Replication Package for
"Deadly Debt Crises: COVID-19 in Emerging Markets"

Authors: Cristina Arellano, Yan Bai, and Gabriel Mihalache

This package contains the code and data necessary to replicate our results, as well as information on how to obtain the data and software from the various sources. This readme file is divided into four section: data availability statements, computational requirements (software, memory, and runtime), the folder layout of the replication package, and the details of each replication step.

1 Data Availability Statements

Some of the data we used is publicly available. For some variables we relied on commercial sources, via institutional subscriptions. This package includes cleaned and merged data, sufficient for replication.

IMF International Financial Statistics (IFS). We used International Monetary Fund (2022, IFS) data for National Accounts variables (Gross Domestic Product, Household Consumption Expenditure, and Government Final Consumption Expenditure) and exchange rates for Argentina, Brazil, Chile, Colombia, Ecuador, and Mexico. The data is public and freely available at https://data.imf.org/IFS and the subset used in our analysis is included in Stata format, in `data/Raw_data/NA_govtdebt.dta`.

World Bank. We use the World Bank (2022) website for public data on population, for all countries, in 2018, freely available at https://data.worldbank.org/. We include this variable in Stata format in `data/Raw_data/pop18.dta`.

Institute for Health Metrics and Evaluation (IHME) Data. The epidemic variables are obtained from the public archive of the Institute for Health Metrics and Evaluation (2022), available at https://www.healthdata.org/node/8787. Mobility (`mobility_mean`) and daily cumulative death (`seir_cumulative_mean`) are from the Nov 19, 2021 file; masking (`mask_use_mean`) and the infection fatality rate (`infection_fatality`) are from the May 6, 2022 file. Variable names follow the original IHME source notation. In our replication packages, this data is stored in `data/Raw_data/IHME.xls` in Excel 97–2003 format.

CEIC. We used the commercial CEIC (2022) website for debt data for all countries and National Accounts data for Paraguay, Peru, and Uruguay. CEIC subscriptions are available at https://www.ceicdata.com/en. The database list primary sources for the debt data, by country.[1]. The data from this source has already been merged into `data/Raw_data/NA_govtdebt.dta`.

GFD. We use the commercial GFDatabase of Global Financial Data (2022) for daily EMBI spreads. The primary source for EMBI indices is JPMorgan Chase. The spreads data is included, in Stata format, in `data/Raw_data/spread_EMBI.dta`.

Statement about Rights. We, the authors of the manuscript, certify that we have legitimate access to and permission to use the data used in this manuscript.

Summary of Availability. CEIC (2022) access for data on debt levels and Global Financial Data (2022) access for spreads require subscriptions. All other data are publicly available.

1. Argentina: Ministry of Treasury. Brazil: Central Bank of Brazil. Chile: Chilean Budget Estimation Directory. Colombia: Ministry of Finance and Public Credit. Ecuador: Ministry of Economy and Finance. Mexico: Secretary of Finance and Public Credit. Paraguay: Under Secretary of State for Economic Affairs. Peru: Central Reserve Bank of Peru (Public Debt). Uruguay: Central Bank of Uruguay

2 Computational Requirements

Software Requirements. Several pieces of software are necessary to replicate our results. Processing the raw data, from `data/Raw_data` files into the final `.txt` files used by our main code, requires *Stata 14* or newer and the `asgen` package, which can be installed directly from Stata with the `ssc install asgen` command. Our main program is written in *Fortran 2018*. We tested our code using both the Intel Fortran compiler ("classic"), included in the freely available Intel OneAPI toolkit, and the commercial HPE/Cray compiler for ARM64. The perfect financial markets case uses the NLopt library (Steven G. Johnson (2023)). Further details on the compilation and execution of our Fortran program are below. Finally, the generation of tables and figures requires *MATLAB R2022b* or newer.

Memory and Runtime Requirements. Generating the figures and tables with MATLAB is fast, taking seconds at most, on typical desktop or laptop computers. The same applies to processing the data, using Stata. The main code, in Fortran, can be run feasibly only as a large cluster job. The baseline program and its extensions each take over 2 hours to run on 14 Intel Xeon nodes, each with 40–48 cores and 256GB of RAM. The `perfect_financial` model runs in serial, on a single node and core, and takes under 5 minutes.

