Documentation of the Oblivious Equilibrium Solver

Przemyslaw Jeziorski*
Stanford University

Gabriel Y. Weintraub
Columbia University

C. Lanier Benkard
Stanford University
and NBER

Benjamin Van Roy
Stanford University

June 25, 2008

1 Introduction

This document covers compilation and customization of **Oblivious Equilibrium Solver** (OES). OES is a software written in C++ that provides routines that compute: Oblivious Equilibrium (OE), Non-stationary OE (NOE) and OE with aggregate shocks (AOE) that are developed in Weintraub, Benkard, and Van Roy (2008a), Weintraub, Benkard, and Van Roy (2008b) and Benkard, Jeziorski, Van Roy, and Weintraub (2008). The solver also computes approximation error bounds.

The software is distributed under GPL3 license as a source code and precompiled binary packages. In case of using binary packages the only customization changes that are possible are changes in model parameters. General model implemented in the solver is explained in Weintraub, Benkard, and Van Roy (2008a). Extra features not documented in the article are explained in the configuration files. In order to meet various needs of potential users, we offer binary versions of logit profit function, logit profit function with two types of consumers and Cournot profit function. The logit profit funtion is essentially the same as in Weintraub, Benkard, and Van Roy (2008a). The Cournot profit function is an equilibrium outcome of Cournot competition with linear demand. Specific parameters are documented in the configuration files. Packages are available for `x86` and `x86_64` running on `GNU/Linux/BSD` and `Solaris` running on `Sparc`. Microsoft Windows is not supported. More complicated changes have to made using the source code distribution.

OES has two interfaces that enable access to the results: `C++` interface and `Matlab` interface. Details about configuration and usage instructions can be found in the remainder of this document.

OES can be used as a self-containing package but is also prepared to utilize `Ipopt` interior point solver and `MUMPS` sparse solver. First package adds more robustness when computing the equilibria of one-shot logit models. Second package enables user to implement transition matrices in general sparse form, it also greatly speeds up the computation of AOE. We highly recommend configuring MUMPS when using AOE solver. Note that binary packages have `Ipopt` and `MUMPS` hard compiled into the Matlab `mex` files. Therefore no action from the user is required to utilize them.

*Corresponding author, questions and comments: przemekj@stanford.edu

The document is organized as follows. In Section 2 reader can find details about using binary packages. Subsequent sections explain how to install and use the solver using source code distribution. Sections 3-5 contain the information relevant to all the solvers. Users interested in AOE solver should also read Section 6. Section 3 contains details about downloading and compiling source code distribution. After compiling the software user can proceed to Section 4 that describes how to configure and use the solver. Advanced customization possibilities are the content of Section 5. For details about compilation and customization of the solver of AOE one should read Section 6.

2 Binary packages

Binary packages are the fastest way to get started with OES. Binary packages contain **only** Matlab interface. All binary packages include solvers of OE, NOE and AOE.

Binary packages can be downloaded from Przemyslaw Jeziorski's home page at

`http://www.stanford.edu/~przemekj`

After obtaining package that matches the architecture of your operating system and decompressing it to the chosen folder, the user can run Matlab and try executing attached, sample m-files. m-files are well commented and self-explanatory.

Files `main_logit1.m`, `main_logit2.m` and `main_cournot.m` are examples of invoking OE and NOE solvers as well as routines that compute the approximation bounds for both types of equilibria.

Files `main_aggr_logit1.m`, `main_aggr_logit2.m` and `main_aggr_cournot.m` are examples of invoking solver of OE with aggregate shocks.

The binary code can be customized only by editing those files. In case any deeper changes are need please download a full source code and proceed to the next section.

NOTE: Binary packages are statically linked to Ipopt and MUMPS, so no action from the user is required for the software to utilize those packages.

3 Compilation

This section explains how to compile the OES using the source code. Source code can be downloaded from the Przemyslaw Jeziorski's web-site

`http://www.stanford.edu/~przemekj`

Following instructions are made for Unix platforms. At this point the solver does not support Microsoft Windows. After downloading the software the next steps are:

1. Untar the solver on the *Unix-type* system using the command

```
$ tar -xjvf solver.tar.bz2
```

or

```
$ gtar -xjvf solver.tar.bz2
```

on *Solaris* or **BSD*.

