

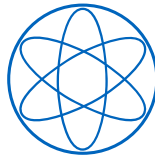


DEPARTMENT OF PHYSICS
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis
in Quantum Science & Technology

**Accelerating Open Quantum System Simulations via
Randomized SVD Tensor Factorization**

Taha Zaffar Syed





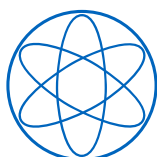
DEPARTMENT OF PHYSICS
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis
in Quantum Science & Technology

**Accelerating Open Quantum System Simulations via
Randomized SVD Tensor Factorization**

**Beschleunigung von Simulationen offener
Quantensysteme durch zufällige
SVD-Tensor-Faktorisierung**

Author: Taha Zaffar Syed
First Referee: Prof. Dr. Christian Mendl
Second Referee: Prof. Dr. Peter Rabl
Submission Date: 01 July 2025



Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe.

Munich, 01 July 2025

Taha Zaffar Syed

Acknowledgements

I would like to thank Richard Milbradt for insightful discussions. I would also like to thank my supervisor, Prof. Dr. Christian Mendl, whose course on tensor networks inspired me to embark on this journey of discovery.

Code and Figures

The DAMPF implementation was developed from scratch in Python, using NumPy for numerical operations and Ray for parallelization. Similarly, the randomized SVD routine was hand-crafted in Python with NumPy as the only external library. All figures in this thesis were created exclusively for this work with Inkscape.

Abstract

Scalable simulation of open quantum systems requires efficient treatment of non-Markovian environments. Dissipation-Assisted Matrix Product Factorization (DAMPF) combines tensor networks with pseudomode mapping to recast the integro-differential dynamics into a time-local Lindblad form. At each time step, DAMPF performs $\mathcal{O}(N^5)$ singular-value decompositions (SVDs) to compress the MPS—this dominates the runtime. In this thesis, I replace deterministic SVD with a randomized SVD (RSVD) scheme and demonstrate that RSVD (with minimal oversampling) retains relative errors below 10^{-4} while cutting the cost of each factorization by up to 50%. Across benchmark problems, this yields overall simulation speedups of up to $2\times$ without compromising accuracy. My results extend DAMPF’s reach to larger systems and longer timescales in open-system dynamics.

Die skalierbare Simulation offener Quantensysteme erfordert eine effiziente Behandlung von nicht-Markov’schen Umgebungen. Dissipationsunterstützte Matrix-Produkt-Faktorisierung (DAMPF) kombiniert Tensornetzwerke mit Pseudomode-Mapping, um die Integro-Differentialdynamik in eine zeitlich-lokale Lindblad-Form umzuformen. In jedem Zeitschritt führt DAMPF $\mathcal{O}(N^5)$ Singulärwertzerlegungen (SVDs) durch, um die MPS zu komprimieren - dies dominiert die Laufzeit. In dieser Arbeit ersetze ich die deterministische SVD durch ein randomisiertes SVD-Schema (RSVD) und zeige, dass die RSVD (mit minimalem Oversampling) die relativen Fehler unter 10^{-4} hält und gleichzeitig die Kosten jeder Faktorisierung um bis zu 50% reduziert. Bei Benchmark-Problemen führt dies zu einer Gesamtbeschleunigung der Simulation von bis zu $2mal$, ohne die Genauigkeit zu beeinträchtigen. Meine Ergebnisse erweitern die Reichweite von DAMPF auf größere Systeme und längere Zeitskalen in der Dynamik offener Systeme.

Contents

Contents	II
List of Figures	IV
List of Algorithms	VI
1 Introduction	1
2 Fundamentals	3
2.1 Matrix Factorization	3
2.1.1 QR decomposition	3
2.1.2 Singular Value Decomposition	3
2.1.3 Randomized Singular Value Decomposition	5
2.2 Tensor Networks	10
2.2.1 Notation and Vocabulary	10
2.2.2 Basic Tensor Operations	11
2.2.3 Graphical/Penrose (network) notation	12
2.2.4 Quantum States as Tensor Networks	13
2.2.5 Operations on MPS	15
2.3 Open Quantum Systems	18
3 Dissipation Assisted Matrix Product Factorization	23
3.1 Setting up the Hamiltonian	23
3.2 Tensor Network Representation	26
3.3 Dynamics	26
3.3.1 Vibrational, Electronic-Vibrational, and Damping Dynamics .	27
3.3.2 Electronic Evolution	29
3.4 Reduced Electronic State	29
3.5 The Algorithm in a Nutshell	31
3.5.1 Initialization	31
3.5.2 Evolution	35
3.6 Memory Footprint and Computational Cost	38
3.6.1 Memory	38
3.6.2 Computational Complexity	39
4 Results	42

5	Conclusion and Outlook	52
A	Action of One Body Operators	57
B	Computational Complexity of Compressing MPS	58
C	Simulation Units	59

List of Figures

2.1	Runtime comparsion between SVD and RSVD	7
2.2	Matrix Accuracy Versus Number of Singular Values Kept	8
2.3	Image Compression Via RSVD	9
2.4	Reshaping a Tesnor	12
2.5	Contraction of Tensors	12
2.6	SVD in Tensor Network Notation	13
2.7	QR in Tensor Network Notation	13
2.8	MPS of a Quantum State	14
2.9	A left to right canonicalization and compression sweep of an MPS via SVD	16
2.10	Open quantum system	18
2.11	Pseudomode Models in DAMPF	21
3.1	Pseudomode reduced environment	24
3.2	Vibrational Degrees of Freedom as a Tensor Network	26
3.3	Universal Local Update Rule of $\mathcal{O}_{m,n}$	27
3.4	Direct Sum of Two MPOs	30
3.5	Memory Blow-up of an Uncompressed State	39
4.1	Population Dynamics of a Dimer	43
4.2	Simulation Time of a Dimer	44
4.3	Truncation Error of a Dimer with $k_c = 1$	45
4.4	Truncation Error of a Dimer ($\chi = 18$) with $k_c = 2$	46
4.5	Behaviour of Truncation Error of Dimer I	46
4.6	Behaviour of Truncation Error of Dimer II	47
4.7	Truncation Error in Non-Perturbative System	48
4.8	Simulation Time of an 8-site Non-Perturbative System	48
4.9	Population dynamics in Non-Perturbative Regime	49
4.10	Population Dynamics of Polaron	50
4.11	Simulation Time of Polaron system	51
4.12	Truncation Errors of Polaron Simulation	51

List of Algorithms

1	Randomised SVD with q Power Iterations	8
2	Initialize vibronic state $\mathcal{O}$ (Algorithm 1)	32
3	Pre-compute damping matrices $\{\mathcal{B}_k\}$	33
4	Pre-compute rotation matrices $\{\mathcal{R}_{n,q}\}$	34
5	Pre-compute e–v coupling matrices $\{\mathcal{C}_{n,q}^{\text{pop}}, \mathcal{C}_{n,q}^{\text{L}}, \mathcal{C}_{n,q}^{\text{R}}\}$	35
6	Local damping step $\mathcal{B}$	36
7	Local vibrational update $\mathcal{R}$	36
8	Local e–v coupling update $\mathcal{C}$	37
9	Electronic evolution with periodic SVD compression	38

Chapter 1

Introduction

Classically representing an N -qubit wave-function requires 2^N complex amplitudes, an exponential overhead in both memory and time. Tensor-network (TN) ansätze—matrix-product states (MPS), projected-entangled-pair states (PEPS) and tree-tensor networks, among others—compress that Hilbert-space explosion by exploiting the entanglement “area law.” This compression has enabled state-of-the-art simulations in condensed-matter physics, quantum chemistry, optimisation and circuit-level quantum-computing benchmarks, because TNs recast quantum correlations into a graph of low-rank tensors that can be contracted or updated at polynomial cost. Their flexibility and favourable scaling properties make TNs the natural computational backbone for the work that follows.

Realistic quantum devices—from colour centres in diamond to light-harvesting complexes—rarely evolve in isolation; they interact with vibrational, photonic or spin baths whose spectral densities often exhibit pronounced structure. Strong coupling or sharp spectral features invalidate the Born–Markov approximations underpinning time-local Lindblad master equations. Tensor-network formulations of the *process-tensor* and TEMPO frameworks encode the full non-Markovian influence functional as a matrix-product operator, allowing memory kernels spanning hundreds of bath-correlation times to be treated exactly. Nevertheless, the computational cost still grows with the size of the bath discretisation, motivating more aggressive reductions of environmental degrees of freedom.

The *pseudomode* technique maps a continuum environment onto a finite set of auxiliary (generally non-physical) modes that obey a time-local master equation. Modern extensions relax complete positivity (quasi-Lindblad pseudomodes) and can shrink the required mode count while faithfully reproducing system dynamics. Damping each pseudomode suppresses system–bath entanglement growth, which in turn caps the bond dimension of an MPS description; the Dissipation-Assisted Matrix-Product Factorisation (DAMPF) protocol exploits exactly this effect, enabling multi-site, kilo-mode simulations on moderate hardware. Embedding pseudomodes inside a TN thus yields a compact yet accurate representation of highly structured, strongly-coupled open quantum systems.

Even with dissipation, extending simulations to longer times inevitably drives up the required bond dimensions. In standard TEBD or DMRG treatments of quantum many-body lattices, the number of singular-value decompositions (SVDs)

per time step grows only linearly with the number of sites, N . By contrast, the multi-mode pseudomode embedding and many-body couplings at the heart of the DAMPF protocol inflate this cost dramatically: each time step entails $\mathcal{O}(N^5)$ SVDs, representing a substantial increase in computational overhead.

Randomised SVD (RSVD) techniques replace deterministic SVDs with probabilistic sketches, reducing contraction costs by one to two orders of magnitude while preserving spectral accuracy with high probability. Recent studies have demonstrated RSVD accelerating Time-Evolving Block Decimation (TEBD) in practice, extending accessible simulation times without sacrificing precision[1]. Integrating RSVD into DAMPF therefore combines dissipation-assisted entanglement control with state-of-the-art low-rank factorisation, establishing the computational foundation of this thesis.

Building on these advances, this thesis analyzes the accuracy–efficiency trade-offs introduced by RSVD-based truncations at each pseudomode-damping step and benchmarks the hybrid DAMPF+RSVD algorithm on paradigmatic spin-boson models.

By uniting pseudomode embeddings, dissipation-assisted entanglement control and modern low-rank linear algebra, the thesis aims to deliver a practical blueprint for simulating open-quantum dynamics that were previously out of reach.

Chapter 2

Fundamentals

2.1 Matrix Factorization

This section follows the pure math notation of denoting matrices with boldface letters. The conjugate transpose of $\mathbf{M}$ is denoted as $\mathbf{M}^*$.

2.1.1 QR decomposition

QR decomposition factorizes a complex matrix $\mathbf{M} \in \mathbb{C}^{m \times n}$ ($m \geq n$) such that: $\mathbf{M} = \mathbf{Q}\mathbf{R}$: $\mathbf{Q} \in \mathbb{C}^{m \times m}$ where $\mathbf{Q}$ is a unitary matrix: $\mathbf{Q}^*\mathbf{Q} = \mathbf{Q}\mathbf{Q}^* = \mathbb{I}_m$ and $\mathbf{R} \in \mathbb{C}^{m \times n}$ is an upper diagonal matrix $R_{ij} = 0$ for $i > j$. When $m > n$, the bottom $(m - n)$ rows of $\mathbf{R}$ are zero and one could write:

$$\begin{aligned}\mathbf{M} &= \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 \\ \mathbb{O} \end{bmatrix} \\ &= \mathbf{Q}_1 \mathbf{R}_1\end{aligned}\tag{2.1}$$

where $\mathbf{Q}_1 \in \mathbb{C}^{m \times n}$, $\mathbf{Q}_2 \in \mathbb{C}^{m \times (m-n)}$ such that $\mathbf{Q}_1^* \mathbf{Q}_1 = \mathbb{I}_n$ and $\mathbf{Q}_2^* \mathbf{Q}_2 = \mathbb{I}_{m-n}$ (orthonormal columns). $\mathbf{R}_1 \in \mathbb{C}^{n \times n}$ contains the upper triangular part of $\mathbf{R}$ and $\mathbb{O}$ is an $(m - n) \times n$ zero matrix. $\mathbf{Q}_1 \mathbf{R}_1$ is called the *thin* or *reduced QR factorization* of $\mathbf{M}$ and is the default output of `numpy.linalg.qr`. If $\text{rank } \mathbf{M} < n$, $\mathbf{R}_1$ is still $n \times n$ dimensional with bottom $n - r$ rows equal to zero.

If $\mathbf{M} \in \mathbb{C}^{m \times n}$ such that $m < n$ (wide matrix), $\mathbf{M} = \mathbf{Q}\mathbf{R}$: $\mathbf{Q} \in \mathbb{C}^{m \times m}$ and $\mathbf{R} \in \mathbb{C}^{m \times n}$. The factorization can no longer be written in the form 2.1. $\mathbf{Q}$ is unitary: $\mathbf{Q}^* \mathbf{Q} = \mathbf{Q}\mathbf{Q}^* = \mathbb{I}_m$.

Computational Cost: QR has a computational complexity of $\mathcal{O}(mn^2)$ ($m \geq n$) and a total flop count of about $8mn^2 - \frac{8}{3}n^3$.

2.1.2 Singular Value Decomposition

Full and compact SVD

Singular value decomposition (SVD) is the factorization of a matrix $\mathbf{M} \in \mathbb{C}^{m \times n}$ into $\mathbf{U}\mathbf{S}\mathbf{V}^*$: $\mathbf{U} \in \mathbb{C}^{m \times m}$, $\mathbf{S} \in \mathbb{R}^{m \times n}$, and $\mathbf{V} \in \mathbb{C}^{n \times n}$. $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices:

$\mathbf{U}\mathbf{U}^* = \mathbf{U}^*\mathbf{U} = \mathbb{I}_m$, $\mathbf{V}\mathbf{V}^* = \mathbf{V}^*\mathbf{V} = \mathbb{I}_n$. $\mathbf{S}$ is a rectangular diagonal matrix of non-negative real numbers σ_{ii} called the singular values (which are unique). The number of non-zero singular values is exactly equal to the rank of $\mathbf{M}$. The columns $\mathbf{u}_1, \dots, \mathbf{u}_m$ and $\mathbf{v}_1, \dots, \mathbf{v}_n$ of $\mathbf{U}$ and $\mathbf{V}$ form two sets of orthonormal bases. If the factorization is done in a way that singular values σ_{ii} in $\mathbf{S}$ appear in a descending order on the diagonal (non-zero singular values appearing in lowest numbered columns/rows is sufficient), then, $\mathbf{M}$ can be exactly written as:

$$\mathbf{M} = \sum_i^r \sigma_{ii} \mathbf{u}_i \mathbf{v}_i^* \quad (2.2)$$

where $r \leq \min(m, n)$ is the rank of $\mathbf{M}$.

Instead of storing fully sized matrices $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$ (*full SVD*), 2.2 makes it possible to write the factorization as $\mathbf{M} = \mathbf{U}_r \mathbf{S}_r \mathbf{V}_r^*$: $\mathbf{U}_r \in \mathbb{C}^{m \times r}$, $\mathbf{S}_r \in \mathbb{R}^{r \times r}$, and $\mathbf{V}_r \in \mathbb{C}^{n \times r}$. $\mathbf{U}_r$ and $\mathbf{V}_r$ are no longer necessarily unitary but their columns are still orthonormal: $\mathbf{U}_r^* \mathbf{U}_r = \mathbb{I}_r$ and $\mathbf{V}_r^* \mathbf{V}_r = \mathbb{I}_r$. This factorization is still exact and referred to as *compact SVD*, *economy-size SVD*, or *thin SVD*. Compact SVD has a smaller memory footprint than a full SVD. In python, `numpy.linalg.svd(M, full_matrices)` performs SVD of input matrix `M` while the boolean `full_matrices` decides between full SVD or compact SVD.

Truncated SVD and Matrix Compression

The Frobenius norm of a matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ is defined as:

$$\|\mathbf{A}\|_F = \sqrt{\sum_i^m \sum_j^n |a_{ij}|^2} \quad (2.3)$$

where $|a_{ij}|^2 = a_{ij} \overline{a_{ij}}$ with $\overline{a_{ij}}$ being the complex conjugate of $|a_{ij}|$. The measure of distance (their *closeness*) between two matrices $\mathbf{A} \in \mathbb{C}^{m \times n}$ and $\mathbf{B} \in \mathbb{C}^{m \times n}$ is defined as the Frobenius norm of their difference:

$$\|\mathbf{A} - \mathbf{B}\|_F = \sqrt{\sum_i^m \sum_j^n |a_{ij} - b_{ij}|^2} \quad (2.4)$$

which can equivalently be written as:

$$\|\mathbf{A} - \mathbf{B}\|_F = \sqrt{\text{Tr}[(\mathbf{A} - \mathbf{B})^* (\mathbf{A} - \mathbf{B})]} \quad (2.5)$$

For a matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ of rank r , the compact SVD gives $\mathbf{M} = \mathbf{U}_r \mathbf{S}_r \mathbf{V}_r^*$. If only k columns of $\mathbf{U}_r$ and $\mathbf{V}_r$, and only k corresponding singular values are kept ($k \leq r$) (correspondence according to 2.2), one can write: $\mathbf{M}'_k = \mathbf{U}'_k \mathbf{S}'_k \mathbf{V}'_k^*$: $\mathbf{U}'_k \in \mathbb{C}^{m \times k}$, $\mathbf{S}'_k \in \mathbb{R}^{k \times k}$ and $\mathbf{V}'_k \in \mathbb{C}^{n \times k}$.

$$\mathbf{M} \in \mathbb{C}^{m \times n}, \quad \text{rank}(\mathbf{M}) = r,$$

The question could be posed that how does one find the set $\{\sigma_k\}$ ($k \leq n$) such that $\|\mathbf{M} - \mathbf{M}'_k\|_F$ is minimized.

The solution to this optimization problem is given by Eckart–Young–Mirsky theorem [2] which states that the optimal rank approximation is obtained by keeping only the largest k singular values:

Theorem 1: Eckart–Young–Mirsky Theorem

Let $\mathbf{M} = \mathbf{U}_r \mathbf{S}_r \mathbf{V}_r^*$, $r = \text{rank}(\mathbf{M})$.

If $\tilde{\mathbf{M}}_k = \tilde{\mathbf{U}}_k \tilde{\mathbf{S}}_k \tilde{\mathbf{V}}_k^*$,

$$\tilde{\mathbf{S}}_k = \text{diag}(\sigma_1, \dots, \sigma_k), \sigma_1 > \sigma_2 > \dots > \sigma_k > \dots > \sigma_r, \{\sigma_i\}_{i=1}^r \subseteq \text{diag}(\mathbf{S}_r),$$

then

$$\|\mathbf{M} - \tilde{\mathbf{M}}_k\|_F = \sqrt{\sum_{i=k+1}^n \sigma_i^2} = \min_{\text{rank}(\mathbf{M}')=k} \|\mathbf{M} - \mathbf{M}'\|_F.$$

Consequently, if a compact SVD of $\mathbf{M}$ is performed in such a way that singular values appear in a descending order on the diagonal of $\mathbf{S}_r$ (as is the case for `np.linalg.svd`), to get a lower rank k ($k \leq r$) approximation one simply deletes the last $r - k$ columns from $\mathbf{U}_r$ and $\mathbf{V}_r$ while also deleting the last $r - k$ rows and columns of $\mathbf{S}_r$. The resulting $\mathbf{M}_k = \mathbf{U}_k \mathbf{S}_k \mathbf{V}_k^*$ is called the *truncated* SVD of $\mathbf{M}$. $\mathbf{M}_k$ is a *compressed* version of $\mathbf{M}$. The Frobenius norm of the difference between the two matrices, provided by Eckart–Young–Mirsky Theorem, is called the *Frobenius Error* of the approximation which can be controlled by keeping track of the discarded singular values.

Computational Cost: The exact mathematical procedure of performing SVD of a matrix is immaterial to the work in this thesis. It suffices to note the complexity: The compact SVD of a real matrix requires $4mn^2 + \frac{8}{3}n^3$ flops where n is the smaller dimension of the matrix (assuming $r = n$) [3]. For complex matrices, the complex arithmetic increases the flop count by a factor of 4 and the total count becomes $16mn^2 + \frac{32}{3}n^3$ [4]. While the complexity $\mathcal{O}(mn^2)$ is the same for both real and complex cases, it takes four times more compute time to factorize a complex matrix.

2.1.3 Randomized Singular Value Decomposition

To calculate a truncated rank factorization of a matrix all standard computational tools in Numpy, Julia, SciPy, etc rely on LAPACK routines (a linear algebra library of algorithms performing computations under the hood). Even if $k < r$, LAPACK evaluates a compact SVD and then truncates the rank, keeping the compute time identical to full and compact SVD. In situations where only a truncated rank approximation is required by design, time and computational resources are spent on information that is discarded anyway. In this section we look at RSVD, a randomness based algorithm that evaluates only the truncated matrices $\mathbf{U}_k$, $\mathbf{S}_k$, and $\mathbf{V}_k$ [5]. It should be mentioned that classical numerical-linear-algebra offers several deterministic ways to avoid the full factorization—e.g. Lanczos iteration or implicitly restarted

Arnoldi, which return only the leading k singular triplets. However, in this thesis, only RSVD will be considered.

To obtain the k -rank approximation $\mathbf{M}_k^{rsvd}$ of a matrix $\mathbf{M} \in \mathbb{C}^{m \times n}$ The randomized reduced-rank SVD proceeds as follows:

- Draw a random complex matrix $\mathbf{\Omega} \in \mathbb{C}^{n \times k}$.
- Form the sample matrix $\mathbf{Y} \in \mathbb{C}^{m \times k} : \mathbf{Y} = \mathbf{M}\mathbf{\Omega}$
- Perform a QR decomposition of $\mathbf{Y} : \mathbf{Y} = \mathbf{Q}\mathbf{R} : \mathbf{Q} \in \mathbb{C}^{m \times k}$ and then create $\mathbf{B} = \mathbf{Q}^* \mathbf{M} : \mathbf{B} \in \mathbb{C}^{k \times n}$
- Perform compact SVD of the smaller matrix $\mathbf{B} : \mathbf{B} = \tilde{\mathbf{U}}\mathbf{S}\mathbf{V}^*$ where $\tilde{\mathbf{U}} \in \mathbb{C}^{k \times k}$, $\mathbf{S} \in \mathbb{R}^{k \times k}$, and $\mathbf{V} \in \mathbb{C}^{k \times n}$. Form $\mathbf{U} = \mathbf{Q}\tilde{\mathbf{U}}$.
- $\mathbf{M}_k^{rsvd} = \mathbf{U}\mathbf{S}\mathbf{V}^*$

The error incurred because of the randomized nature of the truncation can be quantified as:

$$e_k = \|\mathbf{M} - \mathbf{M}_k^{rsvd}\|_F \quad (2.6)$$

The minimum error $\|\mathbf{M} - \mathbf{M}_k\|_F$ which is incurred by the exact truncation $\mathbf{M}_k$ itself equals σ_{k+1} where σ_{k+1} is the $(k+1)^{th}$ singular value of $\mathbf{M}$. To see this we look at Eckart-Young-Theorem and think of the best case scenario: All singular values of $\mathbf{M}$ beyond $i+1$ are zero i.e. $\sigma_j = 0 : j > i+1$. The ratio $\frac{e_k}{\sigma_{k+1}}$ then provides a measure of how close our computed approximation comes to the best-possible. A value near 1 means “nearly optimal”; values significantly above 1 signal loss of accuracy. With an oversampling of $p \geq 2$ the variance of the satisfies the following theorem[5]:

Theorem 2: Error bound for the randomised SVD

Let $\mathbf{M} \in \mathbb{C}^{m \times n}$ have singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$ and let $k < r$ be a target rank. Fix an oversampling parameter $p \geq 2$, and let

$$\mathbf{M}_{k+p}^{rsvd} = \text{RSVD}(\mathbf{M}, k, p)$$

be the rank- k approximation returned by the RSVD algorithm. Then the expected 2-norm error satisfies

$$\mathbb{E}\|\mathbf{M} - \mathbf{M}_{k+p}^{rsvd}\|_2 \leq \left(1 + \sqrt{\frac{k}{p-1}}\right) \sigma_{k+1} + \frac{e \sqrt{k+p}}{p} \left(\sum_{j>k} \sigma_j^2\right)^{1/2}. \quad (2.7)$$

It is immediately noted that the second part of the equation becomes negligible if the singular values of $\mathbf{M}$ are rapidly decaying.

