Tensor product methods in numerical simulation of high-dimensional dynamical problems

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

DISSERTATION

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(Dr. rer. nat.)

im Fachgebiet

Mathematik

vorgelegt

von M.Sc. Sergey Dolgov
geboren am 19.03.1988 in Sankt Petersburg

Die Annahme der Dissertation wurde empfohlen von:

1. Priv.-Doz, Dr. Boris N. Khoromskij (Leipzig)
2. Professor Dr. Reinhold Schneider (Berlin)

Die Verleihung des akademischen Grades erfolgt mit Bestehen
der Verteidigung am 20.08.2014 mit dem Gesamtprädikat *magna cum laude*.

Bibliographische Daten

Tensor product methods in numerical simulation of high-dimensional dynamical problems
Universität Leipzig, Dissertation, 2014
154 Seiten, 30 Abbildungen, 241 Referenzen

Abstract

Quantification of stochastic or quantum systems by a joint probability density or wave function is a notoriously difficult computational task, since the solution depends on all possible states (or realizations) of the system. Due to this combinatorial flavor, even a system containing as few as ten particles may yield as many as 10^{10} discretized states. None of even modern supercomputers are capable to cope with this *curse of dimensionality* straightforwardly, when the amount of quantum particles, for example, grows up to more or less interesting order of hundreds. A traditional approach for a long time was to avoid models formulated in terms of probabilistic functions, and simulate particular system realizations in a randomized process. This is mirrored to some extent by a *pointwise sparse* handling of a multivariate function.

Since different times in different communities, *data-sparse* methods came into play. Generally, they aim to define all data points indirectly, by a map from a low amount of representers, and recast all operations (e.g. linear system solution) from the initial data to the effective parameters. In principle, interpolation may bring any realization-based approach into this class. However, the most advanced techniques can be applied (at least, tried) to any given array, and do not rely explicitly on its origin. The current work contributes further progress to this area in the particular direction: tensor product methods for separation of variables.

The separation of variables has a long history, and is based on the following elementary concept: a function of many variables may be expanded as a product of functions of one variable each, for example, $f(x, y, z) = u(x)v(y)w(z)$. Note that on the discrete level, the number of entries in a three-index array, or *tensor*, encoding f, is proportional to the cubed amount of degrees of freedom, required to store univariate functions u, v, w. The idea can now be got across: instead of a huge array generated by f, we will work with its univariate *factors* with much less efforts.

More elaborated generalizations are usually required in practice. This dissertation gives a short overview of the well-established tensor representations: the canonical PARAFAC, Tucker/MCTDH and Hierarchical Tucker/ML-MCTDH decompositions, Matrix Product States/Tensor Train format, as well as the artificial tensorisation, resulting in the Quantized Tensor Train (QTT) approximation method. Any of these formats has its historical niche and adjustments to particular problems. However, the latter three are especially suitable for high-dimensional problems. The Tensor Train (TT) is the simplest one, so it is convenient to formulate and describe tensor product algorithms on its base.

The contribution of the dissertation consists of both theoretical constructions and practical numerical tools for high-dimensional models, illustrated on the examples of the *Fokker-Planck* and the *chemical master* equations. Both arise from stochastic dynamical processes in multiconfigurational systems, and govern the evolution of the probability function in time. Therefore, a special focus is put on *time propagation* schemes and their properties related to tensor product methods. We show that the considered applications result in large-scale systems of linear equations, and prove analytical separable representations of the involved functions and operators. We propose a new combined tensor format (QTT-Tucker), which descends from the TT format (hence TT algorithms may be generalized smoothly), but provides complexity reduction by

an order of magnitude. We develop a robust iterative solution algorithm, constituting most advantageous properties of the classical iterative methods from numerical analysis and alternating density matrix renormalization group (DMRG) techniques from quantum physics. Numerical experiments confirm that the new method is preferable to previously known DMRG/ALS algorithms. It is as fast as the simplest alternating schemes, but as reliable and accurate as the Krylov methods in traditional linear algebra.

Zusammenfassung

Quantifizierung von stochastischen Systemen oder Quantensystem durch eine gemeinsame Wahrscheinlichkeitsdichte oder Wellenfunktion ist bekanntermaßen eine schwierige Rechenaufgabe, da die Lösung von allen möglichen Zuständen (oder Realisierungen) des Systems abhängt. Durch diese kombinatorische Eigenschaft kann schon ein System mit nur zehn Partikeln zu 10^{10} Diskretisierungszuständen führen. Keiner der modernen Supercomputer kommt mit diesem *Fluch der Dimension* auf direktem Wege zurecht, wenn die Anzahl der Quantenpartikel auf, zum Beispiel, mehrere Hundert ansteigt. An klassischer Ansatz war für lange Zeit war die Vermeidung von Modellen, die in Form von probabilistischen Funktionen formuliert sind. Stattdessen simulierte man eine bestimmte Systemrealisierung in einem Zufallsprozess. Das ist bis zu einer bestimmten Art und Weise ähnlich zum Ansatz der *dünnbesetzten* Speicherung einer hochdimensionalen Funktion.

Zu verschiedenen Zeiten kamen in verschiedenen Forschungsbereichen *datenschwache* Methoden ins Spiel. Allgemein betrachtet zielen diese Methoden darauf ab, die Datenpunkte indirekt durch eine Abbildung von einer geringen Anzahl an Representierungen zu definieren. Es werden alle Operationen (z.B. die Lösung des linearen Systems) aus den initialen Daten in die effektiv verwendeten Parameter umgewandelt. Im Prinzip kann Interpolation jeden realisierungsbasierten Ansatz in diese Form bringen. Allerdings können die meisten fortgeschrittenen Techniken auf jedes beliebige Feld angewendet werden ohne dass dabei die Herkunft der Daten beachtet werden muss. Diese Arbeit leistet einen Betrag zum Vorankommen in diesem Bereich in einer bestimmten Richtung: Tensorproduktmethoden zur Trennung der Variablen.

Die Trennung der Variablen hat eine lange Geschichte und basiert auf dem folgenden grundlegenden Konzept: eine Funktion mehrerer Variablen kann unter Umständen als ein Produkt von Funktionen einer Variablen dargestellt werden. Z.B. $f(x, y, z) = u(x)v(y)w(z)$. Man beachte, dass auf der diskreten Ebene die Anzahl der Einträge eines drei dimensionalen Feldes (oder auch *Tensors*), welche die Punkte von f definieren, proportional zur Anzahl der Freiheitsgrade hoch drei ist, die benötigt wird, um die Funktionen u, v und w zu speichern. Die Idee ist nun wie folgt: anstelle der Speicherung des riesigen Feldes, dass durch f erzeugt wird, speichern wir nur die Funktionen u, v und w mit viel weniger Aufwand.

In der Praxis sind höher entwickelte Verallgemeinungen üblicherweise gefordert. Diese Dissertation gibt einen kurzen Überblick über die wohlbekannten Tensorformate: kanonisches Format/PARAFAC, Tuckerformat/MCTDH- und Hierarchisches Format/ML-MCTDH, Matrix Product States-/Tensor Train Format, sowie eine künstliche Tensorisierung, die im Quantisierten Tensor Train Format (QTT-Format) mündet.

Jedes dieser Formate hat seine eigene historisch bedingte Nische und seine eigenen Anpassungen an bestimmte Problemstellungen. Allerdings sind die letzten drei genannten Formate besonders passend für hochdimensionale Probleme. Das Tensor Train Format (TT) ist das einfachste, also ist es passend, die Tensorproduktalgorithmen auf dessen Basis zu beschreiben.

Der Beitrag dieser Dissertation besteht sowohl aus der theoretischen Konstruktion, als auch aus praktischen numerischen Werkzeugen für hochdimensionale Modelle,

die wir auf Beispiele von der *Fokker-Planck*-Gleichung und der *chemischen Mastergleichung* anwenden wollen. Beide Beispiele haben ihren Ursprung in stochastischen dynamischen Prozessen in Mehrkonfigurationssysteme und sind zeitabhängig. Daher legen wir einen besonderen Fokus auf Zeitentwicklungschemas und deren Eigenschaften in Verbindung mit Tensorproduktmethoden. Wir zeigen, dass die betrachteten Anwendungen in einem großen System linarer Gleichungen resultieren, und leiten analytisch separable Darstellungen der beteiligten Funktionen und Operatoren her. Wir schlagen ein neues, kombiniertes Tensorformat (QTT-Tucker-Format) vor, welches vom TT-Format abstammt (daher können die TT-Algorithmen einfach verallgemeinert werden), jedoch eine Reduzierung der Komplexität um eine Größenordnung bietet. Wir entwickeln einen robusten, iterativen Lösungsalgorithmus. Dieser vereint die vorteilhaftesten Eigenschaften klassischer iterativer Methoden der Numerischen Analysis sowie der alternierenden DMRG-Techniken (Density Matrix Renormalization Group Techniken) aus der Quantenphysik. Numerische Experimente bestätigen, dass die neue Methode bekannten DMRG-/ALS-Verfahren vorzuziehen ist. Sie ist so schnell wie die einfachsten alternierenden Schemas, aber gleichzeitig so verlässlich wie die Krylov-Methoden in der traditionellen linearen Algebra.

Acknowledgements

I composed this dissertation from my research, carried out from 2011 to 2014 in a marvelous atmosphere at the Max Planck Institute for Mathematics in the Sciences, Leipzig. I would like to cordially thank my supervisor, Prof. Boris N. Khoromskij, who's attentive mentoring, wise advices and never-ending patience were invaluable on this hard, but exciting way of our collaboration on the topics of my thesis.

I am indebted to the International Max Planck Research School and personally to Prof Stephan Luckhaus, Dr Hayk Mikayelyan and Dr Georgy Kitavtsev for excellent research and educational opportunities at the Max Planck Institute. I am glad to recall discussions and coffee breaks with Dr Venera Khoromskaia, Dr Maryna Kachanovska, Stefan Handschuh, Christian Schindler, Dr Mike Espig, Dr Alexander Litvinenko and many other people at the institute. I am particularly thankful to Stefan Handschuh for his assistance with the German translation of the abstract.

Organizational questions arise in everyday life, but it is hard to imagine how difficult they could be without a constant help of Valeria Hünniger, Heike Rackwitz and other members of the administration, IT and library team.

I am kindly appreciating a fruitful scientific collaboration with Prof Eugene Tyrtyshnikov, Dr Ivan Oseledets from the Institute of Numerical Mathematics, Moscow and Dr Dmitry Savostyanov from the University of Southampton. My special thanks go to Dmitry for his far not only scientific support and help during these years.

At last, but not the least, I am warmly thankful to my family for only not scientific support and motivation, that is sometimes much more important...

Contents

Introduction

This dissertation is devoted to the numerical solution of high-dimensional problems via tensor product methods. What we mean by the high-dimensional problems, and how they arise in practice? The linear algebra considers vectors and matrices, and by "dimension" one understands usually the size, or the number of elements in a vector. It may be classified as "high", compared to an available computer memory, for example. However, the term "high-dimensional" is reserved for something different. As the main applications we identify quantum and probabilistic physical models, such as the Fokker-Planck, master or Schroedinger equations. To comprehend why are they high-dimensional, we begin with the following introductory example.

Suppose not a particular vector, but instead a *class*, or a family of vectors is defined, such that the elements obey certain independent computational rules. Let the rule for each element be able to throw a finite number of distinct values. Then all *instantiations* of such a class may be collected into another vector: we simply enumerate all possible combinations of outcomes one by one. How many of them are there? If each initial element may take n values, there are already $n \cdot n = n^2$ combinations of two of them, and $n \cdot n \cdots n = n^d$ realizations of a d-elements class. To imagine what a huge number it may be: just 80 elements (requiring as few as 640 bytes to store them with double precision) with only 10 possible values for each yield 10^{80} combinations – the qualitative estimate of the number of atoms in the universe. This toy example illustrates two key points of the current work: the way how we derived an enormous amount of values from a relatively moderate number of initial items will arise in our main applications, and the conception and understanding that we may store not all 10^{80} instances, but just 80 rows of 10 values, defining the governing rules, will lie behind our computational methods.

The idea of a space of instantiations, in more rigorous terms, the *state space* is a cornerstone in quantum and stochastic world. A many-body system may be described by an ordinary differential equation, which stands for the evolution of d coordinates of particles positions, or other degrees of freedom. Typical issues emanate from a nonlinear form of the physical laws, but do not concern the computer storage of the solution, since nowadays even a workstation can handle a billion of unknowns with ease.

However, this is no more true, as soon as the randomness is introduced. Nobody can definitely predict the position of a randomly walking particle. But what is possible to quantify, is the *probability* that the particle visits a prescribed region at the current moment.

Now a natural idea is to compare such probabilities for different regions. So we have

to split the whole space to the enumerated cells (possibly infinitesimally small), and assign a function, which returns a quantitative probability value for a given cell number. To store the probabilistic description in a computer, we have to consider a finite amount of splitting regions. Suppose the particle lives in an ordinary three-dimensional isotropic space. Then there is no preferable direction, and it looks reasonable to select some amount n of region labels for each of three axes. In total, we end up with n^3 probability values for three independent coordinates.

If we would like to describe simultaneously several interacting but randomly driven bodies, the joint probability density function will live in a d-dimensional space, and typically require n^d discretization values, where d is the amount of all coordinates of all particles. Therefore, by the "dimension" we mean the number of *configuration coordinates* d in the system's state space, while n stands for the amount of possible points in each coordinate. Certainly, even the cases $d = 3$ or $d = 2$ may be considered as "high-dimensional", if n is very large. The situation becomes even more dramatic if the physical or mathematical model stipulates to work with d of the order of tens, hundreds and more. Unless the ridiculous $n = 1$ is chosen for most of coordinates, the exponential complexity growth with the dimension prevents straightforward calculations on any supercomputer. For example, in quantum world, spin-1/2 particles (electrons and nuclei can be simplified to "just spins" in appropriate conditions, such as the magnetic resonance experiments) possess only $n = 2$ states, "spin up" and "spin down", for each particle. However, a simple linear chain of $d = 100$ interacting spins (which is usually considered as a toy benchmark problem in quantum physics) is described by a wavefunction with $2^{100} \sim 10^{30}$ unknowns.

This phenomenon of the exponential contribution of the number of configuration coordinates is called the *curse of dimensionality* since [16]. Therefore, the only way to treat such problems is to handle some *effective* discrete information, which requires much less storage than the initial bulk n^d. Along the line with this suggestion, it is natural to expect that we do not actually need all n^d numbers. Quantum, as well as stochastic modeling is usually performed to detect accurate *statistics*, or *observables*, such as the mean, dispersion, energy and other quantities, which are themselves low-dimensional, and only *agglomerate* high-dimensional data in some way.

Among all approaches that involve the reduction of the information volume, the so-called *data-sparse* methods, we may identify problem-oriented and general methods. The first class assumes and utilizes heavily specific properties of the problem, such as the smoothness of involved functions, particular rules how to evaluate statistical quantities, and so on. The well established state of the art includes Monte Carlo methods [178, 65, 1, 12, 57] (in lots of versions and improvements like Quasi MC [184, 220, 171], Markov Chain MC [107], etc.), Smolyak's sparse grids [221, 31, 94, 68], radial basis functions [29, 30], wavelet and other best N-term representations [37, 22], as well as special reduction techniques and bases devised from the physical intuition about particular problems. As one of the most successful approaches of the latter type, we may mention the Gaussian Type Orbitals [58, 110] with extensions to grid-based quadratures in more general bases [151, 152, 153, 137, 138, 140], and Coupled Cluster [13, 172] for the Hartree-Fock calculations, or State Space Restriction [167, 114] for spin dynamics.

General methods do not use the physical meaning of the problem or the outputs explicitly, relying instead on purely mathematical tools to represent the whole high-dimensional set via a wisely chosen mapping from a feasible amount of data. For the sake of justice, it is worth to note that a proof of the *usefulness* of such methods may often require an insight into the problem details. Moreover, specialized methods have more chances to beat general techniques in performance tests. However, a possibility to deliver *any* portion of information (possibly approximate) about the high-dimensional object on request, and a *black-box* interface for the inputs reserve an important niche for general data-sparse methods. For example, one may try such an approach on any new problem without a substantial re-development of the procedures to see whether it works in principle, or cross-verify some other (probably more specific and fast) method.

A remarkably strong member of the general data reduction class is the *tensor product* concept for the separation of variables, developed rapidly in the recent decade, and investigated further in the current work. The common idea is, given a large array with many indices, to represent (or approximate) it with a combination of products and additions of smaller arrays with few degrees of freedom. Importantly, there are particular factorizations and methods, that require only the initial data and use only algebraic tools (such as the singular value decomposition) to determine the reduced set of parameters. Obviously, they do not depend on the underlying physical model of the data, though the actual efficiency of the reduction does depend on the functional properties such as the smoothness. Interestingly, narrowing the conditions imposed on the data, we may even notice that different methods behave similarly in both theory and practice, see, for example, a comparison of sparse grids and separation of variables [95].

The key point in the separation of variables is the representation of a multi-variate function (or its discrete counterpart, a *tensor*), by a product of univariate terms, i.e.

$$x(i_1, \ldots, x_d) = x^{(1)}(i_1)x^{(2)}(i_2)\cdots x^{(d)}(i_d).$$

If this *direct product* decomposition does not hold exactly, one may consider it as a *dictionary*, and approximate a more general object via a combination of direct products. A widely used computational approach is the greedy methods. The fundamental concept is outlined, for example, in the book [226]. A significant contribution to the *tensor product greedy* framework, where a linear combination of direct products is recovered, was brought by [5, 168, 186, 64, 25, 33, 79]. This approach reduces the error (e.g. $\|x-\tilde{x}\|^2$ or another function such as the residual or Rayleigh quotient), associated with the problem, by retrieving subsequently the best (or quasi-best) approximations in the form of the direct product. It is possible to establish a convergence analysis for the greedy methods (see the references given above), provided that each optimization of the direct product components is conducted with a guaranteed accuracy. However, this requirement is hard to satisfy in practice. First, the residual becomes more and more oscillatory in the latter iterations, and its direct product approximation (even optimal) becomes of poorer and poorer quality. Second, the optimality of the approximation is harder to achieve numerically. This is usually the reason why greedy tensor methods get stuck at some error level, which can be unsatisfactory large.

A reliable alternative, and an important part of justification of the tensor product decompositions comes from the *analytical* separable constructions, which are typically

written as convergent series of direct products. One of the most remarkable examples of operators, written directly as a sum of direct products, is the inverse Laplace operator [71, 72, 100, 101] and the related Green functions [142, 146].

The sum of R separable components is called the rank-R *canonical polyadic* decomposition,

$$x(i_1, i_2, \ldots, i_d) = \sum_{\alpha=1}^{R} x_\alpha^{(1)}(i_1) x_\alpha^{(2)}(i_2) \cdots x_\alpha^{(d)}(i_d).$$

Besides the greedy methods, one may try a general error minimization method to fit the elements of the canonical *factors* $x^{(1)}, \ldots, x^{(d)}$, such as the Newton [160, 60, 133] or alternating least squares [106, 34, 28, 38, 23, 24]. However, in case of $R > 1$ and $d > 2$, the error optimization may become an ill-posed problem [42]: it is possible to construct a sequence such that the error converges to zero, but the limit (the minimizer) would have an $\infty - \infty$ uncertainty.

The low-rank *matrix factorization* ($d = 2$) has a crucial difference: the low-rank matrix fitting problem is well-posed, and can be solved efficiently using the singular value decomposition (SVD) [81]. The SVD delivers a minimizer of the error in the euclidean norm on the set of rank-R matrices. Moreover, it is incredibly robust numerically [80], and is highly optimized during the decades of development of the LAPACK library.

If we stick to higher dimensions, several generalizations of the SVD decomposition for many variables have been developed. One idea is to compute singular value decompositions independently for each coordinate. This results in the *Tucker* [228] representation,

$$x(i_1, \ldots, i_d) = \sum_{\gamma_1, \ldots, \gamma_d = 1}^{r_1, \ldots, r_d} x^{(c)}(\gamma_1, \ldots, \gamma_d) x_{\gamma_1}^{(1)}(i_1) \cdots x_{\gamma_d}^{(d)}(i_d).$$

Note that each *Tucker factor* $x^{(k)}$ possesses its own rank index γ_k, contrarily to the shared enumerator α in the canonical representation. This independence allows to solve the approximation problem using the so-called *Higher Order SVD* (HOSVD) algorithm [39, 40, 41], which computes Tucker factors as senior singular vectors of certain matrices, associated with the initial array x. This gives reliability and a quasi-optimal accuracy/rank ratio. The extension of this technique to large-rank canonical inputs, the so-called *Reduced HOSVD* (RHOSVD) was proposed in [151].

The alternating least squares approximation can be employed for the Tucker elements as well. This technique was proposed mainly as a low-rank model fitting tool, and traces back to [165], see also [40, 119]. One may say that the HOSVD was perhaps motivated by the *principal component analysis*, but the general *data compression* purpose was considered at the second place.

The latter was put into stream when the tensor product representations had been targeted to the structured solution of multivariate PDEs [23]. Since then, the key ingredient $r \ll n$ was discovered for smooth functions, both numerically and analytically, with the aid of the theory of the polynomial interpolation, see [72, 142, 100]. This approach is also applicable for a small amount of distinct singularities [150].

Moreover, the functional insight stimulated elaboration of the computational algorithms, including combinations of several techniques. For example, the multigrid scheme provided a significant acceleration of the alternating least squares iterations [151], the mixed canonical-Tucker representation was suggested in [142, 150, 151]. Important results stemmed from the application-related focus in integral equations [210, 187, 194, 150, 146] and electronic structure calculations [153, 151, 66, 136, 143, 152, 196, 137, 140, 139, 138, 211, 195, 82].

The references above consider mostly three-dimensional problems. In higher dimensions, the Tucker *core* $x^{(c)}$ still suffers from the curse of dimensionality, since it contains $\mathcal{O}(r^d)$ elements. The alternative came from *recurrent* two-dimensional factorizations. The idea is as follows: we introduce a reduced basis in one variable, then join it with another index, determine a reduced basis in two variables, and so on. This procedure can be conducted in accordance with a balanced binary tree, and result in the so-called *Hierarchical Tucker* (HT) [105, 90, 166] decomposition, or along the linear tree, which gives the Tensor Train (TT) [197, 188, 191] construction, or even in more general Tensor Tree Networks (TTN) [63] with multi-branched dimension trees. The TT decomposition was discovered and used since longer for many problems, which is reflected by many independent names existing for this construction: *valence bond states* [2], *matrix product states* (MPS) [62, 158, 201] and *density matrix renormalization group* (DMRG) [237] in condensed matter quantum physics, before the "tensor train" term appeared in 2009 in numerical linear algebra [197, 191].

In principle, the Tucker decomposition may be also considered as a d-branched TTN. We see that the general framework, the representation of a multivariate function by a polylinear combination of univariate functions, is common for all tensor decompositions, but the particular rules how to assemble the initial tensor, the so-called *tensor formats*, may differ significantly in both formulation and computational properties.

It is natural to expect that a particular tree will be the most efficient for a particular problem. Nevertheless, more complicated tensor networks require more complicated and lengthy descriptions. Fortunately, the main concepts do not depend explicitly on the tree structure, so we stick to the TT representation to make the presentation more simple and elegant.

The Tensor Train format may be placed between the canonical and Tucker decompositions,

$$x(i_1, \ldots, i_d) = \sum_{\alpha_1, \ldots, \alpha_{d-1}=1}^{r_1, \ldots, r_{d-1}} x^{(1)}_{\alpha_1}(i_1) x^{(2)}_{\alpha_1, \alpha_2}(i_2) \cdots x^{(d-1)}_{\alpha_{d-2}, \alpha_{d-1}}(i_{d-1}) x^{(d)}_{\alpha_{d-1}}(i_d).$$

Each *block* $x^{(k)}$ in the right-hand side is a three-dimensional array of size $\mathcal{O}(nr^2)$, hence the total storage $\mathcal{O}(dnr^2)$ is able to remove the curse of dimensionality, provided that the rank r is moderate.

Discretization of multidimensional PDEs may require fine grids, resulting in large *mode sizes* n. Interestingly, a further cost reduction may be achieved in the very same TT framework. The linear contribution of the dimension to the complexity of the TT format tempts to *increase* the number of variables. The *quantization* suggests to split each index $i_1, \ldots, i_d$ to sub-indices according to the prime factors of n. Applying the

TT approximation to the resulting tensor with $\mathcal{O}(d \log n)$ dimensions, but low mode sizes ($\sim$ 2–3), one ends up with the so-called *Quantized TT* (QTT) format [144, 148]. It is important that the rank bounds for many quantized one-dimensional functions can be estimated rigorously [148, 192], and the logarithmic storage asymptotic with respect to the initial data volume takes place. The same approach is applicable for compression of matrices [189].

Contrarily to the Tucker and one-dimensional QTT formats, general smoothness assumptions for multidimensional functions lead to pessimistic rank-vs-accuracy estimates in the TT format, see e.g. [215], stating a $\mathcal{O}(|\log \varepsilon|^d)$ term. Fortunately, the reality behaves usually much better. *Combined* tensor formats may yield an additional compression and speed-up. For example, the *Extended TT* [198] constitutes the Tucker format with the TT representation of the Tucker core, and the *QTT-Tucker* decomposition [44] employs also the QTT approximation for Tucker factors.

Any tensor structure requires numerical methods for data approximation and other operations. Perhaps it is the strong focus on applications in physics community that gave us many versatile computational algorithms, especially in the TT/MPS format. First, similarly to the Tucker HOSVD method, the TT representation can be computed using singular value decompositions with a guaranteed accuracy. Second, the alternating direction concept developed in quantum physics is essentially more powerful than the alternating least squares for e.g. Tucker format. Numerical Renormalization Group (NRG) and *Density Matrix Renormalization Group* (DMRG) algorithms were extensively used to simulate wavefunctions of spin systems in tensor product formats since 1970's [241], and many impressive modifications and improvements have been developed, which were later adopted in the numerical analysis community as well [115]. A far not complete list of references includes [237, 238, 200, 239, 124, 234, 240, 236, 202, 216, 217]. The power of alternating methods stems from the *linearity* of tree-based tensor formats with respect to a fixed block of entries. Therefore, approximation, linear system or eigenvalue problems, posed for initial tensors, recast to the same formulations on the elements of the formats, which may be solved using standard methods. This stays in sharp contrast with the simultaneous optimization of all format entries at once, which may be a substantially nonlinear and nonconvex problem.

However, the latter phenomenon impedes the reliability of the alternating methods as well. Even the simplest error function $\|x - x_\star\|^2$, recast to the tensor format elements of x, may have numerous local minima. The alternation brings the speed, but it is also a drawback. Since only a part of the format is considered in each step, the simple alternating linear schemes and DMRG algorithms are likely to deliver a local but not the global error minimizer. In many cases this is not a desired result, since we would like to solve an initial physical problem posed in the high-dimensional space. This motivates us to take another approach into account.

As soon as the format is equipped with a reliable approximation procedure, which allows to compress any data with a quasi-optimal storage for a given accuracy threshold, one may think about implementation of classical iterative algorithms with the data-sparse *arithmetics*, such as the solvers for linear systems [210, 104, 195, 162, 156, 141, 145, 163, 6, 45, 9] and partial eigenproblems [170, 164, 116, 102, 176]. We may think about tensor formats in a similar way as about numbers in a computer: additions and

multiplications of the initial data recast to the tensor format blocks, these operations typically increase the amount of elements, and the approximation, like the numerical *rounding*, keeps the storage limited.

The question is, however, what is this particular storage requirement. For numbers, a fixed mantissa length always ensures a guaranteed accuracy. This is not the case with the tensor product methods: the storage depends on both the accuracy and a weakly justified "structure" of the data. Unfortunately, the classical (e.g. Krylov) methods are essentially based on the residual, and suffer from the same difficulty as the greedy methods: the better is the solution, the worse is the structure of the error and other auxiliary vectors, which require either a poor approximation, or a large storage of their tensor formats. Relaxation techniques, developed in the theory of inexact Krylov methods [219], improve the situation to some extent, admitting a rough rounding in the latter iterations, but the methods are still far from being reliable in general.

The *main computational method*, suggested and employed in this dissertation, combines the advances of the alternating tensor optimization algorithms and the classical approximate iterative schemes in terms of the initial tensor elements [52, 53]. During the alternating DMRG iteration, we augment (*enrich*) the tensor format of the solution by the tensor format of the approximate residual. This leads to a remarkable cooperation of the classical and alternating techniques. Since the solution is being improved by the alternating optimization, accurate results are delivered even with a very crude approximation of the residual (and no more Krylov vectors are needed). On the other hand, injection of the global residual information into the local steps of the alternating process supports the latter with proper descent directions, helping it to escape from spurious local minima. As was mentioned, such traps are a notorious problem in variational methods on tensor manifolds. Contrarily, for the new Alternating Minimal Energy (AMEn) method we prove the geometrical convergence with respect to the global tensor elements, similarly to the steepest descent technique. The history behind the name AMEn can be illustrated as follows.

$$\text{Minimal Residual} \rightarrow \begin{array}{c} \textbf{Minimal Energy} \\ \text{(steepest descent)} \end{array} \rightarrow \begin{array}{c} \text{Alternating} \\ \text{Minimal Energy} \end{array}$$
$$\textbf{Alternating} \\ \text{Least Squares} \quad \nearrow$$

Its practical convergence rate is much faster than the theoretical steepest descent estimate, which makes this method reliable even for non-symmetric linear systems, cf. the FOM theory [208].

Applications, considered in this dissertation, remind our discussion about the random walking particle and the related probability distribution. We apply tensor product methods to differential and difference equations, governing the probability functions. A system described with continuous coordinates, affected by stochastic forces, yields the so-called *Fokker-Planck equation*, the high-dimensional diffusion-convection equation on the probability density function [206]. As a particular engineering problem, we consider numerical modeling of liquids with dissolved polymer molecules [26]. Some remarkable non-Newtonian properties of such solutions result from the micro-mechanics of polymer chains, experiencing the Brownian motion of the solvent molecules.

Another application has a crucial role in accurate stochastic simulation of cellular processes, life cycles of viruses and other micro-biological processes, where random fluctuations are significant. This is the case if the amount of reacting molecules is very small compared to the volume of the whole system, and collisions between the molecules occur occasionally. The chemical kinetics becomes governed by the jump Markov process, which may be again described by a multi-variate probability function. Contrarily to the Fokker-Planck model, the configuration states are now discrete, so that the probability function obeys the difference *(chemical) master equation* [232].

Both mentioned equations describe the evolution of the probability function in time. Therefore, as the third application, we show how a general *time propagation scheme*, such as Euler or Crank-Nicolson, can benefit from tensor product methods. We consider time variable as an additional dimension. Then, the QTT approximation in time leads to the logarithmic complexity in the number of time steps.

The presented problems end up with very large-scale linear systems. We establish several analytical separable constructions, showing that the matrices and right-hand sides in these systems can be given or efficiently approximated in tensor product representations, and in particular in the newly developed QTT-Tucker format. The issue is now to compute the solution. In the numerical experiments, we demonstrate that the new iterative optimization algorithm is more fast and accurate than the previously known methods, and may finally pretend to be a "black-box" tool for various problems, which are suitable for the separation of variables concept in principle.

Organization of the dissertation

In Chapter 1 we introduce our three main applications: a stochastic dynamical system and the related Fokker-Planck equation, a stochastic chemical kinetics and the chemical master equation, and the simultaneous space-time discretization scheme for dynamical equations, prepared for the encapsulation into tensor product solution schemes, providing ultimately a logarithmic complexity reduction with the aid of the QTT approximation in time variable.

Chapter 2 is devoted to the description of different sides of the separation of variables approach. It overviews existing tensor decompositions: the canonical/CP, Tucker, Hierarchical Tucker and the Matrix Product States/Tensor Train formats, as well as related algorithms and their properties. In the end, it proposes a new combined tensor representation (QTT-Tucker), agglomerating both analytical and practical insights from the Tucker, TT and QTT formats.

In Chapter 3 we join together the applied problems outlined in the first chapter, and the tensor product techniques from the second chapter. We prove several explicit low-rank TT/QTT tensor representations of the matrices and vectors (tensors), relevant to the differential and difference operators and some illustrative functions in our applications. Among them are the anisotropic diffusion operator, discretized gradients, nearest-neighbor interacting systems and the Gaussian function.

Chapter 4 presents tensor product solvers: classical iterative methods equipped with tensor roundings, alternating iterative optimization techniques developed in quantum physics (DMRG) and numerical mathematics (Alternating Least Squares/Linear

Scheme) communities, and finally the new improved algorithm, which supports the traditional alternating optimization scheme with the classical gradient direction (residual) from the global system. We analyze its convergence, prove the geometric convergence rate, and discuss some practical aspects.

Chapter 5 serves as a confirmation of all said before. We consider several relevant biological systems governed by the chemical master equation, as well as the polymer micro-model described by the Fokker-Planck equation, investigate their features related to the tensor product conception, verify and compare computational algorithms. By various numerical experiments we confirm that the tensor formats and methods are useful as efficient and accurate tools for the discussed applications.

In the end, we summarize the main points and observations in the Conclusion.

Chapter 1

Multidimensional partial differential equations

1.1 Stochastic dynamical systems

1.1.1 Stochastic differential models and Fokker-Planck equation

A mathematical model in terms of ordinary differential equations is probably the most intuitive and understandable description of physical phenomena in nature. The Newton's second law is one example, which everybody can literally feel in hand. In a stochastic dynamical system some components (e.g. forces) emanate from stochastic processes.

How a stochastic force can occur? We begin with an illustrative example, following [206]. Imagine a particle immersed in a fluid. It experiences the friction Stokes' force, and a force arising from collisions with fluid molecules. Therefore, the equation of motion writes

$$m\frac{dv}{dt} = -\alpha v + F(t),$$

where $v(t)$ is the velocity of the particle, m is its mass, α is a friction coefficient, and F is a resultant collisional force. This simple equation has only one drawback: one will usually never manage to solve it exactly. The reason is that it must be coupled with similar equations of motions for all $\sim 10^{23}$ molecules in a fluid.

This phenomenon is known as the *Brownian* motion, and we have to employ the stochastic description, the usual way in thermodynamics. Instead of a certain system, we consider an (Gibbs) ensemble of them. The force F is then a stochastic process, which may be characterized only on average, for example,

$$\langle F(t)\rangle F = 0, \quad \langle F(t)F(t')\rangle = f\delta(t-t').$$

In turn, we can not predict and hence are not interested in the velocity v in a particular realization of the particle and the fluid. However, the statistical properties make sense: we might like to simulate the average velocity $\langle v\rangle$, or a probability that the particle will move with a velocity lying in a prescribed *interval*. If the length of this interval is

infinitesimally small, it holds usually that this probability is proportional to the length, i.e.

$$\mathcal{P}(v_* \leqslant v(t) \leqslant v_* + \delta v) = \psi(v_*, t)\delta v + o(\delta v).$$

In other terms, the probability is differentiable w.r.t. the *system configuration coordinate* v. The derivative $\psi(v)$, called the *probability density*, or distribution function, will be the main interest in this section, because it allows to compute other statistics, for example, any average function on v,

$$\langle g(v(t)) \rangle = \int_{-\infty}^{\infty} \psi(v,t) g(v) dv.$$

Note that ψ itself does not involve stochasticity. Therefore, an equation we are going to formulate to describe ψ directly is deterministic – and we may employ standard tools of analysis. But before we proceed to this question, let us see how the stochastic system and probability density embody the example from Introduction of variable vector elements. One initial component v defines now a continual set of values $\psi(v)$, which can be discretized at n points. Taking in addition several time points, we obtain a two-dimensional array $[\psi(v_i, t_p)]_{i,p}$ with n^2 elements. The same will happen if we consider two particles with the corresponding density function $\psi(v, w)$, and for d degrees of freedom one needs to handle a d-dimensional function.

We will come to the d-dimensional discretization later, now supply the high-dimensional probability density and its governing equation with more details, according to [206].

