REPLICATION FILES FOR „Measuring the Level and Uncertainty of Trend Inflation“

This readme file describes the set of replication files („the replication set“) including data and code for „Measuring the Level and Uncertainty of Trend Inflation“ forthcoming in the Review of Economics and Statistics.

The next section describes the data sources used for the paper. The second section describes the set of replication files. The third and final section describes the code and how to use it.

Author

Elmar Mertens, em@elmarmertens.com

ChangeLog

- September 2018: Corrected description of Livingston Survey data (original readme misstated column of the source file from which the data was taken)
- September 2018: particlefilterPaddingtonGAPSV.f90, added two lines which were inadvertently commented out before publishing the code

DATA SOURCES

As listed in Table 1 of the paper, four groups of input variables are used in the main paper: realized inflation rates (INF), trimmed and median inflation rates (TRM), survey expectations of inflation (SRV) and nominal interest rates (YLD). The data has been provided by the following sources:

1. The data for variables in INF, TRM and YLD has been obtained from the FRED database maintained by the Federal Reserve Bank of St. Louis. (https://research.stlouisfed.org/fred2) The data is publicly available and is provided in a set of text files, listed further below, which were downloaded from FRED on April 19, 2015.
2. Survey expectations from the Survey of Professional Forecasters (SPF) and the Livingston Survey has been obtained from the website of the Federal Reserve Bank of Philadelphia. (See https://www.philadelphiafed.org/research-and-data/real-time-center/real-time-data/.) The data is publicly available and has been included in the attached file `surveydataWITHOUTBLUECHIP.csv`. In

addition, longer-term forecasts from the SPF are provided in separate Excel spreadsheets that have been downloaded from the SPF page on the aforementioned website of the Federal Reserve Bank of Philadelphia. All files are listed in the next section.

3. Additional survey forecasts have been purchased from Blue Chip Economic Indicators. Researcher wanting to use this data should contact Blue Chip Economic Indicators for subscription details. The attached file `surveydataWITHOUTBLUECHIP.csv` contains columns that are reserved for Blue Chip data, but all values have been set to `NaN`. The replication codes will read these as missing values and replicate all results to the extent that they do not depend on Blue Chip data. Researchers with access to Blue Chip data may paste the data into these columns to replicate the related results shown in the paper. (Please note that the results files inlucded in this replication set include estimates that reflect the use of Blue Chip data.)

FILE TYPES

The replication files posted to the dataverse website consist of 9 zip files. The zip files should be unzipped and their contents stored in a common directory. As explained below, some of these zip files contain only estimation results. These files are tagged `RESULTS` in dataverse and need not be unzipped for the purpose of replicating the paper's results (except for recreating charts based on the original result files).

The zip files differ by the type of files that are contained in each zip file. The remainder of this section describes the different file types.

Raw Data

Provided in `inputdata.zip`.

- Text files downloaded from FRED. These files are processed by the MATLAB script `paddingtonData2015.m`. Each file contains header information with further details about the data source and vintage date:
  - `PCEPILFE.txt` (core PCE)
  - `PCEPI.txt` (headline PCE)
  - `CPIAUCSL.txt` (seasonally adjusted CPI)
  - `GDPDEF.txt` (quarterly GDP deflator)

    - `PCETRIM12M159SFRBDAL.txt` (trimmed mean PCE)
    - `TRMMEANCPIM158SFRBCLE.txt` (trimmed mean CPI)
    - `MEDCPIM158SFRBCLE.txt` (Median CPI)
    - `GS10.txt` (10-year Treasury yield)
    - `GS20.txt` (20-year Treasury yield)
    - `GS30.txt` (30-year Treasury yield)
- The `csv` file `surveydataWITHOUTBLUECHIP.csv` containing data on several survey forecasts. The columns of this `csv` file contain:
    - `A`: The date in YYYYmm format
    - `B`: The month in mm format
    - `C`: The year in YYYY format
    - `D`: Blue Chip 4-quarter ahead forecast for CPI inflation (values set to `NaN` due to license restrictions)
    - `E`: Blue Chip 5-to1- year ahead forecast for CPI inflation (values set to `NaN` due to license restrictions)
    - `F`: Blue Chip 4-quarter ahead forecast for GDP deflator inflation (values set to `NaN` due to license restrictions)
    - `G`: Blue Chip 5-to1- year ahead forecast for GDP deflator inflation (values set to `NaN` due to license restrictions)
    - `H`: Livingston Survey data copied from the source file `LIVINGSTONmediangrowthrate.xls` described below
    - `I`, `J`, `K`, `L`: SPF CPI inflation forecasts for one, two, three, and four quarters ahead. Copied from the columns labeled CPI3, CPI4, CPI5, and CPI6 in `Median_CPI_Level.xls` described below.
    - `M`: SPF long-term CPI inflation forecasts, copied from `Median_CPI10_Level.xls` and `Additional-CPIE10.xls` as described below.
    - `N`, `O`, `P`, `Q`: SPF PCE inflation forecasts for one, two, three, and four quarters ahead. Copied from the columns labeled PCE3, PCE4, PCE5, and PCe6 in `Median_PCE_Level.xls` described below.
    - `R`, `S`, `T`, `U`, `V`: : SPF GDP deflator *level* forecasts for the current quarter as well as one, two, three, and four quarters ahead. Copied from the columns labeled PGDP2, PGDP3, PGDP4, PGDP5, and PGDP6 in `Median_PGDP_Level.xls` described below.
- Additional files obtained from the SPF page on the website of the Federal Reserve Bank of Philadelphia (see above):
    - Median survey forecasts of CPI and PCE inflation at longer horizons:
        - `Median_CPI10_Level.xls` The data provided in this spreadsheet has been merged with the data in `Additional-CPIE10.xls` and copied into `surveydataWITHOUTBLUECHIP.csv` as described above. Further information on the additional data sources can be found at https://www.philadelphiafed.org/research-and-data/real-time-

