ECON 504 - Repeated Matching Assignment

Hanifi Suleyman Killioglu

December 8, 2019

1 Introduction

This report intends to provide guidance to the Master-HK.jl and Base-HK.jl files attached to Assignment-3-HK.zip file and address the questions posed in the problem sheet. To that end, the report will follow the order followed in the question sheet in its organization and provide clear references to the Julia files whenever necessary.

Before proceeding to the next sections, it is better to summarize the model in hand to create a reference point. We have a unit mass of workers and a unit mass of jobs: There are two types of workers denoted by $x \in \{1, 2\}$ and two types of jobs denoted by $y \in \{1, 2\}$. At a given state, workers and jobs match and states begin evolving. Speaking of them, the state transition probabilities for the type of worker and states are, respectively, given by $P(x' \mid x, y)$ and $Q(y' \mid x, y)$.

The primitives in this model are the matching occurring at each state, $\boldsymbol{\mu} = [\mu_{11},\ \mu_{12},\ \mu_{21},\ \mu_{22}]$, and they are constrained by the aggregate state of matching, (a_1, b_1). Following the simple matrix given in Figure 1 is useful in explaining the relationship between these two. As it is clear from the figure, the optimal matching to be found later via the Bellman equation for the social planner is bounded by these summations.

	Mass of Type 1 Job	Mass of Type 2 Job	**Total Mass**
Mass of Type 1 Worker	μ_{11}	μ_{12}	$\mu_{11} + \mu_{12} = \mathbf{a_1}$
Mass of Type 2 Worker	μ_{21}	μ_{22}	$\mu_{21} + \mu_{22} = \mathbf{1 - a_1}$
Total Mass	$\mu_{11} + \mu_{21} = \mathbf{b_1}$	$\mu_{12} + \mu_{22} = \mathbf{1 - b_1}$	**1**

Figure 1: The matching and corresponding aggregate state of matching

On the other hand, what determines the aggregate state of matching is the matching occurred in the previous state. That is, we have the following relationship between the aggregate state of matching as a result of the transition probabilities and constraints summarized in Figure 1:

$$\begin{aligned} a_1'(\boldsymbol{\mu}) &= \mu_{11}P(1 \mid 1,1) + \mu_{12}P(1 \mid 1,2) + \mu_{21}P(1 \mid 2,1) + \mu_{22}P(1 \mid 2,2) \\ &= 0.6\mu_{11} + 0.5\mu_{12} + 0.4\mu_{21} + 0.35\mu_{22} \end{aligned} \tag{1}$$

$$
\begin{aligned}
b_1'(\boldsymbol{\mu}) &= \mu_{11}Q(1 \mid 11) + \mu_{21}Q(1 \mid 2,1) + \mu_{12}Q(1 \mid 1,2) + \mu_{22}Q(1 \mid 2,2) \\
&= 0.65\mu_{11} + 0.45\mu_{12} + 0.55\mu_{21} + 0.3\mu_{22}.
\end{aligned} \tag{2}
$$

The matching in a competitive equilibrium solves the social planner's problem of maximizing the present discounted value of economy wide output. In this regard, the Bellman's equation for the social planner is described as follows:

$$
V(a_1, b_1) = \max_{\boldsymbol{\mu} \in F(a_1, b_1)} \left\{ \sum_{x,y} \mu_{x,y}\phi_{x,y} + \delta V\left(a_1'(\mu), b_1'(\mu)\right) \right\} \tag{3}
$$

where $F(a_1, b_1)$ summarizes the constraints in Figure 1 given the aggregate state (a_1, b_1) and the output function $\phi_{x,y} = x * y$ determines the output when a worker of type x and job of type y form a match. We denote the equilibrium matching by $\boldsymbol{\mu}(a_1, b_1)$, which is a continuous function that maps the current state with optimal matching. We are also told that there is a unique equilibrium matching.

Before computing the full solution to the model by solving the social planner's Bellman's equation for all states, I will firstly solve for what is called constant aggregate state in the next section. Afterward, section three will be devoted to the computation of full solution to the model via the value function approximation method.

