

A User’s Guide to the Programs for “Investigating Bid Preferences at Low-Price, Sealed-Bid Auctions with Endogenous Participation”

Timothy P. Hubbard and Harry J. Paarsch

January 2009

In this guide, we describe the programs used to approximate the inverse bid functions of players at low-price, sealed-bid auctions, without and with a preference policy. Two sets of programs are included, one for models without entry and one for models with endogenous entry—a positive entry fee that bidder must pay to participate at the auction. The programs were used by Hubbard and Paarsch (2009); we recommend reading that paper for a complete description of the model, the numerical approach, and the results of an application. We used the programming language AMPL to solve for the approximate Bayes–Nash equilibrium inverse bid functions. All code can be run without downloading any software by accessing the NEOS server.¹ Because AMPL has no graphical interface, we provide MATLAB programs that create graphs. These MATLAB programs are for convenience; a program in any standard language with graphing capabilities could be used instead. We also provide MATLAB programs that call AMPL directly, so users familiar with MATLAB can easily take advantage of the benefits of AMPL.

1 Algorithm

To approximate the equilibrium bid functions at an asymmetric auction, we employed the third method suggested by Bajari (2001), embedding it in a Mathematical Programming with Equilibrium Constraints (MPEC) approach. The MPEC approach has been advocated by Su and Judd (2008) to estimate structural econometric models. We assume that the (inverse) bid functions can be represented by Chebyshev polynomials; we choose the coefficients so that the first-order conditions (FOCs) characterizing Bayes–Nash equilibrium bidding behaviour as well as the associated theoretical restrictions hold approximately. The polynomial coefficients and endogenous variables are chosen to minimize a quadratic objective, subject to theoretical constraints concerning the equilibrium bid functions. By formulating the problem as a constrained optimization problem, we can exploit a direct-optimization approach that treats the endogenous variables and polynomial coefficients jointly.

In the discussion below, we use notation that is consistent with that used in Hubbard and Paarsch (2008); details can be found there. Represent the inverse

¹The NEOS server can be accessed by going to <http://www-neos.mcs.anl.gov/>. The server is free to use. It takes appropriately formatted input and allows the user to access a number of different linear and nonlinear programming solvers.

bid function of a player of class j by the polynomial

$$\beta_j^{-1}(b_t; \alpha_j, \underline{b}, c_1^*, c_2^*) = \underline{b} + \sum_{m=0}^M \alpha_{j,m} \mathbb{T}_m[x(b_t)], \text{ for } j = 1, 2; t = 1, \dots, T \quad (1)$$

where $x(\cdot)$ lies in the interval $[-1, 1]$ and where, for clarity, we have explicitly defined it as a transformation of the bid under consideration. Here, $\mathbb{T}_m(\cdot)$ denotes the m^{th} Chebyshev polynomial and the vector α_j collects the polynomial coefficients for a firm of class j , with T representing the number of bids considered in the algorithm. Collect the vectors α_1 and α_2 in α . The points $\{b_t\}_{t=1}^T$ are chosen to be the Chebyshev points on the interval $[\underline{b}, p_0]$ for nonpreferred players or the interval $[(1 + \rho)\underline{b}, p_0]$ for preferred players, where p_0 denotes the buyer's reservation price ceiling. Typically, p_0 equals $\bar{c}$, either because no price ceiling exists or because it is nonbinding. The transformation $x(\cdot)$ simply maps the Chebyshev points from the relevant interval to the interval $[-1, 1]$, which is the domain over which Chebyshev polynomials are defined.

The goal of the numerical approach is to choose the Chebyshev coefficients to minimize the sum of squared differences involving the FOCs evaluated at the grid points, subject to the known theoretical restrictions discussed in detail in the paper. The theoretical restrictions improve numerical performance when solving for the other endogenous variables of the model; *i.e.*, the low bid $\underline{b}$, the entry threshold for class 1 bidders c_1^* , and the entry threshold for class 2 bidders c_2^* . Specifically, we apply the MPEC approach, writing the solution to the problem as

$$\left(\hat{\alpha}, \hat{\underline{b}}, \hat{c}_1^*, \hat{c}_2^* \right) = \arg \min_{(\alpha, \underline{b}, c_1^*, c_2^*)} \sum_{i=1}^2 \sum_{j=1}^T [g_j(b_t)]^2$$

