EC752 - Problem Set 2

Professor: Johannes Schmieder

This problem set is the second part of the 2 part problem set that will walk you through programming and estimating a non-stationary search model with reservation wages and search effort. The second part focuses on calibrating and estimating the model using a minimum distance estimator.

1 Calibrating the Search Model

The model you programmed in the last period takes the structural parameters ξ as inputs and generates various moments, such as the exit hazard from unemployment and reemployment wages throughtout the unemployment spell.

One way to choose the parameters ξ is to pick them so that the model reproduces some of the empirical regularities that we see in the actual data. For example in the last problem set you were supposed to try to find parameters so that the model roughly reproduces the estimated $\frac{dD}{dP}$ and $\frac{dE[w]}{dP}$ from the paper. Note that in this problem set we will write the empirically (via reduced form methods) estimated moments with a tilde, e.g. $\widetilde{\frac{dD}{dP}}$, while we will write the simulated moments from the model with a hat: $\widehat{\frac{dD}{dP}}$.

In this section we will do this again, but in a more systematic fashion rather than the simple trial and error version before.

1.1 A simple calibration excercise

The file calibrateModel.m provides some basic code to do a grid search to find a version of the model that can replicate the empirical $\widetilde{\frac{dD}{dP}}$. To find this we will simulate our model for a range of possible values for γ and calculate for each the implied $\widehat{\frac{dD}{dP}}(\gamma)$ as well as how close $\widehat{\frac{dD}{dP}}(\gamma)$ is to the empirical $\widetilde{\frac{dD}{dP}}$. As a measure of closeness we will use the squared deviations

(we will denote these as SSE):

$$SSE(\gamma) = \left(\widehat{\frac{dD}{dP}}(\gamma) - \widetilde{\frac{dD}{dP}} \right)^2$$

The best fitting γ is simply given as: $\gamma^* = \arg\min_\gamma SSE$.

There are two pieces missing in the code, the calculation of $\widehat{\frac{dD}{dP}}$ and the calucation of the SSE. Fill in these missing pieces and calculate the best fitting γ and produce the two figures that show $\widehat{\frac{dD}{dP}}(\gamma)$ and $SSE(\gamma)$.

1.2 Calibration with 2 moments

Modify the code to do a grid search over different levels of k (keep the other parameters the same and let $\gamma = 0.145$. Instead of just matching $\frac{dD}{dP}$, modify the code to also match D_{12}, i.e. the expected nonemployment duration when $P = 12$. To do so let

$$SSE(k) = \left(\widehat{\frac{dD}{dP}}(k) - \widetilde{\frac{dD}{dP}} \right)^2 + \left(\widehat{D}_{12}(k) - \widetilde{D}_{12} \right)^2$$

Produce the same figures as in 1.1 but as functions of k. Can you find a k that produces both moments?

Discuss your results (why you can or cannot match the two moments).

1.3 Calibration with 2 moments and 2 parameters

Now modify the code to do a grid search over both k and γ. That is you should define a vector of values for both parameters and run the model for all possible combination (you will need a nested for loop). You should try to match the same two moments as in 1.2, so the SSE will look like this:

$$SSE(\gamma, k) = \left(\widehat{\frac{dD}{dP}}(\gamma, k) - \widetilde{\frac{dD}{dP}} \right)^2 + \left(\widehat{D}_{12}(\gamma, k) - \widetilde{D}_{12} \right)^2$$

Store the results for $SSE(\gamma, k)$ and the simulated moments in a Matrix. How close can you get the simulated moments to the empirical moments?

Produce 3D plots showing the functions $SSE(\gamma, k)$, $\widehat{\frac{dD}{dP}}(\gamma, k)$, and $\widehat{D}_{12}(\gamma, k)$. Matlab can produce plots like this using the **surf** function.

Briefly discuss your results and the following questions: In total our model has 7 parameters, could you use the same approach to calibrate all 7 parameters? What moments could

you use for this? What might be a difficulty in doing this?

