Computation of Equilibrium Thresholds

Chishio Furukawa

March 2019

This document explains the `MATLAB` code to generate numerical examples of $\overline{\beta}, \underline{\beta}$ thresholds for various parameters of the environment. The overview of this algorithm is described in Appendix B1.1 Simulation Step Overview. This document describes the codes in a more detail.

1. Overview

The equilibrium thresholds at each realization of σ_i are determined by the system of $2 \times S$ nonlinear equations,as follows:

$$\begin{aligned}
\overline{\beta}\left(\sigma_i\right) &= \frac{\sigma_i^2}{\mathbb{E}^{\sigma_{-i}}\left[\frac{1}{\sum_{j=0}^{N}\sigma_j^{-2}}\middle|Piv_1\right]}\left\{c-\mathbb{E}^{\sigma_{-i}}\left[\frac{\mathbb{E}^{m_{-i}}\left[\mathbb{E}^{\beta_{-i}}\sum\frac{\beta_{-i}}{\sigma_{-i}^2}\middle|Piv_1\right]}{\sum_{j=0}^{N}\sigma_j^{-2}}\middle|Piv_1\right]\right\} &(1)\\
\underline{\beta}\left(\sigma_i\right) &= \frac{\sigma_i^2}{\mathbb{E}^{\sigma_{-i}}\left[\frac{1}{\sum_{j=0}^{N}\sigma_j^{-2}}\middle|Piv_0\right]}\left\{c-\mathbb{E}^{\sigma_{-i}}\left[\frac{\mathbb{E}^{m_{-i}}\left[\mathbb{E}^{\beta_{-i}}\sum\frac{\beta_{-i}}{\sigma_{-i}^2}\middle|Piv_0\right]}{\sum_{j=0}^{N}\sigma_j^{-2}}\middle|Piv_0\right]\right\} &(2)
\end{aligned}$$

for all discretized standard error values $i \in \{1, ..., S\}$. Let us write n_1' as the number of positive results reported by other researchers, and n_0' as the number of negative results reported by other researchers. Here, Piv_1 denotes the set of realizations of other researchers' messages m_{-i} such that reporting $m_i = 1$ turns the policy decision from $a = 0$ to $a = 1$. In the supermajoritarian rule, this condition Piv_1 is equivalent to $n_1' = n_0'$. Conversely, Piv_0 is the set of message realizations such that casting $m_i = 0$ turn the policy decision from $a = 1$ to $a = 0$. I write expectation operators explicitly in orders following the computation. This condition is equivalent to $n_1' = n_0' + 1$. Other variables are defined in Sections 2.1 Set-Up and 2.2 Analyses of the main text.

2. Algorithm

This Section describes how the combination of arrays and functions decompose the indifference conditions, and discusses an iterative method to compute them.

2.1. Decomposition of indifference conditions

To evaluate the expressions in (1) and (2), we will decompose the formula into multiple arrays and functions. The right-hand-side of the expressions will be decomposed as follows:

$$\underbrace{\frac{\sigma_i^2}{\mathbb{E}^{\sigma_{-i}}\left[\frac{1}{\sum_{j=0}^{N}\sigma_j^{-2}}\Bigg| Piv\right]}}_{\texttt{Eb(Q,M,B)}} \left\{ c - \underbrace{\mathbb{E}^{\sigma_{-i}}\left[\underbrace{\left(\sum_{j=0}^{N} \sigma_j^{-2}\right)^{-1}}_{\texttt{Q}} \times \underbrace{\mathbb{E}^{m_{-i}}\left[\overbrace{\mathbb{E}^{\beta_{-i}}\sum\frac{\beta_{-i}}{\sigma_{-i}^2}\Bigg| m\in Piv}^{\texttt{EbPm(B)}}\right]}_{\texttt{Ebs(M)}} \Bigg| Piv\right]}_{\texttt{Eb(Q,M,B)}}\right\} \tag{3}$$

Arrays: the arrays `B`, `Q`, `M` above, in addition to `epsilon` and `r`, have the following analytical mapping, dimensions, and range:

Code	Meaning	Dimensions	Range
	Arguments		
`B`	β thresholds	$S \times 2 \times N$	$\mathbb{R}^1$
	Preparatory arrays		
`Q`	Combined weights of probability of $\{\sigma_i\}$ $n_1' = n_0'$ or $n_1' = n_0' + 1$	$\underbrace{S \times ... \times S}_{N}$	$\mathbb{R}^1_{++}$
`M`	Message realizations conditional on realizations and their inverse variances	$\sum_{k=0}^{\lfloor N/2 \rfloor} \begin{pmatrix} N \\ k \end{pmatrix} \times (N-1)$	$\{1,0,-1\}$, where $\mathtt{m} = \begin{cases} 1 & 1 \\ 0 \Leftrightarrow m = & \emptyset \\ -1 & 0 \end{cases}$
`epsilon`	Realizations of ϵ_{-i}	$\mathtt{R} \times (N-1)$	
`r`	Correlation coefficients $\rho_{ij} = \frac{\sigma_0^2}{\sqrt{(\sigma_0^2+\sigma_i^2)(\sigma_0^2+\sigma_j^2)}}$	$S \times S$	$(0, 1)$

Functions: the functions $\texttt{Eb}(\texttt{Q},\texttt{M},\texttt{B})$, $\texttt{Ebs}(\texttt{M})$, and $\texttt{EbPm}(\texttt{B})$ take the arrays to evaluate each element of (3) in 3 layors:

1. `Eb` uses `Ebs` and `Q` to compute the average over σ_{-i} realizations;

2. `Ebs` uses `EbPm` and `M` to compute the average over m realizations conditional on Piv;

3. `EbPM` uses empirical distribution adjusting `B` to compute the $\mathbb{E}\sum\frac{\beta}{\sigma^2}$ and $\mathbb{P}$ for each σ, m realization.

Computation: to compute $\mathbb{P}\left(\sigma_{-i}|Piv\right)$ and $\mathbb{E}^{\beta_{-i}}\left[\sum\frac{\beta_{-i}}{\sigma_{-i}^2}|m_{-i},\beta_i\right]$, the algorithm uses rejection sampling to numerically integrate respective arguments over realizations of σ_{-i} and β_{-i}.

- evaluating $\mathbb{E}^{\sigma_{-i}}\left[\cdot\middle|Piv\right]$ requires the conditional probabilities, $\mathbb{P}\left(\sigma_{-i}|Piv\right)$, which differ from unconditional probabilities. For example, if the opponent message is "to omit" (e.g. $m_{-i}=\emptyset$), then it is more likely that the opponent had a large standard error so long as c is close to the prior. The adjustment of probability distribution is made by applying the Bayes rule:

$$\mathbb{P}\left(\sigma|Piv\right)=\frac{\mathbb{P}\left(\sigma\right)\mathbb{P}\left(Piv|\sigma\right)}{\mathbb{P}\left(Piv\right)},$$

  where $\mathbb{P}\left(Piv|\sigma\right)$ is computed from each round of $\texttt{Eb}(\texttt{Q},\texttt{M},\texttt{B})$.

- we evaluate $\mathbb{E}^{\beta_{-i}}\left[\sum\frac{\beta_{-i}}{\sigma_{-i}^2}|m_{-i},\beta_i\right]$ with `R` draws of β_{-i} realizations given specified mean and a variance-covariance matrix. To make computation efficient so that a large `R` can be applied, I use the same vector of random number in every step of iteration. Note that conditional distribution β_{-i} with $\texttt{R}\times(\texttt{N}-\texttt{1})$ is

$$\begin{bmatrix}\beta_2\\ \beta_3\\ \vdots\\ \beta_N\end{bmatrix}\Bigg|\beta_1 \sim\mathcal{N}\left(\begin{bmatrix}\beta_0+\frac{\sigma_2}{\sigma_1}\rho_{12}\left(\beta_1-\beta_0\right)\\ \beta_0+\frac{\sigma_3}{\sigma_1}\rho_{13}\left(\beta_1-\beta_0\right)\\ \vdots\\ \beta_0+\frac{\sigma_N}{\sigma_1}\rho_{1N}\left(\beta_1-\beta_0\right)\end{bmatrix},\begin{bmatrix}\sigma_2^2+\frac{\sigma_1^2}{\sigma_1^2+\sigma_0^2}\sigma_0^2 & \frac{\sigma_1^2}{\sigma_1^2+\sigma_0^2}\sigma_0^2 & \cdots & \frac{\sigma_1^2}{\sigma_1^2+\sigma_0^2}\sigma_0^2\\ \frac{\sigma_1^2}{\sigma_1^2+\sigma_0^2}\sigma_0^2 & \sigma_3^2+\frac{\sigma_1^2}{\sigma_1^2+\sigma_0^2}\sigma_0^2 & & \\ \vdots & & \ddots & \\ \frac{\sigma_1^2}{\sigma_1^2+\sigma_0^2}\sigma_0^2 & & & \sigma_N^2+\frac{\sigma_1^2}{\sigma_1^2+\sigma_0^2}\sigma_0^2\end{bmatrix}\right), \tag{4}$$

  where $i=1$ is index of the researcher himself. The algoriithm computes the truncated mean and probabilities of this distribution in the following three steps:

Step 1. generate $\texttt{R}\times(\texttt{N}-\texttt{1})$ standard normal random variables ϵ_i (noise), and $\texttt{R}\times\texttt{1}$ normal random variable b with standard deviation σ_0;

Step 2. for each realization of σ_i in $\underbrace{S\times...\times S}_{N}$, apply formula $\mu_i=\sqrt{\frac{\sigma_i^2}{\sigma_i^2+\sigma_0^2}}b$ to generate the random variables of the desired mean and variance of the underlying benefit distribution;

Step 3. at each iteration of threshold computation, add constant $\beta_0+\sigma_{-i}\epsilon_{-i}+\frac{\sigma_{-i}}{\sigma_i}\rho_{i,-i}\left(\beta_1-\beta_0\right)$ to the random vector to adjust the mean.

2.2. Iterative Computation

I use iterative methods to find the solution to the system of non-linear equations with following steps.

1. conjecture a solution and compute one value of threshold for some σ using the formula (1) and (2);
2. update the conjectured threshold in the direction of this solution

$$\beta_{new}=\texttt{Change}\beta_{sol}+\left(1-\texttt{Change}\right)\beta_{conjecture},$$

   where β_{new} denotes the updated threshold value, β_{sol} denotes the solution to the non-linear equation, and $\beta_{conjecture}$ denotes the original conjecture from 1. The updates apply across all researchers (impose symmetry in the solution);
3. repeat this until the sum of all updates from every threshold is less than tolerance level `TolDiff`.

Although I initially began with the gradient-based and gradient-free methods to solve the system of nonlinear equations, they performed poorly when N was greater than 3. This iterative method is based on the idea of local stability of the equilibrium: when the iterative adjustment is made steadily, the solution converges to the equilibrium.

To obtain a reasonable conjecture of the solution, I first fix the environment and then apply the solution to the problem of $N-1$ obtained under the same environment. This methodology is analogous to the homotopy method. For $N=2$, I start with some linear function such that $\overline{\beta}\left(\sigma\right)>c$ is increasing and $\underline{\beta}\left(\sigma\right)<c$ is decreasing. The solutions are not sensitive to the choice of the initial linear functions.

The users have to choose appropriate `TolDiff` and `Change` variable. I have set `TolDiff` $=0.05$ and `Change` $=0.5$ for this computation. The optimal rate of adjustment `Change` is less than 1 in some environment because while the methodology relies on the local stability of the solution, the computational update is necessarily discrete. The gradual adjustment is effective in avoiding possible fluctuations.

3. Discussions

There are two limitations of the algorithm that restricts the range of environment that this computation can take place effectively:

- **fragibility of estimation when thresholds are at the tail:** estimating the $\mathbb{E}^{\beta_{-i}} \sum \beta_{-i}$ may not be accurate when the thresholds will be at the tail of normal distribution. If the thresholds β_{-i} were at the tail, then there are no samples drawn from such message realization. Even if we may increase `R`, the number of samples drawn, this may not be effective due to the thin tail property of normal distribution. This problem becomes quickly severe as N increases because the rectangle becomes much smaller compared to elipse. Thus, we cannot set the value of c to be away from the mean, b_0, of the signal distribution.

- **computation time:** the indifference conditions in (1) and (2) exhibit recursive use of expectation operators over σ_{-i} and m_{-i} realizations. To draw the thresholds $\beta(\sigma)$ smoothly, S has to be sufficiently large (e.g. $S = 10$.) When N is large, there are many realization of m_{-i} that can generate either $n_1' = n_0'$ or $n_1' = n_0' + 1$. The computational burden increases by the order of $S^N \times \sum_{k=0}^{\lfloor N/2 \rfloor} \left(\begin{array}{c} N \\ k \end{array} \right)$, where S^N represents the size of σ realizations and $\sum_{k=0}^{\lfloor N/2 \rfloor} \left(\begin{array}{c} N \\ k \end{array} \right)$ reprsents the size of m_{-i} realizations. For example, if $N = 10$, $S = 10$, then the computational time is approximately trillion times that of $N = 2$, $S = 1$. For computational feasibility, the simulation in Figure B6 computes up to $N = 4$.