When the singular values of $\mathbf{M}$ decay slowly, one introduces $q \geq 1$ power-iteration steps before sketching:

$$\mathbf{Y} = (\mathbf{M}\mathbf{M}^*)^q \mathbf{M} \mathbf{\Omega},$$

where $\mathbf{\Omega} \in \mathbb{C}^{n \times (k+p)}$ is a random Gaussian matrix. After computing the QR factorization $\mathbf{Y} = \mathbf{Q}\mathbf{R}$, the algorithm proceeds as before.

Power iterations transform the singular values via

$$\sigma_j((\mathbf{M}\mathbf{M}^*)^q\mathbf{M}) = (\sigma_j(\mathbf{M}))^{2q+1},$$

amplifying the gap between the top k modes and the tail. As a result, the expected spectral-norm error obeys

$$\mathbb{E}\|\mathbf{M} - \mathbf{M}_{k+p}^{\text{rsvd}}\|_2 \leq \left(1 + \sqrt{\frac{k}{p-1}}\right)^{1/(2q+1)} \sigma_{k+1}, \quad (2.8)$$

showing that the factor above σ_{k+1} decays exponentially in q , at the cost of $2q + 1$ passes over $\mathbf{M}$.

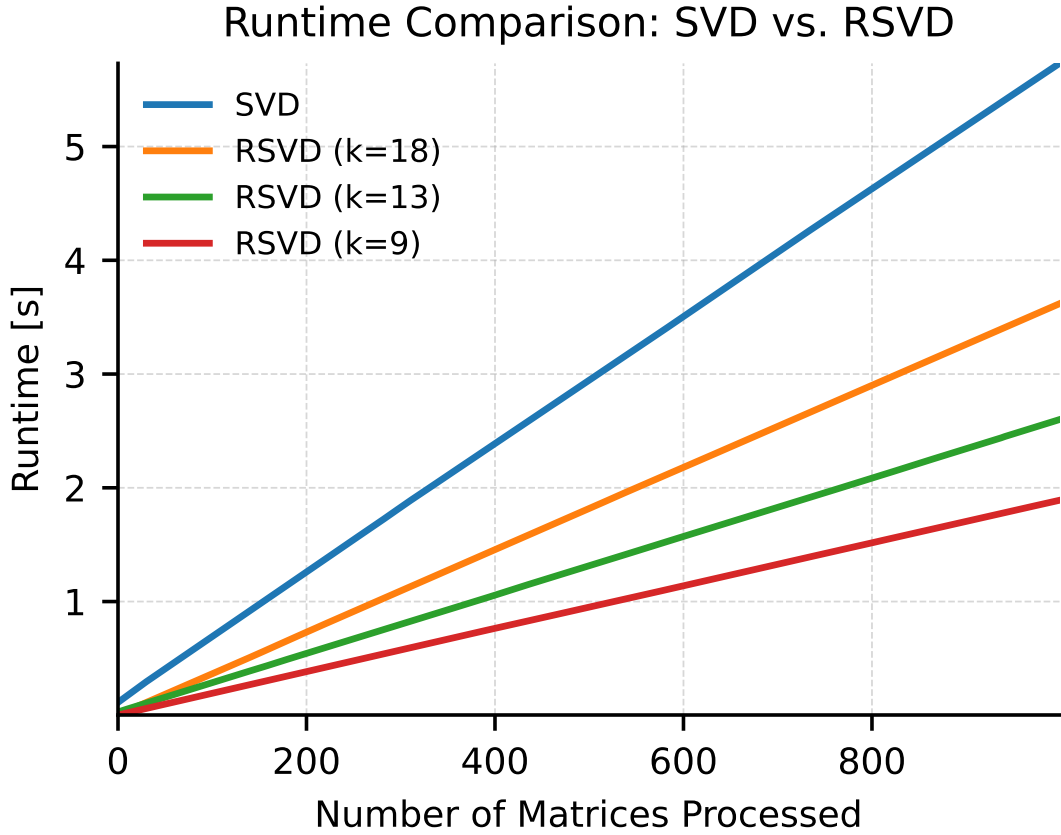


Figure 2.1: Runtime comparison between full SVD and randomized SVD for $1000 (2\chi d^2 \times 2\chi)$ randomly drawn complex matrices with $\chi = 18$ and $d = 8$. For RSVD, three truncation ranks were tested, retaining $k \in \{18, 13, 9\}$ singular values, with *zero* power iterations ($q = 0$) and *zero* oversampling ($p = 0$).

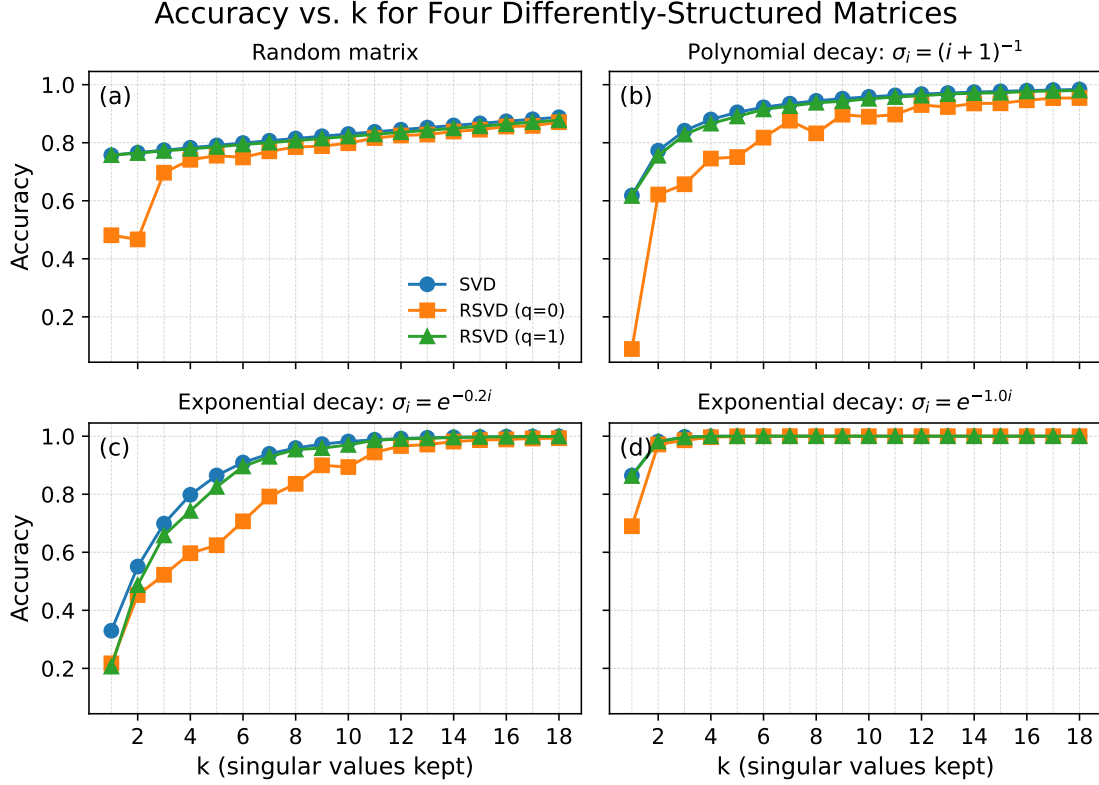


Figure 2.2: Accuracy, measured as $1 - \frac{\|A - A_k\|_F}{\|A\|_F}$, versus the number of retained singular values k for four test matrices of size $(2\chi d^2) \times (2\chi)$ with $\chi = 18$ and $d = 8$. Each subplot compares the standard SVD and two randomized SVD variants (power iterations $q = 0$ and $q = 1$) on: (a) a random matrix; (b) a matrix with polynomially decaying spectrum $\sigma_i = (i + 1)^{-1}$; (c) a matrix with exponential decay $\sigma_i = e^{-0.2i}$; and (d) a matrix with faster exponential decay $\sigma_i = e^{-1.0i}$. Higher values indicate better low-rank approximation accuracy.

Algorithm 1 Randomised SVD with q Power Iterations

Require: $\mathbf{A} \in \mathbb{C}^{m \times n}$, target rank $\ell = k + p$, power-iteration depth $q \geq 0$

Ensure: Approximate rank- k factorisation $\mathbf{A} \approx \mathbf{U}_k \Sigma_k \mathbf{V}_k^\dagger$

- 1: Draw $\Omega \in \mathbb{C}^{n \times \ell}$ with i.i.d. $\mathcal{N}(0, 1)$ entries
- 2: $\mathbf{Y}_0 \leftarrow \mathbf{A}\Omega$
- 3: $[\mathbf{Q}_0, \mathbf{R}_0] \leftarrow \text{qr}(\mathbf{Y}_0)$ $\triangleright$ thin QR
- 4: **for** $j = 1$ **to** q **do**
- 5: $\tilde{\mathbf{Y}}_j \leftarrow \mathbf{A}^\dagger \mathbf{Q}_{j-1}$
- 6: $[\tilde{\mathbf{Q}}_j, \tilde{\mathbf{R}}_j] \leftarrow \text{qr}(\tilde{\mathbf{Y}}_j)$
- 7: $\mathbf{Y}_j \leftarrow \mathbf{A} \tilde{\mathbf{Q}}_j$
- 8: $[\mathbf{Q}_j, \mathbf{R}_j] \leftarrow \text{qr}(\mathbf{Y}_j)$
- 9: **end for**
- 10: $\mathbf{Q} \leftarrow \mathbf{Q}_q$ $\triangleright$ orthonormal basis for the sketch
- 11: $\mathbf{B} \leftarrow \mathbf{Q}^\dagger \mathbf{A}$ $\triangleright \mathbf{B} \in \mathbb{C}^{\ell \times n}$
- 12: $[\tilde{\mathbf{U}}, \Sigma, \tilde{\mathbf{V}}^\dagger] \leftarrow \text{svd}(\mathbf{B})$ $\triangleright$ exact SVD of the small matrix
- 13: $\mathbf{U} \leftarrow \mathbf{Q} \tilde{\mathbf{U}}$ $\triangleright$ lift back to the original space
- 14: **return** $\mathbf{U}_{(:,1:k)}, \Sigma_{1:k,1:k}, \mathbf{V}_{(:,1:k)}$

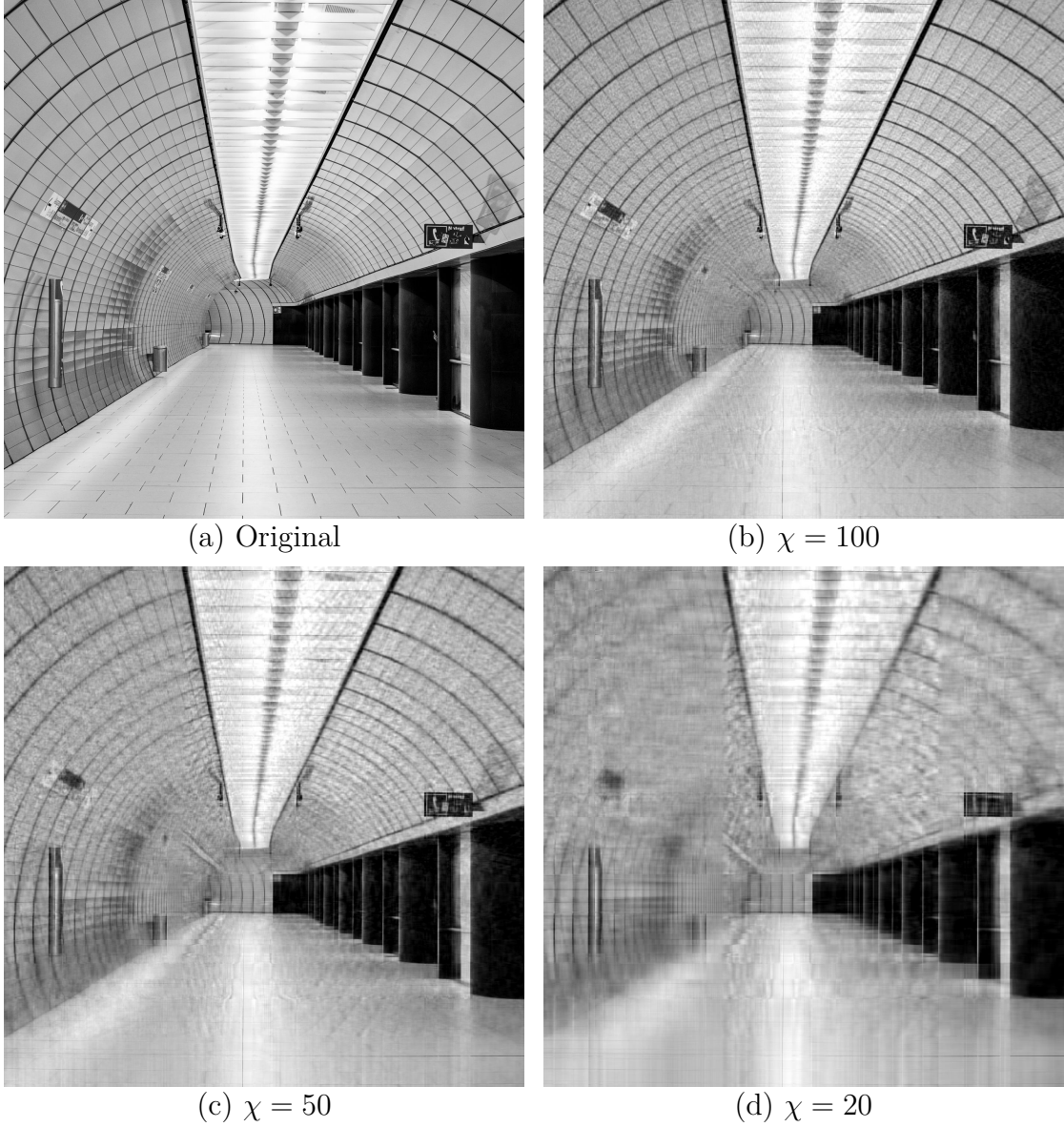


Figure 2.3: An image of Marienplatz U-Bahn station compressed (truncation of the underlying 1400×1400 matrix of pixel values) via Randomized SVD

Photo credits: © Taha Zaffar, 2023. All rights reserved.

Computational Complexity: For a complex $m \times n$ matrix A ($m \geq n$), a rank- k randomized SVD with sketch size $\ell = k + p$ and q power iterations incurs the following major costs:

- **Sketching:** Compute $Y = A\Omega \in \mathbb{R}^{m \times \ell}$ (and q optional power-iterations).
- **Orthogonalization:** Obtain $Q \in \mathbb{R}^{m \times \ell}$ via QR of Y .
- **Projection:** Form the small matrix $B = Q^T A \in \mathbb{R}^{\ell \times n}$.
- **Small SVD:** Compute the SVD of $B = \tilde{U} \Sigma V^T$.
- **Reconstruction:** Recover approximate left singular vectors $U = Q \tilde{U}$.

The total leading-order cost combines to $2(q+1)mn\ell$. Because $\ell = k + p$ with $p \ll k \ll n$, randomized SVD runs in essentially $O(mnk)$ time, much faster than the $O(mn^2)$ cost of a full SVD when $k \ll n$.

2.2 Tensor Networks

2.2.1 Notation and Vocabulary

The knowledge of tensor networks is central to the work in this thesis. This section provides an overview of the topic that is sufficient to follow the subsequent work. The reader is redirected to [6] and [7] for a comprehensive introduction to tensor networks in the context of quantum systems.

Put simply, a *tensor* is a multi-dimensional array (here “dimension” is meant in the physical sense of the word, i.e. how many indices the object carries). For instance, a *vector* is a one-dimensional array and may be written as

$$A_i, \quad i \in \{1, \dots, d_i\},$$

while a *matrix* is a two-dimensional array,

$$A_{ij}, \quad i \in \{1, \dots, d_i\}, j \in \{1, \dots, d_j\}.$$

In tensor-network language we generalise this idea:

A vector is a tensor of **order** 1. A matrix is a tensor of **order** 2. More generally,

$$A_{i_1 i_2 \dots i_n}, \quad i_k \in \{1, \dots, d_{i_k}\},$$

is a tensor of **order** n .

The words *order*, *degree*, and *rank* are used interchangeably in this context (*rank* here should *not* be confused with the rank of a matrix in linear-algebra).

Each index i_k has its own *dimension* d_{i_k} , the number of values it can assume. Equivalently, the tensor

$$A_{ij}, \quad i \in \{1, \dots, d_i\}, j \in \{1, \dots, d_j\},$$

lives in the tensor-product space

$$\mathbb{C}^{d_i} \otimes \mathbb{C}^{d_j}.$$

A tensor whose indices all share the same size d is called *homogeneous of local dimension* d . In Python, tensors are best stored as Numpy arrays. The `shape` of the array provides the structural information of the tensor: number of elements gives the order of the tensor and each element is the local dimension of that index. The shape of rank-4 tensor $A_{ijkl} : i \in \{1, \dots, d_i\}, j \in \{1, \dots, d_j\}, k \in \{1, \dots, d_k\}, l \in \{1, \dots, d_l\}$ is (d_i, d_j, d_k, d_l)

2.2.2 Basic Tensor Operations

The indices of a tensor can be shuffled or merged together into a super-index whose dimension equals the total product of the dimension of merged indices. For example, the first two indices of A_{ijkl} can be merged together to create $\tilde{A}_{xkl} : x \in \{1, \dots, d_i d_j\}$, $k \in \{1, \dots, d_k\}$, $l \in \{1, \dots, d_l\}$ with shape (x, d_k, d_l) . This *reshaping* only changes the way the data is indexed and not the data itself. A tensor can be reshaped to any new shape as long as the total size, the product of dimensions of all indices, is the same for old and new tensor. Fusing all indices together into a single index is called *vectorization* while merging indices together to create only two indices is known as *matricization*. Reshaping higher order tensors into matrices is an extremely important operation in tensor networks because it lets us manipulate data of the original tensor using methods that are only defined for a matrix. In Numpy reshaping is done via `numpy.reshape`.

Let $A_{i_1 \dots i_m j_1 \dots j_p}$ and $B_{j_1 \dots j_p k_1 \dots k_q}$ be two tensors that share the index set $\{j_1, \dots, j_p\}$. Their *contraction* over these shared indices is the tensor

$$(A * B)_{i_1 \dots i_m k_1 \dots k_q} = \sum_{j_1=1}^{d_{j_1}} \cdots \sum_{j_p=1}^{d_{j_p}} A_{i_1 \dots i_m j_1 \dots j_p} B_{j_1 \dots j_p k_1 \dots k_q}.$$

Each contracted pair removes one index from the total order, so the resulting tensor has order $m + q$.

Throughout this note *any subscript that appears exactly twice in a monomial is implicitly summed over all its values* (Einstein notation). With that convention the previous line simplifies to

$$(A * B)_{i_1 \dots i_m k_1 \dots k_q} = A_{i_1 \dots i_m j_1 \dots j_p} B_{j_1 \dots j_p k_1 \dots k_q}.$$

Elementary Examples:

- Vector inner product: For $v_i, w_i \in \mathbb{C}^d$,

$$v_i w_i = \sum_{i=1}^d v_i w_i,$$

a scalar obtained by contracting the single index i .

- Matrix–vector multiplication: Given a matrix M_{ij} and a vector v_j ,

$$u_i = M_{ij} v_j,$$

leaving the uncontracted index i .

- Matrix multiplication: For A_{ik} and B_{kj} ,

$$C_{ij} = A_{ik} B_{kj},$$

a contraction over k that yields another rank-2 tensor.

Computational Cost: Contracting an index of dimension d incurs $\mathcal{O}(d)$ arithmetic operations. Consequently, contracting p such index pairs between tensors of total sizes N and M costs $\mathcal{O}(d^p N M)$. For networks of many tensors, the order in which contractions are performed strongly affects runtime; finding an optimal sequence is in general NP-hard, so heuristic or approximate planners (greedy, tree-width, etc.) are commonly used. If all indices have the same local dimension d , a contraction that removes p index pairs multiplies the computational effort by d^p and lowers the total order by p .

2.2.3 Graphical/Penrose (network) notation

In diagrams every tensor is a node and each subscript is a leg. Free legs represent uncontracted indices; legs that connect two nodes (or form a loop on one node) denote a contracted pair. Matrix multiplication, for example, is drawn as two rank-2 nodes with their adjoining legs joined. Reshape simply means joining two physical legs together into one leg. While the shape of nodes can be chosen arbitrarily, this thesis follows the convention of many authors in physics: an unstructured tensor is drawn as a square/rectangle, a left isometry $\mathbf{M} : \mathbf{M}^\dagger \mathbf{M} = \mathbb{I}$ is denoted with a right-pointing triangle while a right isometry $\mathbf{M} \mathbf{M}^\dagger = \mathbb{I}$ is denoted with a left-pointing triangle. A diagonal matrix is drawn as a 45° tilted square. An identity is drawn as a line (not necessarily straight). Figure 2.5 represents contraction $\sum_j^d A_{ij} B_{jkl} = C_{ikl}$. In Python this would be done as `C = np.tensordot(A, B, axes=([1], [0]))`. However, the resulting tensor C is of shape (j, k, l) and needs a `reshape` operation. `C = np.einsum('ij,jkl->ikl', A, B)` performs the contraction with C in desired shape.

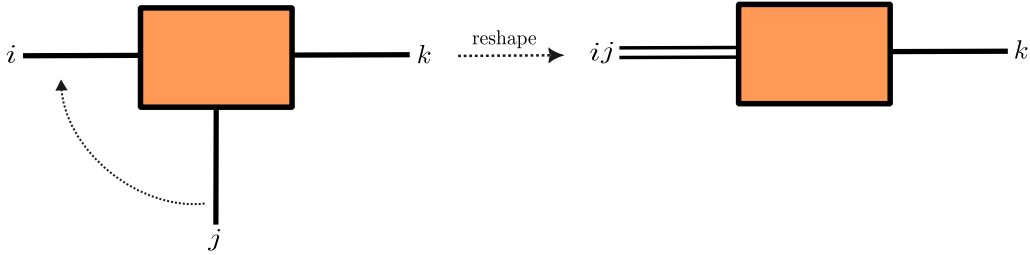


Figure 2.4: Reshaping a tensor of shape (i, j, k) to (ij, k)

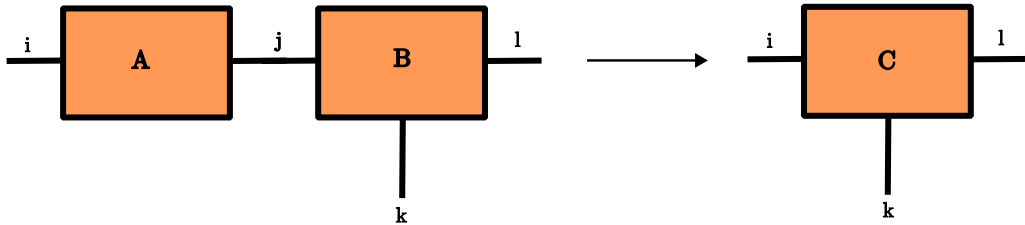


Figure 2.5: Contraction of a matrix with a rank-3 tensor

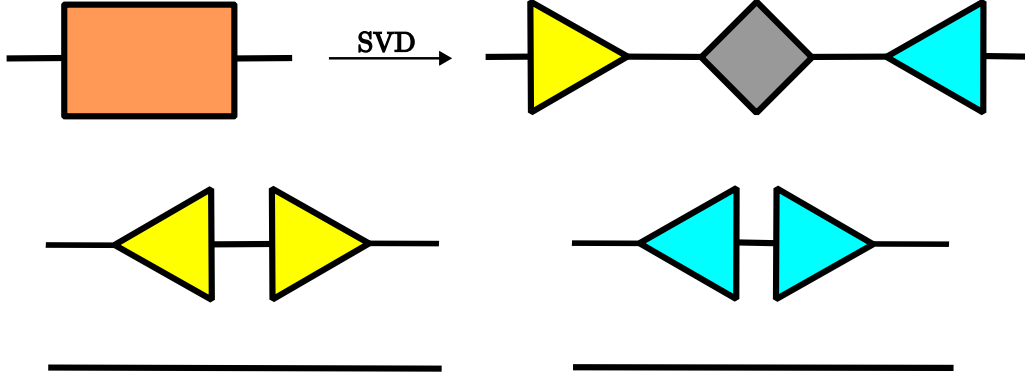


Figure 2.6: Compact SVD in tensor diagram notation.

