ReadMe

24th September 2024

Contents

1 Notations

We denote $X(t) \in \mathbb{Z}^{\mathbb{Z}}$ as the configuration of particles' position at time t.

2 Lists

1. Matlab Code:
   (a) Folder **Utility Functions** (see Section 3.1 for explanation):
      i. `drawArrow.m`
      ii. `MY_intersect.m`
      iii. `MY_setdiff.m`
      iv. `find_next_jump_time.m`
   (b) Folder **Argmax Code** (see Section 3.2 for explanation):
      i. `Polymer_joint_distribution.m`
      ii. `Polymer_psi.m`
      iii. `Polymer_T_pdf.m`
      iv. `PolymerKernel.m`
   (c) Folder **ASEP Code** (see Section 3.3 for explanation):
      i. `ASEP.m`
      ii. `Monte_Carlo_ASEP.m`
      iii. `find_discrepancy.m`
   (d) Folder **ASEPsc Code** (see Section 3.4 for explanation):
      i. `ASEPsc.m`
      ii. `Monte_Carlo_ASEPsc.m`
2. Data
   (a) Folder **Argmax_Data** (see Section 3.2.3 for explanation)
      i. `End_point_Polymer_T_cdf.txt`
      ii. `End_point_Polymer_T_pdf.txt`
   (b) Folder **ASEP_Data** (see Section 3.3.3 for explanation)
      i. Folder **Difference**:
         A. `Diff_Frequncy_200.txt`
         B. `Diff_Frequncy_400.txt`
         C. `Diff_Frequncy_600.txt`
         D. `Diff_Frequncy_800.txt`
         E. `Diff_Frequncy_1000.txt`
         F. `Diff_Frequncy_1200.txt`
         G. `Diff_Frequncy_1400.txt`
         H. `Diff_Frequncy_1600.txt`

I. `Diff_Frequncy_1800.txt`
J. `Diff_Frequncy_2000.txt`

ii. Folder **EndPointBKP**:

A. `EndPoint_BKP_ASEP_Frequncy_500.txt`
B. `EndPoint_BKP_ASEP_Frequncy_1000.txt`
C. `EndPoint_BKP_ASEP_Frequncy_1500.txt`
D. `EndPoint_BKP_ASEP_Frequncy_2000.txt`
E. `EndPoint_BKP_ASEP_Frequncy_2500.txt`
F. `EndPoint_BKP_ASEP_Frequncy_3000.txt`
G. `EndPoint_BKP_ASEP_Frequncy_3500.txt`
H. `EndPoint_BKP_ASEP_Frequncy_4000.txt`
I. `EndPoint_BKP_ASEP_Frequncy_4500.txt`
J. `EndPoint_BKP_ASEP_Frequncy_5000.txt`

iii. Folder **OriginalPosition**

A. `Original_Position_ASEP_Frequncy_200.txt`
B. `Original_Position_ASEP_Frequncy_400.txt`
C. `Original_Position_ASEP_Frequncy_600.txt`
D. `Original_Position_ASEP_Frequncy_800.txt`
E. `Original_Position_ASEP_Frequncy_1000.txt`
F. `Original_Position_ASEP_Frequncy_1200.txt`
G. `Original_Position_ASEP_Frequncy_1400.txt`
H. `Original_Position_ASEP_Frequncy_1600.txt`
I. `Original_Position_ASEP_Frequncy_1800.txt`
J. `Original_Position_ASEP_Frequncy_2000.txt`

(c) Folder **ASEPsc_Data** (see Section 3.4.3 for explanation)

i. Folder **EndPointBKP**

A. `EndPointBkp_ASEPsc_Frequncy_100.txt`
B. `EndPointBkp_ASEPsc_Frequncy_200.txt`
C. `EndPointBkp_ASEPsc_Frequncy_300.txt`
D. `EndPointBkp_ASEPsc_Frequncy_400.txt`
E. `EndPointBkp_ASEPsc_Frequncy_500.txt`
F. `EndPointBkp_ASEPsc_Frequncy_600.txt`
G. `EndPointBkp_ASEPsc_Frequncy_700.txt`
H. `EndPointBkp_ASEPsc_Frequncy_800.txt`
I. `EndPointBkp_ASEPsc_Frequncy_900.txt`
J. `EndPointBkp_ASEPsc_Frequncy_1000.txt`

