include a solving time which should be compared with Table 1 and Table 2 as well as for the PRoTECT case studies additional parameters `b_degree`,γ, λ, c, ϕ can be compared with the outputs from the returned dictionary `b_degree`, `gamma`, `lambda`, `c`, and `confidence`, respectively.

In the outputs for the examples, you might see some lines saying `Error in degree` followed by some information. These errors let the user know that no solution was available for the barrier with the `b_degree` listed in the `Error` message and the cause for this is lack of a barrier certificate e.g. *`Error in degree: 2 -- barrier is scalar!` These factors should not cause concern; rather, they serve to provide users with insights into addressing the complexity of the system being solved.*

3.3. **Table 1: Running Parallel Versions.** For the deterministic case studies, we also run the scripts in the `PRoTECT-versions` folder in parallel. We include a bash script to make this change navigating to the bash-scripts folder and running:

```
./table1_serial_to_parallel.sh
```

and to do the reverse you can run:

```
./table1_parallel_to_serial.sh
```

You can then rerun the original serial scripts again, this time the output will be in parallel. Doing this manually is described in Section 6.

3.4. **Table 2: Running Optimized Versions.** Similar to the parallelism changes, for the optimized examples, small tweaks should be made to each of the python scripts in the folder `benchmarks-stochastic`.

We include a bash script to make this change navigating to the bash-scripts folder and running:

```
./table2_standard_to_optimized.sh
```

and to do the reverse you can run:

```
./table2_optimized_to_standard.sh
```

Then rerun the stochastic examples, this time the results will be for the optimized cases. Manually changing from standard to optimized is described in Section 6.

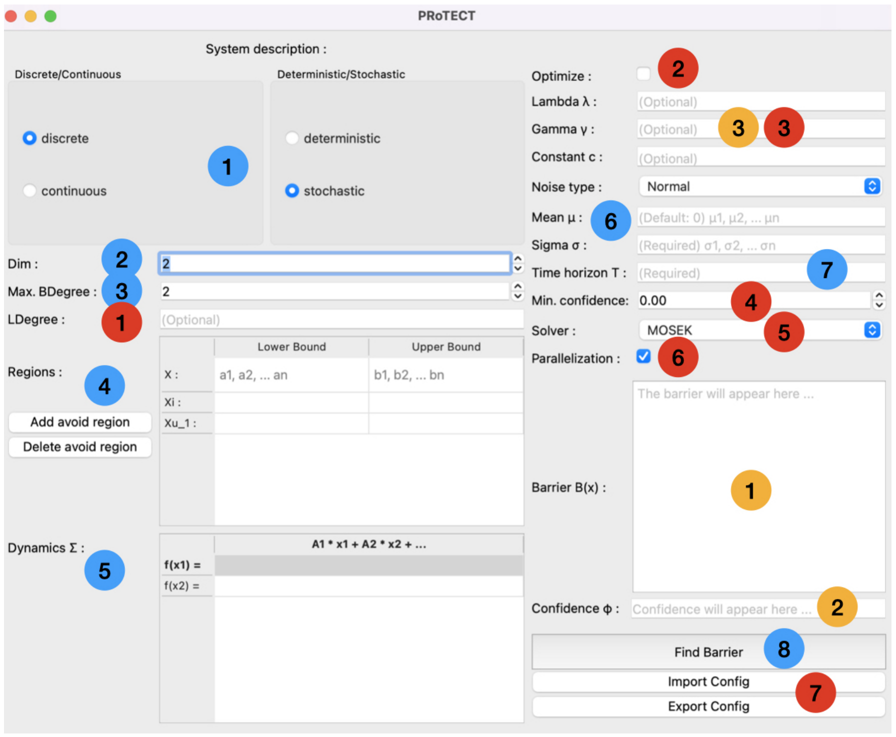
4. Running New Case Studies with PRoTECT

We have tried to make using PRoTECT for your own examples as easy as possible, this can be done in two ways, either through the provided graphic user interface (GUI) or through python scripts that call the PRoTECT functions as an API.

4.1. **Graphic User Interface (GUI).** To enhance accessibility and user-friendliness of the tool, PRoTECT offers the Model-View-Presenter architecture incorporating a GUI. Specifically, a GUI strengthens user-friendliness by abstracting away implementation details for the code, allowing for a push-button method to construct barrier certificates. In Fig. 1, color notation is utilized to represent labels by their corresponding color and number. While PRoTECT provides GUIs for all four classes of systems (see Fig. 1 (blue-1)), we only depict it for dt-SS here to avoid excessive information. Our tool offers two implementations, either serial or parallel (red-6). The tool processes the information entered into the GUI before executing the desired function upon pressing the *Find Barrier* button (blue-8). Outputs of barrier certificate $\mathcal{B}(x)$, confidence ϕ, level sets γ and λ, and constant c are displayed at (yellow-1), (yellow-2), and (yellow-3), respectively. Optionally, the GUI allows for the import and export of configuration parameters in JSON format using the *Import Config* and *Export Config* buttons (red-7), with all the examples from Table 1 and Table 2 of the paper available in the folder `/ex/GUI_config_files`.

