Sequential Monte Carlo tings

Shannon Williams

May 20, 2021

1 State-space models

A *state-space model* is a time series model consisting of two discrete time processes $\{X_t\} := (X_t)_{t\geq 0}$ and $\{Y_t\} := (Y_t)_{t\geq 0}$ taking values in $\mathcal{X}$ and $\mathcal{Y}$, respectively. The model is specified via a parameter vector $\theta \in \Theta$ and a set of densities defining the joint density of the processes:

$$p_0^\theta(x_0) \prod_{t=0}^{T} f_t^\theta \left(y_t | \, x_t \right) \prod_{t=1}^{T} p_t^\theta \left(x_t | \, x_{t-1} \right).$$

This describes a generative probabilistic model: X_0 is drawn according to the initial density $p_0^\theta(x_0)$, each X_t is then drawn conditionally on the $X_{t-1} = x_{t=1}$ according to $p_t^\theta \left(x_t | \, x_{t-1} \right)$, and then each Y_t is drawn conditionally on $X_t = x_t$ according to f_t^θ.

An informal definition of a state-space model is as a Markov chain observed with noise.

1.1 Sequential analysis

Process $\{Y_t\}$ is observed while $\{X_t\}$ is not, and the objective is to derive the distribution of certain X_t's conditional on certain components of $\{Y_t\}$'s. Traditionally this is done without taking into account the parameter uncertainty (θ is assumed to be known and fixed).

Sequential analysis refers to drawing inference on the process X_t sequentially in time at each t, given realised observations $y_{0:t}$ of $Y_{0:t}$ collected up to t. *Filtering* refers to the task of deriving the distribution of X_t conditional on $Y_{0:t} = y_{0:t}$, $t = 0, 1, \ldots$.

1.2 Stochastic volatility models in finance

We are interested in modelling the volatility of log-returns,

$$Y_t = \log \left(\frac{p_t}{p_{t-1}} \right),$$

where p_t is the price of an asset at time t. A standard approach is ARCH (auto-regressive conditional heteroscedastic) or GARCH (generalised ARCH) modelling, where the variance of Y_t is a deterministic function of past data $Y_{0:t-1}$. A different approach takes the volatility to be a stochastic process.

A basic *stochastic volatility model* is

$$Y_t | X_t = x_t \sim \mathcal{N} \left(0, \exp(x_t) \right),$$

where $\{X_t\}$ is an auto-regressive process (define AR processes here?):

$$X_t - \mu = \rho(X_{t-1} - \mu) + U_t,$$

where $U_t \sim \mathcal{N}\left(0, \sigma^2\right)$ and $\theta := \left(\mu, \rho, \sigma^2\right)$. Imposing $|\rho| < 1$ ensures that $\{X_t\}$ is a stationary processes. In practice we might expect $\rho \approxeq 1$ as financial data often exhibit volatility clustering (volatility remains high or low for long periods of time).

Variants of the stochastic volatility model:

- $Y_t|X_t$ may have heavier-than-Gaussian tails, e.g. follow a student distribution.
- Account for a leverage effect by assuming the Guassian noises of $Y_t|X_t$ and $X_t|X_{t-1}$ are correlated.
- Introduce skewness by taking $Y_t = \alpha X_t + \exp(X_t/2)V_t$.

The volatility might be assumed to be an AR process of order $k > 1$. In which case, we can retain the basic structure of a state-space model by increasing the dimension of $\{X_t\}$. If $\mathcal{X} = \mathbb{R}^2$ then

$$X_t - \begin{pmatrix}\mu\\ \mu\end{pmatrix} = \begin{pmatrix}\rho_1 & \rho_2\\ 1 & 0\end{pmatrix}\left(X_{t-1} - \begin{pmatrix}\mu\\ \mu\end{pmatrix}) + \begin{pmatrix}U_t\\ 0\end{pmatrix}\right),$$

where $U_t \sim \mathcal{N}\left(0, \sigma^2\right)$. Then $X_t(1)$ is an AR process of order 2,

$$X_t(1) - \mu = \rho_1(X_{t-1}(1) - \mu) + \rho_2(X_{t-2}(1) - \mu) + U_t,$$

and we can take $\mathrm{Var}(Y_t|X_t) = \exp\left(X_t(1)\right)$.