2 Estimating the Search Model

In this section you will estimate the model using a minimum distance estimator. This is a straightforward extension of the previous calibration excercise, but instead of doing a grid search in order to find the parameter combination that provides the best fit, we will use an optimizer. As in the first part of the problem set we will use fmincon again, although there are other options.

2.1 Estimating 2 parameters with 2 moments

Use fmincon to find the γ^* and k^* that minimizes

$$SSE(\gamma, k) = \left(\widehat{\frac{dD}{dP}}(\gamma, k) - \widetilde{\frac{dD}{dP}} \right)^2 + \left(\widehat{D}_{12}(\gamma, k) - \widetilde{D}_{12} \right)^2$$

from part 1.3 (keeping all other parameters at the same values as before). How does this compare to the best fit from 1.3?

For your estimates γ^* and k^*, create a figure of the implied reemployment wage path and the hazard function.

Here is a list of moments from Schmieder et al (2016):

```
% ===== Moments to Match from Schmieder et al. =====
D12_true=14.225;
D18_true=15.175;
B12_true=6.685;
B18_true=8.455;
LogPostWage12_true=4.0139;
LogPostWage18_true=4.0061;
dDdP_true = 0.16;
dWdP_true = -0.0013;
```

Create a table with these empirical moments and the corresponding moments that are implied by your estimates γ^* and k^*.

Discuss the fit of these moments.

2.2 Estimating 3 parameters with 3 moments

Redo the excercise in 2.1, but now also estimate μ. To do so include the expected log reemployment wage for individuals with 12 months of UI benefits $E[\ln w|P = 12]$ as a moment to match (corresponding to the value of LogPostWage12_true above).

Compare your results with 2.1.

2.3 Estimating 4 parameters with hazard and wage moments

Let $h_{t,P}$ be the hazard rate and $w_{t,P}$ be the log reemployment wage in each period. Let:

$$SSE(\xi) = 500 * \sum_{t,P} \left(\widehat{h}_{t,P}(\xi) - \widetilde{h}\right)^2 + \sum_{t,P} \left(\widehat{w}_{t,P}(\xi) - \widetilde{w}\right)^2$$

Note that we multiply the hazard part of the SSE with 500 since the hazard consists of very small values and the hazard part would otherwise get very little weight in the estimation. If we did a more formal minimum distance estimator, we would use a weighting matrix W, such as the covariance matrix from the estimated reduced form moments. Since the hazard rates are estimated with more precision, this would also put more weight on the hazard part (and more weight on the earlier unemployment periods).

For the empirical moments you can use:

```
Hazard12_true = [0.105 0.135 0.115 0.11 0.08 0.072 ...
0.073 0.062 0.06 0.062 0.059 0.059 0.095 0.06 ...
0.058 0.06 0.05 0.051 0.051 0.047 0.046 0.047 ...
0.039 0.041 0.05 0.04 0.041 0.048 0.039 0.038 ...
0.042];
Hazard18_true = [0.1 0.1349 0.11 0.1 0.079 0.069 0.069 ...
0.059 0.057 0.057 0.048 0.049 0.06 0.057 0.055 ...
0.056 0.055 0.055 0.076 0.055 0.043 0.048 0.047 ...
0.04 0.05 0.04 0.042 0.048 0.04 0.04 0.043];
ReWage12_true = [4.165 4.167 4.13 4.1 4.065 4.06 4.02 4 ...
3.97 3.95 3.95 3.92 3.85 3.88 3.86 3.81 ...
3.81 3.9 3.83 3.78 3.81 3.71 3.79 3.73 3.81];
ReWage18_true = [4.17 4.163 4.13 4.08 4.053 4.054 4.01 ...
4 3.98 3.96 3.97 3.93 3.93 3.885 3.895 3.82 ...
3.85 3.9 3.74 3.83 3.79 3.76 3.77 3.75 3.8];
```

Use this SSE to estimate γ, k, μ, and σ and produce the same outputs as in 2.1. In the

figures for the reemployment wage and hazard rates, you should plot both the empirical moments and the simulated moments for your estimated ξ^*.

