Agricultural and Applied Economics 637
Applied Econometrics II

Assignment 2

Determining Optimal Parameter Values in Nonlinear Regression Models: Two Step and Grid Search Examples

Due: February 20, 2018
80 Total Pts.

[**Note**: When you are ready to hand-in your assignment, place your code, output file(s), etc, in the appropriate Dropbox in the Learn@UW course website. If your output contains a listing of iterative results, edit the output file so only the first and last few iterations are shown. *There is no need to show the results of 100's of estimation iterations.*]

As you will discover throughout the semester, unlike estimating parameters of the classical regression model (CRM), finding *optimal* parameters of a nonlinear (in parameter) regression model usually requires an iterative search process. The definition of what constitutes *optimal* obviously depends on the objective function used as a guide in determining the preferred parameter values. For example, are you trying to find parameter values that minimize the sum of squared differences between predicted and actual dependent variable values (i.e., the sum of squared errors, [SSE's]) or are you trying to find parameter values that maximize the joint probability of obtaining the observed dependent variable values (i.e., the dependent variable joint probability density function, PDF)?

As we noted in class and in contrast to the CRM, a nonlinear regression model requires you to make an initial guess as to parameter values and then to check to determine, if indeed, these parameter values are *optimal*. If *not optimal* then the estimation algorithm you are using should have specific procedures for obtaining new updated parameter estimates. This parameter updating is an iterative process where for each vector change, the parameters are *improved*. Improvement is defined relative to changes in the algorithm's objective function.

1. (**30 pts**) Suppose you have 45 years of data concerning Australian wool exports. The Australian Wool Association (AWA) has provided you 45 annual time series observations of metric tonnes (1,000's) of wool exported (**Wool_Q$_t$**), the average marketing year Australian wool export price (**Wool_P$_t$**) and the average U.S. wholesale price of synthetic fabric (**Syn_P$_t$**), given that the U.S. is a major manufacturer of synthetic fibers. Again the dependent variable is measured in 1,000 MT and the two price series are measured in Australian $ per MT. The wool data is contained in the file ***wool_assign_2_18.xls***. This market data can also be obtained from the data section of the class website.

   Assume you want to estimate the following stochastic relationship between annual Australian wool exports, its export price and the price of synthetic cloth material:

$$\text{Wool\_Q}_t = \beta_0 \text{Wool\_P}_t^{\beta_W} \text{Syn\_P}_t^{\beta_S} \text{Time}^{\beta_T} \exp(\varepsilon_t) \quad t=1970,\ldots,2014 \qquad (1)$$

   where β_0, β_W, β_S and β_T are coefficients whose values are to be estimated, The trend variable, Time =1,…,45, and ε_t is the error term for the t^{th} observation. We include the time trend variable to account for any systematic change in wool exports. For estimation, you use a logrithmic transformation of (1) so the 4 parameters can be estimated the CRM with an additive error.

   Since you are using annual data, you suspect that you have an AR(1) error structure where $\varepsilon_t = \rho\varepsilon_{t-1} + \nu_t$. The term ν_t represents an error term where $\nu_t \sim (0, \sigma^2_\nu)$ (i.e., homoscedastic/non-autocorrelated). The coefficient ρ quantifies the relationship between consecutive error terms. In addition to the β coefficient vector, ρ and its standard error also need to be estimated.

   There are a number of ways to estimate the parameters of (1) while recognizing the possibility of an AR(1) error structure. Remember that the presence of autocorrelated errors represents an efficiency issue under CRM estimation. That is, CRM estimated parameters are still unbiased. In contrast, CRM generated standard errors are no longer efficient. In fact, the parameter covariance matrix, $\Sigma_\beta = \sigma^2(X'X)^{-1}$ is a ***biased estimate*** of this covariance matrix when there is error autocorrelation.