center/survey-of-professional-forecasters/historical-data/inflation-forecasts
    - `Median_CPI5YR_Level.xls` (parsed by `paddingtonData2015.m`)
    - `Median_PCE10_Level.xls` (parsed by `paddingtonData2015.m`)
    - `Median_PCE5YR_Level.xls` (parsed by `paddingtonData2015.m`)
    - `Median_PGDP_Level.xls` copied into `surveydataWITHOUTBLUECHIP.csv` as described above.
    - `Median_PCE_Level.xls` copied into `surveydataWITHOUTBLUECHIP.csv` as described above.
    - `Median_CPI_Level.xls` copied into `surveydataWITHOUTBLUECHIP.csv` as described above.
- Livingston survey forecasts obtained from the website of the Federal Reserve Bank of Philadelphia and stored in `LIVINGSTONmediangrowthrate.xls` The data in column `D` denoted `G_BasePeriod_To_Forecast12Month` of sheet `CPI` has been copied into `surveydataWITHOUTBLUECHIP.csv` as described above.

Intermediate Data Files

Provided in `inputdata.zip`.

- Denoted `[DATACATEGORY].[TYPE].txt`
    - `[DATACATEGORY]` corresponds to the mnemonics used in the main paper. These are:
        - INF
        - TRM
        - YLD
        - INFTRM
        - SRV
        - LONGSRV
        - INFTRMSRV
        - INFTRMSRVYLD
    - `[TYPE]` is one of ...
        - `settings` for text files providing summary information for each data set. The first two rows state the number of variables `N` and observations `T` (including rows full of missing values), the following rows list the variable names. This information will be parsed both by the estimation and output routines coded in FORTRAN and MATLAB. Each data category is constructed for the same sample of monthly observations ranging from January 1960 through March 2015; `T` is thus always equal to 663.
        - `yData` for files with actual data values (and zeros for missing data). Each

row consists of monthly observations for the different variables included in the associated data category. The variable names pertaining to each column are listed in the corresponding `settings` file in the same order as values are shown across the columns of the `yData` file.
    - `yNaN` provides a matrix of zeros or ones that has the same size as the matrix of data values provided by the corresponding `yData` fille. A value of one indicate the a missing value and corresponds to a zero entry in the `yData` file.
    - `dates` provides a vector of MATLAB date numbers for each observation.
- All these files are created by running the MATLAB program `paddingtonData2015.m` that uses the above-mentioned raw data files as inputs.
- These txt files serve as inputs to estimation routines coded in FORTRAN as well as the output routines coded in MATLAB that generate the figures and tables shown in paper and appendix.

Result files

Provided in `mcmcdat1.zip`, `mcmcdat2.zip`, `mcmcdat3.zip`, `particledatfiles.zip`, `RBARdatfiles.zip`, and `GAPSVdatfiles.zip`.