2.4 Estimating all parameters

Redo the same excercise as in section 2.3, but for all elements of ξ except for the discount rate.

As the parameter space grows in dimensionality you should notice that the optimizer will take longer to run. By default Matlab will only use one CPU core for all computations. Most computers have more than one core available and you can use those. An easy way to parallelize your code is to simply instruct fmincon to do so. The option 'UseParallel' for fmincon can be set to true (or 'always' on older matlab versions). You might want to run a quick google search to learn a little bit about this.

How much does estimating more parameters improve the fit? What quantitative and what qualitative features of the empirical moments can the model reproduce and what features has it a hard time fitting?

2.5 Estimating a Model with heterogeneity

So far our model assumed that all individuals are homogenous. It is straightforward to allow for (unobserved) heterogeneity, by assuming there are different types of individuals with different parameter values. We can then simulate the model for each type and integrate the moments to get the aggregate moments that can be compared with the data.

For example, let's assume there are two types with different cost parameters: k_1 and k_2. The share of type 1 individuals is given as p. Let $\xi_1 \equiv \{\delta, k_1, \gamma, \mu_S, S, \pi\}$ and $\xi_2 \equiv \{\delta, k_2, \gamma, \mu_S, S, \pi\}$. We can calculate the simulated moments for each type i, for example: $\widehat{D}_{12,i}$, $\widehat{\frac{dD}{dP}}_i$, $\widehat{h}_{t,P,i}(\xi_i)$ or $\widehat{w}_{t,P,i}(\xi_i)$. We can then aggregate them to the empirically observable moments. This is easy for the moments that do not depend on t. For example:

$$\widehat{D}_{12,agg} = p\widehat{D}_{12,1} + (1-p)\widehat{D}_{12,2}$$

and similarly for $\widehat{\frac{dD}{dP}}_{agg}$ and even the survival function $S_{t,i}$. For the hazard rates and reemployment wage path, which are moments **conditional** on being unemployed / exiting at t this does **not** work! If you think about it you should be able to derive the following:

Calculate the aggregate survival function:

$$\widehat{S}_{t,agg} = p\widehat{S}_{t,1} + (1-p)\widehat{S}_{t,2}$$

Now define the weights that indicate the number share of individuals who are unemployed at t who are of type i:

$$\omega_{i,t} = \frac{p\widehat{S}_{t,i}}{\widehat{S}_{t,agg}}$$

The aggregate hazard is given as:

$$\widehat{h}_{t,agg} = \sum_i \omega_{i,t}\widehat{h}_{t,i}$$

The aggregate reemployment wage conditional on exiting at t is:

$$\widehat{w}_{t,agg} = \frac{\sum_i \omega_{i,t}\widehat{h}_{t,i}w_{i,t}}{\widehat{h}_{t,agg}}$$

Note that the model with two types has two more parameters: an additional cost type and the share of types p. Let: $\xi_{hetero} = \{\delta, k_1, \gamma, \mu_S, S, \pi, k_2, p\}$. We can now define the SSE:

$$SSE(\xi_{hetero}) = 500 * \sum_{t,P}\left(\widehat{h}_{agg,t,P}(\xi_{hetero}) - \widetilde{h}\right)^2 + \sum_{t,P}\left(\widehat{w}_{agg,t,P}(\xi_{hetero}) - \widetilde{w}\right)^2$$

One issue with adding more parameters and in particular different types is that we are increasingly likely to run into convergence issues where the optimizer gets stuck in local minima and the result of the optimizer will be sensitive to the initial values we provide. To solve that a common approach is to try several starting values and pick the best fit. If a large number of starting values lead to the same best fit, and if increasing the number of starting values does not yield a different minimum, this is an indicator that this might indeed be a global minimum. Note however that this is not a guarantee and may not work if your objective function has a strange topology.[1]