2. Enter to the main folder

   ```
   $ cd projekt-distr
   ```

3. Configure the system

   ```
   ./configure
   ```

   You have a following choice of options:

   - `--enable-ipopt` or `--disable-ipopt`
     enables `Ipopt` solver. `Ipopt` is a software package for large-scale nonlinear optimization (disabled by default). It may be used to find a solution to single period pricing game (if applicable to the chosen model).
     If you disable `Ipopt` the pricing game will be solved using build-in Global Newton Rhapson solver and no additional libraries have to be provided.
     If you decide to enable *Ipopt* you need to provide *Ipopt* libraries and headers in the `ThirdParty/Ipopt` directory. Look in the Appendix A for `Ipopt` configuration and installation.
     In case you enable `Ipopt` you can specify the following options:
     (i) `--enable-ipopt_shared` or `--disable-ipopt_shared`
     for advanced users only. Option switches between dynamic and static linking to *Ipopt*. On some systems only one type of linking works.
     (ii) `--with-ipopt_libs=PATH`
     overrides default path (`ThirdParty/Ipopt/lib/ipopt`) to *Ipopt* library to `PATH`
     (iii) `--with-ipopt_headers=PATH`
     overrides default path (`ThirdParty/Ipopt/include`) to *Ipopt* headers to `PATH`
   - `--enable-mumps` or `--disable-mumps` enables and MUMPS sparse solver. Sparse solver drastically speeds up inverting of sparse matrices. This function is used in the aggregate shocks solver. It is highly recommended to used sparse solve when computing OE with aggregate shocks. More details on installation of MUMPS are provided in Appendix B.
   - `--enable-aggregate` or `--disable-aggregate` enables and disables aggregate shocks solver. When enabling aggregate shocks the configuration script automatically tries to search for MUMPS so no `--enable-mumps` is needed.

   Example of use

   ```
   $ ./configure --enable-ipopt --with-ipopt_libs=/home/me/ipopt/libs.
   ```

4. After configure finishes type

   ```
   $ make
   ```

   The current version of the software provides two ways to execute the solver.
   You can use `C++` or `Matlab` interface. The previous command compiles main solver library and `C++` user interface.

   Note: If you want to change any of the configure options run `configure` script again with desired parameters, clean the source by typing

   ```
   $ make clean
   ```

   and recompile with

   ```
   $ make
   ```

3.1 Compilation of Matlab interface

To compile Matlab interface you have to finish all the steps from the previous section and additionally type

```
$ ./compile_matlab
```

or

```
$ ./compile_matlab_32bit
```

for 32-bit systems. In case the compilation process failed try

```
$ ./compile_matlab2
```

it uses different compilation parameters that are suitable for some systems. Above commands require that `Matlab` software is properly installed and `mex` utility is accessible.

4 Usage

This section explains how to use `C++` and `Matlab` interfaces.

4.1 `C++` interface

`C++` interface is invoked using

```
$ src/projekt
```

When using `C++` interface control variables and model parameters, except for profit function parameters, may be set in

```
src/parameters.h
```

Additionally user can choose between three predefined profit functions:

- **Logit model type 1** (default option) - execute (in the main project directory)
  `$ ./choose_model_logit`
- **Logit model type 2 with two types of consumers** - execute (in the main project directory)
  `$ ./choose_model_logit2`
- **Cournot model** - execute (in the main project directory)
  `$ ./choose_model_cournot`

Note:
After changing the model the software needs to be recompiled with
`$ make`

Profit function parameters may be set by editing
`src/parameters_profit.h`

Note:
After changing any of the parameters the software needs to be recompiled with
`$ make`

Note:
Each time you change a model
`src/parameters_profit.h` is replaced by `src/parameters_profit.h.CHOSEN_MODEL`.
Remember to back up your profit function parameter file, otherwise it will be lost.

There are several actions that the solver may perform. Each time the solver is executed OE is computed with given parameters. If one wants to compute:

- `Bounds for Oblivious Equilibrium` - it is necessary to uncomment
  `#define BOUNDS`
  in
  `src/parameters.h`
- `Nonstationary Equilibrium` - it is necessary to uncomment
  `#define NONSTAT`
  in
  `src/parameters.h`
- `Bounds for Nonstationary Equilibrium` - it is necessary to uncomment
  `#define BOUNDS_NONSTAT`
  in
  `src/parameters.h`

Note:
Software need to recompiled after those changes.

Note:
The output file is in the directory in which the solver was executed and is called `output`.

