Matlab files for Pareto Extrapolation

Émilien Gouin-Bonenfant* Alexis Akira Toda†

1 Introduction

This note explains the functionalities in the Matlab package `PE`, which implements the Pareto Extrapolation algorithm. The user should use these files at their own responsibility. Whenever you use these codes for your research, please cite Gouin-Bonenfant and Toda (2022).

2 Package content

There are three main functionalities for Pareto extrapolation:

- `getZeta.m`
- `getQ.m`
- `getTopShares.m`

In addition, `expGrid.m` constructs an exponential grid and `example.m` contains a simple example.

2.1 Pareto exponent

`getZeta.m` computes the Pareto exponent using the Beare and Toda (2017) formula. The usage is

```
[zeta,typeDist] = getZeta(PS,PJ,V,G,zetaBound)
```

where

- `PS` is the $S \times S$ transition probability matrix of exogenous states indexed by $s = 1, \dots, S$,
- `PJ` is the $S^2 \times J$ matrix of conditional probabilities of transitory states indexed by $j = 1, \dots, J$,

*Department of Economics, Columbia University. Email: eg3041@columbia.edu.
†Department of Economics, University of California San Diego. Email: atoda@ucsd.edu.

- `V` is the $S \times S$ matrix of conditional survival probabilities,
- `G` is the $S^2 \times J$ matrix of gross growth rates,
- `zetaBound` is a vector $(\underline{\zeta}, \bar{\zeta})$ that specifies the lower and upper bounds to search for the Pareto exponent (optional),
- `zeta` is the Pareto exponent, and
- `typeDist` is the probability distribution of types in the upper tail.

The S^2 rows in `PJ` and `G` should be ordered such that

$$(s, s') = (1, 1), \dots, (1, S); \dots; (s, 1), \dots, (s, S); \dots; (S, 1), \dots, (S, S).$$

If `PS` $= P = (p_{ss'})$, `PJ` $= (\pi_{ss'j})$, `V` $= (v_{ss'})$, and `G` $= (G_{ss'j})$, then the Pareto exponent $z = \zeta$ is the solution to

$$\rho(P \odot V \odot M(z)) = 1,$$

where ρ is the spectral radius and $M(z) = (M_{ss'}(z))$,

$$M_{ss'}(z) = \sum_{j=1}^{J} \pi_{ss'j} G_{ss'j}^{z},$$

and $\odot$ is the Hadamard (entry-wise) product.

`PJ` must be either $1 \times J$, $S \times J$, or $S^2 \times J$. If it is $1 \times J$, it assumes $\pi_{ss'j} = \pi_j$ depends only on j. If it is $S \times J$, it assumes $\pi_{ss'j} = \pi_{sj}$ depends only on (s, j).

`V` must be either 1×1 or $S \times S$. If it is 1×1, it assumes $v_{ss'} = v$ is constant.

`G` must be either $S \times J$ or $S^2 \times J$. If it is $S \times J$, it assumes $G_{ss'j} = G_{sj}$ depends only on (s, j).

2.2 Joint transition probability matrix

`getQ.m` computes the $SN \times SN$ joint transition probability matrix $Q = (q_{sn,s'n'})$ and the stationary distribution $\pi = (\pi_{sn})$ for the exogenous state s and wealth. The usage is

```
[Q,pi] = getQ(PS,PJ,V,x0,xGrid,gstjn,Gstj,zeta,h)
```

where

- `PS`, `PJ`, `V` are the same as in `getZeta.m`,
- `x0` is the initial wealth of newborn agents,
- `xGrid` is the $1 \times N$ grid of wealth (size variable) w_n,
- `gstjn` is the $S^2 \times JN$ matrix of law of motion for wealth $g_{ss'j}(w_n)$,

- `Gstj` is the $S^2 \times J$ matrix of asymptotic slopes of law of motion $G_{ss'j}$ (optional),
- `zeta` is the Pareto exponent (optional),
- `h` is the grid spacing for hypothetical grid points (optional),
- `Q` is the $SN \times SN$ joint transition probability matrix, and
- `pi` is the $SN \times 1$ stationary distribution.

The JN columns of `gstjn` must be ordered such that the first N columns correspond to $j = 1$, the next N columns correspond to $j = 2$, and so on. `Gstj` is the same as `G` in `getZeta.m`. If unspecified, it uses the slope of the law of motion between the two largest grid points. If `zeta` is unspecified, it calls `getZeta.m` to compute. If `h` is unspecified, it uses the distance between the two largest grid points (which we recommend).

`gstjn` must be either $S \times JN$ or $S^2 \times JN$. If it is $S \times JN$, it assumes $g_{ss'j}(w_n) = g_{sj}(w_n)$ depends only on (s, j, n).

2.3 Top wealth shares

`getTopShares.m` computes the top wealth shares. The usage is

```
topShare = getTopShares(topProb,wGrid,wDist,zeta)
```

where

- `topProb` is the vector of top probabilities to evaluate top shares,
- `wGrid` is the $1 \times N$ vector of wealth grid,
- `wDist` is the $1 \times N$ vector of wealth distribution, and
- `zeta` is the Pareto exponent (optional).

Given the stationary distribution π computed using `getQ.m`, one can compute the wealth distribution as $\pi_n = \sum_{s=1}^{S} \pi_{sn}$. If `zeta` is unspecified, `getTopShares.m` uses spline interpolation to compute top wealth shares.

2.4 Exponential grid

`expGrid.m` constructs an N-point exponential grid on the interval $(a, b]$. The usage is

```
grid = expGrid(a,b,c,N)
```

where

- `a`, `b` are endpoints,
- `c` is the median point satisfying $a < c < \frac{a+b}{2}$, and

- `N` is the number of grid points.

We exclude the lower endpoint a because it is often an absorbing state, but it is straightforward to modify the code to construct a grid on $[a, b]$.

References

Brendan K. Beare and Alexis Akira Toda. Determination of Pareto exponents in economic models driven by Markov multiplicative processes. Revise and resubmit at *Econometrica*, 2017. URL https://arxiv.org/abs/1712.01431.

Émilien Gouin-Bonenfant and Alexis Akira Toda. Pareto extrapolation: An analytical framework for studying tail inequality. *Quantitative Economics*, 2022. URL https://ssrn.com/abstract=3260899. Forthcoming.