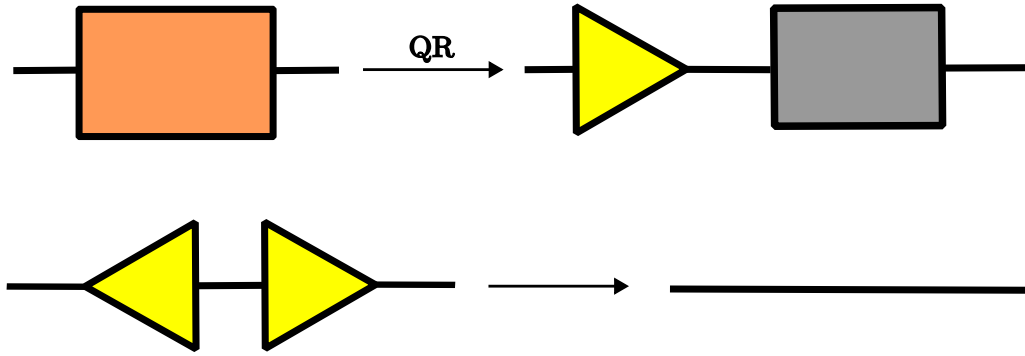


Figure 2.7: Thin QR in tensor diagram notation.

2.2.4 Quantum States as Tensor Networks

A pure state of N qudits lives in the tensor product Hilbert space $\mathcal{H} = \bigotimes_{i=1}^N \mathbb{C}^d$ and can be written in the local basis as

$$|\Psi\rangle = \sum_{s_1, \dots, s_N=1}^d C_{s_1 \dots s_N} |s_1 s_2 \dots s_N\rangle.$$

where $|s_1 s_2 \dots s_N\rangle \equiv |s_1\rangle \otimes |s_2\rangle \otimes \dots \otimes |s_N\rangle$. The d^N amplitudes $C_{s_1 \dots s_N}$ scale exponentially with N , so a direct description soon becomes infeasible.

It is possible to write $|\Psi\rangle$ in the form:

$$|\Psi\rangle = \sum_{\{s_i\}} \left(A_{\chi_0 s_1 \chi_1}^{[1]} A_{\chi_1 s_2 \chi_2}^{[2]} \dots A_{\chi_{N-1} s_N \chi_N}^{[N]} \right) |s_1 s_2 \dots s_N\rangle \quad (2.9)$$

where $A_{\chi_{i-1} s_i \chi_i}^{[i]}$, called the *site tensor*, is a rank-3 tensor with χ_{i-1} and χ_i are its *virtual indices* and s_i its *physical index*. χ_i is the *bond dimension* between $A_{\chi_{i-1} s_i \chi_i}^{[i]}$ and $A_{\chi_i s_{i+1} \chi_{i+1}}^{[i+1]}$. These tensors are obtained via successive SVDs of the coefficient $C_{s_1, \dots, s_N}$. Such a representation is called *Matrix Product State (MPS)* of $|\Psi\rangle$. The maximal internal dimension $\chi = \max \chi_i$ is called the *bond dimension* of the MPS

as a whole. The MPS contains $Nd\chi^2$ elements (if $\chi_i = \chi$) instead of d^N required in exact representation, with χ being the parameter that controls the accuracy of $|\Psi\rangle$ in its MPS representation.

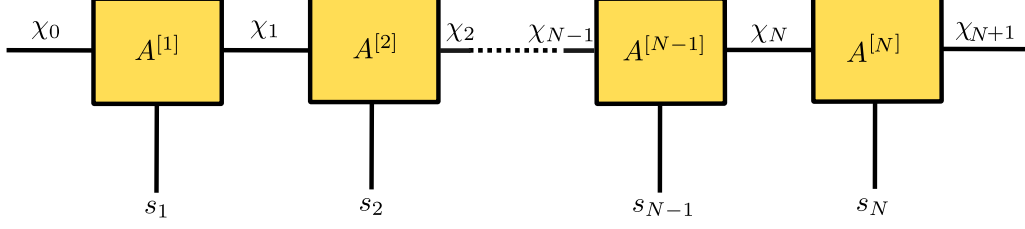


Figure 2.8: MPS of a quantum state in tensor network notation.

An arbitrary linear operator acting on the full Hilbert space $\mathcal{H} = \bigotimes_{i=1}^N \mathbb{C}^d$ is completely fixed by a rank- $2N$ coefficient tensor $O_{\mathbf{s}'}^{\mathbf{s}}$ with $\mathbf{s} = s_1 \dots s_N$ and $\mathbf{s}' = s'_1 \dots s'_N$. In the local product basis $\{|s_1 \dots s_N\rangle\}$ we write

$$\hat{O} = \sum_{\substack{s_1, \dots, s_N=1 \\ s'_1, \dots, s'_N=1}}^d O_{s'_1 \dots s'_N}^{s_1 \dots s_N} |s_1 \dots s_N\rangle \langle s'_1 \dots s'_N|. \quad (2.10)$$

Exactly as a many-body *state* can be factorised into a string of rank-3 tensors (an MPS), the rank- $2N$ coefficient tensor $O_{s'_1 \dots s'_N}^{s_1 \dots s_N}$ of a generic operator can be broken—via successive Schmidt/SVD decompositions—into a chain of rank-4 tensors. The resulting *matrix-product operator* (MPO) reads:

$$\hat{O} = \sum_{\{s_i, s'_i\}} W_{\chi_0 s_1 s'_1 \chi_1}^{[1]} W_{\chi_1 s_2 s'_2 \chi_2}^{[2]} \dots W_{\chi_{N-1} s_N s'_N \chi_N}^{[N]} |s_1 \dots s_N\rangle \langle s'_1 \dots s'_N| \quad (2.11)$$

where $\chi_0 = \chi_N = 1$. Each site tensor $W_{\chi_{i-1} s_i s'_i \chi_i}^{[i]}$ carries two *physical* indices—the ket index s_i and bra index s'_i —and two *virtual* (bond) indices χ_{i-1}, χ_i that transmit correlations along the chain. The largest virtual size $\chi_O = \max_i \chi_i$ is the MPO's *bond dimension*.

In an orthonormal single-site operator basis $\{X_\alpha\}_{\alpha=1}^{d^2}$ that satisfies $\text{Tr}(X_\alpha^\dagger X_\beta) = d\delta_{\alpha\beta}$, any N -site operator can first be written in its fully expanded form

$$\begin{aligned} \hat{O} &= \sum_{\alpha_1, \dots, \alpha_N=1}^{d^2} c_{\alpha_1 \dots \alpha_N} X_{\alpha_1} \otimes X_{\alpha_2} \otimes \dots \otimes X_{\alpha_N} \\ c_{\alpha_1 \dots \alpha_N} &= d^{-N} \text{Tr} \left[\left(X_{\alpha_1} \otimes \dots \otimes X_{\alpha_N} \right)^\dagger \hat{O} \right], \end{aligned} \quad (2.12)$$

which still involves d^{2N} coefficients. In the operator basis 2.11 can be written as:

$$\hat{O} = \sum_{\alpha_1, \dots, \alpha_N=1}^{d^2} \left(M_{\chi_0 \alpha_1 \chi_1}^{[1]} M_{\chi_1 \alpha_2 \chi_2}^{[2]} \dots M_{\chi_{N-1} \alpha_N \chi_N}^{[N]} \right) X_{\alpha_1} \otimes X_{\alpha_2} \otimes \dots \otimes X_{\alpha_N}, \quad (2.13)$$

where $\chi_0 = \chi_N = 1$. Equation (2.13) is simply a reshaped version of the customary ket–bra MPO $W_{\chi_{i-1} s_i s'_i \chi_i}^{[i]}$ introduced in Eq. (2.11). Fuse each bra/ket pair into a single operator-basis index,

$$\alpha_i = (s_i - 1)d + s'_i, \quad 1 \leq \alpha_i \leq d^2,$$

and define

$$M_{\chi_{i-1}\alpha_i\chi_i}^{[i]} \equiv W_{\chi_{i-1}s_i s'_i\chi_i}^{[i]}.$$

This way the operations that are performed on an MPS can be can also be applied to an MPO. For this reason, MPS and MPO are used interchangeably in in this thesis even though the actual object we will be dealing with in later sections is a density operator written in the form of (2.13).

2.2.5 Operations on MPS

Once a state is written in the form of Equation 2.9 the tensor network symbolism provides a highly intuitive way of visualizing the operations. In this section only those operations which are relevant for DAMPF are listed.

Compression Sweeps

One of the most widely used operations on an MPS is compression, which consists of first *canonicalising* the state (transforming tensors into isometries) and then truncating small singular values to reduce the bond dimension. Figure 2.9 shows symbolically how a left-to-right sweep via SVD is performed. We start at the left most tensor $A^{[1]}$ and reshape it from (χ_0, s_1, χ_1) to $(\chi_0 s_1, \chi_1)$. The reshaped matrix is then factorized via SVD (or QR) to $\bar{A}^{[1]} \in \mathbb{C}^{\chi_0 s_1 \times \bar{\chi}_1}$, $S^{[1]} \in \mathbb{R}^{\bar{\chi}_1 \times \bar{\chi}_1}$, and $V_h^{[1]} \in \mathbb{C}^{\bar{\chi}_1 \times \chi_1}$ where $\bar{\chi}_1$ is the truncated rank of $A^{[1]}$. $\bar{A}^{[1]}$ is reshaped back to $(\chi_0, s_1, \bar{\chi}_1)$ and becomes the new tensor at site-1. $S^{[1]}$ and $V_h^{[1]}$ are matrix-multiplied together and then contracted with the tensor at site-2. As a result, the tensor at site-1 is a left isometry and the bond dimension is reduced. The same process is repeated towards the end. For open boundary MPS ($\chi_0 = \chi_1 = 1$) one usually stops at $N - 1$ site after $S^{[N-1]}$ and $V_h^{[N-1]}$ are matrix-multiplied and then contracted with $A^{[N]}$. If QR was used instead of SVD then a rank truncation would not have been possible. A similar process could be started from site- N (only SVD can be used from right-to-left sweep) where $\bar{V}_h^{[N]}$ would be the new tensor at site N and $S^{[N]}$ and $U^{[N]}$ are absorbed into $A^{[N-1]}$. MPS canonicalization sweeps are done in both directions for numerical stability: left-to-right followed by right-to-left or vice-versa. The error that is incurred in such a sweep is bound by the sum of squares of all discarded singular values [8]:

Theorem 3: Compression error for an MPS sweep

Let $|\Psi\rangle$ be a matrix-product state (MPS) on N sites with local dimension d and bond dimensions χ_b ($b = 1, \dots, N - 1$). Fix a cut-off $\chi_c < \max_b \chi_b$. Perform a compression-sweep retaining only the χ_c largest singular values on every bond, and denote the resulting state by $|\tilde{\Psi}\rangle$. If $\sigma_{b,i}$ are the discarded singular values on bond b , then

$$\| |\Psi\rangle - |\tilde{\Psi}\rangle \|^2 \leq 2 \sum_{b=1}^{N-1} \sum_{i > \chi_c} \sigma_{b,i}^2.$$

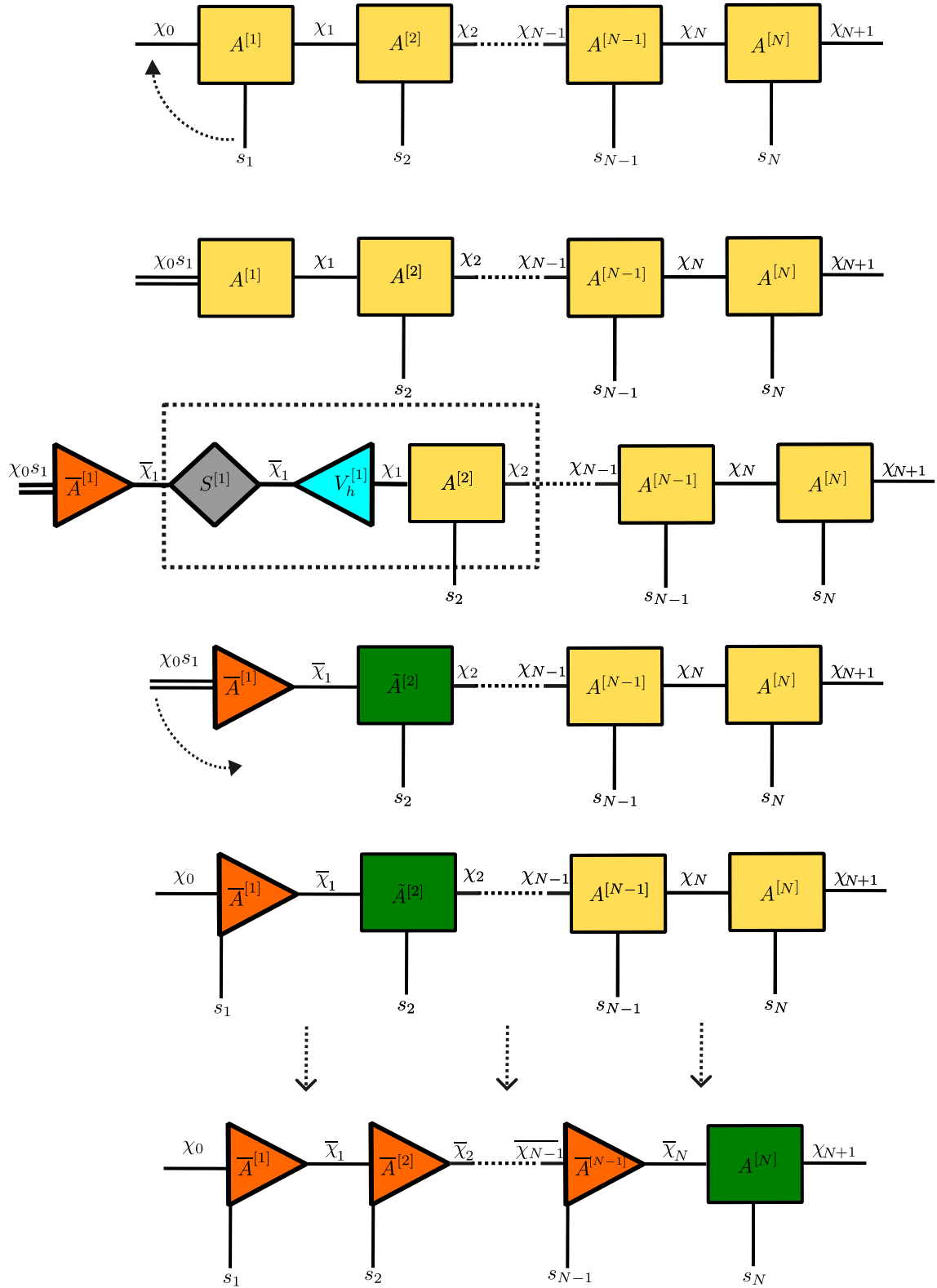


Figure 2.9: A left to right canonicalization and compression sweep of an MPS via SVD

Action by Single-Site Operators

A local operator acting non-trivially only on site i is written as:

$$\mathcal{W}_i = I_1 \otimes \cdots \otimes I_{i-1} \otimes W_i \otimes I_{i+1} \otimes \cdots \otimes I_N,$$

where $W_i \in \mathbb{C}^{d \times d}$ and I_j is the identity on site $j \neq i$.

For the matrix-product state

$$|\Psi\rangle = \sum_{\{s_k\}} A_{\chi_0 s_1 \chi_1}^{[1]} A_{\chi_1 s_2 \chi_2}^{[2]} \cdots A_{\chi_{N-1} s_N \chi_N}^{[N]} |s_1 \dots s_N\rangle,$$

with rank-3 tensors $A^{[k]}$, virtual indices $\chi_k \leq \chi$, and physical indices $s_k \in \{1, \dots, d\}$, $\mathcal{W}_i$ updates a single tensor only (also see Appendix A) :

$$A^{[i]} \longrightarrow A^{[i]} W_i,$$

while all $A^{[j]}$ for $j \neq i$ remain unchanged (Figure 3.3).

Addition and Multiplication

Addition of two MPS/MPO states is performed via a *direct sum*, obtained by block-diagonally stacking the site matrices A^i and B^i :

$$C^i = A^i \oplus B^i = \begin{pmatrix} A^i & 0 \\ 0 & B^i \end{pmatrix}, \quad \chi_i, \chi'_i \longmapsto \chi_i + \chi'_i.$$

Thus each virtual bond dimension grows from χ_i and χ'_i to their sum. In tensor-shape notation,

$$(\chi_{i-1}, s_i, \chi_i) \oplus (\chi'_{i-1}, s_i, \chi'_i) = (\chi_{i-1} + \chi'_{i-1}, s_i, \chi_i + \chi'_i).$$

For MPS with open boundary conditions $\chi_0 = \chi_N = 1$ the direct of site tensors at the ends is performed in the following way (see Figure 3.4)[7]:

$$(1, s_1, \chi_1) \oplus (1, s_i, \chi'_1) = (1, s_1, \chi_1 + \chi'_1).$$

$$(\chi_{N-1}, s_N, 1) \oplus (\chi'_{N-1}, s_N, 1) = (\chi_{N-1} + \chi'_{N-1}, s_N, 1).$$

Scaling an MPS/MPO by a complex scalar c is realised by rescaling the site tensors:

$$|\Psi\rangle \longmapsto c |\Psi\rangle \iff A_{\chi_{i-1}, s_i, \chi_i}^{[i]} \longmapsto \begin{cases} c A_{\chi_{i-1}, s_i, \chi_i}^{[i]}, & \text{for one chosen site } i, \\ c^{1/N} A_{\chi_{i-1}, s_i, \chi_i}^{[i]}, & \text{for every } i=1, \dots, N \text{ (distributed scaling)} \end{cases}$$

Here N denotes the total number of sites in the chain. The first option multiplies an arbitrary site tensor with c while the second option spreads the factor uniformly to enhance numerical stability.

2.3 Open Quantum Systems

Quantum devices never evolve in isolation but interact constantly with some surrounding environment. Formally, the joint state $\varrho(t)$ on $\mathcal{H}_S \otimes \mathcal{H}_E$ obeys the von Neumann equation

$$\frac{d\varrho}{dt} = -\frac{i}{\hbar} [H_S + H_E + H_{\text{int}}, \varrho] \quad (2.14)$$

where $H_{\text{int}} = \sum_{\alpha} S_{\alpha} \otimes B_{\alpha}$ and S_{α} acts on the system and B_{α} on the bath. Tracing out the environment,

$$\rho(t) = \text{Tr}_E[\varrho(t)],$$

yields the reduced state $\rho(t)$, which in general undergoes both coherent evolution and irreversible dissipation or decoherence.

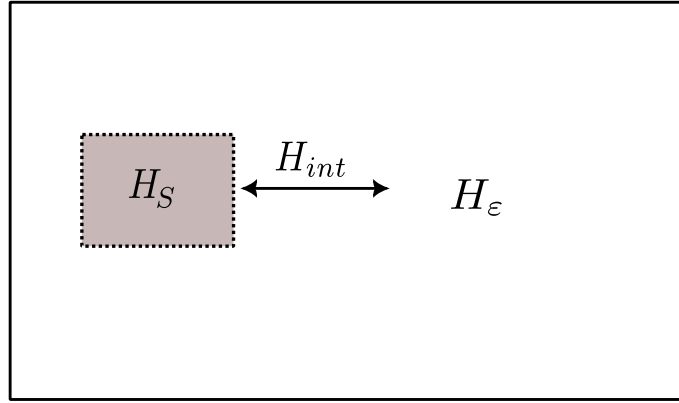


Figure 2.10: An open quantum system characterized by Hamiltonian H_S interacting with the environment ε via interaction Hamiltonian H_{int} .

When the bath “forgets” correlations much faster than the system evolves (i.e. $\tau_B \ll \tau_S$, where τ_B is the bath correlation time and τ_S the system’s timescale set by H_S), and the coupling is weak, one can make the Born–Markov and secular approximations. The result is the celebrated Gorini–Kossakowski–Sudarshan–Lindblad (GKSL) master equation[9]:

$$\frac{d\rho}{dt} = -\frac{i}{\hbar} [H_S + H_{\text{LS}}, \rho] + \sum_k \gamma_k \left(L_k \rho L_k^\dagger - \frac{1}{2} \{L_k^\dagger L_k, \rho\} \right). \quad (2.15)$$

Here H_{LS} is a small Lamb–shift correction to the system Hamiltonian, each L_k is a jump operator describing a particular decay or dephasing channel, and γ_k are positive rates. This equation generates a completely positive, trace-preserving semigroup—i.e. truly memoryless, Markovian dynamics.

Non-Markovian memory kernel and bath correlations In many realistic settings the environment retains memory on a timescale $\tau_B \gtrsim \tau_S$, or its spectral density $J(\omega)$ exhibits sharp features (narrow peaks or band edges). In such cases one cannot collapse the past into instantaneous rates; instead the reduced density matrix $\rho(t)$ obeys the exact Nakajima–Zwanzig integro-differential equation

$$\frac{d}{dt} \rho(t) = -i \mathcal{L}_S \rho(t) + \int_0^t \mathcal{K}(t - \tau) \rho(\tau) d\tau, \quad (2.16)$$

where

$$\mathcal{L}_S[X] = [H_S, X], \quad H_{\text{int}} = S \otimes B, \quad \mathcal{L}_{\text{int}}[X] = [H_{\text{int}}, X],$$

and the memory-kernel superoperator is

$$\mathcal{K}(s)[\rho] = \text{Tr}_\mathcal{E} \left\{ \mathcal{L}_{\text{int}}(s) e^{-i(\mathcal{L}_S + \mathcal{L}_\mathcal{E})s} \mathcal{L}_{\text{int}}(0) (\rho \otimes \rho_\mathcal{E}) \right\}. \quad (2.17)$$

Although formally exact, this formulation requires storing the full history $\{\rho(\tau)\}_{0 \leq \tau \leq t}$, making direct long-time integration prohibitively expensive.

Long-lived correlations arise when the spectral density

$$J(\omega) = \pi \sum_k |g_k|^2 \delta(\omega - \omega_k) \approx \frac{1}{2\pi} \frac{\Gamma \lambda^2}{(\omega - \omega_c)^2 + \lambda^2} \quad (2.18)$$

has narrow structure (with mode couplings g_k , frequencies ω_k ; and for the Lorentzian form, center ω_c , width λ , and strength Γ). The corresponding bath-correlation function

$$C_{BB}(t - \tau) = \langle B(t) B(\tau) \rangle_{\rho_\mathcal{E}} = \text{Tr}_\mathcal{E} \{ B(t) B(\tau) \rho_\mathcal{E} \} \propto e^{-(\lambda + i\omega_c)(t - \tau)}, \quad (2.19)$$

with

$$B(t) = e^{iH_\mathcal{E}t} B e^{-iH_\mathcal{E}t},$$

decays on $\tau_B \sim \lambda^{-1}$, which can rival the system timescale $\tau_S \sim \|H_S\|^{-1}$. In this non-Markovian regime one observes oscillatory populations, coherence revivals, and information backflow from environment to system.