- All files named `*.dat` are text files that contain intermediate output generated by FORTRAN code and will be used by the MATLAB routines described below that are used to generate figures and tables.
    - The `*.dat` files record either draws from mcmc posterior of different estimation objects (list examples) or moments of the posterior.
    - Files labeled `GELMAN*.dat` report convergence statistics for the mcmc algorithm.
    - Output generated by the two alternative versions of the model can be recognized by the suffixes `rbar.dat` (for the model with a real-rate trend) and `gapSV.dat` (model with stochastic volatility in gap shocks).
    - Output generated by the particle filter contains the label `particles` (as in `*.particles.*.dat`)
    - As described in further detail below, some output files store draws from the model's posterior (or prior) distribution, others report posterior moments. In the latter case, the following twelve moments are reported: mean, median, 0.5, 2.5, 5, 15.86, 25, 75, 84.13, 95, 97.5 and 99 percent quantiles.
- Specifically, the MCMC estimation of the main model with `mcmcPaddington.f90` generates the following output files:
    - `settings.[DATACATEGORY].dat` lists a few estimation parameters

- `YDATA.[DATACATEGORY].dat` contains a copy of the input data
- `TAU[N].[DATACATEGORY].dat` reports a time series of posterior moments for trend level of the `[N]`th variable in `[DATACATEGORY]`. The results are arranged in a `T` times `12` matrix. `TAU.[DATACATEGORY].dat` is identical to `TAU1.[DATACATEGORY].dat`.
- `TAU0DELTA[N].[DATACATEGORY].dat` contains draws from the posterior distribution of the difference in initial trend levels for the `[N]`th and the first variable in `[DATACATEGORY]`.
- `SIGTREND.[DATACATEGORY].dat` reports a time series of posterior moments for the stochastic volatility of shocks to the common trend.
- `F.[DATACATEGORY].dat` reports posterior moments for the coefficients of the companion matrix `vec(F)'` of the Gap VAR
- `MAXLAMBDA.[DATACATEGORY].dat` contains draws from the posterior distribution of the largest absoulute eigenvalue of the companion matrix of the gap VAR.
- `SIGMAGAP.[DATACATEGORY].dat` and `SIGMAGAP.DRAWS.[DATACATEGORY].data` report moments and draws, respectively, from the posterior moments of the variance-covariance matrix of gap shocks (`vech(Sigma)`)
- `HVARBAR.[DATACATEGORY].dat` provides draws from the posterior distribution of the variance of shocks to the random walk in the log-variance of trend shocks
- `SHOCKSLOPES.[DATACATEGORY].dat` is a legacy object, contains only zero value
- `GELMAN.f.[DATACATEGORY].dat`, `GELMAN.hvarbar.[DATACATEGORY].dat`, `GELMAN.sigmagap.[DATACATEGORY].dat`, `GELMAN.states.[DATACATEGORY].dat`, `GELMAN.svol.[DATACATEGORY].dat` contains test statistics from the scale reduction test described in the paper for various estimation objects. (`states` refers to the linear states of the model comprising of trend and gap levels)
- Draws from the prior distribution for some model parameters are stored in `hvarbar.prior.[DATACATEGORY].dat`, `maxlambda.prior.[DATACATEGORY].dat`, `shockslopes.prior.[DATACATEGORY].dat`, `sigmagap.prior.[DATACATEGORY].dat`

- The particle filter estimation of the main model with `particlefilterPaddington.f90` generates the following `dat` files
  - `settings.particles.[DATACATEGORY].dat` lists some parameters used in the particle filtering estimation
  - `YDATA.particles.[DATACATEGORY].dat` and `YNAN.particles.[DATACATEGORY].dat` are copies of the input data and flags for missing values

- `ESS.particles.[DATACATEGORY].dat`: a vector of length `T` with ESS statistics for every date in the sample
- `LOGLIKE.particles.[DATACATEGORY].dat`: a vector of length `T` with the log-likelihood contributions for every date in the sample. (`LOGLIKE2.particles.[DATACATEGORY].dat` contains the contributions ot the likelihood for the first variable in `[DATACATEGORY]`, which is PCE headline inflation in most cases.)
- `GAINTAUHAT[N].particles.[DATACATEGORY].dat` is a `T` times `N` matrix of the Kalman gain coefficents for the `N`th trend level. (Despite the common trend assumption, the gain coefficients for the trend levels in different variables may differ due to differences in inference about the initial trend levels). The gain coefficients are computed as particle-weighted averages of the Kalman gains computed for every particle.
- `SVHAT1.particles.[DATACATEGORY].dat` is a vector of length `T` reporting the particle-weighted average of particle draws for the stochastic volatility of trend shocks. `SV1.particles.[DATACATEGORY].dat` and `smootherSV1.particles.[DATACATEGORY].dat` are `T` times `12` matrices reporting posterior moments for the filtered and smoothed posteriors of the stochastic volatility in trend shocks obtained from Monte Carlo simulations.
- `TAUHAT[N].particles.[DATACATEGORY].dat` is a vector of length `T` reporting the particle-weighted average of the conditional means for the trend level of variable `[N]`. `TAU[N].particles.[DATACATEGORY].dat` and `smootherTAU[N].particles.[DATACATEGORY].dat` are `T` times `12` matrices reporting posterior moments for the filtered and smoothed posteriors of the stochastic volatility in trend shocks obtained from Monte Carlo simulations. (Notice that due to the vague priors for the initial trend levels, noticeable differences may arise between the simulated value for the filtered means and the particle-weighted average of the conditional means for the early observatiosn in the sample.)