For a review of the AR(1) error structure and associated parameter estimation issues, refer to p. 327-332, Chapter 8, Judge et al (1988), and p. 384-392, Chapter 9. Using this information I would like you to estimate the parameters of equation (1) assuming an AR(1) error structure via what is often referred to as, the two-step ***Estimated (Feasible) Generalized Least Squares*** **(EGLS** or **FGLS)** estimation procedure. The general two step procedure to obtain the K+1 paramters is outlined in the above pages. Specifically, check out equations [9.5.36] – [9.5.41] in pages 391-392 for an overiewof the estimation procedure. For an example two-step alogorithm for parameter estimation with AR(1) errors refer to this summary document.

Instead of developing this estimation using just raw code, you decide you want to develop a general **AR(1) estimate function** that uses the *Estimated General Least Squares* (EGLS) procedure to obtained parameter estimates not only for this problem but for **any model of any size** so long as there is a possible AR(1) error structure. The general flowchart contained in the summary document should give you a good idea as to how to organize your function.

You want to design the function such that its incorporation into MATLAB code is obtained via something like the following call out to this function from a problem specific command file:

[**Est_Betas**,**Est_rho**,**VCV**,**SE_rho**]=***FI_LI_AR1***(**Rhsvar**,**Depend**,**Int_flag**);

where: **FI** = your first initial;
LI = your last initial;
Est_betas = the (**K***×1) vector of estimated regression coefficients;
Est_rho = the estimated value of ρ, the correlation coefficient;
VCV = the (**K*** × **K***) estimated regression coefficient covariance matrix;
SE_rho = the estimated ρ coefficient standard error;
Rhsvar = the (**T** × **K***) matrix of exogenous explanatory variables;
Depend = the (**T** × **1**) vector of dependent variable values;

$$\textbf{Int\_flag} = \begin{cases} 1, \text{if intercept in the model} \\ 0, \text{otherwise} \end{cases}; \text{ and}$$

K* = **K** + **Int_flag (**i.e., **K** = the number of exogenous variables in eq.1 excluding any intercept);

Using the notation of Judge et al(1988), the EGLS function (i.e., *FI_LI_AR1*) you develop needs to calculate the (T × T) error covariance matrix, $\boldsymbol{\Psi}$, *regardless of the number of observations*. What I have done in the past is to some type of ***for***, ***while***, ***do_until***, etc. loop to build this matrix whose elements vary depending on the amount of time between two error terms.

[***Hint:*** *Note the symmetry and time dependent structure of* $\boldsymbol{\Psi}$ *as shown in Judge et al (1988) p.387, eq. 9.5.19,. Let* $\boldsymbol{\Psi}^*$ *equal the full matrix prior to dividing by* $(1-\rho^2)$*, i.e.,* $\boldsymbol{\Psi} = [\boldsymbol{\Psi}^*/(1-\rho^2)]$ *You only have to build the upper triangle portion of* $\boldsymbol{\Psi}^*$.

Let ***vvv*** *be the upper triangular portion of* $\boldsymbol{\Psi}^*$ *(including a diagonal consisting of T ones) and vvv' its transpose. What does* $(\boldsymbol{vvv}+\boldsymbol{vvv'})$ *equal? How can this summation be transformed so that it is equal to* $\boldsymbol{\Psi}^*$*?*

When developing the code to calculate $\boldsymbol{\Psi}^*$ *remember the general structure is the same for 5 observations as 50 observations so try to first estimate a (5 x 5)* $\boldsymbol{\Psi}^*$ *matrix by first developing this code as standalone and assuming a* $\boldsymbol{\rho}$ *value of 0.5. In the final version the actual value of* $\boldsymbol{\rho}$ *will be determined by the function.*]

Remember that under AR(1) unbiased Generalized Least Squares (GLS) parameter estimates are obtained via the following: $\beta_G=(X'\Psi X)^{-1}X'\Psi^{-1}y$. The GLS parameter covariance matrix is obtained from $\sigma^2(X'\Psi^{-1}X)^{-1}$.

