Appendix: Backward algorithm

Equations (1), (3) and (6) once time discretized, can be formally rewritten as

$$\begin{cases} a(k+1) = F\,[a(k), v(k), f(k)] & k = 0, \ldots, T-1 \qquad (A.1) \\ v(k+1) = G\,[v(k), a(k)] & k = 0, \ldots, T-1 \qquad (A.2) \\ p(j-1) = H\,[p(j), a(j), v(j)] & j = T-1, \ldots, 0 \qquad (A.3) \end{cases}$$

where for the sake of simplicity only temporal indices appear, initial conditions are taken as $a(0) = a_0$, $v(0) = v_0$ and $p(T) = p_T$. Let $T = 2^N$. Eqs. (A.1)-(A.3) can be integrated via the following recursive algorithm[1]:

Algorithm: Backward Algorithm

Initialization:

set $a(0) = a_0$, $v(0) = v_0$, $p(2^N - 1) = p_T$;
set $flag(k) = 0$, for $k = 0, \ldots, 2^N - 1$;
set $m = 0$, $n = 2^N - 1$.

Recursion(m, n):

Forward: for $k = m, m+1, \ldots, n$;
compute $a(k+1) = F\,[a(k), v(k), f(k)]$;
compute $v(k+1) = G\,[v(k), a(k)]$;
If $(k - m = 2^L - 2^J$ for each $J = 0, \ldots, L$ with $L = \log_2(n - m + 1))$:
store $v(k), a(k)$;
$flag(k) = 1$.

Backward: for $k = n, n-1, \ldots, m$;
While $flag(k) = 1$:
compute $p(k-1) = H\,[p(k), a(k), v(k)]$;
If $(k = 0)$: **Termination.**
set $l = \{$ largest integer $< k$ such that $flag(i) = 1\}$;
set $m = l$, $n = k$;
Call **Recursion**(m, n).

[1]We acknowledge useful discussions and suggestions with A. Noullez.