Pseudomode reduction of the environment [10]–[12]: To bridge between exact memory-kernel dynamics and efficient simulation, one may recast the problem in a time-local form via the time-convolutionless (TCL) expansion, or—more powerfully—map structured baths onto a few discrete, damped “pseudomodes.” These pseudomodes absorb the bath’s non-Markovian character and, when appended to the system, render the combined dynamics memoryless once more. We turn next to this pseudomode reduction, which transforms the integro-differential challenge into an extended Lindblad evolution in a larger Hilbert space.

The key idea is to approximate the exact bath-correlation function

$$C_{BB}(t) = \int_0^\infty J(\omega) e^{-i\omega t} d\omega$$

by a sum of damped exponentials,

$$C_{BB}(t) \approx \sum_{j=1}^{N_{\text{PM}}} \alpha_j e^{-(\gamma_j + i\Omega_j)t}, \quad (2.20)$$

where each term corresponds to one pseudomode with frequency Ω_j , damping rate γ_j , and weight α_j . The parameters $\{\alpha_j, \gamma_j, \Omega_j\}$ are chosen—e.g. by fitting the target spectral density $J(\omega)$ via rational-function or Padé techniques—so that the right-hand side of Equation 2.20 matches the true $C_{BB}(t)$ over the time window of interest.

Each pseudomode is then represented by a local harmonic oscillator a_j coupled directly to the system operator S . The enlarged Hamiltonian becomes

$$H_{\text{tot}} = H_S + \sum_{j=1}^{N_{\text{PM}}} \left[\Omega_j a_j^\dagger a_j + g_j S (a_j + a_j^\dagger) \right], \quad (2.21)$$

where the coupling constants g_j satisfy

$$\alpha_j = g_j^2 / (2 \Omega_j). \quad (2.22)$$

Finally, to enforce the correct damping γ_j of each pseudomode, one adds Markovian Lindblad terms,

$$\frac{d\rho}{dt} = -\frac{i}{\hbar} [H_{\text{tot}}, \rho] + \sum_{j=1}^{N_{\text{PM}}} \gamma_j \mathcal{D}[a_j] \rho, \quad (2.23)$$

where $\mathcal{D}[a]\rho = a\rho a^\dagger - \frac{1}{2}\{a^\dagger a, \rho\}$. It can be shown that, upon tracing out the pseudomodes, the reduced system dynamics reproduce the original non-Markovian memory kernel up to the accuracy of the fit in Equation 2.20.

In practice, only a handful of pseudomodes ($N_{\text{PM}} \sim 2-5$) are needed to capture narrow spectral features or long-lived correlations, turning the integro-differential Nakajima–Zwanzig equation into a time-local Lindblad master equation on an enlarged—but still tractable—Hilbert space. While the simplest pseudomode mapping treats each damped mode as an independent harmonic oscillator coupled only to the system, various extensions allow for more intricate bath structures:

- **Interacting pseudomodes.** In environments with overlapping spectral features, one can introduce direct couplings between pseudomodes to capture cross-correlations [10].
- **Reaction-coordinate mapping.** Here a single collective mode (the “reaction coordinate”) is extracted from the bath and coupled to a residual Markovian reservoir; this mode can itself be decomposed into interacting pseudomodes if the residual spectrum is structured [13].
- **Chain-mapping techniques.** By tridiagonalizing the bath Hamiltonian one obtains a 1D chain of nearest-neighbor-coupled oscillators, which can be truncated to a finite interacting pseudomode network [14].
- **Hierarchical pseudomodes.** Multiple layers of pseudomodes with inter-layer couplings can systematically capture increasingly long-lived correlations [15].

Each of these variants uses the same principle—matching the bath correlation function by a network of damped oscillators—but allows one to tailor the pseudomode architecture (couplings, number of modes, damping rates) to the specific features of a given spectral density.

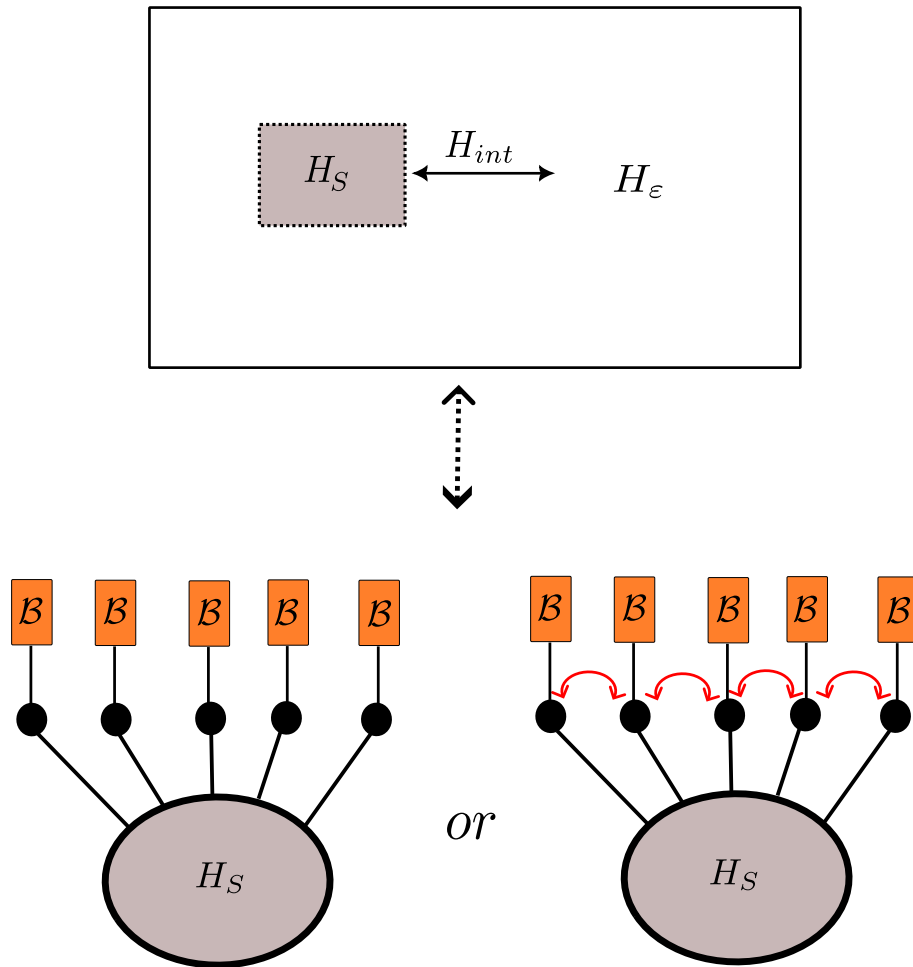


Figure 2.11: Pseudomode models simulated by DAMPF

Dissipation-Assisted Matrix-Product Factorization (dampf)

Having transformed the non-Markovian environment into a small set of discrete pseudomodes (each with its own Lindblad damping), the DAMPF protocol embeds *all* pseudomodes directly into a matrix-product-state (MPS) representation of the system [16]. Each pseudomode j is coupled to a tunable Markovian reservoir via a Lindblad operator $\sqrt{\gamma_j} a_j$, which continuously removes excitations and entanglement from the extended system. This engineered dissipation controls the bipartite entanglement entropy growth, so that the required MPS bond dimension χ remains manageable even at long times—well beyond the light-cone or entanglement barriers that limit standard TEBD/DMRG methods.

In practice, DAMPF achieves stable, long-time evolution with polynomial cost in both the number of pseudomodes N_{PM} and χ . Originally proposed in [17] and detailed in [16], DAMPF serves as the central numerical engine for all simulations in this thesis.

Limitations The efficiency of DAMPF relies on the locality of the pseudomode network: it requires that each pseudomode be either independent or interact only with its immediate neighbors. Arbitrary long-range couplings among pseudomodes can generate rapid entanglement growth across the MPS chain, undermining the controlled bond-dimension scaling that makes DAMPF feasible.

Chapter 3

Dissipation Assisted Matrix Product Factorization

3.1 Setting up the Hamiltonian

This chapter offers a self-contained exposition of DAMPF, tracing the algorithm back to its original formulation in Ref. [17]. Our attention is confined to environments that can be faithfully represented as collections of *independent* discrete oscillators, or *pseudomodes*. Although this restriction excludes the low-frequency, mode-mixing phenomena that emerge when oscillators interact, uncoupled pseudomodes already suffice to capture the essential physics of high-frequency under-damped vibrations—most notably the persistence of electronic coherences, the vibronic progressions visible in optical spectra, and the ultrafast channels for energy transfer. By focusing on non-interacting baths we gain a transparent stage on which to illuminate the core algorithmic ingredients of DAMPF: local basis truncation, bond-dimension optimisation, and the delicate balance between dissipative dynamics and tensor-network entanglement growth. The more involved treatment required for structured low-frequency spectra is therefore deferred to future work, while the material that follows concentrates on the robust and widely applicable framework built upon independent pseudomodes.

Consider a network of N monomers with total (electronic) Hamiltonian:

$$H_e = \sum_{n=1}^N \Omega_n |n\rangle \langle n| + \sum_{m \neq n} J_{m,n} |m\rangle \langle n| \quad (3.1)$$

where Ω_n is the site energy of the n^{th} monomer, and $J_{m,n}$ is the coupling energy between monomer m and n .

Now, let each monomer be coupled to Q quantum oscillators with frequency $\omega_q : q = 1, \dots, Q$ and energy levels d . Without loss of generalization, the assumption is made that each monomer is coupled to identical set of oscillators with same dimension: $\omega_{n,q} = \omega_q$ and $d_{n,q} = d$. The combined Hamiltonian of the system now is:

$$H = H_e + H_v + H_{ev} \quad (3.2)$$

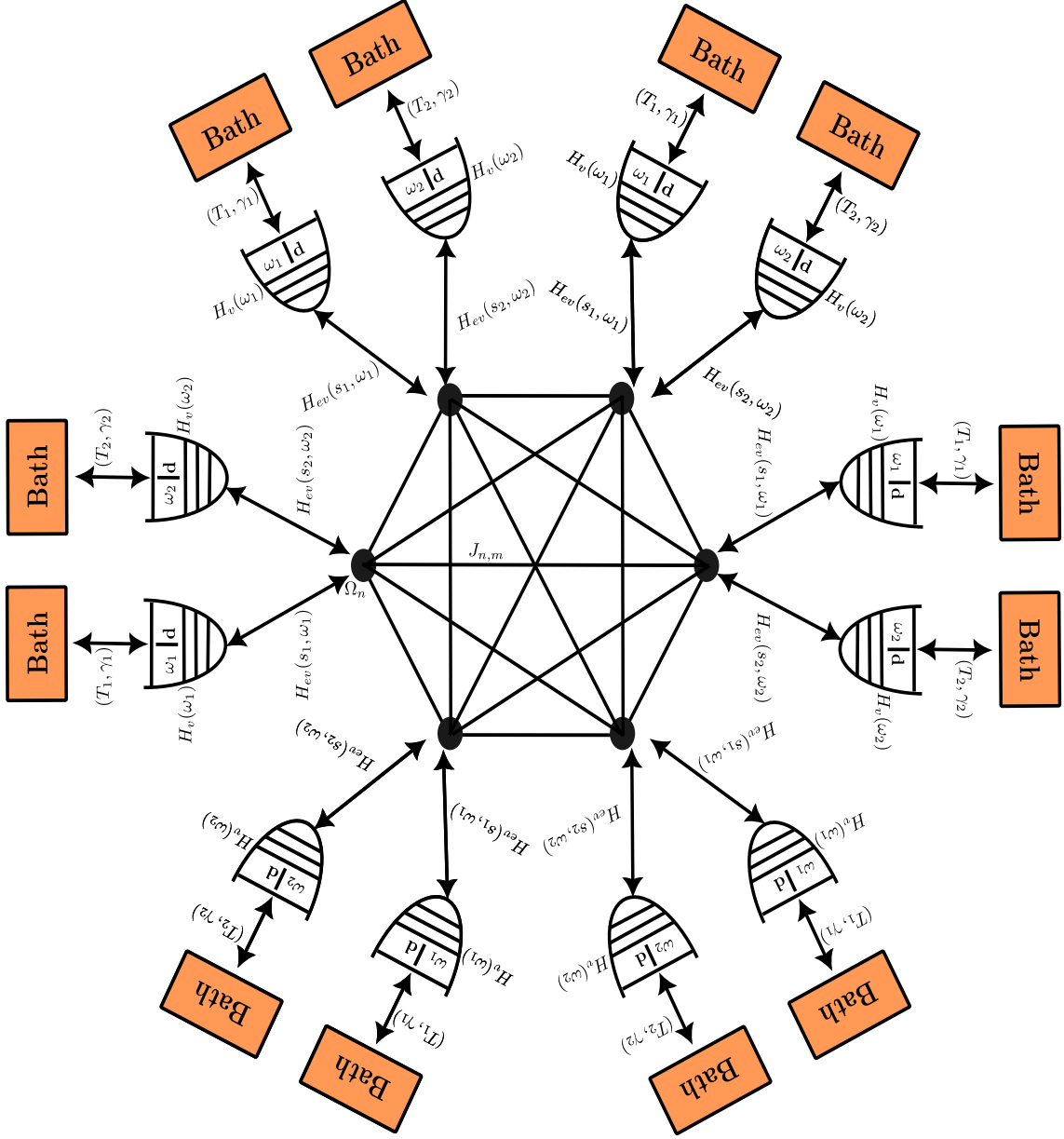


Figure 3.1: 6 Monomer sites each coupled to identical sets of two oscillators, which in turn are coupled to independent baths.

where:

$$H_v = \sum_{n=1}^N \sum_{q=1}^Q \omega_q a_{n,q}^\dagger a_{n,q} \quad (3.3)$$

and

$$H_{ev} = \sum_{n=1}^N \sum_{q=1}^Q \omega_q \sqrt{s_q} |n\rangle \langle n| (a_{n,q} + a_{n,q}^\dagger) \quad (3.4)$$

where H_v (the vibrational part) is the Hamiltonian of the oscillators and H_{ev} captures the interaction between the electronic system (chain of monomers) and the discrete oscillators. s_q is the so called *Huang-Rhys Factor*, a dimensional constant quantifying the coupling strength between the electronic system and the oscillators.

The reorganization energy λ , which measures the total cost to reconfigure the state of oscillators when the electronic system changes state, is related to s_q as:

$$\lambda = \sum_k \hbar \omega_k s_k \quad (3.5)$$

Hereafter, $\hbar$ is set to 1. Furthermore, each oscillator is coupled to an independent thermal reservoir. If ρ is the total state of the chain and oscillators, the Lindblad damping because of the thermal reservoirs is modeled as:

$$\begin{aligned} \mathcal{D}\rho = \sum_{n,q} 2\gamma_q \left[\bar{n}(\omega_q) \left(a_{n,q}^\dagger \rho a_{n,q} - \frac{1}{2} \{a_{n,q} a_{n,q}^\dagger, \rho\} \right) \right. \\ \left. + (\bar{n}(\omega_q) + 1) \left(a_{n,q} \rho a_{n,q}^\dagger - \frac{1}{2} \{a_{n,q}^\dagger a_{n,q}, \rho\} \right) \right] \end{aligned} \quad (3.6)$$

$$\text{with } \bar{n}(\omega_q) = \frac{1}{e^{\beta_q \omega_q} - 1}.$$

β_q is the inverse temperature of the bath to which q^{th} oscillator is coupled to. γ_q (damping rate) and $\bar{n}(\omega_q)$ determine the dissipation of the q^{th} oscillator. Since each monomer site is coupled to a similar set of oscillators it was implicitly assumed that $\gamma_{n,q} = \gamma_q$

Key Parameters

Symbol	Description [Unit]
ω_q	Mode frequency of the q^{th} oscillator [rad/s]
β_q	Inverse temperature of the Markovian bath [J^{-1}]
γ_q	Damping rate of an oscillator to its bath [s^{-1}]
$\bar{n}(\omega_q)$	Bose–Einstein occupation number [dimensionless]
$a_{n,q}$	Annihilation operator [—]
$a_{n,q}^\dagger$	Creation operator [—]
S_q	Huang–Rhys factor (coupling strength between monomer and oscillator) [—]
Ω_n	Site energy of the n^{th} monomer
$J_{n,m}$	Electronic coupling strength between monomers n and m

The state of the system, from now on called the *vibronic state*, can formally be written as:

$$\rho = \sum_{n,m=1}^N |m\rangle \langle n| \otimes \mathcal{O}_{m,n} \quad (3.7)$$

where $\mathcal{O}_{m,n}$ describes the vibrational degrees of freedom (oscillators + coupled Markovian baths) conditional to the electronic operator $|m\rangle \langle n|$:

$$\mathcal{O}_{m,n} = \sum_{j_1, \dots, j_M}^{d^2} C_{j_1, \dots, j_M}^{(m,n)} x_{j_1} \otimes \dots \otimes x_{j_M} \quad (3.8)$$

where $M = NQ$, d is the number of energy levels (*truncation number*) of the oscillators, and $\{x_i\}_{i=1}^{d^2}$ denotes an orthonormal basis with respect to the Hilbert-Schmidt inner product. In the standard basis $\{|i\rangle\}_{i=1}^N$ of electronic sites, the total state can be visualized as:

$$\rho = \begin{bmatrix} \mathcal{O}_{1,1} & \cdots & \mathcal{O}_{1,N} \\ \vdots & \ddots & \vdots \\ \mathcal{O}_{N,1} & \cdots & \mathcal{O}_{N,N} \end{bmatrix} \quad (3.9)$$

3.2 Tensor Network Representation

The central idea of DAMPF is to describe (3.8) by matrix product operators (MPOs) of bond dimension χ . To this end, the coefficients $C_{j_1, \dots, j_M}^{(m,n)}$ in $\mathcal{O}_{m,n}$ are encoded by a set of matrices:

$$C_{j_1, \dots, j_M}^{(m,n)} = A_{1,j_1}^{(m,n)} A_{2,j_2}^{(m,n)}, \dots, A_{M,j_M}^{(m,n)} \quad (3.10)$$

where $A_{i,j_i}^{(m,n)}$ is of size $\chi_i \times \chi_{i+1}$ with $\chi_1 = \chi_{M+1} = 1$ and $j_i \in \{1, \dots, d^2\}$. This is actually an MPS with χ as the bond dimension and j_i as physical legs. It is immediately noted that while the number of coefficients required to fully describe $\mathcal{O}_{m,n}$ in (3.8) is d^{2M} , in the tensor network representation this becomes $Md^2\chi^2$. Formally, (3.8) can now be written as:

$$\mathcal{O}_{m,n} = \sum_{j_1, \dots, j_M}^{d^2} A_{1,j_1}^{(m,n)} A_{2,j_2}^{(m,n)}, \dots, A_{M,j_M}^{(m,n)} x_{j_1} \otimes \dots \otimes x_{j_M} \quad (3.11)$$

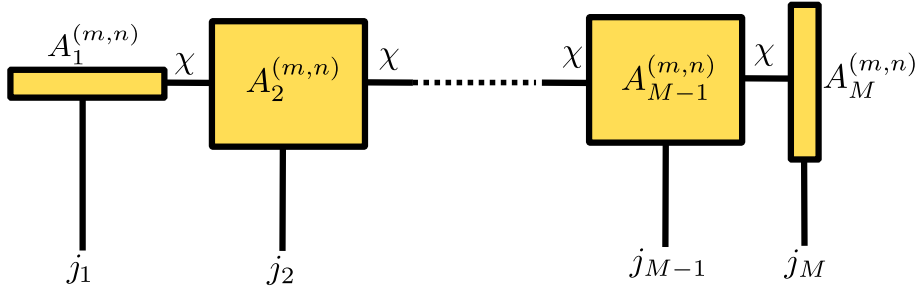


Figure 3.2: $\mathcal{O}_{m,n}$ as a Tensor Network of bond dimension χ

3.3 Dynamics

The dynamics of $\rho(t)$ is governed by the master equation:

$$\frac{d}{dt}\rho(t) = -i[H, \rho(t)] + \mathcal{D}\rho(t), \quad (3.12)$$

and can be recast with the time-independent Liouvillian $\mathcal{V} = -i[H, \cdot] + \mathcal{D}$ as

$$\frac{d}{dt}\rho(t) = \mathcal{V}\rho(t), \quad \rho(t) = e^{\mathcal{V}(t-t_0)}\rho(t_0), \quad (3.13)$$

For a small time step Δt the propagator is approximated with the first-order Suzuki–Trotter splitting:

$$e^{\mathcal{V}\Delta t} = \mathbf{U}_e \mathbf{U}_v \mathbf{U}_{ev} e^{\mathcal{D}\Delta t} + \mathcal{O}(\Delta t^2), \quad (3.14)$$

where $\mathbf{U}_x = \exp[-i[H_x, \cdot]\Delta t]$ such that $\mathbf{U}_x \rho(t) = \mathcal{U}_x(t + \Delta t) \rho(t) \mathcal{U}_x^\dagger(t + \Delta t)$ and $\mathcal{U}_x(\Delta t) = \exp(-iH_x \Delta t)$.

Because the factors $\mathbf{U}_v$, $\mathbf{U}_{ev}$ and $\exp(\mathcal{D}\Delta t)$ are tensor products of *single-oscillator* super-operators, they cannot increase the bond dimension of a matrix-product representation. Their common action can be summarized as below.

3.3.1 Vibrational, Electronic-Vibrational, and Damping Dynamics

Site index and notation. With N monomers and Q oscillators per monomer we have $M = NQ$ oscillator sites. The pair (n, q) ($n = 1, \dots, N$; $q = 1, \dots, Q$) is mapped to a single chain index

$$i = (q - 1)N + n. \quad (3.15)$$

For every electronic matrix element $\langle m | \rho | n \rangle$ we denote the corresponding MPO tensors by $A_{i,j}^{(m,n)}$, where $j = 1, \dots, d^2$ labels the local operator basis $\{x_j\}$.

Universal local update rule. Each single-oscillator map $x \in \{v, ev, D\}$ replaces

$$A_{i,j}^{(m,n)}(t + \Delta t) = \sum_{k=1}^{d^2} W_{x;i,j,k}^{(m,n)} A_{i,k}^{(m,n)}(t). \quad (3.16)$$

Because only the physical index is touched, the MPO bond dimension is unchanged; all sites can be updated independently and in parallel (see A). The rank-3 tensor W_x depends only on local oscillator parameters ω_q , γ_q and s_q .

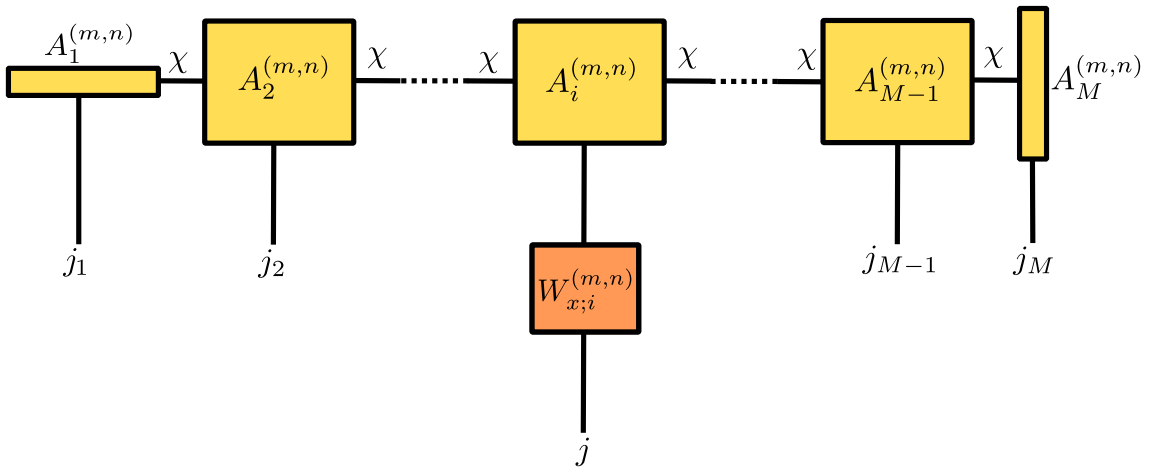


Figure 3.3: Local update of $A_{i,j}^{(m,n)}$.