The ***FI_LI_AR1*** function:

- Undertake CRM estimation to enable you to estimate ρ as outlined in the above AR(1) summary document;
- Display the typical CRM regression results you displayed in the function you created in Assignment #1;
- Undertake an *Estimated Generalized Least Squares* regression model where you create and save to an output file, a table showing the typical regression estimation results *similar in form* to the results shown for your estimation including the ρ coefficient and its standard error. Make sure you place column names over the associated column of the regression results

matrix and you identify the exogenous variable is associated with each results matrix row.

- Provide the R^2 and adjusted R^2 values for the transformed model used in estimation.
- Show the results of testing the null hypothesis that you indeed have an AR1 Error Structure.

(i) (**20 pts**) Apply the ***FI_LI_AR1*** function to the estimation of the ***logrithmic transformation*** of equation (1). Undertake a formal hypothesis test of whether an AR1 error structure exists?

(ii) (**5 pts**) Do the estimated results make sense with respect to the value of the own and cross price effects? That is, are they indivudally:

a. Statistically significant?
b. Statistically different from -1.0 and 1.0, respectively?

(iii) (**5 pts**)Undertake a test of the following null hypothesis:

H_0: Wool own-price elasticity = − Wool synthetics-price elasticity

In answering the above, provide the reason(s) for your answer.

2. (**30 pts**) As noted in the introduction, when using iterative methods to obtain SSE minimization paremeter estimates, if it is determined that the current paramater vector estimate does not generate a SSE minimum value, then whatver estimation algorithm you are using, it should have specific procedures for obtaining new updated parameter estimates. This parameter updating is the iterative process where, for each vector change, parameters are adjusted, in a logical manner, to improve the algorithm's objective function across iteration.

   We have started reviewing alternative methods for conducting the above iterative process. One method we will not be reviewing to any large degree is what is known as the ***Grid Search*** method given its computational intensiveness. For each parameter you divide a *reasonable* range of values into a **finite** set of discrete values and then evaluate the impact of parameter value combinations on the alogorithm's objective function (e.g., the SSE, log-likelihood function (LLF), etc).

Using the same data as that used in Question 1, assume you want to estimate the following relationship between annual Australian wool exports, its export price and the price of synthetic cloth material.

$$Wool\_Q_t = Wool\_P_t^{\beta_W} Syn\_P_t^{\beta_S} + \varepsilon_t \quad t = 1970\text{-}2014 \qquad (2)$$

where t = 1970-2014, the β's are unknown parameters whose values you are trying to estimate and ε_t is an error term for the t^{th} year where $\varepsilon_t \sim (0,\sigma^2)$. Note that (2) cannot be linearized with respect to the parameters given the assumed error structure.

I would like you to develop a **MATLAB** function that will determine the values of β_W and β_S that minimize the SSE from predicting $Wool\_Q_t$ via a ***Grid Search*** method. As noted above, under the ***Grid Search*** method you use a finite number of pre-defined grid points and compare the SSE's under these finite number of candidate parameter combinations. You then choose the combination that results in the minimum SSE of those respresented in the predefined grid.

When implementing a grid search algorithm, there are several questions that need to be addressed:

- *How can I determine reasonable starting values?*

  As with all parameter search algorithms, you have to supply the estimation system reasonable starting values. For this problem, eq. (1) provides a specification, after omitting the β_0, β_T and ρ parameters, you can use via the CRM, to provide starting values for the estimation of eq. (2).
- *What is a reasonable range of parameter values to define the grid?*

  There is no specific answer to this question. Obviously you would like to use a range that encompasses the true, but unlnown, parameter values for SSE minimization. If you could linearize the model of concern, a possible way to determine this range is to use the estimated cofficient standard errors obtained via the linear specification. Of course, the accuracy would depend on that model speicifcation changes needed to go from the nonlinear to linear in paramaters functions.
- *How wide should I make each cell in the grid?*

Again, there is no specific answer to this question. This is a typical judgement question that often needs to be answered when estimating nonlinear regression models.