By navigating to the `PRoTECT` folder in a terminal and running:

```
python3 main.py
```



[✓ discrete] continuous

deterministic [✓ stochastic]

Dim : 2

Max. BDegree : 2

LDegree : (Optional)

	Lower Bound	Upper Bound
X :	a1, a2, ... an	b1, b2, ... bn
Xi :		
Xu_1 :		

f(x1) =

f(x2) =

Optimize : ☐

Lambda λ : (Optional)

Gamma γ : (Optional)

Constant c : (Optional)

Noise type : Normal

Mean μ : (Default: 0) μ1, μ2, ... μn

Sigma σ : (Required) σ1, σ2, ... σn

Time horizon T : (Required)

Min. confidence: 0.00

Solver : MOSEK

Parallelization : ✓

Figure 1. PRoTECT GUI for dt-SS, where required parameters, optional parameters, and outputs are marked with blue, red, and yellow circles, respectively.

you can open the PRoTECT GUI, to then run examples. We have provided extensive *Youtube tutorial videos* to help with this for the four classes of dynamical systems, as well as the importing and exporting of the config files: here.

4.2. **Treating PRoTECT as an API.** In addition to the GUI, as already seen in the artifact evaluation part of this document, we provide python scripts which can be used to call the PRoTECT functions as an API. We will now describe for each of the four dynamical systems how to use these functions as an API, in addition we have a *Youtube tutorial video* which explains the python scripts used by PRoTECTand explains how to edit them: here.

4.2.1. *Discrete-Time Stochastic Systems (dt-SS).* In general, the backend of PRoTECT behaves as an API, with functions that can be called and used in any python program. We provide some generic configuration files in `/ex/benchmarks-stochastic` and `/ex/benchmarks-deterministic`, which demonstrate how to use the functions in a standard python program. The user is expected to provide the following *required* parameters: dimension of the state set $X \subseteq \mathbb{R}^n$ (blue-2), indicated by `dim`, and the degree of the barrier certificate (blue-3), denoted by `b_degree`. The lower and upper bounds of the initial region $X_\mathcal{I}$, labeled as `L_initial` and

`U_initial`; lower and upper bounds of the unsafe region $X_{\mathcal{U}}$, referred to as `L_unsafe` and `U_unsafe`; lower and upper bounds for the state set X, denoted as `L_space` and `U_space`; where the value of each dimension is separated with a comma (blue-4). Due to possible scenarios with multiple unsafe regions, the unsafe region is passed to the functions as a numpy array of numpy arrays describing each individual unsafe region. The transition map f, represented by `f`, written as a SymPy expression[1] for each dimension using states `x1,x2,...` and noise parameters `varsigma1,varsigma2,...` (blue-5). The time horizon $\mathcal{T}$, noted as `t` (blue-7). The distribution of the noise, `NoiseType`, can be specified as either `"normal"`, `"exponential"`, or `"uniform"` (blue-6).

Users may also specify *optional* parameters, with default values provided in Listing 1. These include the degree of the Lagrangian multipliers $l_i(x), l_u(x), l(x)$: `l_degree` (red-1), which, if not specified (i.e., set to `None`), will default to the same value as `b_degree`; the type of solver: `solver` (red-5), that can be either set to `"mosek"` or `"cvxopt"`. The confidence level ϕ (in equation (5) of the paper) can be optimized using `optimize` (red-2), if set to `True`. In this case, due to having a bilinearity between γ and λ (in equation (5) of the paper), the user is required to provide one λ: `lam`, *e.g.,* select $\lambda = 1$ (red-3). The tool will then optimize for the other decision variables including γ and c to provide the highest confidence level ϕ. Alternatively, the user can select a minimum confidence level ϕ (red-4) using `confidence` they desire, so that PRoTECT attempts to search for a barrier certificate satisfying that confidence level. The parameters for the distributions should be specified as follows (blue-6): for normal distributions, the mean μ can be set using `mean`, and the diagonal covariance matrix σ can be provided using `sigma`. For exponential distributions, the rate parameter for each dimension can be set using `rate`. For uniform distributions, the boundaries for each dimension can be set using `a` and `b`. We provide two functions for dt-SS (red-6): the first `dt_SS` finds a barrier for a single degree, and the second `parallel_dt_SS` runs the first function in parallel for all barrier degrees up to the maximum barrier degree specified (also called `b_degree`).

```
 dt_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
   varsigma, f, t, l_degree=None, NoiseType="normal", optimize=False, solver="mosek",
   confidence=None, gam=None, lam=None, c_val=None, mean=None, sigma=None, rate=None, a=None
   , b=None)
parallel_dt_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
    varsigma, f, t, l_degree=None, NoiseType="normal", optimize=False, solver="mosek",
   confidence=None, gam=None, lam=None, c_val=None, mean=None, sigma=None, rate=None, a=None
   , b=None)
```

