

# Replication files for “Optimal Taxation, Marriage, Home Production, and Family Labour Supply”

There are four main stages to the replication. First, the **data setup** stage, which produces a series of empirical moments that are used in the estimation. Second, the **estimation** stage which obtains the model parameter estimates and produces a series of output files (corresponding to the analysis in Section 4 of the paper). Third, the **simulation** stage which uses the estimated model to conduct a number of alternative optimality simulations and produces a series of output files (corresponding to the analysis in Section 5 of the paper). Fourth, there is the **results** stage which collect the output files from all the previous stages and produces the tables and figures as they appear in the paper and the Online Appendix.

Note: Various parts of the code use the MPI communication protocol for parallel computing. The number of MPI processes required varies across jobs. The largest number of required MPI processes for a replication job is 199. Grid engine scripts are provided. The output produced at the various stages assumes a case-sensitive file-system is used.

## 1. Data setup

### 1.1. Moment files

The data setup is conducted in Stata. It processes raw data from the IPUMS USA American Community Survey (ACS) data, and the Multi-Year American Time Use Survey (ATUS) Data Files (available from U.S. Bureau of Labor Statistics). All the Stata files `.do` files are saved in the directory `stata/do/`. Once you have obtained these data sets, you should change the paths defined in `mom_global.do`. The main program to be run is `mom_all.do`. This runs a small number of Stata `.do` files, including `mom_ipums_3.do` which creates the ACS derived moments, `mom_atus_4.do` which creates the ATUS derived moments, and `mom_input_3.do` which creates the input files for the following **estimation** and **simulation** stages. Bootstrapping is performed. The number of replications is given by the Stata global macro `$bsrep` (as defined in `mom_global.do`).

**Output:** The moments files for the main specification used in the paper are saved in `input/moments/ed3_age2545/`. There are separate files for each moment block. In the subdirectory `mode` the files indicate the data source the moments were calculated from (1 for ACS, 2 for ATUS); in the subdirectory `point` the files contain the moment point estimates; in the subdirectory `stdev` the files contain the bootstrapped standard deviations; in `sample/"sample"/` the complete bootstrap output for "sample" from 1 to `$bsrep` is provided.

**Software requirements:** The code may be run with Stata/SE versions 13 and above.

## 1.2. Moment labels

To generate labels for moments to be included in the Fortran code for the following **estimation** and **simulation** stages there is a short Python script. This processes a selected derived type from the Fortran source and generates source code that can then be included. The use is `./cmd_tools_label_mom.py -i inputfile -o outputfile -type typename`.

**Output:** Fortran source code outputfile.

**Software requirements:** Python 2.7.

## 1.3. Replication steps

- `stata-se mom_all.do`
- `./cmd_tools_label_mom.py -i globaldef.f90 -o momlab.f90 -type mom_t`

## 2. Estimation

### 2.1. Parameter estimates

All the estimation code is written in Fortran 2008 and it uses MPI to parallelize the calculation of the objective function, the non-linear constraints, and their derivatives. A makefile is provided for compilation and compiler settings and library paths should be modified as appropriate. As an input it uses the estimation moments that were calculated in the **data setup** stage, and which are saved in the directory **input**. Grid engine scripts are provided for running the estimation code, and these may need to be modified based on your cluster configuration. The usage is `mpi_est_market_mpec.sh [tag] [mode]`, where **tag** is an identifier, and **mode** allows different versions of estimation to be performed. The default is `mode=0`, which the main results reported in the paper are calculated with. Alternative values of **mode** are described below in the context of robustness exercises. The associated Fortran program source file is `est_mpec_market_mpi.f90`.

**Output:** The estimation results are saved in `output/estimation/["tag"]`. An output log file, an iteration file, and a solution file are produced. Only the solution file (`xsol_market_mpec_mpi.asc`), which contains the model parameter estimates, is used elsewhere.

**Software requirements:** Requires a Fortran 2008 compiler (such as the Intel Fortran Compiler), and the NAG Fortran Library (which is used to minimise the objective function subject to constraints). The latter can be substituted with alternative optimisation routines.

## 2.2. Variance matrix

Once the estimation has concluded, it is necessary to calculate the variance matrix of the estimator, and generate samples of the parameter vector for the subsequent tax simulation bootstrapping. This is done using the script `mpi_variance.sh [tag] [mode]`. The associated Fortran program source file is `tools_variance_matrix.f90`.

**Output:** The variance matrix and intermediate objects are saved in `output/estimation/["tag"]/variance`. Samples from the distribution are saved in `output/estimation/["tag"]/variance/samples`.

**Software requirements:** Requires a Fortran 2008 compiler and the NAG Fortran Library.

## 2.3. Descriptives

Given the model estimates, output files are produced that describe the model fit, summarise the behavioural implications of the model (time allocation and marriage market elasticities), calculate the revenue target, and generate the expected utility possibility frontiers. These are obtained by running the program file `mainkid.out [tag] [mode]`. Note that this is an executable and not a grid engine script.