1. Pure vibrational rotation U_v

For oscillator (n, q) define the unitary

$$U_v^{(n,q)} = \exp[-i \omega_q a_{n,q}^\dagger a_{n,q} \Delta t]. \quad (3.17)$$

Its action yields a site-independent matrix

$$\mathcal{R}_{q;j,k}^{(v)} = \text{Tr}(x_j^\dagger U_v^{(n,q)} x_k U_v^{(n,q)\dagger}), \quad (3.18)$$

and Eq. (3.16) is realised with $W_{v;i,j,k}^{(m,n)} = \mathcal{R}_{q;j,k}^{(v)}$ for the index $i \equiv (n, q)$.

2. Electron–vibration coupling U_{ev}

The model employs a **Holstein coupling**, i.e. an *on-site* electron–phonon interaction in which each electronic excitation displaces only its *own* local vibrational coordinate. The coupling Hamiltonian reads

$$H_{ev} = \sum_{n=1}^N \sum_{q=1}^Q \omega_{n,q} \sqrt{s_{n,q}} |n\rangle \langle n| (a_{n,q} + a_{n,q}^\dagger),$$

with Huang–Rhys factors $s_{n,q}$. For a Suzuki–Trotter time step Δt the single-oscillator propagator attached to monomer n and mode q is

$$U_{ev}^{(n,q)} = \exp[-i \Delta t \omega_{n,q} \sqrt{s_{n,q}} (a_{n,q} + a_{n,q}^\dagger)].$$

Because the projector $|n\rangle \langle n|$ is diagonal in the electronic basis, the action of $U_{ev}^{(n,q)}$ depends on whether the electronic block is a *population* ($m = n$) or a *coherence* ($m \neq n$). Written in the local operator basis $\{x_\sigma\}_{\sigma=1}^{d^2}$ introduced in Eq. (3.28) this gives three $d^2 \times d^2$ coefficient matrices:

$$[C_{n,q}^{\text{pop}}]_{\sigma\sigma'} = \text{Tr}(x_\sigma^\dagger U_{ev}^{(n,q)} x_{\sigma'} U_{ev}^{(n,q)\dagger}), \quad (\text{population block, } m = n), \quad (3.19)$$

$$[C_{m,q}^{\text{L}}]_{\sigma\sigma'} = \text{Tr}(x_\sigma^\dagger U_{ev}^{(m,q)} x_{\sigma'}), \quad (\text{left oscillator of a coherence, } m \neq n), \quad (3.20)$$

$$[C_{n,q}^{\text{R}}]_{\sigma\sigma'} = \text{Tr}(x_\sigma^\dagger x_{\sigma'} U_{ev}^{(n,q)\dagger}), \quad (\text{right oscillator of a coherence, } m \neq n). \quad (3.21)$$

In terms of W matrices in (3.16).

$$W_{ev;k}^{(m,n)} = \begin{cases} C_{n,q}^{\text{pop}}, & m = n, \\ C_{m,q}^{\text{L}}, & m \neq n, k \in \text{monomer } m, \\ C_{n,q}^{\text{R}}, & m \neq n, k \in \text{monomer } n, \\ I, & \text{otherwise,} \end{cases} \quad (3.22)$$

3. Local Lindblad damping $\exp(\mathcal{D}\Delta t)$

For each oscillator we introduce the completely positive map

$$\Lambda_q = \exp(\mathcal{D}_q \Delta t), \quad (3.23)$$

where $\mathcal{D}_q$ is the single-mode dissipator specified in Equation 3.6. Its Choi matrix in the basis $\{x_j\}$ is

$$\mathcal{B}_{q;j,k} = \text{Tr}(x_j^\dagger \Lambda_q(x_k)), \quad (3.24)$$

and we set $W_{D;i,j,k}^{(m,n)} = \mathcal{B}_{q;j,k}$ for $i = (q-1)N + n$. These coefficients are identical for all (m, n) .

3.3.2 Electronic Evolution

The remaining Suzuki–Trotter factor, $\mathbf{U}_e = \exp[-i[H_e, \cdot]\Delta t]$, acts *exclusively* on the electronic subspace and is therefore non-local with respect to the oscillator chain. Let $[U_e]_{m,n} = \langle m|U_e|n\rangle$ denote its matrix elements in the site basis $\{|n\rangle\}_{n=1}^N$. Using Eqs. (3.7)–(3.9), one obtains

$$\mathbf{U}_e \rho = \sum_{m,n=1}^N |m\rangle\langle n| \otimes \sum_{k,l=1}^N [U_e]_{m,k} [U_e^\dagger]_{l,n} \mathcal{O}_{k,l}, \quad (3.25)$$

Hence the updated oscillator block $\tilde{\mathcal{O}}_{m,n}$ reads

$$\tilde{\mathcal{O}}_{m,n} = \sum_{k,l=1}^N c_{k,l}^{(m,n)} \mathcal{O}_{k,l}, \quad c_{k,l}^{(m,n)} = [U_e]_{m,k} [U_e^\dagger]_{l,n}. \quad (3.26)$$

Let $A_{i,j}^{(k,l)}$ be the MPO tensors representing $\mathcal{O}_{k,l}$ according to Eq. (3.11). Because Eq. (3.26) is a *sum of MPOs*, the most straight-forward representation of $\tilde{\mathcal{O}}_{m,n}$ is obtained by the direct sum of the individual matrices,

$$\tilde{A}_{i,j}^{(m,n)} = \bigoplus_{k,l=1}^N c_{k,l}^{(m,n)} A_{i,j}^{(k,l)}. \quad (3.27)$$

While Eq. (3.27) is formally exact, it increases the bond dimension from χ to at most $(N^2 - 1)\chi$. To keep the virtual dimension within tractable limits, a subsequent MPO compression (e.g. truncated SVD) must be applied after a suitable (from a memory point of view) number of direct sums.

3.4 Reduced Electronic State

To obtain the reduced electronic state $\rho_S = \text{Tr}_v(\rho)$ we choose the single-oscillator operator basis $\{x_i\}_{i=1}^{d^2}$ such that

$$x_1 = d^{-1/2}\mathbb{I}, \quad \text{Tr}(x_i) = 0 \text{ for } i > 1, \quad \text{Tr}(x_i^\dagger x_j) = \delta_{ij}. \quad (3.28)$$

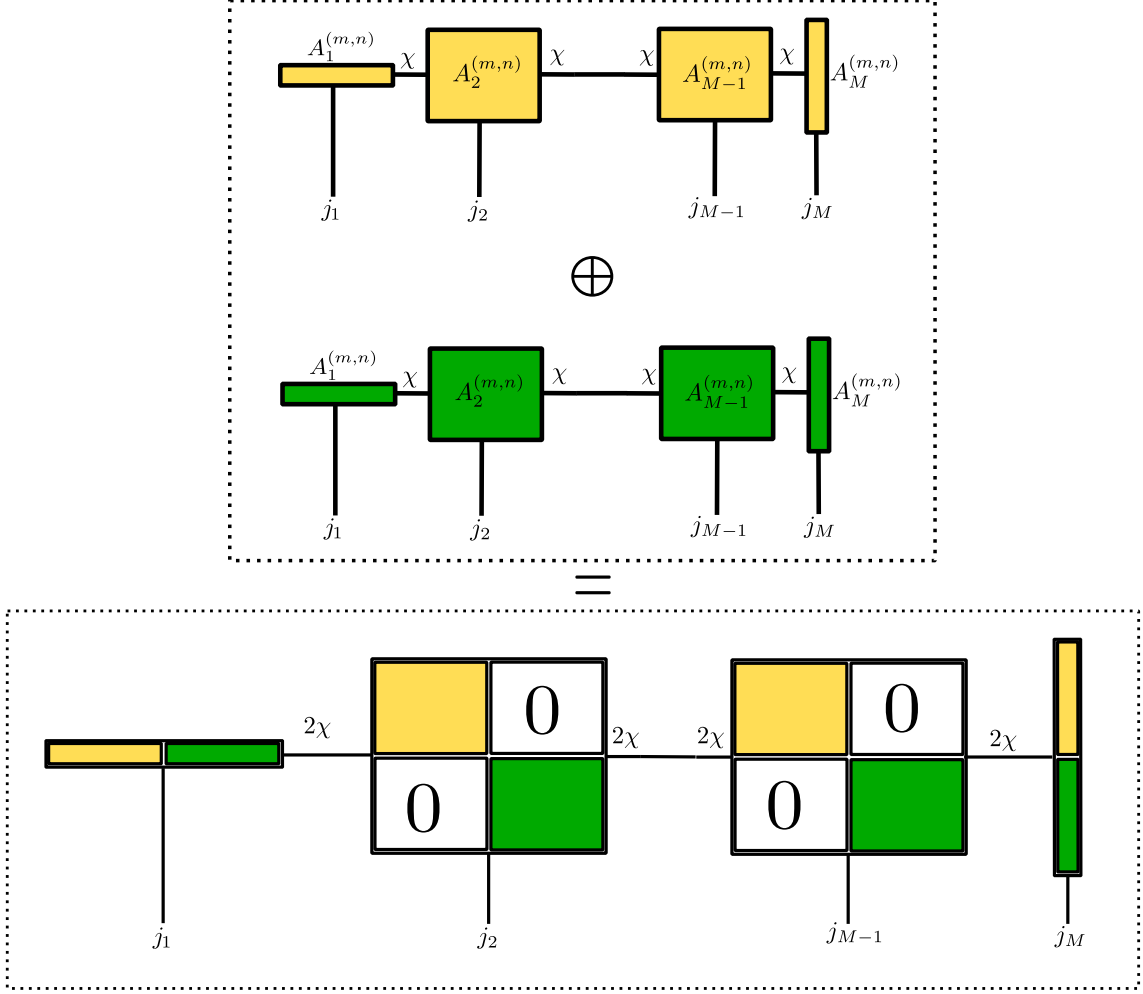


Figure 3.4: Direct Sum of Two MPOs. Bond dimension increases.

Equation (3.28) guarantees Hilbert–Schmidt orthonormality while separating the identity from the traceless sector. Inserting the MPO representation (3.11) into the vibronic state Equation 3.7 one finds

$$\rho = \sum_{m,n=1}^N |m\rangle\langle n| \sum_{j_1, \dots, j_M=1}^{d^2} \left[\prod_{i=1}^M A_{i,j_i}^{(m,n)} \right] x_{j_1} \otimes \cdots \otimes x_{j_M}. \quad (3.29)$$

The partial trace over the vibrational Hilbert space acts term–wise,

$$\text{Tr}_v(x_{j_1} \otimes \cdots \otimes x_{j_M}) = \prod_{i=1}^M \text{Tr}(x_{j_i}).$$

Because $\text{Tr}(x_i) = 0$ for every $i > 1$, the product vanishes unless $j_1 = j_2 = \cdots = j_M = 1$; in that unique case $\text{Tr}(x_1) = d^{1/2}$ by (3.28). Hence only the *first* physical index of each local tensor contributes, and (3.29) collapses to

$$\rho_S = \text{Tr}_v(\rho) = d^{M/2} \sum_{m,n=1}^N \left[\prod_{i=1}^M A_{i,1}^{(m,n)} \right] |m\rangle\langle n|, \quad (3.30)$$

The prefactor $d^{M/2} = (\text{Tr } x_1)^M$ originates from the normalisation of the identity operator in (3.28) and restores the correct electronic trace, $\text{Tr}_S(\rho_S) = 1$.

Equation (3.30) shows that the reduced electronic density matrix can be evaluated *without* contracting the full oscillator network. For each electronic block (m, n) one has to contract only the M MPO tensors whose physical index equals 1, $\{A_{i,1}^{(m,n)}\}_{i=1}^M$. Each $A_{i,1}^{(m,n)}$ is a $\chi_i \times \chi_{i+1}$ matrix, with boundary dimensions $\chi_1 = \chi_{M+1} = 1$; their ordered product is thus a chain of matrix multiplications

$$\prod_{i=1}^M A_{i,1}^{(m,n)} = A_{1,1}^{(m,n)} A_{2,1}^{(m,n)} \cdots A_{M,1}^{(m,n)},$$

whose result is a single complex number multiplying $|m\rangle\langle n|$.

3.5 The Algorithm in a Nutshell

3.5.1 Initialization

Initial State

DAMPF assumes the initial state to be *uncorrelated* between the electronic system and its vibrational environment,

$$\rho(t_0) = \rho_e(t_0) \otimes \rho_v, \quad \rho_e(t_0) = \sum_{m,n=1}^N \rho_e(m, n) |m\rangle\langle n|,$$

while every oscillator k starts in an independent thermal state $\rho_{\beta_k} = e^{-\beta_k \omega_k a_k^\dagger a_k} / Z_k$. Writing ρ_{β_k} in the operator basis $\{x_{\sigma_k}\}_{\sigma_k=1}^{d_k^2}$ of Eq. (3.28) yields the coefficient vector $v_k(\sigma_k) = \text{Tr}(\rho_{\beta_k} x_{\sigma_k})$. Because the global state is a *tensor product*, only the tensor element with virtual indices $(1, 1)$ is populated initially,

$$\kappa(m, n, k, \sigma_k) = \rho_e(m, n)^{1/M} v_k(\sigma_k), \quad M = NQ,$$

Algorithm 2 translates this factorised structure into a set of N^2 matrix-product operators $\mathcal{O}_{m,n} \equiv \{A_k^{(m,n)}\}_{k=1}^M$, one for every electronic matrix element. Each local tensor $A_k^{(m,n)}$ is allocated with the minimal bond dimension compatible with its position in the chain: $(1, d_k^2, \chi)$ at the left boundary, (χ, d_k^2, χ) in the bulk, and $(\chi, d_k^2, 1)$ at the right boundary. All entries are set to zero except for the slice $(1, \sigma_k, 1)$, which is filled with the corresponding coefficient $\kappa(m, n, k, \sigma_k)$. χ could be set to 1 while initializing the state.

Algorithm 2 Initialize vibronic state $\mathcal{O}$ (Algorithm 1)

Require: $N, Q, d, \{\omega_k\}, \{\beta_k\}, \rho_e, \chi$

```

1:  $M \leftarrow NQ$  ▷ total number of oscillators
2:  $\mathcal{O} \leftarrow \text{array}(N, N)$  ▷  $N^2$  MPO blocks
3: for  $m \leftarrow 1$  to  $N$  do ▷ electronic row index
4:   for  $n \leftarrow 1$  to  $N$  do ▷ electronic column index
5:      $\mathcal{O}_{m,n} \leftarrow \text{array}(M)$ 
6:     for  $k \leftarrow 1$  to  $M$  do ▷ oscillator index
7:        $(q, n_{\text{site}}) \leftarrow \lfloor (k-1)/N \rfloor, k - Nq$ 
8:        $d_k \leftarrow d_{n_{\text{site}}, q}, \kappa \leftarrow \rho_e(m, n)^{1/M} v_k$ 
9:       if  $k = 1$  then
10:         $A \leftarrow \text{zeros}(1, d_k^2, \chi)$ 
11:       else if  $k = M$  then
12:         $A \leftarrow \text{zeros}(\chi, d_k^2, 1)$ 
13:       else
14:         $A \leftarrow \text{zeros}(\chi, d_k^2, \chi)$ 
15:       end if
16:       for  $\sigma \leftarrow 1$  to  $d_k^2$  do
17:         $A[1, \sigma, 1] \leftarrow \kappa(\sigma)$ 
18:       end for
19:        $\mathcal{O}_{m,n}[k] \leftarrow A$ 
20:     end for
21:      $\mathcal{O}[m, n] \leftarrow \mathcal{O}_{m,n}$ 
22:   end for
23: end for
24: return  $\mathcal{O}$ 
    
```

Precomputing Damping Matrices

The local Lindblad term for oscillator k is a time-independent super-operator

$$\mathcal{D}_k[\cdot] = \gamma_k \left(\bar{n}_k \left[a_k^\dagger(\cdot) a_k - \frac{1}{2} \{ a_k a_k^\dagger, \cdot \} \right] + (\bar{n}_k + 1) \left[a_k(\cdot) a_k^\dagger - \frac{1}{2} \{ a_k^\dagger a_k, \cdot \} \right] \right)$$

where $\bar{n}_k = \frac{1}{e^{\omega_k/(k_B T_k)} - 1}$. In the single-oscillator operator basis $\{x_j\}_{j=1}^{d^2}$ introduced in Eq. (3.28) we define the $d^2 \times d^2$ matrix representation $\Lambda_k = [\Lambda_k]_{j\ell} = \text{Tr}(x_j^\dagger \mathcal{D}_k[x_\ell])$. The short-time damping map that appears in the Suzuki–Trotter step is

$$\mathcal{B}_k = \exp(\Lambda_k \Delta t),$$

Because $\mathcal{D}_k$ depends only on the local parameters $(\omega_k, \gamma_k, T_k)$ and on the oscillator truncation d , all $\mathcal{B}_k$ matrices can be pre-computed once and reused throughout the simulation. The procedure is summarised in Algorithm 3.

Algorithm 3 Pre-compute damping matrices $\{\mathcal{B}_k\}$

Require: $\Delta t, N, Q, d, \{\omega_k\}, \{\gamma_k\}, \{T_k\}$

- 1: $M \leftarrow NQ$ $\triangleright$ total number of oscillators
- 2: $\mathcal{B} \leftarrow \text{array}(M)$
- 3: **for** $k \leftarrow 1$ **to** M **do**
- 4: build truncated annihilation operator $(a_k, a_k^\dagger)$
- 5: $\bar{n}_k \leftarrow \begin{cases} 1/(e^{\omega_k/(k_B T_k)} - 1), & T_k > 0, \\ 0, & T_k = 0 \end{cases}$
- 6: **for** $j \leftarrow 1$ **to** d^2 **do**
- 7: **for** $\ell \leftarrow 1$ **to** d^2 **do**
- 8: $[\Lambda_k]_{j\ell} \leftarrow \text{Tr}(x_j^\dagger \mathcal{D}_k[x_\ell])$
- 9: **end for**
- 10: **end for**
- 11: $\mathcal{B}_k \leftarrow \exp(\Lambda_k \Delta t)$
- 12: store $\mathcal{B}_k$ in $\mathcal{B}[k]$
- 13: **end for**
- 14: **return** $\mathcal{B}$ $\triangleright$ shape (N, Q, d^2, d^2)

Each matrix $\mathcal{B}_k$ enters the local update rule (3.16) whenever the damping map $\exp(\mathcal{D} \Delta t)$ is applied during time evolution.

Precomputing Vibrational Dynamics Matrices

Each bosonic mode evolves coherently under the quadratic Hamiltonian $H_v^{(n,q)} = \omega_{n,q} a_{n,q}^\dagger a_{n,q}$. For a discrete time step Δt the corresponding unitary is

$$U_v^{(n,q)} = \exp[-i \Delta t \omega_{n,q} a_{n,q}^\dagger a_{n,q}].$$

Represented in the single-site operator basis $\{x_j\}_{j=1}^{d^2}$, the action of this rotation is captured by the matrix

$$[\mathcal{R}_{n,q}]_{j\ell} = \text{Tr}(x_j^\dagger U_v^{(n,q)} x_\ell U_v^{(n,q)\dagger}), \quad j, \ell = 1, \dots, d^2.$$

The $d^2 \times d^2$ matrices $\mathcal{R}_{n,q}$ are re-used in every Suzuki–Trotter sweep (see the update rule (3.16)) and are therefore pre-computed once.

Algorithm 4 Pre-compute rotation matrices $\{\mathcal{R}_{n,q}\}$

Require: $\Delta t, N, Q, \{\omega_{n,q}\}, \{d_{n,q}\}$

```

1:  $\mathcal{R} \leftarrow \text{array}(N, Q)$ 
2: for  $n \leftarrow 1$  to  $N$  do ▷ monomer index
3:   for  $q \leftarrow 1$  to  $Q$  do ▷ oscillator index
4:      $d \leftarrow d_{n,q}$ ; build  $(a, a^\dagger)$  of size  $d$ 
5:      $U_v \leftarrow \exp[-i \Delta t \omega_{n,q} a^\dagger a]$ 
6:     create zero matrix  $R \in \mathbb{C}^{d^2 \times d^2}$ 
7:     for  $j \leftarrow 1$  to  $d^2$  do
8:       for  $\ell \leftarrow 1$  to  $d^2$  do
9:          $X \leftarrow U_v x_\ell U_v^\dagger$ 
10:         $R_{j\ell} \leftarrow \text{Tr}(x_j^\dagger X)$ 
11:      end for
12:    end for
13:     $\mathcal{R}[n, q] \leftarrow R$ 
14:  end for
15: end for
16: return  $\mathcal{R}$  ▷ shape  $(N, Q, d^2, d^2)$ 

```

The array $\mathcal{R}$ provides the coefficients $W_{v;i,j,k}$ in Eq. (3.16) whenever the vibrational map $\mathbf{U}_v$ is applied.

Precomputing Electron-Vibrational Coupling Matrices

The short-time propagator for the Holstein-type coupling $H_{ev}^{(n,q)} = \omega_{n,q} \sqrt{s_{n,q}} (a_{n,q} + a_{n,q}^\dagger) |n\rangle\langle n|$ is

$$U_{ev}^{(n,q)} = \exp\left[-i \Delta t \omega_{n,q} \sqrt{s_{n,q}} (a_{n,q} + a_{n,q}^\dagger)\right].$$

Because H_{ev} is diagonal in the electronic basis, three distinct single-oscillator maps are needed (see the update rule Equation 3.22):

- **Population block** ($m = n$): $x \mapsto U_{ev} x U_{ev}^\dagger$
- **Coherence, left oscillator** ($m \neq n$ and site belongs to m): $x \mapsto U_{ev} x$
- **Coherence, right oscillator** ($m \neq n$ and site belongs to n): $x \mapsto x U_{ev}^\dagger$

Represented in the local operator basis $\{x_j\}_{j=1}^{d^2}$, these maps are encoded by three sets of $d^2 \times d^2$ matrices

$$\begin{aligned}
 [\mathcal{C}_{n,q}^{\text{pop}}]_{j\ell} &= \text{Tr}(x_j^\dagger U_{ev}^{(n,q)} x_\ell U_{ev}^{(n,q)\dagger}), \\
 [\mathcal{C}_{n,q}^{\text{L}}]_{j\ell} &= \text{Tr}(x_j^\dagger U_{ev}^{(n,q)} x_\ell), \quad [\mathcal{C}_{n,q}^{\text{R}}]_{j\ell} = \text{Tr}(x_j^\dagger x_\ell U_{ev}^{(n,q)\dagger}).
 \end{aligned}$$

Once computed, the collection $\{\mathcal{C}_{n,q}^{\text{pop}}, \mathcal{C}_{n,q}^{\text{L}}, \mathcal{C}_{n,q}^{\text{R}}\}$ is reused in every Suzuki–Trotter sweep.

