i) $\pi(x) = \frac{1}{Z_\beta} e^{-\beta V(x)}$, $V(x) \geq 0$, $x \in E$, $|E| < \infty$.

Let $M(x,y)$ be a reversible proposal transition, and let $\alpha = \alpha(x,y) := 1 \wedge \left(\frac{\pi(y)}{\pi(x)} \cdot \frac{M(y,x)}{M(x,y)} \right) = 1 \wedge \frac{\pi(y)}{\pi(x)} = e^{-\beta (V(y) - V(x))_+}$ (in our case)

The algorithm is defined as follows:

```
x ← GetInitialState()        ← tuning parameter
for i in range(N_steps):
    y ← GenerateNewState(x, M)
    u ← GetRandomNumber()    (~ U(0,1))
    if u < e^{-β(V(y)-V(x))}:
        x ← y
```

It means that at each step we do a simple Markov transition according to M, and we accept the transition with probability α. The larger β, the less we accept.

This 2-step transition is also a Markov chain, which corresponds to some matrix K. It can be shown that $\pi = \pi K$. So if $\exists m: K^m(x,y) > 0 \ \forall x, y \in E$, then the algorithm works: the generated chain converges to π (according to the theorem) exponentially fast.

ii) For each n we do the MH algorithm with β_n, passing the final state as an initial state to the MH algorithm with β_{n+1}. Pseudocode:

```
x ← GetInitialState()        ← cannot do that for every n ∈ ℕ, obviously
for n in range(N):           ← tuning parameters
    x ← MHAlgo(β_n, x, M, N_steps^n)    (function returns the last state)
```

The larger β, the more π_β concentrates on V's global minimums, because probabilities of other points tend to 0 exponentially fast. In fact, $\pi_{\beta_n} \xrightarrow{n \to \infty} \pi^*$: $\pi^*(x) = \frac{\mathbb{1}(x \in V^*)}{|V^*|}$, where $V^* = \{ x \in E : \forall y \in E \ V(x) \leq V(y) \}$.

So if $\beta_n \to \infty$, then we can consider the SA algorithm as an algorithm of finding global minimums of V. It's important that we can approach this problem gradually, because if we take $\beta \approx \infty$ initially, then we'll reject every transition, ending up in a local minimum of V. In SA we explore the state space (with small β_n's) first, and step by step we make our transitions more accurate, ending up sampling global minimums uniformly.

iii) Let $(\xi_0^i)_{i=\overline{0,N-1}}$ be some initial states. In order to get (ξ_{n+1}^i) from (ξ_n^i), we do the following:

$\xi_n^i \xrightarrow[\text{(mutation)}]{\text{MH with } \beta_n} \hat{\xi}_n^i \xrightarrow{\text{(selection)}} \tilde{\xi}_n^i = \begin{cases} \hat{\xi}_n^i & \text{with probability } e^{-(\beta_{n+1} - \beta_n) V(\hat{\xi}_n^i)} \\ \text{otherwise we sample from } \pi_n^{(dx)} := \sum_{i=0}^{N-1} \frac{e^{-(\beta_{n+1} - \beta_n) V(\hat{\xi}_n^i)}}{\sum_j e^{-(\beta_{n+1} - \beta_n) M(\hat{\xi}_n^j, \hat{\xi}_n^i)}} \delta(dx) \end{cases}$

It means that on selection step $\forall i$ we pick a point that is closer to V's minimum (likely), because the less $V(x)$, the larger the probability that we pick x.

A good thing is that mutation step can be done in parallel, as well as selection step. By increasing N, we generate more candidates on mutation step, and we pick the best out of them on selection step. If $N = 1$, then it's just the SA algorithm.

