Learning-by-Doing, Organizational Forgetting, and Industry Dynamics – Code Description and Instructions –

David Besanko* Ulrich Doraszelski† Yaroslav Kryukov‡
Mark Satterthwaite§

June 18, 2009

*Kellogg School of Management, Northwestern University, Evanston, IL 60208, d-besanko@kellogg.northwestern.edu.

†Department of Economics, Harvard University, Cambridge, MA 02138, doraszelski@harvard.edu.

‡Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213, kryukov@cmu.edu.

§Kellogg School of Management, Northwestern University, Evanston, IL 60208, m-satterthwaite@kellogg.northwestern.edu.

1 Overview

We provide code implementing the model and solution methods described in Besanko, Doraszelski, Kryukov, and Satterthwaite (2009). This code is intended for replicating our results. A reader interested in applying the homotopy method to another model can find a more detailed description of the homotopy method and instructions for adapting the code in Borkovsky, Doraszelski, and Kryukov (2008).

There are four main groups of programs:

1. Matlab code that computes a starting point using the Pakes and McGuire (1994) algorithm and generates a binary input file for the homotopy.

2. Model-specific Fortran90 files that read the binary input file, run HOMPACK90 (Watson, Sosonkina, Melville, Morgan, and Walker 1997) and supply it with the system of equations and its Jacobian. We used ADIFOR2.1 (Bischof, Khademi, Mauer, and Carle 1996) to compute the Jacobian.

3. Matlab code that reads the binary output files generated by HOMPACK90 and uses them to compute various summary statistics, notably the Herfindahl indices.

4. HOMPACK90 including BLAS and LAPACK libraries.

The rest of this note provides instructions for compiling and running the code, followed by a brief description of the main groups of programs.

2 Compiling and running

In order to compile and run the code, you must have:

1. Matlab 6.0 or newer version. Matlab is a commercial product developed and distributed by Mathworks. We have used versions 6.5 and 7+.

2. Fortran90 compiler. There are several commercial compilers available and there is a free version developed under the GNU project. We have successfully compiled our code with Compaq Visual Fortran 6.6.a under Windows and Portland Group PGF95 on a Linux system.

Following a solution path can take a substantial amount of time, up to hours depending on the model specification, range of parameter values to cover, and computer performance. We have found it most efficient to use dedicated servers, typically running Linux. At the same time, code development is much easier on a desktop, typically a Windows machine. Because of this, our code is designed to work on both platforms with minimal changes.

To compile and run the code, proceed as follows:

1. **Compile the C code used by the Matlab code.** Open Matlab and run `mex_PM.m`. Matlab typically ships with its own C compiler or automatically finds a compiler available on the system. If no compiler is available, edit `duopolyNE.m` file to comment out the calls to the compiled C files, and uncomment the calls to the Matlab functions or scripts (this reduces computation speed a lot).

2. **Compile the Fortran files.**

   (a) Compile the following Fortran files:

      i. `NE_Main.f`
      ii. `g_NE_rho1.f`
      iii. `NE_rho.f`
      iv. `NE_QRT.f`
      v. `HOMPACK90/hompack90.f`
      vi. `HOMPACK90/rhojsA.f`
      vii. `HOMPACK90/lapack.f`
      viii. `HOMPACK90/hom_fileIO.f`
      ix. `HOMPACK90/blas*.f`

      In case of compilation errors, see Section 4.

   (b) Make sure the resulting executable is in the same directory as the Matlab .m files.

   (c) Open `homNE90_Start.m`, uncomment line 117 if running on a Windows system or line 119 if on a Unix-based one. Make sure there is the correct name of the compiled executable after the "!" character.

3. **Compute starting point**. Edit parameter values in `masterNE.m` and run it to compute and save an equilibrium. Set two (or more) values of δ or ρ to use the equilibrium for the first one as the starting point for the next one.

4. **Run the homotopy**. Edit starting and ending points in `homNE90_Start.m` and run it. Watch the homotopy run (it can take from minutes to hours), then proceed to convert step files from binary to Matlab `.mat` files (`homNE90_read.m`). Finally, compute and plot Herfindahl indices (`hom_steps.m`). Besides the Matlab figure, the output includes two files in *HomRes* directory:

   (a) `DeltaRho_*.mat` contains δ and ρ for each step in the run.

   (b) `Herf_*.mat` contains Herfindahl indices for each step.