2 Constant Aggregate State

In our model, a constant aggregate state corresponds to a state denoted by (a_1^*, b_1^*) in which the equilibrium matching is such that $\left(a_1'\left(\boldsymbol{\mu}\left(a_1^\star, b_1^\star\right)\right), b_1'\left(\boldsymbol{\mu}\left(a_1^\star, b_1^\star\right)\right)\right) = (a_1^\star, b_1^\star)$. Correspondingly, we have the following Bellman's equation for the constant aggregate state

$$
V^\star = \max_{\boldsymbol{\mu} \in F\left(a_1^*, b_1^*\right)} \left\{ \sum_{x,y} \mu_{x,y}\phi_{x,y} + \delta V^* \right\}. \tag{4}
$$

More explicitly we have the following Bellman's equation

$$
V^\star = \max_{\boldsymbol{\mu} \in F\left(a_1^*, b_1^*\right)} \left\{ \mu_{11} + 2\mu_{12} + 2\mu_{21} + 4\mu_{22} + \delta V^* \right\} \tag{5}
$$

where $F(a_1^*, b_1^*) = \{\boldsymbol{\mu} \in [0,1]^4 : \mu_{11} + \mu_{12} = a_1^*,\ \mu_{11} + \mu_{21} = b_1^*, \text{ and } \mu_{22} - \mu_{11} = 1 - a_1^* - b_1^*\}$. Moreover, the fact that (a_1^*, b_1^*) is a constant aggregate state provides us with two more constraints for the optimal policies, $\boldsymbol{\mu}^*(a_1^*, b_1^*) = [\mu_{11}^*(a_1^*, b_1^*),\ \mu_{12}^*(a_1^*, b_1^*),\ \mu_{21}^*(a_1^*, b_1^*),\ \mu_{22}^*(a_1^*, b_1^*)]$:

$$
a_1^* = 0.6\mu_{11}^*(a_1^*, b_1^*) + 0.5\mu_{12}^*(a_1^*, b_1^*) + 0.4\mu_{21}^*(a_1^*, b_1^*) + 0.35\mu_{22}^*(a_1^*, b_1^*) \tag{6}
$$

$$
b_1^* = 0.65\mu_{11}^*(a_1^*, b_1^*) + 0.45\mu_{12}^*(a_1^*, b_1^*) + 0.55\mu_{21}^*(a_1^*, b_1^*) + 0.3\mu_{22}^*(a_1^*, b_1^*). \tag{7}
$$

At this point, one can construct a nested algorithm that maximizes V^* by choosing $(a_1, b_1) \in [0,1]^2$. Such an algorithm would call the inner optimization problem, which is given by equation 5, for each choice of $(a_1, b_1) \in [0,1]^2$ and calculate the corresponding V^* by finding out the sub-optimal policies for the given state. The underlying main idea of such an algorithm would be the fact that V^* is increasing in $\boldsymbol{\mu}$ and V^* is maximized at the constant aggregate state (a_1^*, b_1^*).

Equivalently, one can also construct a simpler algorithm based on the same underlying reasoning through a simplified version of this problem. An equivalent yet much simpler version can be obtained based on the fact that determining either one of $\{\mu_{11}, \mu_{12}, \mu_{21}, \mu_{22}\}$ is equivalent to determining all of them at a given state by recalling the constraints in Figure 1. Thus, one can apply a change of variables via the original constraints by writing $\mu_{12}, \mu_{21},$ and μ_{22} in terms of μ_{11}. Then, the problem will be the following:

$$\begin{aligned} V^\star = \max_{\mu_{11} \in [0,1]} \{4 - 2(a^* + b^*) + \mu_{11} + \delta V^*\} \text{ s.t.} \\ \mu_{11} \leq b_1^*, \quad \mu_{11} \leq a_1^*, \quad \text{and} \quad 0 \leq 1 - a_1^* - b_1^* + \mu_{11} \leq 1 \end{aligned} \tag{8}$$

with the equilibrium conditions

$$85a_1^* - 5b_1^* - 35 = 5\mu_{11}^*(a_1^*, b_1^*) \tag{9}$$

$$15a_1^* - 75b_1^* + 30 = 5\mu_{11}^*(a_1^*, b_1^*) \tag{10}$$

which imply that $a_1^* + b_1^* = \frac{13}{14}$. If one further incorporates this into our objective function, the problem's final form will be

$$V^\star = \max_{\mu_{11} \in [0,1]} \left\{ \frac{15}{17} + \mu_{11} + \delta V^* \right\} \tag{11}$$

with the same constraints in equation 8. Then, since V^* is just a scalar, one can easily solve this problem by solving the following optimization problem:

$$\begin{aligned} \max_{\mu_{11} \in [0,1] \text{ and } (a_1^*, b_1^*) \in [0,1]^2} \left\{ \frac{15}{17} + \mu_{11} + \delta V^* \right\} \text{ s.t.} \\ \mu_{11} = 17a_1^* - b_1^* - 7 \\ \mu_{11} = 3a_1^* - 15b_1^* + 6 \\ 0 \leq 1 - a_1^* - b_1^* + \mu_{11} \leq 1 \\ \mu_{11} \leq b_1^* \\ \mu_{11} \leq a_1^*. \end{aligned} \tag{12}$$