**Output:** All output is saved in text files in `output/post/["tag"]`.

**Software requirements:** Requires a Fortran 2008 compiler.

## 2.4. Robustness exercises

The paper describes some robustness exercises in footnotes. To estimate a version of the model with alternative market definitions specify `mode=2`. To estimate a version of the model with the transfer value of Medicaid incorporated in the budget constraint, specify `mode=3`. To estimate a version of the model with four education types, it is necessary to i) define the macro `_NTYPE4_` (rather than `_NTYPE3_` for three education types) in `makefile` and recompile; ii) modify the grid engine script to specify the correct number of MPI processes.

**Output:** As above.

**Software requirement:** As above.

## 2.5. Estimation replication steps

The following steps replicate the main estimation and produce the output required for both the tables and figures, and as inputs for the simulation exercises (note that “600” is the tag name used).

- `./mpi_est_market_mpec.sh 600`
- `./mpi_variance.sh 600`
- `./mainkid.out 600`

## 3. Simulation

All the simulation code is written in Fortran 2008 and uses MPI for parallel computing. The code can be run using grid engine scripts similar to those used in the initial estimation exercise.

### 3.1. Primal problem

The main primal simulations (maximise welfare subject to a revenue constraint) can be run with the script `mpi_res_primal_optimal.sh [tag] [taxins] [delta] [mode]`. The parameter `taxins` describes the tax instrument set. Using the terminology from the paper, this can take the following values: 1 (Unrestricted); 2 (Independent); 5 (Income splitting); 6 (Income aggregation); 8 (Gender-based taxation); 9 (Full gender-based taxation). Note that while 8 and 9 both allow the tax schedule for married couples to vary by gender, only the latter allows the schedule for single individuals to be gender-specific. The distributional parameter is to be set as  $\delta = -\text{delta}/10$ .

**Output:** The output of the simulation is a parameter vector (`xtax_sol.asc`) that describes the parametrisation of the tax function together with some meta-data. It is saved in `output/simulation/["tag"]/primal"taxins"_t"delta"/`. Additionally, output files are created that mirror the set of estimation moments (here calculated at the solution to the design problem). Note also that welfare weight calculations are performed when `taxins=1`.

**Software requirement:** Requires a Fortran 2008 compiler and the NAG Fortran Library.

### 3.2. Confidence bands

Bootstrapped tax schedules for the primal problem can be obtained by using the script `mpi_res_sample_optimal.sh tag taxins delta sample0 sample1`. The associated Fortran driver program is `optimal_level_sample_mpi.f90`. This should be done so we have samples from 1 up to `nsamples` (as defined in `tools_variance_matrix.f90`).

**Output:** This creates the bootstrapped schedule for samples from the range `sample0` to `sample1` (inclusive) saved in `output/"tag"/samples"taxins"/` as `xtax_sol"sample".asc`. It does not save moments and allocations associated with each bootstrap sample.

**Software requirement:** requires a Fortran 2008 compiler and the NAG Fortran Library.

### 3.3. Dual problem

The dual problem is solved to evaluate the potential welfare gains from tax jointness (maximise revenue subject to a given level of social welfare). As it uses the level of social welfare from the primal problem with `taxins=1` as the welfare target, it requires that the primal problem has been first solved. The grid engine script is `mpi_res_dual_optimal.sh [tag] [taxins] [delta] [estmode]`, with the associated Fortran driver program `optimal_level_restricted_dual_mpi.f90`.

**Output:** The output of the dual problem is of the same form as the solution to the primal problem. Namely, a tax parameter vector and files describing model moments. These are saved in `output/simulation/["tag"]/dual"taxins"_t"delta"/`.

**Software requirement:** Requires a Fortran 2008 compiler and the NAG Fortran Library.

### 3.4. Fixed marriage market

The simulations with a fixed marriage market can be performed using the script `mpi_res_fixed_optimal.sh tag taxins delta mode`. This script calls two Fortran driver programs sequentially `optimal_level_fixed_mpi.f90` and `optimal_level_restricted_dual_mpi.f90`.

**Output:** The first program run (`optimal_level_fixed_mpi.out`) saves two solution vectors in `output/simulation/"tag"/fixed"taxins"_t"delta"/`. The first, `xtax_sol_f.asc`, is obtained as the solution to the out-of-equilibrium problem, while `xtax_sol.asc` is in equilibrium. The revenue saved in the `xtax_sol_f.asc` metadata is equal to the target that is calculated in the **estimation** stage and used in the primal tax simulation experiments. This revenue will in general differ once we calculate the equilibrium given this (and as saved in `xtax_sol.asc`). Additionally, `xtax_noneq.asc` reports the marriage market imbalance. The tax solution vector for the dual problem is then saved as `output/simulation/"tag"/fixed"taxins"_t"delta"/dual/xtax_sol.asc`.

**Software requirement:** requires a Fortran 2008 compiler and the NAG Fortran Library.

### 3.5. Perturbation experiments

The perturbation experiments can be obtained by first running the script `./mpi_res_perturbation_optimal.sh tag taxins delta perturb`. The corresponding Fortran driver program is `optimal_level_perturb_new_mpi.f90` and this solves the primal problem. Different values of `perturb` will conduct different experiments: `perturb=7` sets the efficiency of home time to be zero; `perturb=21` endogenously changes the extent of assortative mating; `perturb=37` changes the gender wage gap.

