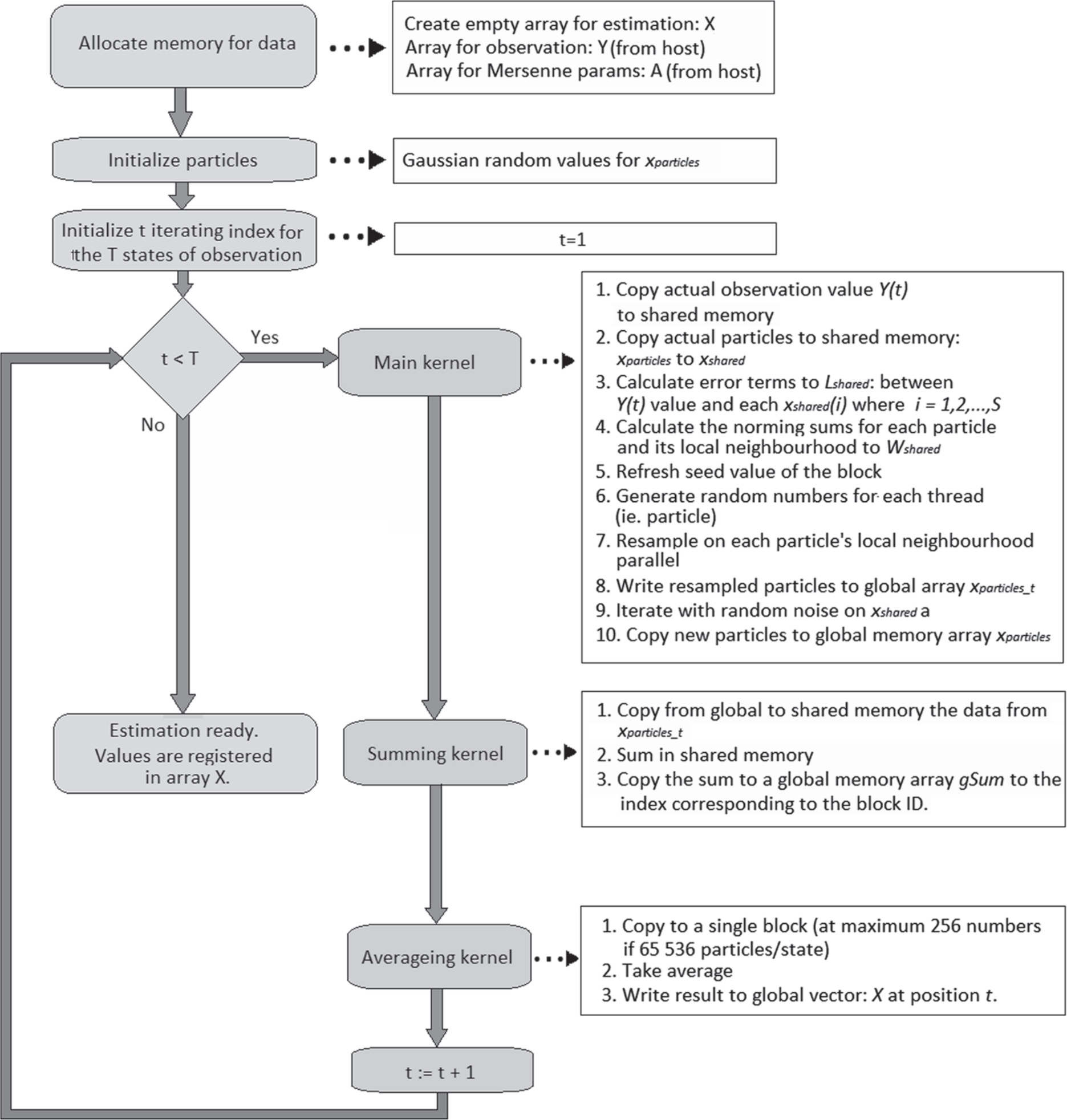
Allocate memory for data
Create empty array for estimation: X
Array for observation: Y (from host)
Array for Mersenne params: A (from host)
Initialize particles
Gaussian random values for $x_{particles}$
Initialize t iterating index for the T states of observation
t=1
t < T
Yes
No
Main kernel
1. Copy actual observation value $Y(t)$ to shared memory
2. Copy actual particles to shared memory: $x_{particles}$ to x_{shared}
3. Calculate error terms to L_{shared}: between $Y(t)$ value and each $x_{shared}(i)$ where $i = 1,2,...,S$
4. Calculate the norming sums for each particle and its local neighbourhood to W_{shared}
5. Refresh seed value of the block
6. Generate random numbers for each thread (ie. particle)
7. Resample on each particle's local neighbourhood parallel
8. Write resampled particles to global array $x_{particles_t}$
9. Iterate with random noise on x_{shared} a
10. Copy new particles to global memory array $x_{particles}$
Estimation ready. Values are registered in array X.
Summing kernel
1. Copy from global to shared memory the data from $x_{particles_t}$
2. Sum in shared memory
3. Copy the sum to a global memory array *gSum* to the index corresponding to the block ID.
Averageing kernel
1. Copy to a single block (at maximum 256 numbers if 65 536 particles/state)
2. Take average
3. Write result to global vector: *X* at position *t*.
t := t + 1