Reproduction code package for “Decarboninzation Investment Strategies in an Uncertain Climate”

By: Adam Michael Bauer – adammb4 [at] illinois [dot] edu

Citation of journal article: Bauer, A. M., F. McIsaac, S. Hallegatte. Decarbonization Investment Strategies in an Uncertain Climate. Forthcoming in *Earth’s Future*, 2025. Link to accepted version. (Link to journal page will be added once paper is in print.)

The working paper version of our paper is: Bauer, A. M., F. McIsaac, S. Hallegatte. *How Delayed Learning about Climate Uncertainty Impacts Decarbonization Investment Strategies.* World Bank Policy Research Working Paper No. WPS10743, World Bank Group, Washington DC, 2024.

General package overview

This set of codes reproduces all of the figures and analysis carried out in *Decarbonization Investment Strategies in an Uncertain Climate.* This package uses Gurobi, a commerical nonlinear programming solver that is free for academics, but may not be free for everyone. (It is unclear to me if it’s freely available for *all* researchers or just researchers *at universities.* I imagine Gurobi will handle this on a case-by-case basis, so just shoot their customer support staff an email and they can help.)

Each code is assigned a number corresponding to the figure it creates. Note an `si` before the script name indicates that figure is in the *Supplementary Information.* So code `01_xxx.py` makes Figure 1 from the *main text*, which shows our calibration of the marginal abatement cost curves, while code `si01_xxx.py` makes the Figure 1 from the *Supplementary Information.* Here is the full table for both versions:

Figure Desired	Code to Run	Notes
Figure 1: Example abatement investment model and “strawman” model output.	`01_simplified_simulations.sh`	-
Figure 2: Calibrating marginal abatement costs	`02_mac_calibration.sh`	-
Figure 3: Effect of delayed learning on aggregate policy cost	`03_effect_of_learning_low_linear.sh`	-
Figure 4: Effect of delayed learning on the temporal distribution of spending	`04_temporal_redistribution_low_linear.sh`	-
Figure 5: Effect of delayed learning on sectoral allocation of abatement investment	`05_sectoral_response.sh`	-
Figure 6: Effect of delayed learning on the carbon price	`06_carbon_price_response.sh`	-
Figure SI 1: Effect of delayed learning on aggregate policy cost including direct air capture technologies	`si01_dac_effect_of_learning.sh`	-
Figure SI 2: Impact of delayed learning on sectoral allocation of abatement investment when direct air capture technologies are present	`si02_dac_vs_no_dac_comp.sh`	-

Figure Desired	Code to Run	Notes
Figure SI 3: Effect of delayed learning on aggregate policy cost, growing emissions baseline	`si03_effect_of_learning_emis.sh`	-
Figure SI 4: Effect of delayed learning on the temporal distribution of spending, growing emissions baseline	`si04_temporal_redistribution_emis.sh`	-
Figure SI 5: Effect of delayed learning on aggregate policy cost, high-bound calibration	`si05_effect_of_learning_high_linear.sh`	-
Figure SI 6: Effect of delayed learning on the temporal distribution of spending, high-bound calibration	`si06_temporal_redistribution_high_linear.sh`	-
Figure SI 7: Effect of delayed learning on aggregate policy cost, nonlinear calibration	`si07_effect_of_learning_pow.sh`	This figure was verified virtually. See *Known issues* below.
Figure SI 8: Effect of delayed learning on the temporal distribution of spending, nonlinear calibration	`si08_temporal_redistribution_pow.sh`	-
Figure SI 9: Effect of delayed learning on aggregate policy cost, T*= 1.5 deg C	`si09_effect_of_learning_t15.sh`	-
Figure SI 10: Effect of delayed learning on the temporal distribution of spending, T*=1.5 deg C	`si10_temporal_redistribution_t15.sh`	-
Figure SI 11: Carbon prices as a function of learning date	`si11_carbon_price_sensitivity.sh`	Quantitative results may vary. See *Known issues* below.

If you're an academic, you can email Gurobi customer support to get a free academic license. It's easy to install, and once it's installed, I believe you'll be good to go to run the codes.

A final note is that you should consider using the `.yml` file provided in this directory to establish a virtual python environment that should include all of the necessary dependencies for the code to run smoothly. I recommend using `conda` or `mamba` to do this.

How to run the code

To run the codes, simply navigate to the `codes` directory and run the numbered code to recreate the desired figure. If you want to run the program `script_name`, you may need to execute:

```
    chmod +x script_name
```

to grant execution permissions (hence the `+x`) to the script you want to run.

As an example, if you want to recreate Figure 1 which shows our calibration of the marginal abatement cost curves, you would simply run:

```
./02_mac_calibration.sh
```

Notice the first bit of the above program name, `02_mac_calibration.sh`, matches the figure number we wanted to create, Figure 1.

All figures will be deposited into the `codes/figs` folder. To run indiviudal simulations, you can run any of the files in `simulation_mains`, and to make individual figures, you can run any file in the `figure_mains` folder. **Note:** You should run all scripts from the `codes` directory. As an example, let's say you want to run the `invBase_cvxpy_main.py` file in the `ar6_15` calibration, but not save the output. Then in your command line, you'd use:

```
python simulation_mains/invBase_cvxpy_main.py ar6_15 1 0
```

Note: You should be operating in the Python environment provided at the head directory. Without it, I make no guarantees any of this will run on your machine (and even then, well, mileage may vary. . .).

Known issues

The only figures that were not able to be reproduced on a member of the World Bank Group's Reproducibility team's computer were scripts `si07` and `si11`. `si07` was verified virtually, with a team member joining via video call and watching the code run on the author's laptop. `si11` remains slightly different between the reproducer's run and the lead authors.

The hypothesized reason is that the simulations required for `si07` and `si11` have highly convex objective functions, which requires more powerful hardware to solve precisely than what was available to the reproducibility team member. The team member got an `optimal_inaccurate` solution during optimization step of `si07`. The code will throw an error when anything other than an `optimal` solution is found. It was verified that the original author of the code gets an `optimal` solution when the `si07` code is run.

For `si11`, what is plotted is the optimal carbon price for each run, which is the Lagrange dual of emissions reductions in the model. This is found by solving the KCP dual optimization problem, which is even more sensitive to highly nonlinear objective functions than the minimization step itself (which is required for `si07` and throws errors for the World Bank team member). While the results for a given user may vary, the overall qualitative implications of the figure as discussed in the paper hold for both the World Bank team member and the original author.

If a user gets an `optimal_inaccurate` solution for either of these codes, one possible course of action is to edit the `scale` parameter found in the `codes/simulation_mains/invRec_RiskPrem_cvxpy_main.py` file. This can be found on lines 34 through 42. This issue of scaling the objective function is common in optimization.

The hardware of the original author is a 2023 MacBook Pro with an M2 Pro Chip and 16 GB of RAM.

Last edited: 8 May, 2025.