Suppose a stochastic process $\boldsymbol{\xi}(t) \in \mathbb{R}^d$, and time points $t^1, \ldots, t^n$ are given. The probability to find a system in corresponding state volumes $[\boldsymbol{q}^1, \boldsymbol{q}^1 + \boldsymbol{dq}^1], \ldots, [\boldsymbol{q}^n, \boldsymbol{q}^n + \boldsymbol{dq}^n]$[1] writes as

$$\begin{aligned} &\psi(\boldsymbol{q}^n, t^n, \ldots, \boldsymbol{q}^1, t^1) \boldsymbol{dq}^1 \cdots \boldsymbol{dq}^n, \quad \text{where} \\ &\psi(\boldsymbol{q}^n, t^n, \ldots, \boldsymbol{q}^1, t^1) = \langle \delta(\boldsymbol{q}^1 - \boldsymbol{\xi}(t^1)) \cdots \delta(\boldsymbol{q}^n - \boldsymbol{\xi}(t^n)) \rangle. \end{aligned}$$

We may also define a conditional probability of the system transition from the history $\boldsymbol{q}^1, \ldots, \boldsymbol{q}^{n-1}$ to $\boldsymbol{q}^n$,

$$\begin{aligned} P(\boldsymbol{q}^n, t^n | \boldsymbol{q}^{n-1}, t^{n-1}, \ldots, \boldsymbol{q}^1, t^1) &= \langle \delta(\boldsymbol{q}^n - \boldsymbol{\xi}(t^n)) \rangle |_{\boldsymbol{\xi}(t^{n-1}) = \boldsymbol{q}^{n-1}, \ldots, \boldsymbol{\xi}(t^1) = \boldsymbol{q}^1} \\ &= \frac{\psi(\boldsymbol{q}^n, t^n, \ldots, \boldsymbol{q}^1, t^1)}{\psi(\boldsymbol{q}^{n-1}, t^{n-1}, \ldots, \boldsymbol{q}^1, t^1)}. \end{aligned}$$

For Markov processes, the transition probability P does not depend on the whole history, but only on the previous state $t^{n-1}, \boldsymbol{q}^{n-1}$. So we may omit n and write a two-states expansion

$$P(\boldsymbol{q}, t' | \boldsymbol{q'}, t) = \frac{\psi(\boldsymbol{q}, t', \boldsymbol{q'}, t)}{\psi(\boldsymbol{q'}, t)}.$$

[1] if $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_d)$ is a multi-dimensional vector, these intervals read as hypercubes, e.g. $[q_1^1, q_1^1 + dq_1^1] \otimes \cdots \otimes [q_d^1, q_d^1 + dq_d^1]$. The same holds for differentials, $\boldsymbol{dq}^1 = dq_1^1 \cdots dq_d^1$, and delta functions $\delta(\boldsymbol{q} - \boldsymbol{s}) = \delta(q_1 - s_1) \cdots \delta(q_d - s_d)$.

Choosing an infinitesimal time step $t' = t + \delta t$, and integrating over $\boldsymbol{q'}$, obtain

$$\psi(\boldsymbol{q}, t + \delta t) = \int P(\boldsymbol{q}, t + \delta t | \boldsymbol{q'}, t) \psi(\boldsymbol{q'}, t) \boldsymbol{dq'}. \tag{1.1}$$

Define the $(j_1 + \cdots + j_m)$-order moments

$$\begin{aligned} M^m_{j_1,\ldots,j_m}(\boldsymbol{q'}, t, \delta t) &= \langle (\xi_{j_1}(t + \delta t) - \xi_{j_1}(t)) \cdots (\xi_{j_m}(t + \delta t) - \xi_{j_m}(t)) \rangle \\ &= \int (q_{j_1} - q'_{j_1}) \cdots (q_{j_m} - q'_{j_m}) P(\boldsymbol{q}, t + \delta t | \boldsymbol{q'}, t) \boldsymbol{dq}, \end{aligned}$$

where $j_1, \ldots, j_m \in \{1, \ldots, d\}$, possibly repeating. Now we may derive the Kramers-Moyal expansion. First, rewrite

$$P(\boldsymbol{q}, t + \delta t | \boldsymbol{q'}, t) = \int \delta(\boldsymbol{s} - \boldsymbol{q}) P(\boldsymbol{s}, t + \delta t | \boldsymbol{q'}, t) \boldsymbol{ds},$$

and expand the delta function into the Taylor series at $\boldsymbol{s} = \boldsymbol{q'}$,

$$\begin{aligned} \delta(\boldsymbol{s} - \boldsymbol{q}) &= \delta(\boldsymbol{q'} - \boldsymbol{q} + \boldsymbol{s} - \boldsymbol{q'}) \\ &= \sum_{m=0}^{\infty} \sum_{\boldsymbol{j}} \frac{1}{m!} (s_{j_1} - q'_{j_1}) \cdots (s_{j_m} - q'_{j_m}) \frac{\partial^m}{\partial q'_{j_1} \cdots \partial q'_{j_m}} \delta(\boldsymbol{q'} - \boldsymbol{q}) \\ &= \sum_{m=0}^{\infty} \sum_{\boldsymbol{j}} \frac{1}{m!} \frac{(-1)^m \partial^m}{\partial q_{j_1} \cdots \partial q_{j_m}} (s_{j_1} - q'_{j_1}) \cdots (s_{j_m} - q'_{j_m}) \delta(\boldsymbol{q'} - \boldsymbol{q}), \end{aligned}$$

where $\boldsymbol{j} = (j_1, \ldots, j_m)$, summation for each j_k goes from 1 to d. From the last two equations obtain

$$P(\boldsymbol{q}, t + \delta t | \boldsymbol{q'}, t) = \sum_{m=0}^{\infty} \sum_{\boldsymbol{j}} \frac{1}{m!} \frac{(-1)^m \partial^m}{\partial q_{j_1} \cdots \partial q_{j_m}} M^m_{j_1,\ldots,j_m}(\boldsymbol{q}, t, \delta t) \delta(\boldsymbol{q'} - \boldsymbol{q}),$$

and plug this into (1.1):

$$\psi(\boldsymbol{q}, t + \delta t) = \psi(\boldsymbol{q}, t) + \sum_{m=1}^{\infty} \sum_{\boldsymbol{j}} \frac{1}{m!} \frac{(-1)^m \partial^m}{\partial q_{j_1} \cdots \partial q_{j_m}} M^m_{j_1,\ldots,j_m}(\boldsymbol{q}, t, \delta t) \psi(\boldsymbol{q}, t).$$

To deduce the time derivative, assume that the moments expand w.r.t. δt,

$$M^m_{j_1,\ldots,j_m}(\boldsymbol{q}, t, \delta t) = D^m_{j_1,\ldots,j_m}(\boldsymbol{q}, t) m! \cdot \delta t + o(\delta t),$$

then the final Kramers-Moyal series in the limit $\delta t \to 0$ writes

$$\frac{\partial \psi(\boldsymbol{q}, t)}{\partial t} = \sum_{m=1}^{\infty} \sum_{\boldsymbol{j}} \frac{(-1)^m \partial^m}{\partial q_{j_1} \cdots \partial q_{j_m}} D^m_{j_1,\ldots,j_m}(\boldsymbol{q}, t) \psi(\boldsymbol{q}, t). \tag{1.2}$$

The *Fokker-Planck* (after Fokker [67] and Planck [203], proposed it for the description of the Brownian motion), or Forward Kolmogorov equation is the Kramers-Moyal expansion, truncated after the second term:

$$\frac{\partial \psi(\boldsymbol{q}, t)}{\partial t} = -\sum_{k=1}^{d} \frac{\partial}{\partial q_k} C_k(\boldsymbol{q}, t) \psi(\boldsymbol{q}, t) + \sum_{k,m=1}^{d} \frac{\partial^2}{\partial q_k \partial q_m} D_{k,m}(\boldsymbol{q}, t) \psi(\boldsymbol{q}, t). \tag{1.3}$$

One may see that it is a second-order convection-diffusion equation, with

- *drift* coefficients $C_k(\boldsymbol{q},t) = \lim\limits_{\delta t\to 0} \frac{1}{\delta t}\langle \xi_k(t+\delta t) - \xi_k(t)\rangle|_{\boldsymbol{\xi}(t)=\boldsymbol{q}}$, and
- *diffusion* coefficients
  $D_{k,m}(\boldsymbol{q},t) = \frac{1}{2}\lim\limits_{\delta t\to 0} \frac{1}{\delta t}\left\langle \left(\xi_k(t+\delta t) - \xi_k(t)\right)\left(\xi_m(t+\delta t) - \xi_m(t)\right)\right\rangle|_{\boldsymbol{\xi}(t)=\boldsymbol{q}}$.

By construction, the diffusion matrix is symmetric and semi-positive definite, i.e. (1.3) is a proper *parabolic* equation. Moreover, it can be seen as a continuity equation

$$\frac{\partial \psi}{\partial t} + \nabla S = 0,$$

where the probability flux components are defined as follows,

$$S_k = C_k\psi - \sum_{m=1}^{d} \frac{\partial}{\partial q_m} D_{k,m}\psi.$$

If the configuration domain $\boldsymbol{q} \in \Omega$ is the whole space, or the flux vanishes at the boundary, it leads to the conservation of the probability normalization $\int \psi(\boldsymbol{q},t)\boldsymbol{dq}$ and non-negativity of the solution, provided the initial state $\psi(\boldsymbol{q},0)$ was nonnegative and of a finite norm.

The justification of the Fokker-Planck equation is provided by the Pawula theorem, which says that the Kramers-Moyal expansion (1.2) either stops after the second term (many physical processes, like the Brownian motion), and in this case the Fokker-Planck equation models the probability density exactly, or contains infinite number of terms. The latter case is more difficult (even higher-order terms may yield a rather slowly convergent approximation), and is not considered in the current work.

1.1.2 Discretization of the Fokker-Planck equation

We now focus on the numerical solution of the equation (1.3). First, recall how the curse of dimensionality arises. The accuracy of finite element methods, for example, is governed by the number of basis functions n introduced in each coordinate q_k. In most cases neither of components of C or diagonal submatrices of D vanishes, so all dimensions have approximately equal importance. Hence, it is reasonable to introduce comparable numbers of points in each variable, but it results in $\mathcal{O}(n^d)$ finite element coefficients or grid values required to define the solution $\psi(\boldsymbol{q},t)$. We will address this issue by an approximate indirect representation of n^d elements via a much smaller amount of data in tensor product formats. In the next subsection we show an example, where such a low-parametric format may be derived analytically.

The particular discretization scheme must be chosen in accordance with the Fokker-Planck coefficients C and D. Most difficulties arise from the convection term: the simple finite difference or finite element methods suffer from spurious oscillations if the convection coefficient is significantly larger than the diffusion, and the grid is not fine enough. As we will see, the convection coefficient may be even infinite, and though the solution of the exact equation is well defined, no grid refinement can help to make the finite difference scheme stable.

However, since all data will be stored in a compact form, and the solution exhibits usually a high order of smoothness, a method of choice can be the spectral discretization methods [227]. This approach yields dense stiffness matrices and hardly adapts to an arbitrary geometry of the domain. Fortunately, in our case the problem is posed in a direct product of low-dimensional domains (the whole space, for example), while the storage and computational complexities are effectively one-dimensional. Thus, dense, but small (due to a rapid convergence) differentiation matrices do not produce difficulties. Moreover, a great advantage is that even essentially convection dominating equations may be properly resolved, with the maximum principle persisting already on rather moderate grids.

We will identify appropriate discretization schemes in more details in the particular examples. Here let us see that if the Galerkin basis, or the discretization grid are tensor product, the discrete counterpart of (1.3) stays within the tensor product structure. Indeed, suppose the chosen basis functions $\phi_{\boldsymbol{i}}(\boldsymbol{q}) = \phi_{i_1}(q_1)\cdots\phi_{i_d}(q_d)$, and the convection coefficients $C_k(\boldsymbol{q}) = C_k^{(1)}(q_1)\cdots C_k^{(d)}(q_d)$ factorize. Then the Galerkin element for the convection reads

$$\begin{aligned}\int \phi_{\boldsymbol{i}} \frac{\partial}{\partial q_p} C_k \phi_{\boldsymbol{j}} \boldsymbol{dq} = &\left(\int \phi_{i_1} C_k^{(1)} \phi_{j_1} dq_1\right)\cdots\left(\int \phi_{i_{p-1}} C_k^{(p-1)} \phi_{j_{p-1}} dq_{p-1}\right)\\ &\cdot\left(\int \phi_{i_p} \frac{dC_k^{(p)}\phi_{j_p}}{dq_p} dq_p\right)\\ &\cdot\left(\int \phi_{i_{p+1}} C_k^{(p+1)} \phi_{j_{p+1}} dq_{p+1}\right)\cdots\left(\int \phi_{i_d} C_k^{(d)} \phi_{j_d} dq_d\right),\end{aligned}$$

i.e. retains the factorized form.

The diffusion terms may be written similarly. Tensor product decomposition issues for the diffusion part will be addressed in Section 3.2.1.

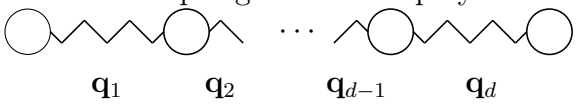
If a finite (or spectral) difference scheme is used, the situation is even simpler: each partial derivative casts to the Kronecker product of matrices, and the coefficients cast to diagonal matrices,

$$\begin{aligned}\frac{\partial}{\partial q_p} &\to I\otimes\cdots\otimes I\otimes\nabla_p\otimes I\otimes\cdots\otimes I,\\ C_k(\boldsymbol{q}) &\to \operatorname{diag}(C_k^{(1)}(q_1(i_1)))\otimes\cdots\otimes\operatorname{diag}(C_k^{(d)}(q_d(i_d))).\end{aligned}$$

1.1.3 Bead-spring chain in the Brownian motion: a micro-model for non-Newtonian dilute polymer flows

An area of great interest in physics and engineering is the viscoelastic materials, for example, polymer solutions and melts, liquid crystals and so on. Such liquids are called non-Newtonian due to nontrivial features of their flows: hysteresis effects, die swells or jet break-ups and more. These phenomena arise because the macroscopic stress depends on the previous history of deformations, as a result of a nonlinear coupling between the macroscopic flow parameters (geometry, rheological response) and the micro-configurations and motions of the dissolved molecules.

Figure 1.1: Bead-spring model of a polymer in a fluid



As we noted in the beginning of the section, an exact simulation of such flows would require a coupled modeling of all polymer and fluid molecules, which is surely infeasible in the macroscopic scales. As an alternative, micro-macro models have been proposed in the computational rheology (see e.g. the review [135]). They consider the macroscopic continuum models (Navier-Stokes equations) together with the stochastic kinetic theory at the molecular level. The latter may involve more or less accurate mechanical descriptions of the molecules, for example, linear chains of beads connected by rigid rods or elastic springs (depending on the number of chain segments and their properties). In particular, we will focus on the bead-spring chains, see Fig. 1.1. Each bead experiences elastic, drag and Brownian forces. It is the latter that makes the stochastic description necessary. So we derive the micro-scale Fokker-Planck equation, following [26].

Let the position vectors of beads be $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_{d+1}$, then the equations of motion write

$$\mathtt{m}\frac{d^2\boldsymbol{R}_k}{dt^2} = \boldsymbol{G}_k(\boldsymbol{R}_k) + \boldsymbol{F}_k(\boldsymbol{R}_{k+1} - \boldsymbol{R}_k) - \boldsymbol{F}_{k-1}(\boldsymbol{R}_k - \boldsymbol{R}_{k-1}) + f_k(t),$$

where $\boldsymbol{G}_k$ is the viscous drag force, $\boldsymbol{F}_k$ is the elastic force of the spring between k-th and $k+1$-t beads, and f_k is the Brownian stochastic force, such that

$$\langle f_k(t)\rangle = 0, \quad \langle f_k(t) f_m(t')\rangle = \mathtt{k}T\delta(t-t')\delta_{k,m},$$

where $\mathtt{k}$ is the Boltzmann constant (not the index k). For brevity, we may agree that $\boldsymbol{F}_0 = \boldsymbol{F}_{d+1} = 0$.

The drag force is proportional to the total velocity of the bead w.r.t. the fluid, and reads

$$\boldsymbol{G}_k = -\eta\left(\frac{d\boldsymbol{R}_k}{dt} - \boldsymbol{v}_0 - (\nabla_{\boldsymbol{y}}\boldsymbol{v})\boldsymbol{R}_k\right),$$

where η is the drag constant, $\boldsymbol{v}$ is the fluid velocity field, and $\boldsymbol{v}_0$ is its average. The gradient $\nabla_{\boldsymbol{y}}$ is taken for each velocity component with respect to the spatial coordinates in the fluid, such that $\nabla_{\boldsymbol{y}}\boldsymbol{v}$ is a 3×3 matrix. The first-order (*homogeneous*) expansion $\boldsymbol{v} = \boldsymbol{v}_0 + (\nabla_{\boldsymbol{y}}\boldsymbol{v})\boldsymbol{R}_k$ is reasonable, since in most cases the velocity does not change significantly at the distances comparable to the polymer molecule size. In addition, we neglect the inertia of a bead, i.e. $\mathtt{m} = 0$.

Therefore, it is convenient to transform the position vectors to the elongations of the springs, $\boldsymbol{q}_k = \boldsymbol{R}_{k+1} - \boldsymbol{R}_k$ (see Fig. 1.1), which we choose as the configuration coordinates of the system, and write

$$\eta\frac{d\boldsymbol{q}_k}{dt} = (\nabla_{\boldsymbol{y}}\boldsymbol{v})\boldsymbol{q}_k + \boldsymbol{F}_{k-1}(\boldsymbol{q}_{k-1}) - 2\boldsymbol{F}_k(\boldsymbol{q}_k) + \boldsymbol{F}_{k+1}(\boldsymbol{q}_{k+1}) + f_{k+1}(t) - f_k(t).$$

The last two terms may be gathered into another stochastic force $g_k(t) = f_{k+1}(t) - f_k(t)$, with the moments

$$\langle g_k(t)\rangle = 0, \quad \langle g_k(t) g_m(t')\rangle = \mathtt{k}T\delta(t-t')A_{k,m},$$

where

$$A_{k,m} = \begin{cases} 2, & k = m, \\ -1, & k = m \pm 1, \\ 0, & \text{otherwise} \end{cases}$$

is the so-called *Rouse* matrix. Recalling the definitions of the Fokker-Planck coefficients, we are now ready to write the scale-normalized equation,

$$\begin{aligned} \frac{\partial \psi(\boldsymbol{q},t)}{\partial t} = &- \sum_{k=1}^{d} \frac{\partial}{\partial \boldsymbol{q}_k} \left((\nabla_{\boldsymbol{y}} \boldsymbol{v}) \boldsymbol{q}_k - \frac{1}{4W} \sum_{m=1}^{d} A_{k,m} \boldsymbol{F}_m(\boldsymbol{q}_m) \right) \psi \\ &+ \sum_{k,m=1}^{d} \frac{1}{4W} A_{k,m} \frac{\partial^2}{\partial \boldsymbol{q}_k \partial \boldsymbol{q}_m} \psi, \end{aligned} \tag{1.4}$$

where $\boldsymbol{q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_d) = (q_1^1, q_1^2, q_1^3, \ldots, q_d^3)$ is the tuple of elongation components of all springs, and $W = 2\eta/\mathtt{k}T$ is the *Weissenberg* number, which reflects the relation between macro and micro scales.

The domain for Eq. (1.4) depends on the elastic forces. In the Rouse, or Hookean model $\boldsymbol{F}_m = \boldsymbol{q}_m$, the domain is the whole space. If a finitely extensible spring law is used, the domain restricts to the tensor product of balls,

$$\Omega = \mathbb{B}_{\sqrt{b_1}} \otimes \mathbb{B}_{\sqrt{b_2}} \otimes \cdots \otimes \mathbb{B}_{\sqrt{b_d}},$$

where $\sqrt{b_m}$ is the maximal length of the m-th spring.

The following finitely extensible laws are typically chosen:

- Inverse Langevin $\boldsymbol{F}_m = \dfrac{\sqrt{b_m}}{3} L^{-1}\left(\dfrac{|\boldsymbol{q}_m|}{\sqrt{b_m}}\right) \dfrac{\boldsymbol{q}_m}{|\boldsymbol{q}_m|}$, where $L(s) = \coth(s) - 1/s$ is the Langevin function, or its approximations:
- CPAIL, $\boldsymbol{F}_m = \dfrac{1 - |\boldsymbol{q}_m|^2/(3b_m)}{1 - |\boldsymbol{q}_m|^2/b_m} \boldsymbol{q}_m$, and
- FENE $\boldsymbol{F}_m = \dfrac{\boldsymbol{q}_m}{1 - |\boldsymbol{q}_m|^2/b_m}$.

All these functions have a pole at $|\boldsymbol{q}_m| = \sqrt{b_m}$, and the probability density vanishes at the boundary. In terms of well-posedness and discretization issues, it is convenient to pose natural non-leak, or zero flux boundary conditions [64],

$$\left((\nabla_{\boldsymbol{y}} \boldsymbol{v}) \boldsymbol{q}_k \psi - \frac{1}{4W} \sum_{m=1}^{d} A_{k,m} \left(\boldsymbol{F}_m \psi + \frac{\partial \psi}{\partial \boldsymbol{q}_m} \right) \right) \cdot \boldsymbol{n}_{\boldsymbol{q}_k} = 0, \quad \boldsymbol{q} \in \partial\Omega,$$

where $\boldsymbol{n}_{\boldsymbol{q}_k}$ is the normal vector at $\partial \mathbb{B}_{\sqrt{b_k}}$.

Remark 1.1.1. Generally, in addition to the configuration space of spring elongations, the probability density diffuses in the physical space, and one has to add a dependence on the position in the fluid $\boldsymbol{y}$ of the form

$$\frac{(l/L)^2}{4W(d+1)} \Delta_{\boldsymbol{y}} \psi - \nabla_{\boldsymbol{y}} \cdot (\boldsymbol{v}\psi)$$

to the right-hand side of (1.4), where l is the length scale of a spring, and L is the macro length scale. However, in many cases $l \ll L$, while $\nabla_{\boldsymbol{y}} \cdot \boldsymbol{v} = 0$ due to the incompressibility of the fluid, and we will consider (1.4) only in the configuration coordinates for simplicity. The only term depending on $\boldsymbol{y}$ is now $\nabla_{\boldsymbol{y}}\boldsymbol{v}$, but it allows to solve equations (1.4) uncoupled for each $\boldsymbol{y}$ point. So it is enough to discuss the configuration equation with constant $\nabla_{\boldsymbol{y}}\boldsymbol{v}$.

Remark 1.1.2. We have presented the full three-dimensional model, i.e. where each configuration degree of freedom $\boldsymbol{q}_k = (q_k^1, q_k^2, q_k^3)$ constitutes a three-dimensional elongation vector. Some processes (linear stretch, planar shear) are effectively one- or two-dimensional. In these cases, we may consider reduced models with $\boldsymbol{q}_k = (q_k^1, q_k^2)$ or even one-dimensional formulation.

The micro-model influences the macroscopic properties of the flow via the stress tensor. The Navier-Stokes equation for the macro-flow writes

$$\begin{aligned} \frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla_{\boldsymbol{y}} \boldsymbol{v} &= \frac{1}{\rho} \nabla_{\boldsymbol{y}} \cdot \left(-pI + \eta_s \nabla_{\boldsymbol{y}} \boldsymbol{v} + \eta_s (\nabla_{\boldsymbol{y}} \boldsymbol{v})^\top + \boldsymbol{\tau} \right), \\ \nabla_{\boldsymbol{y}} \cdot \boldsymbol{v} &= 0, \end{aligned} \tag{1.5}$$

where ρ is the fluid density, p is the pressure, I is the identity matrix, η_s is the viscosity coefficient such that $\eta_s \nabla_{\boldsymbol{y}} \boldsymbol{v} + \eta_s (\nabla_{\boldsymbol{y}} \boldsymbol{v})^\top$ is the stress contribution from the solvent, and $\boldsymbol{\tau}$ is the stress contribution from the polymer.

The latter may be derived as follows [26]. By definition, the stress is the force per unit area, which results from the action of beads on the medium. Without external (e.g. gravitational or electric) forces, we are left with the spring tensions and the motion of beads. Suppose the polymer concentration is c, then each spring acts in the volume $\boldsymbol{q}_k \cdot \boldsymbol{S}$, where $c\boldsymbol{q}_k \cdot \boldsymbol{n} S$ beads are contained (here $\boldsymbol{S}$ is an arbitrary plane of area S and the normal $\boldsymbol{n}$). Therefore, the spring contribution to the stress (i.e. we take $S = 1$) will be

$$-c \sum_{k=1}^{d} \langle \boldsymbol{q}_k \boldsymbol{F}_k \rangle = -c \int \sum_{k=1}^{d} \boldsymbol{q}_k \boldsymbol{F}_k \psi(\boldsymbol{q}) d\boldsymbol{q}.$$

The beads motion provides the stress component in terms of the average value of the momentum flux, which in the case of the Brownian motion with the Maxwellian velocity distribution writes simply as $cd\mathbb{k}TI$. Normalizing the scales, we obtain the following Kramers expression for the polymeric stress tensor:

$$\boldsymbol{\tau} = dI - \int \sum_{k=1}^{d} \boldsymbol{q}_k \boldsymbol{F}_k \psi(\boldsymbol{q}) d\boldsymbol{q}, \tag{1.6}$$

provided that the probability density is normalized such that $\int \psi d\boldsymbol{q} = 1$. Note that (1.6) is a linear functional on the high-dimensional Fokker-Planck solution.

The case of zero velocity gradient $\nabla_{\boldsymbol{y}} \boldsymbol{v} = 0$ leads to the potential form of the right-hand side in (1.4), i.e.

$$\frac{\partial \psi(\boldsymbol{q}, t)}{\partial t} = \frac{1}{4W} \sum_{k,m=1}^{d} A_{k,m} \frac{\partial}{\partial \boldsymbol{q}_k} \left(-\psi \frac{\partial V_m(\boldsymbol{q}_m)}{\partial \boldsymbol{q}_m} + \frac{\partial}{\partial \boldsymbol{q}_m} \psi \right),$$

since Hookean or FENE forces are potential, $\boldsymbol{F}_m = -\partial V_m(\boldsymbol{q}_m)/\partial \boldsymbol{q}_m$. We may introduce the total potential $V = V_1(\boldsymbol{q}_1) + \cdots + V_d(\boldsymbol{q}_d)$ and cast the Fokker-Planck right-hand side to the symmetric form [64]:

$$\frac{\partial \psi(\boldsymbol{q},t)}{\partial t} = \frac{1}{4W}\sum_{k,m=1}^{d} \frac{\partial}{\partial \boldsymbol{q}_k} A_{k,m} M \frac{\partial}{\partial \boldsymbol{q}_m}\frac{1}{M}\psi,$$

where we introduce the (unnormalized) Maxwellian $M = \exp(-V)$ and use

$$-\psi\frac{\partial V_m(\boldsymbol{q}_m)}{\partial \boldsymbol{q}_m} + \frac{\partial}{\partial \boldsymbol{q}_m}\psi = M\frac{\partial}{\partial \boldsymbol{q}_m}\frac{1}{M}\psi.$$

Though the Fokker-Planck is now a diffusion operator, its numerical treatment is difficult due to an extremely high variability of M with $\boldsymbol{q}$. However, it provides nice theoretical arguments, for example, one may see immediately that the Maxwellian satisfies the stationarity condition,

$$\psi_\star(\boldsymbol{q}) = M(\boldsymbol{q}), \quad \frac{\partial \psi_\star}{\partial t} = 0.$$

In addition, the Lax-Milgram theorem may be employed to show the existence and uniqueness of a weak solution. Here, we would like to note that he Maxwellian is a factorisable function: since the potential is additive, it holds

$$M(\boldsymbol{q}) = \exp(-V_1(\boldsymbol{q}_1))\exp(-V_2(\boldsymbol{q}_2))\cdots\exp(-V_d(\boldsymbol{q}_d)).$$

Therefore, the computation of the stress (1.6) does not involve in fact a multidimensional integration, only univariate terms

$$\frac{\int \boldsymbol{q}_k \boldsymbol{F}_k(\boldsymbol{q}_k)\exp(-V_k(\boldsymbol{q}_k))d\boldsymbol{q}_k}{\int \exp(-V_k(\boldsymbol{q}_k))d\boldsymbol{q}_k}$$

have to be evaluated. This example shows a strong motivation to the application of tensor product techniques to this problem, especially for velocity gradients of moderate magnitude.

For the Hookean spring force $\boldsymbol{F}_m = \boldsymbol{q}_m$, the analytical stationary solution may be written not only for zero velocity gradient. Indeed, in this case the convection Fokker-Planck coefficient becomes linear in all $\boldsymbol{q}$ components,

$$\frac{\partial \psi}{\partial t} = -\sum_{k=1}^{dD}\frac{\partial}{\partial q_k}\sum_{m=1}^{dD} C_{k,m} q_m \psi + \sum_{k,m=1}^{dD}\mathcal{A}_{k,m}\frac{\partial^2}{\partial q_k \partial q_m}\psi, \tag{1.7}$$

where D is the physical dimension (1, 2 or 3), $\boldsymbol{q} = [q_k]$, and the matrices read

$$C = I_{d\times d}\otimes(\nabla_{\boldsymbol{y}}\boldsymbol{v}) - \mathcal{A}, \quad \mathcal{A} = \frac{1}{4W}A\otimes I_{D\times D}.$$

It was proven for example in [32] that the (unnormalized) steady probability density is given by the generalized Gaussian

$$\psi|_{t\to\infty} = \exp(-\boldsymbol{q}^\top B\boldsymbol{q}), \tag{1.8}$$

where B is the solution of the following Lyapunov matrix equation,

$$CB^{-1} + B^{-1}C^{\top} = 4\mathcal{A}.$$

For a nonzero $\nabla_{\boldsymbol{y}}\boldsymbol{v}$, the matrix B is not diagonal, and the solution (1.8) is not factorisable. However, we will see how it is still possible to approximate it via more elaborated separation of variables techniques.

Finally, consider the two-dimensional physical space and finitely extensible springs. Since both the spectral methods and tensor decompositions are suited for cubic domains, we employ the polar coordinates to get rid of the ball domains,

$$\boldsymbol{q}_k = (q_k^1, q_k^2) \to (r_k, \theta_k) \in \left([0, \sqrt{b_k}) \otimes [0, 2\pi)\right).$$

Now we have periodic boundary conditions over θ_k, and natural boundary conditions over r_k. Using the differentiation rules, one obtains immediately

$$\begin{aligned}
\frac{\partial}{\partial q_k^1} &= \cos(\theta_k)\frac{\partial}{\partial r_k} - \frac{\sin(\theta_k)}{r_k}\frac{\partial}{\partial \theta_k}, \\
\frac{\partial}{\partial q_k^2} &= \sin(\theta_k)\frac{\partial}{\partial r_k} + \frac{\cos(\theta_k)}{r_k}\frac{\partial}{\partial \theta_k},
\end{aligned}$$

and the Fokker-Planck operator (1.4) rewrites accordingly. The FENE force, for example, casts to a factorisable function

$$\boldsymbol{F}_k = \frac{r_k}{1 - r_k^2/b_k} \cdot \begin{bmatrix}\cos(\theta_k) & \sin(\theta_k)\end{bmatrix}.$$

The spectral matrix elements are written according to [227]: the periodic Sinc interpolation elements are introduced in each θ_k, and the Chebyshev polynomials are used for the radial discretization, with the Galerkin coefficients (e.g. $\int S_i(\theta_k) f(\theta_k) \nabla_{\theta_k} S_j d\theta_k$) calculated using the quadrature rules of the same classes with twofold amount of points.

1.2 Chemical master equation for stochastic chemical kinetics

A study of biological systems and molecular biology is undergoing a rapid development in recent years, and demonstrates impressive advances in understanding of genome sequences, cell behavior, vaccine design and other relevant problems. A proper description of the processes in living organisms is particularly challenging, due to both experimental issues (accurate measurements and separation of different substances) and the complexity of the systems. The latter means that a separate consideration of system components (genes or proteins) is not enough to understand possible very nontrivial and sometimes counter-intuitive phenomena (such as in the cell differentiation). The components experience complex interconnections, resulting in a nonlinear dynamical evolution, which has to be considered at the level of the whole system.

The *system biology*, a research area devoted to both molecular biology and system theory, takes into account specific properties of the components and their interactions

in the system, aiming for revealing and understanding of biological laws, and finally for development of new biological systems, efficient production of vaccines and drugs or treatment of diseases.

The mathematical modeling in this area is crucial. Each experiment *in vivo* may be very complicated and expensive. Moreover, a mathematical model allows to isolate some phenomena in order to understand their contribution to the whole picture, or to put a system in conditions, which could be impossible in real life. Of course, discrepancies between the predictions provided by a model and actual measurements may evidence for both inappropriate model and inaccurate experiments. So, it is important to keep the experimental investigation of real biological systems and *in silico* modeling connected, supplying corrections and insights from one to another.

Different scales yield different levels and methods of simulations. Approximately, they may be classified into four categories.

- Macroscopic deterministic scale,
- mesoscopic (e.g. cell) scale,
- classical microscopic (molecular) scale, and
- quantum formalism and corresponding scales.

The first approach is suitable if the number of molecules is sufficiently large, of the order of the Avogadro constant. In this case, the quantum and stochastic fluctuations are negligible, and the chemical kinetics or biological dynamics can be described in terms of macroscopic concentrations by ordinary differential equations (ODEs) with a satisfactory accuracy. Standard analytical or numerical tools can be employed to construct a complete portrait of the system.

Essentially microscale processes at the molecular and atom levels have to be considered via the molecular dynamics, tracking all coordinates, velocities and physical forces (e.g. via the Fokker-Planck equation). Though being rather accurate, this way may be too computationally hard for large molecules, typical for biological systems, such as DNA, RNA or proteins.

Even more involving is the quantum simulation. The Schroedinger equation proved to be in an extremely accurate correspondence with the experiment for small systems: for example, the energy levels for the Helium were predicted with 8 and more decimal digits [56]. However, it seems to be infeasible yet to employ the quantum level for calculation of protein reactions.

At the *mesoscopic* scale, the system description is still more or less phenomenological (the kinetic rates are usually estimated from experiments and intuition rather than quantum ab initio models), but the stochastic noise brings already a significant contribution, which can not be properly resolved by deterministic ODEs. In intracellular systems, the number of molecules of chemical species is typically of the order of hundreds. For such dilute concentrations, stochastic fluctuations in the numbers of molecules may hit the level $\mathcal{O}(10^{-1})$ [223, 7], since collisions between molecular species and corresponding reactions occur in an inherently random way. Moreover, this biological noise itself plays an important role in inter- and intra-cellular functions. Such systems may exhibit unexpected responses, e.g. metastability patterns [69, 183].

Therefore, the stochastic kinetics is more appropriate for such systems, since it can cope with the stochastic noise. The *system state* is introduced as a vector of copy numbers of species, interacting through several biochemical *reaction channels*. The states are essentially discrete: a copy number of substances is always integer, and a reaction may also yield only an integer change in the number of molecules. The reaction rate is cast now to the *propensity* function, which defines a probability that a reaction will occur in the next infinitesimal time interval.

Thus, the dynamical evolution of the stochastic reaction system can be seen as a jump Markov process. An accurate stochastic description of that may be given by the so-called *chemical master equation* (CME), which simulates the probability distribution over all possible system states [232, 74, 77].

During the history of the chemical kinetics modeling in biology, different approaches have been developed. Monte Carlo methods are based on a statistically large ensemble of realizations of the stochastic process associated with the CME. The most famous is the stochastic simulation algorithm (SSA) by Gillespie [74]. In each step, SSA selects the next reaction to fire according to its propensity and a thrown random number, and performs the corresponding transition to the new state. However, SSA is often very computationally demanding by several reasons. First, due to the randomized setting of a single realization, a lot of simulation trajectories (10^6–10^8 and more) are required to ensure correct statistical outputs. It may be especially challenging to estimate the probability of rare events, since only a few trajectories may enter the proper regimes of a system.

Second, even a phenomenological biological model may contain multiple reactions and species, significantly separated in magnitude with respect to both time and population scales. For example, fast reactions stabilize the related effects very rapidly, while slow reactions may require much longer simulation to be satisfactory captured. However, in most of the steps the SSA samples the fast reactions, which may be not so interesting in the end.

Several improvements include advanced sampling techniques [112], τ-leaping methods [75], or system-partitioning hybrid methods [88, 111, 123]. Additionally, the chemical Fokker-Planck equation may be considered [76] to treat high-concentration systems. The Fokker-Planck equation can be discretized with a coarse grid, containing less unknowns than the initial CME.