ii. Folder **OriginalPosition**

A. `Particle_Position_100.txt`
B. `Particle_Position_200.txt`
C. `Particle_Position_300.txt`

D. `Particle_Position_400.txt`
E. `Particle_Position_500.txt`
F. `Particle_Position_600.txt`
G. `Particle_Position_700.txt`
H. `Particle_Position_800.txt`
I. `Particle_Position_900.txt`
J. `Particle_Position_1000.txt`

3. *Mathematica* code: `DataAnalysis.mb` is used to analyze the data in the files mentioned above.

Remark 2.1. *Let $X(t)$ be a random process, we will use the following rule to store its simulation results.*

1. *The naming convention follows the pattern:* *<Process>_<Setting>_<Quantity>_<Time>.txt.*
   (a) *<Process> refers to the name of the random process, such as "Original_Position" for $X(t)$.*
   (b) *<Setting> indicates the model setting, for instance, "ASEP" for the Asymmetric Simple Exclusion Process.*
   (c) *<Quantity> specifies the type of data, like "Frequency".*
   (d) *<Time> is an integer representing the specific time t, such as "2000" for Monte Carlo data at $X(2000)$.*
2. *The format of the data is a two-column matrix, where the first column contains the values of $X(t)$, and the second column represents the frequency of each specific value.*

3 Matlab Code

3.1 Utility Functions

1. `drawArrow.m` contains a function to draw an arrow between two 2D points A and B.
2. `MY_intersect.m` contains a function to compute $A \cap B$ for two sets A and B.
3. `MY_setdiff.m` contains a function to compute $A \setminus B$ for two sets A and B.
4. `find_next_jump_time.m`: For a given vector $v \in \mathbb{R}^N$ and a matrix $M \in \mathbb{R}^{M \times N}$, this file returns $\nu \in \mathbb{R}^N$ such that

$$\nu_i := \max\{M_{i,r} \mid M_{i,r} > v_i\}, \quad \forall i. \tag{3.1}$$

3.2 Implementation for $\operatorname{argmax}\{\mathcal{A}_2(u) - u^2\}$

3.2.1 Description

The implementation of distribution of $\hat{u} := \operatorname{argmax}\{\mathcal{A}_2(u) - u^2\}$ is based on the formula in[1] [4]. Let Ai be Airy function and operator P_0 as the projection on positive real line, i.e., $P_0 f(x) = \mathbb{1}_{x>0} f(x)$. For $t, m \in \mathbb{R}$, define the function $B_m(x, y) := \mathrm{Ai}(x + y + m)$ and also

$$\psi_{t,m}(x) := 2e^{xt} \left[t \,\mathrm{Ai}\left(x + m + t^2\right) + \mathrm{Ai}'\left(x + m + t^2\right) \right] \tag{3.2}$$

and the kernel

$$\Psi_{t,m}(x, y) := 2^{1/3} \psi_{t,m}\left(2^{1/3} x\right) \psi_{-t,m}\left(2^{1/3} y\right). \tag{3.3}$$

Define $\mathcal{M} := \max\{\mathcal{A}_2(u) - u^2\}$, the formula of joint density of $(\hat{u}, \mathcal{M})$ is given by [4, Theorem 2]:

$$f(t, m) = \det\left(I - P_0 B_{4^{1/3}} P_0 + P_0 \Psi_{t,m} P_0\right) - F_{\mathrm{GOE}}\left(4^{1/3} m\right), \tag{3.4}$$

where F_{GOE} is GOE Tracy-Widom distribution. Both quantities on the right hand side of (3.4) can be obtained numerically via Bornemann's method [2]. Integrating over $\mathcal{M}$, we then obtain the distribution of $\hat{u}$.

3.2.2 Code

1. `Polymer_joint_distribution.m` contains implementation of (3.4).
2. `Polymer_psi.m` contains implementation of (3.2).
3. `Polymer_T_pdf.m` contains implementation of

$$f_{\hat{u}}(t) := \int_{\mathbb{R}} \mathrm{d}m f(t, m), \tag{3.5}$$

i.e., the probability density function of $\hat{u}$.

