A User's Guide to the Programs for "Using Economic Theory to Guide Numerical Analysis: Solving for Equilibria in Models of Asymmetric First-Price Auctions"

Timothy P. Hubbard, René Kirkegaard, and Harry J. Paarsch

This user guide is intended to complement the AMPL and MATLAB code used to create, solve, and plot the examples in Hubbard, Kirkegaard, and Paarsch's forthcoming article in *Computational Economics*. We simply refer to this as "our paper" in going forward. A zipped folder containing the code as well as a copy of this user's guide can be downloaded from:

http://www.colby.edu/economics/faculty/thubbard/code/hkpcode.zip.

 Here is a formal citation of our work:

> Hubbard, Timothy P., René Kirkegaard, and Harry J. Paarsch: "Using Economic Theory to Guide Numerical Analysis: Solving for Equilibria in Models of Asymmetric First-Price Auctions," *Computational Economics*, forthcoming.

In this user's guide, we document programs we used to solve asymmetric auctions for our paper, provide an example for how the code can be executed via MATLAB if the user has AMPL on her/his computer, and direct users who might not have an AMPL license to alternative, albeit a little more tedious, ways in which the code can be used.

1 Programs Included in the Zipped Folder

There are both AMPL and MATLAB programs in the zipped folder. In short, MATLAB is used to set parameter values and interpret the AMPL output; AMPL is used to actually model the relevant optimization problem. The solver SNOPT is called within the AMPL driver file to actually solve the constrained nonlinear optimization problem. This solver can, of course, be changed easily and we had success using the KNITRO solver as well. Note, too, that a user could replace the MATLAB files with those from any other user-preferred language to allow for the problem set-up and plotting without changing the optimization routine contained in the AMPL code. If a user did this, it might be wise to change the AMPL code used to print the solution results as this is set to be MATLAB-friendly.

1.1 AMPL Programs

1. `fpsb_piecewise.dr`: an AMPL driver (commands) file. This file calls the data file `fpsb.dat` (described below) and the model file (described next) as well as defines the solver to be used and any solver-specific options. After solving the instance, this file also contains code to output the results in a format that is convenient for MATLAB. The output is printed to the file `fpsbsoln.m` (described below).

2. `fpsb_piecewise.md`: an AMPL model file which is the primary file to all of the code. It defines all variables as well as functions of variables (and parameters) involved in the optimization problem. It specifies the nonlinear objective and details all the constraints. If you have not used AMPL before, this is where you'll want to spend most of your time understanding how the code works. If you want to change the distributions used you will make adjustments to this code so that the CDFs and their associated PDFs are changed. Fortunately, AMPL is an intuitive language to read (and write) so the code should be fairly easy to look through once you've read our paper.

1.2 MATLAB Programs

1. `cheby_asymmetric_dr.m`: a MATLAB driver file which defines all the parameter and instance-specific values which are then output to `fpsb.dat` (described below) automatically. After that, the AMPL driver file `fpsb_piecewise.dr` is called to solve the problem. The AMPL driver file generates a MATLAB-friendly output file `fpsbsoln.m` which is then loaded by this MATLAB driver file. The solutions are then evaluated (with the help of the other MATLAB files described next) and figures are generated which plot the (inverse-)bid functions in one figure and the ratios discussed in our paper in another figure.

2. `evalchebypoly.m`: a MATLAB file which evaluates each of the Chebyshev polynomials (up to a user-specified degree) at a grid of points.

3. `calcexppayoff.m`: a MATLAB file which calculates the expected payoffs needed for the exogenous-endogenous ratio plot.

4. `genpiecewiseCDFs_onecross.m`: a MATLAB file which generates the piecewise CDF used in the example involving one crossing in our paper (specifically, this CDF is $F_2(v)$ in EXAMPLE 2).

5. `genpiecewiseCDFs_twocross.m`: a MATLAB file which generates the piecewise CDF used in the example involving two crossings in our paper (specifically, this CDF is $F_2(v)$ in EXAMPLE 3).

6. `fprintAmplParamCLSU.m`: a MATLAB file provided to us from Che-Lin Su (who modified the MATLAB-AMPL link provided by Ronald Hochreiter) which prints parameter values assigned within a MATLAB script in AMPL-ready format for a data file.

7. `fprintAmplParamCLSU_initial_guess.m`: a MATLAB file modified from the one Che-Lin Su provided us and described above to allow for initial guesses assigned within a MATLAB script to be converted to an AMPL-ready format for a data file.

1.3 Other Files

1. `fpsb.dat`: a data file describing the values of parameters assigned to a particular instance as well as an (optional) initial guess to the optimization routine. One of these files is included for the sake of completeness (in the event a user does not have access to MATLAB) but this file is automatically generated from the MATLAB file `cheby_asymmetric_dr.m` defined above. In fact, each time this MATLAB file is run, this data file gets replaced. Users with access to MATLAB should adjust settings by changing values in the beginning of the `cheby_asymmetric_dr.m` file, and then re-running this file. Users without MATLAB will need to adjust parameter values across instances by opening this data file in a preferred text editor, making changes to any values or parameters, and then saving the file.

2. `fpsbsoln.m`: a text file with a `.m` extension so that MATLAB reads it easily. This file is output from `fpsb_piecewise.dr` and taken as post-solution input by the primary MATLAB file (the MATLAB driver file) `cheby_asymmetric_dr.m`. One of these files (corresponding to the example in the next section) is included for the sake of completeness but this file gets rewritten each time `fpsb_piecewise.dr` runs.

3. `myampl.bat`: a batch file which informs MATLAB of the directory in which AMPL is installed on your machine. To have all of our code run through MATLAB, you must update this file first. Simply change the "`C:\Program Files (x86)\AMPL`" path to the directory containing `ampl.exe` on your machine.