3 The Folder Structure of the Package

The replication package's main folder consists of several subfolders:

- `matlab_tables_figures/`: MATLAB scripts for loading results from all other folders and generating the figures and tables of the paper
- `data/`: Stata script for processing data and preparing input for other programs
- `baseline/`: Fortran code of baseline model
- `perfect_financial/`: Fortran code for model with perfect financial markets
- `persistent_recession/`: Fortran code for model with a persistent recession
- `debt_suspension/`: Fortran code for the model with the debt suspension program
- `fixed_L/`: Fortran code for the fixed social distancing counterfactual (in Appendix)

It is possible to generate directly the final tables and figures of the paper without processing the raw data or running the various Fortran programs. See the following section for instructions on producing the final output and then details for reproducing the intermediate steps, if needed.

4 Details of Replication

Replication of Tables and Figures. For convenience, a full set of results from all the Fortran programs is included in the replication package. To reproduce immediately the tables and figures in the paper, using this output,

1. Change the working directory of MATLAB to `matlab_tables_figures/`
2. Execute the `loadResultsToMatlab.m` script. It loads results from several other folders in the replication package, consolidates them, and generates the file `inMatlab.mat`. It produces no console output
3. Execute the `replicate_tables_and_figures.m` script. It will output *all* tables and figures of the draft, to console and in separate windows, and label them following the paper's notation, e.g., "Figure 1" or "Table 2" etc.

These scripts were developed on MATLAB R2022b and ran on Windows and Linux (Ubuntu).

Processing Data for Use by Main Program. All data-related files are in the `data/` subfolder of the main replication package. The Stata script `Main.do` imports the raw data files from `data/Raw_data/` and generates four `.txt` files (`data_L.txt`, `data_D.txt`, `data_mask.txt`, `data_dates.txt`) required by all the model programs and an Excel file used in the production of the final tables (`Output.xls`). The script has been tested on Stata 14 and newer. It requires the `asgen` package, which is available through `ssc`.

Main Model Analysis. All programs are implemented in Fortran 2018 and tested using

- the HPE/Cray compiler for ARM64, on the Ookami Supercomputer, and
- the Intel Fortran compiler "classic," on the Ohio Supercomputer Center's Pitzer cluster.[2]

For the *perfect financial markets* case only, the code employs the NLopt library (https://nlopt.readthedocs.io/en/latest/) and its NLopt-f Fortran interface (https://github.com/grimme-lab/nlopt-f). The output of all programs is loaded in MATLAB for plotting and statistics, as described below.

General Structure of Programs. The code is parallelized over multiple nodes using Open MPI and over all processors of each node using OpenMP, except for the `perfect_financial/` program, which is serial and uses the NLopt library. The folder structure of *each* program is given by

- `src/`: Fortran source files for program and modules
- `bin/`: location of executable
- `results/`: location of program output
- `Makefile`: GNU Make compilation script
- `slurm_script.sh`: Sample SLURM job description script

The code requires that the executable is placed in `bin/` but that it is called/executed from the same folder as the `Makefile`, one level up from `bin/`. All programs will load data from `../data/data_*.txt` and generate output in `results/`, roughly 16GB of binary files and `.tab` files (per program). The minimum output necessary to generate final tables is included in the replication package. (Which files are necessary varies by program.) The majority of the output files is `results/` can be discarded after the final MATLAB stage, as they are only required during Fortran execution.

Configuring the GNU Makefile build script. To build each program using Make, edit the `Makefile` and choose `intel` or `cray` on line 2. The other compiler options were not tested with the most recent version of the programs (`arm`, `gnu`, etc.). Confirm that your compiler command is correct by editing the appropriate `compiler =` line. For example, `mpifort` versus `mpiifort` on some systems. If using a newer version of Intel's compiler, you might need to change `-mkl` to `-qmkl` on line 7.

Running the Code, Execution Time. The code is parallelized with MPI over nodes and OpenMP over all processors of each node. This requires that the executable is called with `mpirun` or `srun`, depending on your system. On the OSC Pitzer cluster, on Intel hardware, the `baseline` program takes 2 hours to run, on 14 nodes $\times$ 40 cores. The `perfect_financial/` case takes $\sim$ 5 minutes,

2. Information on the Ookami system is available at https://www.stonybrook.edu/ookami/ while the Ohio Supercomputer Center website is https://www.osc.edu/. Both systems use Linux distributions and the SLURM resource scheduler. The Intel compiler is available as part of the OneAPI HPC Toolkit, at https://www.intel.com/content/www/us/en/developer/tools/oneapi/hpc-toolkit.html

executed in serial (1 node × 1 core). The data (Stata) and processing (MATLAB) steps take seconds. On ARM hardware on the Ookami cluster, comparable execution times are possible with 40 nodes × 48 cores each. A sample script is included, for reference, with the name `slurm_script.sh`. Partition names, accounts, etc. need to adapted to your system. Due to the way MPI parallelism is implemented in our code, the number of nodes used must be a divisor of 2,268,000.