A principal alternative to the Monte Carlo-type methods is the solution of the master equation directly as a linear ODE. For many systems, the probability distribution vanishes rapidly outside a bounded domain. Thus, it is possible to truncate the state space to a finite domain, without a significant deterioration of the solution [181].

However, this setting inherits the same difficulty as the Fokker-Planck equation: even the truncated state space volume is usually still very large and grows exponentially with the number of species. Therefore, some data-sparse approximation is needed. The sparse grids technique was one of the first in this direction [108], followed by tensor product methods, demonstrated already their potential in the form of greedy algorithms in the canonical tensor format [4, 64, 168, 33, 25, 109], and the so-called *Dirac-Frenkel* dynamical approach on the Tucker manifold (see [159, 175, 193] and [122] for a particular application to the CME). We use more advanced tensor formats

and methods, which allow to simulate systems of higher complexity.

Now, we present the formulation and basic properties of the chemical master equation in more detail. Suppose that d different active chemical species $S_1, \ldots, S_d$ in a well-stirred medium can react in M reaction channels. Each channel is specified by a *stoichiometric vector* $\boldsymbol{z}^m \in \mathbb{Z}^d$, and a *propensity* function $w^m(\boldsymbol{i}) : \mathbb{R}_+^d \to \mathbb{R}_+$, $m = 1, \ldots, M$, $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geqslant 0\}$, such that the m-th reaction writes

$$S_1 a_1^m + \cdots + S_d a_d^m \xrightarrow{w^m} S_1(a_1^m + z_1^m) + \cdots + S_d(a_d^m + z_d^m)$$

in the classical chemical notation.

To introduce the stochastic description, we denote the states by $\boldsymbol{i} = (i_1, \ldots, i_d)$, and always mean the copy numbers of species, so that i_k is a nonnegative integer, $i_k \in (\{0\} \cup \mathbb{N})$. The probabilistic role of the propensity functions is the following: for an infinitesimal time interval dt,

$$W^m(\boldsymbol{i}, t, dt) = w^m(\boldsymbol{i})dt$$

is the probability that, given copy numbers $\boldsymbol{i}$ at the time t, one reaction in the m-th channel will occur in the system in the next time interval $[t, t + dt)$.

The state $\boldsymbol{i}$ is quantified by the probability that the numbers of molecules of species $S_1, \ldots, S_d$ take particular values $i_1, \ldots, i_d$ at the time t,

$$\Psi(\boldsymbol{i}, t) : (\{0\} \cup \mathbb{N})^d \cup [0, T] \to \mathbb{R}_+.$$

This probability is in fact conditional, and depends also on the initial state, $\boldsymbol{i}|_{t=0}$, but we omit it for the sake of brevity.

Now, taking dt small enough, such that the probability that more than one reaction will occur in $[t, t + dt)$ is negligible, we may write the distribution in the end of the interval, $\Psi(\boldsymbol{i}, t + dt)$, using the probability addition and multiplication laws for independent and mutually exclusive events,

$$\Psi(\boldsymbol{i}, t + dt) = \underbrace{\Psi(\boldsymbol{i}, t)\left(1 - \sum_{m=1}^{M} w^m(\boldsymbol{i})dt\right)}_{\substack{\text{state was } \boldsymbol{i}\text{, and} \\ \text{no reaction fired} \\ \equiv \texttt{not}(\text{any reaction fired})}} + \sum_{m=1}^{M} \underbrace{\Psi(\boldsymbol{i} - \boldsymbol{z}^m, t) \cdot w^m(\boldsymbol{i} - \boldsymbol{z}^m)dt}_{\substack{m\text{-th reaction occurs} \\ \text{s.t. final state is } \boldsymbol{i}\text{, hence} \\ \text{initial state is } \boldsymbol{i} - \boldsymbol{z}^m}}.$$

Combining the terms $\Psi(\boldsymbol{i}, t + dt) - \Psi(\boldsymbol{i}, t)$ in the left-hand side and taking the limit $dt \to 0$, derive [77, 78] the chemical master equation,

$$\frac{d\Psi(\boldsymbol{i}, t)}{dt} = \sum_{m=1}^{M} w^m(\boldsymbol{i} - \boldsymbol{z}^m)\Psi(\boldsymbol{i} - \boldsymbol{z}^m, t) - w^m(\boldsymbol{i})\Psi(\boldsymbol{i}, t). \tag{1.9}$$

Any copy numbers of species are potentially possible, so Eq. (1.9) is an infinite-size ODE. Of course, to conduct numerical simulations, we need a procedure to truncate it to a finite problem. The *finite state space projection* (FSP) algorithm [181] employs the fact that very large copy numbers are unlike to appear in a finite time,

$$\Psi(\boldsymbol{i}, t) \to 0, \ |\boldsymbol{i}| \to \infty.$$

So, we consider each i_k in a finite range, $i_k = 0, \dots, n_k - 1$, taking n_k large enough, such that $\Psi(\boldsymbol{i}, t)$ is e.g. below the machine precision for $i_k > n_k$, and one can neglect the error introduced by the state space truncation.

As in the previous section, even if each $n_k = \mathcal{O}(n)$ is of the order of tens, the total number of degrees of freedom scales as n^d, and indirect storage and processing for Ψ is indispensable. We will postpone this for the next chapters, when we introduce tensor product notations and methods. Now let us focus on algebraic and spectral properties of the CME operator.

First, introduce a more convenient counterpart to (1.9) with the help of the shift matrices. Denote

$$J^z = \begin{bmatrix} 0 & & & & \\ \vdots & \ddots & & & \\ 1 & & \ddots & & \\ & \ddots & & \ddots & \\ & & 1 & \cdots & 0 \end{bmatrix} \leftarrow \quad \text{row} \quad z+1, \quad \text{if } z \geqslant 0, \tag{1.10}$$

and for $z < 0$ we define $J^z = (J^{-z})^\top$. Now we write the finite state approximation (FSP) of (1.9) as a linear ODE,

$$\frac{d\psi(t)}{dt} = A\psi(t), \quad A = \sum_{m=1}^{M} (\mathbf{J}^{\boldsymbol{z}^m} - \mathbf{J}^0) \operatorname{diag}(w^m), \quad \psi(t) \in \mathbb{R}_+^{\prod_{k=1}^d n_k}, \tag{1.11}$$

where the multidimensional shift operator reads

$$\mathbf{J}^{\boldsymbol{z}} = J^{z_1} \otimes \cdots \otimes J^{z_d},$$

$w^m = \{w^m(\boldsymbol{i})\}$ and $\psi(t) = \{\psi(\boldsymbol{i}, t)\}$, $\boldsymbol{i} \in \bigotimes_{k=1}^{d} \{0, \dots, n_k - 1\}$, are the corresponding values of w^m and ψ stacked into vectors, $\operatorname{diag}(w^m)$ is a diagonal matrix with the values of w^m stretched along the diagonal, and $\otimes$ means the Kronecker product in the usual sense (see Eq. 1.18). Note that $\mathbf{J}^0$ is just an identity matrix of the proper sizes. The finite solution $\psi(\boldsymbol{i}, t)$ does not generally coincide with the infinite one $\Psi(\boldsymbol{i}, t)$ at the finite state points, even if the initial state was projected exactly, $\psi(\boldsymbol{i}, 0) = \Psi(\boldsymbol{i}, 0)$, though the discrepancy can be quantified, see Thm. 1.2.1.

The boundary values of the propensity functions w^m require a specific consideration. Suppose that a reaction decreasing the number of molecules of some kind, $z_k^m < 0$, is allowed to process in the case when there is no sufficient amount of the corresponding component, $w^m(\boldsymbol{i}) > 0$ for $i_k < |z_k^m|$. It would result in a nonphysical phenomenon: negative i_k occur with a nonzero probability. To avoid this situation, we shall always pose *boundary conditions*:

$$w^m(\boldsymbol{i}) = 0 \quad \text{if any of } \boldsymbol{i} + \boldsymbol{z}^m < 0. \tag{1.12}$$

These *non-leak* boundary conditions are enough for the infinite equation (1.9) to be well-posed in physical and probabilistic senses, i.e. negative copy numbers never occur,

the probability Ψ is nonnegative, and the total normalization $\sum_{\boldsymbol{i}} \Psi(\boldsymbol{i}, t)$ is conserved during the time evolution (provided that $\Psi(\boldsymbol{i}, 0)$ obeys these properties). However, it might be not the case, if the FSP is applied straightforwardly, with no adjustment of w^m.

Basic properties of the FSP approximation established in [181] are the following. First, if $\psi(\boldsymbol{i}, 0) \geqslant 0$ and $w^m(\boldsymbol{i}) \geqslant 0$ then it remains $\psi(\boldsymbol{i}, t) \geqslant 0$. Second, the error in the solution "does not blow up", and is related to the probability normalization loss.

Theorem 1.2.1 ([181], Theorem 2.2)**.** Suppose $\psi(\boldsymbol{i}, 0) = \Psi(\boldsymbol{i}, 0) \geqslant 0$, $w^m(\boldsymbol{i}) \geqslant 0$. If it holds for some $\varepsilon > 0$ and $t \geqslant 0$ that

$$\sum_{\boldsymbol{i}} \psi(\boldsymbol{i}, t) \geqslant 1 - \varepsilon,$$

then

$$\psi(\boldsymbol{i}, t) \leqslant \Psi(\boldsymbol{i}, t) \leqslant \psi(\boldsymbol{i}, t) + \varepsilon, \quad \boldsymbol{i} \in \bigotimes_{k=1}^{d} \{0, \ldots, n_k - 1\}.$$

Further regularity analysis may be found in [70].

As was shown in [120], all eigenvalues of the CME operator in (1.11) have nonpositive real parts. Indeed, each row sum of $\mathbf{J}^{\boldsymbol{z}^m} - \mathbf{J}^0$ is either -1 or 0, and $\operatorname{diag}(w^m)$ is nonnegative, so both the diagonal entries and row sums of A are nonpositive. By the Gershgorin's theorem, all eigenvalues lie in the left half of the complex plane. This provides stability of the CME dynamics. However, if both w^m and ψ are nonzero at the points i_k such that $i_k + z_k^m \geqslant n_k$, all eigenvalues are strictly negative in real part, and the norm of the solution $\psi(t)$ decreases with t. It leads to the accumulation of the error shown in Theorem 1.2.1. This evidences for the necessity of taking n_k large enough, such that the truncated part of ψ is negligibly small.

It is possible to recover the normalization conservation for the truncated problem [121]. All what we need is to perturb the propensity functions, setting

$$w^m(\boldsymbol{i}) = 0 \quad \text{if any of } i_k + z_k^m \geqslant n_k, \quad k = 1, \ldots, d, \tag{1.13}$$

in addition to the natural boundary conditions (1.12). Now, following [121], we may observe that

$$\sum_{\boldsymbol{i}} w^m(\boldsymbol{i} - \boldsymbol{z^m})\psi(\boldsymbol{i} - \boldsymbol{z^m}) = \sum_{\boldsymbol{i} + \boldsymbol{z^m}} w^m(\boldsymbol{i})\psi(\boldsymbol{i}) = \sum_{\boldsymbol{i}} w^m(\boldsymbol{i})\psi(\boldsymbol{i}),$$

and hence $\boldsymbol{e}^\top(\mathbf{J}^{\boldsymbol{z}^m} - \mathbf{J}^0)\operatorname{diag}(w^m)\psi = 0$, where $\boldsymbol{e}$ is a vector of all ones, constituting the summation over $\boldsymbol{i}$.

Lemma 1.2.2. Suppose that both left (1.12) and right (1.13) boundary conditions hold for the FSP truncated CME. Then the minimal singular value of A is zero, with $\boldsymbol{e}$ being the left singular vector, and a stationary solution $\psi_\star : \frac{d\psi_\star}{dt} = 0$ being the right singular vector.

The second claim comes immediately from the fact that $A\psi = 0$ yields $\frac{d\psi}{dt} = 0$.

Remark 1.2.3. The *multiplicity* of the kernel of A requires additional study. As was shown in [97], a sort of non-redundancy in the set of species and reactions guarantees the uniqueness of the stationary solution. However, generally the kernel dimension can be as large as n and more (a reversible conversion reaction $S_1 + S_2 \leftrightarrows S_3$ with a constant propensity may be an example). In this case, the initial state, governing the actual kernel vector the process converges to, is crucial.

Now let us estimate the maximal singular value of the CME operator.

Lemma 1.2.4.

$$\|A\|_2 \leqslant 2 \sum_{m=1}^{M} \max(w^m).$$

This can be shown via the simple matrix norm inequality applied to (1.11): $\|\mathbf{J}^{\boldsymbol{z}^m} - \mathbf{J}^0\| \leqslant 2$ (see the next section for a detailed argumentation), and the norm of a diagonal matrix is its maximal element.

Notice that the matrix may be rather stiff: if w^m is a degree-p polynomial, its maximal element may be bounded by $(n-1)^p$. Therefore one may need a lot of small time steps to make the approximate dynamics accurate. The next section presents a conception how to address this issue in a smart way.

Finally, we may note that the nonpositivity of the CME operator helps to damp a high-frequency noise arising from tensor approximations in each step. Generally one could expect the tensor approximation error to grow in each time step, but we observed that even in a long dynamics, the noise maintains at a stable level.

1.3 From dynamical to stationary problem and back

1.3.1 Simultaneous space-time discretization

Supposing some discretization is introduced in space or state variables, we are left with an ODE

$$\begin{aligned} &\frac{dx(t)}{dt} + Ax(t) = f(t) \in \mathbb{C}^N \\ &x(0) = v, \quad t \in [0, T], \end{aligned} \tag{1.14}$$

possibly of a very large size $N = n^d$.

For simplicity, we assume that the matrix A is independent on t and x, though a generalization is also possible. To be precise, let us focus on the well-known Crank-Nicolson scheme (which is always suitable for a linear ODE as soon as the time step is small enough). Given a time grid (let it be uniform),

$$t \in \{t_p\}_{p=0}^{N_t}, \quad t_p = p\delta t, \quad p = 0, \dots, N_t, \quad f_p = f(t_p), \quad T = N_t \delta t,$$

the approximate solution of (1.14) may be computed stepwise, from the following linear system in each step,

$$\left(I + \frac{\delta t}{2} A\right) x_{p+1} = \left(I - \frac{\delta t}{2} A\right) x_p + \frac{\delta t}{2}(f_p + f_{p+1}), \quad p = 0, \dots, N_t - 1, \tag{1.15}$$

and $x_0 = v$. Provided the matrix A is nonnegative definite, it holds that the spectral norm of the *transition* matrix is bounded by one,

$$\left\| \left(I + \frac{\delta t}{2} A \right)^{-1} \left(I - \frac{\delta t}{2} A \right) \right\|_2 \leqslant 1,$$

and hence the method is absolutely stable [134].

The scheme possesses the second order of approximation, $\|x(t_p) - x_p\| = \mathcal{O}(\delta t^2)$. However, if the matrix A is stiff, the number of time steps required to ensure a reasonable accuracy in a long time dynamics may be quite large. On the other hand, the smaller time step we take, the smaller error is allowed in solution of the linear system (1.15) – but we would like to keep this error "not too small", to achieve a good compressibility of the solution in a tensor product format.

As a non-standard alternative, we may consider time as an additional dimension and agglomerate several time steps into one global linear system,

$$\begin{bmatrix} I + \frac{\delta t}{2} A & & & \\ -I + \frac{\delta t}{2} A & I + \frac{\delta t}{2} A & & \\ & \ddots & \ddots & \\ & & -I + \frac{\delta t}{2} A & I + \frac{\delta t}{2} A \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N_t} \end{bmatrix} = \begin{bmatrix} v - \frac{\delta t}{2} A v \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \frac{\delta t}{2} g, \qquad (1.16)$$

where $g = \begin{bmatrix} f_0 + f_1 & f_1 + f_2 & \cdots & f_{N_t - 1} + f_{N_t} \end{bmatrix}^\top$. Of course, this system makes no sense if we try to keep it in a general form – the stepwise elimination (1.15) is the optimal method for a bidiagonal matrix. However, imagine that it is possible to compute its solution more efficiently in a *structured* representation via a fast algorithm. Then it becomes tempting to obtain the whole time history at once, and use excessively small time steps to guarantee that all spectral components of A are properly resolved.

It is natural to expect that a structured representation is efficient under certain assumptions on the solution. For example, such a statement was considered with the *sparse grids* approach to parameter reduction [96, 235], where the numbers of spatial and time degrees of freedom contribute additively to the total complexity. To achieve that, appropriate smoothness conditions are imposed.

In this work, we focus on tensor product approximations. Generally, it is difficult to relate their efficiency to the smoothness of the solution. Existing estimates are comparable with the sparse grids complexity (see [215, 95]), but in most cases the actual cost is significantly lower than the theoretical predictions. Moreover, there is no general prevention for tensor product algorithms to be useful for some non-smooth examples as well. The low-rank separation of space and time variables was also employed in [224]. With multidimensional tensor techniques, however, we may introduce additional virtual dimensions, and ultimately approximate both spatial and temporal parts of (1.16) with the asymptotically logarithmic storage reduction, $\mathcal{O}(\log(N) \log(N_t))$, which is confirmed in numerical experiments.

The solution scheme will be described in more details below, after the tensor product formalism will be introduced. Now, let us focus on spectral properties of (1.16). First of all, a more convenient counterpart may be written via Kronecker products,

$$\mathcal{A} x = \mathcal{F}, \quad \mathcal{A} = I \otimes G_t + A \otimes \frac{\delta t}{2} M_t, \quad \mathcal{F} = \left(v - \frac{\delta t}{2} A v \right) \otimes e_1 + \frac{\delta t}{2} g, \qquad (1.17)$$

where the Kronecker product $\otimes$ defines the following block matrix,

$$A \otimes B = \begin{bmatrix} AB_{1,1} & \cdots & AB_{1,n} \\ \vdots & & \vdots \\ AB_{m,1} & \cdots & AB_{m,n} \end{bmatrix}, \tag{1.18}$$

e_1 is the first identity vector, and G_t, M_t are the temporal stiffness and mass matrices,

$$G_t = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}, \quad M_t = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix}.$$

We remind that $x = \begin{bmatrix} x_1 & x_2 & \cdots & x_{N_t} \end{bmatrix}^\top$ is the global solution vector, containing all time steps of the initial Cranck-Nicolson scheme.

Since G_t and M_t are triangular, one may conclude immediately that all their eigenvalues are equal to 1. To ensure the stability of the system, the spectrum of A is assumed to lie in the right half plane. Under this condition, we may prove the well-posedness for (1.16).

Theorem 1.3.1. Suppose $\mathrm{Re}\lambda(A) \geqslant 0$, then the critical singular values of $\mathcal{A}$ in (1.17) are estimated as follows,

$$\sigma_{\max}(\mathcal{A}) \leqslant 2 + \delta t \|A\|_2, \quad \sigma_{\min}(\mathcal{A}) \geqslant \frac{1}{N_t}, \tag{1.19}$$

so that $\mathrm{cond}(\mathcal{A}) \leqslant 2N_t + T\|A\|_2$.

Proof. Since both $\mathrm{Re}\lambda(A) \geqslant 0$ and $\lambda(M_t) > 0$, we may claim that $\|\mathcal{A}z\| \geqslant \|(I \otimes G_t)z\|$ for any vector z. In particular, using also the properties of the Kronecker product, we may estimate $\sigma_{\min}(\mathcal{A}) \geqslant \sigma_{\min}(G_t)$. The latter is computed using the spectral norm of G_t^{-1}, which can be bounded via the following inequality,

$$\|M\|_2 \leqslant \sqrt{\|M\|_1 \|M\|_\infty}, \quad \|M\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{m} |M_{i,j}|, \quad \|M\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^{n} |M_{i,j}|$$

for any $m \times n$ matrix M. One may verify straightforwardly that

$$G_t^{-1} = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ \vdots & \ddots & \ddots & \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

with both the row and column sums achieving N_t. Hence,

$$\sigma_{\min}(\mathcal{A}) \geqslant \sigma_{\min}(G_t) = \frac{1}{\|G_t^{-1}\|_2} \geqslant \frac{1}{\sqrt{\|G_t^{-1}\|_1 \|G_t^{-1}\|_\infty}} = \frac{1}{N_t}.$$

The second claim is proved.

The first estimate is derived similarly: the triangle inequality yields

$$\sigma_{\max}(\mathcal{A}) = \|\mathcal{A}\|_2 \leqslant \|G_t\|_2 + \frac{\delta t}{2}\|M_t\|_2\|A\|_2,$$

whereas

$$\|G_t\|_1 = \|G_t\|_\infty = \|M_t\|_1 = \|M_t\|_\infty = 2,$$

so the spectral norms are not greater than 2. Finally, recalling that $\delta t = T/N_t$, we finish the proof. □

It is not unnatural that the condition number depends linearly on all main properties of the system: number of time steps, time interval length, and the norm of the spatial matrix. Of course, this theorem is very general, and taking into account particular properties of A, one may provide refined results. Is there other ways to reduce the condition number? One of the mostly used approach is preconditioning. A simple and intuitive step is to relax the temporal contribution. Namely, we multiply (1.17) with the matrix $I \otimes G_t^{-1}$, so that we come to

$$\begin{aligned} \tilde{\mathcal{A}}x = \tilde{\mathcal{F}}, \quad \tilde{\mathcal{A}} = I \otimes I + A \otimes \frac{\delta t}{2}(G_t^{-1}M_t), \\ \tilde{\mathcal{F}} = \left(v - \frac{\delta t}{2}Av\right) \otimes e + \frac{\delta t}{2}(I \otimes G_t^{-1})g, \end{aligned} \tag{1.20}$$

where $e = G_t^{-1}e_1$ is a vector of all ones.

Theorem 1.3.2. Suppose $\mathrm{Re}\lambda(A) \geqslant 0$, then the critical singular values of $\tilde{\mathcal{A}}$ in (1.20) are estimated as follows,

$$\sigma_{\max}(\tilde{\mathcal{A}}) \leqslant 1 + T\|A\|_2, \quad \sigma_{\min}(\tilde{\mathcal{A}}) \geqslant 1, \tag{1.21}$$

so that $\mathrm{cond}(\tilde{\mathcal{A}}) \leqslant 1 + T\|A\|_2$.

Proof. Using the same arguments as before, we deduce that $\sigma_{\min}(\tilde{\mathcal{A}}) \geqslant 1$. Since both G_t and M_t are triangular, so is $G_t^{-1}M_t$, which contains all real positive eigenvalues 1 on the diagonal, and the property $\mathrm{Re}\lambda(A \otimes (G_t^{-1}M_t)) \geqslant 0$ persists. The upper bound writes

$$\sigma_{\max}(\tilde{\mathcal{A}}) \leqslant 1 + \frac{\delta t}{2}\|G_t^{-1}\|_2\|M_t\|_2\|A\|_2 \leqslant 1 + \delta t N_t\|A\|_2 = 1 + T\|A\|_2.$$

□

Now, choosing the time interval, we may make the simultaneous space-time matrix as well conditioned as prescribed.

Remark 1.3.3. Note that the time interval $[0, T]$ might not mean the whole time range required in application. We are not obliged to dispose the time stepping framework completely – the global time schemes (1.16) – (1.20) may be considered as the techniques to conduct "large" time steps of size T each. Suppose that the desired interval is

$[0, \hat{T}]$, we split it into subintervals $[0, T], [T, 2T], \ldots, [\hat{T} - T, \hat{T}]$, and use (1.16) – (1.20) with *restarts*. When the system for $[(q-1)T, qT]$, $q = 1, \ldots, \hat{T}/T$ is solved, we extract the last snapshot x_{N_t}, and use it as the initial state for the next interval. We may use the freedom in T to achieve the fastest computations.

Remark 1.3.4. If A is not symmetric, the fact that the spectrum of $G_t^{-1} M_t$ is real is important for such type of preconditioning to be useful. Indeed, suppose that both $\lambda(A)$ and $\lambda(G_t^{-1} M_t)$ are complex. Then it is possible that $\mathrm{Re}\left(\lambda(A)\lambda(G_t^{-1} M_t)\right) < 0$ for some eigenvalues. It may yield even worse conditioning than in (1.17), if $1 + \mathrm{Re}\lambda(A \otimes (G_t^{-1} M_t))$ will be too close to zero.

The complex eigenvalues of the temporal matrices may appear if the Galerkin or spectral [218, 128, 227] discretizations are used. Advantages of the spectral methods stem from their rapid convergence w.r.t. the number of degrees of freedom – if no more than $\mathcal{O}(40)$ temporal basis functions are used, one may safely proceed with the unpreconditioned counterpart of (1.17).

1.3.2 Solution of stationary problems by dynamical evolution

In some problems, posed initially as dynamical equations, we need only the stationary solution. Suppose for simplicity that the attractor is a point (e.g. a linear ODE $dx/dt + Ax = 0$ is considered), which obeys the null-space condition $Ax_\star = 0$. However, computation of $x_\star$ as a solution of the partial eigen- or singular value problem may suffer from some drawbacks. First, for a non-symmetric matrix the eigenvalue problem is difficult to solve, especially in tensor formats, since no variational formulation exists. A tensor-structured algorithm for the partial singular value decomposition could improve the situation, but it is unclear now how it will perform.

More important is the second issue: how to separate multiple kernel vectors, and filter them from higher eigen/singular vectors in case of very small spectral gaps. As we noted in Remark 1.2.3, a kernel of dimension greater than one may be a natural situation in the chemical master equation, and it is important to select a proper projection of the initial state to the null-space.

It is the latter consideration that gives us a natural idea how to solve this problem. All we need is to conduct a dynamical evolution, starting from a desired initial guess, until the residue (typically $|x(t + \delta t) - x(t)|$ or $|Ax(t)|$ is used) falls below a prescribed threshold. In this case, the fine discretization in the block scheme (1.16) is superfluous. Instead, we use the implicit Euler, a.k.a. the inverse power iteration,

$$(I + \delta t A)\, x_p = x_{p-1}, \quad p = 1, \ldots, N_t. \tag{1.22}$$

Note that here δt may be much larger than in (1.16). The intermediate solutions approximate the transient processes poorly, but as soon as the method converges, it recovers an accurate component in the lowest eigenspace of A. As was noted, the number of steps N_t is chosen such that

$$\eta = \frac{\|x_{N_t+1} - x_{N_t}\|}{\delta t \|x_{N_t}\|} \leqslant \epsilon, \quad \text{or} \quad \eta = \frac{\|Ax_{N_t}\|}{\|x_{N_t}\|} \leqslant \epsilon. \tag{1.23}$$

Provided the ODE system is stable, i.e. $A \geqslant 0$, one may quickly conclude that the condition number of the matrix in (1.22) is bounded by $1 + \delta t \|A\|_2$. As usual in the power method, the convergence rate $(1+\delta t|\mathrm{Re}\lambda_2(A)|)^{-1}$ emerges from the spectral gap, and higher δt leads to a faster convergence, but one has to care about the difficulty of the shifted matrix inversion.

Since the latter operation is usually done iteratively up to some accuracy (and this is the only way in the tensor approximation framework), we propose the following trick to reduce the cost. Compute the residual by one of the definitions in (1.23). If η is large, we do not need to solve (1.22) very precisely. When η diminishes, the accuracy may be improved. Practically, we use a rule of the form $\varepsilon = c\eta$ (e.g. $c = 10^{-1}$), where ε is the threshold for the solution of (1.22). This approach decreases the complexity of intermediate iterations significantly.

In the following we may refer to this method as the (implicit) Euler iterations. No ambiguity arises since the global space-time formulations (1.16)–(1.20) use the Crank-Nicolson scheme.

Chapter 2

Tensor product representations and approximations

We have seen that multidimensional arrays arise in many applications, but their direct storage and processing is impossible for really high dimensions, since usually we have to face the exponential growth of the number of degrees of freedom to maintain the same accuracy level with the increase of dimensionality.

Fortunately, in practical applications tensors are usually not very general, but arise from some physical problem, and possess a certain hidden structure, which is to be determined. Sometimes this structure follows obviously from the model. For example, we may mention (pointwise) sparse, or shift invariant (Toeplitz or Hankel) tensors [8]. However, these particular classes are not wide enough for our purposes, and we will consider some other low-parametric representations.

2.1 Separation of variables in two and many dimensions

A method of choice in this work is the separation of variables. Suppose we are given a tensor $x = [x(i_1, \ldots, i_d)]$, with $i_k = 1, \ldots, n_k \leqslant n$, $k = 1, \ldots, d$, such that the cardinality of x is bounded by n^d. However, assume that x can be written as a direct product of univariate arrays (vectors), i.e.

$$x(i_1, \ldots, i_d) = x^{(1)}(i_1) x^{(2)}(i_2) \cdots x^{(d)}(i_d). \tag{2.1}$$

Note that each $x^{(k)}$ requires only n elements to store, but they define in fact all values of x. So, the effective storage cost reduces to $nd \ll n^d$. Of course, the ultimate separation (2.1) is too restrictive in practice. To generalize (2.1) efficiently, we need two important ingredients: first, the representation should allow a summation of several product terms, and second, it should admit approximate computations.

2.1.1 Matrix low-rank decomposition

To understand the basic ideas, consider first the two-dimensional case. Let $X = \big[x(i_1, i_2)\big]$ be a matrix, then the variables are naturally separated as follows,

$$x(i_1, i_2) = \sum_{\alpha=1}^{n} x_\alpha^{(1)}(i_1) x_\alpha^{(2)}(i_2) \quad \Leftrightarrow \quad X = X^{(1)}(X^{(2)})^\top.$$

This equation is called the dyadic decomposition, and the (approximate) reduction is done by limiting the range of the *rank index* α from n to $r < n$, such that

$$\big\|X - X^{(1)}(X^{(2)})^\top\big\| \leqslant \varepsilon, \quad X^{(k)} = \Big[x_\alpha^{(k)}(i_k)\Big]_{\alpha, i_k = 1}^{r, n}.$$

So, $x^{(1)}$ and $x^{(2)}$ contain now $2nr$[1] entries, and we may suggest a choice of these *factors*, and how the approximation error ε depends on the *rank* r. A special property of the matrix case is that the *optimal* factorization is provided by the very particular decomposition, which is robustly computable using the well-established software (e.g. LAPACK).

Theorem 2.1.1. Any matrix X admits the *singular value decomposition* (SVD),

$$x(i_1, i_2) = \sum_{\alpha=1}^{n} U_\alpha(i_1) \sigma_\alpha V_\alpha(i_2), \tag{2.2}$$

where $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n \geqslant 0$, $\sigma_i^2 \in \lambda(X^* X)$, and U and V are matrices with orthonormal columns. Moreover, the *truncated* rank-r SVD yields an optimal rank-r approximation, i.e.

$$\left\| X - \sum_{\alpha=1}^{r} U_\alpha \sigma_\alpha V_\alpha^\top \right\|_2 = \min_{\operatorname{rank}(Y)=r} \|X - Y\|_2 = \varepsilon.$$

Obviously, one may assign $X^{(1)} = U, X_\alpha^{(2)} = \sigma_\alpha V_\alpha$ or $X_\alpha^{(1)} = U_\alpha \sigma_\alpha, X^{(2)} = V$, obtaining either *left-* or *right-orthogonal* decompositions, respectively. The conception of orthogonality will be used heavily in computational algorithms and thus recalled in the tensor formats sections below.

Though generally $r(\varepsilon)$ is unknown, if x comes from the discretization of a smooth function, reasonable bounds may be proven, typically of the form [230, 231, 103]

$$r = \mathcal{O}(\log^\beta(1/\varepsilon) \log^\gamma(n)), \quad \beta, \gamma > 0.$$

One drawback of the SVD is perhaps its cost: it requires all elements of a matrix, and $\mathcal{O}(mn \min(m, n))$ operations, so it may be difficult to apply if n is tens thousands or more. In such a case, we may be satisfied with a quasi-optimal, but much cheaper technique: the Gaussian elimination, also known as the cross interpolation scheme

[1] In the following, to write asymptotic estimates in a brief way, we will implicitly assume $n_k \leqslant n$, and so on.

[85, 86, 87, 84, 229]. It is based on the selection of "representative" columns and rows of a matrix, i.e.

$$X(i_1, i_2) \approx \tilde{X}(i_1, i_2) = \sum_{\alpha,\alpha'=1}^{r} X(i_1, \mathcal{J}^2_\alpha) M_{\alpha,\alpha'} X(\mathcal{J}^1_{\alpha'}, i_2), \tag{2.3}$$

where $M = (X(\mathcal{J}^1, \mathcal{J}^2))^{-1}$, and $\mathcal{J}^1 \subset \{1, \ldots, n_1\}$, $\mathcal{J}^2 \subset \{1, \ldots, n_2\}$ are the chosen sets of indices of cardinality r. To select these "cross" indices properly is crucial to ensure good approximation properties. In [84], the following *maximum volume principle* has been proven.

Lemma 2.1.2. If $\mathcal{J}^1$ and $\mathcal{J}^2$ are such that $\det X(\mathcal{J}^1, \mathcal{J}^2)$ is maximal among all $r \times r$ submatrices of X, then

$$\|X - \tilde{X}\|_C \leqslant (r+1) \min_{\mathrm{rank}(Y)=r} \|X - Y\|_2 = (r+1)\varepsilon.$$

Though the exact volume (determinant) maximization is a NP-hard problem, a heuristic quasi-optimal scheme, proposed in [83] (the so-called *maxvol algorithm*), gives satisfactory results in most relevant cases. There are also specific estimates for the cross approximation applied directly to smooth functions, see e.g. [214].

2.1.2 Low-parametric canonical and Tucker formats

A generalization to the higher-dimensional case is not obvious. The first idea that comes in mind is to take a simple sum of rank-1 components (2.1) in the same way as in two dimensions.

Definition 2.1.3. A tensor x is said to be presented (or approximated) in the *Canonical* format if it holds

$$x(i_1, i_2, \ldots, i_d) \approx \sum_{\alpha=1}^{R} x^{(1)}_\alpha(i_1) x^{(2)}_\alpha(i_2) \cdots x^{(d)}_\alpha(i_d). \tag{2.4}$$

This representation *format* is known since [113]. R is called the *canonical rank* of a tensor, and $x^{(k)} \in \mathbb{C}^{n_k \times R}$ are canonical *factors*. It was then used heavily for data structuring under several other names such as CANDECOMP, Canonical Polyadic (CP) format or PARAFAC, see the review [160] and references therein, e.g. [106, 34, 28, 38, 23, 24].

For certain classes of tensors (typically arising from discretizations of integral operators) it is possible to prove existence of low-rank canonical approximations analytically [230, 103, 36, 23, 24, 72, 100, 101, 89, 99, 145, 49]. These proofs are often constructive and allow to build approximations with modest R (e.g. tens or hundreds), such that the total storage reduction to $\mathcal{O}(dnR)$ is reasonable.

However, the analytical considerations are of limited applicability, and provide usually sub-optimal rank estimates. Moreover, there is no robust method to compute the CP format for a given array (data *compression* procedure), since it suffers from an intrinsic instability [42], which prevents efficient calculations. There are several working

techniques (see [60, 133] and reviews [24, 160]), but still they may converge slowly, and require a priori knowledge of the rank.

So, we come to the main problem of tensor approximations: given elements (or a procedure to compute them) of a tensor, find its low-parametric approximation. Since a reliable approach is known only in two dimensions for the dyadic factorization, several ways how to bring it to higher dimensions were developed.

The first representation of that type was the so-called Tucker [228] format.

Definition 2.1.4. A tensor x is said to be presented (or approximated) in the *Tucker* format if it holds

$$x(i_1, \dots, i_d) \approx \sum_{\gamma_1, \dots, \gamma_d} x^{(c)}(\gamma_1, \dots, \gamma_d) x^{(1)}_{\gamma_1}(i_1) \cdots x^{(d)}_{\gamma_d}(i_d). \tag{2.5}$$

The rank indices γ_k vary in the ranges $\gamma_k = 1, \dots, r_k$, where $r_k = r_k(x)$ are the so-called *Tucker*, or *multilinear* ranks, the tensor $x^{(c)} \in \mathbb{C}^{r_1 \times \cdots \times r_d}$ is called the *Tucker core*, and $x^{(k)} \in \mathbb{C}^{n_k \times r_k}$ are the *Tucker factors*.

The first glance at the Tucker format shows that it is not free from the curse of dimensionality: to store the core, still $\mathcal{O}(r^d)$ elements are required. Therefore, for quite a long time it was used only in low-dimensional cases. However, the Tucker format possesses several advantageous properties, and the problem of the core storage can be resolved in a separate way. We address this issue later in Sections 2.1.3 and 2.2.2.