Listing 1. dt-SS functions.

[1]https://docs.sympy.org/latest/tutorials/intro-tutorial/basic_operations.html

4.2.2. *Discrete-Time Deterministic System (dt-DS).* The user is required to input necessary (and optional) parameters as outlined in Subsection 4.2.1, excluding those parameters relevant to stochasticity (*e.g.,* time horizon, constant c, noise distribution, and confidence level). Optionally, the user can specify the level sets γ using `gam` or λ using `lam`. It is important to note that optimization for the level sets γ and λ is not performed, as any feasible solution with $\lambda > \gamma$ ensures a safety guarantee over an infinite time horizon. Similarly, we provide two functions `dt_DS` and `parallel_dt_DS` for the serial and parallel execution.

```
dt_DS(b_degree,dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x, f,
  l_degree=None, solver="mosek",gam=None,lam=None)
parallel_dt_DS(b_degree,dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
  f, l_degree=None, solver="mosek",gam=None,lam=None)
```

Listing 2. dt-DS functions.

4.2.3. *Continuous-Time Stochastic System (ct-SS).* The user is asked to enter necessary (and optional) parameters as detailed in Subsection 4.2.1. Additionally, via the corresponding GUI, users must provide the diffusion term δ using `delta` for Brownian motion, the reset term ρ using `rho` for Poisson process, and the Poisson process rate ω using `p_rate`. For cases lacking either Brownian motion or Poisson processes, the corresponding parameter should be set to zero. The confidence level ϕ can also be optimized if `optimize` is set to `True`. We provide functions `ct_SS` and `parallel_ct_SS` for the serial and parallel execution.

```
ct_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x, f, t,
  l_degree=None, delta=None, rho=None, p_rate=None, optimize=False, solver="mosek",
  confidence=None, gam=None, lam=None, c_val=None)
parallel_ct_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
   f, t, l_degree=None, delta=None, rho=None, p_rate=None, optimize=False, solver="mosek",
  confidence=None, gam=None, lam=None, c_val=None)
```

Listing 3. ct-SS functions.

4.2.4. *Continuous-Time Deterministic System (ct-DS).* The user is expected to input necessary (and optional) parameters as detailed in Subsection 4.2.1, omitting parameters pertinent to stochasticity (*e.g.,* time horizon, constant c, and confidence level). Optionally, users can define the level sets γ using `gam` or λ using `lam`. Optimization for level sets γ and λ is not conducted, *i.e.,* any feasible solution where $\lambda > \gamma$ ensures a safety guarantee over an infinite time horizon. Functions `ct_DS` and `parallel_ct_DS` are employed for the serial and parallel execution.

```
ct_DS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x, f,
  l_degree=None, solver="mosek", gam=None, lam=None)
```

```
parallel_ct_DS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
    f, l_degree=None, solver="mosek", gam=None, lam=None)
```

Listing 4. ct-DS functions.

5. Manual Installation

We have tried our utmost to make the following instructions exhaustive for the VM under consideration, there is also a YouTube video here which also walks through the installation instructions in general. Since we include installation details for FOSSIL in this artifact evaluation, and a couple specific tweaks for this VM, the YouTube tutorial should be used in support of the following instructions and not a straight replacement.

The username of the VM is `artifact` and the password (required for sudo commands) is also `artifact`.

5.1. Install PRoTECT.

You download the repository from Zenodo and save it in the home directory, after unzipping the directory change the folder name to PRoTECT to help with the later steps. Alternatively, you can clone Github Release v1.2 which is connected to this Zenodo DOI via the following steps. The first step is to navigate to your home directory (the location that contains the folder `Documents`) and open a terminal (right click then `Open in Terminal`), or open a terminal and run the command

```
cd ~
```

to navigate there.