2 An Example

In our paper, we discuss using a degree-three approximation first to obtain an initial guess at little cost. Consider solving EXAMPLE 2 from our paper by assuming a degree-three approximation. We demonstrate use of the code from a best-case scenario (the setup we used) but give other usage suggestions in the next section. To run this instance, complete the following steps:

1. open `cheby_asymmetric_dr.m` in MATLAB;
2. set
   `dists = 'piecewise1';`
   (the 1 is for one crossing of the CDFs) on line 16 of `cheby_asymmetric_dr.m`;
3. set
   `d = 3;`
   (for degree three polynomials) on line 33 of `cheby_asymmetric_dr.m`;
4. run `cheby_asymmetric_dr.m`.

After running this code, the program generates the following output to the MATLAB Command Window:

```
ampl fpsb_piecewise.dr
SNOPT 7.2-8 : outlev=1
feas_tol=1e-6
SNOPT 7.2-8 : Optimal solution found.
33 iterations, objective 0.04335718302
Nonlin evals: obj = 16, grad = 15, constrs = 16, Jac = 15.
_ampl_time = 0.109201

 _total_solve_time = 0.0780005
```

There will also appear a new file in the directory called `fpsbsoln.m` (described above) which will get called by MATLAB to generate the following two figures (which of course, are a bad approximation to the true solution for the reasons discussed in our paper):

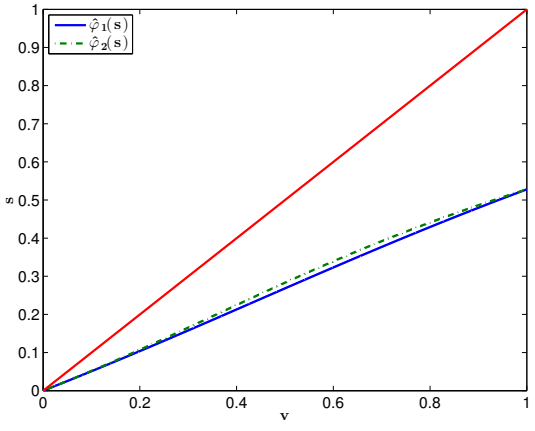

Figure 1: (Inverse-)Bid Functions ($d = 3$)

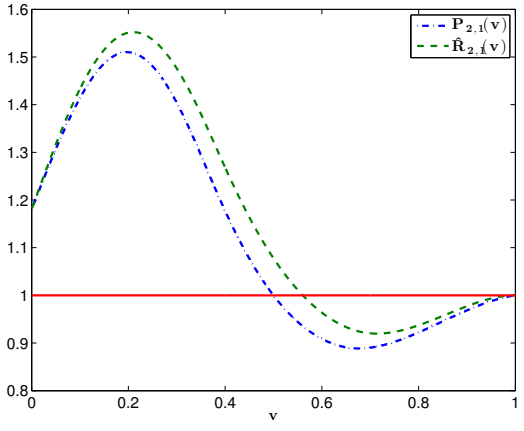

Figure 2: Example of Ratios from Our Paper

The code, as it stands allows for both of the examples in our paper to be solved (a user need simply specify `piecewise1` or `piecewise2` in step 2. of the above example). Changing other algorithm-related settings

such as the degree of the polynomials or the number of points that enter the objective should be obvious from the commented code in `cheby_asymmetric_dr.m`. Changing the problem to another asymmetric auction setting will require some adjustment to the AMPL file `fpsb_piecewise.md`. Specifically, users may want to try different distributions by adjusting the CDFs specified on line 85, corresponding PDFs specified on line 100, and the value of the PDFs at the high valuation on line 155 of `fpsb_piecewise.md`. Note, too, that while these line reference numbers seem overwhelmingly high, it is because the code is formated to be user-friendly by having separated out various commands across different lines, inserting comments throughout the code, etc.

3 Other Comments

MATLAB is certainly more prevalent within economics than is AMPL, although the scholars involved with the Initiative for Computational Economics (http://ice.uchicago.edu/) have been advocating AMPL and GAMS as attractive modeling languages for some time now. We used AMPL to gain access to better solvers for the nonlinear constrained optimization problem described in our paper. We found these were far more successful for solving asymmetric auctions than the built-in solvers that MATLAB provides in its Optimization Toolbox (e.g., `fmincon`). Given all this, some users may not currently have AMPL on their machine. Fear not, there are some options still available:

1. AMPL Student Edition can be downloaded for free from

   http://www.ampl.com/DOWNLOADS/details.html.

   The Student Edition limits the size of problems to 300 constraints and objectives and 300 variables. This may be sufficient for many asymmetric auction problems depending on the number of players, the degree of the approximating polynomials, the number of grid points, etc. Of course, full AMPL licenses and compatible solvers can be purchased from http://www.ampl.com/vendors.html.

2. AMPL code can be executed for free (without limits on the number of constraints, objectives, or variables) from the NEOS Server. For example, to use AMPL code which calls the SNOPT solver, go to: http://www.neos-server.org/neos/solvers/nco:SNOPT/AMPL.html. Submitting code to the NEOS Server requires your job wait in a queue and not exceed a five-minute runtime limit (this should not be an issue for solving an asymmetric auction example). We recommend you read the materials at http://neos-guide.org/NEOS/index.php/NEOS_Wiki if you plan on using the NEOS Server as this may require minor modifications to the AMPL driver (commands) file `fpsb_piecewise.dr` (by "minor" we mean commenting out a few lines of the code as it currently stands).

We appreciate your interest in our paper and, presumably, given you've made it this far, your interest in applying our code to other problems. Should you have comments, questions, or suggestions please feel free to email Tim at timothy.hubbard@colby.edu. Good luck!