Algorithm 5 Pre-compute e–v coupling matrices $\{\mathcal{C}_{n,q}^{\text{pop}}, \mathcal{C}_{n,q}^{\text{L}}, \mathcal{C}_{n,q}^{\text{R}}\}$

Require: $\Delta t, N, Q, \{\omega_{n,q}\}, \{s_{n,q}\}, \{d_{n,q}\}$

- 1: $\mathcal{C}^{\text{pop}}, \mathcal{C}^{\text{L}}, \mathcal{C}^{\text{R}} \leftarrow \text{array}(N, Q)$
- 2: **for** $n \leftarrow 1$ **to** N **do** ▷ monomer index
- 3: **for** $q \leftarrow 1$ **to** Q **do** ▷ oscillator index
- 4: $d \leftarrow d_{n,q}$, build $(a, a^\dagger) \in \mathbb{C}^{d \times d}$
- 5: $U_{ev} \leftarrow \exp[-i \Delta t \omega_{n,q} \sqrt{s_{n,q}} (a + a^\dagger)]$
- 6: initialise zero matrices $C^{\text{pop}}, C^{\text{L}}, C^{\text{R}} \in \mathbb{C}^{d^2 \times d^2}$
- 7: **for** $j \leftarrow 1$ **to** d^2 **do**
- 8: **for** $\ell \leftarrow 1$ **to** d^2 **do**
- 9: $X_{\text{pop}} \leftarrow U_{ev} x_\ell U_{ev}^\dagger$
- 10: $X_{\text{L}} \leftarrow U_{ev} x_\ell$
- 11: $X_{\text{R}} \leftarrow x_\ell U_{ev}^\dagger$
- 12: $C_{j\ell}^{\text{pop}} \leftarrow \text{Tr}(x_j^\dagger X_{\text{pop}})$
- 13: $C_{j\ell}^{\text{L}} \leftarrow \text{Tr}(x_j^\dagger X_{\text{L}})$
- 14: $C_{j\ell}^{\text{R}} \leftarrow \text{Tr}(x_j^\dagger X_{\text{R}})$
- 15: **end for**
- 16: **end for**
- 17: store $\mathcal{C}_{n,q}^{\text{pop}} \leftarrow C^{\text{pop}}, \mathcal{C}_{n,q}^{\text{L}} \leftarrow C^{\text{L}}, \mathcal{C}_{n,q}^{\text{R}} \leftarrow C^{\text{R}}$
- 18: **end for**
- 19: **end for**
- 20: **return** $(\mathcal{C}^{\text{pop}}, \mathcal{C}^{\text{L}}, \mathcal{C}^{\text{R}})$ ▷ shape (N, Q, d^2, d^2) each

During time evolution the matrices $\mathcal{C}_{n,q}^{\text{pop}}, \mathcal{C}_{n,q}^{\text{L}}, \mathcal{C}_{n,q}^{\text{R}}$ supply the coefficients $W_{ev;i,j,k}^{(m,n)}$ in Eq. (3.16), according to whether the oscillator belongs to an electronic population block or to the left/right part of a coherence block.

3.5.2 Evolution

Applying Damping Matrices

Each oscillator carries a local Lindblad map $\mathcal{B}_{n,q} \in \mathbb{C}^{d^2 \times d^2}$ pre-computed by Algorithm 3. During a Suzuki–Trotter sweep the map is applied to the physical leg of every local tensor,

$$\tilde{A}_{i,\sigma,j}^{(m,n)} = \sum_{k=1}^{d^2} [\mathcal{B}_{n,q}]_{\sigma k} A_{i,k,j}^{(m,n)},$$

where i and j are the left and right virtual indices, the composite oscillator index is $k = (q-1)N + n$, and (m, n) labels the electronic block.

Algorithm 6 Local damping step $\mathcal{B}$ **Require:** vibronic MPO array $\mathcal{O}[N, N]$;

```

1: damping matrices  $\mathcal{B}[N, Q]$ 
2:  $\tilde{\mathcal{O}} \leftarrow \text{array}(N, N)$ 
3: for  $m \leftarrow 1$  to  $N$  do
4:   for  $n \leftarrow 1$  to  $N$  do
5:     for  $n_{\text{site}} \leftarrow 1$  to  $N$  do
6:       for  $q \leftarrow 1$  to  $Q$  do
7:          $A \leftarrow \mathcal{O}_{m,n}[n_{\text{site}}, q]$ 
8:          $B \leftarrow \mathcal{B}_{n_{\text{site}}, q}$   $\triangleright d^2 \times d^2$ 
9:          $\tilde{A}_{i,\sigma,j} \leftarrow \sum_k B_{\sigma k} A_{i,k,j}$ 
10:         $\tilde{\mathcal{O}}_{m,n}[n_{\text{site}}, q] \leftarrow \tilde{A}$ 
11:      end for
12:    end for
13:  end for
14: end for
15: return  $\tilde{\mathcal{O}}$ 

```

Applying vibrational Coupling

The coherent vibrational propagator $\mathbf{U}_v$ acts locally on every oscillator via the matrices $\mathcal{R}_{n,q} \in \mathbb{C}^{d^2 \times d^2}$ pre-computed by Algorithm 4. For a local MPO tensor $A_{i,\sigma,j}^{(m,n)}$ the update rule is

$$\tilde{A}_{i,\sigma,j}^{(m,n)} = \sum_{k=1}^{d^2} [\mathcal{R}_{n,q}]_{\sigma k} A_{i,k,j}^{(m,n)}, \quad (*)$$

where the composite oscillator index is $k = (q-1)N + n$. Operation $*$ modifies only the physical leg; the bond dimension is unchanged.

Algorithm 7 Local vibrational update $\mathcal{R}$ **Require:** vibronic MPO array $\mathcal{O}[N, N]$;

```

1: rotation matrices  $\mathcal{R}[N, Q]$ 
2:  $\tilde{\mathcal{O}} \leftarrow \text{array}(N, N)$ 
3: for  $m \leftarrow 1$  to  $N$  do
4:   for  $n \leftarrow 1$  to  $N$  do
5:     for  $n_{\text{site}} \leftarrow 1$  to  $N$  do
6:       for  $q \leftarrow 1$  to  $Q$  do
7:          $A \leftarrow \mathcal{O}_{m,n}[n_{\text{site}}, q]$   $\triangleright (\chi, d^2, \chi)$ 
8:          $R \leftarrow \mathcal{R}_{n_{\text{site}}, q}$   $\triangleright (d^2, d^2)$ 
9:          $\tilde{A}_{i,\sigma,j} \leftarrow \sum_k R_{\sigma k} A_{i,k,j}$ 
10:         $\tilde{\mathcal{O}}_{m,n}[n_{\text{site}}, q] \leftarrow \tilde{A}$ 
11:      end for
12:    end for
13:  end for
14: end for
15: return  $\tilde{\mathcal{O}}$ 

```

Applying EV coupling Matrices

During the Suzuki–Trotter sweep the Holstein coupling operator $\mathbf{U}_{ev}$ is processed with the three families of matrices $\mathcal{C}_{n,q}^{\text{pop}}$, $\mathcal{C}_{n,q}^{\text{L}}$ and $\mathcal{C}_{n,q}^{\text{R}}$ returned by Algorithm 5. For an electronic block (m, n) we distinguish two cases:

- **Population block** ($m = n$) — only the oscillators belonging to monomer m are affected. Update rule $A_{i,\sigma,j} \rightarrow \sum_k [\mathcal{C}_{m,q}^{\text{pop}}]_{\sigma k} A_{i,k,j}$.
- **Coherence block** ($m \neq n$) — oscillators of monomer m use $\mathcal{C}_{m,q}^{\text{L}}$, those of monomer n use $\mathcal{C}_{n,q}^{\text{R}}$, every other site is left untouched.

Algorithm 8 Local e–v coupling update $\mathcal{C}$

Require: vibronic MPO array $\mathcal{O}[N, N]$;

```

1: coupling matrices  $\mathcal{C}^{\text{pop}}, \mathcal{C}^{\text{L}}, \mathcal{C}^{\text{R}}$ 
2:  $\tilde{\mathcal{O}} \leftarrow \mathcal{O}$  ▷ in-place copy
3: for  $m \leftarrow 1$  to  $N$  do
4:   for  $n \leftarrow 1$  to  $N$  do
5:     if  $m = n$  then ▷ population block
6:       for  $q \leftarrow 1$  to  $Q$  do
7:          $A \leftarrow \mathcal{O}_{m,n}[m, q]$ 
8:          $C \leftarrow \mathcal{C}_{m,q}^{\text{pop}}$ 
9:          $\tilde{A}_{i,\sigma,j} \leftarrow \sum_k C_{\sigma k} A_{i,k,j}$ 
10:         $\tilde{\mathcal{O}}_{m,n}[m, q] \leftarrow \tilde{A}$ 
11:      end for
12:    else ▷ coherence block
13:      for  $q \leftarrow 1$  to  $Q$  do
14:         $A \leftarrow \mathcal{O}_{m,n}[m, q]; C \leftarrow \mathcal{C}_{m,q}^{\text{L}}; \tilde{A}_{i,\sigma,j} \leftarrow \sum_k C_{\sigma k} A_{i,k,j}; \tilde{\mathcal{O}}_{m,n}[m, q] \leftarrow$   

         $\tilde{A}$ 
15:      end for
16:      for  $q \leftarrow 1$  to  $Q$  do
17:         $A \leftarrow \mathcal{O}_{m,n}[n, q]; C \leftarrow \mathcal{C}_{n,q}^{\text{R}}; \tilde{A}_{i,\sigma,j} \leftarrow \sum_k C_{\sigma k} A_{i,k,j}; \tilde{\mathcal{O}}_{m,n}[n, q] \leftarrow \tilde{A}$ 
18:      end for
19:    end if
20:  end for
21: end for
22: return  $\tilde{\mathcal{O}}$ 
    
```

Electronic Evolution

The purely electronic part of the Liouvillian, $\mathbf{U}_e = \exp[-i\Delta t H_e, \cdot]$, acts *globally* on the vibronic MPO and mixes all vibrational blocks according to Eq. (3.25). A direct implementation would concatenate the corresponding matrix–product operators by repeated *direct sums*, inflating the bond dimension from its initial value χ up to $N^2\chi$. To keep the virtual dimension manageable we accumulate the required sums

sequentially and trigger a truncated singular-value decomposition (SVD) after a fixed number p of terms, retaining at most $\chi_{\max}$ singular values. The SVD may sweep from left to right or right to left, selected by a Boolean flag **RtL**. The resulting procedure is summarised in Algorithm 9;

Algorithm 9 Electronic evolution with periodic SVD compression

Require: vibronic MPO array $\mathcal{O}[N, N]$; time step Δt ;

```

1: electronic Hamiltonian  $H_e$ ;
2: maximum bond dimension  $\chi_{\max}$ ; compression interval  $p$ ;
3: sweep direction flag RtL
4:  $U \leftarrow \exp[-i\Delta t H_e]$ 
5:  $\mathcal{O}^{\text{new}} \leftarrow \text{array}(N, N)$ 
6: for  $k \leftarrow 1$  to  $N$  do
7:   for  $l \leftarrow 1$  to  $k$  do ▷ construct lower triangle
8:      $\text{acc} \leftarrow \emptyset$ ;  $\text{count} \leftarrow 0$ 
9:     for  $m \leftarrow 1$  to  $N$  do
10:      for  $n \leftarrow 1$  to  $N$  do
11:         $\text{count} \leftarrow \text{count} + 1$ 
12:         $c \leftarrow U_{km} U_{ln}^*$ 
13:         $\text{term} \leftarrow c \mathcal{O}_{m,n}$  ▷ scale MPO
14:         $\text{acc} \leftarrow (\text{acc} = \emptyset) ? \text{term} : \text{direct\_sum}(\text{acc}, \text{term})$ 
15:        if  $\text{count} \bmod p = 0$  then
16:          if RtL then
17:             $\text{acc} \leftarrow \text{compress\_SVD\_RtL}(\text{acc}, \chi_{\max})$ 
18:          else
19:             $\text{acc} \leftarrow \text{compress\_SVD\_LtR}(\text{acc}, \chi_{\max})$ 
20:          end if
21:        end if
22:      end for
23:    end for
24:     $\mathcal{O}_{k,l}^{\text{new}} \leftarrow \text{acc}$ 
25:     $\mathcal{O}_{l,k}^{\text{new}} \leftarrow \text{acc}^\dagger$  ▷ Hermitian symmetry
26:  end for
27: end for
28: return  $\mathcal{O}^{\text{new}}$ 

```

3.6 Memory Footprint and Computational Cost

3.6.1 Memory

The full state ρ contains N^2 MPO blocks where each $\mathcal{O}_{m,n}$ contains $(M-2)\chi^2 d^2 + 2\chi d^2$ `np.complex128` elements. For moderate system parameters $N = 20, Q = 4, d = 8$, and $\chi = 12$ the state occupies about 4.6 GBs (gigabytes) of memory (excluding the overhead for storing tensors as objects inside the state array). The $\mathcal{O}(\chi^2)$ scaling of the state memory size becomes fundamentally crucial in the implementation of DAMPF. Consider a substantially small system: a dimer with each site coupled to

two pseudomodes ($N = 2, Q = 2$). Further, we choose parameters $\chi = 12$ and $d = 8$ (modest values). As noted in subsection 3.3.2, electronic evolution changes the bond dimension χ to $N^2\chi$ per Suzuki-Trotter step which, unless a compression scheme is implemented, causes catastrophic memory blow-up.

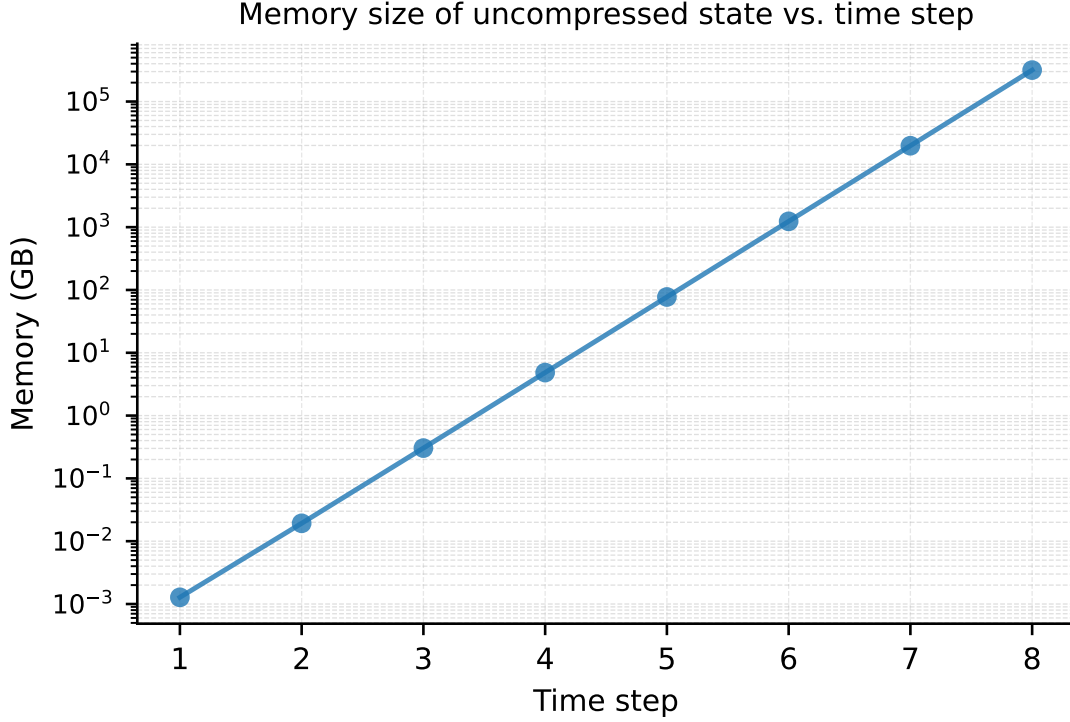


Figure 3.5: Catastrophic memory blow-up of the uncompressed state of a dimer with $Q = 2, d = 8$, and $\chi = 12$

This makes implementation of a compression scheme after suitable number of direct sums a necessary and fundamental step in the algorithm that cannot be do away with.

All matrices $\mathcal{R}_{n,q}^{(v)}$, $\mathcal{B}_{n,q}$ and the families $\mathcal{C}_{n,q}^{(ev)}(m, n)$ have dimension $d^2 \times d^2$ and are pre-computed once the oscillator parameters are fixed. In practice, they are stored in five arrays of shapes (N, Q, d^2, d^2) of `np.complex128` elements: one each for $\mathcal{R}_{n,q}^{(v)}$, $\mathcal{B}_{n,q}$, $\mathcal{C}_{n,q}^{pop}$, $\mathcal{C}_{n,q}^L$, and $\mathcal{C}_{n,q}^R$. Each array occupies $16 \times NQd^2$ bytes of memory. For parameters $N = 20, Q = 4$, and $d = 8$, this amounts to about 5 MBs (megabytes) of memory per array. These local updates do not change the bond dimension χ and do not, in principle, require further memory.

3.6.2 Computational Complexity

The local update of $A_{i,j}^{(m,n)}$ in $\mathcal{O}_{m,n}$ via $W_{x;i}$ ($x \in \{v, ev, \mathcal{D}\}$) in Equation 3.16 requires contracting a $d^2 \times d^2$ matrix with a tensor of shape (χ, d^2, χ) requiring $2\chi^2 d^4$ flops and

scaling as $\mathcal{O}(\chi^2 d^4)$. For each MPO $\mathcal{O}_{m,n}$, $\mathcal{R}_{n,q}^{(v)}$ and $\mathcal{B}_{n,q}$ perform M such operations, while $\mathcal{C}_{n,q}^{(ev)}(m, n)$ perform $3Q$ (Q for populations and $2Q$ for coherences). Therefore, vibrational evolution and damping cost $2N^2 M \chi^2 d^4$ flops per Suzuki-Trotter step while electron-vibrational coupling requires $6M \chi^2 d^4$ flops. It is immediately noted that these local transformations are independent and can be performed in parallel not only across tensors in $\mathcal{O}_{m,n}$ but across all the MPOs in ρ . Such a parallelization was not implemented in this thesis because the computational cost of local updates is tiny as compared to electronic evolution and we see below.

Taking the partial-trace of ρ after every time-step requires $2M\chi^2 + 2\chi$ flops and the computational cost $\mathcal{O}(M\chi^2)$ scales linearly with the chain length for naive matrix multiplication.

The electronic evolution without compression requires only scalar multiplications and additions (direct sums) which have negligible computational cost but cause memory blow-up as noted above. Performing SVD of MPO (after chosen number of direct sums) thus becomes as necessary. Compression can be performed in different ways: left-to-right SVD compression followed by right-to-left SVD canonicalization or left-to-right SVD canonicalization followed by right-to-left SVD compression. QR could also be used for left-to-right canonicalization. A left-to-right compression followed by right-to-left SVD canonicalization of an MPO requires $\approx 96M\chi^3 d^2$ flops, see Appendix B. For one Suzuki-Trotter step this amounts to $96MN^4\chi^3 d^2$ flops. It can also be noted that electronic evolution is 'embarrassingly parallel': Each MPO block can be evolved in parallel.

Operation per time step	Number of Flops	Complexity
$\mathbf{U}_v$	$2N^3 Q \chi^2 d^4$	$\mathcal{O}(N^3 Q \chi^2 d^4)$
$\mathbf{U}_{ev}$	$6N Q \chi^2 d^4$	$\mathcal{O}(N Q \chi^2 d^4)$
$e^{\mathcal{D}\Delta t}$	$2N^3 Q \chi^2 d^4$	$\mathcal{O}(N^3 Q \chi^2 d^4)$
$\mathbf{U}_e$	$96N^5 Q \chi^3 d^2$	$\mathcal{O}(N^5 Q \chi^3 d^2)$
Partial Trace	$2N Q \chi^2$	$\mathcal{O}(N Q \chi^2)$

Table 3.1: Flop counts and asymptotic complexities for operations per time step assuming all MPO blocks are evaluated.

Errors

DAMPF is an essentially exact numerical approach to spin-boson models. Its only intrinsic error sources are:

- **Trotter-step error**,
- **Oscillator Hilbert-space truncation error**, and
- **MPO compression error**,

each of which can be systematically reduced to meet any prescribed accuracy. However, when you represent a *structured* environment by a discrete set of damped harmonic oscillators, an additional **bath-mapping error** appears. This error quantifies how faithfully the discrete bath's correlation function reproduces that of the true continuum environment, and can only be reduced by improving the fit between the oscillator network's correlation function and the target spectral density[18]. Such an analysis is not done in this thesis.

Parallelization

Electronic evolution is the most expensive part of the algorithm. Noting that $\rho = \rho^\dagger \implies \mathcal{O}_{m,n} = \mathcal{O}_{n,m}^\dagger$, it is sufficient to calculate only $N(N+1)/2$ MPOs instead of all N^2 blocks per time step. This reduces total computational time by a factor of $1/2 + (1/N^2)$. Further, creation of new MPO blocks at every time step can be performed in parallel. If parallelization is done in a way that CPU cores are equally divided to evaluate MPOs in parallel it is optimal for moderate system parameter sizes to assign 1 – 2 cores for each process. I did not see a linear speed-up when more than two cores are used per 'worker' process.

Chapter 4

Results

In this section I present DAMPF simulations of various systems. I look at different environmental regimes based on *reorganization energy* and compare the simulations done via SVD based DAMPF and RSVD based DAMPF. In all simulations MPOs are compressed in a right-to-left compression sweep followed by a left-to-right canonicalization via QR. The comparison between SVD and RSVD based simulations is done by switching between the two in the right-to-left compression step, which from now on will be referred to as SVD-DAMPF and RSVD-DAMPF. Unless stated otherwise the simulations are performed with an integration time of 0.5 fs.

The error in SVD-DAMPF is quantified as:

$$\varepsilon_{svd-dampf} = \sum_{m,n} \sum_{i=1}^{\frac{N^2-1}{k_c}} \|\mathcal{O}_i^{m,n} - \tilde{\mathcal{O}}_i^{m,n}\|^2 \quad (4.1)$$

where k_c is the compression interval. $\mathcal{O}_i^{m,n}$ is the MPO at (m,n) site after k_c direct sums which is compressed via truncated SVD to $\tilde{\mathcal{O}}_i^{m,n}$.

In RSVD-DAMPF the error is measured as:

$$\varepsilon_{rsvd-dampf} = \sum_{m,n} \sum_{i=1}^{\frac{N^2-1}{k_c}} \|\tilde{\mathcal{O}}_i^{m,n} - \mathcal{O}_{i;rsvd}^{m,n}\|^2 \quad (4.2)$$

where $\mathcal{O}_{i;rsvd}^{m,n}$ is obtained by truncating $\mathcal{O}_i^{m,n}$ via RSVD. ε_{rsvd} is measured by comparing the accuracy to SVD truncated MPO $\|\tilde{\mathcal{O}}_i^{m,n}\|$ because RSVD does not construct singular that it doesn't keep.

Figure 4.1 shows population dynamics of a dimer with $\Omega_1 = \Omega_2$ and $J_{12} = 500 \text{ cm}^{-1}$ with initial excitation at site 1. Each site is coupled to two oscillators:

- $\omega_1 = 1500 \text{ cm}^{-1}$, $\gamma_1 = (1 \text{ ps})^{-1}$, $T_1 = 300 \text{ K}$, $s_1 = 0.2$, $d_1 = 8$
- $\omega_2 = 500 \text{ cm}^{-1}$, $\gamma_2 = (100 \text{ fs})^{-1}$, $T_2 = 300 \text{ K}$, $s_2 = 0.1$, $d_2 = 8$

The simulations were performed without parallelization on an 8-core machine with a memory of 8 GB. The SVD-DAMPF version for $\chi = 18$ and $k_c = 1$ was first reported in [16].