2 Feynman-Kac models

Start with a *Markov probability law* defined on a state space $\mathcal{X}$ with initial distribution $\mathbb{M}_)$ and transition kernels $M_{1:T}$:

$$\mathbb{M}_T(dx_{0:T}) = \mathbb{M}_0(dx_0)\prod_{t=1}^{T} M_t(x_{t-1}, dx_t).$$

Consider a sequence of *potential functions* $G_0 : \mathcal{X} \to \mathbb{R}^+$ and $G_t : \mathcal{X}^2 \to \mathbb{R}^+$, $t = 1, \ldots, T$. For $t = 0, \ldots, T$, a sequence of Feynman-Kac models is given by probability measures on $\left(\mathcal{X}^{t+1}, \mathcal{B}(\mathcal{X})^{t+1}\right)$, obtained as the following changes of measure from $\mathbb{M}_t$:

$$\mathbb{Q}_t(dx_{0:t}) := \frac{1}{L_t} G_0(x_0)\left\{\prod_{s=1}^{t} G_s(x_{s-1}, x_s)\right\}\mathbb{M}_t(dx_{0:t}),$$

where L_t is the normalising constant needed for $\mathbb{Q}_t$ to be a probability measure.

We refer to T, $\mathbb{M}_0$, G_0 and $M_t(x_{t-1}, dx_t)$, $G_t(x_{t-1}, x_t)$, $t = 1, \ldots, T$ as the components of the Feynman-Kac model.

2.1 Feynman-Kac formalisms of a state-space model

Consider a state-space model with initial distribution $\mathbb{P}_0(dx_0)$, signal transition kernels $P_t(x_{t-1}, dx_t)$, and observation densities $f_t(y_t|x_t)$. We define its *bootstrap Feynman-Kac formalism* to be the Feynman-Kac model with the components:

$$\begin{aligned} \mathbb{M}_0(dx_0) &= \mathbb{P}_0(dx_0), & G_0(x_0) &= f_0(y_0|x_0), \quad (1)\\ M_t(x_{t-1}, dx_t) &= P_t(x_{t-1}, dx_t), & G_t(x_{t-1}, x_t) &= f_t(y_t|x_t). \quad (2) \end{aligned}$$

Then

$$\begin{aligned} \mathbb{Q}_{t-1}(dx_{0:t}) &= \mathbb{P}_t\left(X_{0:t} \in dx_{0:t} \left| Y_{0:t-1} = y_{0:t-1}\right.\right), \quad (3)\\ \mathbb{Q}_t(dx_{0:t}) &= \mathbb{P}_t\left(X_{0:t} \in dx_{0:t} \left| Y_{0:t} = y_{0:t}\right.\right), \quad (4)\\ L_t &= p_t(Y_{0:t}), \quad (5)\\ l_t &:= \frac{L_t}{L_{t-1}} = p_t\left(y_t \left| y_{0:t-1}\right.\right). \quad (6) \end{aligned}$$

The potential functions G_t for $t \geq 1$ depend only on x_t, and G_t depends implicitly on the (fixed) datapoint y_t.

3 Resampling

Resampling: the action of drawing randomly from a weighted sample, so as to obtain an unweighted sample. We can view resampling as a random weight importance sampling technique.

Suppose we have the following particle approximation of measure $\mathbb{Q}_0(dx_0)$:

$$\mathbb{Q}_0^N(dx_0) = \sum_{n=1}^N W_0^N \delta_{X_0^n}, \qquad , X_0^n \sim \mathbb{M}_0, \qquad W_0^n = \frac{w_0(X_0^n)}{\sum_{m=1}^N w_0(X_0^m)},$$

obtained through importance sampling based on the proposal $\mathbb{M}_0$ and weight function $w_0 \propto d\mathbb{Q}_0/d\mathbb{M}_0$. We want to use this to approximate the extended probability measure

$$(\mathbb{Q}_0 M_1)(dx_{0:1}) = \mathbb{Q}_0(dx_0) M_1(x_0, dx_1). \quad (7)$$

Importance resampling uses a two-step approximation: first replace $\mathbb{Q}_0$ by $\mathbb{Q}_0^N$ in (7),

$$\mathbb{Q}_0^N(dx_0) M_1(x_0, dx_1) = \sum_{n=1}^N W_0^n M_1(X_0^n, dx_1) \delta_{X_0^n}(dx_0), \quad (8)$$

and then sample N times from this intermediate approximation to form

$$\frac{1}{N}\sum_{n=1}^N \delta_{\tilde{X}_{0:1}^n}, \qquad \tilde{X}_{0:1}^n \sim \mathbb{Q}_0^N(dx_0) M_1(x_0, dx_1).$$