Install git with the following:

```
sudo apt-get install git
```

Clone the repository for the tool PRoTECT, then go into the folder and install the required dependencies via:

```
git clone https://github.com/Kiguli/PRoTECT --branch v1.2
cd PRoTECT
pip install -r requirements.txt
```

For the GUI to run correctly, this specific VM seems to also require the installation of `libxcb-cursor0` (often it is already installed, such as in the VM used in the Youtube tutorial video):

```
sudo apt-get install libxcb-cursor0
```

You can test this part of the installation from the `PRoTECT` directory, the GUI should load with no issues by running:

```
python3 main.py
```

5.2. **Get Mosek licence.** For the results in Tables 1 and 2 the solver Mosek was used, therefore we now install the licence for Mosek (there is a free trial as well as it being free for academics).

Fill in your details to get a Mosek licence at https://www.mosek.com/license/request/?i=acp. The licence file will be emailed to you with instructions of where to place the file in your home directory. On the VM this will be adding a folder `mosek` to the home directory and add the licence file `mosek.lic` into this directory.

If for whatever reason, the Mosek licence is not available for the reviewer to get, there is an open-source solver CVXOPT which can also be used, in all the example files the line `'solver': "mosek",` *in the dictionary* `fixed_params` *should be changed to* `'solver': "cvxopt"`*. The table results might be a little different to the solutions that used Mosek, see Section 7 for the CVXOPT tables. The following script run from inside the folder* `bash-scripts` *can be used to change all examples from the default* `mosek` *to* `cvxopt`*:*

```
sudo ./all_examples_to_cvxopt.sh
```

Similarly, they can be changed back to `"mosek"` *using:*

```
sudo ./all_examples_to_mosek.sh
```

5.3. **Installing FOSSIL.** To install FOSSIL, navigate to the PRoTECT folder and move the folder fossil-main to the home directory, using:

```
cd ∼/PRoTECT
mv fossil-main ..
```

Or alternatively navigate the terminal back to the home directory with `cd ∼`. Then clone the FOSSIL repository (specifically release 2.0) from Github using the following commands:

```
git clone https://github.com/oxford-oxcav/fossil --branch 2.0
sudo apt-get install -y python3 python3-pip curl
```

After either of the two methods above, install the dependencies via:

```
curl -fsSL https://raw.githubusercontent.com/dreal/dreal4/master/setup/ubuntu/20.04/install.sh
| sudo bash
cd fossil
pip3 install .
```

Note: *there is one command starting from* `curl -fsSL` *and ending at* `bash` *which is one single long command and should not be split across two lines like this document has auto-formatted it to do. Additionally, the dot in the last command is important to point to the current directory.*

5.4. **Copying PRoTECT models into FOSSIL.** To run the provided FOSSIL versions of our examples from Table 1, we need to copy the used models from PRoTECT into FOSSIL. You can append (concatenate) this file directly to the end of the other file using the following command (all one line):

```
cat /PRoTECT/ex/benchmarks-deterministic/FOSSIL-versions/models.py >>
/FOSSIL/experiments/benchmarks/models.py
```

You can also do it *without* the terminal. First navigate inside of the `PRoTECT` folder to the path:

```
~/PRoTECT/ex/benchmarks-deterministic/FOSSIL-versions/
```

Open the file `models.py` and copy all of the content from this file. Then navigate to the FOSSIL tool (the `fossil` folder) to the path:

```
~/fossil/experiments/benchmarks/
```

open the file `models.py` and paste the copied code at the end of the file, then save and exit.

5.5. **Setup PYTHONPATH.** The final task in the setup and installation of this artifact evaluation is to add the paths of both PRoTECT and FOSSIL to the PYTHONPATH so that the python scripts we will call can find the functions they require from the PYTHONPATH. Again this section can be completed without needing the terminal.

Go to the home directory and open the hidden file `.profile`. Assuming you are using the VM mentioned, and that both the folders `PRoTECT` and `fossil` are in the home directory, add the following line to the end of the `.profile` file:

```
export PYTHONPATH=$PYTHONPATH:/home/artifact/PRoTECT:/home/artifact/fossil
```

This adds the location of both repositories we have downloaded to the PYTHONPATH. After this is done, save and exit the file and then restart the VM to enable the PYTHONPATH to be permanently updated on the PC.

If not already visible, you can see hidden files in the file manager by going to the menu and ticking the box "Show Hidden Files".