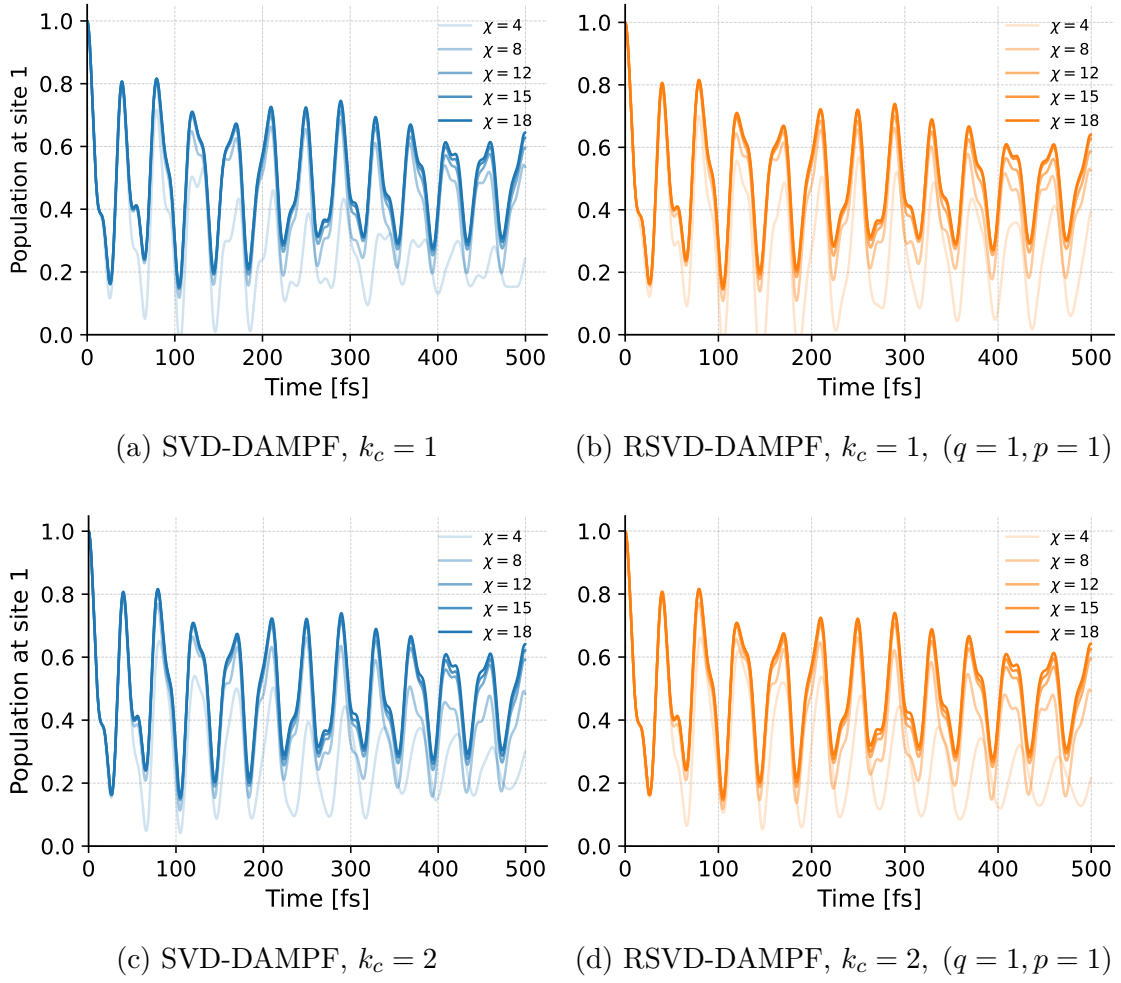


Figure 4.1: Population dynamics of a dimer obtained with SVD-DAMPF(left column) and RSVD-DAMPF(right column) at two compression intervals: $k_c = 1$ (top) and $k_c = 2$ (bottom).

The dynamics shown here match the result of authors exactly. My result also confirms the need for using a bond dimension of 18 for a converged solution. RSVD-DAMPF(with power iterations $q = 1$ and oversampling $p = 1$) also shows a similar convergence.

Figure 4.2 shows the simulation time for both the methods. At lower bond dimensions the simulation time is similar(SVD-DAMPF is actually slightly faster at $\chi = 4$). However, the gap between the simulation times widens with increasing bond-dimension. At $\chi = 18$ and $k_c = 1$ RSVD-DAMPF provides a speed-up of ≈ 1.29 . This speed-up increases to ≈ 1.59 with $k_c = 2$. We also notice an interesting result: the simulation time of SVD-DAMPF increases when the compression interval k_c is increased while RSVD-DAMPF simulation time decreases.

This can be explained by looking at the computational complexity of SVD and RSVD and noting that $k_c = 2$ means the simulations are doing only half the number of compression steps but with bigger matrix sizes of $(3\chi \times 3\chi d^2)$ instead of

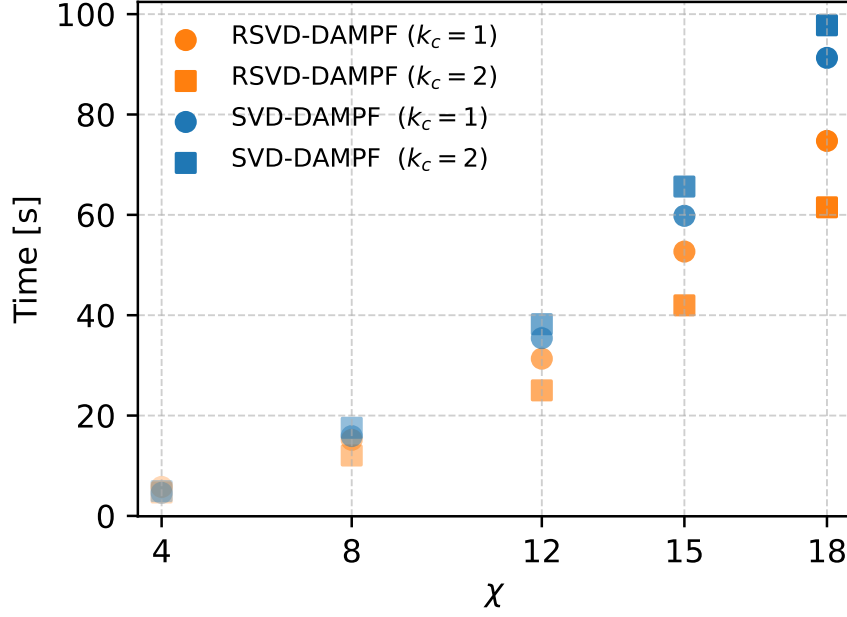


Figure 4.2: Simulation time of dimer

$(2\chi \times 2\chi d^2)$ when $k_c = 1$. For SVD, the computational cost scales as $\mathcal{O}((2\chi d^2)(2\chi)^2)$ and $\mathcal{O}((3\chi d^2)(3\chi)^2)$ for $k_c = 1$ and $k_c = 2$ respectively. Even while doing only half the number of compressions the factor 3^2 eventually results in an increase in compute time for $k_c = 2$. For RSVD, the complexities read $\mathcal{O}((2\chi d^2)(2\chi^2))$ and $\mathcal{O}((3\chi d^2)(3\chi^2))$ for $k_c = 1$ and $k_c = 2$ respectively. The complexity doesn't increase by an exponential factor and doing only half the number of compressions wins. While this might suggest that one can keep increasing k_c even larger and get quicker results but this is prevented by the size of the state in memory. A larger compression interval temporarily creates larger MPOs in memory before compressing them which can result in the compute node running out of memory. I noticed a significant slowdown at $k_c = 3$ while $k_c = 4$ caused the system to run out of memory and crash even when χ was fixed. On systems with larger memory one can fiddle with the compression interval depending on the size of the state

Figure 4.3 shows the truncation error for the case of $\chi = 18$. While Equation 2.1.3 provides an oversampling knob to control errors incurred via RSVD, it can be noted that oversampling of $p = 1$ in RSVD-DAMPF in the case of the dimer reduces $\varepsilon_{rsvd-dampf}$ significantly as compared to $p = 0$ and is sufficient to produce accurate dynamics. Figure 4.4 shows truncation errors when compression interval k_c is increased to 2. We notice $\varepsilon_{svd-dampf}$ increase but $\varepsilon_{rsvd-dampf}$ almost becomes negligible. This can be explained by looking Equation 2.1.3 and noting that the variance of the error is directly proportional to k : smaller truncated approximations have a smaller error variance.

However, as equations Equation 4.1 and Equation 4.2 show, a vanishing $\varepsilon_{rsvd-dampf}$ does *not* imply that the *total* truncation error is negligible. Recall that $\varepsilon_{rsvd-dampf}$

measures only the extra error introduced by the randomized SVD step, *on top of* the baseline truncation error $\varepsilon_{\text{svd-dampf}}$, which arises from discarding the smallest singular values before any decomposition. In fact, for $k_c = 2$ we observe that $\varepsilon_{\text{svd-dampf}}$ *increases* (see Figure 4.4). This behavior is expected, since retaining only χ singular values out of a $3\chi \times 3\chi$ matrix necessarily discards more information than keeping χ out of a $2\chi \times 2\chi$ matrix.

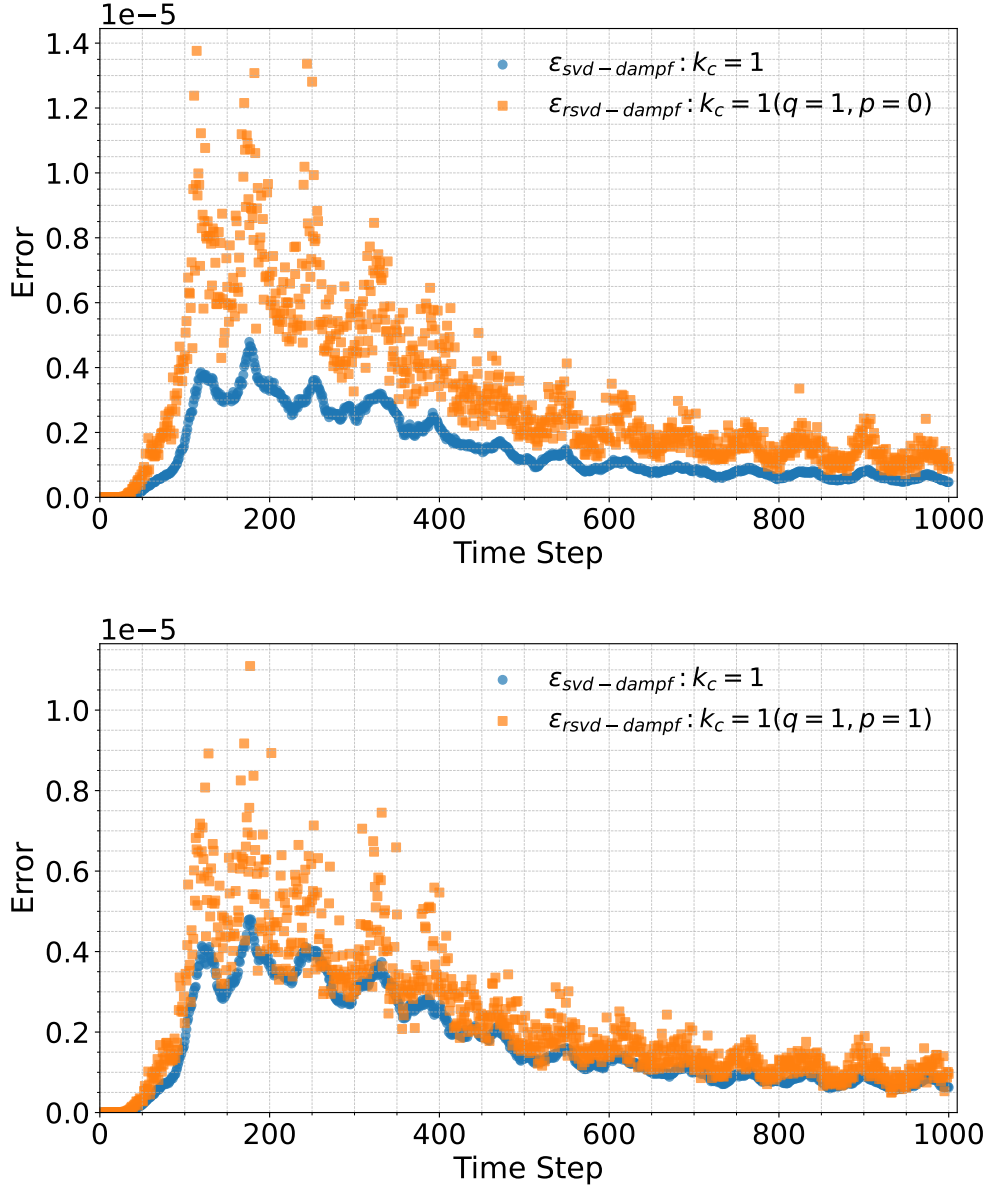


Figure 4.3: Truncation errors at each time step for bond-dimension $\chi = 18$ for $k_c = 1$ with oversampling $p = 0$ (top figure) and $p = 1$ (bottom figure)

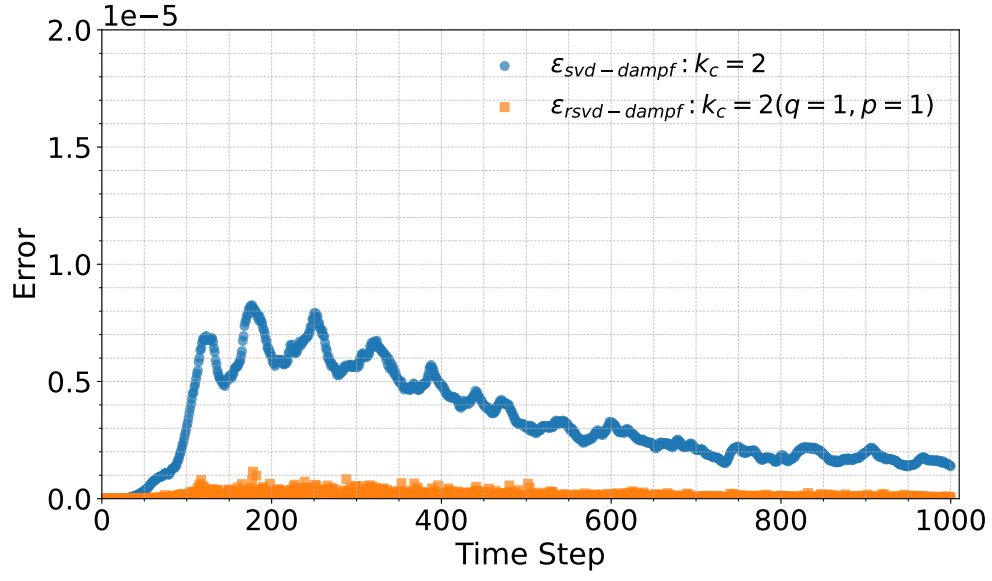


Figure 4.4: Truncation errors of the dimer at $\chi = 18$ when compression interval k_c is increased to 2

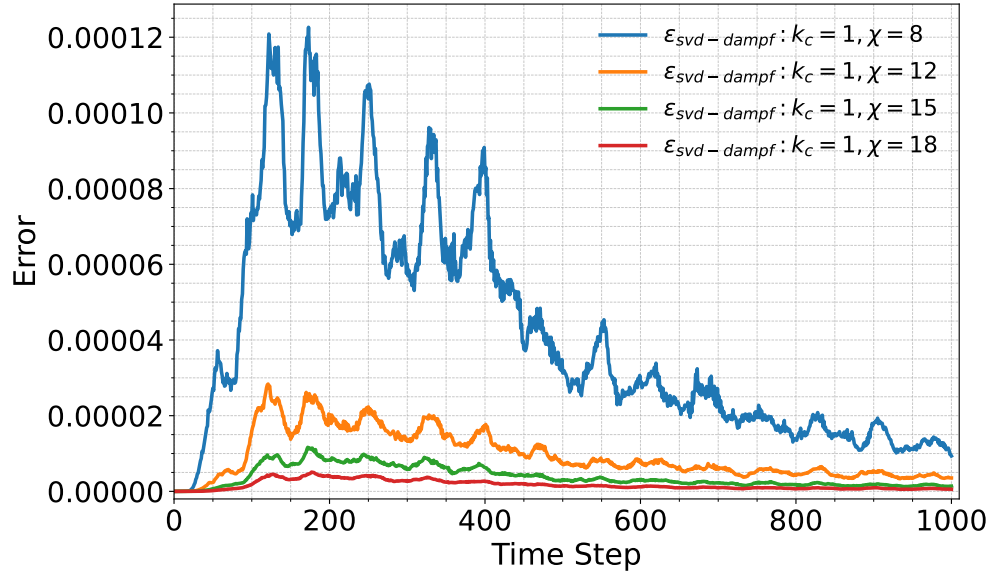


Figure 4.5: Behaviour of $\varepsilon_{\text{svd-dampf}} k_c = 1$ at different bond dimensions.

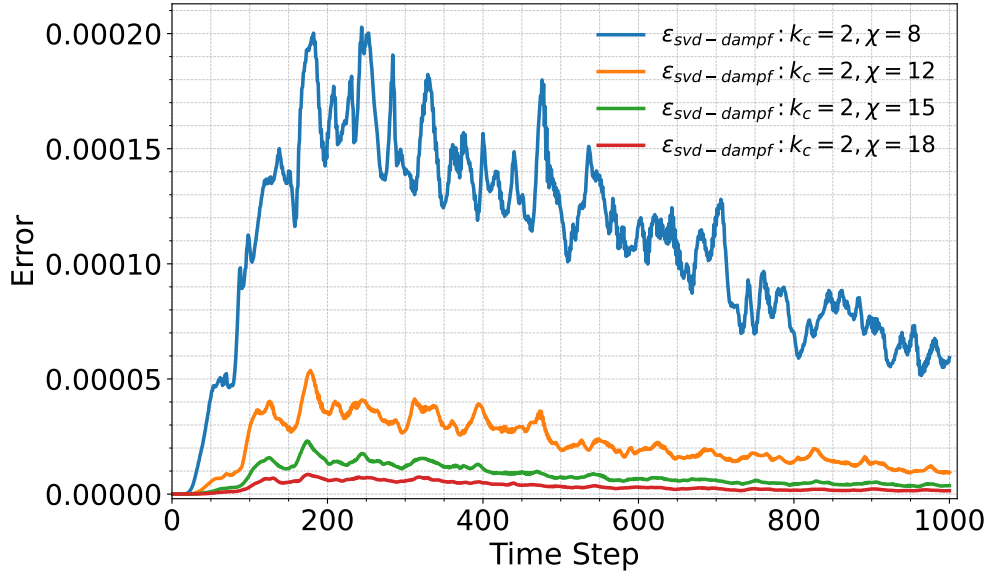


Figure 4.6: Behaviour of $\varepsilon_{\text{svd-dampf}}$ for different bond dimensions. Doubling the Compression interval almost doubles the error.

Now, consider a non-periodic chain of 8 sites with $\Omega_n = \Omega_m$ and $J_{n,n+1} = 300\text{cm}^{-1}$ where each site is coupled to 6 oscillators with the following parameters:

Table 4.1: Parameters of the six vibrational oscillators.

Mode	Frequency ω (cm^{-1})	Damping rate γ	Huang-Rhys s
1	600	$(100 \text{ fs})^{-1}$	0.100
2	750	$(1 \text{ ps})^{-1}$	0.046
3	900	$(1 \text{ ps})^{-1}$	0.046
4	1050	$(1 \text{ ps})^{-1}$	0.046
5	1200	$(1 \text{ ps})^{-1}$	0.046
6	1350	$(1 \text{ ps})^{-1}$	0.046

with truncation level $d = 8$ and temperature $T = 300\text{K}$. The oscillator parameters were chosen such that the reorganization energy $\lambda = \sum_q s_q \omega_q \approx J_{n,n+1}$. In this non-perturbative regime—where the reorganization energy λ is on the same order as the electronic coupling $J_{n,n+1}$ between sites—energy transfer exhibits a hybrid character with both coherent oscillations and dissipative hopping

Figure 4.9 shows population dynamics in non-perturbative regime. The simulations were performed on a 36 core node with a parallel implementation of SVD-DAMPF and RSVD-DAMPF($q = 1, p = 1$).

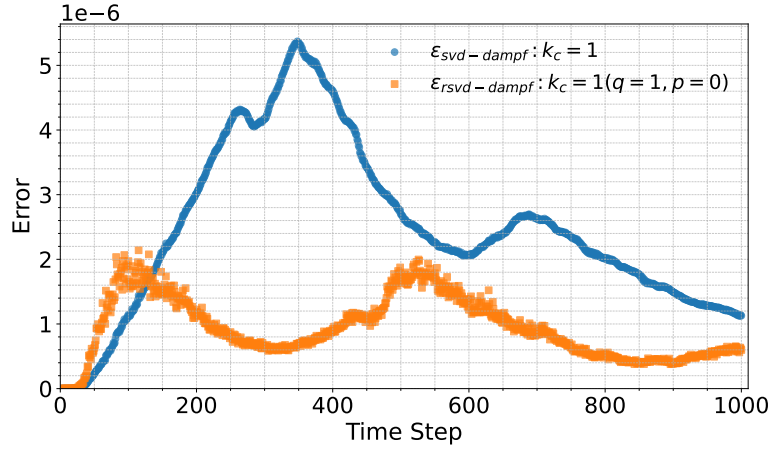


Figure 4.7: Truncation errors for non-perturbative system

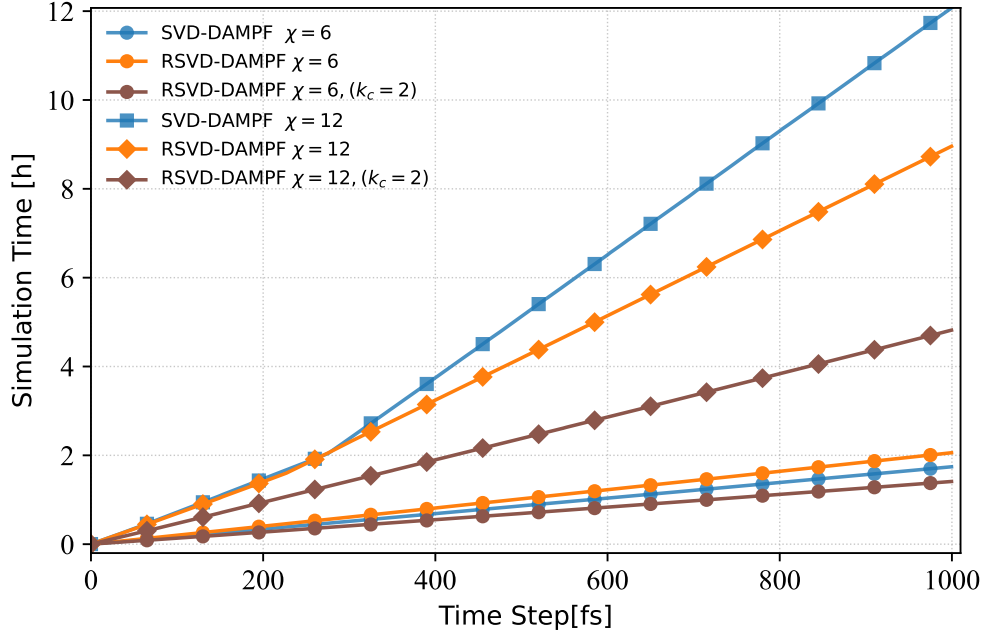


Figure 4.8: Simulation time in non-perturbative example

Figure 4.9(c) and (d) are identical while RSVD-DAMPF provides a ≈ 1.33 speed-up. The trend from the dimer simulations follows again: RSVD-DAMPF scales better with increasing bond-dimension.

While doubling the compression interval to $k_c = 2$ halves the runtime, it does not reproduce the $k_c = 1$ dynamics in our non-dimer system. In the dimer benchmark, $\chi = 18$ was sufficient to suppress truncation error, so doubling k_c merely doubled an already negligible error. Here, however, χ is not large enough: if the baseline truncation error at $k_c = 1$ is $\varepsilon \ll 1$, then at $k_c = 2$ it becomes roughly 2ε . A failure to match the $k_c = 1$ result therefore indicates that χ must be increased until ε is small enough that even a factor-of-two increase remains acceptable.