The algorithm starting at line 6 of Master-HK.jl file solves this problem by calling Ipopt in Julia. Moreover, after solving for a_1^*, b_1^*, and $\mu_{11}^*(a_1^*, b_1^*)$ in the first part of the algorithm, the second half of the while loop solves the very same problem considering μ_{11} as the only choice variable by taking (a_1^*, b_1^*) given from the first part. This amounts to checking if we really

stay in (a_1^*, b_1^*) forever when we enter this state. Finally, the results are provided in Figure 2 for further discussion. Accordingly, I can now move on to the discussion of their economic interpretation.

(a_1^*, b_1^*)	V^*	μ_{11}	μ_{12}	μ_{21}	μ_{22}
$\approx (0.47, 0.46)$	≈ 52.1	≈ 0.462	≈ 0.004	≈ 0.0	≈ 0.534

Figure 2: Constant aggregate state and the corresponding values

The striking result is that the matching in the constant aggregate state leaves practically no room for an 'unbalanced matching', by which I denote the kinds of matching where $x \neq y$ in μ_{xy}. To my understanding, the reasoning that leads to such a result is twofold.

First of all, the form of production function suggests that matching between workers and jobs of type 2 increases the economy-wide output the most compared to other possible matching and matching between workers and jobs of type 1 increases the economy-wide output the least. Actually, four different ways of writing the objective function, based on the fact that determining either one of $\{\mu_{11}, \mu_{12}, \mu_{21}, \mu_{22}\}$ is equivalent to determining all of them, yield to an illuminating observation: While the objective function is increasing in μ_{11} and μ_{22}, it is decreasing in μ_{12} and μ_{21} based on the constraints denoted by $\boldsymbol{\mu} \in F(a, b)$. From this perspective, a social planner who seeks to maximize the economy-wide output would be wise in assigning a positive match to both μ_{11} and μ_{22} in a supporting way with our results presented in Figure 2. However, this is only one side of the underlying reasoning.

Recalling Figure 1, what restricts the maximum amount of matching between workers and jobs of type 2 are $1 - a_1$ and $1 - b_1$. If we somehow manage start putting all the weight on μ_{22}, we will be limiting the next period's μ_{22} the least because $\mu_{22}P(1 \mid 2, 2) = 0.35\mu_{22}$ and $\mu_{22}Q(1 \mid 2, 2) = 0.3\mu_{22}$. However, we will still be limiting it because a_1' and b_1' will have increased. Therefore, there will be a room for other three matching. Considering the fact that μ_{11} also increases the objective function, it is then the best to put all remaining weight to μ_{11} as it is the case according to Figure 2. Once we reach to the state $\approx (0.47, 0.46)$, this reasoning combined with the probabilities in front of μ_{11} and μ_{22} will lead to the assignment presented in Figure 2.

3 Complete Solution

In this part of the assignment, the aim is to characterize a solution to Bellman's equation. That is, I will solve equation 3 via the value function approximation method and present a policy function evaluated at the adjusted Chebyshev nodes I specify in the next paragraphs. For the sake of clarity, I begin by providing a detailed outline of the algorithm employed in the solution.

3.1 The Setup

In this subsection, I will explain the basics of algorithm starting at line 66 of the Master-HK.jl file.

1. We have two state variables: $(a_1, b_1) \in [0,1]^2$. Therefore I employ two-dimensional value function approximation.

2. M denotes the number of basis functions per dimension. In the algorithm, I implement the Chebyshev basis functions in two distinct ways. During the setup of algorithm where I generate the nodes, `ChebyshevT` function at line 8 of Base-HK.jl file is called. This function is recursively constructed and accepts two arguments: `m` corresponds to the degree of Chebyshev basis function and `x` corresponds to the value at which the function is evaluated.

3. w denotes the number of nodes for each which the algorithm solves the equation 3 by considering (a_1, b_1) as the current node.