The first positive feature is that the Tucker formats admits a black-box SVD-based approximation procedure, see [39, 40, 41]. It follows from the crucial property that each Tucker rank is in fact a rank of a certain matrix, constructed from the elements of the initial tensor. To show this, we first need to introduce several definitions.

Definition 2.1.5. By a *multiindex* $\boldsymbol{i} = \boldsymbol{i}_{1,\dots,k} = \overline{i_1, \dots, i_k}$ we denote an index which takes all possible combination of values of $i_1, \dots, i_k$, i.e. if $i_m = 1, \dots, n_m$, $m = 1, \dots, k$, then[2]

$$\overline{i_1, \dots, i_k} = i_1 + (i_2 - 1)n_1 + \cdots + (i_k - 1)n_1 \cdots n_{k-1}.$$

This conception of the index grouping is crucial in the description of multidimensional tensors. We may recall the considerations from the first chapter: a discrete solution of a partial differential equation can be seen as a tensor $x(i_1, \dots, i_d)$, but the calculation of the same data via an iterative process with the Galerkin linear system is written in terms of a vector $x(\boldsymbol{i}) = x(\overline{i_1, \dots, i_d})$.

The semantic $\boldsymbol{i}_{1,\dots,k}$ allows us to write encapsulated index sets more briefly in some special cases. For example,

$$\boldsymbol{i}_{\neq k} = \overline{i_1, \dots, i_{k-1}, i_{k+1}, \dots, i_d}, \qquad \boldsymbol{i}_{>k} = \overline{i_{k+1}, \dots, i_d}, \tag{2.6}$$

and so on. Now, we are ready to define the *unfolding matrices*.

[2]We adopt here the *little-endian* convention, which is used in e.g. Arabic numerals, Fortran and MATLAB indexing, contrarily to the *big-endian* used in the positional system or C language. In most of this work the particular endiannes is not important, as soon as the index set is fixed. The Kronecker product introduced in (1.18) is also consistent with the little-endian index grouping.

Definition 2.1.6. Given a tensor $x(i_1, \dots, i_d)$. A k-th Tucker unfolding matrix writes

$$\begin{aligned} x^{[k]} &= \left[x^{[k]}_{i_k, \boldsymbol{i}_{\neq k}}\right] \in \mathbb{C}^{n_k \times n_1 \cdots n_{k-1} n_{k+1} \cdots n_d}, \quad \text{where} \\ x^{[k]}_{i_k, \boldsymbol{i}_{\neq k}} &= x^{[k]}(i_k, \overline{i_1, \dots, i_{k-1}, i_{k+1}, \dots, i_d}) = x(i_1, \dots, i_d). \end{aligned}$$

Assume that the Tucker decomposition (2.5) holds exactly. Denoting an auxiliary matrix

$$V^{[k]}(\gamma_k, \boldsymbol{i}_{\neq k}) = \sum_{\boldsymbol{\gamma}_{\neq k}} x^{(c)}(\boldsymbol{\gamma}) x^{(1)}_{\gamma_1}(i_1) \cdots x^{(k-1)}_{\gamma_{k-1}}(i_{k-1}) x^{(k+1)}_{\gamma_{k+1}}(i_{k+1}) \cdots x^{(d)}_{\gamma_d}(i_d),$$

we may conclude that the Tucker expansion provides a dyadic decomposition for each Tucker unfolding matrix,

$$x^{[k]} = x^{(k)} V^{[k]}, \qquad x^{(k)} \in \mathbb{C}^{n_k \times r_k}, \quad V^{[k]} \in \mathbb{C}^{r_k \times n_1 \cdots n_{k-1} n_{k+1} \cdots n_d}. \tag{2.7}$$

This gives immediately the first corollary: if the Tucker decomposition is exact, the ranks of unfolding matrices are not larger than the Tucker ranks, $\operatorname{rank}(x^{[k]}) \leqslant r_k$, $k = 1, \dots, d$. In fact, we may choose a non-redundant representation and turn the last expression into an equality. The second corollary is even more useful, since it addresses the approximation problem formulated above.

Theorem 2.1.7. Given a tensor $y(i_1, \dots, i_d)$, the error minimization problem in the Tucker format with prescribed ranks $\boldsymbol{r} = (r_1, \dots, r_d)$ has the solution, i.e.[3]

$$\exists x_{min} : \ \|x_{min} - y\| = \inf_x \|x - y\| = \min_x \|x - y\| = \varepsilon,$$

where x (and x_{min}) are representable in the form (2.5) with the Tucker ranks $r_1, \dots, r_d$.

A quasi-optimal approximation x_* ensuring

$$\|x_* - y\| \leqslant \sqrt{d}\varepsilon$$

is provided by computing d SVD decompositions of unfolding matrices.

Proof. Though we could address to [39, 40], we present the proof here, since it will be generalized to some other tensor formats below. To prove the first part, recall that "inf" means that there exists a sequence x_s such that $\lim\limits_{s\to\infty} \|y - x_s\| = \varepsilon$. Since all elements of x_s are bounded, there exists a subsequence x_{s_t}, which is pointwise convergent to x_{min}. Directly from the Tucker form (2.5) the same follows for the unfolding matrices $x^{[k]}_{s_t} \to x^{[k]}_{min}$. Since a sequence of matrices with uniformly bounded ranks cannot converge to a matrix with a larger rank, it holds $\operatorname{rank}(x^{[k]}_{min}) \leqslant r_k$, and in the same time $\|y^{[k]} - x^{[k]}_{min}\| = \varepsilon$. It means that the minimizer exists in the set of tensors with Tucker ranks bounded by r_k, i.e. the first claim of the theorem.

[3] Unless otherwise noted, we use the Frobenius norms for tensors and vectors, $\|x\| = \|x\|_2 = \|x\|_F = \sqrt{\sum\limits_{\boldsymbol{i}} |x(\boldsymbol{i})|^2}$.

The practical computational algorithm, called *HOSVD* [39, 40], resembles (2.7) "backwardly". For each unfolding, compute the truncated SVD,

$$y^{[k]}_{i_k,\boldsymbol{i}_{\neq k}} \approx x^{[k]}_{i_k,\boldsymbol{i}_{\neq k}} = \sum_{\gamma_k=1}^{r_k} x^{(1)}_{\gamma_k}(i_k)\sigma_{\gamma_k}V^{[k]}(\gamma_k,\boldsymbol{i}_{\neq k}),$$

ensuring $\|y^{[k]} - x^{[k]}\| \leqslant \varepsilon_k \leqslant \varepsilon$. The orthogonal matrix of left singular vectors is taken as the k-th Tucker factor, and in the final step, the Tucker core is recovered as a projection

$$x^{(c)}(\gamma_1,\ldots,\gamma_d) = \sum_{\boldsymbol{i}} \left(x^{(1)}_{\gamma_1}(i_1)\right)^* \cdots \left(x^{(d)}_{\gamma_d}(i_d)\right)^* y(i_1,\ldots,i_d).$$

Introduce the orthogonal projectors as follows,

$$P_k = \underbrace{I\otimes\cdots\otimes I}_{k-1} \otimes \left(x^{(k)}\left(x^{(k)}\right)^*\right) \otimes \underbrace{I\otimes\cdots\otimes I}_{d-k},$$

then the final approximant $x_*(\boldsymbol{i})$ may be written as $x_* = P_1\cdots P_d y$[4]. Now, using the "add-and-subtract" trick and orthogonality, deduce

$$\begin{aligned}\|y-x_*\|^2 &= \|(y-P_1y)+(P_1y-P_1P_2y)+(P_1P_2y-P_1P_2P_3y)+\cdots\\ &+ (P_1\cdots P_{d-1}y - P_1\cdots P_d y)\|^2\\ &\leqslant \|y-P_1y\|^2+\|y-P_2y\|^2+\cdots+\|y-P_dy\|^2\\ &\leqslant \varepsilon_1^2+\cdots+\varepsilon_d^2,\end{aligned}$$

which yields immediately the second claim of the theorem. □

Similarly to the two-dimensional case, the actual dependence of the Tucker ranks on the accuracy is governed by the data. The Tucker format has a long history of applications in data mining or image processing (the seminal paper [228] was devoted to chemometrics; many references may be found in the review [160]). Tensor numerical methods for PDEs have originated from a theoretical justification of the exponential convergence rate of the Tucker approximation for tensors obtained from the discretization of analytic functions [72, 142, 100], even with a finite amount of point singularities [150]. The latter paper presents many numerical demonstrations of the exponential convergence rate of the Tucker approximation of functional tensors arising, in particular, in electronic structure calculations. A challenging feature of the canonical and Tucker approximations in numerical discretization of PDEs is a large mode size n, of the order of 10^4–10^5, resulting from a fine grid representation of functions and operators. In this case the standard algorithms described in [160] are no longer applicable. The efficient multigrid method for the Tucker and canonical-to-Tucker approximations of large function-related tensors was proposed in [151].

To estimate the Tucker ranks using the smoothness properties, consider $f(q_1,\ldots,q_d)$ as a univariate function, depending on the other variables as parameters, i.e. $f^{[k]}_{\boldsymbol{q}_{\neq k}}(q_k) =$

[4]Note that we have switched to the vector notation $y = [y(\boldsymbol{i})] \in \mathbb{C}^{n_1\cdots n_d}$ to make the matrix-by-vector products consistent!

$f(q_1, \dots, q_d)$. Then we may apply a polynomial approximation in q_k, which is known to converge exponentially fast with the polynomial degree for analytical functions [20, 225]. On the other hand, $f^{[k]}$ yields naturally the unfolding matrix, with the ε-rank bounded by the polynomial degree. So, the following general statement for the multidimensional case was proven in [142, 72].

Theorem 2.1.8. Assume that an analytic function $f(q_1, \dots, q_d)$, $\boldsymbol{q} \in \Omega = [-1, 1]^d$, is given, and a tensor $x(i_1, \dots, i_d) = f(q_1(i_1), \dots, q_d(i_d))$ is its discretization on a tensor product grid. Suppose f admits an extension to the ρ_k-Bernstein ellipse

$$\mathcal{E}_{\rho_k} = \left\{ z \in \mathbb{C} : \ |1 + z| + |1 - z| \leqslant \rho_k + \frac{1}{\rho_k} \right\}$$

in each variable q_k. Then there exists an ε-approximation to x in the Tucker format with the rank bounds $r_k \leqslant C|\log(\varepsilon)|/\log(\rho_k)$.

We will give the proof and a refined result in connection with the newer Tucker-based format, see Section 2.2.2. Here we mention several successful applications of the Tucker format in integral equations [210, 187, 150, 146] and quantum chemistry [153, 151, 66, 136, 143, 152, 196, 137, 140, 139, 138], as well as improved (e.g. of lower than the HOSVD complexity) general type Tucker approximation methods:

- alternating least squares compression of matrices [118],
- Newton method for the error minimization [36],
- compression from the canonical to the Tucker format by the ALS combined with multigrid [151] and Cholesky decomposition [211], as well as
- ALS- and greedy-type algorithms for Tucker arithmetics (cf. Sec. 2.1.5), i.e. rank reduction of sums and products of Tucker tensors [194, 209, 195, 82].

It is also interesting to note that the same representation was exploited independently in the chemistry community under the name MCTDH [179].

However, to proceed to higher dimensions efficiently, we need some other approach. The most recent surveys and descriptions can be found in [98, 93, 149, 147].

2.1.3 Tensor Trains and trees: recurrent decompositions

The Tucker format separates each variable from the rest of them, starting each time from the initial tensor, which results in a d-dimensional core. To get rid of the curse of dimensionality, we need a way to shrink several variables into the structured form. First, we use the *index grouping* to reduce the dimension formally, and then perform the actual revealing of a reduced subspace. We show two main approaches developed in the numerical linear algebra, both relying essentially on sequential *recurrent* applications of dyadic or Tucker factorizations.

First, we may group all indices pairwise, $\boldsymbol{i}_{k,k+1} = \overline{i_k, i_{k+1}}$, obtain a $d/2$-dimensional tensor, and apply the Tucker decomposition, motivating ourselves with a lower-dimensional

core as a result: if the Tucker ranks remained moderate, the new core is easier to work with. We may stop if the cost $r^{d/2}$ is satisfactory, otherwise proceed further, grouping the indices in the *core*, applying the Tucker decomposition again, obtaining a $d/4$-dimensional core, and so on until we reach a two-dimensional tensor. Due to such hierarchical way of Tucker approximations, the resulting format was called the *Hierarchical Tucker* [105, 90, 166], or Multilevel-MCTDH (ML-MCTDH) [179]. Since the indices of the initial tensor may have a special role (e.g. the discretized coordinates), the first Tucker step is usually done without the binary grouping.

Definition 2.1.9. A tensor x is said to be presented (or approximated) in the (binary) Hierarchical Tucker (HT) format if there exists the following sequence of tensors, each being the Tucker core for the forthcoming one:

$$x^{(L+1)}(\gamma_1^{L+1},\dots,\gamma_{d_{L+1}}^{L+1}) \approx \sum_{\gamma_1^1,\dots,\gamma_{d_L}^L} \prod_{l=1}^{L}\prod_{k_l=1}^{d_l} x_{\gamma_{k_l}^l}^{(l,k_l)}(\gamma_{2k_l-1}^{l+1},\gamma_{2k_l}^{l+1}), \tag{2.8}$$

where the *levels* run in the range $L = 1,\dots,\lceil\log_2 d\rceil - 1$, and the last level is associated with the initial Tucker core of (2.5), $x^{(\lceil\log_2 d\rceil)} = x^{(c)}$; the dimensions are reduced recurrently as $d_{\lceil\log_2 d\rceil} = d$, $d_L = \lceil d_{L+1}/2\rceil$, and the building blocks $x^{(l,k_l)} \in \mathbb{C}^{r_{k_l}^l \times r_{2k_l-1}^{l+1} \times r_{2k_l}^{l+1}}$ are called the *HT factors* with the *HT ranks* $r_{k_l}^l$. If $2k_l < d_{l+1}$, the last index is agreed to degenerate to the range $\{1\}$.

The factors are the main ingredient of such decomposition, since they are only 3-dimensional tensors, accounting for the mapping of the subspace of size $r_{2k_l-1}^{l+1} r_{2k_l}^{l+1}$, obtained by the direct product of two variables at the higher level, to a subspace of size $r_{k_l}^l$ at the lower level. Each rank $r_{k_l}^l$ represents thus a dyadic separation rank between a group of initial variables (belonging to the current branch of the tree) and the rest of them. If these ranks are bounded by a moderate constant, the total storage of the HT format scales as $\mathcal{O}(dnr + dr^3)$ (the first term comes from the initial Tucker factors, and the second from the hierarchical ones), and the memory reduction is significant w.r.t. n^d.

Obviously, the ranks will depend on the order, in which the dimensions are plugged into the tree (i.e. the index grouping). One may easily show examples where one particular order will yield significantly smaller ranks than another. Nevertheless, in any tree structure there are edges corresponding to the separation of nearly half of dimensions from the others, which are likely to manifest the highest ranks. Taking into account also the practical experience, we believe that the *order* of the dimensions is indeed important (and it may be determined from the physical reasons, or using adaptive algorithms, e.g. [10]), but the indices may be grouped in most cases in a linear fashion starting from the first dimension, i.e. fixing the hierarchy tree to the *unbalanced linear* form.

Definition 2.1.10. A tensor x is said to be presented (or approximated) in the *Matrix Product States* (MPS), or *Tensor Train* (TT) format, if it holds

$$x(i_1,\dots,i_d) \approx \sum_{\alpha_1,\dots,\alpha_{d-1}} x_{\alpha_1}^{(1)}(i_1) x_{\alpha_1,\alpha_2}^{(2)}(i_2)\cdots x_{\alpha_{d-2},\alpha_{d-1}}^{(d-1)}(i_{d-1}) x_{\alpha_{d-1}}^{(d)}(i_d), \tag{2.9}$$

where $x^{(k)} \in \mathbb{C}^{r_{k-1} \times n_k \times r_k}$ are called TT *cores* (*blocks*, *factors*), the ranges $r_k = r_k(x) \leqslant r(x)$ of *rank indices* $\alpha_k = 1, \ldots, r_k$ are called *TT ranks*, and it is agreed that $r_0 = r_d = 1$ for uniformity.

As many useful techniques, this format was discovered several times, as MPS in quantum physics since late 80's [2, 62, 158, 237, 201], and independently in the numerical linear algebra as TT in 2009 [197, 188, 191]. The term Matrix Product States appears from the fact that if we fix $i_1, \ldots, i_d$, the TT blocks become matrices of sizes $r_{k-1} \times r_k$, and an element of x writes as a product of matrices,

$$x(i_1, \ldots, i_d) = x^{(1)}(i_1) \cdots x^{(d)}(i_d).$$

Notice that rank-1 tensors are invariant in all formats: if $r_{k-1} = r_k = 1$, the above expression is a product of numbers, coinciding with the CP format (2.4) with $R = 1$ and the Tucker format (2.5) with $x^{(c)} = 1$. Alternatively, fixing the rank indices, one may write

$$x = \sum_{\alpha_1, \ldots, \alpha_{d-1}} x^{(1)}_{\alpha_1} \otimes x^{(2)}_{\alpha_1, \alpha_2} \otimes \cdots \otimes x^{(d-1)}_{\alpha_{d-2}, \alpha_{d-1}} \otimes x^{(d)}_{\alpha_{d-1}}, \tag{2.10}$$

where $\otimes$ is the usual Kronecker product (see Eq. 1.18).

Besides the linear structure, another interesting difference from the (Hierarchical) Tucker format is that neither of TT blocks can be clearly identified as "factor" or "core" (thus "block", "factor" and "core" may be used interchangeably), since all of them carry both initial and auxiliary rank indices. This simplifies both the analytical presentations and implementation of algorithms. The asymptotic storage of the TT format is $\mathcal{O}(dnr^2)$, which is smaller than in the HT format for higher ranks and small ranges n of the initial indices i_k (the *mode sizes*), which is the case in e.g. spin systems. If the mode sizes are large, one may apply the TT format to the Tucker core only, obtaining the so-called *Extended TT* decomposition [198], which is a special case of the HT format with a linear tree.

In the rest of the work, we overview existing and present new computational tensor product methods and constructions. Except the sections devoted to the special combined format (Sec. 2.2.2–2.2.5), we will explain the analysis and algorithms on the base of the TT format. Its simplicity allows to develop elegant notations and get the ideas across without burying them under lots of indices.

2.1.4 Tensor product notations

To begin with, consider the Matrix-by-Vector product in a large dimension,

$$y = Ax, \quad x, y \in \mathbb{C}^{n^d}, \text{ i.e. } \quad y(\boldsymbol{i}) = \sum_{\boldsymbol{j}} A(\boldsymbol{i}, \boldsymbol{j}) x(\boldsymbol{j}),$$

where $\boldsymbol{i} = \overline{i_1, \ldots, i_d}$ and $\boldsymbol{j} = \overline{j_1, \ldots, j_d}$. Suppose that x is presented in the TT format, and we would like to preserve it for y, staying within the tractable complexity. What representation should we choose for A? The first idea that comes in mind is just a $2d$-dimensional TT, $A^{(1)}(i_1) \cdots A^{(2d)}(j_d)$. However, in this case $r_d = \operatorname{rank}(A) = n^d$ almost always (even for the identity $A = I$).

Therefore, the TT format for matrices, or *Matrix Product Operator* is introduced using the permuted and grouped indices,

$$A(i_1,\dots,i_d,j_1,\dots,j_d) = \sum_{\gamma_1,\dots,\gamma_{d-1}} A^{(1)}_{\gamma_1}(i_1,j_1)A^{(2)}_{\gamma_1,\gamma_2}(i_2,j_2)\cdots A^{(d)}_{\gamma_{d-1}}(i_d,j_d). \qquad (2.11)$$

Then the Matrix-Vector product writes independently for each TT block: the result is the TT format $y(\boldsymbol{i}) = y^{(1)}(i_1)\cdots y^{(d)}(i_d)$ with

$$y^{(k)}_{\beta_{k-1},\beta_k} = A^{(k)}_{\gamma_{k-1},\gamma_k}x^{(k)}_{\alpha_{k-1},\alpha_k}, \quad \text{i.e.} \quad y^{(k)}_{\beta_{k-1},\beta_k}(i_k) = \sum_{j_k} A^{(k)}_{\gamma_{k-1},\gamma_k}(i_k,j_k)x^{(k)}_{\alpha_{k-1},\alpha_k}(j_k), \quad (2.12)$$

where $\beta_m = \overline{\alpha_m,\gamma_m} = 1,\dots,r_k(A)r_k(x)$, $m = 0,\dots,d$. Moreover, this conception naturally resolves to the standard Kronecker products if all TT ranks are ones. Otherwise, the TT ranks of the matrix and vector multiply, i.e. $r_k(y) = r_k(A)r_k(x)$, which is still feasible if $r(A)$ and $r(x)$ are moderate.

So, our first special definition will be the common counterpart of (2.9) and (2.11), a slightly extended version of notations used in [207].

Definition 2.1.11. Given TT formats $\{x^{(k)}(i_k)\}$ or $\{A^{(k)}(i_k,j_k)\}$. A (vector) *tensor train map* expands a set of TT blocks to a TT tensor as follows,

$$\tau(x^{(p)},\dots,x^{(q)}) : \{x^{(k)}\}_{k=p}^{q} \to x^{(p,\dots,q)} \in \mathbb{C}^{r_{p-1}\times n_p\cdots n_q\times r_q}, \quad \text{where}$$

$$x^{(p,\dots,q)}_{\alpha_{p-1},\alpha_q}(\overline{i_p,\dots,i_q}) = \sum_{\alpha_p,\dots,\alpha_{q-1}} x^{(p)}_{\alpha_{p-1},\alpha_p}(i_p)\cdots x^{(q)}_{\alpha_{q-1},\alpha_q}(i_q),$$

for $1 \leqslant p \leqslant q \leqslant d$. For boundary cases we agree that

$$x^{(1,\dots,q)} = x^{(\leqslant q)} = x^{(<q+1)} \in \mathbb{C}^{n_1\cdots n_q\times r_q}, \quad x^{(p,\dots,d)} = x^{(\geqslant p)} = x^{(>p-1)} \in \mathbb{C}^{r_{p-1}\times n_p\cdots n_d},$$

and $x^{(1,\dots,d)} = x \in \mathbb{C}^{n_1\cdots n_d}$.

A matrix tensor train map reads

$$\tau(A^{(p)},\dots,A^{(q)}) : \{A^{(k)}\}_{k=p}^{q} \to A^{(p,\dots,q)} \in \mathbb{C}^{r_{p-1}\times n_p\cdots n_q\times m_p\cdots m_q\times r_q}, \quad \text{where}$$

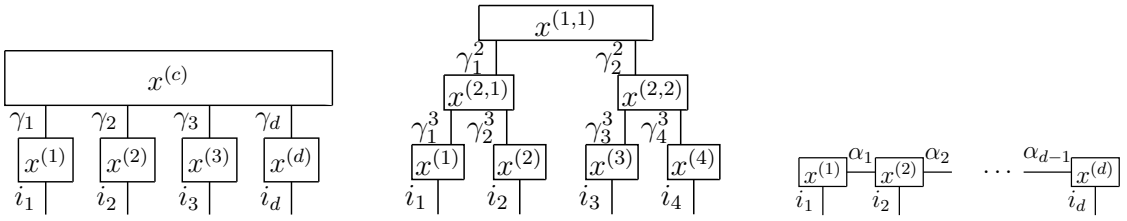
$$A^{(p,\dots,q)}_{\gamma_{p-1},\gamma_q}(\overline{i_p,\dots,i_q},\overline{j_p,\dots,j_q}) = \sum_{\gamma_p,\dots,\gamma_{q-1}} A^{(p)}_{\alpha_{p-1},\alpha_p}(i_p,j_p)\cdots A^{(q)}_{\gamma_{q-1},\gamma_q}(i_q,j_q),$$

with the analogous boundary notations.

The map τ may be used for the extraction of a tensor train *chunk* – an operation, on which the alternating optimization methods rely heavily. In addition, note that in the image of τ, the initial indices i_k are always glued into a multiindex, which will allow us to write consistent matrix products (cf. the boundary cases: these chunks are just matrices).

Another useful type of index grouping may be devised from the conception of subspaces in the HT format (2.8): recall that each factor $x^{(l,k_l)}$ may be seen as a map $\mathbb{C}^{r^{l+1}_{2k_l-1}r^{l+1}_{2k_l}} \to \mathbb{C}^{r^l_{k_l}}$, associated with a matrix instead of a 3-index array. The same consideration may be applied to the TT factors, with the maps defined as follows.

Figure 2.1: Tucker (left), HT (middle), and TT/MPS (right) tensor formats depicted via tensor network diagrams



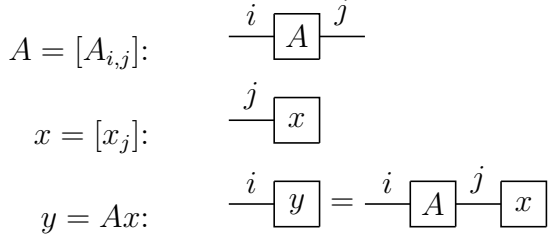
Definition 2.1.12. Given a TT block $x^{(k)}$, introduce the following reshapes:

- *Left-folded block*: $x^{|k\rangle}(\overline{\alpha_{k-1}, i_k}, \alpha_k) = x^{(k)}_{\alpha_{k-1},\alpha_k}(i_k), \quad x^{|k\rangle} \in \mathbb{C}^{r_{k-1}n_k \times r_k}$.
- *Right-folded block*: $x^{\langle k|}(\alpha_{k-1}, \overline{i_k, \alpha_k}) = x^{(k)}_{\alpha_{k-1},\alpha_k}(i_k), \quad x^{\langle k|} \in \mathbb{C}^{r_{k-1} \times n_k r_k}$.
- *Center-folded block*: $x^{|k|}(i_k, \overline{\alpha_{k-1}, \alpha_k}) = x^{(k)}_{\alpha_{k-1},\alpha_k}(i_k), \quad x^{|k|} \in \mathbb{C}^{n_k \times r_{k-1} r_k}$.

It is important that all folded versions point to the same data stored in a TT block – the well-known *pointer* conception in programming. It will allow us to assign e.g. $x^{|k\rangle} = Q$, without a need to state explicitly $x^{(k)}_{\alpha_{k-1},\alpha_k}(i_k) = x^{|k\rangle}(\overline{\alpha_{k-1}, i_k}, \alpha_k)$ afterwards, since all reshapes share the same entries by default.

As the fourth "silent" reshape, we reuse the initial notation $x^{(k)}$ for a vector of TT block elements, $\left[x^{(k)}(\overline{\alpha_{k-1}, i_k, \alpha_k})\right]$, exactly in the same way as $x(\boldsymbol{i})$ and $x(i_1, \ldots, i_d)$ are unified. The meaning of $x^{(k)}$ will be unambiguous from the context: $x^{(k)}(i_k)$ in the tensor train (2.9) is still a two-dimensional slice from a three-dimensional tensor, while the matrix product of the form $A_k x^{(k)}$ assumes that the elements of $x^{(k)}$ are stretched in a column vector.

Another notation that became popular in the physics community is the *tensor network diagrams* [238, 201, 117], especially convenient to describe complicated formats. The diagrammatic formalism denotes any array as a block, any index as a line, if a line connects two blocks, it means that we multiply the elements of the corresponding arrays, and sum over the common index. The indices hanging from only one block are "free", and an equation is supposed to hold for all possible values of free indices. To "quick start", a matrix, a vector, and their product may be written as follows:



The Tucker, HT and TT formats may be denoted as shown in Fig. 2.1.

2.1.5 Principal operations in the TT format

We showed already the Matrix-Vector product (2.12). Other multilinear operations may be also generalized from the Canonical format. For example, to multiply a tensor by a scalar, it is enough to multiply any of its TT blocks.

The **summation** of two tensors joins the ranges of the rank indices, so that $z = x+y$ is represented in the TT format with the following blocks,

$$z^{(1)}(i_1) = \begin{bmatrix} x^{(1)}(i_1) & y^{(1)}(i_1) \end{bmatrix}, \quad z^{(d)}(i_d) = \begin{bmatrix} x^{(d)}(i_d) \\ y^{(d)}(i_d) \end{bmatrix}, \quad z^{(k)}(i_k) = \begin{bmatrix} x^{(k)}(i_k) & \\ & y^{(k)}(i_k) \end{bmatrix}$$

for $k = 2, \ldots, d-1$. Obviously, the summation adds TT ranks, $r_k(z) = r_k(x) + r_k(y)$, $k = 1, \ldots, d-1$.

Remark 2.1.13. This procedure can be used to convert the canonical format (2.4) to the TT representation. Indeed, each rank-1 component of the canonical format may be seen as a rank-1 TT tensor, and the summation writes as shown above. It follows that if x is given in the canonical form, its TT ranks are also bounded, $r_k(x) \leqslant R$.

The **diagonal** matrix is constructed from a vector in the TT format with no change of TT ranks, by constructing the diagonal matrix over each i_k,

$$A^{(k)}(i_k, j_k) = x^{(k)}(i_k)\delta_{i_k,j_k}, \quad A = \tau(A^{(1)}, \ldots, A^{(d)}) = \operatorname{diag}(x) = \operatorname{diag}\left(\tau\left(x^{(1)}, \ldots, x^{(d)}\right)\right).$$

Vice versa, the vector of diagonal elements may be extracted from a matrix.

The **pointwise** (Hadamard) product of vectors $z = x \odot y$ may be computed as a product of a diagonal matrix by vector, $x \odot y = \operatorname{diag}(x)y$, applying (2.12):

$$z = \tau\left(z^{(1)}, \ldots, z^{(d)}\right), \quad z^{(k)}_{\beta_{k-1},\beta_k} = x^{(k)}_{\gamma_{k-1},\gamma_k} \odot y^{(k)}_{\alpha_{k-1},\alpha_k},$$

which also yields the multiplication of ranks, $\beta_k = \overline{\alpha_k, \gamma_k} = 1, \ldots, r_k(z) = r_k(x)r_k(y)$.

The **scalar** (dot, inner) product $s = (x, y)$ of two vectors is equal to a product of all TT elements of both vectors, followed by a summation over all rank and initial indices. However, the straightforward implementation would require $\mathcal{O}(dnr^2(x)r^2(y)) = \mathcal{O}(dnr^4)$ complexity. Using the blocks folding (def. 2.1.12) and auxiliary quantities, the scalar product may be computed with $\mathcal{O}(dn(r^2(x)r(y)+r(x)r^2(y)) = \mathcal{O}(dnr^3)$ cost, as shown in Algorithm 1.

Now let us focus on the main advantage of tensor trees – the **re-compression**, or **rounding** procedure. We saw that algebraic operations may increase TT ranks, such that they may be unnecessarily large for the output data. The TT rounding is an analog of the HOSVD for the quasi-optimal rank reduction up to a given accuracy. Given a tensor x, we will denote its compression to y as follows,

$$y = \mathcal{T}_\varepsilon(x), \quad y = \mathcal{T}_r(x), \quad y = \mathcal{T}_{\varepsilon,r}(x),$$

where subscripts of $\mathcal{T}$ denote the compression strategies:

- ε-strategy: require $\|y-x\| \leqslant \varepsilon\|x\|$, ranks are as small as possible but not limited.

Algorithm 1 Scalar product in the TT format

Require: Tensors x, y in the TT format.
Ensure: Scalar product $s = (x, y)$.
1: Initialize $s_0 = 1$.
2: **for** $k = 1, \ldots, d$ **do**
3: $\quad z^{\langle k|} = s_{k-1} y^{\langle k|} \in \mathbb{C}^{r_{k-1}(x) \times n_k r_k(y)}$.
4: $\quad s_k = \left(x^{|k\rangle}\right)^* z^{|k\rangle} \in \mathbb{C}^{r_k(x) \times r_k(y)}$.
5: **end for**
6: **return** $s = s_d \in \mathbb{C}^{r_d(x) \times r_d(y)} = \mathbb{C}$.

- r-strategy: limit $r(y) \leqslant r$, unpredictable but quasi-optimal resulting accuracy.
- ε, r-strategy: follow ε-strategy when possible, but limit $r_k(y) = r$ if ε-truncation suggests a larger value.

So, let us describe the TT rounding following [199].

We begin with the definition of the TT version of the unfolding matrices.

Definition 2.1.14. Given a tensor $x(i_1, \ldots, i_d)$. A k-th TT unfolding matrix writes

$$\begin{aligned} x^{\{k\}} &= \left[x^{\{k\}}_{\boldsymbol{i}_{\leqslant k}, \boldsymbol{i}_{>k}}\right] \in \mathbb{C}^{n_1 \cdots n_k \times n_{k+1} \cdots n_d}, \quad \text{where} \\ x^{\{k\}}_{\boldsymbol{i}_{\leqslant k}, \boldsymbol{i}_{>k}} &= x^{\{k\}}(\overline{i_1, \ldots, i_k}, \overline{i_{k+1}, \ldots, i_d}) = x(i_1, \ldots, i_d). \end{aligned}$$

If (2.9) holds exactly, one may notice immediately from the multilinearity that $r_k = \operatorname{rank}(x^{\{k\}})$. In an inexact case, given $x(i_1, \ldots, i_d) = x^{(1)}(i_1) \cdots x^{(d)}(i_d)$ with TT ranks $\hat{r}_k$, we would like to determine an optimal rank r_k, yielding an ε-approximation to the k-th unfolding matrix. Starting from $k = d$, obtain

$$x^{\{d\}}_{\boldsymbol{i}_{<d}, i_d} = \sum_{\alpha_{d-1}=1}^{\hat{r}_{d-1}} x^{(<d)}_{\alpha_{d-1}}(\boldsymbol{i}_{<d}) x^{(d)}_{\alpha_{d-1}}(i_d), \quad x^{(<d)} = \tau(x^{(1)}, \ldots, x^{(d-1)}). \tag{2.13}$$

Supposing there is a QR decomposition

$$x^{(<d)} = QR, \quad Q^*Q = I,$$

we may cast the SVD problem to a small $\hat{r}_{d-1} \times n_d$ matrix

$$\left(Rx^{\langle d|}\right) \approx U \Sigma V.$$

Filtering r_{d-1} senior singular values, one obtains immediately $y^{\langle d|} = V$. However, $x^{(<d)}$ is a very large matrix, and its orthogonalization should be computed in a structured way.

Definition 2.1.15. A TT block $x^{(k)}$ is said to be *left-* or *right-orthogonal*, if it holds

$$\left(x^{|k\rangle}\right)^* x^{|k\rangle} = I, \quad \text{or} \quad x^{\langle k|} \left(x^{\langle k|}\right)^* = I,$$

respectively.

Note that for the first and the last blocks, the left and right orthogonalities mean simply the orthogonality w.r.t. the tensor indices i_1 and i_d, in the same way as in the dyadic decomposition of a matrix. Since the TT representation is not unique,

$$x(\boldsymbol{i}) = \left(x^{(1)}(i_1)R_1\right)\left(R_1^{-1}x^{(2)}(i_2)R_2\right)R_2^{-1}\cdots R_{d-1}\left(R_{d-1}^{-1}x^{(d)}(i_d)\right)$$

for any nonsingular matrix R_k of proper sizes, the orthogonalities may be ensured without changing the whole tensor. Indeed, neighboring TT blocks $x^{|k\rangle}x^{\langle k+1|}$ constitute a dyadic decomposition, which may be orthogonalized either as

$$q^{|k\rangle}\left(Rx^{\langle k+1|}\right), \quad \text{or} \quad \left(x^{|k\rangle}L\right)q^{\langle k+1|},$$

where $q^{|k\rangle}R = x^{|k\rangle}$ or $Lq^{\langle k+1|} = x^{\langle k+1|}$, respectively. Repeating this procedure for all blocks (see Algorithms 2 and 3) yields a TT representation with the corresponding orthogonality. The complexity of this procedure is $\mathcal{O}(dnr^3)$, but the orthogonality may be ensured for large TT chunks.