subject to

$$\begin{aligned} [p_0 - \beta_1^{-1}(p_0)] (1 - F[\beta_1^{-1}(p_0)])^{n_1-1} \left(1 - F \left[\beta_2^{-1} \left(\frac{p_0}{1 + \rho} \right) \right] \right)^{n_2} &= \kappa \\ [p_0 - \beta_2^{-1}(p_0)] (1 - F[\beta_1^{-1}(p_0)])^{n_1} (1 - F[\beta_2^{-1}(p_0)])^{n_2-1} &= \kappa \\ \beta_1^{-1} [(1 + \rho) \underline{b}] &= \underline{c} \\ \beta_2^{-1} (\underline{b}) &= \underline{c} \\ \beta_1^{-1} (p_0) &= c_1^* \\ \beta_2^{-1} (p_0) &= c_2^*. \end{aligned}$$

where $g_j(\cdot)$ is the FOC of a class j player evaluated at a specific bid from the grid space and ρ is the preference rate used by the buyer to discount bids from class 1 firms. In this guide, and in our paper, we assumed that more than one preferred player exists at the auction. Note, too, that, in the specification above, $\beta_j^{-1}(\cdot)$ is the approximated inverse-bid function, as specified in equation (1). Also, two other constraints, which are often left unstated, discipline the bid functions: monotonicity and rationality. Rationality requires players to bid

higher than their costs, so

$$b_t \geq \beta_j^{-1}(b_t) \quad \forall j = 1, 2; \quad t = 1, \dots, T.$$

Monotonicity requires players with lower (higher) costs make lower (higher) bids, so

$$\beta_j^{-1}(b_t) \geq \beta_j^{-1}(b_{t-1}) \quad \forall j = 1, 2; \quad t = 2, \dots, T.$$

The advantage of imposing these constraints is that they help discipline the choice of the polynomial coefficients, ensuring that, at least at a finite number of points T , the approximated (inverse) bid functions are well-behaved. The AMPL Student Version allows only 300 constraints, however, which limits the number of points that can be constrained.

2 Listing of Programs and Uses

Below, we list the programs contained in the folder `HPprefcode.zip` that is posted online. In the following, the term `<dist>` can take on one of the following strings: `{exp, norm, unif, wbl}`, which correspond to the exponential, normal, uniform, and Weibull distribution, respectively. While using AMPL has many benefits, one downside is that function calls cannot be easily incorporated into scripts (at least at the time of this writing). Thus, we have different AMPL files for each distribution and depending on whether the entry fee is positive.

- AMPL programs to solve numerically for the inverse bid functions:
 - `ibf_cheby_ecost_<dist>.dr`—driver file (commands file) for solving for the inverse bid functions when entry is endogenous and bidders draw costs from the specified distribution;
 - `ibf_cheby_ecost_<dist>.md`—model file defines the nonlinear constrained optimization problem when entry is endogenous and bidders draw costs from the specified distribution;
 - `ibf_cheby_ecost_<dist>.dat`—an example data file defining the parameter values for a specific case when entry is endogenous and bidders draw costs from the specified distribution;
 - `ibf_cheby_noecost_<dist>.dr`—driver file (commands file) for solving for the inverse bid functions when entry is costless and bidders draw costs from the specified distribution;
 - `ibf_cheby_noecost_<dist>.md`—model file defines the nonlinear constrained optimization problem when entry is costless and bidders draw costs from the specified distribution;
 - `ibf_cheby_noecost_<dist>.dat`—an example data file defining the parameter values for a specific case when entry is costless and bidders draw costs from the specified distribution.

- MATLAB programs to invoke AMPL using a system call and/or graphing bid functions:
 - `approxbf_ecost.m`—calls AMPL directly from MATLAB and allows the user to change parameter values in MATLAB for the cases with a positive entry fee;
 - `approxbf_noecost.m`—calls AMPL directly from MATLAB and allows the user to change parameter values in MATLAB for the cases with no entry fee;
 - `plotbfcheby_ecost.m`—graphs the bid functions given the polynomial coefficients, low bid, threshold costs, and model parameters for the case of a positive entry fee;
 - `plotbfcheby_noecost.m`—graphs the bid functions given the polynomial coefficients, low bid, and model parameters for the case of no entry fee;
 - `invbidcheby.m`—evaluates the inverse bid function for a certain player given the approximated Chebyshev coefficients, low bid, and model parameters for that case;
 - `fprintAmplParamCLSU.m`—used to generate the AMPL data file when AMPL is invoked through MATLAB ;
 - `tr<dist>cdf.m`—computes the value of the truncated cumulative distribution function at a vector of points given the parameters of the distribution (the user does not need to specify the distribution here as the code will automatically recognize which distribution is assumed);
 - `tr<dist>df.m`—computes the value of the truncated probability density function at a vector of points given the parameters of the distribution (the user does not need to specify the distribution here as the code will automatically recognize which distribution is assumed);
 - `neosplot.m`—graphs the bid functions when the user receives the approximated solution for a given case from the NEOS server.
- Other programs:
 - `myampl.bat`—ensures the AMPL directory is added to your “path” variable.