[1] Another formula is obtained in [5], and it was shown in [1] that they are the same.

4. `PolymerKernel.m` contains implementation of the kernel $P_0 B_{4^{1/3}} P_0 + P_0 \Psi_{t,m} P_0$ in (3.4).

Remark 3.1. *In order to run these programs properly, one needs Bornemann's Fredholm determinant toolbox [3].*

3.2.3 Data

The data is stored under the path "Mathematica/Argmax_Data".

The file `End_point_Polymer_T_pdf.txt` (resp. `End_point_Polymer_T_cdf.txt`) contains the discretization of the probability density function (resp. distribution function) $f_{\hat{u}}(\cdot)$ (resp. $F_{\hat{u}}(\cdot)$) on the interval $[-2, 2]$ with a mesh size of 0.01.

3.3 Implementation for ASEP

3.3.1 Description

We have simulated ASEP with asymmetric parameter p. Denote the total simulation time as $t_{\max}$ and the number of particles as N. Initially, we have $X_n(0) = 2n-1$ for $n \in \{1, \dots, N\}$. We will first prepare three random matrix $\mathcal{J}, \mathcal{U}, \mathcal{D} \in \mathcal{R}_{\geq 0}^{N \times C_N t_{\max}}$ for some $C_N > 0$ depending on N in the following way:

1. $\mathcal{J}_{i,j} - \mathcal{J}_{i-1,j} \sim Exp(1)$ is i.i.d. random variable. Column n of this matrix provides the information when particle X_n will attempt to jump: the $m-$the attempted jump time of particle X_n is $\mathcal{J}_{m,n}$.

2. $\mathcal{U}_{i,j} \sim Unif([0,1])$ is i.i.d. random variable and

$$\mathcal{D}_{i,j} := \begin{cases} 1, & \text{if } \mathcal{U}_{i,j} \leq p, \\ -1, & \text{otherwise.} \end{cases} \tag{3.6}$$

   The $n-$th column of $\mathcal{D}$ provides the information about how particle X_n should attempt to jump: if $\mathcal{D}_{m,n} = 1$, then the dirction of $m-$th attempted jump of particle X_n is from left to right otherwise from right to left.

In the simulation, we will choose C_N (depending on N) large enough such that $\min_n\{\sum_{i=1}^{C_N t_{\max}} \mathcal{J}_{i,n}\} \geq t_{\max}$ with high probability. For a given initial condition $X(0)$ and matrices $\mathcal{J}, \mathcal{D}$, the final state $X(t)$ is a deterministic result. We record current particle's time-space position in two vectors $\mathcal{S}, \mathcal{T} \in \mathcal{R}_{\geq 0}^{N+1}$ with $\mathcal{T}_1 = \mathcal{T}_{N+1} := t_{\max}$ and $\mathcal{S}_{N+1} := \infty$. The information about next attempted jump direction is stored in vector $\mathcal{A} \in \{-1, 1\}^N$. The updating rule is as follows:

1. **Inital step:** Set $\mathcal{S}_n = X_n(0)$, $\mathcal{T}_n = \mathcal{J}_{1,n}$, $\mathcal{A}_n = \mathcal{D}_{1,n}$ and $\mathcal{H}_n = 2$ for all $n \in \{1, \dots, N\}$.

2. **Induction step:** For given $\mathcal{S}$, $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{H}$, until $\min\{\mathcal{T}_n\} \geq t_{\max}$, we choose $m \in \{2, \ldots, N\}$ such that $\mathcal{T}_m \leq \min\{\mathcal{T}_{m+1}, \mathcal{T}_{m-1}\}$ and update

$$\mathcal{S}_m = \begin{cases} \mathcal{S}_m + 1, & \text{if } \mathcal{A}_m = 1 \text{ and } \mathcal{S}_{m+1} \neq \mathcal{S}_m + 1, \\ \mathcal{S}_m - 1, & \text{if } \mathcal{D}_m = -1 \text{ and } \mathcal{S}_{m-1} \neq \mathcal{S}_m - 1, \\ \mathcal{S}_m, & \text{otherwise.} \end{cases} \tag{3.7}$$