3 Starting point code

We provide Matlab code for computing a starting point for the homotopy. While several parts of the code are written to accommodate models with entry and exit, we focus here on models without entry or exit.

1. *GridDat* – Directory with saved equilibrium files, mainly accessed through the code. Already contains one equilibrium ($\delta = 0$, $\rho = 0.85$).
2. `mex_PM.m` – Compiles all C files.
3. `masterNE.m` – Main control script that computes and saves equilibria for one or several sets of parameter values. In the latter case, an already computed equilibrium serves as a starting point for the next parameterization. Before starting the computation, the script attempts to load a saved equilibrium to use as a starting point. The script has the functionality to handle multiple equilibria (the string variable `FileMod` becomes part of the filename), compute transient and limiting (ergodic) distributions and Herfindahl indices, and do summary plots of equilibria.
4. `InitParamsEE.m` – Set up and initialize the global variables used in computation.
5. `duopolyNE.m` – Computes an equilibrium using the Pakes and McGuire (1994) algorithm.
6. `plotEE_Iter.m` – Plots iteration progress.
7. `cost.m` – Cost as a function of experience level.
8. `FOC.m` – FOC for best response problem, as a Matlab function.
9. `FOCc.c` – FOC for best response problem, as a C function (run `mex_PM.m` to compile).
10. `SolveFOCc.c` – Solution to FOC, as a C function (run `mex_PM.m` to compile).
11. `Fsale.m` – Probability of sale going to firm 1.
12. `Fsale0.m` – Probability of sale going to the outside good.
13. `WL.m` – Conditional expectations, as a Matlab script, no entry or exit.
14. `WLEc.c` – Conditional expectation calculation, as a C function, entry and exit (run `mex_PM.m` to compile).

4 Homotopy code

This set of programs initialize the homotopy path-following package HOMPACK90 and describe the model to it. The package itself is described in Section 6.

1. *HomXpt* – Directory that receives the steps files generated by the homotopy.
2. `homNE90_Start.m` – Runs the homotopy:
   (a) loads a saved equilibrium that serves a starting point of the homotopy;
   (b) writes the equilibrium and parameter values into a binary file (`hom_start.dat`);
   (c) calls the executable compiled from Fortran90 files;
   (d) generates the "name" of the run (`fileMod` variable);
   (e) initiates output processing by running `homNE90_read.m` and `hom_steps.m`.
3. `NE_Main.f` – Starting point of the homotopy:
   (a) reads the binary file created by `homNE90_Start.m`;
   (b) sets precision for the path-following algorithm (`ARCRE` and `ARCAE` variables, we do not recommend making precision more strict than 10^{-12});
   (c) calls the path-following algorithm.
4. `NE_rho.f` – System of equations.
5. `NE_SparseStru.f` – Defines sparsity structure of the Jacobian.
6. `g_NE_rho1.f` – ADIFOR-generated code that computes one column of the Jacobian.

The Fortran files (both those describe above and in Section 6) might need minor edits depending on compiler and operating system used. The code contains preprocessor directives (`#ifdef`, `#else`, `#endif`, etc.) that attempt to substitute in correct syntax automatically, but they do not always work. In this case, it is safe to comment out the preprocessor directives and one of the statements that they enclose. The possible code changes that we have identified are:

1. Different compilers use different syntax for opening binary files. This affects `open` statements in `NE_Main.f` and `HOMPACK90/hom_fileIO.f`.
2. Unix uses forward slash ("/") as the directory separator, Windows uses backslash ("\"). This affects the statement "`write(filenm ... HomXpt ...`" in `HOMPACK90/hom_fileIO.f`, see the description of that file in Section 6.
3. At least one compiler (Absoft) does not allow a comma in the `write` statement after parentheses that enclose output unit and format string. This affects multiple files.

5 Output processing and summary statistics

HOMPACK90 output is converted into Matlab `.mat` files, which are saved into an automatically-named subdirectory; Herfindahl indices for every 10th step of the run are saved in a separate file for easy access and plotting.