3 Approximating the Inverse Bid Functions

The AMPL code can be processed a number of different ways. Below, we describe three ways you can use the code for free, two ways if you do not have access to MATLAB . We present the alternatives because some users may want to use the code frequently, while others may only run it occasionally. We also discuss how the algorithm can be adjusted for different parameter values. Users should first download all files from

<http://myweb.uiowa.edu/tphubbar/code/HPprefcode.zip>

and unpack the zip file into a directory. How you will use the code will determine where this directory should live on your computer. We discuss further details regarding each of the options below.

If you are interested in using the code frequently, then we recommend you download AMPL Student Edition (which is what we used) from AMPL's website

<http://www.ampl.com/DOWNLOADS/>.

The language, as well as many of the AMPL-compatible solvers (including SNOPT and MINOS), can be downloaded from the website for free.² We used the solver SNOPT, version 6.1-1, for our work; users should download SNOPT and save the solver to the same directory to which AMPL was stored.

3.1 For Frequent Users with MATLAB

After downloading the AMPL Student Edition, the SNOPT solver, and unpacking the zip file containing the code, open the file `myampl.bat`, and you will see the following two lines of code:

```
set path=%path%;C:\Program Files\amplcml  
ampl %1
```

Change the path variable to the directory that you have downloaded the AMPL Student Edition; *i.e.*, replace `C:\Program Files\amplcml` with the full path of the directory containing `ampl.exe` and `snopt.exe`.

To compute the bid functions at an auction, open either

```
approxbf_ecost.m
```

or

```
approxbf_noecost.m
```

in MATLAB. The `m`-file(s) can be executed directly from the MATLAB editor. These files will automatically generate a new AMPL data file for the case being considered, call AMPL, approximate the (inverse) bid functions for each player and the other endogenous variables, and then graph the bid functions. Changing parameter values, or the case under consideration, is easy as the user can simply change the value for a given parameter from the MATLAB editor and re-run the script. For example, the following changes are possible from within

```
approxbf_ecost.m
```

²Users who have had experience with AMPL/GAMS may be familiar with the Kestrel client/server. Downloading the Kestrel client will allow you to access the NEOS server directly from your computer, as if you had all the solvers on the NEOS server installed on your computer. While we do not discuss processing the code through Kestrel, you will see in our driver files, which we represent by the filename extension `.dr`, that you can remove the comment indicators from the beginning of the lines related to the Kestrel call to access the NEOS server directly.

or

`approxbf_noecost.m`

without the user having to change anything else:

- cost distribution—change the string variable `ind` to take one of the values “exponential”, “normal”, “uniform”, “weibull”;
- preference rate—change the value of the variable `rho` to be greater than or equal to zero, although convergence may be difficult to obtain for extreme policies, such as $\rho \geq 0.5$, depending on the cost distribution and other parameter values;
- order of Chebyshev polynomials—change the value of the variable `ncoefp` (`ncoefnp`), which corresponds to the number of Chebyshev coefficients for preferred (nonpreferred) players;
- number of points at which the FOCs are evaluated—change the value of the variable `npts`;
- number of preferred (nonpreferred) bidders—change the value of the variable `npref` (`nnonpref`), however, the number of preferred bidders should be greater than one;
- entry fee (for `approxbf_ecost.m` only)—change the value of the variable `ecost` to be something greater than zero, although if the cost is set too high then no bidders will enter the auction;
- support of the cost distribution—change the value of the variable `clow` or `chigh`, where the high cost should be greater than the low cost;
- cost distribution-specific parameters—change the value of the variable `lambda` (for the exponential distribution), `mu` or `sigma` (for the normal distribution), or `wbla` or `wblb` (for the Weibull distribution).

Limits on the number of variables and/or constraints will restrict the (joint) choices of some of the variables such as `npts`, `ncoefp`, `ncoefnp`, and so forth. However, if this is an issue, then AMPL will produce an error message and the user can change these parameter values. The solver SNOPT is automatically set to let the user know, through the MATLAB Command Window, whether convergence obtained. If convergence did not obtain (which is rare and probably because the parameters are incompatible with an equilibrium where both classes of bidders are active), then a different guess can be used by defining explicitly vectors `polyp` and `polynp` before the AMPL data file is created in

`approxbf_ecost.m`

or

`approxbf_noecost.m`.