6. Manual Tweaks for Parallel and Optimized Versions

6.1. **Table 1: Tweaking the Parallelism Option.** To do this manually, a small tweak should be done to each of the python scripts at approx. lines $60 - 70$ of the script. The following lines can be found in the script (in this case for a ct-DS example):

```
### Uncomment this line to run the parallel implementation
#result = parallel_ct_DS(max_degree_values, **fixed_params)

### Uncomment this line to run the serial implementation
result = ct_DS(single_degree_values, **fixed_params)
```

The character `#` is used to comment out a line of code so it does not get called by the script to run. To change from serial implementation to parallel implementation, simply remove `#` from the second line above and add `#` to the fourth line, as here:

```
### Uncomment this line to run the parallel implementation
result = parallel_ct_DS(max_degree_values, **fixed_params)

### Uncomment this line to run the serial implementation
#result = ct_DS(single_degree_values, **fixed_params)
```

After saving and exiting this change, you can then run the script file again using:

```
python3 example.py
```

to run the parallel version of the example.

6.2. **Table 2: Tweaking the Optimization Option.** Similar to the parallelism changes, for the Optimized examples, small tweaks should be made to each of the python scripts in the folder `benchmarks-stochastic`. In particular, the dictionary `fixed_params` should be edited to set the parameter `‘Optimize’` to `True` and the parameter `‘lam’` to the λ value for the case study found in Table 2 (usually 10 except for the Van der Pol oscillator which is 1000).

Inside the script the following code:

```
fixed_params = {
...
‘optimize’: False,
‘solver’: "mosek",
‘confidence’: None,
‘gam’: None,
‘lam’: None,
...
}
```

should be changed by setting the values `‘Optimize’` to `True` and the parameter `‘lam’` to the λ value (in this case 10):

```
fixed_params = {
...
‘optimize’:  True,
‘solver’:  "mosek",
‘confidence’:  None,
‘gam’:  None,
‘lam’:  10,
...
}
```

Once changed, save and exit the file. Then run the script with:

```
python3 example.py
```

for the optimized case.

7. Table 1 and 2 via CVXOPT

Table 1. Comparison for deterministic systems: PRoTECT vs. FOSSIL using the CVXOPT solver. All case studies were run on the same desktop computer (Intel i9-12900).

					PRoTECT		FOSSIL
	n	system	γ	λ	serial (sec)	parallel (sec)	(sec)
1D system	1	dt-DS	4.28	4.42	0.09	1.23	0.20
DC Motor	2	dt-DS	1.251	1.253	0.27	1.43	0.22
barr$_2$room$_{DT}$	2	dt-DS	6.00	6.30	0.28	1.54	0.44
Jet Engine	2	ct-DS	1.47	1.64	0.27	0.33	0.28
*hi-ord$_4$	4	ct-DS	11.12	11.28	2.91	3.33	27.5
*hi-ord$_6$	6	ct-DS	22.64	22.85	37.74	47.6	11.6
hi-ord$_8$	8	ct-DS	32.94	33.54	476.7	595.8	21.3

TABLE 2. BC construction for *stochastic systems* via PRoTECT using CVXOPT solver. We compare case studies across three barrier degrees (2, 4, and 6), returning the barrier with the highest confidence. The CVXOPT solver was unable to find a solution when running Optimized VDP. All experiments were conducted on a desktop computer (Intel i9-12900).

				Standard						Optimized					
	n	system	$\mathcal{T}$	`b_degree`	γ	λ	c	ϕ	time (sec)	`b_degree`	γ	λ	c	ϕ	time(sec)
RoomTemp	1	ct-SS	5	6	0.54	2.77	0.37	0.14	0.84	6	$1.4e^{-9}$	10	$2.6e^{-10}$	0.99	1.43
RoomTemp	1	dt-SS	5	6	0.37	3.60	0.58	0.09	0.64	6	$6.3e^{-10}$	10	$2.6e^{-10}$	0.99	1.43
VDP	2	dt-SS	5	6	22.10	25.74	0.71	0.004	1566	N/A	N/A	N/A	N/A	N/A	1826
ex_lin$_1$	2	ct-SS	5	6	7.41	11.68	0.80	0.02	2.73	6	0.34	10	0.04	0.95	6.04
ex_nonlin$_1$	2	ct-SS	5	6	8.31	15.65	1.41	0.02	2.63	6	1.84	10	0.20	0.72	14.15
TwoTanks	2	dt-SS	5	2	2.16	9.08	1.27	0.06	6.74	2	0.01	10	0.005	0.99	52.9
RoomTemp	3	dt-SS	3	2	4.86	9.29	1.28	0.06	1906			10			
hi-ord$_4$	4	ct-SS	3	2	4.39	18.25	4.44	0.03	4511			10			