1. *HomDat* – Directory that stores step files in Matlab `.mat` format, each run gets its own subdirectory.
2. *HomRes* – Directory that stores short files with homotopy results (δ and ρ's, Herfindahl indices).
3. `homNE90_read.m` – Reads the step files generated by the homotopy from the *HomXpt* directory and saves them as Matlab `.mat` files in a subdirectory of *HomDat*. Also writes out list of δ and ρ's to a file in HomRes directory. The name of the run as generated by `homNE90_Start.m` is used for both these files and the subdirectory of *HomDat*.
4. `hom_steps.m` – Computes Herfindahl indices for a given run, plots them and saves them in a file in the *HomXpt* directory; this file again is named after the run. Can also plot and test step files.
5. `EE_symmetry.m` – Symmetry measures (Herfindahl indices).
6. `EE_welfExp.m` – Welfare measures.
7. `markEE_Lim.m` – Limiting (ergodic) distribution, linear algebra computation.
8. `partition.m` – Part of limiting distribution calculation.
9. `components.m` – Part of limiting distribution calculation.
10. `markEE_LimDirect.m` – Limiting (ergodic) distribution, direct computation ($T = 1024$)
11. `markEE_Trans.m` – Transient distributions.
12. `transE.m` – Experience transition probabilities, entry and exit.
13. `transLC.m` – Experience transition probabilities, no exit or entry.
14. `TransMatEE_duo.m` – Markov kernel (matrix of state-to-state transition probabilities).
15. `meanmode.m` – Computes mean and mode of distribution over states.
16. `plotNE_ResTrans.m` – Plots summary of equilibrium.
17. `subtitle.m` – Fills in "title" area in the figure (top middle).

18. `ffooter.m` – Fills in "footer" area (bottom-left corner).

19. `fpage.m` – Fills in "page #" area (bottom-right corner).

6 HOMPACK90 code

The HOMPACK90 code is stored in the *HOMPACK90* directory and consists entirely of Fortran90 `.f` files. We expanded HOMPACK90 by writing several subroutines (`hom_fileIO.f`, `homjac*.f`, and `homjs*.f`) that provide input and output via binary files as well as Jacobian computation.

1. `lapack.f`, `blas*.f` – LAPACK and BLAS packages for linear algebra and dealing with sparse matrices.

2. `hompack90.f` – HOMPACK90 package. Includes the entry-point subroutines for the path-following algorithm (FIXPNS subroutine), as well as all subroutines and functions that it calls.

3. `hom_fileIO.f` – Implements output into and input from binary files. Output is written to a pre-existing *HomXpt* subdirectory. Pre-processor directives (`#ifdef`, etc.) are used for two purposes:

   (a) To ensure the use of a correct directory separator ("\" or "/") on both Windows and Unix machine. It is assumed that a Windows compiler has the "`_WIN32`" symbol defined.

   (b) To use compiler-specific syntax for opening files. To use the "gfortran" syntax, the user must define the `GFORTRAN` symbol in the compiler environment or use the `-dGFORTRAN` switch in the command line. The same setup is used for input files in the model-specific `Main.f` file.

4. `homjsA.f`, `homjsN.f` – Subroutines that compute sparse Jacobian used in `FIXP*S` algorithms, inlcuding FIXPNS that is used by the current code. `homjsN.f` computes numeric Jacobian via a two-sided finite difference scheme. `homjsA.F` computes analytic Jacobian by calling the model-specific ADIFOR-generated file.

5. `homjacA.f`, `homjacN.f` – Subroutines that compute dense Jacobian used in `FIXP*F` algorithms. `homjacN.f` computes numeric Jacobian via two-sided finite difference scheme. `homjacA.F` computes analytic Jacobian by calling the model-specific ADIFOR-generated file. Both files actually compute a specified column of the Jacobian, HOMAPCK90 assembles the Jacobian itself.

References

Besanko, D., U. Doraszelski, Y. Kryukov, and M. Satterthwaite (2009): "Learning-by-doing, organizational forgetting, and industry dynamics," Working paper, Harvard University, Cambridge.

Bischof, C., P. Khademi, A. Mauer, and A. Carle (1996): "ADIFOR 2.0: Automatic differentiation of Fortran 77 programs," *IEEE Computational Science and Engineering*, 3(3), 18–32.

Borkovsky, R., U. Doraszelski, and S. Kryukov (2008): "A user's guide to solving dynamic stochastic games using the homotopy method," Working paper, Northwestern University, Evanston.

Pakes, A., and P. McGuire (1994): "Computing Markov-perfect Nash equilibria: Numerical implications of a dynamic differentiated product model," *Rand Journal of Economics*, 25(4), 555–589.

Watson, L., M. Sosonkina, R. Melville, A. Morgan, and H. Walker (1997): "Algorithm 777: HOMPACK90: A suite of Fortran 90 codes for globally convergent homotopy algorithms," *ACM Transcations on Mathematical Software*, 23(4), 514–549.