4. At line 74 and 75, I firstly generate w Chebyshev interpolation nodes, $e_j = -cos(\frac{2j-1}{w}\pi)$ $\forall j = 1, 2, \ldots, w$. Then, I construct two-dimensional nodes by taking their Cartesian product and store them in $w \times w$ matrix of 1×2 matrices, named as `Nodes`.

5. At line 77, I construct a $w \times w$ matrix of $M \times M$ matrices, named as `Regressors`. This special matrix is constructed to store the regressor values instead of re-calculating at each iteration since these values are fixed across iterations. As it is an important matrix and can be daunting due to its size, how it looks like is depicted below.

$$\begin{bmatrix} \begin{bmatrix} \phi_1(e_1)\phi_1(e_1) & \ldots & \phi_1(e_1)\phi_M(e_1) \\ \phi_2(e_1)\phi_1(e_1) & \ldots & \phi_2(e_1)\phi_M(e_1) \\ \vdots & \ddots & \vdots \\ \phi_M(e_1)\phi_1(e_1) & \ldots & \phi_M(e_1)\phi_M(e_1) \end{bmatrix} & \cdots & \begin{bmatrix} \phi_1(e_1)\phi_1(e_w) & \ldots & \phi_1(e_1)\phi_M(e_w) \\ \phi_2(e_1)\phi_1(e_w) & \ldots & \phi_2(e_1)\phi_M(e_w) \\ \vdots & \ddots & \vdots \\ \phi_M(e_1)\phi_1(e_w) & \ldots & \phi_M(e_1)\phi_M(e_w) \end{bmatrix} \\ \vdots & \ddots & \vdots \\ \begin{bmatrix} \phi_1(e_w)\phi_1(e_1) & \ldots & \phi_1(e_w)\phi_M(e_1) \\ \phi_2(e_w)\phi_1(e_1) & \ldots & \phi_2(e_w)\phi_M(e_1) \\ \vdots & \ddots & \vdots \\ \phi_M(e_w)\phi_1(e_1) & \ldots & \phi_M(e_w)\phi_M(e_1) \end{bmatrix} & \cdots & \begin{bmatrix} \phi_1(e_w)\phi_1(e_w) & \ldots & \phi_1(e_w)\phi_M(e_w) \\ \phi_2(e_w)\phi_1(e_w) & \ldots & \phi_2(e_w)\phi_M(e_w) \\ \vdots & \ddots & \vdots \\ \phi_M(e_w)\phi_1(e_w) & \ldots & \phi_M(e_w)\phi_M(e_w) \end{bmatrix} \end{bmatrix}$$

6. At line 78, I construct an $M \times M$ matrix, named `SquaredRegressors` to store the denominator of the 'regression' required for the update of Chebyshev coefficients. How this matrix looks like is also depicted below.

$$\begin{bmatrix} \sum_{j_1}^{w}\sum_{j_2}^{w} \phi_1^2(e_{j1})\phi_1^2(e_{j2}) & \ldots & \sum_{j_1}^{w}\sum_{j_2}^{w} \phi_1^2(e_{j1})\phi_M^2(e_{j2}) \\ \sum_{j_1}^{w}\sum_{j_2}^{w} \phi_2^2(e_{j1})\phi_1^2(e_{j2}) & \ldots & \sum_{j_1}^{w}\sum_{j_2}^{w} \phi_2^2(e_{j1})\phi_M^2(e_{j2}) \\ \vdots & \ddots & \vdots \\ \sum_{j_1}^{w}\sum_{j_2}^{w} \phi_M^2(e_{j1})\phi_1^2(e_{j2}) & \ldots & \sum_{j_1}^{w}\sum_{j_2}^{w} \phi_M^2(e_{j1})\phi_M^2(e_{j2}) \end{bmatrix}$$

7. Lastly, I construct w *adjusted* Chebyshev interpolation nodes at line 81, $x_j = (e_j+1)/2$ $\forall j = 1, 2, \ldots, w$.

8. In the same fashion with step 4, I construct two-dimensional *adjusted* nodes by taking their Cartesian product and store them in $w \times w$ matrix of 1×2 matrices, named as `NodesAdjusted`. A typical element of this matrix is a tuple of nodes as $[x_{1_{j1}}, x_{2_{j2}}]$ where $j1, j2 \in \{1, 2, \ldots, w\}^2$.