4.2 Matlab interface

Matlab can be invoked with the command `$ matlab`

Warning for advanced users: In the special case when **both** options `--enable-ipopt --enable-ipopt_shared` are set `Matlab` has to be invoked using the script provided in the main project directory `$ ./run_matlab` If Ipopt is enabled but no `--enable-ipopt_shared` is specified one does not have to use `$ ./run_matlab`.

When using `Matlab` interface the parameters of the models are set by providing arguments to the functions. Files `src/parameters.h` `src/parameters_profit.h` do not have any effect.
Documented examples of use are:

- `matlab/main_logit.m` for 'Logit model type 1' model
- `matlab/main_logit2.m` for 'Logit model type 2' model
- `matlab/main_cournot.m` for 'Cournot' model

Important:
In case of using `Matlab` the software does not need to be recompiled when changing model parameters.

User can choose between three implemented models:

- **Logit model type 1** (default option) - execute (in the main project directory)
  `$ ./choose_model_logit`
- **Logit model type 2 with two consumers** - execute (in the main project directory)
  `$ ./choose_model_logit2`
- `Cournot model` - execute (in the main project directory)
  `$ ./choose_model_cournot`

Note:
After changing the model the software needs to be recompiled with
`$ make`
`$ ./compile_matlab` or `$ ./compile_matlab2`

4.3 Advanced parameters

To set advanced parameters of the solver like multi-threading options, or debugging options one needs to edit `src/compoe/setup.h` and recompile the software. Some of those parameters are explained in the next section, others are documented in the `src/compoe/setup.h` file. Advanced parameters affect both `C++` and `Matlab` interfaces.

5 Customization

This section gives examples of advanced customization that can be done using source code distribution. It concerns all the solvers. For additional configuration options of AOE solver also read section 6.

5.1 Profit Function

To implement your own profit function one needs to implement method `eval` in the file `src/compoe/functionprofit.cpp`. Method `eval` takes a current state of the market and returns the value of profit for the company in each state, as well as other statistics. Look in the file for detailed documentation. If your profit function has closed form solution no other files than `src/compoe/functionprofit.cpp` need to be implemented. However if you need to solve for Nash Equilibrium to get the value of the profit you need to take further steps. To use different first order conditions than in the logit models you need to modify the following files:

- `src/compoe/profitFOC.cpp` - first order condition of equilibrium of the single period game.
- `src/compoe/profitJacobian.cpp` - Jacobian of the FOC vector. Matrix is stored in an one-dimensional vector, where each row is placed after another.
- `src/compoe/profitHessian.cpp` - Hessian of the Lagrangian for the *Ipopt* solver. This file needs to be implemented only if *Ipopt* is enabled. Ipopt solves a following constrained optimization problem.

$$\max 1$$

$$\text{s.t.} FOC(p) = 0$$

  Solution to this problem is the same as solution to the FOC. The solver needs a Hessian of the Lagrangian of this problem. One can also choose to enable `NUMERICAL_HESSIAN` option in the `src/compoe/setup.h` file. In this case user can disregard this file even when using *Ipopt*. It is however recommended to implement this file if closed form solution for the third derivatives of the profit function is known. For details of implementation look in the `src/compoe/profitHessian.cpp`

Note:
If you do not use `Ipopt` only the following files need to be implemented
`src/compoe/functionprofit.cpp`
`src/compoe/profitFOC.cpp`
`src/compoe/profitJacobian.cpp`
Hessian does not need to be provided.

Important:
Running `./change_model` scripts will replace all files mentioned in the section by appropriate files for the chosen model. Remember to save your work before choosing one of the predefined models.

5.1.1 Marginal Cost

To change the marginal cost one has to edit the macro `MACRO__MARGINAL_COST` in the file `src/compoe/marginalcost.h`

5.2 Transition Matrix

To implement the transition matrix one has three options depending on the characteristics of the transition process.

1. **Dense Matrix** - If your matrix is dense you
   should implement `src/compoe/compoe-tranprob_dense.cpp` file and set `#define TRANSITION_MATRIX 0` in `src/compoe/setup.h`
2. **Tri-diagonal matrix** - If your matrix is tridiagonal you
   should implement `src/compoe/compoe-tranprob_tridiag.cpp` file and set `#define TRANSITION_MATRIX 1` in `src/compoe/setup.h`
3. **General Sparse Matrix** - In future release

Important: Do not forget to appropriately adjust `src/compoe/compoe-optinv_MATRIX-TYPE.cpp` to incorporate changes in the transition dynamics. For numerical solvers one can implement `function` class that provides Global Newton-Rhapson routine.