The simplest way to sample from (8): for $n = 1, \dots, N$, sample independently the pairs $(A_1^n, \tilde{X}_{0:1}^n)$ as

$$A_1^n \sim \mathcal{M}(W_0^{1:N}), \qquad \tilde{X}_{0:1}^N = \left(X_0^{A_1^n}, X_1^n\right), \qquad X_1^n \sim M_1\left(X_0^{A_1^n}, dx_1\right),$$

where $\mathcal{M}(W_0^{1:N})$ is the multinomial distribution that generates value n with probability W_0^n, for $n = 1, \dots, N$.

3.1 Multinomial resampling

Multinomial resampling describes an efficient algorithm for simulating the *ancestor* variables A^n from the multinomial distribution $\mathcal{M}(W^{1:N})$.

Use the inverse CDF transformation method: generate N uniform variates U^m, $m = 1, \dots, N$, and set A^m according to

$$C^{n-1} \leq U^m \leq C^n \quad \Longleftrightarrow \quad A^m = n,$$

where the C^n's are the cumulative weights:

$$C^0 = 0, \quad C^n = \sum_{l=1}^{n} W^l, \quad n = 1, \dots, N.$$

Each individual simulation requires $\mathcal{O}(N)$ comparisons to be made, so $\mathcal{O}\left(N^2\right)$ comparisons should be performed to generate N draws. However, if the U^m are ordered in a preliminary step we obtain an algorithm with $\mathcal{O}(N)$ complexity.

Algorithm 1: Computing the inverse of the multinomial CDF $x \to \sum_{n=1}^{N} W^n \mathbb{I}\{n \leq x\}$.

```
Input: Normalised weights $W^{1:N}$ and ordered uniform points $0 < U^{(1)} < \dots < U^{(N)} < 1$.
Output: Ordered ancestor variables $1 \leq A^1 \leq \dots \leq A^N \leq N$ in {1, ..., N}.
Initialise $s \leftarrow W^1$, $m \leftarrow 1$.
for $n = 1, \dots, N$ do
    while $s < U^{(n)}$ do
        $m \leftarrow m + 1$.
        $s \leftarrow s + W^m$.
    end
    $A^n \leftarrow m$.
end
```

Note the A^n are not i.i.d. draws from $\mathcal{M}(W^{1:N})$ but correspond to the order statistics of a vector of N draws from this distribution.

To generate the *uniform spacings* $0 < U^{(1)} < \dots < U^{(N)} < 1$, we use an $\mathcal{O}(N)$ algorithm.

The overall approach for multinomial resampling is then:

Can be interpreted as a random weight importance technique with random weight $\mathcal{W}^n$ such that

- $\mathcal{W}^n$ is integer-valued and represents the number of offsprings of particle n;
- $\mathbb{E}\left[\mathcal{W}^n\right] = NW^n$;

Algorithm 2: Generation of uniform spacings.

```
Input: An integer $N$
. Output: An ordered sequence $0 < U^{(1)} < \cdots < U^{(N)} < 1$ in $(0, 1)$.
Initisalise $S^0 \leftarrow 0$.
for $n$ = 1, . . . , N+1 do
    Draw $E^n \sim \mathcal{E}(1)$.
    $S^n \leftarrow S^{n-1} + E^n$.
end
for $n$ = 1, . . . , N do
    $U^{(n)} \leftarrow S^n / S^{N+1}$.
end
```

Algorithm 3: Multinomial resampling.

```
Input: Normalised weights $W^{1:N}$ such that $\sum_{n=1}^N W^n = 1$ and $W^n \geq 0$, $n = 1, \ldots, N$.
Output: $N$ draws from $\mathcal{M}(W^{1:n})$.
Obtain $U^{(1:N)}$ via Algorithm 2.
Obtain $A^{1:N}$ via Algorithm 1.
```

- $\sum_{n=1}^N \mathcal{W}^n = N$.