Tradeoff: the larger N ~ the better quality ~ the better server is needed

$Y = \sqrt{1-\varepsilon}\, x + \sqrt{\varepsilon}\cdot W \sim M(x, dy)$, $\pi(dx) = \frac{1}{Z_A} \mathbb{1}_A(x)\, \lambda(dx)$, $\lambda \hookrightarrow N(0,1)$

i) $Y|X=x \sim N(\sqrt{1-\varepsilon}\, x, \varepsilon) \Rightarrow$

$$\lambda(dx)\, M(x,dy) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \cdot \frac{1}{\sqrt{2\pi\varepsilon}} e^{-\frac{(y-\sqrt{1-\varepsilon}\,x)^2}{2\varepsilon}} dy =$$

$$= \frac{1}{2\pi\sqrt{\varepsilon}} e^{-\frac{x^2}{2\varepsilon} - \frac{y^2}{2\varepsilon} + \frac{\sqrt{1-\varepsilon}}{\varepsilon} xy}\, dx\, dy$$

$$\lambda(dy)\, M(y,dx) = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy \cdot \frac{1}{\sqrt{2\pi\varepsilon}} e^{-\frac{(x-\sqrt{1-\varepsilon}\,y)^2}{2\varepsilon}} dx =$$

$$= \frac{1}{2\pi\sqrt{\varepsilon}} e^{-\frac{x^2}{2\varepsilon} - \frac{y^2}{2\varepsilon} + \frac{\sqrt{1-\varepsilon}}{\varepsilon} xy}\, dx\, dy = \lambda(dx)\, M(x, dy), \quad \text{QED.}$$

ii) $\alpha(x,y) = 1 \wedge \left(\frac{\pi(dy)}{\pi(dx)} \cdot \frac{M(y,dx)}{M(x,dy)} \right) = 1 \wedge \left(\frac{\mathbb{1}_A(y)\, \lambda(dy)\, M(y,dx)}{\mathbb{1}_A(x)\, \lambda(dx)\, M(x,dy)} \right) = \mathbb{1}_A(y)$

($\mathbb{1}_A(x) = 1$)

It means that we accept the transition from x to y, if and only if $y \in A$. Pseudocode:

```
x ← GetInitialState()   (x ∈ A)
for i in range(Nsteps):        ← tuning parameter
    y ← √(1-ε)·x + √ε·W        (W - sample from N(0,1))
    if y ∈ A:                  ← tuning parameter
        x ← y
```

If ε is too small, then we won't explore the space ($\mathbb{R}$).

If ε is too big, then we'll reject almost every transition.

N3

$$\begin{cases} X_n = a_n(X_{n-1}, W_n) \in \mathbb{R} & X_0 \in \mathcal{N}(0,1) \\ Y_n = h_n(X_n) + V_n \in \mathbb{R} & W_n, V_n \sim \mathcal{N}(0,1) \end{cases}$$

1) Assume that $p(x_k | y_0, .., y_{k-1}) \overset{dx_k}{\approx} \frac{1}{N} \sum_{i=1}^{N} \delta_{\xi_k^i}(dx_k)$. Then

$$p(x_k | y_0, .., y_k) dx_k = \frac{p(x_k | y_0 .. y_{k-1}) \cdot p(y_k | x_k) dx_k}{\int_{x_k'} p(x_k' | y_0 .. y_{k-1}) p(y_k | x_k') dx_k'} \quad \left(\approx \left(p(x_k | y_0 .. y_{k-1}) \overset{dx_k}{\approx} \frac{1}{N} \sum \delta \right) \right)$$

$$\approx \sum_{i=1}^{N} \frac{p(y_k | \xi_k^i)}{\sum_{j=1}^{N} p(y_k | \xi_k^j)} \delta_{\xi_k^i}(dx_k) \quad (*)$$ It means that we can do the following:

1) Sample $(\xi_0^i)_{1 \le i \le N} \sim X_0 \sim \mathcal{N}(0,1)$

2) for k in range(n): (selection)

$(\tilde{\xi}_k^i)_{1 \le i \le N} \sim$ Resample() (according to $\pi_k(dx_k) = \sum_{i=1}^{N} \frac{p(y_k | \xi_k^i)}{\sum p} \delta_{\xi_k^i}(dx_k)$)

for i from 1 to N: (exploration)

$\xi_{k+1}^i = a_n(\tilde{\xi}_k^i, W_n)$

Because of (*), at the end we get $(\tilde{\xi}_n^i) \sim p(x_n | y_0, y_1, .., y_n)$ (approx), QED.

