Calculating Propagators and Saving Memory

For the continuous-Gaussian-chain model such as that in the "standard" model used in PSCF, a block of length N has two (one-end-integrated) propagators: the forward propagator $q(\mathbf{r},s)$ and the backward propagator $q^\dagger(\mathbf{r},s)$, where $\mathbf{r}$ denotes the spatial position, and $s=0,\ldots,n_s\Delta s$ with n_s being the number of steps along the chain contour into which the block is uniformly discretized and $\Delta s \equiv N/n_s$ being the step-size; in the REPS-K method (K=0,…,4; see REPS.pdf for details), n_s must be an integer multiple of 2^K (*i.e.*, $M \equiv n_s/2^K$ must be a positive integer). PSCF calculates the propagators in two steps: in Step I it solves all the forward propagators, and in Step II it solves all the backward propagators; that is, for each BCP component in the system, the labeling of the two end-segments s=0 and s=N for all its blocks and the order of solving all its propagators are determined automatically in PSCF based on its chain architecture.

To calculate the contribution of the above block to the volume-fraction field of its segment type, we need to evaluate $\int_0^N \mathrm{d}s\, q(\mathbf{r},s)q^\dagger(\mathbf{r},s) \approx \Delta s \sum_{k'=1}^{M}\sum_{k=0}^{2^K} c_k q_{2^K(k'-1)+k} q^\dagger_{2^K(k'-1)+k}$ at all $\mathbf{r}$ using the Romberg integration (denoted by RI-K; see RI.pdf for details), where c_k's are the coefficients used in RI-K, $q_j \equiv q(\mathbf{r}, j\Delta s)$, and $q_j^\dagger \equiv q^\dagger(\mathbf{r}, j\Delta s)$. Usually (as in PSCF) such integrals are calculated after all the propagators are solved and stored. The "slice" algorithm proposed in Ref. 1, however, is based on the fact that once the contribution of segment s on the block to the above integral (and stresses) is calculated, q_j and $q_j^\dagger$ are no longer needed. In particular, q_j for each block are stored only at the "**check points**", where j=0 or $j = n_s - \left(\sum_{k=1}^{l} k - 1\right) = n_s - \frac{l(l+1)}{2} + 1$ for ***integer*** values of $l \in [1, l_{\max})$ (in the ***descending*** order of l), in Step I above; to ensure j>0 in the latter case, one finds $l_{\max} = \left(\sqrt{8n_s+9}-1\right)/2$. This splits the block into (at least two) "**slices**", each containing one "check point" corresponding to the smallest j-value in the "slice". In Step II above, only $q^\dagger_{j^*}$ for the current value of j^* ($=n_s, n_s-1,\ldots,0$) is stored, and the above integral is calculated (accumulated) "slice" by "slice"; note that $q^\dagger_{j^*-1}$ needs to be calculated out-of-place in REPS-K due to its successive halving of Δs. In the "slice" containing j^*, if q_{j^*} is available (*e.g.*, when j^* is a check point), the contribution of j^* to the above integral is then directly calculated; otherwise, all q_j from the "check point" in the "slice" are re-calculated and stored in the same places as the no-longer-needed q-values, then their contribution to the above integral is calculated.

The procedure of the "slice" algorithm can be illustrated with n_s=8 as follows:

(1) $q_{j=0}$ is the ***known*** initial condition of the forward propagator. Since j=0 is a "check point", $q_{j=0}$ is stored in `q[0]`, the first element of a four-element ***array*** for storing the forward propagators (note that each element is actually an array for storing the propagator values at all $\mathbf{r}$). q_j for j=1,2,…,8 are then obtained ***sequentially*** by solving the modified diffusion equation (MDE) for the forward propagator via, for example, the REPS-3 method; since here only j=3, 6 and 8 (*i.e.*, l=3, 2 and 1, respectively) are "check points", $q_{j=3}$, $q_{j=6}$ and $q_{j=8}$ are stored in `q[1]`, `q[2]` and `q[3]`, respectively. Therefore, there are four "slices".