- *What value should I assign a parameter for a particular cell?*
  The usual procedure is to assign the midpoint of the parameter range associated with a particular cell.

Similar to the AR(1) function you created in Question #1, you decide to create a single function to obtain both (i) starting values from a CRM and (ii) undertaking a grid search of parameters under the nonlinear specification. This function should print to an output file: (i) each iteration number; (ii) associated parameter estimates; and (iii) the resulting **SSE** value. At the end of the output file you should print the paramater values that provides you the smallest SSE and the SSE value.

Finally, the function you develop would be called out by a problem specific command file which should look something like the following:

[Est_Betas, Row_ID, Col_ID,Min_SSE]=
FI_LI_Grid(LLim, ULim, Num_Rows, Num_Col)

where: FI = your first initial
LI = your last initial
LLim = the (**K×1**) vector of lower limits for the K parameters for the initial grid search;
ULim = the (**K×1**) vector of parameter upper limits for the initial grid search;
Num_Rows = grid rows associated with the 1st parameter;
Num_Cols = grid columns for the 2nd parameter;
Est_Betas = the (**K×1**) vector of estimated parameters;
Row_ID = row number associated with 1st parameter;
Col_ID = column number associated with 2nd parameter; and
Min_SSE = the minimum SSE.

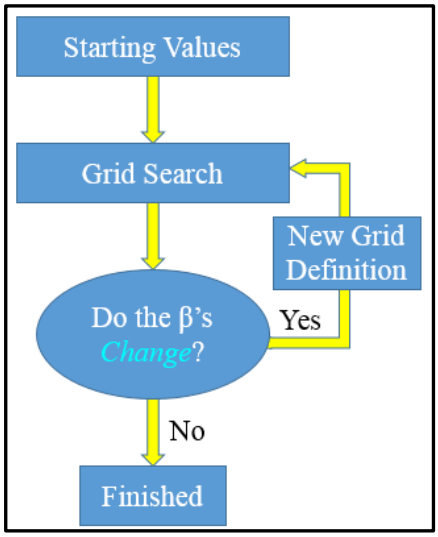
3. (20 pts) Now lets fine tune your estimation by redoing your grid search around the solution obtained in Question #2 above. That is, given the coefficients that you found to generate the minimum SSE in #2 above (i.e. , β^*_W, β^*_S) modify your function such that you undertake another grid search with the upper and lower limits of the grid for each iteration are defined by $ULim\_I_t$ and $Llim\_I_t$:

$$\text{ULim\_I}_t = \begin{cases} \text{ULim for } t=1 \\ \text{ULim\_I}_t, t=2,...,T \end{cases} \quad \text{LLim\_I}_t = \begin{cases} \text{LLim for } t=1 \\ \text{LLim\_I}_t, t=2,...,T \end{cases} \qquad (3)$$

The challenge for you is to determine the appropriate ULim_I_t and ULim_I_t when t=2,…,T. Obviously, the width of the these limits is based on the *optimal grid cell width* used in the *previous iteration*.

Once you obtain a new pair of coefficients that generate the smallest SSE within this new grid, (which should be no larger than the minimum SSE found under the previous grid search) use (3) to define a new grid cell width using this new pair of coefficients as the midpoint. Repeat this process until neither of these coefficients *change*. What is you definition of no change that can used regardless of the SSE units. Your definition should be some value greater than zero. What criteria did you use?

In summary, we can represent this system expansion via the flowchart shown to the right. You will have one additional input when calling out this new function, [Est_Betas, Row_ID, Col_ID,Min_SSE] = ***FI_LI_Grid_V2***(LLim, ULim, Num_Rows, Num_Col, *Crit_Val*)



where *Crit_Val* is the criteria value that defines whether the coefficients have *changed* however you define this criteria. For each intermediate grid search make sure you print out the *Intermediate Grid Number*, *Minimum SSE* for that grid and *Beta vector* that generated the minimum SSE. Under this new system, the output Min_SSE is the minimum SSE obtained at the end of your iterations.