and update its current time position as $\mathcal{T}_m := \mathcal{J}_{\mathcal{H}_m, m}$. In the case when $\mathcal{A}_m = 1$ and $\mathcal{S}_{m+1} = \mathcal{S}_m + 1$ (resp. $\mathcal{A}_m = -1$ and $\mathcal{S}_{m-1} = \mathcal{S}_m - 1$), this jump will be recorded as suppressed left to right jump (resp. right to left jump) and stored in $\mathcal{T}^{\not\to}$ (resp. $\mathcal{T}^{\not\leftarrow}$). Updated $\mathcal{A}_m := \mathcal{D}_{\mathcal{H}_m, m}$ and $\mathcal{H}_m := \mathcal{H}_m + 1$.

End-point of backwards path in ASEP

For a given time $t \in [0, t_{\max}]$ and $n \in \{1, \ldots, N\}$, the end-point of backwards path of particle X_n is obtained as follows:

1. **Initial step:** At time t, we set $N(t \downarrow t) = n$.
2. **Induction step:** Given $m = N(t \downarrow s)$, find $\tau := \max\{r \leq s \mid r \in \mathcal{T}_m^{\not\to} \cup \mathcal{T}_m^{\not\leftarrow}\}$. If $\tau \in \mathcal{T}_m^{\not\to}$ (resp. $\tau \in \mathcal{T}^{\not\leftarrow}$), then update $N(t \downarrow \tau) = m + 1$ (resp. $N(t \downarrow \tau) = m - 1$).

Discrepancy in ASEP

For a given $t \in [0, t_{\max}]$ and $n \in \{1, \ldots, N\}$, after obtaining $N(t \downarrow 0)$ described as above, we will implement

$$D_n(t) := X^{\text{step}}_{n - N(t\downarrow 0)}(t) + 2N(t \downarrow 0) - X_n(t) \tag{3.8}$$

as follows:

1. Define a new ASEP $\hat{X}(t)$ with initial condition $\hat{X}_j(0) = X_{N(t\downarrow 0)}(0) - (N(t \downarrow 0) - j)$ for all $j \leq N(t \downarrow 0)$.
2. Applying same realization of random matrix $\mathcal{J}$ and $\mathcal{D}$ as the one for original process $X(t)$: for $n \leq N(t \downarrow 0)$, $\mathcal{J}_{i,j}$ (resp. $\mathcal{D}_{i,j}$) is the $i-$th attempted jump time (resp. the direction of $i-$th attempted jump) of particle $\hat{X}_j$.

Then $D_n(t)$ in (3.8) is implemented by $\hat{X}_n(t) - X_n(t)$.

3.3.2 Code

The main parts of the code for ASEP is contained in file `ASEP.m`, where we define a class called ASEP. It has essentially four methods:

1. Construction function: The function *ASEP* constructs an ASEP object after obtaining basic inputs such as the random seed, initial condition, asymmetric parameter, and so on. In particular, the matrices $\mathcal{J}$ and $\mathcal{D}$ are generated here.

2. Evolution function: The function *jump* evolves the dynamics according to the rule introduced above.

3. Backwards index: The function *backwards_index* delivers $N(t \downarrow 0)$ for particle X_n at time t. The dynamics are as described above.

4. Draw Harris Graph: The function *draw_graph* outputs an image showing the space-time trajectories of all particles.

Apart from `ASEP.m`, there is also a file called `find_discrepancy.m`, where we implement $D_n(t)$ in (3.8) as described above. Finally, the Monte Carlo code is contained in file `Monte_Carlo_ASEP.m`. The raw data is given in folder **Raw Data**.

3.3.3 Data

1. Folder **Difference** contains the data for $D_n(t_i)$ (see (3.8) for definition) with $t_i \in \{200, 400, \ldots, 2000\}$.