9. Lastly, an initial guess for $\boldsymbol{\alpha}_{M\times M}$ is made.

3.2 The Algorithm

Having presented the setup for algorithm starting at line 91 of Master-HK.jl file, I will now explain the value function approximation method that constitutes the fundamental of the complete solution to the model.

Our goal is to approximate $V(a_1, b_1)$ in equation 3. In this regard, we consider the following form approximation for a given node $(x_{1_{j1}}, x_{2_{j2}})$:

$$V(x'_{1_{j1}}, x'_{2_{j2}}) = \sum_{m_1}^{M}\sum_{m_2}^{M} \alpha_{m1,m2}\phi_{m1}(2x'_{1_{j1}} - 1)\phi_{m2}(2x'_{2_{j2}} - 1).$$

Since (a_{j_1}, b_{j_1}) is not an adjusted node, I adjust them before placing inside the basis functions. Given an initial guess for $\boldsymbol{\alpha}_{M\times M}$, our approach is to iterate the value function until further iteration does not improve the approximation. Again for the sake of clarity, I briefly explain the methodology before presenting the results and providing a discussion in that regard.

1. An initial guess $\boldsymbol{\alpha}^0$ is defined to be an $M \times M$ matrix of ones.

2. Given $w \times w$ matrix of 1×2 matrices, `NodesAdjusted`, the the algorithm begins by solving

$$\max_{\boldsymbol{\mu}\in F\left(x_{1_{j1}}, x_{2_{j2}}\right)} \left\{ \sum_{x,y} \mu_{x,y}\phi_{x,y} + \delta \sum_{m_1}^{M}\sum_{m_2}^{M} \alpha^0_{m1,m2}\phi_{m1}(2x'_{1_{j1}} - 1)\phi_{m2}(2x'_{2_{j2}} - 1) \right\} \tag{13}$$

   for each $(x_{1_{j1}}, x_{2_{j2}})$ in `NodesAdjusted` where $(x'_{1_{j1}}, x'_{2_{j2}})$ is given by equations 1 and 2.

3. Then, for each $(x_{1_{j1}}, x_{2_{j2}})$ in `NodesAdjusted`, it stores

$$\hat{V}(x_{1_{j1}}, x_{2_{j2}}) = \sum_{x,y} \mu^*_{x,y}\phi_{x,y} + \delta \sum_{m_1}^{M}\sum_{m_2}^{M} \alpha^0_{m1,m2}\phi_{m1}(2x'_{1_{j1}}(\boldsymbol{\mu}^*) - 1)\phi_{m2}(2x'_{2_{j2}}(\boldsymbol{\mu}^*) - 1)$$

   as $y^*_{j1,j2}$ where $\boldsymbol{y}^*$ is $w \times w$ matrix that stores the values of objective function at optimum for each node.

4. After performing the above optimization for each $(x_{1_{j1}}, x_{2_{j2}})$ in `NodesAdjusted`, the algorithm updates $\boldsymbol{\alpha}^0$ according to the following equation

$$\alpha^1_{m1,m2} = \frac{\sum_{j_1}^{w}\sum_{j_2}^{w} y^*_{j1,j2}\phi_{m1}(e_{j1})\phi_{m2}(e_{j2})}{\sum_{j_1}^{w}\sum_{j_2}^{w} \phi^2_{m1}(e_{j1})\phi^2_{m2}(e_{j2})} \quad \forall m1, m2 \in \{1, 2, \ldots, M\} \tag{14}$$

5. While the algorithm keeps updating $\boldsymbol{\alpha}$, the iteration stops if and only if Bellman's equation errors for $w \times w$ adjusted interpolation nodes stop decreasing. This correspondence to the case where $\sum_{j1=1}^{w}\sum_{j2=1}^{w}(y^{*,k}_{j1,j2} - y^{*,k-1}_{j1,j2})^2 <$ Tolerance.

3.3 Results

3.3.1 Approximation Accuracy

As it is even clear in the most basic case with $M = 4$ and $w = 6$, we can easily argue that our solution for the value function is settling down with an increasing number of basis functions and nodes. While there are very small noises for $M = 4$ and $M = 6$, the results for $M = 8$ demonstrate that the value function is getting more affine. In particular, the coefficients that are significantly different than zero are $\alpha_{1,1}$, $\alpha_{1,2}$, $\alpha_{2,1}$ and $\alpha_{2,2}$. Recalling to which basis functions these four coefficients correspond, we can get a better sense of the continuation value. Moreover, we can also benefit from our results from section three. In that section, we found out that the continuation value is around 52.1 once we enter the constant aggregate state. Not so surprisingly, the coefficients in figure 7 signal a very similar pattern. $\alpha_{1,1}$ in particular is around 51.42 and it corresponds to $\phi_1(a)\phi_2(b) = 1$. That is, the value function is almost constant and its value is almost the same with that of in the constant aggregate state. Moreover, the minus sign in front of $\alpha_{1,2}$ and $\alpha_{2,1}$ supports our previous result as they imply that the continuation value is decreasing in a and b.