(2) At the end of Step I, all the forward propagators are calculated, which then gives the normalized

single-chain partition function of each component and $q^{\dagger}_{j=n_s}$ (*i.e.*, the initial condition of the backward propagator). Since $j=n_s$ (*i.e.*, $l=1$) is always a "check point" (which is the only j-value contained in Slice 1), $q^{\dagger}_{j=n_s}$ is stored as `qd`, the ***variable*** for storing the backward propagators (again it is actually an array for storing the propagator values at all $\mathbf{r}$), and the contribution of segment $s=N$ to the above integral, $c_8 q_{j=8} q^{\dagger}_{j=8}$, is calculated using `q[3]` and `qd`.

(3) Slice 2 contains $j=7$ and 6. $q^{\dagger}_{j=7}$ is obtained from `qd` by solving the MDE for the backward propagator and stored in `qd` (which is overwritten since $q^{\dagger}_{j=8}$ is no longer needed). Since $j=7$ is not a "check point", $q_{j=7}$ is calculated (again) from `q[2]` by solving the MDE for the forward propagator and stored in `q[3]`. The contribution $c_7 q_{j=7} q^{\dagger}_{j=7}$ to the above integral is then calculated (and accumulated) using `q[3]` and `qd`. Similar to $q^{\dagger}_{j=7}$, $q^{\dagger}_{j=6}$ is obtained from `qd` and then stored in `qd`. Since $j=6$ is a "check point", however, the contribution $c_6 q_{j=6} q^{\dagger}_{j=6}$ to the above integral is calculated (and accumulated) using `q[2]` and `qd`.

(4) Slice 3 contains $j=5$, 4 and 3. Similar to (3), $q_{j=4}$ is calculated (again) from `q[1]` and stored in `q[2]`, and $q_{j=5}$ is then calculated (again) from `q[2]` and stored in `q[3]`; this clever design, where the number of j-values contained in the "slices" follows an arithmetic sequence of 1,2,3,…, efficiently re-uses the no-longer-needed storage for the forward propagators. For $j=5$, 4 and 3, $q^{\dagger}_j$ is obtained from `qd` and then stored in `qd`, and the contribution $c_j q_j q^{\dagger}_j$ to the above integral is then calculated (and accumulated) using `q[3]`, `q[2]` and `q[1]`, respectively, and `qd`.

(5) The last slice, Slice 4, contains the rest j-values (*i.e.*, $j=2$, 1 and 0). Similar to (4), $q_{j=1}$ is calculated (again) from `q[0]` and stored in `q[1]`, and $q_{j=2}$ is then calculated (again) from `q[1]` and stored in `q[2]`. For $j=2$, 1 and 0, $q^{\dagger}_j$ is obtained from `qd` and then stored in `qd`, and the contribution $c_j q_j q^{\dagger}_j$ to the above integral is then calculated (and accumulated) using `q[2]`, `q[1]` and `q[0]`, respectively, and `qd`.

Compared to the usual approach of calculating and storing all propagators ***before*** calculating the volume-fraction fields, the "slice" algorithm reduces the memory usage by a factor of $\sqrt{2n_s}$ for large n_s at the cost of increasing the computation by 50% (*i.e.*, solving q_j twice). Similarly, to calculate the contribution of the block to the system stresses[2] (which vanish when the bulk periodicity of an ordered phase formed by BCP self-assembly is found), we evaluate $\int_0^N \mathrm{d}s\hat{q}(\mathbf{q},s)\hat{q}^{\dagger}(-\mathbf{q},s)$ in the same way as above, where $\hat{q}(\mathbf{q},s) \equiv \int \mathbf{dr} \exp\left(-\sqrt{-1}\mathbf{q}\bullet\mathbf{r}\right) q(\mathbf{r},s) \Big/ V$ and $\hat{q}^{\dagger}(\mathbf{q},s) \equiv \int \mathbf{dr} \exp\left(-\sqrt{-1}\mathbf{q}\bullet\mathbf{r}\right) q^{\dagger}(\mathbf{r},s) \Big/ V$ with $\mathbf{q}$ being the wavevector and V the system volume.

References:

1. Qiang, Y. Chap. 6.3.2 in The High-Performance Algorithms for Self-Consistent Field Theory. PhD Thesis, Fudan University, Shanghai, China, 2022.
2. Tyler, C. A.; Morse, D. C., Stress in Self-Consistent-Field Theory. *Macromolecules* **2003,** *36* (21), 8184-8188.