- Similarly named `dat` files are generates also by the estimation routines for the extended model that includes a real-rate trend (with suffix `rbar.dat`) and the alternative model with stochastic volatility in the gap shocks (with suffix `gapSV.dat`).

Estimation Log Files

Provided in `logfiles.zip`.

The screen output of each run of every FORTRAN routine has been captured in a `*.log` file. The log files are named `[PROGRAM].[DATACATEGORY].log` where `[PROGRAM]` is the name of one of the FORTRAN routines. The possible values for

[DATACATEGORY] have been described above.

CODE

Provided in code.zip, the code comes in three parts:

1. Data creation via the MATLAB script paddingtonData2015.m This script generates the intermediate data files described above.
2. Estimation via MCMC and a particle filter has been encoded in FORTRAN. These routines are called from the shell and require that the intermediate data files are in the present working directory. Compilation of the FORTRAN code is described further below. The individual files and the usage of their executables (after compilation) is described here:
   - mcmcPaddington.f90 estimates the main model of the paper with MCMC. To run the estimation call ./mcmcPaddington [DATACATEGORY] on the Linux shell (where [DATACATEGORY] is a placeholder for one of the mnemonics defined above. Output from this routine is labeled *.[DATACATEGORY].dat
   - particlefilterPaddington.f90 computes particle filtered and smoothed estimates of the main model conditional on parameter values estimated with MCMC. To run this program, it is required that the appropriate intermediate data files and MCMC result files are stored in the current directory. To execute this program call ./particlefilterPaddington [DATACATEGORY]. To execute this program using a different set of parameter values --- as is the case for estimated gains obtained from a larger state space, as shown in Figure 4 and 5 of the main paper --- call ./particlefilterPaddington [DATACATEGORY1] [DATACATEGORY2] where [DATACATEGORY2] should correspond to the mnemonic associated with the dat files containing the desired parameter values. Output from this routine is labeled *particles.[DATACATEGORY].dat
   - mcmcPaddingtonRBAR.f90 and particlefilterPaddingtonRBAR.f90 compute MCMC and particle-filtered estimates for the extended version of the model that allows for a time-varying trend in real interest rates. Both programs should be called only with a composite data category that contains YLD at the end. The corresponding settings file contains an additional line reporting the location of the first yield in the data vector, which will be aligned with the real-rate trend in the estimation. Oiutpu
   - mcmcPaddingtonGAPSV.f90 and particlefilterPaddingtonGAPSV.f90 compute MCMC and particle-filtered estimates for the alternative model, described in the appendix, with stochastic volatility in the gap shocks.

- Other `*.f90` files provide modules with estimation routines and helper functions that are needed by the main routines listed above. These module fiels are compiled and linked to to the main routines by the makefile described next
- `makefile` for compilation and execution of FORTRAN code - All computations needed to replicate the results of the main paper are handled by the file `makefile` - The computations related to an extended version of the model with stochastic volatility in gap shocks are handled by `gapSV.makefile`
- For each FORTRAN routine exists a separate `pbs` scripts of the name (but replacing the `f90` sufffix by `pbs`) that allows to schedule the execution of the compiled code on a distributed computing cluster. Estimation of different data categories is implemented via a `pbs` array where each element of the array corresponds to a different element of the array. The number of each array element corresponds to the order of appearance in the list of data categories above. (Each `pbs` script parses the list of data categories from the text file `paddingtonlist`.) For example, `qsub -t 1 mcmcPaddington.pbs` calls the `pbs` script for the MCMC estimation of the main model conditional on the first data category, which is `INF`.

3. Figures and tables are created by MATLAB code based on the `*.dat` output files created by the FORTRAN routines described above. These MATLAB routines are standalone scripts, that can be executed independently of each other (provided all `*.dat` files are stored in the present working directory). The MATLAB routines are labeled `figurePaddingtonFIG*.m` and `figureAppendixPaddington*.m` where the asterisk `*` is a placeholder for the specific figure name. Each script can be called directly on the MATLAB command line. To store the resulting figures in a file, please follow the instructions about changing the `wrap` variable at the beginning of each script.