Running the Programs. For the `baseline/`, `perfect_financial/`, and `persistent_recession/` models, each folder is ready for compilation and execution, using the default/initial state of the source files. The case of perfect financial markets requires the use of the NLopt library and its NLopt-f interface. See the `Makefile` in the folder for more details.

Simulating the Voluntary Restructuring. Start with a fresh version of the `baseline/` folder. Inside `src/` edit `Param.f90` and change

- line 10: `simFileName = "baseline_vol"`
- line 16: `useResetB = .TRUE.`
- line 80: `resetB = 0.512966_wp * 52.0_wp`

after executing the code, the file `sir_baseline_vol.tab` will be saved in `results/`.

The Loan Program. To compute the case of the loan program (a 10% of output loan relative to the baseline), start with a fresh version of the `baseline/` folder. Inside `src/` edit `Param.f90` and change

- line 10: `simFileName = "baseline_loan"`
- line 17: `exoLoan = .TRUE.`
- line 50: `exoLoanSz = 0.1_wp * 52_wp`

after executing the code, the file `sir_baseline_loan.tab` will be saved in `results/`.

To compute the loans at 50% and 70% initial debt to output ratio, follow the same steps: first, use the `useResetB` flag on line 16 to enable the reset of initial debt on line 80, to generate `baseline_50` and `baseline_70` results, then repeat with the loan flags on as well, to generate the `baseline_50_loan` and `baseline_70_loan` files, respectively.

The Debt Service Suspension Initiative. This program is stored in `debt_suspension/`. There are two parameterizations of interest, $\kappa^{\text{DSSI}} = 1$ and $\kappa^{\text{DSSI}} = \kappa$. To configure the latter, edit `src/Param.f90` and set

- line 10: `simFileName = "dssi"`
- line 48: `kappaDSSI = kappa`

while for the former use

- line 10: `simFileName = "dssi_k1"`
- line 48: `kappaDSSI = 1.0_wp`

Sensitivity Analysis. To reproduce the sensitivity analysis in Table 5, use the fresh `baseline/` folder and edit `src/Param.f90` as follows:

- For $\chi = 3000$ set line 10 to `simFileName = "baseline_chi"` and line 31 to `chi = 3000.0_wp`
- For $\pi_{D,1} = 0.04$ set `simFileName = "baseline_piD1"` and line 57 to `piDsq = 0.04_wp * pp`

- For short event, set `simFileName = "baseline_H2"` and on line 46 `H = 2 * 52`
- For long event, set `simFileName = "baseline_H4"` and on line 46 `H = 4 * 52`
- For the unexpected second wave case, set `simFileName = "baseline_2wave"` and line 19 to `surpriseWave2 = .TRUE.`

Note that once these sensitivity cases are ran, the output of the proper baseline case will be overwritten and baseline will need to be run again if you plan to run the comprehensive MATLAB script again. Only the `sir_*.tab` files in `results/` have different names and are loaded by their name in MATLAB. The binary policies and other `.tab` files are required to match the baseline case. This also applies for the loan program case above.

The Fixed L Case (Appendix). The fixed social distancing counterfactual is implemented by the program in the `fixed_L/` folder. Is it self-contained by it requires a `.txt` file with the time path of the L choice variable from the perfect financial markets case, which is supplied as `perfectL.txt` in the folder, for your convenience.

References

CEIC. 2022. "Global Database." Accessed February 1, 2023. https://www.ceicdata.com/en.

Global Financial Data. 2022. "GFDatabase." Accessed February 1, 2023. https://globalfinancialdata.com/.

Institute for Health Metrics and Evaluation. 2022. "COVID-19 Mortality, Infection, Testing, Hospital Resource Use, and Social Distancing Projections." University of Washington. Accessed February 1, 2023. https://www.healthdata.org/.

International Monetary Fund. 2022. "International Financial Statistics." Accessed February 1, 2023. https://data.imf.org/.

Steven G. Johnson. 2023. "The NLopt nonlinear-optimization package." Accessed February 1, 2023. http://github.com/stevengj/nlopt.

World Bank. 2022. "Population Data." Accessed February 1, 2023. https://data.worldbank.org/.