5.3 Investment Cost

To change the investment cost one has to edit the macro `MACRO__MINVCOST`
in the file `src/compoe/investmentCost.h`

6 Aggregate shocks

This section contains additional configuration options for AOE solver. Note that even if user is interested only in AOE he should also read sections 3-5.

6.1 Compilation

Adding the options `--enable-aggregate` or `--disable-aggregate` to `configure` script enables and disables aggregate shocks solver.

Because solving for OE with aggregate shocks involves inverting big sparse matrices it is **highly** recommended to install MUMPS solver in the `ThirdParty/MUMPS` directory. Installation of MUMPS involves downloading the source code from `http://graal.ens-lyon.fr/MUMPS`, unpacking into `ThirdParty/MUMPS` and following the `README` file.

`configure` script will check if MUMPS is installed properly and print out the message on the screen.

6.2 C++ interface

Computation of OE with aggregate shocks might be turned on of by uncommenting the option `#define AGGR_OE` in `src/parameters.h` file. Uncommenting `#define BOUNDS_AGGR_OE` additionally computes bounds.

6.2.1 Customization

All the customization procedures for stationary OE apply to aggregate shocks equilibrium (AOE) solver. AOE solve uses the same configuration files. Additional customization options:

- Shock transition matrix is specified directly in `src/project.cpp` file
- The way how shocks affect the parameters of the model is specified in `src/compoe/compoe_aggr-setShock.cpp`.

Any of the above changes requires recompilation of the whole project using `make` in the top project directory.

6.3 Matlab interface

`compile_matlab` script automatically compiles Matlab interface for aggregate shocks if `--enable-aggregate` switch was specified when invoking `configure` script. No additional switches for `compile_matlab` are necessary.

Documented example of use can be found in `matlab/main_aggr_logit.m`.

6.3.1 Customization

All the customization procedures for stationary OE apply to aggregate shocks equilibrium (AOE) solver. AOE solver uses the same configuration files. Additional customization options:

- Shock transition matrix is supplied in the form of sparse matrix as argument of the function `compoe_aggregate_matlab`. Example of usage can be found in `matlab/main_aggr_logit.m`.
- The way how shocks affect the parameters of the model is specified in `src/compoe/compoe_aggr-setShock.cpp`. Recompilation of the whole project using `make` in the top project directory is necessary. Additionally we need to recompile Matlab interface using `compile_matlab` script.

A Installation of Ipopt

1. Download ipopt and follow the Ipopt documentation how to install sparse solver.
2. Go to main Ipopt directory
3. Run
   `$ CXXFLAGS="-fPIC" CFLAGS="-fPIC" ./configure --prefix=PATH_TO_THE_COMPOE/Thirdparty/Ipopt/ --enable-static`
   Replacing `PATH_TO_THE_COMPOE` with the path to the downloaded Oblivious Equilibrium Solver source code.
4. Compile
   `$ make`

5. Install
   `$ make install`

B Installation of MUMPS

1. Download MUMPS from `http://mumps.enseeiht.fr`
2. Unpack MUMPS source code into `PATH_TO_THE_COMPOE/Thirdparty/MUMPS/` directory
3. Follow MUMPS readme file and choose appropriate `Makefile.inc` (explained in readme file)
4. Compile MUMPS with `make`
5. Done.

References

Amestoy, P. R., I. S. Duff, J. Koster, and J.-Y. L'Excellent (2001): "A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling," *SIAM Journal on Matrix Analysis and Applications*, 23(1), 15–41.

Amestoy, P. R., I. S. Duff, and J.-Y. L'Excellent (2000): "Multifrontal parallel distributed symmetricand unsymmetric solvers," *Comput. Methods Appl. Mech. Eng.*, 184, 501–520.

Amestoy, P. R., A. Guermouche, J.-Y. L'Excellent, and S. Pralet (2006): "Hybrid scheduling for the parallel solution of linear systems," *Parallel Computing*, 32(2), 136–156.

Benkard, C. L., P. Jeziorski, B. Van Roy, and G. Y. Weintraub (2008): "Nonstationary Oblivious Equilibrium," Working Paper, Stanford University.

Wächter, A., and L. T. Biegler (2006): "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, 106(1), 25–57.

Weintraub, G. Y., C. L. Benkard, and B. Van Roy (2008a): "Computational Methods for Oblivious Equilibrium," Working Paper, Stanford University.

——— (2008b): "Markov Perfect Industry Dynamics with Many Firms," Forthcoming, *Econometrica*.