ii) We need to apply the same algorithm, but, instead of (i), we need to keep track of the whole path. So, in exploration step, we do $\xi_{k+1}^i = (\tilde{\xi}_k^i, a_n(\tilde{\xi}_k^i, W_n))$ instead of $\xi_{k+1}^i = a_n(\tilde{\xi}_k^i, W_n)$, and in selection step we calculate $p(y_k | \xi_k^i[-1])$ instead of $p(y_k | \xi_k^i)$. ($\xi_k^i[-1]$ ← terminal point of $\xi_k^i \in \mathbb{R}^k$). We'll end up with $(\xi_n^i) \sim p(x_0, .., x_n | y_0 .. y_n)$ (by the same argument as (*))

iii) As we can see, the filtering problem is a particular case of FK-model, with $G_n(x) = p(y_n | x) = g(y_n - h_n(x))$, where $g(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$.

observation — independent, if x is fixed

So, $\gamma_n(1) = \mathbb{E} \prod_{k=0}^{n-1} G_k(X_k) = \mathbb{E}\, p(y_0, .., y_{n-1} | x_0, .. x_{n-1}) = p(y_0, .., y_{n-1}) =$

$= \frac{\gamma_{n-1}(G_{n-1}(X_{n-1}))}{\gamma_{n-1}(1)} \cdot \gamma_{n-1}(1) = \prod_{k=0}^{n-1} \eta_k(G_k(X_k))$. We have

$\eta_k(G_k(x_k)) = \frac{\int G_k(x_k) \prod_{0 \le s < k} G_s(x_s) \, p(x_0 .. x_k) dx_0 .. x_k}{\int \prod_{0 \le s < k} G_s(x_s) p(x_0 .. x_k) dx_0 .. x_k}$

$\approx \frac{1}{N} \sum_{i=1}^{N} \delta_{\xi_k^i}$

$\approx \frac{1}{N} \sum_{i=1}^{N} p(y_k | \xi_k^i) =: \eta_k^N(G_k(X_k))$,

So we need to apply an algorithm from (i), at the each step accumulating calculating η_k^N at each step. Their product is an estimate of $p(y_0 .. y_n)$

$(X_n)_{n\in\mathbb{N}}$ – a simple random walk, $A = [-L, L]$.

i) This is a FK model with $G_n(x) = \mathbb{1}_A(x)$. In order to obtain $X_0, \dots, X_n \mid X_p \in A,\ 0 \le p \le n$, we can apply the particle algorithm for n-th marginal, considering $X_n' = (X_0, \dots, X_n)$ as a Markov chain. The algorithm is defined as follows:

1) $(\xi_0^i)_{1\le i\le N} \sim$ sampled iid copies of X_0 (just 0 in our case)

2) for k in range(n): — selection

$(\tilde{\xi}_k^i)_{1\le i\le N} \sim$ Resample() (according to $\sum_{i=1}^{N} \frac{\mathbb{1}(\xi_k^i[-1] \in A)}{\sum_j \mathbb{1}(\xi_k^j[-1]\in A)} \delta_{\xi_k^i}$)

$\mathbb{1}(-L \le \xi_k^i[-1] \le L)$ ⇒ terminal point of ξ_k^i

for i from 1 to N: — exploration

$\xi_{k+1}^i = (\tilde{\xi}_k^i, x_k^i)$, where $x_k^i = \begin{cases} \tilde{\xi}_k^i[-1] + 1, & \text{with probability } \frac{1}{2} \\ \tilde{\xi}_k^i[-1] - 1, & \text{otherwise} \end{cases}$

At the end we get $(\xi_n^i) \sim p(x_0, \dots, x_n \mid x_p \in A,\ 0 \le p \le n)$

ii) Again (just like problem 3),

$$\gamma_n(1) = \frac{\gamma_{n-1}(G_{n-1}(X_{n-1}))}{\gamma_{n-1}(1)} \cdot \gamma_{n-1}(1) = \eta_{n-1}(G_{n-1}(X_{n-1})) \cdot \gamma_{n-1}(1) = \mathbb{P}(X_0 \in A, \dots, X_{n-1} \in A) = \prod_{k=0}^{n-1} \eta_k(G_k(X_k)).$$

$$\eta_k(G_k(X_k)) = \sum_{0\le s\le k} p(\dots) \approx \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(-L \le \xi_k^i[-1] \le L) = \eta_k^N$$

So we need to apply an algorithm from (i), maintaining a cumulative product of η_k^N, which we easily can get from selection step.