3.2 For Frequent Users without MATLAB

After downloading the AMPL Student Edition, the SNOPT solver, and unpacking the zip file containing the code, move all files to the directory containing `ampl.exe` and `snopt.exe`.

Open AMPL (by running the executable file `ampl.exe`). The driver file (commands file) for each case automatically calls the model and data files, so you just need to load one of the driver files into AMPL. To do this, at the AMPL command prompt, type

```
ampl: reset; include <file>.dr;
```

where `<file>` corresponds to either

```
ibf_cheby_ecost_<dist>
```

or

```
ibf_cheby_noecost_<dist>
```

and `<dist>` takes on one of the following strings: `{exp, norm, unif, wbl}`. Hit Enter. The program will output some information on the solver being used (the default is SNOPT with a tolerance of 1e-12) and notify you if an optimal solution obtained. After solving the model, the approximated solution will be printed to the screen and two new files will be automatically generated:

```
info_ecost_<dist>.out
```

and

```
results_ecost_<dist>.m,
```

where `<dist>` corresponds to the cost distribution specified. These files can be ignored as they are called from MATLAB to receive the output from AMPL. Since MATLAB is not being used under this alternative, the files are irrelevant.

Changing parameter values, or the case under consideration, is fairly simple, but requires the user to open one of the files in a text editor and to change the parameter value, manually. Specifically, the user should open the file

```
ibf_cheby_ecost_<dist>.dat
```

or

```
ibf_cheby_noecost_<dist>.dat
```

for the cost distribution under consideration and change the appropriate parameter. For example, the following changes are possible without the user having to change anything else:

- cost distribution—the user can simply run a different driver file from the command prompt by again typing

```
ampl: reset; include <file>.dr;
```

for a new filename;

- preference rate—open

```
ibf_cheby_ecost_<dist>.dat
```

or

```
ibf_cheby_noecost_<dist>.dat
```

and change the value of the variable `rho` to be greater than or equal to zero, although convergence may be difficult to obtain for extreme policies, such as $\rho \geq 0.5$, depending on the cost distribution and other parameter values;

- order of Chebyshev polynomials—open

```
ibf_cheby_ecost_<dist>.dat
```

or

```
ibf_cheby_noecost_<dist>.dat
```

and change the value of the variable `ncoefp` (`ncoefnp`), which corresponds to the number of Chebyshev coefficients for preferred (nonpreferred) players;

- number of points the FOCs are evaluated at—open

```
ibf_cheby_ecost_<dist>.dat
```

or

```
ibf_cheby_noecost_<dist>.dat
```

and change the value of the variable `npts`;

- number of preferred (nonpreferred) bidders—open

```
ibf_cheby_ecost_<dist>.dat
```

or

`ibf_cheby_noecost_<dist>.dat`

and change the value of the variable `np` (`nnp`), however, the number of preferred bidders should be greater than one;

- entry fee (for `ibf_cheby_ecost_<dist>.dr` only)—open

`ibf_cheby_ecost_<dist>.dat`

and change the value of the `ecost` to be something greater than zero, although if the cost is set too high then no bidders will enter the auction;

- support of the cost distribution—open

`ibf_cheby_ecost_<dist>.dat`

or

`ibf_cheby_noecost_<dist>.dat`

and change the value of the variable `clow` or `chigh`, where the high cost should be greater than the low cost;

- cost distribution-specific parameters—open

`ibf_cheby_ecost_<dist>.dat`

or

`ibf_cheby_noecost_<dist>.dat`

and change the value of the variable `lambda` (for the exponential distribution), `mu` or `sigma` (for the normal distribution), or `wbla` or `wblb` (for the Weibull distribution).

Limits on the number of variables and/or constraints will restrict the (joint) choices of some of the variables such as `npts`, `ncoefp`, `ncoefnp`, and so forth. However, if this is an issue, then AMPL will produce an error message and the user can change these parameter values. The solver SNOPT is automatically set to let the user know, through `ampl.exe`, whether convergence obtained. If convergence did not obtain (which is rare and probably because the parameters are incompatible with an equilibrium where both classes of bidders are active), then a different guess can be used by stating explicitly the values AMPL should consider for initial values. To provide a different initial guess, open the file

`ibf_cheby_ecost_<dist>.dr`

or

```
ibf_cheby_noecost_<dist>.dr
```

and insert, before the line `solve;` is executed, the following:

```
let polyp[1] := xp1;
let polyp[2] := xp2;
.
.
.
let polyp[ncoefp] := xpn;
let polynp[1] := xnp1;
let polynp[2] := xnp2;
.
.
.
let polynp[ncoefnp] := xnpn;
```

where `xpi` (`xnpi`) corresponds to the guess for the i^{th} coefficient of the polynomial approximating the preferred (nonpreferred) players' (inverse) bid function. Of course, an initial guess need not be specified for all coefficients, but can be specified for a subset of the coefficients as well.