Lemma 2.1.16. For a TT tensor with blocks $1,\ldots,d-1$ left-orthogonal, it holds

$$\left(x^{(<p)}\right)^* x^{(<p)} = I, \quad p = 2,\ldots,d.$$

Proof. The product

$$\left(\left(x^{(<p)}\right)^* x^{(<p)}\right)_{\alpha_{p-1},\beta_{p-1}} = \sum_{i_1,\ldots,i_{p-1}} \bar{x}^{(<p)}_{\alpha_{p-1}}(\overline{i_1,\ldots,i_{p-1}})\cdot x^{(<p)}_{\beta_{p-1}}(\overline{i_1,\ldots,i_{p-1}})$$

may be computed as an unfinished scalar product (Alg. 1) with $y = x$, interrupting the process at s_{p-1}. If $x^{(1)}$ is left-orthogonal, from the first step it holds $s_1 = I$. Let $s_{k-1} = I$, then $z^{\langle k|} = x^{\langle k|}$, where $x^{|k\rangle}$ is also left-orthogonal for $k < p$. Therefore, s_k will be also the identity, and the proofs completes by the recursion. □

Now, it is clear how to build the TT rounding procedure: in the first step, we run Alg. 2, making the TT format left-orthogonal, and so will be $x^{(<d)}$ in (2.13). Then we perform a small-sizes SVD $\left(Rx^{\langle d|}\right) \approx U\Sigma V$ and cast $U\Sigma$ to the block $x^{(d-1)}$. Setting $y^{\langle d|} = V$ ensures its right orthogonality, while the left orthogonality of $x^{(<d-1)}$ also takes place. So we may perform the SVD of the block $x^{\langle d-1|}$ and so on. The whole procedure is summarized in Algorithm 4.

In diagrammatic notations, left-, resp. right-orthogonal tensor trains may be shown as in Fig. 2.2: summing the block multiplied with itself over the indices emerging from the filled part, we obtain the identity matrix w.r.t. the indices emerging from the blank part of the block. In addition, in Fig. 2.2 (right), the reduced sizes of the blocks denote the reduced TT ranks after the approximation procedure.

As well as the QR decomposition, each SVD of a $\hat{r} \times n\hat{r}$ matrix takes $\mathcal{O}(n\hat{r}\cdot\hat{r}^2)$ operations, resulting in $\mathcal{O}(dn\hat{r}^3)$ overall complexity. The quasi-optimal error accumulation $\|x-y\|^2 \leqslant \|x\|^2\sum\varepsilon_k^2$ may be proven using the very same projection arguments that were used in the HOSVD Theorem 2.1.7.

If all elements of a tensor are given, and we would like to compress it into the TT format, Algorithm 4 may be reused, omitting the left orthogonalization step (the SVD

Algorithm 2 Left TT orthogonalization

Require: A tensor x in the TT format.
Ensure: A tensor x with all TT blocks except $x^{(d)}$ left-orthogonal.
1: **for** $k = 1, \dots, d-1$ **do**
2: Find QR decomposition $x^{|k\rangle} = q^{|k\rangle} R$, $\left(q^{|k\rangle}\right)^* q^{|k\rangle} = I$.
3: Replace $x^{\langle k+1|} = R x^{\langle k+1|}$, $x^{|k\rangle} = q^{|k\rangle}$.
4: **end for**

Algorithm 3 Right TT orthogonalization

Require: A tensor x in the TT format.
Ensure: A tensor x with all TT blocks except $x^{(1)}$ right-orthogonal.
1: **for** $k = d, d-1, \dots, 2$ **do**
2: Find LQ decomposition $x^{\langle k|} = L q^{\langle k|}$, $q^{\langle k|} \left(q^{\langle k|}\right)^* = I$.
3: Replace $x^{|k-1\rangle} = x^{|k-1\rangle} L$, $x^{\langle k|} = q^{\langle k|}$.
4: **end for**

will consider the whole tensor anyway), and replacing the block $x^{\langle k|}$ by a large tensor $X^{\langle k|} = \left[X^{\langle k|}(\overline{i_1, \dots, i_{k-1}}, \overline{i_k, \alpha_k})\right]$ for $k = 1, \dots, d-1$, and $X^{\langle d|} = x^{\{d\}}$.

However, the SVD-based algorithm may be too expensive to be applied to the full tensor, and even to a TT-structured input, but of very high ranks. The latter case may come from the Matrix-Vector product (2.12) with $r(A) \sim r(x) \lesssim r$, such that $r(y) \sim r^2$, so its rounding takes $\mathcal{O}(dnr^6)$ operations. In this situation, there exist faster, but heuristic techniques. For example, various higher-dimensional generalization of the cross interpolation (2.3) may be used to recover a tensor from a few amount of entries [199, 213, 59, 15, 11, 212]. If a fast *partial* scalar product (like the one used in Lemma 2.1.16) of the input and an approximant is available, which is the case in the approximation of e.g. Matrix-Vector product [190], a more reliable Alternating Least Squares approach may be proposed. This framework is very fruitful and elegant, but several issues are connected with its robustness and adaptivity, and we devote a separate section 4.2 for that.

2.2 Quantized tensor approximation

2.2.1 Quantized Tensor Train

Though tensor formats were originally developed to cope with essentially high-dimensional arrays, the linear in the dimension complexity of the TT format motivates to apply this tool to some "lower"-dimensional data, coming e.g. from a discretization of 1D–2D PDEs. We saw already how tensors generated by multivariate functions may be treated as vectors or matrices (cf. Def 2.1.12). This is the standard reshaping (or folding) operation. However, it can be conducted in the opposite direction: a given vector may be recast to a tensor and approximated in a separable format. What compression rate can be achieved?

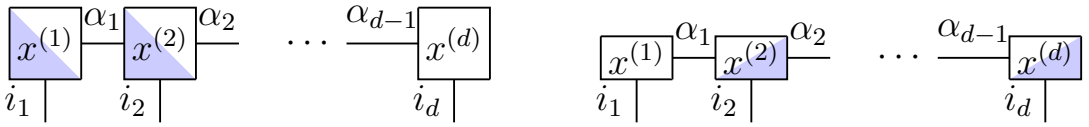
Consider for brevity a one-dimensional vector $x = [x(i)]$. Suppose the number of

Algorithm 4 TT rounding (right-to-left)

Require: A tensor x in the TT format, accuracies ε_k, $k = 1, \dots, d-1$.
Ensure: A tensor $y : \|x - y\|^2 \leqslant \|x\|^2 \sum \varepsilon_k^2$ with optimal ranks.

1: Apply left orthogonalization Algorithm 2 to x.
2: **for** $k = d, d-1, \dots, 2$ **do**
3: Compute SVD $x^{\langle k|} = U \operatorname{diag}(\sigma) V$.
4: Determine rank r_{k-1} : $\sum_{\beta > r_{k-1}} \sigma_\beta^2 \leqslant \varepsilon_{k-1}^2 \|\sigma\|^2$.
5: Take r_{k-1} components $\tilde{U}_\beta = U_\beta$, $\tilde{V}_\beta = V_\beta$, $\tilde{\sigma}_\beta = \sigma_\beta$, $\beta = 1, \dots, r_{k-1}$.
6: Replace $x^{|k-1\rangle} = x^{|k-1\rangle} \cdot \tilde{U} \operatorname{diag}(\tilde{\sigma})$, $\quad y^{\langle k|} = \tilde{V}$.
7: **end for**
8: $y^{|1\rangle} = x^{|1\rangle}$.

Figure 2.2: Diagrams of the TT format states after Alg. 2 (left) and Alg. 4 (right).



admissible values for each i is a power of 2, i.e. $n = 2^L$. Looking at the positional representation of i with binary digits, obtain

$$i = \sum_{l=1}^{L} i_l \cdot 2^{l-1}, \quad i_l \in \{0, 1\}. \tag{2.14}$$

It corresponds to the reshaping of a vector into a tensor with L *virtual quantized* dimensions. Now the TT decomposition can be applied,

$$x(i) = x(i_1, \dots, i_L) \approx x^{(1)}(i_1) \cdots x^{(L)}(i_L).$$

If the TT ranks of this tensor are small, then the storage is *logarithmic*, $\mathcal{O}(L \cdot 2 \cdot r^2) = \mathcal{O}(\log n)$. This technique was proposed in [189] for $2^L \times 2^L$ matrices and illustrated by numerical examples. In the case of vectors and high-order tensors this approach was developed and theoretically justified in [144, 148] under the name *Quantized Tensor Train* (QTT) approximation. The name QTT is due to the term "quant", the minimal possible portion of information gained from determination of each digit i_l.

The usual question one always has to ask working with data-sparse representations, is what are the particular values of the TT (or *QTT*) ranks r. For many one-dimensional function-related vectors beautiful analytic QTT constructions have been discovered. We give here the simplest and most important examples, providing the theoretical background for the QTT approximation theory. More elaborated analytical TT structures, obtained in connection with the author's research, will be shown in the next chapter.

1. Exponential vector has all QTT ranks equal 1 [148],

$$\exp(\kappa i) = \mathrm{e}^{(1)}(i_1) \cdots \mathrm{e}^{(L)}(i_L), \quad \mathrm{e}^{(l)}(i_l) = \exp(\kappa i_l \cdot 2^{l-1}),$$

the indices $i_l \in \{0,1\}$ are according to (2.14). It requires thus $2L = 2\log_2 n$ numbers to define all 2^L entries of the vector exactly.

2. Identity vector (QTT ranks 1) [148] writes,

$$e_j(i) = \delta_{i,j} = \delta^{(1)}(i_1)\cdots\delta^{(L)}(i_L), \quad \delta^{(l)}(i_l) = \delta_{i_l,j_l}, \quad j = \sum_{l=1}^{L} j_l \cdot 2^{l-1}, \quad \delta_{i,j} = \begin{cases} 1, & i=j, \\ 0, & i \neq j. \end{cases}$$

3. Sine vector $\sin(\kappa i + \phi) = s^{(1)}(i_1)\ldots s^{(L)}(i_l)$ is of QTT ranks 2 [148],

$$s^{(1)}(i_1) = \begin{bmatrix} \sin(\kappa i_1 + \phi) & \cos(\kappa i_1 + \phi) \end{bmatrix}, \quad s^{(L)}(i_L) = \begin{bmatrix} \cos(\kappa i_L \cdot 2^{L-1}) \\ \sin(\kappa i_L \cdot 2^{L-1}) \end{bmatrix},$$
$$s^{(l)}(i_l) = \begin{bmatrix} \cos(\kappa i_l \cdot 2^{l-1}) & -\sin(\kappa i_l \cdot 2^{l-1}) \\ \sin(\kappa i_l \cdot 2^{l-1}) & \cos(\kappa i_l \cdot 2^{l-1}) \end{bmatrix}, \quad \text{for} \quad l = 2, \ldots, L-1.$$

4. Polynomial $\sum_{m=0}^{p} a_m i^m = P^{(1)}(i_1)\cdots P^{(L)}(i_L)$ (QTT ranks $p+1$) [148], [192], [91],

$$P^{(1)}(i_1) = \begin{bmatrix} \sum_{m=0}^{p} a_m C_m^0 i_1^m & \sum_{m=1}^{p} a_m C_m^1 i_1^{m-1} & \cdots & \sum_{m=p-1}^{p} a_m C_m^{p-1} i_1^{m-p+1} & a_p \end{bmatrix},$$
$$P^{(L)}(i_L) = \begin{bmatrix} 1 \\ i_L \\ i_L^2 \\ \vdots \\ i_L^p \end{bmatrix}, \quad P^{(l)}(i_l) = \begin{bmatrix} C_0^0 & & & & \\ C_1^1 i_l^1 & C_1^0 & & & \\ C_2^2 i_l^2 & C_2^1 i_l^1 & C_2^0 & & \\ \vdots & & & \ddots & \\ C_p^p i_l^p & & \cdots & C_p^1 i_l^1 & C_p^0 \end{bmatrix},$$

for $l = 2, \ldots, L-1$, where $C_m^k = m!/(k!(m-k)!)$.

The last example expands the concept for almost any smooth function, which may be approximated via the polynomial interpolation (in the similar way as in Theorem 2.1.8 for Tucker rank estimation). For example, the vector with elements $1/(1+i)$ can be numerically approximated with QTT ranks bounded by 8 up to the accuracy 10^{-10}, independently on the length 2^L.

Moreover, the QTT format applied to matrices (recall Matrix TT (2.11)) allows simple constructive representations of basic operators (Laplace and gradient on a uniform grid, or shift), see [131] and Section 3.1, and further results in [44] and Section 3.2.1.

The logarithmic complexity w.r.t. the cardinality of the initial tensor makes the QTT format a very promising tool for large-scale problems. There are algorithms, which rely essentially on the binary QTT structure, such as the super-fast data-sparse Fourier transform [50], convolution [127] and wavelet transforms [154, 132], all with logarithmic complexity scaling. Introducing the QTT format in time variable for the simultaneous discretization presented in Section 1.3, we build a very efficient time integration scheme, see also Section 3.1 and [47, 73].

2.2.2 QTT-Tucker: two-level separation of initial and virtual dimensions

Surely, the QTT decomposition is not limited to one dimensional problems, but may be employed in addition to the "traditional" usage of formats, when only the initial indices $i_1, \dots, i_d$ with (possibly) large ranges $n_1, \dots, n_d$ are separated. Indeed, each i_k may be further coded via (2.14),

$$i_k = \sum_{l=1}^{L_k} i_{k,l} \cdot 2^{l-1}, \quad i_{k,l} \in \{0, 1\},$$

so that the global index reads

$$\boldsymbol{i} = \overline{i_{1,1}, \dots, i_{1,L_1}, i_{2,1}, \dots, i_{d,L_d}}.$$

Enumerating the entries of a tensor with indices $\{i_{k,l}\}$, one obtains a $\sum L_k \leqslant dL$-dimensional tensor, which is then compressed in a tensor format.

What particular format is favorable for this purpose? Since all *new* mode sizes are small, $\#\{i_{k,l}\} = 2$, the TT (or QTT if we stress the use of quantization; the term QTT is established for dL-dimensional tensors as well) representation with the smallest asymptotic rank complexity $\mathcal{O}(dLr^2)$ looks perfect at the first glance. It does indeed a very good job in many cases if the TT ranks are moderate. However, the separation ranks of virtual dimensions (especially near the middle of the tensor train) may grow quite rapidly with the accuracy. For some classes of tensors, for example solutions of the Fokker-Planck equation, it appears to be more efficient to sacrifice the linear structure of the format, and switch to a potentially higher cost $\mathcal{O}(r^3)$, if the new ranks r will be smaller.

So, we start from the Tucker decomposition

$$x(i_1, \dots, i_d) = \sum_{\gamma_1, \dots, \gamma_d} x^{(c)}(\gamma_1, \dots, \gamma_d) x^{(1)}_{\gamma_1}(i_1) \cdots x^{(d)}_{\gamma_d}(i_d), \tag{2.15}$$

removing the curse of dimensionality by storing the Tucker core in the TT format,

$$x^{(c)}(\gamma_1, \dots, \gamma_d) = \sum_{\alpha_1, \dots, \alpha_{d-1}} x^{c(1)}_{\alpha_1}(\gamma_1) x^{c(2)}_{\alpha_1, \alpha_2}(\gamma_2) \cdots x^{c(d)}_{\alpha_{d-1}}(\gamma_d). \tag{2.16}$$

Now, if the Tucker factors $x^{(k)}_{\gamma_k}(i_k)$ are still difficult to handle due to large sizes n_k, the quantization may be introduced for index i_k,

$$x^{(k)}_{\gamma_k}(i_k) = \sum_{\gamma_{k,L}, \dots, \gamma_{k,1}} x^{f(k,L)}_{\gamma_{k,L-1}}(i_{k,L}) x^{f(k,L-1)}_{\gamma_{k,L-1}, \gamma_{k,L-2}}(i_{k,L-1}) \cdots x^{f(k,1)}_{\gamma_{k,1}, \gamma_k}(i_{k,1}). \tag{2.17}$$

To reduce already enormous amount of subscripts, we assume that all $L_k = L$ are the same. Note that the QTT block $x^{f(k,1)}$ in the right-hand side contains *two* rank indices: while γ_k would correspond to $\alpha_d \in \{1\}$ in the traditional TT format (2.9), here it enumerates the vectors in Tucker factors, which are stored *simultaneously* in

the same QTT format. We could also put the Tucker rank index in the last QTT block $x^{f(k,L)}$, but it will be convenient to assign $\gamma_{k,0} = \gamma_k$.

Summarizing all the introduced forms, i.e. plugging the *inner* levels of structure (2.16), (2.17) into the *outer* Tucker (2.15), we obtain the final tree tensor network, called *QTT-Tucker* [44].

Definition 2.2.1. A tensor x is said to be presented (or approximated) in the QTT-Tucker format, if it holds

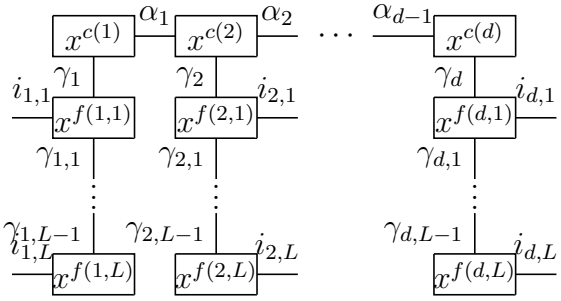
$$x(i_1, \dots, i_d) \approx \sum_{\gamma_k, \gamma_{k,l}, \alpha_k} \prod_{k=1}^{d} x^{c(k)}_{\alpha_{k-1}, \alpha_k}(\gamma_k) \prod_{l_k=1}^{L_k} x^{f(k,l_k)}_{\gamma_{k,l_k}, \gamma_{k,l_k-1}}(i_{k,l_k}), \tag{2.18}$$

where we agree that $\gamma_k = \gamma_{k,0}$, $n_k = 2^{L_k}$, and $i_k = \sum_{l_k=1}^{L_k} i_{l_k} \cdot 2^{l_k - 1}$. The parameter d is called the *physical, or core dimension*, L_k are the *quantics, or factor dimensions*, $x^{c(k)} \in \mathbb{C}^{r_{k-1} \times R_k \times r_k}$ is the *k-th core block*, $x^{f(k,l)} \in \mathbb{C}^{R_{k,l} \times n_{k,l} \times R_{k,l-1}}$ is the *k, l-th factor block*, and the indices vary in the ranges:

- $i_{k,l} = 0, \dots, n_{k,l} - 1, \quad n_{k,l} \leqslant n_Q$ (*quantized mode* size, e.g. $n_Q = 2$),
- $\alpha_k = 1, \dots, r_k, \quad r_k \leqslant r_C$ (*core* rank),
- $\gamma_k = \gamma_{k,0} = 1, \dots, R_k, \quad R_k = R_{k,0} \leqslant r_T$ (*Tucker* rank), and
- $\gamma_{k,l} = 1, \dots, R_{k,l}, \quad R_{k,l} \leqslant r_F$ (*factor* rank).

The boundary rank agreements are $r_0 = r_d = R_{k,L_k} = 1$.

In the diagrammatic notation, the QTT-Tucker format is shown in Equation (2.19).



(2.19)

Introducing a slight abuse of notations, we will denote the original Tucker blocks of the decomposition (2.5) as $x^{f(k)} = x^{f(k)}_{\gamma_k}(i_k)$, first, because it is consistent with the QTT factor block notation $x^{f(k,l)}$, and second, because the term $x^{(k)}$ is reserved for the TT blocks of the initial tensor (2.9).

Summing the sizes of all QTT-Tucker blocks, we find its storage cost.

Lemma 2.2.2. The storage complexity of the QTT-Tucker format is bounded by

$$dLn_Q r_F^2 + dr_T r_C^2, \quad \text{or} \quad \mathcal{O}(\log(n) dr^2 + dr^3), \tag{2.20}$$

assuming that all tensor ranks are bounded by the same constant r, and $n = n_Q^L$.

The $\mathcal{O}(r^3)$ contribution of the Tucker core to (2.20) may prevail in a high rank case, and if a tensor develops the same ranks for a given accuracy threshold as in the QTT format, we may loose the performance gain. For example, if we apply the HT format directly to the $2 \times \cdots \times 2$-reshaped tensor, it will contain $\mathcal{O}(d \log(n))$ blocks of sizes $r \times r \times r$, resulting in the total complexity of $\mathcal{O}(d \log(n) r + d \log(n) r^3)$ instead of $r \times 2 \times r$ in the straightforward *linear* QTT representation. This complexity overhead was illustrated in [164] on the spin system example, where the spin state variables were artificially agglomerated to make the HT decomposition useful.

The heuristic pros for the QTT-Tucker are the following. First, we have only d blocks with three rank indices, whereas $d \log(n)$ blocks possess the quadratic dependence like in the linear QTT. Moreover, the sizes of these d blocks are governed by TT and Tucker ranks, which are usually smaller than the ranks between virtual dimensions in the QTT decomposition. Second, even the QTT ranks of Tucker factors appear to be smaller for the same accuracy than the corresponding QTT ranks in the global linear format.

The latter phenomenon may be argued using the assumption that a tensor x comes as a discretized smooth function. The original TT block $x^{(k)}_{\alpha_{k-1},\alpha_k}(i_k)$ may be seen as a set of r_C^2 vectors of size n. These vectors are derived from univariate smooth functions, and we may expect a good QTT compressibility of each,

$$x^{(k)}_{\alpha_{k-1},\alpha_k}(i_k) = \sum_{\alpha_{k,1}\ldots,\alpha_{k,L-1}} x^{(k,1)}_{\alpha_{k-1},\alpha_{k,1}}(i_{k,1}) \cdots x^{(k,L)}_{\alpha_{k,L-1},\alpha_k}(i_{k,L})$$

with $\alpha_{k,l} = 1, \ldots, \tilde{r}$, and $\tilde{r}$ is moderate. If all vectors are "independent", the total QTT rank of x is bounded by $r_C^2 \tilde{r} = \mathcal{O}(r^2 \tilde{r})$. As a result, the storage of the QTT format estimates as $\mathcal{O}(d \log(n) r^4 \tilde{r}^2)$.

Applying the similar considerations to the Tucker factors, we may expect a lower bound for the factor ranks, $r_F = \mathcal{O}(r_T \tilde{r})$. If the Tucker ranks are similar to the TT ones, $r_F \sim r_C \sim r$ (which tends to happen in practice), the overall complexity of the QTT-Tucker format may be bounded by $\mathcal{O}(d \log(n) r^2 \tilde{r}^2 + d r^3)$, which is already smaller than in the QTT.

Another interesting feature of the new format is that it may inherit the rank estimates from the Tucker format, which may be easier to establish than in the TT format, using the polynomial interpolation. Here we extend and prove Theorem 2.1.8 for the QTT-Tucker format.

Theorem 2.2.3. Assume that an analytic function $f(q_1, \ldots, q_d)$ in $\Omega = [-1, 1]^d$ is given, and a tensor $x(i_1, \ldots, i_d) = f(q_1(i_1), \ldots, q_d(i_d))$ is its discretization on a tensor product uniform grid. Suppose f admits an extension to the ρ_k-Bernstein ellipse

$$\mathcal{E}_{\rho_k} = \left\{ z \in \mathbb{C} : \ |1 + z| + |1 - z| \leqslant \rho_k + \frac{1}{\rho_k} \right\}$$

in each variable q_k, such that $\mathcal{M} = \max\limits_{\boldsymbol{z} \in \mathcal{E}_{\rho_1} \otimes \cdots \otimes \mathcal{E}_{\rho_d}} |f(\boldsymbol{z})| < \infty$. Then the Tucker ranks of the ε-approximation are bounded by $R_k \leqslant C |\log(\varepsilon)| / \log(\rho_k)$, and the QTT ranks of the Tucker factors are bounded by $R_{k,l} = R_k + 1$, $l = 1, \ldots, L - 1$.

Proof. The function f may be considered as a univariate function $f^{[k]}(q_k)$, depending on the rest coordinates as parameters. So we can apply the following result from the approximation theory: if $f^{[k]}$ admits an analytical extension to the ρ_k-Bernstein ellipse, then there exists the polynomial interpolation P_p of degree p and the accuracy

$$\|f^{[k]}(z) - P_p(z)\|_\infty \leqslant C \log(n) \frac{M_k}{1-\rho_k} \rho_k^{-p}, \quad z \in \mathcal{E}_{\rho_k},\ \rho_k > 1, \quad M_k = \max_{z \in \mathcal{E}_{\rho_k}} |f^{[k]}(z)|,$$

where n is the number of grid points, and C does not depend on p, n, M_k, ρ_k [20, 225]. However, here the constant M_k depends on the other variables $q_1, \ldots, q_d$ excluding q_k. To get rid of them, let us assume the uniform bound $\mathcal{M} = \max_k M_k$. Now we may say that for each $f^{[k]}$ there exists a polynomial $P_{p_k}^{[k]}$ such that

$$\varepsilon = \|f^{[k]}(q_k) - P_{p_k}^{[k]}(q_k)\|_\infty \leqslant C \rho_k^{-p_k}, \quad \text{or} \quad p_k = C|\log(\varepsilon)|/\log(\rho_k).$$

Therefore, we may take a tensor product of degree p_k polynomials as the new basis. Obviously, the coefficients in this basis yield a $p_1 \times \cdots \times p_d$ tensor, which may be considered as a Tucker core, and a set of p_k polynomials on a grid is the k-th Tucker factor. The estimate for R_k is proven.

Recalling that all polynomials of degree p_k on a uniform grid are representable in the same QTT format with the QTT ranks $p_k + 1$ (see the previous section) we obtain the second claim of the Theorem. □

2.2.3 TT to Extended TT (QTT-Tucker) conversion

Having several tensor representations, it is worth to have a procedure to cast data from one to another. For example, the full tensor is constructed from the TT format as a sum of Kronecker products (2.10). Given a QTT-Tucker (or Extended TT) of a tensor, its TT blocks are computed via the summation over Tucker rank indices,

$$x^{(k)}_{\alpha_{k-1},\alpha_k}(i_k) = \sum_{\gamma_{k,L-1},\ldots,\gamma_{k,1},\gamma_k} x^{f(k,L)}_{\gamma_{k,L-1}}(i_{k,L}) \cdots x^{f(k,1)}_{\gamma_{k,1},\gamma_k}(i_{k,1}) x^{c(k)}_{\alpha_{k-1},\alpha_k}(\gamma_k). \tag{2.21}$$

Obviously having a TT form in the right-hand side, this expression was called an *extended factor* in [44]. In the same time, it is in turn a block of another tensor train. This shows the two-level structure from another point of view.

The inverse operation may require approximate computations. Consider the *center-folded* (see Def. 2.1.12) k-th TT block $x^{|k|} = \left[x^{|k|}(i_k, \overline{\alpha_{k-1}, \alpha_k})\right]$. Assume the TT blocks $1, \ldots, k-1$ are left-orthogonal, and $k+1, \ldots, d$ are right-orthogonal, so that the perturbation introduced to $x^{|k|}$ yields the same error level for x. Now, we may perform the (truncated) SVD decomposition

$$x^{|k|} \approx U \Sigma V,$$

and associate either $x^{f(k)} = U$, $x^{c|k|} = \Sigma V$ or $x^{f(k)} = U\Sigma$, $x^{c|k|} = V$, depending on what type of orthogonality we would like to ensure. Surely, the k-th Tucker rank is revealed exactly as the rank of this decomposition. The new rank bound may be

formulated immediately as $R_k \leqslant \min(n_k, r_{k-1}r_k) = \mathcal{O}(\min(n, r^2))$ (cf. the similar TT vs HT comparison in [92]), as well as the complexity $\mathcal{O}(nr^2 \min(n, r^2))$.

In an exact case, such decomposition may be performed analytically. Let us demonstrate it on a sum of univariate objects (e.g. Laplace operator, sum of variables $q_1 + \cdots + q_d$, etc.),

$$x(i_1, \ldots, i_d) = a(i_1) \cdot b(i_2) \cdots b(i_d) + \cdots + b(i_1) \cdots b(i_{d-1}) \cdot a(i_d).$$

The *exact* rank-2 TT representation [131] reads

$$x(\boldsymbol{i}) = \begin{bmatrix} a(i_1) & b(i_1) \end{bmatrix} \begin{bmatrix} b(i_2) & 0 \\ a(i_2) & b(i_2) \end{bmatrix} \cdots \begin{bmatrix} b(i_{d-1}) & 0 \\ a(i_{d-1}) & b(i_{d-1}) \end{bmatrix} \begin{bmatrix} b(i_d) \\ a(i_d) \end{bmatrix}.$$

Since each block contains only 2 linearly independent elements, all Tucker factors write

$$x^{f(k)}(i_k) = \begin{bmatrix} a(i_k) & b(i_k) \end{bmatrix},$$

and the core ($2 \times \cdots \times 2$ tensor) reads

$$x^{(c)}(\boldsymbol{\gamma}) = \begin{bmatrix} e_0(\gamma_1) & e_1(\gamma_1) \end{bmatrix} \begin{bmatrix} e_1(\gamma_2) & 0 \\ e_0(\gamma_2) & e_1(\gamma_2) \end{bmatrix} \cdots \begin{bmatrix} e_1(\gamma_{d-1}) & 0 \\ e_0(\gamma_{d-1}) & e_1(\gamma_{d-1}) \end{bmatrix} \begin{bmatrix} e_1(\gamma_d) \\ e_0(\gamma_d) \end{bmatrix},$$

where $e_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}$, and $e_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}$.

Remark 2.2.4. Other tensor networks can be converted first to the TT format (or a sum of TT tensors, for example, the Tensor Chain), and then into the QTT-Tucker. It could be helpful in quantum physical problems, if a model provides initial data in a complicated tensor network.

2.2.4 QTT-Tucker arithmetics

The tensor format counterparts of algebraic operations over initial tensor elements, described in Section 2.1.5, may be similarly extended to the QTT-Tucker.

Thus, the **sum** of QTT-Tucker tensors $z = x + y$ is performed as a concatenation of factors

$$z^{f(k,L)} = \begin{bmatrix} x^{f(k,L)}(i_{k,L}) & y^{(k,L)}(i_{k,L}) \end{bmatrix}, \quad z^{f(k,l)} = \begin{bmatrix} x^{f(k,l)}(i_{k,l}) & \\ & y^{(k,l)}(i_{k,l}) \end{bmatrix}$$

for $l = L-1, \ldots, 1$, and core blocks

$$z^{c(k)}(\gamma_k) = \begin{cases} \begin{bmatrix} x^{c(k)}(\gamma_k) & 0 \\ 0 & 0 \end{bmatrix}, & \text{if} \quad \gamma_k = 1, \ldots, R_k(x), \\ \begin{bmatrix} 0 & 0 \\ 0 & y^{c(k)}(\gamma_k - R_k(x)) \end{bmatrix}, & \text{if} \quad \gamma_k = R_k(x)+1, \ldots, R_k(x) + R_k(y). \end{cases}$$

The **matrix** product $y = Ax$ casts to the corresponding product of factors, but with the Tucker rank indices $\gamma_k(A)$, $\gamma_k(x)$ joined,

$$y^{f(k,l)}(i_{k,l}) = \sum_{j_{k,l}} A^{f(k,p)}(i_{k,l}, j_{k,l}) \otimes x^{f(k,l)}(j_{k,l}),$$

where the standard Kronecker products $\otimes$ are taken w.r.t. the rank indices (recall that for fixed $i_{k,l}, j_{k,l}$, the blocks become just matrices), and the 3-dimensional Kronecker product applied to the core blocks,

$$y^{c(k)}(\overline{\gamma_k(A), \gamma_k(x)}) = A^{c(k)}(\gamma_k(A)) \otimes x^{c(k)}(\gamma_k(x)).$$

All ranks are multiplied: $r(y) = r(A)r(x)$. Note that the matrix QTT-Tucker format contains two initial indices in the factor blocks only, but the core blocks are also 3-dimensional, like in the vector representation.

The **scalar** product begins with the TT scalar product Algorithm 1 applied to factors. The difference is that we start not from the first, but from the L-th factor block, and obtain as a result not a number but a matrix $s_k \in \mathbb{C}^{R_k(x) \times R_k(y)}$, $s_k(\gamma_k(x), \gamma_k(y)) = \left(x^{f(k)}_{\gamma_k(x)}, y^{f(k)}_{\gamma_k(y)}\right)$. When all factors are processed, the matrices s_k, $k = 1, \ldots, d$, are multiplied with either of the Tucker cores to obtain a new intermediate tensor z of the same sizes as another core. Without loss of generality, let it be $y^{(c)}$, then

$$z = \tau(z^{(1)}, \ldots, z^{(d)}), \qquad z^{|k|} = s_k y^{c|k|}, \quad k = 1, \ldots, d,$$

and the result is delivered by one additional TT scalar product, $(x, y) = \left(x^{(c)}, z\right)$. One may verify straightforwardly that it will be exactly a product of elements of all cores, summed over all indices. The complexity of such algorithm is $\mathcal{O}(d \log(n) r_F^3) + \mathcal{O}(d r_F^2 r_C^2) + \mathcal{O}(d r_F r_C^3)$, where the first term comes from the product of factors, the second is the construction of z, and the last is the product of cores.

2.2.5 QTT-Tucker rounding

Though the derivation of the QTT-Tucker was based on the properties of the Tucker decomposition, it is more convenient to present alternating algorithms (the rank truncation procedure also belongs to this family) in terms of a two-level TT decomposition: the original TT (2.9) with each block being presented in the form of the extended factor (2.21). This allows to reuse the TT versions the algorithms.

As soon as all TT blocks except the k-th are properly orthogonal, we may approximate $x^{(k)}$ by invoking the TT approximation procedure for its extended factor form

$$\sum_{\gamma_{k,L-1}, \ldots, \gamma_{k,1}, \gamma_k} x^{f(k,L)}_{\gamma_{k,L-1}}(i_{k,L}) \cdots x^{f(k,1)}_{\gamma_{k,1},\gamma_k}(i_{k,1}) x^{c|k|}_{\gamma_k}(\overline{\alpha_{k-1}, \alpha_k}).$$

Note that the last term here is a center-folded core block $x^{c(k)}$.

In turn, the QR decomposition of, say, $x^{|k\rangle}$ may be computed efficiently, by applying the left-orthogonalization Algorithm 2 to the extended factor, followed by a small-sized factorization of the core block

$$q^{c|k\rangle} R = x^{c|k\rangle} \in \mathbb{C}^{r_{k-1} R_k \times r_k}.$$

The whole method is summarized in Algorithm 5.

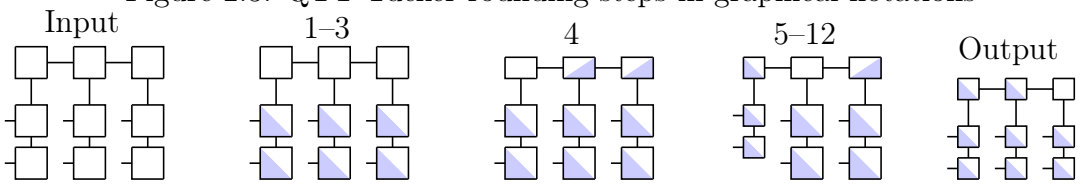
The complexity of Algorithm 5 comes directly from the complexities of the TT routines. Namely, the orthogonalization/truncation requires $\mathcal{O}(dLr_F^3)$ operations for factors and $\mathcal{O}(dr_C^4)$ for the core, resulting in the total cost $\mathcal{O}(dLr_F^3 + dr_C^4)$.

Algorithm 5 QTT-Tucker round

Require: QTT-Tucker tensor x, accuracy bounds ε_k, η.
Ensure: QTT-Tucker tensor y with smaller ranks s.t. $\|y - x\|^2 \leqslant \sum \varepsilon_k^2 + \eta^2$.

1: **for** $k = 1, \ldots, d$ **do**
2: Left-orthogonalize k-th extended factor via Alg. 2, such that $\left(x^{f|k\rangle}\right)^* x^{f|k\rangle} = I$.
3: **end for**
4: Approximate the core $y^{(c)} : \|y^{(c)} - x^{(c)}\| \leqslant \eta \|x^{(c)}\|$ via Alg. 4.
5: **for** $k = 1, \ldots, d$ **do**
6: Approximate the extended factor $y^{(k)} : \|y^{(k)} - x^{(k)}\| \leqslant \varepsilon_k \|x^{(k)}\|$ via Alg. 4.
7: **if** $k < d$ **then**
8: Ensure the left-orthogonality of $y^{f|k\rangle}$ (Alg. 2).
9: Find QR decomposition $y^{c|k\rangle} = q^{c|k\rangle} R$.
10: Replace $y^{c|k\rangle} = q^{c|k\rangle}, \quad y^{c\langle k+1|} = R y^{c\langle k+1|}$.
11: **end if**
12: **end for**

Figure 2.3: QTT-Tucker rounding steps in graphical notations



It is also interesting, that the orthogonalization/rounding of the extended factors (lines 2 and 6 of the algorithm) mimic in fact the Tucker HOSVD: the difference is in using only one *block of the core* instead of the whole d-dimensional core. In diagrammatic notations, the states of the QTT-Tucker format after certain lines of Algorithm 5 are shown in Fig. 2.3: the filled parts of the blocks denote the orthogonality, and the sizes of the blocks are proportional to their ranks.