2. Folder **EndPointBKP** contains the data for $X_{N(t\downarrow 0)}$ with $t_i \in \{500, 1000, \ldots, 5000\}$.

3. Folder **OriginalPosition** contains the data for $X_n(t_i)$ with $t_i \in \{200, 400, \ldots, 2000\}$.

3.4 Implementation for ASEPsc

3.4.1 Description

We will store the current space-time position of particles in $X(t)$ in vectors $\mathcal{S}$ and $\mathcal{T}$ respectively. For **random sequential update**, we also need $\mathcal{R} \in \mathbb{R}^N$ to store the current jump scheme of each particle and $\mathcal{P} \in \mathbb{R}^{N+1}$ for the probability of each particle being chosen. We also label the jump schemes in Figure 1 as $1, \ldots, 16$. For given jump rate, we then determine $X(t)$ as follows:

1. **Initial step:**

   (a) For space-time trajectory: set $\mathcal{S}_n = 2n - 1$ and $\mathcal{T}_n = 0$ for all n.

   (b) For jump schema: For each $n \in \{1, \ldots, N\}$, we find its left (resp. right) jump rate θ_ℓ^n (resp. θ_r^n) according to its local enviroment and Figure 1. Define $\mathcal{R}_n := \theta_\ell^n + \theta_r^n$.

   (c) For probability being choosen: Let $\mathcal{P}_n := (\sum_{i=1}^n \mathcal{R}_i)/(\sum_{i=1}^N \mathcal{R}_i)$ for all $n \geq 1$ and $\mathcal{P}_0 := 0$.

2. **Induction step:** For given $\mathcal{S}$, $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{P}$, until $\min_n\{\mathcal{T}_n\} \geq t_{\max}$, we let $\mathcal{U} \sim Unif([0,1])$ and find $m \in \{1, \ldots, N\}$ such that $\mathcal{P}_{m-1} < \mathcal{U} \leq \mathcal{P}_m$. Then we update the $\mathcal{S}$, $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{P}$ as follows:

   (a) **For time position:** Set $\mathcal{T}_n := \mathcal{T}_n + \mathcal{J}$, where $\mathcal{J} \sim Exp(\mathcal{R}_m)$ for all n.