3.3 For Occasional Users

If you are interested in a specific case, then it may be most convenient to download the code and send it to the NEOS server for processing. The server takes appropriately-formatted optimization problems and solves them automatically. An e-mail containing the output is then sent to a specified address. To use this option, first unpack the zip file containing the code into any directory on your computer. If you want to consider a model with endogenous entry, then open the file

```
ibf_cheby_ecost_<dist>.dr
```

and comment-out the following lines:

```
#model ibf_cheby_ecost_<dist>.md;
#data ibf_cheby_ecost_<dist>.dat;
```

by placing the # (pound) sign at the beginning of the line. Likewise, if you want to consider a model with no entry fee, then open the file

```
ibf_cheby_noecost_<dist>.dr
```

and comment-out the following lines:

```
#model ibf_cheby_noecost_<dist>.md;
#data ibf_cheby_noecost_<dist>.dat;
```

by placing the # (pound) sign at the beginning of the line. If changes were necessary, then make sure you saved them. The only other requirement is to upload the code to the NEOS server. To do this, follow these instructions:

1. Click the link below to get access to the SNOPT solver that takes AMPL input on the NEOS server
<http://neos.mcs.anl.gov/neos/solvers/nco:SNOPT/AMPL.html>

2. Click the Browse link for the Model File and upload the file

`ibf_cheby_ecost_<dist>.md`

or

`ibf_cheby_noecost_<dist>.md=`

from the directory you saved the code;

3. Click the Browse link for the Data File and upload the file

`ibf_cheby_ecost_<dist>.dat`

or

`ibf_cheby_noecost_<dist>.dat`

from the directory you saved the code;

4. Click the Browse link for the Commands File and upload either the file

`ibf_cheby_ecost_<dist>.dr`

or

`ibf_cheby_noecost_<dist>.dr`

from the directory you saved the code;

5. Enter your e-mail address in the box and click the Submit to NEOS link.

After completing this process, the NEOS server will assign your problem to the solver queue and, once the SNOPT solver processes the code, the results will be e-mailed to you. The problem takes about a second to run, but the time to receive the e-mail can be long depending on server traffic. The e-mail will contain the output from running the programs and users with MATLAB can run the programs included with the code to graph the bid functions.

Specifically, to graph the approximated solution open the file `neosplot.m` and change the parameters of the case (the entry fee, preference rate, and distribution) to match those of the case of interest. Paste the output from the NEOS server, from the line starting `polyp(1) = x;` (where `x` corresponds to the first approximated coefficient of the preferred bidders' polynomial) until the end of the e-mail, into the m-file after the comment line

`% paste approximated solution from NEOS server here`

The code is set to call the appropriate graphing function automatically given the output. You can then process the `m`-file and the function `plotbfcheby_ecost.m` or `plotbfcheby_noecost.m` will be called and the bid functions graphed.

Changing parameter values, or the case under consideration, is fairly simple, but requires the user to open one of the files in a text editor, to change the parameter value manually, and then to resubmit the code to the NEOS server. Specifically, the user should open the file

`ibf_cheby_ecost_<dist>.dat`

or

`ibf_cheby_noecost_<dist>.dat`

for the cost distribution under consideration and change the appropriate parameter. Changes can be made in the exact same way as discussed in the preceding subsection, with the exception that, after changes are made, rather than processing the code on your local machine the user should repeat the instructions outlined above for submitting the code to the NEOS server.

References

- [1] Bajari, Patrick (2001): “Comparing Competition and Collusion: A Numerical Approach,” *Economic Theory*, 18, 187–205.
- [2] Hubbard, Timothy P. and Harry J. Paarsch (2009): “Investigating Bid Preferences at Low-Price, Sealed-Bid Auctions with Endogenous Participation,” *International Journal of Industrial Organization*, 27(1), 1–14.
- [3] Su, Che-Lin and Kenneth L. Judd (2008): “Constrained Optimization Approaches to Estimation of Structural Models.” Typescript. Evanston: Northwestern University.