Any method that generates ancestor variables $A^{1:N}$ such that the number of copies $\mathcal{W}^n = \sum_{m=1}^n \mathbb{I}\{A^m = n\}$ fulfilles the above three properites will be an *unbiased resampling scheme*, satisfying

$$\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^N \varphi\left(X^{A^n}\right) \middle| X^{1:N}\right] = \mathbb{E}\left[\frac{1}{N}\sum_{n=1}^N \mathcal{W}^n \varphi(X^n) \middle| X^{1:N}\right] = \sum_{n=1}^N W^n \varphi(X^n).$$

That is, it generates an unweighted sample that provides estimates with the same expectation (but increased variance) as the original weighted sample.

3.2 Systematic resampling

A strategy for variance reduction is to replace the i.i.d. uniforms by values that cover $[0, 1]$ more regularly. A *systematic resampling approach* is similar to Algorithm 1, specialised to the structure of the $U^{(n)}$'s. We simulate a single uniform random variable $U \sim \mathcal{U}([0, 1])$ and take $U^{(n)} = (n - 1 + U)/N$.

Algorithm 4: Systematic resampling approach for obtaining ancestor variables $A^{1:N}$.

```
Input: Normalised weights $W^{1:N}$ and $U \in [0, 1]$.
Output: $N$ random indices $A^{1:N}$ taking values in $1 : N$.
Initialise $s \leftarrow U$ and $m \leftarrow 1$.
Compute the cumulative weights $v^n := \sum_{m=1}^n NW^m$, $n = 1, \ldots, N$.
for $n = 1, \ldots, N$ do
    while $v^m < s$ do
        Update $m \leftarrow m + 1$, $A^n \leftarrow m$, $s \leftarrow s + 1$.
    end
end
```

This is unbiased: $\mathbb{E}[\mathcal{W}^n] = NW^n$.

Compared to other variance reduction schemes (not discussed here), systematic resampling is often recommended in practice as it is fast and tends to (empirically) work better than other schemes in that it yields lower-variance estimates.

4 The bootstrap filter for state-space models

Consider a state space model $\{(X_t, Y_t)\}$ with initial law $\mathbb{P}_0$, transition kenels $P_t(x_{t-1}, dx_t)$ and observation densities $f_t\left(y_t \middle| x_t\right)$. Consider the basic Feynman-Kac representation of the model (1), (2), for $t \geq 1$.

Algorithm 5: The bootstrap filter for state-space models with adaptive resampling (resampling only when the ESS is too low – reduces computational time of the algorithm).

```
Input: A state-space model such that we can simulate from $\mathbb{P}_0(dx_0)$ and from $P_t(x_{t-1}, dx_t)$
       for each $x_{t-1}$ and $t$, and compute the function $f_t\left(y_t \middle| x_t\right)$ point-wise, a number of
       particles $N$, a choice of a resampling scheme, and an ESS threshold $\mathrm{ESS}_{\min}$.
Output: $\hat{w}_0^n, \ldots, \hat{w}_{T-1}^n$, $X_1^n, \ldots, X_T^n$, $w_0^n, \ldots, w_T^n$, $W_0^n, \ldots, W_T^n$, for $n = 1, \ldots, N$.
All operations involving index $n$ must be performed for $n = 1, \ldots, N$.
Simulate $X_0^n \sim \mathbb{M}_0(dx_0)$.
$w_0^n \leftarrow G_0(X_0^n)$.
$W_0^n \leftarrow w_0^n / \sum_{m=1}^M w_0^m$.
for $t = 1, \ldots, T$ do
    if $\mathrm{ESS}(W_{t-1}^{1:N}) < \mathrm{ESS}_{\min}$ then
        Resample $A_t^{1:N}$ with weights $W_{t-1}^{1:N}$.
        $\hat{w}_{t-1}^n \leftarrow 1$.
    end
    else
        $A_n^t \leftarrow n$.
        $\hat{w}_{t-1}^n \leftarrow w_{t-1}^n$.
    end
    Draw $X_t^n \sim P_t\left(X_{t-1}^{A_t^n} \middle| dx_t\right)$.
    $w_t^n \leftarrow \hat{w}_{t-1}^n f_t\left(y_t \middle| X_t^n\right)$.
    $W_t^n \leftarrow w_t^n / \sum_{m=1}^N w_t^m$.
end
```