To finish with rounding, we present the quasioptimality result for Algorithm 5.

Theorem 2.2.5. Suppose that each truncation in QTT-Tucker factors is done via the SVD with the relative Frobenius-norm thresholds $\varepsilon_{k,l}$, the Tucker ranks are determined with ε_k, and each core block is truncated with η_k. Then the relative Frobenius norm of the total error is bounded by

$$\frac{\|y - x\|^2}{\|x\|^2} \leqslant \sum_{k,l=1}^{d,L-1} \varepsilon_{k,l}^2 + \sum_{k=1}^{d} \varepsilon_k^2 + \sum_{k=1}^{d-1} \eta_k^2. \tag{2.22}$$

We omit the proof, since it resembles the proof of the HOSVD theorem 2.1.7 to a large extent: we associate each SVD with the corresponding orthogonal projector in terms of initial vectors, and then using the add-and-subtract trick, and orthogonality of any vectors of form $x = Pz$ and $y = z - Pz$, derive the result.

Chapter 3

Tensor structure properties of some classes of operators and functions

During the exploitation of various tensor formats, it is helpful to construct some simple and widely used objects explicitly, by specifying the format elements directly. For the TT and QTT format this work was started in [148, 192, 131, 44], and for the canonical format in [72, 100, 101]. It is especially convenient if a tensor has an exact decomposition of a small rank – performing its compression from the full, or even canonical formats may be quite difficult computationally, and corrupt the accuracy. The main tool for derivation of exact tensor structures is the recurrent analytical counterpart of the TT rounding algorithm 4. In each step, instead of computing SVD, we extract intuitively simple linearly independent elements to form the current TT block. If the resting tensors (cf. $X^{\langle k|}(\overline{i_1,\dots,i_{k-1}},\overline{i_k,\alpha_k})$ in the end of Section 2.1.5) have the same analytical form for any k, it means that the recurrence takes place, and all inner TT blocks are constructed by the same rule.

Another way, which is in many cases the only technique for deduction of *approximate* tensor structures, is the assembly of a complex tensor from simpler ones via tensor arithmetics. Mostly used is the canonical format, since usually the approximation theory provides results in terms of convergent series. If each member of the series has a simple (e.g. rank 1) tensor representation, we may take a finite amount of them, obtaining a canonical or TT format (see Remark 2.1.13) with controlled rank-vs-accuracy estimates.

3.1 Separabilities of gradients and the block time scheme

One of the principal contributions of this work is the simultaneous space-time Crank-Nicolson scheme (1.17) or (1.20). To solve it efficiently in tensor formats, we need three ingredients: tensor representativity of the inputs (i.e. the matrix, right-hand side and initial state), tensor representativity of the solution, and an algorithm to actually compute the solution, provided all data are well structured in principle.

The second part is quite hard for theoretical analysis, and one often has to try it numerically. A robust approach for the third component will be presented in the next chapter. Now we focus on the first part: we investigate the contribution of the time

dimension itself, assuming that other inputs are given in a tensor format, and then show several examples when this assumption does take place.

3.1.1 Tensor structure of the space-time matrix

Considering first the unpreconditioned variant (1.17), assume that the right-hand side g is zero, the matrix A is written in the TT format (2.11), and the initial state v is given in the TT format (2.9). The global matrix is then easily constructed as a $d+1$-dimensional TT format,

$$\mathcal{A} = \sum_{\boldsymbol{\alpha}} \mathcal{A}^{(1)}_{\alpha_1} \otimes \mathcal{A}^{(2)}_{\alpha_1,\alpha_2} \otimes \cdots \otimes \mathcal{A}^{(d)}_{\alpha_{d-1},\alpha_d} \otimes \mathcal{A}^{(d+1)}_{\alpha_d},$$

$$\mathcal{A}^{(k)}_{\alpha_{k-1},\alpha_k} = \begin{cases} A^{(k)}_{\alpha_{k-1},\alpha_k}, & \alpha_{k-1} = 1,\ldots,r_{k-1}(A), \\ & \alpha_k = 1,\ldots,r_k(A), \\ I, & \alpha_{k-1} = r_{k-1}(A)+1, \text{ and} \\ & \alpha_k = r_k(A)+1, \\ 0, & \alpha_{k-1} = r_{k-1}(A)+1,\ \alpha_k \leqslant r_k(A), \text{ or} \\ & \alpha_k = r_k(A)+1,\ \alpha_{k-1} \leqslant r_{k-1}(A), \end{cases}$$

for $k = 1,\ldots,d$, and

$$\mathcal{A}^{(d+1)}_1 = \frac{\delta t}{2} M_t, \quad \mathcal{A}^{(d+1)}_2 = G_t.$$

One notices immediately that the TT ranks $r_1,\ldots,r_{d-1}$ are increased by one compared to $\boldsymbol{r}(A)$, and $r_d = 2$. Therefore, contrarily to the full format version (1.16), the tensor product storage of the global matrix is not much larger than in each Crank-Nicolson step.

The initial state is assembled similarly: given a d-dimensional TT representation for $v - \frac{\delta t}{2} Av$, it is concatenated with the $d+1$-t TT block being the temporal unit vector e_1, without changing the ranks.

The TT format for the preconditioned scheme (1.20) writes in the same way with $\tilde{\mathcal{A}}^{(k)} = \mathcal{A}^{(k)}$ for $k = 1,\ldots,d$, and only the last block changes,

$$\tilde{\mathcal{A}}^{(d+1)}_1 = \frac{\delta t}{2} G_t^{-1} M_t, \quad \tilde{\mathcal{A}}^{(d+1)}_2 = I.$$

Similarly, e_1 in the right-hand side is replaced by a vector of all ones.

3.1.2 Shift and gradient matrices in the QTT format

The number of time steps required to resolve multi scale processes may be quite large, and the use of the Quantized TT w.r.t. the time variable is very reasonable [47, 46].

The identity matrix I, the all-ones vector e and the unit vector e_1 (see Section 2.2.1) have rank-1 QTT representations for any size. More interesting is the structure of the gradient and mass matrices G_t, M_t. As was shown in [127, 131], a shift matrix

of any order and size possesses a rank-2 QTT representation. In particular, the Jordan matrix $\boldsymbol{J} \in \mathbb{R}^{2^L \times 2^L}$ has the following form[1]:

$$\boldsymbol{J}(\boldsymbol{i}, \boldsymbol{j}) = \begin{bmatrix} J_{i_1,j_1} & J^\top_{i_1,j_1} \end{bmatrix} \cdots \begin{bmatrix} I_{i_k,j_k} & \\ J_{i_k,j_k} & J^\top_{i_k,j_k} \end{bmatrix} \cdots \begin{bmatrix} I_{i_L,j_L} \\ J_{i_L,j_L} \end{bmatrix},$$

where $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $J = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ are the elementary identity and Jordan matrices, respectively. Note that the inner blocks are constructed in the same way for all $k = 2, \ldots, L-1$.

As a simple illustration, let us see how the TT rounding algorithm is conducted analytically. The temporal matrices write $G_t = I - \boldsymbol{J}^\top$, $M_t = I + \boldsymbol{J}^\top$, i.e. as rank-3 structures at the first glance. However, consider the linear dependence elimination in the last blocks,

$$\begin{bmatrix} I_{i_L,j_L} \\ J^\top_{i_L,j_L} \\ I_{i_L,j_L} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} I_{i_L,j_L} \\ J^\top_{i_L,j_L} \end{bmatrix},$$

and for $k = L-1$,

$$\begin{bmatrix} I_{i_k,j_k} & & \\ J^\top_{i_k,j_k} & J_{i_k,j_k} & \\ & & I_{i_k,j_k} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} I_{i_k,j_k} & \\ J^\top_{i_k,j_k} & J_{i_k,j_k} \\ I_{i_k,j_k} & \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} I_{i_k,j_k} & \\ J^\top_{i_k,j_k} & J_{i_k,j_k} \end{bmatrix}.$$

Since the scalar factor is thrown to the left-hand side, the recursion can be continued for all k, and finished in the first block,

$$\begin{bmatrix} J^\top_{i_1,j_1} & J_{i_1,j_1} & I_{i_1,k_1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} M_{i_1,j_1} & J_{i_1,j_1} \end{bmatrix},$$

where $M = I + J^\top$ is an elementary mass matrix. That is, both G_t and M_t are representable *simultaneously* in the QTT format with the blocks $k = 2, \ldots, L$ all the same, belonging to the transposed Jordan matrix, and only the first QTT block distinguishes the initial matrices. We may write them in the form used in the QTT-Tucker Section 2.2.2: the zeroth TT rank stands for the enumerator, i.e.

$$\begin{bmatrix} G_t(\boldsymbol{i}, \boldsymbol{j}) \\ M_t(\boldsymbol{i}, \boldsymbol{j}) \end{bmatrix} = \begin{bmatrix} G_{i_1,j_1} & J_{i_1,j_1} \\ M_{i_1,j_1} & J_{i_1,j_1} \end{bmatrix} \cdots \begin{bmatrix} I_{i_k,j_k} & \\ J^\top_{i_k,j_k} & J_{i_k,j_k} \end{bmatrix} \cdots \begin{bmatrix} I_{i_L,j_L} \\ J^\top_{i_L,j_L} \end{bmatrix},$$

where $G = I - J^\top$ is an elementary gradient.

The preconditioned scheme (1.20) can be analyzed similarly. The matrix G_t^{-1} is a Toeplitz generated by a QTT rank-1 vector $\begin{bmatrix} 0 & \cdots & 0 & 1 & \cdots & 1 \end{bmatrix}$, and hence has a rank-2 QTT structure [127]. Therefore, the matrix $G_t^{-1} M_t$ is of QTT rank 4.

[1]Throughout the work, we use the little-endian convention, e.g. indices i_1, j_1 are the fastest, and i_L, j_L are the slowest varying ones, cf. (2.14). This is different to [131, 127], where the big-endian convention is used. So, the QTT representation writes in the "reversed" order compared to [127].

3.2 Tensor properties of the Fokker-Planck and chemical master equations

In this section we present general and refined analysis of the tensor structures of the two main problems in this work, the Fokker-Planck and chemical master equations. We emphasize again that analytical solutions, as well as the tensor format properties of the solutions can be rarely derived rigorously, most of all such results require rather narrow class of inputs. Fortunately, the numerical evidences say in favor of separation of variables in much wider settings, where theoretical estimates are not provided. Besides, many relevant cases have nevertheless rather simple operator or initial state, which are not so difficult to analyze for the separability. This will be our task for this chapter of the manuscript.

As a naive, but most general way, assume that the *coefficients* in the operators admit a tensor representation. What can we say about the structure of the operator itself? First, consider the Fokker-Planck equation,

$$\frac{\partial \psi}{\partial t} = -\sum_{k=1}^{d} \frac{\partial}{\partial q_k} F_k \psi + \sum_{k,m=1}^{d} \frac{\partial}{\partial q_k} B_{k,m} \frac{\partial}{\partial q_m} \psi.$$

In the right-hand side, each gradient term acts only in one variable,

$$\frac{\partial}{\partial q_k} = I \otimes \cdots \otimes I \otimes \nabla_k \otimes I \otimes \cdots \otimes I,$$

i.e. is a rank-1 tensor after the discretization. Assuming the uniform rank bounds for the coefficients, $r(F_k) \leqslant r_F$ and $r(B_{k,m}) \leqslant r_B$, obtain the total estimate for the Fokker-Planck operator by the straightforward summation,

$$r \leqslant d r_F + d^2 r_B.$$

It is natural to expect this estimate to be far not sharp. In particular, in the next section we will show that if the diffusion matrix B is scalar (i.e. independent on $\boldsymbol{q}$), the rank of the diffusion operator can not grow faster than linearly in d. Similar refinements may be introduced to the convection part, if more information on the forces F_k is known.

In an analogous way, for the chemical master equation (1.11)

$$\frac{d\psi}{dt} = \sum_{m=1}^{M} \left(\boldsymbol{J}^{\boldsymbol{z}^m} - I \right) \operatorname{diag}(w_m) \psi,$$

where $\boldsymbol{J}^{\boldsymbol{z}^m} = J^{z_1^m} \otimes J^{z_2^m} \otimes \cdots \otimes J^{z_d^m}$ is a multilevel shift matrix of the order $(z_1^m, \ldots, z_d^m)$, we conclude that the TT ranks of the operator in the right-hand side are bounded by

$$r \leqslant \sum_{m=1}^{M} 2 r(w_m) \leqslant 2 M r_w, \quad \text{if } r(w_m) \leqslant r_w.$$

The factor 2 arises from the fact that both $\boldsymbol{J}^{\boldsymbol{z}^m}$ and I have rank-1 representations.

3.2.1 Bilinear form in the TT format

One of the interesting nontrivial tensor structures is the bilinear form:

$$V(Q', Q) = \sum_{k,m=1}^{d} B_{k,m}\, Q'_k \circledast Q_m, \quad \text{where} \tag{3.1}$$

Q'_k, Q_m are the following rank-1 tensor product objects:

$$\begin{aligned} &Q'_k = e_1 \otimes e_2 \otimes \cdots \otimes q'_k \otimes \cdots \otimes e_d, \\ &Q_k = e_1 \otimes e_2 \otimes \cdots \otimes q_k \otimes \cdots \otimes e_d, \\ &e_k, q_k, q'_k \in V \sim \mathbb{R}^n, \text{ a certain euclidean vector space,} \end{aligned}$$

$\circledast : V \times V \to V$ is a multiplicative bilinear operation, distributive with "$\otimes$" (e.g. Hadamard or matrix product), e_k are the unities with respect to $\circledast$, and B is a matrix of scalars (of course we assume that a product of Q, Q' by a number is also defined). Moreover, we denote $q'_k \circledast q_k \equiv q_k^2$ for brevity.

The application of the bilinear form in the Fokker-Planck equation is twofold. First, the quadratic *Lyapunov* function is a principal ingredient in the Fokker-Planck equation with a quadratic force potential, where the stationary probability density function is given by $\psi = \exp(-V)$, recall (1.8). Here, e_k are the vectors of all ones, $q_k = q'_k$ are the vectors of grid points, and $\circledast$ is the Hadamard product, so V is indeed a multivariate polynomial of second degree. For a general polynomial, the tensor structure was investigated in [155], and for degree 2 the general theorem gives the rank bound $d + 2$. Focusing on the bilinear form, we obtain a refined result.

Second, a d-dimensional anisotropic diffusion operator $\sum\limits_{k,m} \nabla_k^\top B_{k,m} \nabla_m$ with a constant matrix B is also seen as a bilinear form V, by setting $e_k = I_k$, $q_k = \nabla_k$, $q'_k = \nabla_k^\top$, $q_k^2 = \Delta_k$, and $\circledast$ being the operator composition.

For the ease of presentation, let us assume that $V = [V(\boldsymbol{i})]$, as well as $q_k = [q_k(i_k)]$ and so on, i.e. we join row and column indices if V is an operator. Now, perform the analytic TT rounding and see what does the TT structure of (3.1) depend on.

Theorem 3.2.1 ([44])**.** Introduce the off-diagonal lower $C^k = B_{k+1:d,\ 1:k}$, and upper $\hat{C}^k = B_{1:k,\ k+1:d}$ submatrices. The bilinear form (3.1) possesses an exact TT decomposition $V(\boldsymbol{i}) = V^{(1)}(i_1) \cdots V^{(d)}(i_d)$ with the ranks:

$$r_k = \operatorname{rank}(C^k) + \operatorname{rank}(\hat{C}^k) + 2 \leqslant d + 2. \tag{3.2}$$

Moreover, if $q_k = q'_k$ (symmetric case), the ranks reduce to

$$r_k = \operatorname{rank}\Big((C^{k\top} + \hat{C}^k)/2\Big) + 2 \leqslant d/2 + 2.$$

In the QTT-Tucker format, the estimates above correspond to the TT-ranks of the core. The Tucker ranks R_k equal 4 in the general case, and 3 in the symmetric case. The QTT ranks of the Tucker factors depend on e_k, q_k, q'_k, q_k^2:

$$R_k \leqslant r(e_k) + r(q_k) + r(q'_k) + r(q_k^2).$$

The QTT rank bound in the linear tree is $r(e_k) + \operatorname{rank}(C^k) r(q_k) + \operatorname{rank}(\hat{C}^k) r(q'_k) + r(q_k^2)$.

Proof. Denote the off-diagonal dyadic decompositions in B,

$$B_{k+1:d,\ 1:k} = C^k = W^k U^k, \qquad W^k \in \mathbb{R}^{d-k \times r_k}, \quad U^k \in \mathbb{R}^{r_k \times k},$$
$$B_{1:k,\ k+1:d} = \hat{C}^k = (\hat{U}^k)^\top (\hat{W}^k)^\top, \quad \hat{W}^k \in \mathbb{R}^{d-k \times \hat{r}_k}, \quad \hat{U}^k \in \mathbb{R}^{\hat{r}_k \times k},$$

with orthonormal U^k, $\hat{U}^k$.

The rank structure is rather complicated, so we omit the initial indices $\boldsymbol{i}, i_1, \ldots, i_d$ in the rest of the proof, i.e. agree that $V = V(\boldsymbol{i})$, $q_k = q_k(i_k)$ and so on. No ambiguity arises, since we do not investigate the matter with respect to $\boldsymbol{i}$ or i_k.

In the first step of the recursion we have

$$V = B_{1,1} Q'_1 \circledast Q_1 + \sum_{p=2}^{d} B_{p,1} Q'_p \circledast Q_1 + \sum_{m=2}^{d} B_{1,m} Q'_1 \circledast Q_m + V^{[2]},$$

where $V^{[k]} = \sum\limits_{p,m=k}^{d} B_{p,m}\ Q'_p \circledast Q_m$. So,

$$V = \begin{bmatrix} q_1^2 B_{1,1} & q'_1 & q_1 & e_1 \end{bmatrix} \begin{bmatrix} E^{[2]} & \sum\limits_{m=2}^{d} Q_m^{[2]} B_{1,m} & \sum\limits_{p=2}^{d} Q_p'^{[2]} B_{p,1} & V^{[2]} \end{bmatrix}^\top, \tag{3.3}$$

where

$$\begin{aligned} E^{[k]} &= e_k \cdots e_d, \\ Q_m^{[k]} &= e_k \cdots e_{m-1} \cdot q_m \cdot e_{m+1} \cdots e_d, \\ Q_p'^{[k]} &= e_k \cdots e_{p-1} \cdot q'_p \cdot e_{p+1} \cdots e_d, \quad p, m \geqslant k. \end{aligned}$$

The first term in (3.3) is the first TT block $V^{(1)}$. On the other hand, we can represent it using the dyadic factors of B:

$$V = \begin{bmatrix} q_1^2 B_{1,1} & q'_1 \hat{U}^1 & q_1 U^1 & e_1 \end{bmatrix} \begin{bmatrix} E^{[2]} & \sum\limits_{m=2}^{d} Q_m^{[2]} \hat{W}^1_{m-1,1} & \sum\limits_{p=2}^{d} Q_p'^{[2]} W^1_{p-1,1} & V^{[2]} \end{bmatrix}^\top.$$

Now, we need to derive the recursive representation for the second term.

Suppose we have

$$\tilde{V} = \begin{bmatrix} E^{[k]} & \sum\limits_{m=k}^{d} Q_m^{[k]} \hat{W}^{k-1}_{m-k+1,:} & \sum\limits_{p=k}^{d} Q_p'^{[k]} W^{k-1}_{p-k+1,:} & V^{[k]} \end{bmatrix}^\top, \tag{3.4}$$

where ":" denotes the whole range of the indices. Splitting the first dimension from $V^{[k]}$, obtain (I_s is the identity matrix of size s):

$$\tilde{V} = \begin{bmatrix} e_k & & & \\ q_k (\hat{W}^{k-1}_{1,:})^\top & e_k I_{\hat{r}_{k-1}} & & \\ q'_k (W^{k-1}_{1,:})^\top & & e_k I_{r_{k-1}} & \\ q_k^2 B_{k,k} & & & q'_k \quad q_k \quad e_k \end{bmatrix} \begin{bmatrix} E^{[k+1]} \\ \sum\limits_{m=k+1}^{d} Q_m^{[k+1]} (\hat{W}^{k-1}_{m-k+1,:})^\top \\ \sum\limits_{p=k+1}^{d} Q_p'^{[k+1]} (W^{k-1}_{p-k+1,:})^\top \\ \sum\limits_{m=k+1}^{d} Q_m^{[k+1]} B_{k,m} \\ \sum\limits_{p=k+1}^{d} Q_p'^{[k+1]} B_{p,k} \\ V^{[k+1]} \end{bmatrix}.$$

Note that in the terms like $(W^{k-1}_{1,:})^\top$, we *first* take the slice, and then transpose it, so the column $r \times 1$ and row $1 \times r$ vectors are correctly distinguished. Separating the scalar coefficients from Q-related data in the last term, obtain

$$\begin{bmatrix} E^{[k+1]} \\ \sum_{m=k+1}^{d} Q_m^{[k+1]} (\hat{W}^{k-1}_{m-k+1,:})^\top \\ \sum_{p=k+1}^{d} Q_p'^{[k+1]} (W^{k-1}_{p-k+1,:})^\top \\ \sum_{m=k+1}^{d} Q_m^{[k+1]} B_{k,m} \\ \sum_{p=k+1}^{d} Q_p'^{[k+1]} B_{p,k} \\ V^{[k+1]} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & (\hat{W}^{k-1}_{2:d-k+1,:})^\top & & \\ & & (W^{k-1}_{2:d-k+1,:})^\top & \\ & B_{k,k+1:d} & & \\ & & (B_{k+1:d,k})^\top & \\ & & & 1 \end{bmatrix} \begin{bmatrix} E^{[k+1]} \\ Q^{[k+1]}_{k+1:d} \\ Q'^{[k+1]}_{k+1:d} \\ V^{[k+1]} \end{bmatrix}. \quad (3.5)$$

The term $Q^{[k+1]}_{k+1:d}$ denotes $Q^{[k+1]}_{k+1}, \ldots, Q^{[k+1]}_{d}$ stacked columnwise, the same is for $Q'^{[k+1]}_{k+1:d}$. Now, recall that

$$(W^{k-1}_{2:d-k+1,:})^\top = U^{k-1}(B_{k+1:d,1:k-1})^\top = U^{k-1}(U^k_{:,1:k-1})^\top (W^k_{1:d-k,:})^\top.$$

Similar equation holds for $\hat{W}^{k-1}$. That is, (3.5) splits to the scalar part, and a counterpart of (3.4) with k substituted by $k+1$. Therefore,

$$\tilde{V} = V^{(k)} \begin{bmatrix} E^{[k+1]} \\ \sum_{m=k+1}^{d} Q_m^{[k+1]} (\hat{W}^{k}_{m-k,:})^\top \\ \sum_{p=k+1}^{d} Q_p'^{[k+1]} (W^{k}_{p-k,:})^\top \\ V^{[k+1]} \end{bmatrix},$$

where

$$V^{(k)} = \begin{bmatrix} e_k & & & \\ q_k(\hat{W}^{k-1}_{1,:})^\top & e_k\ \hat{U}^{k-1}(\hat{U}^k_{:,1:k-1})^\top & & \\ q'_k(W^{k-1}_{1,:})^\top & & e_k\ U^{k-1}(U^k_{:,1:k-1})^\top & \\ q_k^2 B_{k,k} & q'_k(\hat{U}^k_{:,k})^\top & q_k(U^k_{:,k})^\top & e_k \end{bmatrix} \quad (3.6)$$

is nothing else than the k-th TT block of V, since it contains only q_k, e_k. It has the sizes $(1 + \hat{r}_{k-1} + r_{k-1} + 1) \times (1 + \hat{r}_k + r_k + 1)$, which confirms the first statement of the theorem. Assuming the QTT separability of basis elements e_k, q_k and so on, obtain the linear QTT rank bound. With the rest $\tilde{V}$ we can proceed recursively, and the last TT block reads (since $V^{[d]}$ is just a one-dimensional $q_d^2 B_{d,d}$)

$$V^{(d)} = \begin{bmatrix} e_d & q_d\ \hat{W}^{d-1}_{1,:} & q'_d\ W^{d-1}_{1,:} & q_d^2 B_{d,d} \end{bmatrix}^\top. \quad (3.7)$$

If the first-order term is presented with a unique object $q_k = q'_k$, the ranks are reduced as follows. First,

$$V^{(1)} = \begin{bmatrix} q_1^2 B_{1,1} & 2q_1 & e_1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 0.5 & 0.5 \\ & & 1 \end{bmatrix}.$$

Reassigning $\bar{V}^{(1)} = \begin{bmatrix} q_1^2 B_{1,1} & 2q_1 & e_1 \end{bmatrix}$, we contract the rest matrix with the middle blocks, obtaining

$$\tilde{V}^{(k)} = \begin{bmatrix} e_k & & \\ q_k (W^{k-1}_{1,:})^\top & \frac{1}{2} e_k \; U^{k-1} (U^k_{:,1:k-1})^\top & \frac{1}{2} e_k \; U^{k-1} (U^k_{:,1:k-1})^\top \\ q_k^2 B_{k,k} & q_k (U^k_{:,k})^\top & q_k (U^k_{:,k})^\top & e_k \end{bmatrix}.$$

We note here that the constraint $q_k = q'_k$ yields $V \equiv 0$ if $B = -B^\top$, so that only the symmetric part $B := 0.5(B + B^\top)$ is relevant. Thus, $\hat{W}^k = W^k$, $\hat{U}^k = U^k$. All versions of $\tilde{V}$ are simplified as follows,

$$\begin{bmatrix} E^{[k+1]} \\ \sum_{m=k+1}^{d} Q_m^{[k+1]} (\hat{W}^k_{m-k,:})^\top \\ \sum_{p=k+1}^{d} Q_p'^{[k+1]} (W^k_{p-k,:})^\top \\ V^{[k+1]} \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} E^{[k+1]} \\ \sum_{m=k+1}^{d} Q_m^{[k+1]} (W^k_{m-k,:})^\top \\ V^{[k+1]} \end{bmatrix}.$$

Multiplying $\tilde{V}^{(k)}$ with the scalar matrix appeared here, we obtain the reduced block

$$\bar{V}^{(k)} = \begin{bmatrix} e_k & & \\ q_k (W^{k-1}_{1,:})^\top & e_k \; U^{k-1} (U^k_{:,1:k-1})^\top & \\ q_k^2 B_{k,k} & 2 q_k (U^k_{:,k})^\top & e_k \end{bmatrix} \tag{3.8}$$

of size $(r_{k-1} + 2) \times (r_k + 2)$.

Applying the TT-to-Tucker conversion technique, we obtain immediately, that the TT-ranks of the Tucker core are equal to the ranks obtained above, and the Tucker ranks equal 4 (3 in the symmetric case), since there are 4 (resp. 3) linearly independent elements in each block, e_k, q_k, q'_k and q_k^2. □

The proof gives a constructive routine for fast assembly of a bilinear form in the TT format. Indeed, performing the SVDs of submatrices of B (their sizes are of the order of tens) and building the blocks from (3.3), (3.6) (or (3.8) in symmetric case) and (3.7), we get the analytical TT representation.

In the case of a degree 2 polynomial on a uniform grid, the QTT ranks of the Tucker factors are equal to 3 (Section 2.2.1 and [192, 148]).

The ranks of the off-diagonal blocks are the so-called *quasiseparable* ranks. Theorem 3.2.1 establishes a connection between the coefficient matrix quasiseparability and the TT structure of the resulting bilinear form. This rank estimate is sharp: if the matrix is diagonal, i.e. $(C^k)^\top = \hat{C}^k = 0$, then the TT ranks equal 2 (Laplace operator, harmonic oscillator potential).

In our work, we always perform the recursion from left to right, revealing TT blocks one by one. In principle, if a recurrence relation is established for inner blocks, it is also possible to conduct decompositions from both sides of the tensor train, and finish them at any intermediate block k. This approach was presented in an independent work [129], devoted to the tensor structure analysis of the diffusion operator.

3.2.2 Gaussian distribution in the QTT format

Let us address now the second approach mentioned in the beginning of this chapter, and investigate an *approximate* TT decomposition and the corresponding ε-ranks. We saw in the previous section that the solution of the Fokker-Planck equation may be given by a multidimensional generalized Gaussian function. Unfortunately, if the matrix B in the Lyapunov function (3.1) is not diagonal, the corresponding Gaussian function may manifest quite large TT ranks – in fact, the advantages of the QTT-Tucker format w.r.t. the linear tree will be demonstrated on such an example. However, in the diagonal (also called *isotropic*) case one may see immediately that the Gaussian has all TT ranks ones:

$$\exp\left(-b_1 q_1^2 - b_2 q_2^2 - \cdots - b_d q_d^2\right) = \exp\left(-b_1 q_1^2\right) \cdots \exp\left(-b_d q_d^2\right).$$

So, our question now is the QTT structure estimation for a one dimensional Gaussian vector on a uniform grid, $g = \left[\exp(-\frac{q(i)^2}{2p^2})\right]$.

Lemma 3.2.2 ([47])**.** Suppose uniform grid points $-a = q(0) < q(1) < \cdots < q(n) = a$, $q(i) = -a + hi$, $n = 2^L$ are given on an interval $[-a, a]$, and the vector g is defined by its elements $g(i) = \mathrm{e}^{-\frac{q(i)^2}{2p^2}}$, $i = 0, \ldots, n-1$. Suppose in addition that $\int_a^\infty \mathrm{e}^{-\frac{q^2}{2p^2}} dq \leqslant \frac{\varepsilon}{2} < 1$. Then for all sufficiently small $\varepsilon > 0$ there exists a QTT approximation g_r with the ranks bounded as

$$r(g_r) \leqslant c \frac{a}{p} \sqrt{\log\left(\frac{1}{\varepsilon} \frac{p}{1+a}\right)}, \tag{3.9}$$

and the accuracy

$$|g - g_r| \leqslant \left(\frac{r}{2a} + 1\right) \varepsilon = \left(c \frac{1}{p} \sqrt{\log\left(\frac{1}{\varepsilon} \frac{p}{1+a}\right)} + 1\right) \varepsilon,$$

where c does not depend on a, p, ε or n.

Proof. We employ the Fourier transform of the Gaussian function. Indeed, consider approximation via a partial Fourier sum

$$\mathrm{e}^{-\frac{q^2}{2p^2}} = \sum_{m=0}^{M} \alpha_m \cos\left(\frac{\pi m q}{a}\right) + \eta \quad \text{on } [-a, a],$$

where $|\eta| = \left| \sum_{m=M+1}^{\infty} \alpha_m \cos\left(\frac{\pi m q}{a}\right) \right| < \varepsilon$. There are no sin functions in this sum, since the Gaussian function is even with respect to the origin. If we discretize now this sum on a uniform grid, all vectors generated by cos functions will have exact QTT-representations with all ranks equal to 2, see [148] and Section 2.2.1. So it is enough to provide an estimate on M.

The Fourier coefficients are computed as

$$\alpha_m = \frac{\int\limits_{-a}^{a} \mathrm{e}^{-\frac{q^2}{2p^2}} \cos\left(\frac{\pi m q}{a}\right) dq}{\int\limits_{-a}^{a} \cos^2\left(\frac{\pi m q}{a}\right) dq},$$

where all denominators are equal to a if $m > 0$, and $2a$ if $m = 0$. Let us denote them

$$|C_m|^2 = \begin{cases} 2a, & \text{if } m = 0, \\ a, & \text{otherwise.} \end{cases}$$

In the nominator, we note that the cos function is bounded by 1, and

$$\int\limits_{-\infty}^{\infty} \mathrm{e}^{-\frac{q^2}{2p^2}} dq = \int\limits_{-a}^{a} \mathrm{e}^{-\frac{q^2}{2p^2}} dq + 2\int\limits_{a}^{\infty} \mathrm{e}^{-\frac{q^2}{2p^2}} dq \leqslant \int\limits_{-a}^{a} \mathrm{e}^{-\frac{q^2}{2p^2}} dq + \varepsilon,$$

so we approximate

$$\alpha_m = \left(\int\limits_{-\infty}^{\infty} \mathrm{e}^{-\frac{q^2}{2p^2}} \cos\left(\frac{\pi m q}{a}\right) dq - \xi_m \right) / |C_m|^2, \quad 0 < \xi_m < \varepsilon.$$

We deduce the integral over the whole axis from the continuous Fourier transform: indeed, it is known that the Fourier image of the Gaussian function is another Gaussian function:

$$\int\limits_{-\infty}^{\infty} \mathrm{e}^{-\frac{q^2}{2p^2}} \mathrm{e}^{\mathrm{i}\omega q} dq = \int\limits_{-\infty}^{\infty} \mathrm{e}^{-\frac{q^2}{2p^2}} \cos(\omega q) dq = p\mathrm{e}^{-\frac{\omega^2 p^2}{2}},$$

where i is the imaginary unity. So, plugging here $\omega = \frac{\pi m}{a}$ in, we get the statement for α_m:

$$\alpha_m = \left(p\mathrm{e}^{-\frac{\pi^2 m^2 p^2}{2a^2}} - \xi_m \right) / |C_m|^2$$

Now we truncate the series at a value $m = M$ such that $\alpha_M \leqslant \varepsilon$, hence the condition on M writes

$$M \geqslant \frac{\sqrt{2}}{\pi} \frac{a}{p} \log^{0.5}\left(\frac{p}{(1 + |C_M|^2)\varepsilon} \right) = \frac{\sqrt{2}}{\pi} \frac{a}{p} \log^{0.5}\left(\frac{p}{1 + a} \frac{1}{\varepsilon} \right),$$

which gives the first result of the lemma (up to rank 2 of each cosine function). Note that due to the very fast decay of the Fourier coefficients, the threshold $\alpha_M \leqslant \varepsilon$ implies $\sum\limits_{m=M+1}^{\infty} \alpha_m \leqslant \varepsilon$ as well, starting from a small enough ε.

To obtain an expression for the error, recall that

$$g(q) = \mathrm{e}^{-\frac{q^2}{2p^2}} = \sum_{m=0}^{M} \frac{1}{|C_m|^2} \left(p\mathrm{e}^{-\frac{\pi^2 m^2 p^2}{2a^2}} - \xi_m \right) \cos\left(\frac{\pi m q}{a}\right) + \eta,$$

$$g_r(q) = \sum_{m=0}^{M} \frac{1}{|C_m|^2} p \mathrm{e}^{-\frac{\pi^2 m^2 p^2}{2a^2}} \cos\left(\frac{\pi m q}{a}\right).$$

Now taking into account the bounds $|\xi_m| < \varepsilon$, $|\cos\left(\frac{\pi m q}{a}\right)| \leqslant 1$, $|\eta| < \varepsilon$, $r = 2M$, we get the estimate for $|g - g_r|$. □

Remark 3.2.3. Requiring that $\mathrm{e}^{-\frac{a^2}{2p^2}} \leqslant \varepsilon$ (it is to be imposed naturally in order to compute a physically relevant solution without significant boundary effects) we obtain $a = \sqrt{2}p \log^{0.5}(1/\varepsilon)$, so that $r(g_r) \leqslant c \log(1/\varepsilon)$. This result can be deduced via the polynomial approximation. But now we have the *uniform* estimate with respect to all parameters except the accuracy, hence the rank does not grow with $1/p$.

Remark 3.2.4. Since the Gaussians with different p are distinguished by the Fourier coefficients α_m only, a set of Gaussians may be represented in a common QTT format simultaneously, with the rank bound (3.9), where the minimal p from the set is plugged in. Such a shared representation was recently used in quantum chemistry calculations [139].

In a similar way, some analytical examples of the stationary solutions and propensities in the chemical master equation (e.g. the Poisson distribution vector) may be derived using truncated polynomial or trigonometric series [126, 130].

3.2.3 Cascade operator

Considering the chemical master equation, we saw that the tensor structure of the CME operator is governed naturally by the tensor structure of the inputs (propensities), and in a general case is defined by the sum of all reaction terms. However, a couple of most important cases of *system interactions* are worth to be analyzed in more details.