Since adding heterogeneity and multiple starting values adds quite a bit of complexity, the matlab file estimateModel_multiInits provides the basic structure of this with just a few blanks to fill in. Read through the code and make sure you understand it and fill in the blanks based on the equations above. Note that I also programmed this in a way that makes it very easy to pick which parameters to estimate, which makes this much easier to test the code (since it runs much faster with fewer parameters). Also note that even with

[1] A useful test of whether the minimizer is likely to obtain the true global minimum is to simulate the model with a vector of parameters that you choose. Then run the estimation algorithm on using these simulated moments as the empirical moments to match and see whether you can recover the original parameters (obviously starting at different parameter values).

using parallel computing this quickly becomes computationally intensive. On my desktop with 8 cores this runs for about an hour with just 20 starting values.

As before estimate the model and generate the same results as in sectino 2.3. How well can you fit the data? Discuss again what quantitative and qualitative features from Schmieder et al. (2016) you can and cannot fit. What is your estimate of π, how does it compare to the IV estimate in the paper?

Schmieder et al. (2016) emphasize the negative effect of UI on reemployment wages $\frac{dE[w]}{dP}$ alongside the negative effect of UI on nonemployment durations $\frac{dD}{dP}$. Modify the SSE so that it puts a lot of weight on these two moments:

$$SSE(\xi_{hetero}) = 500 * \sum_{t,P} \left(\widehat{h}_{agg,t,P}(\xi_{hetero}) - \widetilde{h}\right)^2 + \sum_{t,P} \left(\widehat{w}_{agg,t,P}(\xi_{hetero}) - \widetilde{w}\right)^2$$
$$+ 1000 * \left(\widehat{\frac{dE[w]}{dP}}_{agg}(\xi_{hetero}) - \widetilde{\frac{dE[w]}{dP}}\right)^2 + 1000 * \left(\widehat{\frac{dE[w]}{dP}}_{agg}(\xi_{hetero}) - \widetilde{\frac{dE[w]}{dP}}\right)^2$$

How does this change the estimates? What is your estimate of π?

2.6 Alternative assumptions and extensions

Presumably the fit of the model is still not perfect and the model doesn't replicate some features from the data. Discuss 2 possible extensions / modifications to the model that one could implement and how you think they might affect the model fit. How might they affect your conclusions about π?

3 The Welfare Effects of UI extensions

Read Schmieder and von Wachter (2016). Calculate the welfare expressions in equations (7) and (8) in that paper. For simplicity assume that the employed individuals whose marginal utility show up in that formula have log wages of 4.15 and therefore $c_e = \exp(4.15)$. Furthermore assume that $\frac{\tau}{b} = 0.5$.

Would it be more beneficial to extend benefits or the increase benefit levels?

The formula can also be implemented using only reduced form estimates. What do you think the advantage of the structural model might be? What do you think the disadvantage might be?

4 Discussion of Structural vs. Reduced Form Estimation

In your own words, discuss the advantages and disadvantages of the structural vs. the reduced form approach in the light of your results from this problem set. You should write about 500 words.

References

Schmieder, J. F. and T. von Wachter (2016). The Effects of Unemployment Insurance: New Evidence and Interpretation. *Annual Review of Economics 8*.

Schmieder, J. F., T. von Wachter, and S. Bender (2016). The effect of unemployment benefits and nonemployment durations on wages. *American Economic Review 106*(3), 739–77.