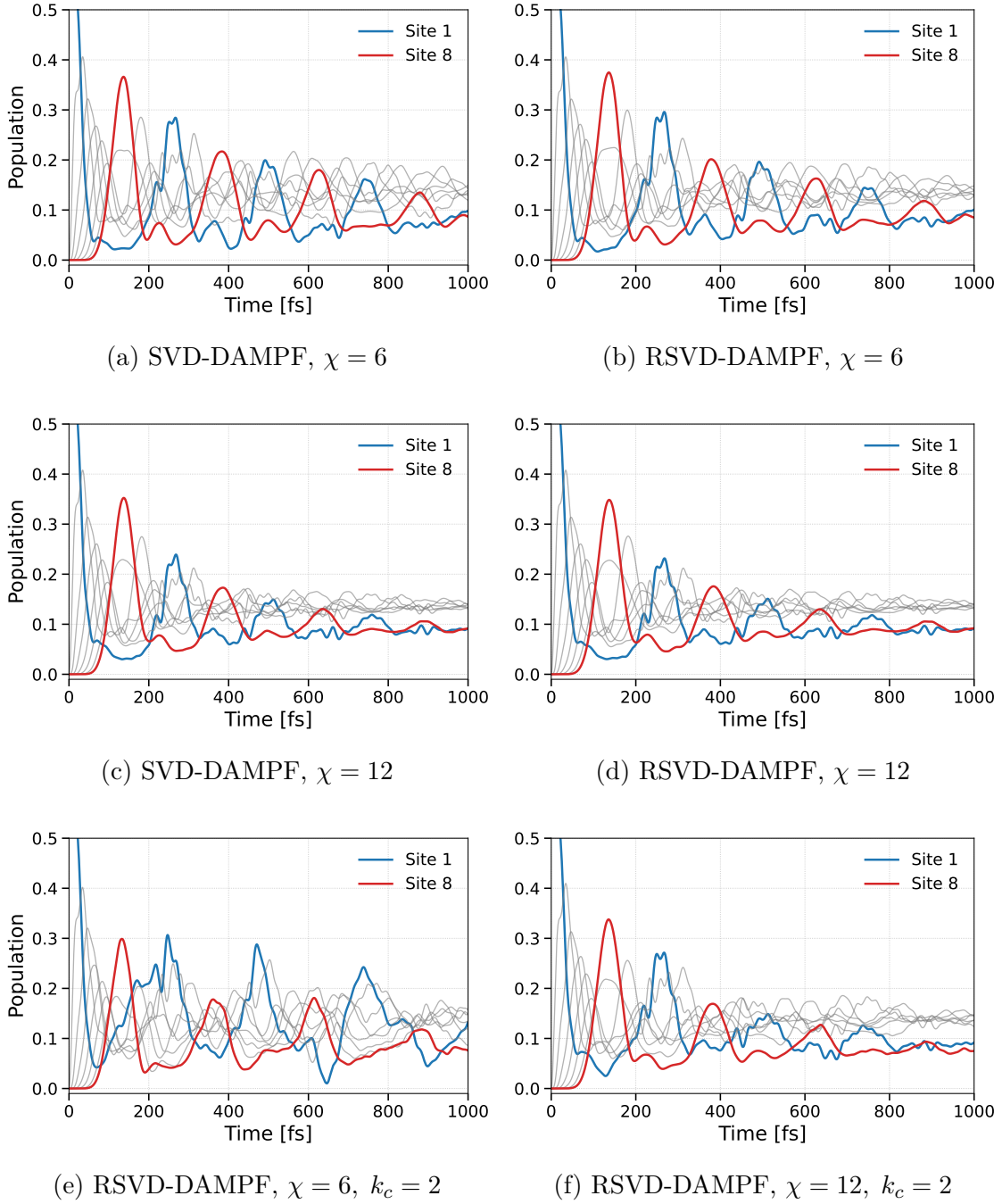


Figure 4.9: Population dynamics in a non-perturbative chain of 8 sites with each site coupled to 6 pseudomodes.

Now, if the Huang-Rhys factors of the last five oscillators are changed uniformly such that $\lambda \gg J_{n,n+1}$, a relatively large reorganization energy makes inter-site transfer energetically unfavorable and slows it far below competing relaxation processes. The excitation stays localized at the initial site in what is known as Polaron formation. Figure 4.10 showcases this phenomenon for the system under consideration.

Figure 4.11 shows the same trend as before: RSVD-DAMPF scales better in time for increasing bond dimension. Figure 4.12 shows truncation errors for the bond-

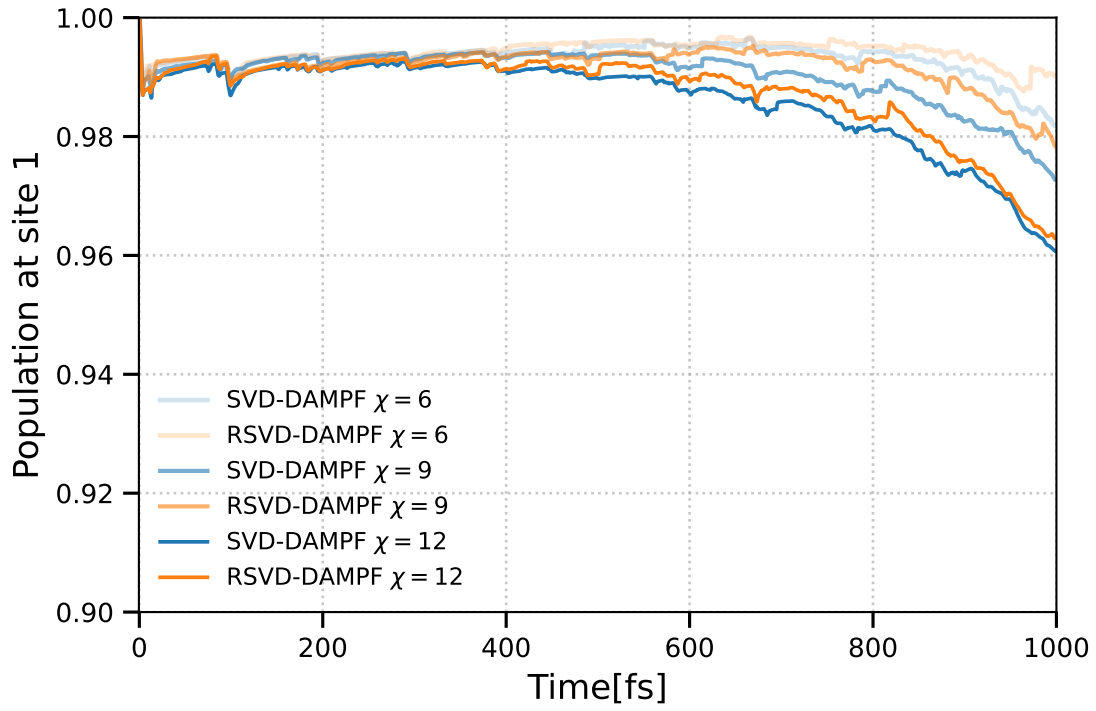


Figure 4.10: Population dynamics of the first site when $\lambda \gg J$. The excitation is localized at the initial site with high probability.

dimension $\chi = 9$. Even at 0 oversampling and low bond-dimension we find $\varepsilon_{rsvd-dampf}$ within acceptable limits.

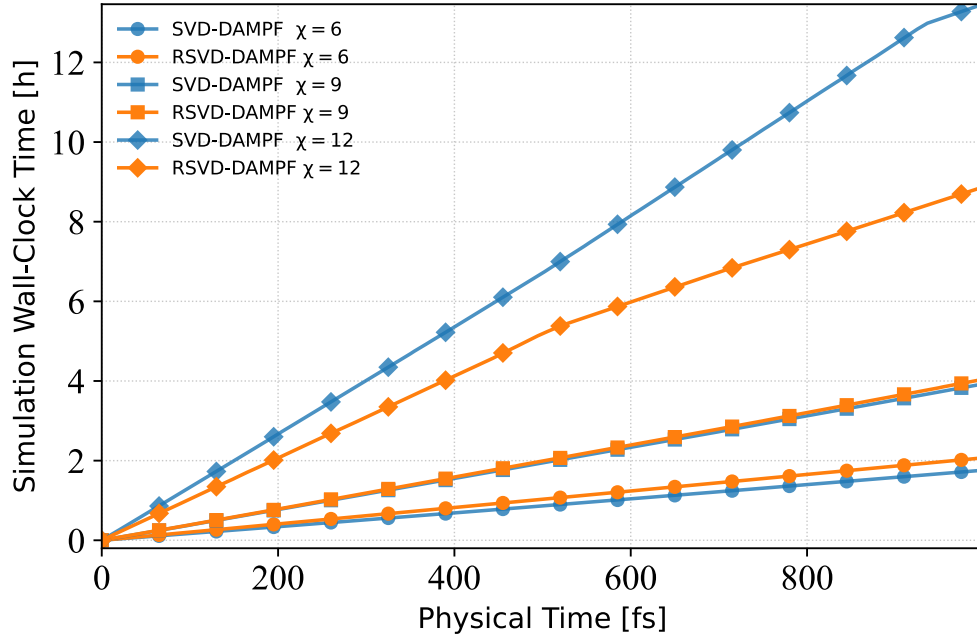


Figure 4.11: Total simulation time for the system with large reorganization energy.

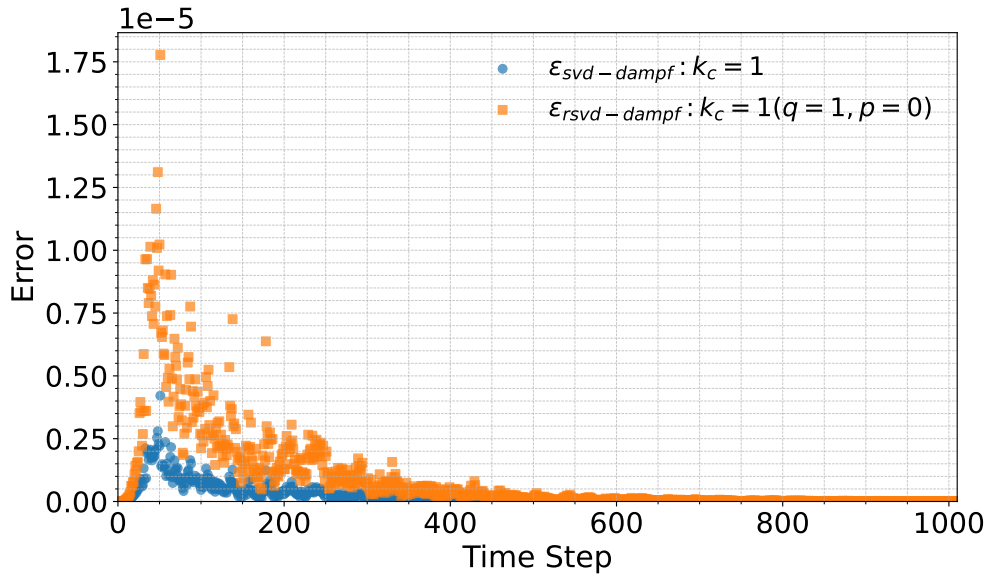


Figure 4.12: Truncation errors in Polaron Simulation evaluated at $\chi = 9$

Chapter 5

Conclusion and Outlook

The randomized-factorization variant of DAMPF demonstrates superior scaling with bond dimension χ compared to the deterministic SVD approach. In my benchmarks, RSVD-DAMPF attained relative errors below 10^{-4} (figs. 4.3, 4.7 and 4.12) while running up to $1.8\times$ faster—even with minimal oversampling and without extra power iterations. For large-scale simulations, such speedups can translate into wall-clock savings of days.

That said, RSVD is not a universal drop-in replacement for SVD: it requires the user to specify the target rank in advance, making it most effective when χ is both large and known a priori. Consequently, integrating RSVD into DAMPF demands careful tuning of system parameters, particularly the compression interval k_c . For example, increasing k_c from 1 to 2 halves the simulation time and drives $\varepsilon_{rsvd-dampf}$ to negligible levels, even though $\varepsilon_{svd-dampf}$ roughly doubles. If $\varepsilon_{svd-dampf}$ is very small (or equivalently the bond dimension is large enough to minimize errors) one is at a practical “sweet spot” for randomized compression.

Stability: In addition to its speed and accuracy benefits, RSVD-DAMPF exhibits enhanced numerical stability. Over my entire test suite, only a single RSVD step failed to converge—an event that is far from rare in deterministic SVD-based algorithms. Such divergences typically arise when the ratio between the largest and smallest singular values of a site tensor becomes excessively large [7]. Because RSVD never explicitly computes the trailing singular values, it is inherently protected against this form of numerical breakdown. The final two examples in the preceding section were chosen to demonstrate that RSVD-DAMPF can accurately handle even extreme regimes—specifically, underdamped modes of frequency $(1\text{ ps})^{-1}$, whose bath correlations decay slowly. The simulations are scalable and the inherent probabilistic nature of factorization, even with minimal oversampling, doesn’t seem to pose a problem.

Algorithmic perspective: While DAMPF itself is an exact method, the problem lies in the limited usefulness of the way oscillators are modeled. The assumption that each oscillator interact with an independent bath is adequate for weakly structured environments but can misrepresent strongly correlated or engineered reservoirs. Recent work shows that, if all oscillators are coupled to the *same* Markovian bath,

the environment can be compressed to a handful of pseudomodes, yielding a far smaller effective system size and opening the door to simplified propagators [19]. Incorporating this shared-bath reduction into DAMPF is a promising direction for future research.

Future numerical improvements: *Mathematically*, replacing the dense Gaussian test matrix Ω in RSVD with a structured transform—e.g. a subsampled randomized Hadamard or Fourier matrix—reduces the complexity from $\mathcal{O}(mnk)$ to $\mathcal{O}(mn \log k)$ [18]. This method lacks a strong theoretical analysis of errors but has proven to be accurate in practice. *Computationally*, the Python ecosystem still lacks a high-performance RSVD kernel: NumPy has none, and SciPy’s reference implementation ran about 30% slower than my custom routine. While RSVD still employs deterministic SVD and QR at its core, it does so sequentially. A compiled backend that exploits batched BLAS, GPU acceleration, and cuSOLVER could plausibly deliver another 1.5–2 $\times$ speed-up for larger systems, pushing randomized DAMPF closer to hardware peak performance.

Bibliography

- [1] D. Tamascelli, R. Rosenbach, and M. B. Plenio, “Improved scaling of time-evolving block-decimation algorithm through reduced-rank randomized singular value decomposition,” *Phys. Rev. E*, vol. 91, p. 063306, 6 Jun. 2015. DOI: 10.1103/PhysRevE.91.063306. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevE.91.063306>.
- [2] C. Eckart and G. Young, “The approximation of one matrix by another of lower rank,” *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936. DOI: 10.1007/BF02288367.
- [3] G. H. Golub and C. Reinsch, “Singular value decomposition and least squares solutions,” *Numerische Mathematik*, vol. 14, pp. 403–420, 1970. DOI: 10.1007/BF02163027.
- [4] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff, “A set of level 3 basic linear algebra subprograms,” vol. 16, no. 1, 1990, ISSN: 0098-3500. [Online]. Available: <https://doi.org/10.1145/77626.79170>.
- [5] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011. DOI: 10.1137/090771806.
- [6] J. C. Bridgeman and C. T. Chubb, “Hand-waving and interpretive dance: An introductory course on tensor networks,” *Journal of Physics A: Mathematical and Theoretical*, vol. 50, no. 22, p. 223001, May 2017. DOI: 10.1088/1751-8121/aa6dc3. [Online]. Available: <https://dx.doi.org/10.1088/1751-8121/aa6dc3>.
- [7] C. Hubig, I. P. McCulloch, and U. Schollwöck, “Generic construction of efficient matrix product operators,” *Phys. Rev. B*, vol. 95, p. 035129, 3 Jan. 2017. DOI: 10.1103/PhysRevB.95.035129. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevB.95.035129>.
- [8] F. Verstraete and J. I. Cirac, “Matrix product states represent ground states faithfully,” *Phys. Rev. B*, vol. 73, p. 094423, 9 Mar. 2006. DOI: 10.1103/PhysRevB.73.094423. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevB.73.094423>.

- [9] “Bibliography,” in *The Theory of Open Quantum Systems*, Oxford University Press, Jan. 2007, ISBN: 9780199213900. DOI: 10.1093/acprof:oso/9780199213900.002.14006. eprint: https://academic.oup.com/book/0/chapter/197977158/chapter-ag-pdf/45505569/book_27757_section_197977158.ag.pdf. [Online]. Available: <https://doi.org/10.1093/acprof:oso/9780199213900.002.14006>.
- [10] B. M. Garraway, “Nonperturbative decay of an atomic system in a cavity,” *Physical Review A*, vol. 55, no. 3, pp. 2290–2303, 1997. DOI: 10.1103/PhysRevA.55.2290.
- [11] F. Mascherpa, A. Smirne, S. F. Huelga, and M. B. Plenio, “Open systems with error bounds: Spin-boson model with spectral density variations,” *Phys. Rev. Lett.*, vol. 118, p. 100 401, 10 Mar. 2017. DOI: 10.1103/PhysRevLett.118.100401. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.118.100401>.
- [12] A. Lemmer, C. Cormick, D. Tamascelli, T. Schätz, S. F. Huelga, and M. B. Plenio, “A trapped-ion simulator for spin-boson models with structured environments,” *New Journal of Physics*, vol. 20, no. 7, p. 073 002, 2018. DOI: 10.1088/1367-2630/aac053. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1367-2630/aac053>.
- [13] J. Iles-Smith, N. Lambert, A. Nazir, and A. Olaya-Castro, “Environmental dynamics, correlations, and the emergence of non-canonical equilibrium states in open quantum systems,” *Physical Review A*, vol. 90, no. 3, p. 032 114, 2014. DOI: 10.1103/PhysRevA.90.032114.
- [14] J. Prior, A. W. Chin, S. F. Huelga, and M. B. Plenio, “Efficient simulation of strong system-environment interactions,” *Physical Review Letters*, vol. 105, no. 5, p. 050 404, 2010. DOI: 10.1103/PhysRevLett.105.050404.
- [15] D. Tamascelli, A. Smirne, S. F. Huelga, and M. B. Plenio, “Efficient simulation of finite-temperature reservoirs with error bounds,” *Physical Review Letters*, vol. 123, no. 9, p. 090 402, 2019. DOI: 10.1103/PhysRevLett.123.090402.
- [16] A. D. Somoza, O. Marty, J. Lim, S. F. Huelga, and M. B. Plenio, “Dissipation-assisted matrix-product factorization,” *Physical Review Letters*, vol. 123, p. 100 502, 2019. DOI: 10.1103/PhysRevLett.123.100502.
- [17] A. Somoza Márquez, “Dissipation-assisted matrix product factorization of vibronic networks,” Available at , PhD thesis, University of Ulm, Example City, CA, Feb. 2023. DOI: 10.18725/OPARU-47490.
- [18] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert, “Randomized algorithms for the low-rank approximation of matrices,” *Proceedings of the National Academy of Sciences*, vol. 104, no. 51, pp. 20 167–20 172, 2007. DOI: 10.1073/pnas.0709640104. eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.0709640104>. [Online]. Available: <https://www.pnas.org/doi/abs/10.1073/pnas.0709640104>.

- [19] L. K. Zhou, G. R. Jin, and W. Yang, “Systematic and efficient pseudomode method to simulate open quantum systems under a bosonic environment,” *Phys. Rev. A*, vol. 110, p. 022 221, 2 Aug. 2024. DOI: 10.1103/PhysRevA.110.022221. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.110.022221>.

Appendix A

Action of One Body Operators

An operator that acts only on a single oscillator, labelled q , is written as

$$V_q^{(1)} = \bigotimes_{j=1}^{q-1} \mathbb{I}_j \otimes V_q \otimes \bigotimes_{j=q+1}^M \mathbb{I}_j.$$

A generic matrix-product operator (MPO) for a chain of M oscillators can be expressed as

$$\rho = \sum_{\boldsymbol{\sigma}} A_{\sigma_1} A_{\sigma_2} \cdots A_{\sigma_M} x_{\sigma_1} \otimes x_{\sigma_2} \otimes \cdots \otimes x_{\sigma_M},$$

where the multi-index $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_M)$ collects the local basis labels and $\{x_{\sigma_q}\}$ is a single-site operator basis.

Because $V_q^{(1)}$ modifies only site q , its action on ρ merely replaces the factor x_{σ_q} while leaving the remaining tensor factors unchanged:

$$V_q^{(1)} \rho = \sum_{\boldsymbol{\sigma}} A_{\sigma_1} \cdots A_{\sigma_M} x_{\sigma_1} \otimes \cdots \otimes (V_q x_{\sigma_q}) \otimes \cdots \otimes x_{\sigma_M}.$$

For each basis element the operator V_q acts linearly,

$$V_q x_{\sigma_q} = \sum_{\alpha_q} (V_q)_{\alpha_q \sigma_q} x_{\alpha_q},$$

or, omitting the site index for brevity,

$$V x_{\sigma} = \sum_{\alpha} (V)_{\alpha \sigma} x_{\alpha},$$

with expansion coefficients obtained from the Hilbert–Schmidt inner product,

$$(V)_{\alpha \sigma} = \text{Tr}(x_{\alpha}^{\dagger} V x_{\sigma}).$$

Since the scalar product $A_{\sigma_1} \cdots A_{\sigma_M}$ is unaffected, the effect of V_q can be absorbed into a new local tensor,

$$V_q^{(1)} \rho = \sum_{\boldsymbol{\sigma}} A_{\sigma_1} \cdots \tilde{A}_{\sigma_q} \cdots A_{\sigma_M} x_{\sigma_1} \otimes \cdots \otimes x_{\sigma_M},$$

with the update rule

$$\tilde{A}_{\sigma_q} = \sum_{\alpha_q} (V_q)_{\alpha_q \sigma_q} A_{\alpha_q}.$$

Thus a single-site operator acts locally on an MPO. When such operators $V_q^{(1)}$ are applied successively for $q = 1, \dots, M$, the updates of the local tensors are mutually independent and can therefore be performed in parallel.

Appendix B

Computational Complexity of Compressing MPS

Consider an MPS of $M = NQ$ tensors such that $\chi_0 = \chi_N = 1$ and $\chi_i = \chi$. Let the dimension of the physical leg be d^2 such that $d^2 \geq 2\chi$. After a direct sum the bond-dimension doubles to 2χ . A left-to-right compression via SVD does the following operations (see Figure 2.9)):

- one SVD on a matrix of shape $(d^2, 2\chi)$
- $(M - 2)$ contractions of tensors with shapes $(\chi, 2\chi)$ and $(2\chi, d^2, 2\chi)$
- $(M - 2)$ SVDs of matrices with shape $(d^2\chi, 2\chi)$
- one contraction between tensors of shape $(\chi, 2\chi)$ and $(2\chi, d^2, 1)$

These operations, in the same order as above, amount to the cost: $64d^2\chi^2 + 32(M - 2)\chi^3d^2 + 64(M - 2)\chi^3d^2 + 16\chi^3d^2$.

This ignores the matrix multiplication costs of S and V_h matrices.

Appendix C

Simulation Units

All molecular-dynamics simulations in this thesis are carried out in natural units with

$$\hbar = 1, \quad E_0 = 1 \text{ cm}^{-1},$$

so that energy and frequency are both expressed in cm^{-1} . Under these conventions the Boltzmann constant becomes

$$k_B = \frac{1.380649 \times 10^{-23} \text{ J/K}}{(hc) 1 \text{ cm}^{-1}} \approx 0.695 \text{ cm}^{-1} \text{ K}^{-1},$$

providing a direct mapping between temperature and energy.

The fundamental time unit follows from

$$t_0 = \frac{\hbar}{E_0} = \frac{1.054571817 \times 10^{-34} \text{ J s}}{1.98644586 \times 10^{-23} \text{ J}} \approx 5.31 \text{ ps}.$$

Accordingly, any physical frequency ω_{phys} (in ps^{-1}) is converted to simulation units via

$$\omega_{\text{sim}} = \omega_{\text{phys}} t_0,$$

so that

$$1 \text{ ps}^{-1} \approx 5.31, \quad 1 \text{ fs}^{-1} = 10^3 \text{ ps}^{-1} \times t_0 \approx 5.31 \times 10^3.$$

Likewise, energies specified in cm^{-1} carry over directly as simulation-unit frequencies.