At each time $t = 1, \ldots, T$, we generate simulations from the law of the Markov chain $\{X_t\}_{t \geq 1}$, weight these simulations according to how "compatible" they are with the datapoint Y_t, and resample if necessary. We obtain the approximations:

$$\frac{1}{\sum_{n=1}^N \hat{w}_{t-1}^n} \sum_{n=1}^N \hat{w}_{t-1}^n \varphi(X_t^n) \approx \mathbb{E}\left[\varphi(X_t) \,|Y_{0:t-1} = y_{0:t-1}\right]$$

$$\sum_{n=1}^N W_t^n \varphi(X_t^n) \approx \mathbb{E}\left[\varphi(X_t) \,|Y_{0:t} = y_{0:t}\right]$$

$$\ell_t^N \approx p_t\left(y_t \middle| y_{0:t-1}\right)$$

$$L_t^N = \prod_{s=1}^t \ell_s^N \approx p(y_{0:t})$$

where

$$\ell_t^N = \begin{cases} \frac{1}{N}\sum_{n=1}^N w_t^n & \text{if resampling occurred at time } t, \\ \frac{\sum_{n=1}^N w_t^n}{\sum_{n=1}^N w_{t-1}^n} & \text{otherwise} \end{cases}$$

Note $\approx$ means the LHS approximates the RHS: that is, the approximation error tends to zero as $N \to \infty$ with rate $\mathcal{O}\left(N^{-\frac{1}{2}}\right)$ (the standard Monte Carlo rate).

The bootstrap filter is very simple and widely applicable with few requirements. However, if samples particles X_t "blindly" from $P_t(x_{t-1}, dx_t)$ without any guarantee that these simulated particles will be compatible with the datapoint y_t (i.e. have non-negligible weights).

5 Bayesian estimation of state-space models and particle MCMC

We consider the problem of estiamting the parameter θ of a state-space model. We view θ as the realisation of the random variable Θ with prior distribution $\nu(d\theta)$ and wish to compute the (typically intractable) posterior distribution, i.e. the distribution of Θ conditional on observed data $y_{0:T}$.

5.1 Pseudo-marginal samplers

Consider a Bayesian model with prior distribution $\nu(d\theta)$ and likelihood function $p^\theta(y)$. Furthermore, assume the likelihood is intractable – this prevents us from deriving a MH sampler that would leave invariant the posterior distribution

$$\mathbb{P}\left(d\theta \middle| Y = y\right) = \nu(d\theta)\frac{p^\theta(y)}{p(y)}.$$

A *pseudo-marginal MCMC* sampler can be described (informally) as a Metropolis-Hastings sampler where the intractable likelihood is replaced by an unbiased estimate. That is, we assume we know how to sample an auxiliary variable $Z \sim \mathbb{M}^\theta(dz)$ and how to compute a non-negative function $L(\theta, z)$ such That

$$\mathbb{E}\left[L(\theta, Z)\right] = \int \mathbb{M}^\theta(dz) L(\theta, z) = p^\theta(y),$$

for any $\theta \in \Theta$. Then, given the current pair (Θ, Z) we generate a new pair $(\tilde{\Theta}, \tilde{Z})$ and use the ratio

$$r_{\text{PM}}(\theta, z, \tilde{\theta}, \tilde{z}) = \frac{\nu(\tilde{\theta})L(\tilde{\theta}, \tilde{z})\tilde{m}\left(\theta \middle| \tilde{\theta}\right)}{\nu(\theta)L(\theta, z)\tilde{m}\left(\tilde{\theta} \middle| \theta\right)}$$

to decide on the acceptance of the proposed values.

5.2 PMMH: Particle marginal Metropolis-Hastings

We have a state-space model parameterised by $\theta \in \Theta$ and we want to sample from its posterior, given data $(y_{0:T})$. We can run a pseudo-marginal sampler, where the intractable likelihood $p_T^\theta(y_{0:t})$ of the model is replaced by an unbiased estimate obtained from a particle filter.