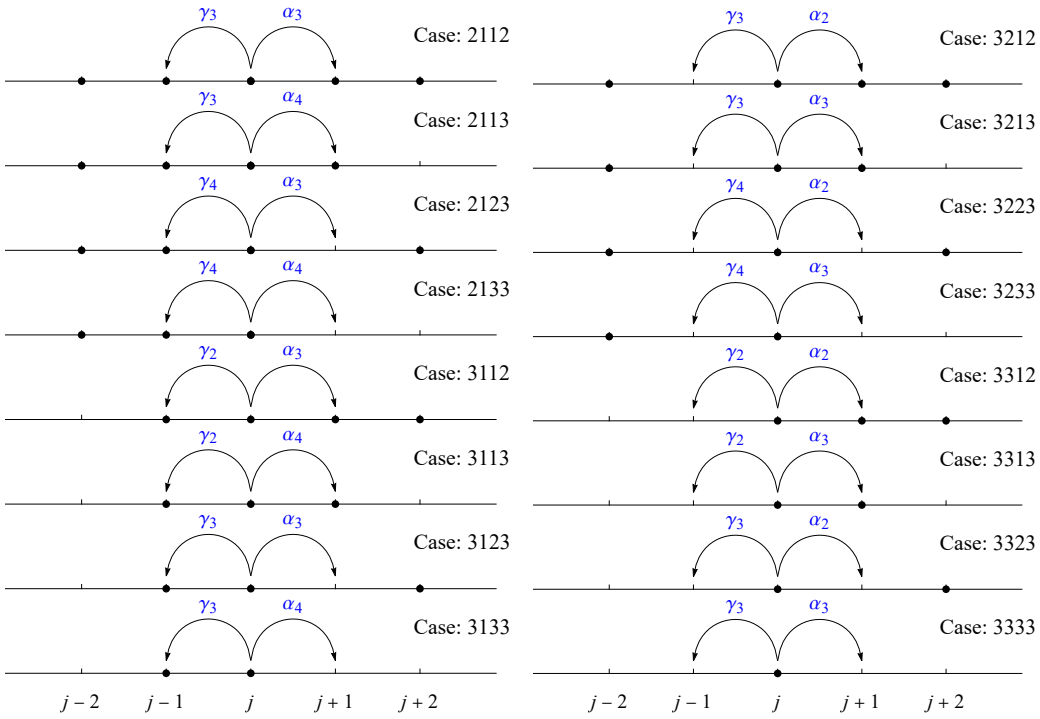

Figure 1: Illustration of the jump rate used in the simulation for ASEPsc. The four digits represent the distances between the position and its neighbors. For example, 2112 means the distance to the second-left neighbor is 2, to the nearest-left neighbor is 1, to the nearest-right neighbor is 1, and to the second-right neighbor is 2. If the distance is greater than 2, the digit is 3.

(b) **For space position:** Let $\mathcal{K} \sim Unif([0,1])$ be i.i.d. and set

$$\mathcal{S}_m := \begin{cases} \mathcal{S}_m + 1, & \text{if } \mathcal{K}_m \le \frac{\theta_r^m}{\theta_\ell^m + \theta_r^m} \text{ and } \mathcal{S}_{n+1} > \mathcal{S}_m + 1, \\ \mathcal{S}_m - 1, & \text{if } \mathcal{K}_m > \frac{\theta_r^m}{\theta_\ell^m + \theta_r^m} \text{ and } \mathcal{S}_n > \mathcal{S}_{n-1} + 1, \\ \mathcal{S}_m, & \text{otherwise.} \end{cases} \tag{3.9}$$

In the case when $\mathcal{K}_m \le \frac{\theta_r^m}{\theta_\ell^m + \theta_r^m}$ and $\mathcal{S}_{n+1} = \mathcal{S}_m + 1$ (resp. $\mathcal{K}_m > \frac{\theta_r^m}{\theta_\ell^m + \theta_r^m}$ and $\mathcal{S}_n = \mathcal{S}_{n-1} + 1$), we mark this attempted jump as suppressed right (resp. left) jump.

(c) Updated $\mathcal{P}$ and $\mathcal{R}$ according to current status of $\mathcal{S}$ and Figure 1.

After constructing the process $X(t)$, the end-point of backwards path can be constructed as same as the one for ASEP.

3.4.2 Code

The main parts of the code for ASEP is contained in file `ASEPsc.m`, where we define a class called ASEP. It has essentially four methods:

1. Construction function: Function *ASEPsc* willl construct an object of ASEP after obtainging the basic input such as random seed, initial condition, β and E and so on.

2. Evolution function: Function *jump* will let the dynamic evolve according to the rule we introduced above.

3. Backwards index: Function *backwards_index* will deliver $N(t \downarrow 0)$ of particle X_n at time t. The dynamic is given as described above.

4. Draw Harris Graph: Function *draw_graph* will output a image to demonstrate the space-time trajectories of all particles.

Apart from `ASEPsc.m`, there is also a file called `Monte_Carlo_ASEP.m` for Monte Carlo simulations. The raw data is given in folder **Raw Data**.

3.4.3 Data

1. Folder **EndPointBKP** contains the data for $X_{N(t\downarrow 0)}$ with $t_i \in \{100, 200, \ldots, 1000\}$.

2. Folder **OriginalPosition** contains the data for $X_n(t_i)$ with $t_i \in \{100, 200, \ldots, 1000\}$.

References

[1] J. Baik, K. Liechty, and G. Schehr, *On the joint distribution of the maximum and its position of the Airy$_2$ process minus a parabola*, J. Math. Phys. **53** (2012), 083303.

[2] F. Bornemann, *On the numerical evaluation of Fredholm determinants*, Math. Comput. **79** (2009), 871–915.

[3] A. Borodin, *Loop-free markov chains as determinantal point processes*, Ann. Inst. H. Poincaré Probab. Statist. **44** (2008), 19–28.

[4] G. Moreno Flores, J. Quastel, and D. Remenik, *Endpoint distribution of directed polymers in 1+ 1 dimensions*, Ann. Inst. H. Poincaré Probab. Statist. **51** (2015), 1–17.

[5] G. Schehr, *Extremes of N vicious walkers for large N: application to the directed polymer and KPZ interfaces*, Journal of Statistical Physics **149** (2012), no. 3, 385–410.