Matlab Templates

```
function calibrateModel
clc;
clear;
close all;

% ===== Fixed Parameters =====
pre_wage = exp(4.15);
b_UI = pre_wage * 0.60; % UI benefits
b_UA = pre_wage * 0.30; % UA benefits

b1 = [ones(1,12).*b_UI ones(1,84).*b_UA];
b2 = [ones(1,18).*b_UI ones(1,78).*b_UA];

xi=[0.995, 150, 0.145, 4.1, 0.5, 12, 0];

% ===== Moments to Match from Schmieder et al. =====
D12_true=14.225;
D18_true=15.175;
B12_true=6.685;
B18_true=8.455;
LogPostWage12_true=4.0139;
LogPostWage18_true=4.0061;
dDdP_true = 0.16;
dWdP_true = -0.0013;



% ===== Part 1.1 =====

    % ===== Grid Search over Gamma =====
    grid = 0.1:0.01:1;

    D12 = zeros(1,length(grid));
    D18 = zeros(1,length(grid));
    SSE = zeros(1,length(grid));

    for i = 1:length(grid);

        xi(3) = grid(i);
        [s1,logphi1,haz1,logw1,surv1,D12(i)] = solveModel(xi,b1);
        [s2,logphi2,haz2,logw2,surv2,D18(i)] = solveModel(xi,b2);
```

```
        % Use the estimated moments to calculate the implied dD/dP:
        dDdP(i) = XXXXX;

% Calculate the squared deviation from the true moment
        SSE(i)=XXXXX;
    end

    [minwr ir] = min(SSE)
    SSE(ir)
    xstar = grid(ir)


end


function estimateModel_multiInits

clc;
clear;
close all;


% ===== Fixed Parameters =====
pre_wage = exp(4.15);
b_UI = pre_wage * 0.60; % UI benefits
b_UA = pre_wage * 0.30; % UA benefits
b1 = [ones(1,12).*b_UI ones(1,24).*b_UA];
b2 = [ones(1,18).*b_UI ones(1,18).*b_UA];



% ===== Moments to Match from Schmieder et al. =====
D12_true=14.225;
D18_true=15.175;
B12_true=6.685;
B18_true=8.455;
LogPostWage12_true=4.0139;
LogPostWage18_true=4.0061;
dDdP_true = 0.16;
dWdP_true = -0.0013;

Moments_true = [D12_true, D18_true, B12_true, B18_true, ...
    LogPostWage12_true, LogPostWage18_true, dDdP_true, dWdP_true]

Hazard12_true = [0.105 0.135 0.115 0.11 0.08 0.072 ...
    0.073 0.062 0.06 0.062 0.059 0.059 0.095 0.06 ...
    0.058 0.06 0.05 0.051 0.051 0.047 0.046 0.047 ...
    0.039 0.041 0.05 0.04 0.041 0.048 0.039 0.038 ...
    0.042];

Hazard18_true = [0.1 0.1349 0.11 0.1 0.079 0.069 0.069 ...
    0.059 0.057 0.057 0.048 0.049 0.06 0.057 0.055 ...
    0.056 0.055 0.055 0.076 0.055 0.043 0.048 0.047 ...
    0.04 0.05 0.04 0.042 0.048 0.04 0.04 0.043];

ReWage12_true = [4.165 4.167 4.13 4.1 4.065 4.06 4.02    4 ...
    3.97 3.95 3.95 3.92 3.85 3.88 3.86 3.81 ...
    3.81 3.9 3.83 3.78 3.81 3.71 3.79 3.73 3.81];

ReWage18_true = [4.17 4.163 4.13 4.08 4.053 4.054 4.01 ...
    4 3.98 3.96 3.97 3.93 3.93 3.885 3.895 3.82 ...
    3.85 3.9 3.74 3.83 3.79 3.76 3.77 3.75 3.8];


% ================================================
% ================== Part 2.5 ====================
% ================================================

% ===== Solve Model with Heterogeneity =====
    function [s1,logphi1,haz_agg,w_reemp_agg,survival,D] = solveModelAggr(xi_hetero,b)

        xi = xi_hetero(1:7);
        p = xi_hetero(8);
        k2 = xi_hetero(9);
```

```
        % Type 1
        [s1,logphi1,haz1,w_reemp1,surv1,D1] = solveModel(xi,b);

        % Type 2
        xi(2)=k2;
        [s2,logphi2,haz2,w_reemp2,surv2,D2] = solveModel(xi,b);

        %Aggregate Survival
        survival = p.*surv1 + (1-p).*surv2;