Associate our parametric state-space model with a parametric FK model, with kernels $M_t^\theta(x_{t-1}, dx_t)$ and functions G_t^θ that depend on θ. The FK model must be such that the FK measure $\mathbb{Q}_T^\theta(dx_{0:t})$ matches the posterior distribution of the states, conditional on $\Theta = \theta$:

$$\begin{aligned}
\mathbb{Q}_T^\theta(dx_{0:t}) &= \frac{1}{L_T^\Theta}\mathbb{M}_0^\theta(dx_0)\prod_{t=1}^T M_t^\theta(x_{t-1}, dx_t)G_0^\theta(x_0)\prod_{t=1}^T G_t^\theta(x_{t-1}, x_t) \\
&= \frac{1}{p_T^\theta(y_{0:T})}\mathbb{P}_T^\theta(dx_{0:T})\prod_{t=0}^T f_t^\theta\left(y_t | x_t\right) \\
&= \mathbb{P}_T\left(dx_{0:T} | \Theta = \theta, Y_{0:T} = y_{0:T}\right).
\end{aligned}$$

In particular,we reuire that $L_T^\Theta = p^\theta(y_{0:T})$ for all $\theta \in \Theta$.

To specify this parametric FK model (with the bootstrap formalism), $M_t^\theta(x_{t-1}, dx_t) = P_t^\theta(x_{t-1}, dx_t)$ so that

$$G_t^\theta(x_{t-1}, x_t) = f_t^\theta\left(y_t | x_t\right).$$

for $t \geq 1$.

Since $L_T^\theta = p_T^\theta(y_{0:T})$ we may, for any $\theta \in \Theta$, run an SMC algorithm associated with the considered FK model and compute

$$L_T^N\left(\theta, X_{0:T}^{1:N}, A_{1:T}^{1:N}\right) = \left(\frac{1}{N}\sum_{n=1}^N G_0^\theta(X_0^n)\right)\prod_{t=1}^T\left(\frac{1}{N}\sum_{n=1}^N G_t^\theta\left(X_{t-1}^{A_t^n}, X_t^n\right)\right)$$

as an unbiased estimate of $p_T^\theta(y_{0:t})$.

We refer to pseudo-marginal algorithms based on particle filters as particle marginal Metropolis-Hastings (PMMH).

5.3 SMC2 for sequential inference in state-space models

We want to implement the IBIS (iterative Bayesian importance sampling) algorithm where $\gamma_t(\theta) = p_t^\theta(y_{0:t})$ is intractable. We consider a pseudo-marginal version of the algorithm, where p_t^θ is estimated unbiasedly by a particle filter run until time t. The algorithm carries forward N_θ parameter values $\Theta_t^1, \ldots, \Theta_t^{N_\theta}$ and N_θ associated particle filters, which provide unbiased estimates of $p_t^\theta(y_{0:t})$ for $\theta = \Theta_t^m$, $m = 1, \ldots, N_\theta$.

The particle filters are guided filters of size N_x which generate up to time t variables $Z_t^m := \left(X_{0:t}^{m,1:N_x}, A_{1:t}^{m,1:N_x}\right)$ and the corresponding likelihood estimate of the form

$$L_t^{N_x}\left(\Theta_t^m, X_{0:t}^{m,1:N_x}, A_{1:t}^{m,1:N_x}\right) = \left(\frac{1}{N}\sum_{N=1}^{N_x} G_0^{\Theta_t^m}\left(X_0^{m,n}\right)\right)\prod_{s=1}^t\left(\frac{1}{N}\sum_{n=1}^{N_x} G_s^{\Theta_t^m}\left(X_{t-1}^{m,A_t^n}, X_t^{m,n}\right)\right).$$

PMCMC algorithms leave invariant the extended distribution

$$\pi_t\left(d\theta, dx_{0:t}^{1:N_x}, a_{1:t}^{1:N_x}\right) = \frac{1}{p_t(y_{0:t})}\nu(d\theta)\psi_t^\theta\left(dx_{0:t}^{1:N_x}, a_{1:t}^{1:N_x}\right)L_t^{N_x}\left(\theta, x_{0:t}^{1:N_x}, a_{1:t}^{1:N_x}\right) \tag{9}$$