The simplest example is a sum of independent actions or operators, described in Section 2.2.3,

$$\begin{aligned} x(\boldsymbol{i}) &= a(i_1) \cdot b(i_2) \cdots b(i_d) + \cdots + b(i_1) \cdots b(i_{d-1}) \cdot a(i_d) \\ &= \begin{bmatrix} a(i_1) & b(i_1) \end{bmatrix} \begin{bmatrix} b(i_2) & 0 \\ a(i_2) & b(i_2) \end{bmatrix} \cdots \begin{bmatrix} b(i_{d-1}) & 0 \\ a(i_{d-1}) & b(i_{d-1}) \end{bmatrix} \begin{bmatrix} b(i_d) \\ a(i_d) \end{bmatrix}. \end{aligned}$$

One step further is the *pairwise*, or *cascadic* interaction, which is also well representable in the TT format.

Lemma 3.2.5 ([46])**.** Given the elementary vectors or matrices $E_k(i_k)$, $F_k^k(i_k)$, $F_k^{k+1}(i_k)$ for $k = 1, \ldots, d$. The cascadic sum

$$H = F_1^1 \otimes \left(\bigotimes_{j=2}^{d} E_j \right) + \sum_{k=2}^{d} \left(\bigotimes_{j=1}^{k-2} E_j \right) \otimes F_{k-1}^k \otimes F_k^k \otimes \left(\bigotimes_{j=k+1}^{d} E_j \right) \tag{3.10}$$

possesses an exact rank-3 TT decomposition $H(\boldsymbol{i}) = H^{(1)}(i_1)\cdots H^{(d)}(i_d)$, where

$$\begin{aligned} H^{(1)}(i_1) &= \begin{bmatrix} E_1(i_1) & F_1^2(i_1) & F_1^1(i_1) \end{bmatrix}, \quad H^{(d)}(i_d) = \begin{bmatrix} 0 \\ F_d^d(i_d) \\ E_d(i_d) \end{bmatrix}, \\ H^{(k)}(i_k) &= \begin{bmatrix} E_k(i_k) & F_k^{k+1}(i_k) & 0 \\ 0 & 0 & F_k^k(i_k) \\ 0 & 0 & E_k(i_k) \end{bmatrix}, \quad \text{if } k = 2, \ldots, d-1. \end{aligned} \tag{3.11}$$

For the Tucker decomposition the same rank-3 bound holds.

Proof. The first step of the recurrent splitting writes

$$H(\boldsymbol{i}) = \begin{bmatrix} E_1(i_1) & F_1^2(i_1) & F_1^1(i_1) \end{bmatrix} \begin{bmatrix} \tilde{H}_2(i_2, \ldots, i_d) \\ F_2^2(i_2)E_3(i_3)\cdots E_d(i_d) \\ E_2(i_2)\cdots E_d(i_d) \end{bmatrix},$$

where the first term is exactly the first TT block, and in the rest we denote

$$\tilde{H}_k = F_k^{k+1} \otimes F_{k+1}^{k+1} \otimes E_{k+1} \otimes \cdots \otimes E_d + \cdots + E_k \otimes \cdots \otimes E_{d-2} \otimes F_{d-1}^d \otimes F_d^d.$$

In the general form of the second term we split the k-th dimension from each row in the same manner,

$$\begin{bmatrix} \tilde{H}_k(i_k, \ldots, i_d) \\ F_k^k(i_k)E_{k+1}(i_{k+1})\cdots E_d(i_d) \\ E_k(i_k)\cdots E_d(i_d) \end{bmatrix} = H^{(k)}(i_k) \begin{bmatrix} \tilde{H}_{k+1}(i_{k+1}, \ldots, i_d) \\ F_{k+1}^{k+1}(i_{k+1})E_{k+2}(i_{k+2})\cdots E_d(i_d) \\ E_{k+1}(i_{k+1})\cdots E_d(i_d) \end{bmatrix},$$

and derive the k-th TT block. The recurrence rule is now established, so we can continue the process. The last two factors are separated as follows,

$$\begin{bmatrix} F_{d-1}^d(i_{d-1})F_d^d(i_d) \\ F_{d-1}^{d-1}(i_{d-1})E_d(i_d) \\ E_{d-1}(i_{d-1})E_d(i_d) \end{bmatrix} = \begin{bmatrix} F_{d-1}^d(i_{d-1}) & 0 \\ 0 & F_{d-1}^{d-1}(i_{d-1}) \\ 0 & E_{d-1}(i_{d-1}) \end{bmatrix} \begin{bmatrix} F_d^d(i_d) \\ E_d(i_d) \end{bmatrix}.$$

Finally, padding the second term with zeros to achieve the form of $H^{(d)}$ in (3.11), the block $H^{(d-1)}$ is cast to the common form $H^{(k)}$. We see that all TT ranks are equal to 3, which confirms the claim of the lemma. To obtain the Tucker rank estimate, it is sufficient to note that each TT block contains only 3 independent elements, and follow the TT-to-Tucker procedure. □

Remark 3.2.6. In (3.10), each summand is a rank-1 tensor. However, we can straightforwardly generalize the structure to the case, when the neighboring terms are summed from several components,

$$\sum_{\alpha_{k-1}=1}^{r_{k-1}} F_{k-1,\alpha_k}^k \otimes F_{k,\alpha_k}^k \quad \text{instead of} \quad F_{k-1}^k \otimes F_k^k,$$

i.e., the TT rank of each two-variate contribution (e.g. the propensity function in the CME) is not equal to 1. In this case, we can collect respectively the row and column vectors

$$F^k_{k-1}(i_{k-1}) = \begin{bmatrix} F^k_{k-1,1}(i_{k-1}) & \cdots & F^k_{k-1,r_{k-1}}(i_{k-1}) \end{bmatrix}, \quad F^k_k(i_k) = \begin{bmatrix} F^k_{k,1}(i_k) \\ \vdots \\ F^k_{k,r_{k-1}}(i_k) \end{bmatrix},$$

and the constructions (3.11) will be considered as block matrices, with the sizes (i.e. the TT ranks) $(2 + r_{k-1}) \times (2 + r_k)$. Counting the linearly independent elements in each TT factor, we conclude that the k-th Tucker rank is bounded by $1 + r_{k-1} + r_k$.

Remark 3.2.7. For simplicity, we considered the vector case, i.e. $H = [H(\boldsymbol{i})]$. However, all the same arguments hold for matrices, i.e. we may freely substitute $\boldsymbol{i}$ by $\boldsymbol{i}, \boldsymbol{j}$, i_k by i_k, j_k, and so on. This generalizes the result to the cascadic CME operator.

Remark 3.2.8. If $F^1_1 = 0$, the result of Lemma 3.2.5 can be derived from Theorem 3.2.1 on the bilinear form, by setting $q_k = F^{k+1}_k$, $q'_k = F^k_k$, $q^2_k = 0$, $e_k = E_k$, and $B = J^1$ according to (1.10). However, Remark 3.2.6 is not covered by Thm. 3.2.1.

3.3 Inverse Laplace operator and Fourier transform

In some cases, not only a particular solution, but the whole inverse operator may be constructed in a structured form. One of the first and remarkable examples is the Laplace operator [23, 71, 89, 72, 100, 101, 21].

Given a positive definite matrix A, its inverse can be written as follows,

$$A^{-1} = \int_0^\infty \exp(-tA)dt,$$

which may be simply verified by multiplying the right-hand side with A. Now, if

$$A = A^{(1)} \otimes I \otimes \cdots \otimes I + \cdots + I \otimes \cdots \otimes I \otimes A^{(d)},$$

the exponential factorizes to the tensor product of univariate terms,

$$A^{-1} = \int_0^\infty \exp(-tA^{(1)}) \otimes \exp(-tA^{(2)}) \otimes \cdots \otimes \exp(-tA^{(d)})dt.$$

A constructive approximation of A^{-1} in the canonical tensor format is now given by replacing the integral by a finite quadrature. For example, the Stenger rule [89] computes

$$\begin{aligned} A^{-1} &\approx \sum_{m=-M}^{M} c_m \bigotimes_{k=1}^{d} \exp(-t_m A^{(k)}), \\ t_m &= \log\left(\exp(mh) + \sqrt{1 + \exp(2mh)}\right), \quad h = \pi^2/\sqrt{M}, \\ c_m &= h/\sqrt{1 + \exp(-2mh)}, \end{aligned}$$

providing the sub-exponential convergence

$$\left\| A^{-1} - \sum_{m=-M}^{M} c_m \bigotimes_{k=1}^{d} \exp(-t_m A^{(k)}) \right\|_2 \leqslant C(4 + 2\|A\|_2) \exp\left(-\pi\sqrt{2M}\right).$$

Obviously, the canonical rank estimates as $R = (2M + 1) = \mathcal{O}(\log^2(1/\varepsilon))$. There also exist sinc-quadratures with $\mathcal{O}(\exp(-cM/\log M))$ convergence, see e.g. the survey [147].

Interestingly, the approximation quality barely depends on small perturbations in the quadrature rule. For example, in order to compute matrix exponentials $B_m^k = \exp(-t_m A^{(k)})$ via fast recursion formulae, e.g. $B_{m+1}^k = (B_m^k)^2$, a modified Sinc-quadrature rule was proposed in [157]. Given an original Sinc scheme

$$h = \pi/\sqrt{M}, \quad t_m = \mathrm{e}^{mh}, \quad c_m = h t_m,$$

we set formally $h = \log 2$, so that $t_m = 2^m = 2t_{m-1}$. The exact rule would require a noninteger $M = (\pi/h)^2 \approx 20.54$, so we choose $M = 21$, which still provides a reasonable approximation, especially for preconditioning purposes. The same procedure may be applied to more general matrices of the Kronecker form, for example, arising from the Lyapunov equation [17].

If not the whole inverse operator, but only a solution to a linear system $Ax = y$ is required, it may be more efficient to avoid the full matrices $\exp(-t_m A^{(k)})$, but instead compute the products $\exp(-t_m A^{(k)}) y^{(k)}$ directly, using the fast Fourier and trigonometric transforms. The discrete multidimensional Fourier transform writes by definition

$$\hat{y}(j_1, \ldots, j_d) = \sum_{i_1, \ldots, i_d} \exp\left(-\frac{2\pi\mathrm{i}}{n_1} i_1 j_1\right) \cdots \exp\left(-\frac{2\pi\mathrm{i}}{n_d} i_d j_d\right) y(i_1, \ldots, i_d),$$

which constitutes a rank-1 multilevel matrix by a vector product

$$\hat{y} = \left(F^{(1)} \otimes \cdots \otimes F^{(d)}\right) y, \quad F^{(k)}_{j_k, i_k} = \exp\left(-\frac{2\pi\mathrm{i}}{n_k} i_k j_k\right).$$

However, there is no such straightforward extension to the QTT format, since the one-dimensional Fourier matrix has an irreducible QTT rank 2^L for any accuracy level up to $1 - \delta$. Nevertheless, the *computation* of the Fourier transform can be performed in the QTT format with a $\mathcal{O}(r^3 L^2)$ cost [50], though at a price of intermediate vectors, which may potentially develop large QTT ranks even if both input and output are well structured.

Chapter 4

Classical and alternating tensor approximation and solution methods

We have mentioned only explicit operations, which can be performed in a finite number of steps with a guaranteed result. Even an inverse operator in the last section was constructed analytically. However, the variety of tensor operations is not limited to a basic multilinear algebra. Not of less importance are the *implicit* solutions, which can be computed only iteratively and approximately in many cases. Successful high-dimensional simulation (via Fokker-Planck or chemical master equations, for example) requires solution of linear systems (stationary problems and implicit time schemes), eigenvalue problems, more sophisticated matrix functions like exponential, etc. In this work, matrix functions and eigenvalue problems are left aside, since all applications identified in the beginning may be treated via the linear system approach. We devote this chapter to a short review of known methods and a family of novel algorithms, addressing the linear solution problem.

To shorten the formulae, throughout this chapter we reserve the anonymous rank notation r, r_k for the tensor ranks of the solution vector x, i.e. $r = r(x)$, $r_k = r_k(x)$. For other quantities we denote the origin as proposed in Def. 2.1.10, e.g. $r_k(A)$, $r_k(b)$.

4.1 Truncated iterations

The tensor linear algebra together with the rounding procedure allows to think in terms of classical algorithms, and the first attempts were connected with the usage of traditional iterative methods, equipped with the tensor arithmetics:

- Richardson and Newton iterations [210, 104, 141, 145, 156, 6, 19],
- conjugate and bi-conjugate gradients [169, 170, 162, 163, 6, 19, 27], or
- different GMRES realizations [9, 45].

The simplest truncated Richardson iteration may be performed as shown in Alg. 6.

The (left) preconditioner B may be easily incorporated by repeating steps 2 and 3 for the preconditioner action, i.e. the total MatVec reads $w = \mathcal{T}_\varepsilon\left(B\mathcal{T}_\varepsilon(Ax_0)\right)$, and b in Line 4 must be replaced by Bb.

Algorithm 6 TT-Richardson iteration

Require: Matrix A in the TT format (2.11), initial guess x_0 and right-hand side b in the TT formats (2.9), step size λ, approximation accuracy ε.
Ensure: Updated solution x_1.
1: **for** until convergence **do**
2: Compute the MatVec $w = Ax_0$ in the TT format.
3: Approximate $w = \mathcal{T}_\varepsilon(w)$. {Optional}
4: Compute $x_1 = x_0 + \lambda b - \lambda w$ in the TT format.
5: Approximate $x_1 = \mathcal{T}_\varepsilon(x_1)$.
6: **end for**

Basic error analysis may be provided in presence of the solution truncation (Line 5 in Alg. 6), but in assumption of exact residual, $w = Ax_0$.

Lemma 4.1.1. Given the linear system $Ax = b$ with the exact solution $x_\star$, denote the initial error as $e_0 = x_\star - x_0$, and the error after one ε-perturbed Richardson correction as $e_1 = x_\star - x_1$. Suppose that the unperturbed method provides the progress $\|e_1\|/\|e_0\| \leqslant \Omega < 1$. Then the inexact Alg. 6 is convergent with the rate bound

$$\frac{\|e_1\|}{\|e_0\|} \leqslant \Omega \left(1 + \varepsilon \frac{\|x_0\|}{\|e_0\|}\right), \tag{4.1}$$

until the following neighborhood of the exact solution is reached,

$$\frac{\|e_0\|}{\|x_0\|} \leqslant \varepsilon \frac{\Omega}{1-\Omega}.$$

Proof. Let us consider the solution truncation in the first step, i.e. we see Alg. 6 in the form

1. truncate the initial guess $u = \mathcal{T}_\varepsilon(x_0)$;
2. correct the solution $x_1 = u + \lambda(b - Au)$.

Then the progress may be written

$$\frac{\|x_\star - x_1\|}{\|x_\star - x_0\|} = \frac{\|x_\star - x_1\|}{\|x_\star - u\|} \cdot \frac{\|x_\star - u\|}{\|x_\star - x_0\|} \leqslant \Omega \cdot \frac{\|x_\star - u\|}{\|x_\star - x_0\|}.$$

Now we plug in the perturbation $u = x_0 + \eta$, $\|\eta\| \leqslant \varepsilon \|x_0\|$, to obtain

$$\frac{\|x_\star - x_1\|}{\|x_\star - x_0\|} \leqslant \Omega \cdot \frac{\|x_\star - x_0\| + \|\eta\|}{\|x_\star - x_0\|} \leqslant \Omega \left(1 + \varepsilon \left(\frac{\|x_\star - x_0\|}{\|x_0\|}\right)^{-1}\right).$$

The error decay is maintained below one if $\|e_0\|/\|x_0\| \geqslant \varepsilon \cdot \Omega/(1-\Omega)$. □

The formula (4.1) does not only show that the convergence deteriorates when the approximation approaches the $\mathcal{O}(\varepsilon)$-vicinity of the true solution. It also gives us an idea of *relaxation*: we are not obliged to keep the same ε for all iterations, but rather may admit less accurate and cheaper computations in certain parts of the process. Typically, in the first iterations the error $\|e_0\|$ is large, and we may use quite rough ε, still having a convergent method.

Algorithm 7 TT-GMRES(m) [45]

Require: Right-hand side b, initial vector x_0 in the TT format, matrix A as a tensor MatVec procedure $y = \mathcal{T}_{\varepsilon,r}(Ax)$, accuracy ε and/or maximal TT rank r.

Ensure: Approximate solution $x_j : \ \|Ax_j - b\|/\|b\| \leqslant \varepsilon$.

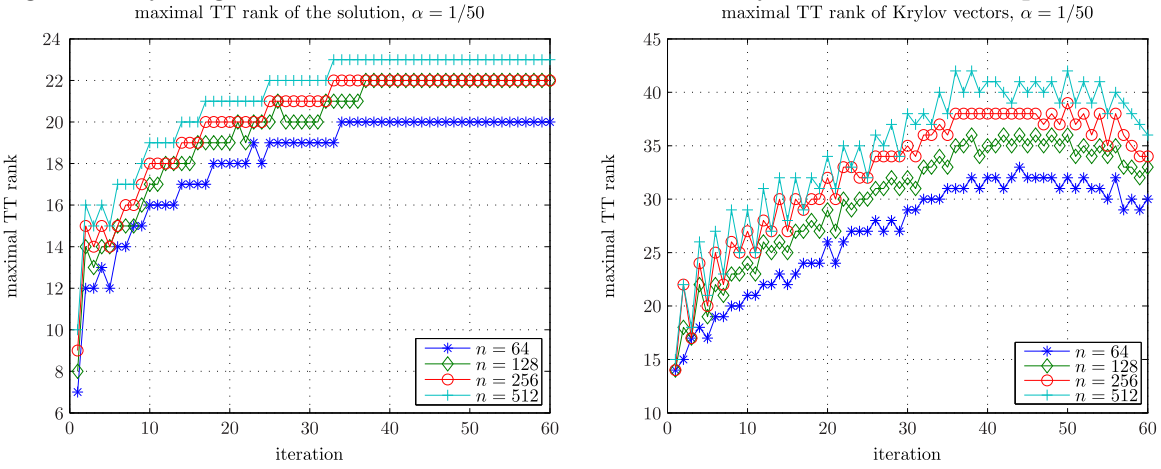
Start: compute $z_0 = \mathcal{T}_{\varepsilon,r}(b - Ax_0)$, $\beta = \|z_0\|$ $v_1 = z_0/\beta$.
Iterations:
for $j = 1, 2, \ldots, m$ **do**
  Compute the relaxed accuracy $\delta = \dfrac{\varepsilon}{\|\tilde{z}_{j-1}\|/\beta}$.
  $w = \mathcal{T}_{\delta,r}(Av_j)$: new Krylov vector.
  for $i = 1, 2, \ldots, j$ **do**
    $h_{i,j} = (w, v_i)$,
    $w = w - h_{i,j}v_i$, {orthogonalization}
  end for
  $w = \mathcal{T}_{\delta,r}(w)$. {compression}
  $h_{j+1,j} = \|w\|$, $v_{j+1} = w/h_{j+1,j}$.
  Assemble the matrix $\bar{H}_j = [h_{i,k}]$, $k = 1, \ldots, j$, $i = 1, \ldots, j+1$.
  Compute the reduced solution: $y_j = \arg\min_y \|\beta e_1 - \bar{H}_j y\|$.
  Check the residual $\|\tilde{z}_j\| = \|\beta e_1 - \bar{H}_j y_j\|$: if $\|\tilde{z}_j\|/\|b\| \leqslant \varepsilon$, then break.
end for
Update the solution: initialize $x_j = x_0$,
for $i = 1, 2, \ldots, j$ **do**
  $x_j = x_j + y_j(i)v_i$ {correction}
end for
$x_j = \mathcal{T}_{\varepsilon,r}(x)$ {compression}
Restart: if $\|\tilde{z}_j\|/\|b\| > \varepsilon$, then set $x_0 = x_j$, go to 1.

Remark 4.1.2. The progress Ω is typically estimated from the spectrum of the matrix A, e.g. in the form $\Omega \lesssim 1 - 1/\mathrm{cond}(A)$, where $\mathrm{cond}(A)$ is the condition number of the matrix. The error overhead is thus of the order of the condition number, $\Omega/(1-\Omega) \sim \mathrm{cond}(A)$. While it could be harmless in the standard machine arithmetics with the precision $\mathcal{O}(10^{-16})$, tensor format calculations involve much larger errors, e.g. 10^{-3}—10^{-8}. Therefore, the use of a spectrally equivalent preconditioner is crucial.

Not any preconditioning technique is easily adjustable for tensor structured representations. For example, the ILU approach requires the access to all elements of the matrix, which would lead to the prohibitively costly expansion of the tensor format. Among feasible techniques are multigrid methods (mostly geometric [9]; algebraic versions must be free from the full matrix inspection, e.g. the BPX variant in [6]), and approximate inverse operators with explicit low-rank structures. A remarkable example of the latter is the inverse Laplace operator (see Sec. 3.3), employed in [141, 145, 156, 45].

The lack of an efficient general-purpose preconditioner may be partially compensated by the use of more advanced tools. For example, the GMRES method with approximate tensor computations of the Krylov vectors may be implemented as shown in Alg 7. Note that the accuracy of the Krylov vectors is relaxed according to the cur-

Figure 4.1: An example from [45]: the TT-GMRES delivers the solution with rapidly saturating TT ranks (left), while the TT ranks of the last Krylov vector (right) are significantly larger and the relaxation reduces them only in the end of the process.



rent residual (Line 4). As shown in the theory of inexact Krylov methods [219], such a relaxation does not destroy the convergence. More precisely, the following statement holds.

Statement 4.1.3 (Corollary 4.1 [45] from Thm. 5.3 [219])**.** Suppose m GMRES iterations are conducted. If for any $i \leqslant m$ the relative error introduced in the Matrix-by-Vector multiplication satisfies

$$\frac{\|w\|}{\|Av_j\|} \leqslant \frac{1}{m \cdot \mathrm{cond}(A)} \frac{1}{\|\tilde{z}_{i-1}\|/\|b\|}\epsilon, \tag{4.2}$$

then the real relative residual and the one computed in the local GMRES problem are related as

$$\frac{\|z_m\|}{\|b\|} \leqslant \frac{\|\tilde{z}_m\|}{\|b\|} + \epsilon.$$

However, the Krylov vectors develop usually large tensor ranks despite the relaxation of the accuracy, see Fig. 4.1. This phenomenon may be argued by the observation that the GMRES extracts the spectral harmonics of the matrix subsequently, and the smoother the solution is, the higher are the oscillations in the residual and the Krylov vectors. More and more complicated structure of the latter Krylov vectors is reflected by large tensor ranks required to assure their approximation with a reasonable accuracy. This limits the applicability of the Krylov methods in tensor formats.

4.2 Constrained minimization on tensor format elements

4.2.1 Alternating vs. classical iterations

In the previous section we saw that the classical iterative methods with the tensor arithmetics are not very robust, since various auxiliary quantities may require large

tensor ranks, even if the matrix, right-hand side and the solution are well representable in the format. To get rid of additional vectors, another family of methods suggests to search for the elements of a tensor format directly.

The statement of the optimization problem on tensor format entries begins with the target function. The simplest choice is the Frobenius-norm error in terms of the full tensors,

$$\mathtt{E}_{x_\star} = \|x_\star - \tau(x^{(1)}, \dots, x^{(d)})\|^2 \to \min \tag{4.3}$$

over the TT blocks $x^{(k)} \in \mathbb{C}^{r_{k-1} \times n_k \times r_k}$. Note that due to the polylinear action of the TT map τ, the error function, being quadratic for the tensor elements, becomes highly nonlinear and nonconvex w.r.t. to the TT elements. Though the straightforward optimization is developed to some extent (see e.g. Newton and quasi-Newton methods for the canonical format [160, 36, 60, 133], or generalized eigenvalue decomposition [55]), this approach is usually applied to specific problems, and is far from being reliable in a more general setting.

To relax the nonlinearity, a family of *alternating* methods was proposed. A general approach is to substitute the problem (4.3) by a sequence of quadratic optimizations over the elements of each TT block. Since 1980's, the *alternating least squares* (ALS) method for the Tucker [165] and canonical [34] formats became the method of choices for low-rank data fitting in psychometrics and other classification problems.

Despite a ridiculously slow convergence in many cases, the ALS method for tensor product formats possesses one important advantage: since the format is linear with respect to each of its blocks, the target function restricted to the block elements remains of the same polynomial degree, as it was defined on the initial tensor elements. This makes the restricted problem much easier to solve than the simultaneous optimization (4.3). How to recover a fast convergence and avoid trapping in local minima will be a matter of this and next sections.

We first show that in particular the TT format is indeed linear with respect to a chosen block. Recall the definition of the TT map 2.1.11 for a chunk of the tensor train:

$$\begin{aligned} x^{(<k)} &= \tau(x^{(1)}, \dots, x^{(k-1)}) \in \mathbb{C}^{n_1 \cdots n_{k-1} \times r_{k-1}}, \\ x^{(>k)} &= \tau(x^{(k+1)}, \dots, x^{(d)}) \in \mathbb{C}^{r_k \times n_{k+1} \cdots n_d}. \end{aligned}$$

Now we may define the *frame* matrix.

Definition 4.2.1. Given a tensor train $x^{(1)}, \dots, x^{(d)}$, the k-th frame matrix reads

$$X_{\neq k} = x^{(<k)} \otimes I_{n_k} \otimes \left(x^{(>k)}\right)^\top \in \mathbb{C}^{n_1 \cdots n_d \times r_{k-1} n_k r_k}. \tag{4.4}$$

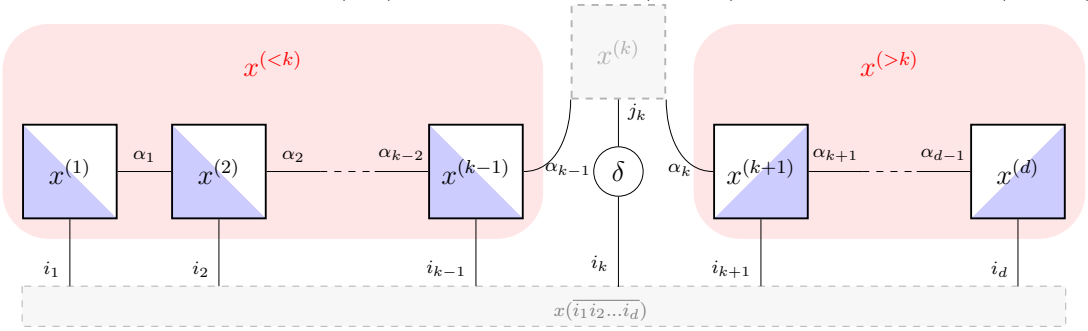
Directly from the tensor train definition follows the linearity statement.

Statement 4.2.2. The frame matrix performs a linear map from the elements of the TT block to the elements of the initial tensor,

$$x = X_{\neq k} x^{(k)}. \tag{4.5}$$

Moreover, imposing the orthogonality conditions on the TT blocks, we may make the whole frame matrix orthogonal. Indeed, recall from Lemma 2.1.16 that the chunk

Figure 4.2: The frame matrix (4.4) maps a TT block (above) into a large vector (below).



$x^{(<k)}$ is left-orthogonal if so are the corresponding TT blocks $x^{(1)}, \ldots, x^{(k-1)}$. Similarly, the right-orthogonality of the blocks $x^{(k+1)}, \ldots, x^{(d)}$ ensures the orthogonality of the right chunk $x^{(>k)}$, and together they yield an orthogonal frame matrix, as shown in Fig. 4.2.

4.2.2 Solution of linear algebra problems by optimization

The initial algebraic problem, such as an approximation, linear system solution or symmetric eigenvalue problem may be often considered as an optimization of a certain quadratic target function. The most important are:

1. error $\mathtt{E}_{x_\star}(x) = \|x_\star - x\|^2 = \mathtt{J}_{I,x_\star}$,
2. energy $\mathtt{J}_{A,b}(x) = \|x_\star - x\|_A^2 = (x, Ax) - 2\mathrm{Re}(x, b) + \mathrm{const}$ for $b = Ax_\star$,
3. residual $\mathtt{R}_{A,b}(x) = \|Ax - b\|^2 = \mathtt{J}_{A^*A,A^*b}(x) + \mathrm{const}$, and
4. Rayleigh quotient $\mathtt{Q}_A(x) = (x, Ax)/(x, x)$.

The minimum of the Rayleigh quotient is attained at the extreme eigenvalue problem solution $Ax = \lambda x$ with $\lambda = \min \mathtt{Q}_A(x)$, and is left aside in the current work, while the rest three functions are related to the solution of some (may be trivial with the identity matrix) linear system. The energy function is defined for a symmetric positive definite (SPD) matrix, $A = A^* > 0$, with the A-scalar product and A-norm introduced in the usual sense $(x, y)_A = (x, Ay)$, $\|x\|_A = \sqrt{(x, x)_A}$.

Now we recall the alternating least squares (ALS) method, a.k.a. the alternating *linear scheme* according to [115], where it was also shown that the ALS scheme coincides with the (one-site) *Density Matrix Renormalization Group* (DMRG) approach from quantum physics. The original DMRG scheme [237, 238] was proposed to compute the ground state of a system by the minimization of the Rayleigh quotient. It was later applied [124] to solve a SPD linear system $Ax = b$, where A and b are given in the TT format, by the minimization of the energy function. We focus on the latter linear system problem, but the connection with the "original" DMRG is straightforward, and most of insights can be adopted.

So, we restrict the optimization of $\mathtt{J}_{A,b}(x)$ to vectors $x = \tau(x^{(1)}, \ldots, x^{(d)})$ that are represented by the TT format with *fixed* TT ranks $\boldsymbol{r} = (r_1, \ldots, r_{d-1})$, and perform the

actual computations in a sequence of *microsteps*, i.e. *consecutive* optimizations over TT blocks $x^{(k)}$. Each such *local problem* writes

$$u^{(k)} = \arg\min_{x^{(k)}} \mathtt{J}_{A,b}(\tau(x^{(1)}, \dots, x^{(d)})) \quad \text{over} \quad x^{(k)} \in \mathbb{C}^{r_{k-1} \times n_k \times r_k}. \tag{4.6}$$

The TT core $x^{(k)}$ is then replaced by $u^{(k)}$, and the next core is considered; usually the cores are updated in a one-by-one sweeping through the tensor train, e.g. $k = 1, \dots, d$ (*forward half–sweep*), or $k = d, \dots, 1$ (*backward half–sweep*), and so on until x converges.

The linearity of the format (4.5) allows us to rewrite (4.6) as follows

$$\begin{gathered} u^{(k)} = \arg\min_{x^{(k)}} \mathtt{J}_{A_k,b_k}(x^{(k)}) \quad \text{over} \quad x^{(k)} \in \mathbb{C}^{r_{k-1} n_k r_k}, \\ A_k = X_{\neq k}^* A X_{\neq k} \in \mathbb{C}^{(r_{k-1} n_k r_k) \times (r_{k-1} n_k r_k)}, \qquad b_k = X_{\neq k}^* b \in \mathbb{C}^{r_{k-1} n_k r_k}. \end{gathered} \tag{4.7}$$

The unique minimum is delivered by the solution of the *local linear system* $A_k u^{(k)} = b_k$,[1] which is of a reasonable size and can be solved by a standard method, e.g. iterative schemes [208]. As shown in Fig. 4.3, A_k and b_k can be assembled from the TT blocks of $A = \tau(\{A^{(k)}\})$, $x = \tau(\{x^{(k)}\})$, and $b = \tau(\{b^{(k)}\})$, avoiding exponentially large arrays to appear.

The accuracy of the obtained solution crucially depends on the *conditioning* of the local system. Fortunately, it can be put under control using the orthogonality constraints on the TT blocks, and hence the orthogonality of the frame matrix. In the sequential sweeping through the TT blocks, the alternating optimization may be synchronized with the orthogonalization steps of Alg. 2 and 3, as shown in Alg. 8.

If $X_{\neq k}$ is orthogonal, the spectrum of A_k lies within the spectral range of A. Indeed,

$$\begin{aligned} \lambda_{\min}(A_k) = \lambda_{\min}(X_{\neq k}^* A X_{\neq k}) &= \min_{\|v\|=1} (X_{\neq k} v, A X_{\neq k} v) = \min_{\substack{\|u\|=1 \\ u \in \operatorname{span} X_{\neq k}}} (u, Au) \\ &\geqslant \min_{\|u\|=1} (u, Au) = \lambda_{\min}(A), \end{aligned}$$

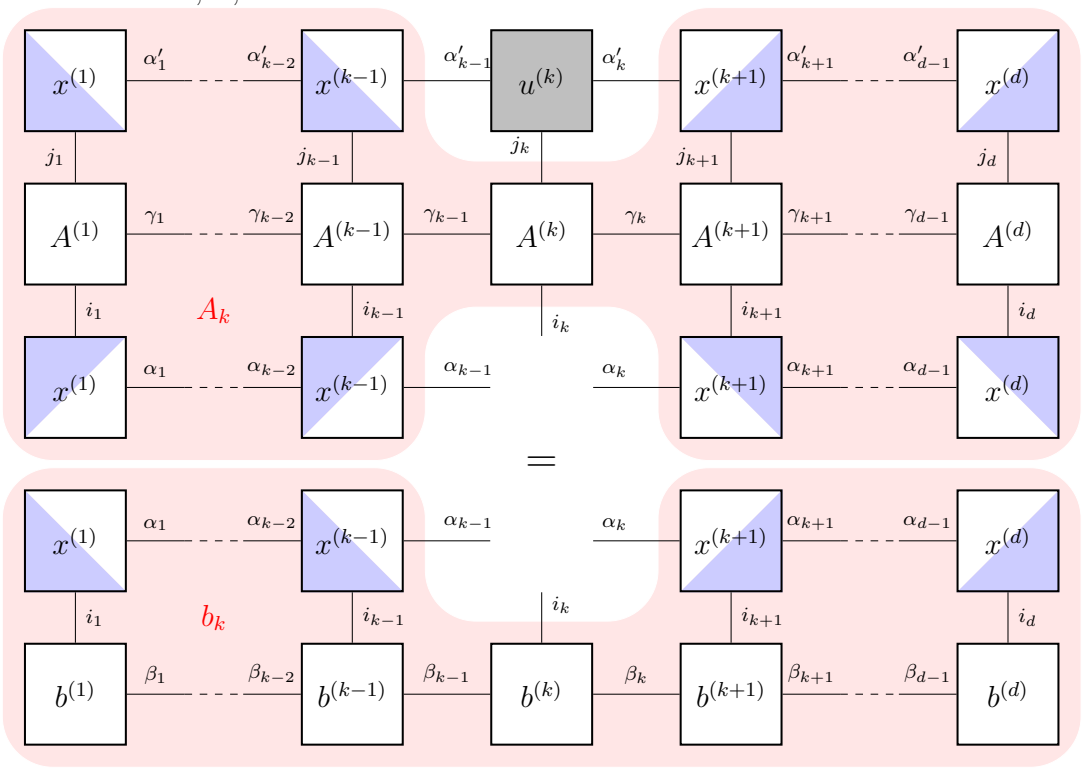
and similarly $\lambda_{\max}(A_k) \leqslant \lambda_{\max}(A)$. It follows that the *condition numbers* satisfy $\operatorname{cond}(A_k) \leqslant \operatorname{cond}(A)$, i.e. the local system (4.6) is conditioned not worse than $Ax = b$.

4.2.3 Rank adaptation problem and two-site DMRG

The drawback of the one-site DMRG algorithm described above is that the TT ranks remain the same during the computations. Therefore, we have to guess the TT ranks of the solution *a priori,* which might be difficult; if we underestimate them, the returned solution will be far from the exact one; if we overestimate them, the local problems will be more difficult to solve. Moreover, the convergence even to a quasi-optimal solution with prescribed TT ranks may be dramatically slow.

[1] Recall Def. 2.1.12 and text thereafter: the product $A_k u^{(k)}$ implies the "vector" reshape $u^{(k)} \in \mathbb{C}^{r_{k-1} n_k r_k}$.

Figure 4.3: The linear system $A_k u^{(k)} = b_k$ defined by (4.7), assembled from the cores of TT formats of A, x, and b.



Since the seminal papers [237, 238], the *two-site* DMRG became the method of choice to *change* (usually to *increase*) the TT ranks adaptively during the computations. A nice review of different DMRG techniques can be found in [216] and its "second edition" [217]. To the numerical linear algebra community the two-site DMRG was brought in [115] under the name *Modified Alternating Linear Scheme* (MALS).