Software Requirements

- Scientific Linux 5.11
- Intel FORTRAN 13.1.0
- MATLAB 2012a on linux or Windows (2013a or later versions should also work).
- Note: When data creation (via paddington2015data.m) is done in Windows, it is better to post-process the text files with the dos2unix script in Linux to ensure they are properly parsed by the (optional) pbs scripts and FORTRAN routines.
- The FORTRAN code was executed on the Federal Reserve Board's higher-performance computing cluster, using multi core machines with Intel Xeon 2.6 Ghx CPUs and 128 GB of RAM.

How to Run the Code

- First, copy all files (`.txt`, `.m`, `.f90`, `makefile`, `.pbs`, if desired, and for the data creation also `.xkls.` and `.csv`,) into one common directory.
- Optional step: call `make clean` on the linux command to remove unwanted result files and compiler objects
- Create input data (beware of missing blue chip data) by calling paddingtonDta2015. Note: the resulting *.txt files are also contained in the set of replication files. A user wanting to use those, may also skip this step.
- Compile and run the FORTRAN code. - Main paper: Invoke `make` to compile only, or `make run` (or `make pbsrun`) to compile and launch the computation. This makefile handles MCMC and particle filter routines for the baseline model as well as the extended version with a real-rate trend.
  - For the alternative moedl with stochastic volatility in the gap shocks invoke `make -f gapSV.makefile`, `make -f gapSV.makefile run`, or `make -f gapSV.makefile pbsrun`

The Use of OpenMP for Parallel Programming

All FORTRAN routines have been designed for parallel processing based on OpenMP library, which is activated during compilation with the `-openmp` flag as seen in line 7 of `makefile`. While OpenMP is not necessary for executing the code, some of its settings affect the setup of the MCMC routines.

Specifically, each mcmc*.f90 program executes Nstreams independent MCMC chains whose output forms the basis for the convergence statistics implemented by each program. When a multiprocessor environment with OpenMP is available, these chains will be executed in parallel. Absent the capability of running different threads in parallel, the code can still be executed, however the user may want to bear in mind that, by default, the variable Nstreams is set equal to the maximum of either the value 4 or the number of available threads on the system, as indicated by the shell's environment variable OMP_NUM_THREADS. On a single processor system, users may thus want to set OMP_NUM_THREADS to the desired number of MCMC chains, for example, the command ''export OMP_NUM_THREADS=8'' chooses 8 threads / chains. Alternatively, users can also manipulate the variable NStreams in mcmcPaddington.f90 on line 100 (likewise for the other mcmc*.f90 files).

The particle filtering routines `particle*.f90` also seek to exploit parallel processing capabilities via OpenMP, again depending on the number of available threads as indicated by the shell variable `OMP_NUM_THREADS` but will also work just the same in single processor mode.

FORTRAN Setup

Two makefiles accompany this set of replication files: `makefile` and gapSV.makefile. `Both compile and execute different versions of the empirical model used in the main paper and the appendix.` Makefile `handles the baseline version of the model as well as the extended version of with a real-rate trend; this makefile is thus sufficient to replicate all results of the main paper (and the bulk of what is shown in the appendix).` GapSV.makefile` contains analogue instruction for the alternative version of the model with stochastic volatility in gap shocks described in the appendix.

Both makefiles document the compiler settings that have been used, in particular the use of the `-openmp` flag for parallel processing as well as the `-O3` optimization flag and the `-mkl` flag for linking the Intel Math Kernel Library that comes with the Intel compiler.

The makefiles contain two run sections `run` and `pbsrun` that execute the FORTRAN code (and compile the code if necessary). `make run` calls a sequence of estimation routines *in the current shell* whereas `make pbsrun` contains a set of pbs-scripts that schedule the different estimation steps (for each data category) as separate jobs on a cluster system. (The scheduling commands insert a dependency between the particle-filtering jobs and the MCMC estimation, where the former will only launch once the latter have finished, since the particle filtering code depends on the parameter estimates obtained from the MCMC estimation.) To call the main makefile simply say `make run` or `make pbsrun` on the Linux command shell; for `gapSV.makefile` simply add `-f gapSV.makefile` to each call.

Shell Environment Variables

In addition to the use of `OMP_NUM_THREADS` discussed above, it may be necessary to ensure the availablility of sufficient stack memory for the execution of the estimation routines. It is recommended to issue a call to `ulimit -s unlimited` at

the beginning of each shell session. In addition, please set `KMP_STACKSIZE` to 2GB as in `export KMP_STACKSIZE=2GB` to ensure the availability of ample stack memory for each OpenMP worker. (Other values than 2GB might suffice as well.) The use of these environment variables is also documented in the header of each `*.pbs` script.