The dual problem, whose solution is necessary to evaluate the gains from tax jointness, is then solved by running the script `mpi_res_perturbation_dual.sh tag taxins delta perturb`. The associated Fortran driver program is `optimal_level_perturb_dual.f90`.

**Output:** The output from the perturbed primal problem is of the same form as the main primal simulations. These are respectively saved in `output/simulation/"tag"/perturbation"taxins"_t"delta"_p"perturb"` and `output/simulation/"tag"/perturbationdual"taxins"_t"delta"_p"perturb"`.

### 3.6. Fine tax grid

To perform the simulations where the density of the tax grid is increased, it is necessary to define the preprocessor macro `FINETAXGRID` in `makefile` and recompile the programs. Then the script `mpi_res_fine_primal_optimal.sh [tag] [taxins] [delta]` should be run. The corresponding Fortran driver program is `optimal_level_restricted_mpi.f90`.

**Output:** The tax solution vector and allocations are saved in the directory `output/[tag/]primal_fine[taxins]_t[delta]/`.

**Software requirement:** requires a Fortran 2008 compiler, and the NAG Fortran Library.

### 3.7. Replication steps

The following steps replicate the tax simulations and produce all the output required to produce all tables and figures for the paper (note that “600” is the tag name used).

- `./mpi_res_primal_optimal.sh 600 1 00`
- `./mpi_res_primal_optimal.sh 600 1 10`
- `./mpi_res_sample_optimal.sh 600 1 00 1 200`
- `./mpi_res_sample_optimal.sh 600 1 10 1 200`
- `./mpi_res_primal_optimal.sh 600 2 00`
- `./mpi_res_primal_optimal.sh 600 5 00`
- `./mpi_res_primal_optimal.sh 600 6 00`

- ./mpi\_res\_primal\_optimal.sh 600 8 00
- ./mpi\_res\_primal\_optimal.sh 600 9 00
- ./mpi\_res\_dual\_optimal.sh 600 2 00
- ./mpi\_res\_dual\_optimal.sh 600 5 00
- ./mpi\_res\_dual\_optimal.sh 600 6 00
- ./mpi\_res\_dual\_optimal.sh 600 9 00
- ./mpi\_res\_fixed\_optimal.sh 600 1 00 0
- ./mpi\_res\_fixed\_optimal.sh 600 2 00 0
- ./mpi\_res\_fixed\_optimal.sh 600 5 00 0
- ./mpi\_res\_fixed\_optimal.sh 600 6 00 0
- ./mpi\_res\_perturbation\_optimal.sh 600 1 00 7
- ./mpi\_res\_perturbation\_optimal.sh 600 1 00 21
- ./mpi\_res\_perturbation\_optimal.sh 600 1 00 37
- ./mpi\_res\_perturbation\_dual.sh 600 2 00 7
- ./mpi\_res\_perturbation\_dual.sh 600 2 00 21
- ./mpi\_res\_perturbation\_dual.sh 600 2 00 37
- ./mpi\_res\_fine\_primal\_optimal.sh 600 1 0\*

\*Note: Requires FINETAXGRID preprocessor macro to be defined as described in 3.6.

## 4. Results

The figures and tables are produced using Matlab scripts (saved in the `matlab` directory). They use the output from the previous **estimation** and **simulation** stages. There are two main programs that are to be run. First, is `figures_estimation.m`. This produces all the tables and figures that relate to the **estimation** stage (Table 1, Table 2, Table 3, Appendix Table 7, Figure 1, Figure 2, Figure 3, Figure 4). The second is `figures_simulations.m`, which produces the tables and figures that relate to the **simulation** stage (Figure 5, Figure 6, Table 4, Table 5, Table 6, Appendix Table 8, Appendix Table 9, Figure 7, Appendix Figure 8, Appendix Figure 9, Appendix Figure 10, Appendix Figure 11, Appendix Figure 12). There are separate files for each figure and table which are called from these main scripts, and these are all clearly identified in the code through comments. These are all contained in the `utility` subdirectory.

**Output:** tables are saved as `.tex` files in `matlab/tables/"tag"/`. Figures are saved as `.tex` and `.eps` files in `matlab/figures/"tag"/`. The figure files can be included in LaTeX documents with the `pstool` package. "tag" is defined in the Matlab scripts.

**Software requirement:** In addition to Matlab, a few utility programs freely available from the Mathworks File Exchange are required. These are `matlabfrag`, `linspecer`, `legendflex`, which should be accessible in the search path.

#### **4.1. Replication steps**

From the `matlab` directory simply run

- `matlab -no-desktop figures_estimation.m`
- `matlab -no-desktop figures_simulation.m`

#### **5. Contact details**

Please address any questions to George-Levi Gayle ([ggayle@wustl.edu](mailto:ggayle@wustl.edu)) and Andrew Shephard ([asheph@econ.upenn.edu](mailto:asheph@econ.upenn.edu)).