$\alpha_{m1,m2}$	1	2	3	4
1	50.89	-1.06	-0.05	-0.00
2	-1.05	0.29	0.00	-0.02
3	-0.05	0.00	0.06	0.00
4	0.00	-0.02	0.00	0.03

(a) $w = 6$

$\alpha_{m1,m2}$	1	2	3	4
1	50.78	-1.06	-0.05	0.00
2	-1.05	0.28	0.00	-0.02
3	-0.04	0.00	0.06	0.00
4	0.00	-0.02	0.00	0.03

(b) $w = 9$

Figure 3: $\boldsymbol{\alpha}$ for M = 4 and Tolerance 10^{-4}

3.3.2 Policy Function Analysis

For the policy function analysis, I will rely on the best results we have so far: Optimal policies resulting from the value function approximation with $M = 8$, $w = 16$, and Tolerance 10^{-3}. Moreover, it is vital to consider the adjusted nodes at which the algorithm evaluates the inner optimization problem given by equation 13 since the results stored in mustar$_{M=8_w=16}$.csv correspond to those nodes.

$\alpha_{m1,m2}$	1	2	3	4	5	6
1	51.32	-1.06	-0.05	0.00	0.00	0.00
2	-1.05	0.28	0.00	-0.02	0.00	0.00
3	-0.05	0.00	0.06	0.00	-0.01	0.00
4	0.00	-0.02	0.00	0.03	0.00	-0.01
5	0.00	0.00	-0.01	0.00	0.02	0.00
6	0.00	0.00	0.00	-0.01	0.00	0.01

Figure 4: $\boldsymbol{\alpha}$ for $M = 6$, $w = 9$, and Tolerance 10^{-4}

$\alpha_{m1,m2}$	1	2	3	4	5	6
1	51.25	-1.06	-0.05	0.00	0.00	0.00
2	-1.05	0.28	0.00	-0.02	0.00	0.00
3	-0.05	0.00	0.05	0.00	-0.01	0.00
4	0.00	-0.02	0.00	0.02	0.00	0.00
5	0.00	0.00	-0.01	0.00	0.01	0.00
6	0.00	0.00	0.00	0.00	0.00	0.01

Figure 5: $\boldsymbol{\alpha}$ for $M = 6$, $w = 12$, and Tolerance 10^{-4}

$\alpha_{m1,m2}$	1	2	3	4	5	6	7	8
1	51.47	-1.06	-0.05	0.00	0.00	0.00	0.00	0.00
2	-1.04	0.28	0.01	-0.02	0.00	0.00	0.00	0.00
3	-0.05	0.00	0.05	0.00	-0.01	0.00	0.00	0.00
4	0.00	-0.02	0.00	0.02	0.00	0.00	0.00	0.00
5	0.00	0.00	-0.01	0.00	0.01	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01

Figure 6: $\boldsymbol{\alpha}$ for $M = 8$, $w = 12$, and Tolerance 10^{-3}

$\alpha_{m1,m2}$	1	2	3	4	5	6	7	8
1	51.42	-1.06	-0.05	0.00	0.00	0.00	0.00	0.00
2	-1.04	0.28	0.01	-0.02	0.00	0.00	0.00	0.00
3	-0.05	0.00	0.05	0.00	-0.01	0.00	0.00	0.00
4	0.00	-0.02	0.00	0.02	0.00	0.00	0.00	0.00
5	0.00	0.00	-0.01	0.00	0.01	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
8	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00

Figure 7: $\boldsymbol{\alpha}$ for $M = 8$, $w = 16$, and Tolerance 10^{-3}

In this regard, a straightforward result appears right away: As (a, b) increases, μ_{22}^* decreases when we move along the diagonals of both matrices, NodesAdjusted.csv and mustar$_M$ = $8_w = 16$.csv. Moreover, as we would expect based on our previous results, μ_{11}^* increases at the same time.