Algorithm 6: One-step PMMH.

```
Input: $\Theta$, $X_{0:T}^{1:N}, A_{1:T}^{1:N}$
Output:
Draw $\tilde{\Theta} \sim \tilde{M}(\Theta, d\theta)$. ;
Run SMC algorithm to generate variables $\left(\tilde{X}_{0:T}^{1:N}, \tilde{A}_{1:T}^{1:N}\right)$ given $\tilde{\Theta}$. ;
Draw $U \sim \mathcal{U}([0,1])$. ;
$v \leftarrow \log r_{\text{PMMH}}\left(\Theta, X_{0:T}^{1:N}, A_{1:T}^{1:N}, \tilde{\Theta}, \tilde{X}_{0:T}^{1:N}, \tilde{A}_{1:T}^{1:N}\right)$ where
    $$r_{\text{PMMH}}\left(\Theta, X_{0:T}^{1:N}, A_{1:T}^{1:N}, \tilde{\Theta}, \tilde{X}_{0:T}^{1:N}, \tilde{A}_{1:T}^{1:N}\right) := \frac{\nu(\tilde{\theta}) L_T^N\left(\tilde{\theta}, \tilde{x}_{0:T}^{1:N}, \tilde{a}_{1:T}^{1:N}\right) \tilde{m}\left(\theta \middle| \tilde{\theta}\right)}{\nu(\theta) L_T^N\left(\theta, x_{0:t}^{1:N}, a_{1:T}^{1:N}\right) \tilde{m}\left(\tilde{\theta} \middle| \theta\right)}.$$
if $\log U \leq v$ then
    return $\tilde{\Theta}, \tilde{X}_{0:T}^{1:N}, \tilde{A}_{1:T}^{1:N}$. ;
end
else
    return $\Theta, X_{0:T}^{1:N}, A_{1:T}^{1:N}$. ;
end
```

where $\psi_t^\theta(\cdot)$ is the distribution of the random variables generated by a particle filter associated with parameter θ. This distribution admits the true posterior $\mathbb{P}_t\left(d\theta \middle| y_{0:t}\right)$ as a marginal.

We can define the SMC^2 algorithm as an SMC sampler that targets the sequence of π_t's. Note successive distributions π_t do not have the same support: to perform an importance sampling step from π_{t-1} to π_t we first extend the space by simulating $\left(X_t^{1:N}, A_t^{1:N}\right)$. Let $\psi_t^\theta\left(dx_t^{1:N_x}, a_t^{1:N_x} \middle| x_{0:t-1}^{1:N_x}, a_{1:t-1}^{1:N_x}\right)$ denote the distribution of the variables generated at time t, conditional on the previous steps:

$$\psi_t^\theta\left(dx_t^{1:N_x}, a_t^{1:N_x} \middle| x_{0:t-1}^{1:N_x}, a_{1:t-1}^{1:N_x}\right) = \prod_{n=1}^{N_x} W_{t-1}^{a_t^n} M_t^\theta\left(x_{t-1}^{a_t^n}, dx_t^n\right),$$

where

$$W_{t-1}^n = \frac{G_{t-1}^\theta\left(x_{t-2}^{a_{t-1}^n}, x_{t-1}^n\right)}{\sum_{m=1}^M G_{t-1}^\theta\left(x_{t-2}^{a_{t-1}^m}, x_{t-1}^m\right)}.$$

Then define the importance sampling weight function:

$$\frac{\pi_t\left(d\theta, dx_{0:t}^{1:N_x}, a_{1:t}^{1:N_x}\right)}{\pi_{t-1}\left(d\theta, dx_{0:t-1}^{1:N_x}, a_{1:t-1}^{1:N_x}\right) \psi_t^\theta\left(dx_t^{1:N_x}, a_t^{1:N_x} \middle| x_{0:t-1}^{1:N_x}, a_{1:t-1}^{1:N_x}\right)} \propto \frac{1}{N} \sum_{n=1}^{N_x} G_t^\theta\left(x_{t-1}^{a_t^n}, x_t^n\right), \tag{10}$$

where the normalising constant is $p_t\left(y_t \middle| y_{0:t-1}\right)$.

Defining the SMC^2 sampler:

- Initialise the algorithm by sampling N_θ particles from the prior $\nu(d\theta)$. For each particle Θ_0^m, perform iteration 0 of a particle filter to generate variables $X_0^{m,1:N_x}$ and obtain an unbiased estimate of $p_0^\theta(y_0)$ for $\theta = \Theta_0^1, \ldots, \Theta_0^{N_\theta}$.
- At time $t \geq 1$, perform iteration t of the N_θ particle filters and reweight the particles according to (10).
- If the ESS is too low, resample the particles and move them according to a kenerl K_t that leaves invariant the current target distribution (this must be a PMCMC kernel such as PMMH or Particle Gibbs).