        %Calculate share of each type left at beginning in unemp
        weight1 = XXXXX % fill in the omega from the problem set
        weight2 = XXXXX

        %Calculate aggregate hazard and average reemployment wage of leavers
        haz_agg = XXXXX;
        w_reemp_agg = XXXXXX;

        D = XXXX;
    end

% ===== ojbectiveFunction (SSE) =====
    function [SSE, Moments_hat] = objFun(paramsToOptimizeArg)

        paramv = parsInit;
        paramv([paramsToUse])=paramsToOptimizeArg;

        % Evaluate Model under the two UI regimes given xi
        [s1,logphi1,haz1,logw1,surv1,D12] = solveModelAggr(paramv,b1);
        [s2,logphi2,haz2,logw2,surv2,D18] = solveModelAggr(paramv,b2);

        % density (surv * haz):
        dens1 = haz1.*surv1;
        dens2 = haz2.*surv2;

        % Expected Reemployment Wage
        LogPostWage12 = sum(dens1.*logw1)/sum(dens1);
        LogPostWage18 = sum(dens2.*logw2)/sum(dens2);

        % Compute extra Moments
        dDdP = (D18-D12)/(18-12);
        dWdP = (LogPostWage18 - LogPostWage12)/(18-12);

        B12=sum(surv1(1:12));
        B18=sum(surv2(1:18));

        Moments_hat = [D12, D18, B12, B18, ...
            LogPostWage12, LogPostWage18, dDdP, dWdP];

        % Compute squared deviations from moments to match
        SSE_wage = sum((ReWage12_true-logw1(1:25)).^2) ...
            + sum((ReWage18_true-logw2(1:25)).^2);
        SSE_haz = sum((Hazard12_true-haz1(1:31)).^2) ...
            + sum((Hazard18_true-haz2(1:31)).^2);

        SSE =  500 * SSE_haz + SSE_wage;

    end

    % Generate random initial starting values lying within the bounds defined by lb and ub
    function [searchInits] = getSearchInits_benchmark(noSearchInits)
        lb_short = lb_rep([paramsToUse]);
        ub_short = ub_rep([paramsToUse]);
        lb=repmat(lb_short',1,noSearchInits);
        ub=repmat(ub_short',1,noSearchInits);
        searchInits=lb+(ub-lb).*rand(noOfParams,noSearchInits);
    end


    % Lower and upper bound of parameter space (note that S cannot be < 0)
    % delta, k, gamma, mu, sigma, S, pi, p, k2
    lb_rep = [0.1, 0.1, 0.1, 0.1, 0, 0,  -1, 0, 0.1];
    ub_rep = [1,   1000, 10, 10, 1, 300,  1, 1, 1000];
```

```
    % Initial value for fmincon
    %     x0=[1 0.7148    4.120   0.5 12 0 .5 10];


    % Number of Initial values:
    noSearchInits = 20;

    % Here you can set which parameters the algorithm uses,
    % e.g. paramsToUse = [2]
    % only estimates the cost of the first type
    paramsToUse = [2 3 4 5 6 7 8 9];
%     paramsToUse = [2 3 8 9];

% This is the vector of benchmark parameters, if a parameter is not
% estimated, the corresponding value from this vector is taken.
    parsInit = [0.995, 2, 0.145, 4.1, 0.5, 12, 0, 0.5, 10];
    noOfParams = length(paramsToUse);
    searchInits = getSearchInits_benchmark(noSearchInits);

    paramHats=zeros(noOfParams,noSearchInits);
    sse= repmat(-9999,1,noSearchInits);
    converged=repmat(-9999,1,noSearchInits);
    exitFlag=repmat(-9999,1,noSearchInits);

    % To use parallel computing, makes sure parallel pool is not running
    delete(gcp('nocreate'))
    parpool;
    for j = 1:noSearchInits