Algorithm 7: SMC^2 algorithm.

```
Input:
  • Number of particles $N_x$ and $N_\theta$.
  • A parametric family of state-space mdoels with parameter $\theta \in \Theta$.
  • A prior distribution $\nu(d\theta)$ for $\Theta$.
  • A ($\theta$-indexed) class of particle filters which, for a given $\theta$ and when run until time $t$,
    provide an unbiased estimate of the likelihood $p_t\theta(y_{0:t})$.
  • A sequence of PMCMC kernels $K_t$ that leaves invariant distribution (9).
  • A choice of resampling scheme and a threshold $\mathrm{ESS}_{\min}$.
Output:
All operations referring to $m$ must be performed for $m = 1, \ldots, N_\theta$.
Draw $\Theta_0^m \sim \nu(d\theta)$.
Draw $X_0^{m,1:N_x} \sim \psi_0^{\Theta_0^m}\left(dx_0^{1:N_x}\right)$. $w_0^m \leftarrow \sum_{n=1}^{N_x} G_0^{\Theta_0^m}\left(X_0^{m,n}\right)/N$ . $W_0^m \leftarrow w_0^m / \sum_{m=1}^{N_\theta} w_0^m$.
for $t = 1, \ldots, T$ do
    if ESS $\left(W_{t-1}^{1:N_\theta}\right) < \mathrm{ESS}_{\min}$ then
        Move the particles through PMCMC kernel $K_t$:
        $\left(\Theta_t^m, \bar{X}_{0:t-1}^{m,1:N_x}, \bar{A}_{1:t-1}^{1:N_x}\right) \sim K_t\left(\left(\Theta_{t-1}^m, X_{0:t-1}^{m,1:N_x}, A_{1:t-1}^{m,1:N_x}\right), d\left(\theta, x_{0:t-1}^{1:N_x}, a_{1:t-1}^{1:N_x}\right)\right)$.
        $\left(X_{0:t-1}^{1:N_x}, A_{1:t-1}^{1:N_x}\right) \leftarrow \left(\bar{X}_{0:t-1}^{1:N_x}, \bar{A}_{1:t-1}^{1:N_x}\right)$.
        $w_{t-1}^m \leftarrow 1$.
    end
    else
        $\Theta_t^m \leftarrow \Theta_{t-1}^m$.
    end
    $\left(X_t^{m,1:N_x}, A_t^{m,1:N}\right) \sim \psi_t^{\Theta_t^m}\left(dx_t^{1:N_x}, a_t^{1:N_x} \middle| x_{0:t-1}^{1:N_x}, a_{1:t-1}^{1:N_x}\right)$.
    $w_t \leftarrow w_{t-1}^m \sum_{n=1}^{N_x} G_t^{\Theta_t^m}\left(X_{t-1}^{m,A_t^{m,n}}, X_t^{m,n}\right)/N$.
    $W_t^m \leftarrow w_t^m / \sum_{l=1}^{N_\theta} w_t^l$.
end
```

Choosing a PMMH kernel means there is no need to store the complete history of the N_θ particle filters: it is sufficient to store only the variables generated at the previous iteration $\left(X_t^{m,1:N_x}, A_t^{m,1:N_x}\right)$ and the current likelihood estimates, reducing the memory cost to $\mathcal{O}\left(N\theta N_x\right)$.

Increasing N_x increases the CPU cost but may also improve performance. Generally, take $N_x = \mathcal{O}\left(t\right)$ to ensure that, if implemented at time t, the chosen PMMH kernel mixes sufficiently well. A basic recipe (for when to increase N_x over the course of an SMC^2 algorithm) is to monitor the acceptance rate of the PMMH kernels. When the acceptance rate is too low, say below 10%, double N_x.

Let N_x denote the current number of x-particles, and $\tilde{N}_x$ the value we wish to change to. Two better options for increasing N_x:

1. For each $m = 1, \dots, N_\theta$, generate a new particle filter of size $\tilde{N}_x$. Then exchange "old" particle filters (of size N_x) with the new ones via an importance sampling step, with weights equal to the ratio of the new likelihood estimates and the old components:

$$\frac{L_t^{\tilde{N}_x}\left(\theta, \tilde{x}_{0:t}^{1:\tilde{N}_x}\right), \tilde{a}_{1:t}^{1:\tilde{N}_x}}{L_t^{N_x}\left(\theta, x_{0:t}^{1:N_x}, a_{1:t}^{1:N_x}\right)}.$$

2. Use CSMC kernel (not discussed here).