        tic
        j

        x0 = searchInits(:,j);
        lb_res = lb(:,j);
        ub_res = ub(:,j);

        % If you do not want to use parallel computing
        options = optimset('Display', 'iter','Algorithm','interior-point');

        % To use parallel computing with older versions of Matlab (e.g. 2013)
        options = optimset('Display', 'iter','Algorithm','interior-point','UseParallel','always');

        % To use parallel computing with older versions of Matlab (e.g. 2015)
        %  options = optimset('Display', 'iter','Algorithm','interior-point','UseParallel',true);

        try
         [paramHats(:,j), sse(j), exitFlag(j),outputf(j)] = ...
             fmincon(@objFun,x0,[],[],[],[],lb_res,ub_res,[],options)
        catch
         warning('Problem in fmincon');
         disp('Initial Values: ');
         disp(x0);
         exitFlag(j) = -1;

        end

        converged(j)=-9999;
        if (exitFlag(j) <= 0)
            converged(j)=0;
        end
        if (exitFlag(j) > 0)
            converged(j)=1;
        end

        toc
        elapsed_time_full(1,j) = toc/60;
    end

    searchInits=searchInits';
    paramHats=paramHats';

    %% display some information about the estimates
    noOfAcceptableSolutions=sum(converged); % converged estimates
    disp('Converged solutions:  '); disp(noOfAcceptableSolutions);
    convergedParamHats=repmat(-9999,noOfAcceptableSolutions,noOfParams);
```

```
    convergedInits=repmat(-9999,noOfAcceptableSolutions,noOfParams);
    convergedSse=repmat(-9999,noOfAcceptableSolutions,1);
    convergedflags=repmat(-9999,noOfAcceptableSolutions,1);
    elapsed_time=repmat(-9999,noOfAcceptableSolutions,1);
    sorter=repmat(-9999,noOfAcceptableSolutions,1);

    %We only care about the estimates that converged
    j=0;
    sorter_temp = [1:noSearchInits];
    for i=1:noSearchInits
        if converged(i)==1
            j=j+1;
            convergedParamHats(j,:)=paramHats(i,:);
            convergedInits(j,:)=searchInits(i,:);
            convergedSse(j)=sse(i);
            convergedflags(j) = exitFlag(i);
            %outputfmincon(j) = outputf(i);
            elapsed_time(j) = elapsed_time_full(i);
            sorter(j) = sorter_temp(i);
        end
    end
    sseParams=[convergedSse convergedParamHats convergedInits convergedflags elapsed_time sorter];
    sseParamsSorted=sortrows(sseParams);
    convergedSse=sseParamsSorted(:,1);
    convergedParamHats=sseParamsSorted(:,2:noOfParams+1);
    convergedInits=sseParamsSorted(:,noOfParams+2:noOfParams+1+noOfParams);
    convergedflags=sseParamsSorted(:,noOfParams+noOfParams+2:noOfParams+noOfParams+2);
    elapsed_time=sseParamsSorted(:,noOfParams+noOfParams+3:noOfParams+noOfParams+3);
    sorter=sseParamsSorted(:,noOfParams+noOfParams+4:noOfParams+noOfParams+4);
    outputfmincon = outputf(sorter);

    %Number of estimates within the desired SSE range
    %     noSmallSSE=sum(convergedSse<maxSSE);
    %Estimate with the smallest SSE

    bestEstimate=convergedParamHats(1,:)
    % SSE = convergedSse(1)

    xi_hetero_star = parsInit;
    xi_hetero_star([paramsToUse])=bestEstimate;
    xi_hetero_star




end
```