This method considers the vector in the following form,

$$x = \tau(x^{(1)}, \dots, x^{(k-1)}, x^{(k,k+1)}, x^{(k+2)}, \dots, x^{(d)}), \tag{4.8}$$

where $x^{(k,k+1)} = x^{(k,\dots,k+1)} = \tau(x^{(k)}, x^{(k+1)})$ according to Def. 2.1.11. The local optimization step is performed similarly to (4.6), but over the elements of the *supercore* $x^{(k,k+1)}$ as follows,

$$u^{(k,k+1)} = \arg\min_{x^{(k,k+1)}} \mathrm{J}_{A_{k,k+1},b_{k,k+1}}(x^{(k,k+1)}) \quad \text{over} \quad x^{(k,k+1)} \in \mathbb{C}^{r_{k-1} n_k n_{k+1} r_{k+1}}, \tag{4.9}$$

and is equivalent to solving the *two-site local linear system* $A_{k,k+1} u^{(k,k+1)} = b_{k,k+1}$ with

$$\begin{aligned} A_{k,k+1} &= X^*_{\neq k,k+1} A X_{\neq k,k+1} \in \mathbb{C}^{(r_{k-1} n_k n_{k+1} r_{k+1}) \times (r_{k-1} n_k n_{k+1} r_{k+1})}, \\ b_{k,k+1} &= X^*_{\neq k,k+1} b \in \mathbb{C}^{r_{k-1} n_k n_{n+1} r_{k+1}}, \end{aligned} \tag{4.10}$$

where $X_{\neq k,k+1}$ is the *two-site frame matrix*

$$X_{\neq k,k+1} = x^{(<k)} \otimes I_{n_k} \otimes I_{n_{k+1}} \otimes \left(x^{(>k+1)}\right)^\top. \tag{4.11}$$

Algorithm 8 One-site DMRG for $Ax = b$ (forward half–sweep)

Require: Initial guess $t = \tau(\{t^{(k)}\})$ in the TT format (2.9).
Ensure: Updated vector $x = \tau(\{x^{(k)}\})$ s.t. $\mathtt{J}_{A,b}(x) \leqslant \mathtt{J}_{A,b}(t)$.
1: Copy $x^{(k)} = t^{(k)}$, $k = 1, \ldots, d$.
2: **for** $k = d, \ldots, 2$ **do** {Orthogonalization of the right chunk}
3: Find LQ decomposition $x^{\langle k|} = LQ$, $QQ^* = I$.
4: Replace $x^{\langle k|} := Q$, and $x^{|k-1\rangle} := x^{|k-1\rangle}L$.
5: **end for**
6: **for** $k = 1, \ldots, d$ **do** {Optimization over TT cores}
7: Form A_k and b_k by (4.7).
8: Solve $A_k u^{(k)} = b_k$. {If iterative algorithm is used, take $x^{(k)}$ as an initial guess}
9: Replace $x^{(k)} := u^{(k)}$.
10: **if** $k \neq d$ **then** {Orthogonalization of the left interface, if required}
11: Find QR decomposition $x^{|k\rangle} = QR$, $Q^*Q = I$.
12: Replace $x^{|k\rangle} := Q$, and $x^{\langle k+1|} := Rx^{\langle k+1|}$.
13: **end if**
14: **end for**
15: **return** $x = \tau(x^{(1)}, \ldots, x^{(d)})$.

As previously, we assume that the orthogonality $X^*_{\neq k,k+1} X_{\neq k,k+1} = I_{r_{k-1}n_k n_{k+1} r_{k+1}}$ is ensured, and the conditioning of the local problem is not worse that the one of the original system $Ax = b$.

The updated TT core $u^{(k,k+1)}$ is then separated back to $u^{(k)}$ and $u^{(k+1)}$ to recover the original TT structure: we reshape

$$u^{(k,k+1)}_{\alpha_{k-1},\alpha_{k+1}}(\overline{i_k i_{k+1}}) = u^{\langle k,k+1\rangle}(\overline{\alpha_{k-1} i_k}, \overline{i_{k+1}\alpha_{k+1}}),$$

and perform the approximate dyadic decomposition using e.g. the truncated SVD (2.2) or cross (2.3),

$$u^{\langle k,k+1\rangle} \approx \tilde{u}^{\langle k,k+1\rangle} = u^{|k\rangle} u^{\langle k+1|}, \quad u^{|k\rangle} \in \mathbb{C}^{r_{k-1}n_k \times r'_k}. \tag{4.12}$$

The perturbation introduced to $u^{\langle k,k+1\rangle}$ in this decomposition step affects the whole vector x, and the orthogonality of the frame matrix $X_{\neq k,k+1}$ guarantees that the Frobenius norms of the local perturbation (to $u^{\langle k,k+1\rangle}$) and the global perturbation (to x) are the same. Moreover, the orthogonality of the frame matrices can be easily maintained during the sweep, since the SVD returns orthogonal singular vectors, which just need to be assigned to $u^{|k\rangle}$. The whole procedure is presented in Alg. 9.

As in the TT rounding procedure (Section 2.1.5), there are three strategies to choose the new rank r'_k of the decomposition (4.12):

- require the *rank bound* $r'_k \leqslant r$ (which will be hit with probability one if $r \leqslant \min\{r_{k-1}n_k, n_{k+1}r_{k+1}\}$),
- require the relative *accuracy level* ε, and find the lowest rank r'_k that provides (4.12) such that $\|u^{\langle k,k+1\rangle} - \tilde{u}^{\langle k,k+1\rangle}\| \leqslant \varepsilon \|u^{\langle k,k+1\rangle}\|$, or

Algorithm 9 Two-site DMRG for $Ax = b$ (forward half–sweep)

Require: Initial guess $t = \tau(\{t^{(k)}\})$, accuracy ε or rank bound r.
Ensure: Updated vector $x = \tau(\{x^{(k)}\})$.
1: Copy $x^{(k)} = t^{(k)}$, $k = 1, \ldots, d$.
2: **for** $k = d, \ldots, 2$ **do** {Orthogonalization of the right chunk}
3: Find LQ decomposition $x^{\langle k|} = LQ$, $QQ^* = I$.
4: Replace $x^{\langle k|} := Q$, and $x^{|k-1\rangle} := x^{|k-1\rangle}L$.
5: **end for**
6: **for** $k = 1, \ldots, d-1$ **do** {Optimization over TT cores}
7: Form $A_{k,k+1}$ and $b_{k,k+1}$ by (4.10).
8: Solve $A_{k,k+1}u^{(k,k+1)} = b_{k,k+1}$. {Use $x^{(k,k+1)}$ as an initial guess}
9: Decompose $u^{(k,k+1)}$ into $u^{(k)}$ and $u^{(k+1)}$ by (4.12) s.t. $(u^{|k\rangle})^* u^{|k\rangle} = I$.
10: Choose r'_k s.t. $r'_k \leqslant r$ and/or $\|u^{\langle k,k+1\rangle} - \tilde{u}^{\langle k,k+1\rangle}\| \leqslant \varepsilon \|u^{\langle k,k+1\rangle}\|$ are satisfied.
11: Replace $x^{(k)} := u^{(k)}$ and $x^{(k+1)} := u^{(k+1)}$.
12: **end for**
13: **return** $x = \tau(x^{(1)}, \ldots, x^{(d)})$.

- use ε-strategy when possible, and limit $r'_k = r$ if the previous condition suggests a larger value.

The accuracy strategy may be also classified with respect to the particular *norm* used in the filtering condition. For example, if $\|x - \tilde{x}\| = \|u^{\langle k,k+1\rangle} - \tilde{u}^{\langle k,k+1\rangle}\| \leqslant \varepsilon \|u^{\langle k,k+1\rangle}\|$ is satisfied in the Frobenius norm, the residual estimates as follows,

$$\frac{\|A\tilde{x} - b\|}{\|b\|} \leqslant \frac{\|A\|\|\tilde{x} - x_\star\|}{\|Ax_\star\|} \leqslant \mathrm{cond}(A) \frac{\|\tilde{x} - x_\star\|}{\|x_\star\|} \leqslant \mathrm{cond}(A)\varepsilon.$$

To control the output residual more precisely, the following trick was proposed in [51]: we filter the singular vectors in the decomposition (4.12) not by the Frobenius criterion, but satisfying

$$\|A_{k,k+1}\tilde{u}^{(k,k+1)} - b_{k,k+1}\| \leqslant \varepsilon \|b_{k,k+1}\|$$

instead. Of course, this heuristics does not guarantee that the total residual is exactly below ε, but the overhead is usually close to one, since $A_{k,k+1}$ appears to reflect the spectrum of A relatively good. In a similar way, a criterion of the form $\|u^{(k,k+1)} - \tilde{u}^{(k,k+1)}\|_{A_{k,k+1}} \leqslant \varepsilon \|u^{(k,k+1)}\|_{A_{k,k+1}}$ may be requested.

After the decomposition is done, the TT cores $x^{(k)}$ and $x^{(k+1)}$ are replaced by $u^{(k)}$ and $u^{(k+1)}$, and the TT rank r_k is substituted by r'_k. The tensor structure changes in each step, and further optimization is carried out over the updated *tensor manifold*. This generally speeds up the convergence, but makes the process more difficult to analyze. In addition, even this two-site DMRG may converge not to the true solution $x_\star$, but to some local minimum of $\mathtt{J}_{A,b}(\tau(\{x^{(k)}\}))$, especially if started from a low-rank initial guess.

Remark 4.2.3. Both Alg. 8 and 9 may be targeted to solve the approximation problem

$$\min \mathtt{E}_{x_\star}(x) = \|x - x_\star\|^2 = \mathtt{J}_{I,x_\star}(x).$$

As soon as the interface matrices are orthogonal, the identity is preserved in local problems (4.7) and (4.10): for example,

$$A_k = X^*_{\neq k} A X_{\neq k} = X^*_{\neq k} I X_{\neq k} = I.$$

Therefore, in local problems we just assign $u^{(k)} = b_k$ or $u^{(k,k+1)} = b_{k,k+1}$, and proceed with no additional changes in the algorithms. This approach is especially useful for a fast approximation of matrix and Hadamard products in the TT format: a sought vector may have the form $x_\star = Ay$ with TT ranks $r(x_\star) = r(A)r(y)$. Its direct approximation is difficult, but the projection $X^*_{\neq k} Ay$ is much easier to compute if the approximant x is of small ranks.

Remark 4.2.4. Both Alg. 8 and 9 may be formally applied to solve non-symmetric systems. In this case, we consider the non-symmetric local systems (4.7) and (4.10) as general-type Galerkin projections. Though such a method is not variational (i.e. no target function for monotonous optimization can be associated), it works pretty well in some not very complicated cases, see e.g. [51, 47, 128, 126].

Interestingly, the alternating Galerkin bases generated from the current approximant may be used not only for solution of the linear system directly, but also for derivation of optimized shift parameters in the traditional ADI methods [18]. This can perhaps render alternating schemes from another point of view (such as Chebyshev or Krylov theories), but currently it is unclear whether a rigorous analysis is possible.

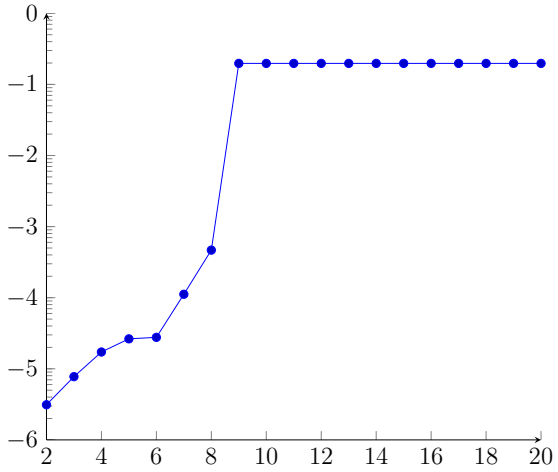
4.3 Adaptive alternating energy minimization as a black-box linear solver

4.3.1 A conception of enrichment

Due to the *local* manner of optimization, the DMRG methods (even two-site) may loose important portions of information about the direction towards the exact solution, and stagnate at some local minimum far from the desired threshold. Usually this phenomenon expresses with increasing dimensionality. For example, the cascade CME problem (see Section 3.2.3 and 5.1 for more detailed description and experiments) extends well to an arbitrary dimension, so we study the performance of the two-site DMRG versus the dimension in Fig. 4.4. We see that this algorithm performs well up to the dimension 6, but later the quality of the delivered solution deteriorates drastically, becoming worthlessly poor at the dimensions higher than 10. Is it possible to overcome this problem and develop reliable and efficient solution and approximation schemes for high dimensions? This section shows that it actually is.

Since [51] it was noted that it is surprisingly useful to add to the solution a TT tensor of small ranks with randomly filled blocks from time to time. Not only the previous accuracy level is recovered in the very next microstep (4.10), but the convergence becomes even faster in the forthcoming iterations. In [190], devoted to the fast DMRG-based MatVec approximation (see Remark 4.2.3), the conception of the *random kick*

Figure 4.4: Residual provided by the DMRG method vs. the dimension



was further developed: when the updated TT blocks $u^{(k)}, u^{(k+1)}$ are computed, the Line 11 in Alg. 9 is modified to the following rule,

$$x^{|k\rangle} = \begin{bmatrix} u^{|k\rangle} & s^{|k\rangle} \end{bmatrix}, \quad x^{\langle k+1|} = \begin{bmatrix} u^{\langle k+1|} \\ 0 \end{bmatrix}, \tag{4.13}$$

where $s^{|k\rangle} \in \mathbb{C}^{r_{k-1} n_k \times \rho_k}$ are randomly populated vectors, orthogonalized to $u^{|k\rangle}$, and the zero block in $x^{\langle k+1|}$ has the sizes $\rho_k \times n_{k+1} r_{k+1}$. The *enrichment* (4.13) may be seen as a zero TT-addition to the superblock: $x^{(k,k+1)} = u^{(k,k+1)} + s^{(k,k+1)}$, but the correction $s^{(k,k+1)} = \tau(s^{|k\rangle}, 0) = 0$. Hence both $x = \tau(x^{(1)}, \dots, x^{(d)})$ and $\mathtt{J}_{A,b}(x)$ are preserved by this operation, but not the interface matrix $x^{(<k+1)}$. It is easy to see that

$$x^{(<k+1)} = \begin{bmatrix} u^{(<k+1)} & \tau(u^{(<k)}, s^{(k)}) \end{bmatrix},$$

and hence the frame matrix expands as follows,

$$X_{\neq k+1,k+2} := \begin{bmatrix} X_{\neq k+1,k+2} & \tau(u^{(<k)}, s^{(k)}) \otimes I \otimes I \otimes (t^{(>k+2)})^\top \end{bmatrix}.$$

The second term brings new basis components for the optimization (at least inside the k-th block), and hence may accelerate the convergence.

However, for complicated problems even this trick does not prevent the method from trapping in local minima. Sometimes additional heuristics (e.g. artificially decreased accuracy threshold [45, Sec. 5]) improve the situation to some extent, but they still lack robustness.

Another difficulty of the two-site DMRG is at least cubic complexity w.r.t. the mode size n, due to the truncation step (4.12), while the one-site DMRG possesses an asymptotic $\mathcal{O}(n^2)$, or even $\mathcal{O}(n)$, if the sparsity of the matrix blocks $A^{(k)}$ is employed. Therefore, if the mode sizes are large (not any problem benefits from the QTT format), the one-site iteration may be significantly faster than the two-site counterpart.

Fortunately, note that the expansion (4.13) equips even a one-site scheme with the rank adaptivity. Thus the first issue of the one-site approach is resolved. The second issue (stagnation in local minima) can be now reformulated as a question, how to

choose the enrichment block $s^{(k)}$ in a smarter way than just at random, such that both the numerical efficiency and the *global* convergence in terms of full tensor elements are satisfactory.

4.3.2 Steepest descent technique and its error analysis

We have observed that the alternating schemes perform Galerkin projections (4.7), (4.10) of the initial system to the elements of each TT block. In this sense this approach is similar to the classical projection methods, e.g. GMRES Alg. 7 in the TT format; the difference is that the frame matrices do not approximate Krylov bases, and hence the stagnation may occur. Our principal idea will be to combine the alternating and classical schemes, enriching the frame matrix with the Krylov-type information.

Since we would like to avoid senior Krylov vectors with large TT ranks, we consider the simplest technique, that provides nevertheless a geometrical convergence – the *steepest descent* (SD) algorithm. The traditional SD algorithm uses only the first Krylov vector, i.e. the residual, to drive the approximant towards the exact solution. Since all methods in the rest of the section will be *one-step*, it will be convenient to change a bit the classical notation used for iterative algorithms.

An iterative algorithm starts from the *initial guess*, denoted usually as x_0, and generates a sequence of approximations $x_1, x_2, \ldots$, s.t. $x_i \to x_\star = A^{-1}b$, the exact solution. A method is called *geometrically* convergent if there exists a uniform bound $\Omega < 1$, called the *convergence rate*, such that $\|x_\star - x_i\| \leqslant \|x_\star - x_0\|\Omega^i$ with Ω independent on x_0.

One-step techniques (SD, ALS and DMRG (in terms of full cycles over all TT cores) belong to this family) use the same formulae in all iterations, independently on i or x_i. Therefore, we will estimate Ω from one iteration, i.e. $\|x_\star - x_1\|/\|x_\star - x_0\| \leqslant \Omega$ for any x_0. The subscripted number of iteration i thus becomes superfluous, and we adopt a simplified notation, typical in the analysis of one-step methods (see e.g. [185] and the transition from a general to the Markov process in Section 1.1.1): we agree that

- $t \equiv x_0$ is an initial guess,
- $x \equiv x_1$ is the newer approximation,
- $z = b - At$ is the residual,
- $c = x_\star - t$ is the initial error, and
- $f = x_\star - x$ is the newer error.

The subscript is reserved for microsteps $k = 1, \ldots, d$ of the optimization over TT cores in the alternating framework.

So, we begin with a short description of the classical steepest descent step. It minimizes the energy function in the gradient direction, i.e.

$$\begin{aligned} z &= -\nabla \mathsf{J}_{A,b}(t) = b - At, \\ h &= \arg\min_{h'} \mathsf{J}_{A,b}(t + zh') = \frac{(z,z)}{(z,Az)}. \end{aligned} \tag{4.14}$$

In other terms, the new solution $x = t + zh$ satisfies the *Galerkin condition* on the residual vector, $(z, b - Ax) = 0$. The new error writes

$$f = c - zh = c - \frac{z(z,z)}{(z,Az)} = (I - \mathcal{P}_{A,z})c, \qquad \text{where} \quad \mathcal{P}_{A,z} = \frac{zz^*A}{z^*Az}$$

is the A–orthogonal projector onto $\mathrm{span}(z)$, and may be estimated as follows,

$$\frac{\mathtt{J}_{A,b}(x)}{\mathtt{J}_{A,b}(t)} = \frac{\|f\|_A^2}{\|c\|_A^2} = \frac{(c,(I-\mathcal{P}_{A,z})^*A(I-\mathcal{P}_{A,z})c)}{(c,Ac)} = 1 - \frac{(c,\mathcal{P}_{A,z}c)_A}{(c,c)_A} = \omega_{z,z}^2. \tag{4.15}$$

The orthoprojection yields the monotonicity

$$\mathtt{J}_{A,b}(x) = \|f\|_A^2 \leqslant \|c\|_A^2 = \mathtt{J}_{A,b}(t).$$

For a future convenience, we feature the *exact progress* $\omega_{z,z}$, equal to the quotient of errors. Surely, it depends *a posteriori* on the current initial guess t. However, it is uniformly bounded from above, according to the *Kantorovich inequality* [125],

$$\omega_{z,z} = \sqrt{1 - \frac{(z,z)}{(z,Az)}\frac{(z,z)}{(z,A^{-1}z)}} \leqslant \frac{\lambda_{\max}(A) - \lambda_{\min}(A)}{\lambda_{\max}(A) + \lambda_{\min}(A)} = \frac{\mathrm{cond}(A) - 1}{\mathrm{cond}(A) + 1} = \Omega < 1, \tag{4.16}$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are minimal and maximal eigenvalues of A, respectively.

Remark 4.3.1. For $z = u_{\min}(A) + u_{\max}(A)$, where $u_{\min}(A)$ and $u_{\max}(A)$ are normalized eigenvectors of A corresponding to $\lambda_{\min}$ and $\lambda_{\max}$, (4.25) turns into an equality, that is, the bound Ω is sharp.

As was discussed alongside with the local system (4.7), a limited conditioning of the problem, $\mathrm{cond}(A) \leqslant C < \infty$ is crucial to ensure an accurate solution. Therefore, we may always assume that the *a priori* bound $\omega_{z,z} \leqslant \Omega < 1$ exists.

Now consider the *inexact* SD step,

$$x = t + \tilde{z}h, \qquad h = \arg\min_{h'} \mathtt{J}_{A,b}(t + \tilde{z}h') = \frac{(\tilde{z},z)}{(\tilde{z},A\tilde{z})}, \tag{4.17}$$

which uses the approximate residual $\tilde{z} \approx z$, and results in the perturbed new error $\tilde{f} = x_\star - x = (I - \mathcal{P}_{A,\tilde{z}})c$. It is important to take approximation errors into account, since they are non-negligible in numerically efficient tensor product methods.

Unlike the inexact GMRES convergence Statement 4.1.3, where bounds on the error $\|z - \tilde{z}\|$ were required, the steepest descent scheme allows to impose very mild limitations, based on the *angle* $\angle(z;\tilde{z})$ between z and $\tilde{z}$.

Statement 4.3.2 (Convergence of the inexact SD [182])**.** Given a SPD linear system $Ax = b$ and an initial vector t, consider the residual $z = b - At$ and a vector $\tilde{z}$, s.t. $\angle(z;\tilde{z}) \leqslant \theta < \pi/2$. The inexact SD step (4.17) provides the following progress,

$$\omega_{z,\tilde{z}} = \frac{\|\tilde{f}\|_A}{\|c\|_A} \leqslant \frac{\tilde{\kappa} - 1}{\tilde{\kappa} + 1} = \tilde{\Omega} < 1, \qquad \tilde{\kappa} = \mathrm{cond}(A)\frac{1 + \sin\theta}{1 - \sin\theta}. \tag{4.18}$$

Proof. Directly from $\tilde{f} = (I - \mathcal{P}_{A,\tilde{z}})c$ we obtain

$$\omega_{z,\tilde{z}}^2 = \frac{\mathtt{J}_{A,b}(x)}{\mathtt{J}_{A,b}(t)} = \frac{\|\tilde{f}\|_A^2}{\|c\|_A^2} = 1 - \frac{|(\tilde{z}, z)|^2}{(\tilde{z}, A\tilde{z})(z, A^{-1}z)}. \tag{4.19}$$

To bound $\omega_{z,\tilde{z}}$ from above similarly to (4.16), we use the generalization of the Kantorovich inequality from [14, Corrolary IV],

$$\frac{(\tilde{z}, A\tilde{z})(z, A^{-1}z)}{\|\tilde{z}\|^2\|z\|^2} \leqslant \frac{\left((\kappa+1) + (\kappa-1)\sin\theta\right)^2}{4\kappa}, \quad \kappa = \mathrm{cond}(A). \tag{4.20}$$

Together with $\cos\angle(\tilde{z}; z) \geqslant \cos\theta$ and identity $\frac{1+\sin\theta}{\cos\theta} = \frac{\cos\theta}{1-\sin\theta} = \sqrt{\frac{1+\sin\theta}{1-\sin\theta}}$ it gives

$$\frac{|(\tilde{z}, z)|^2}{(\tilde{z}, A\tilde{z})(z, A^{-1}z)} \geqslant \frac{4\kappa\cos^2\theta}{\left(\kappa(1+\sin\theta) + (1-\sin\theta)\right)^2} = \left(\frac{2}{\tilde{\kappa}^{1/2} + \tilde{\kappa}^{-1/2}}\right)^2,$$

which, plugged into (4.19), completes the proof. □

In case of exact computations, $z = \tilde{z}$, the inequality (4.20) turns into the Kantorovich inequality (4.16). Another aesthetically pleasant feature of the inexact SD is that it makes progress on *any* step, which is not exactly perpendicular to the residual. In tensor product algorithms, $\tilde{z}$ is usually an orthogonal projection of z onto a low-rank format (e.g. in the SVD-based TT rounding, or ALS tuned for the approximation as in Remark 4.2.3),

$$z = \tilde{z} + \delta z, \qquad (\tilde{z}, \delta z) = 0, \qquad \|\delta z\| \leqslant \varepsilon\|z\|. \tag{4.21}$$

This gives $\sin\angle(\tilde{z}; z) = \|\delta z\|/\|z\| \leqslant \varepsilon$, and hence $\angle(\tilde{z}; z) < \pi/2$ as soon as $\varepsilon < 1$. Interestingly, even very rough approximation thresholds may ensure a sensible rate $\tilde{\Omega}$. Indeed, the complements of Ω and $\tilde{\Omega}$ to one can be related as follows,

$$1 \leqslant \frac{1-\Omega}{1-\tilde{\Omega}} \leqslant \frac{\frac{1+\varepsilon}{1-\varepsilon}\kappa + 1}{\kappa+1} = \frac{1 + \varepsilon\frac{\kappa-1}{\kappa+1}}{1-\varepsilon} \leqslant 3, \quad \text{for} \quad \varepsilon \leqslant \frac{1}{2}.$$

Thus, in addition to a known Ω, we may assume that ε is chosen *a priori* or controlled during the computations in such a way that $\tilde{\Omega}$ in (4.18) is a reasonable *a priori* bound for the convergence rate of the inexact SD algorithm.

We will also need a *wide* steepest descent generalization, since the interface and frame matrices contain sets of vectors, instead of a single residual. Given a nonsingular matrix Z of a suitable size, the solution is corrected as follows, $x = t + Zv$ where

$$v = \arg\min_{v'} \mathtt{J}_{A,b}(t + Zv') = (Z^*AZ)^{-1}Z^*z. \tag{4.22}$$

Compared to the ordinary (4.14), or the inexact (4.17) SD steps, the optimization (4.22) is performed over vectors in the *wider* manifold $t + \mathrm{span}(Z)$, hence the name. The updated error may be written as an orthogonal projection as well, $f = (I - \mathcal{P}_{A,Z})c$, where the A–orthogonal projector $\mathcal{P}_{A,Z}$ is defined for a nonsingular Z as follows,

$$\mathcal{P}_{A,Z} = Z(Z^*AZ)^{-1}Z^*A. \tag{4.23}$$

The progress of the wide SD writes analogously,

$$\omega_{z,Z}^2 = \frac{\mathtt{J}_{A,b}(x)}{\mathtt{J}_{A,b}(t)} = \frac{\|f\|_A^2}{\|c\|_A^2} = 1 - \frac{(c, \mathcal{P}_{A,Z} c)_A}{(c,c)_A}. \tag{4.24}$$

In alternating tensor product algorithms, Z will play the role of the interface or frame matrix. Since the TT format is linear it follows from (4.5) that $\tilde{z} = Z_{\neq k} z^{(k)}$, that is $\tilde{z} \in \mathrm{span}(Z)$, if $Z = Z_{\neq k}$. Therefore, we need to relate the progress of the wide SD to the progress of the one-dimensional inexact method.

Lemma 4.3.3. If $\tilde{z} \in \mathrm{span}(Z)$ then $\omega_{z,Z} \leqslant \omega_{z,\tilde{z}}$, i.e. the progress (4.24) of the wide SD step (4.22) is not worse than the progress (4.19) of the inexact step (4.17).

Proof. For the complements of $\omega_{z,\tilde{z}}^2$ and $\omega_{z,Z}^2$ to one, it holds

$$\|\mathcal{P}_{A,\tilde{z}} c\|_A^2 = \|\mathcal{P}_{A,\tilde{z}} \mathcal{P}_{A,Z} c\|_A^2 \leqslant \|\mathcal{P}_{A,Z} c\|_A^2.$$

□

Corollary 4.3.4. If $\angle(Z; z) = \min_{\tilde{z} \in \mathrm{span}(Z)} \angle(\tilde{z}; z) \leqslant \theta < \pi/2$, then

$$\omega_{z,Z} \leqslant \frac{\tilde{\kappa} - 1}{\tilde{\kappa} + 1} = \tilde{\Omega} < 1, \qquad \tilde{\kappa} = \mathrm{cond}(A) \frac{1 + \sin\theta}{1 - \sin\theta}. \tag{4.25}$$

Proof. Apply Lemma 4.3.3 for $\tilde{z} \in \mathrm{span}(Z)$ s.t. $\angle(\tilde{z}; z) = \angle(Z; z) \leqslant \theta$, and estimate $\omega_{z,\tilde{z}}$ by (4.18). □

Remark 4.3.5. In practical computations we could expect that the progress of the wide SD is better than the one of the inexact SD, e.g. $\omega_{z,Z} \ll \omega_{z,\tilde{z}}$ if $\dim(Z) \gg 1$. Generally, however, the inequality in Lemma 4.3.3 is sharp. For example, $\omega_{z,Z} = \omega_{z,\tilde{z}}$ for $Z = \begin{bmatrix} \tilde{z} & s \end{bmatrix}$ with such s that $(\tilde{z}, s)_A = 0$ and $(c, s)_A = 0$.

Proof. For $(\tilde{z}, s)_A = 0$ it holds $\mathcal{P}_{A,Z} = \mathcal{P}_{A,\tilde{z}} + \mathcal{P}_{A,s}$, and the first condition $(\tilde{z}, s)_A = 0$ gives $\|\mathcal{P}_{A,Z} c\|_A^2 = \|\mathcal{P}_{A,\tilde{z}} c\|_A^2 + \|\mathcal{P}_{A,s} c\|_A^2$. The condition $(c, s)_A = 0$ annihilates the second term and proves the sharpness. □

4.3.3 AMEn: alternating optimization meets steepest descent

All versions of the steepest descent method presented above are *variational* w.r.t. the energy function: in each step, they seek a constrained energy minimizer, cf. (4.22). Inspired by the *minimal residual* (MR) algorithm (see e.g. [208]), we may call the steepest descent also the *minimal energy* (MEn) method.

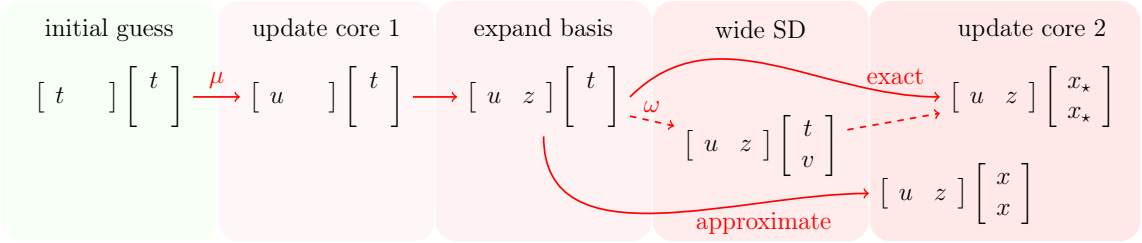
In this section we develop the algorithm for the linear system solution in the TT format in the DMRG fashion, equipped with the enrichment (4.13) by the wide steepest descent subspace. This motivates the name AMEn: *alternating minimal energy*.

The description and analysis in this section follow [52, 53]. Though it is possible to develop an algorithm with a *global* enrichment, which changes all TT blocks at a time (the so-called ALS($t + z$) method), we focus on the AMEn algorithm with the *local* enrichment (4.13), which appears to be more fast and accurate.

Algorithm 10 AMEn for SPD linear system $Ax = b$ (recurrent version)

Require: Initial guess $t = \tau(\{t^{(k)}\})$, accuracy ε or rank bounds $\rho_1, \dots, \rho_{d-1}$
Ensure: Updated vector $x = \tau(\{x^{(k)}\})$ with ranks $r'_k \leqslant r_k + \rho_k$, and $\mathtt{J}_{A,b}(x) < \mathtt{J}_{A,b}(t)$
1: Form $A_1 = T^*_{\neq 1} A T_{\neq 1}$, $b_1 = T^*_{\neq 1} b$, and solve $A_1 u^{(1)} = b_1$.
2: Let $u = \tau(u^{(1)}, t^{(2)}, \dots, t^{(d)})$ and $z = b - Au$.
3: Approximate $z \approx \tilde{z} = \tau(\{z^{(k)}\})$, s.t. $\|\tilde{z} - z\| \leqslant \varepsilon \|z\|$ or $r_k(\tilde{z}) < \rho_k$
4: Expand the basis $x^{(1)} := \begin{bmatrix} u^{(1)} & z^{(1)} \end{bmatrix}$, $t^{(2)} := \begin{bmatrix} t^{(2)} \\ 0 \end{bmatrix}$
5: Consider the $(d-1)$–dimensional system $A_{\geqslant 2} x^{(\geqslant 2)} = b_{\geqslant 2}$ given by (4.26)
6: **if** $d = 2$ **then**
7: Form $A_{\geqslant 2} x^{(\geqslant 2)} = b_{\geqslant 2}$ and solve it directly
8: **else**
9: Solve $A_{\geqslant 2} x^{(\geqslant 2)} = b_{\geqslant 2}$ by AMEn, obtain $x_{\geqslant 2} = \tau(x^{(2)}, \dots, x^{(d)})$
10: **end if**
11: **return** $x = \tau(x^{(1)}, x^{(2)}, \dots, x^{(d)})$

Figure 4.5: Illustration of the AMEn algorithm in two dimensions



The TT format was introduced as a recurrent generalization of the dyadic decomposition, and similarly we begin the presentation of the AMEn algorithm with the two-dimensional case. Note that in the enrichment (4.13) we may equivalently write $x^{(1)}$ instead of $x^{|1\rangle}$, since the left rank dimension vanishes.

The one-site DMRG Alg. 8 equipped with the enrichment (4.13) after Line 9 may be written as follows (we omit the orthogonalization steps, and assume that all frame matrices $X_{\neq p}$ are made orthogonal).

1. Copy $x^{(k)} = t^{(k)}$, $k = 1, \dots, d$.
2. Form $A_1 = X^*_{\neq 1} A X_{\neq 1} = T^*_{\neq 1} A T_{\neq 1}$, $b_1 = X^*_{\neq 1} b = T^*_{\neq 1} b$.
3. Solve $A_1 u^{(1)} = b_1$.
4. Replace $x^{(1)} := \begin{bmatrix} u^{(1)} & s^{(1)} \end{bmatrix}$, $x^{(2)} := \begin{bmatrix} t^{(2)} \\ 0 \end{bmatrix}$.
5. Solve (possible approximately) $A_{\geqslant 2} x^{(\geqslant 2)} = b_{\geqslant 2}$.

According to the discussion in the previous section, it is reasonable to choose the expansion $s^{(1)}$ related to the residual. At the moment $s^{(1)}$ is employed, the solution

reads $u = \tau(u^{(1)}, t^{(\geqslant 2)})$. The steepest descent analysis did not contain the local ALS update $t^{(1)} \to u^{(1)}$, so we have to overload the notation slightly, and denote the quantities, initial for the *wide SD step*, but not for the whole algorithm:

- approximation $u = \tau(u^{(1)}, t^{(\geqslant 2)})$,
- residual $z = b - Au$, and
- error $c = x_\star - u$.

Now if we computed the approximate residual, $\tilde{z} = \tau(z^{(1)}, z^{(\geqslant 2)}) \approx z$, it is natural to take its interface matrix as the wide SD basis, $Z = Z_1 = z^{|1\rangle} \otimes I_{n_2 \cdots n_d}$. To achieve this, we perform the enrichment using the first TT core of the residual, $s^{(1)} = z^{(1)}$. In this case, $\tilde{z} \in \operatorname{span}(Z)$, and Lemma 4.3.3 will take place if we show that the error is indeed projected onto a subspace containing Z.

The latter analysis can be conducted according to the scheme in Fig. 4.5. The *reduced* system $A_{\geqslant 2} x^{(\geqslant 2)} = b_{\geqslant 2}$ assembles as

$$\begin{aligned} A_{\geqslant 2} = X^*_{\neq\{2,\ldots,d\}} A X_{\neq\{2,\ldots,d\}}, \quad b_{\geqslant 2} = X^*_{\neq\{2,\ldots,d\}} b, \quad \text{where} \\ X_{\neq\{2,\ldots,d\}} \equiv X_1 = x^{|1\rangle} \otimes I_{n_2 \cdots n_d} \in \mathbb{C}^{(n_1 \cdots n_d) \times (r_1 n_2 \cdots n_d)}. \end{aligned} \tag{4.26}$$

Due to the construction of $x^{|1\rangle}$ it holds that $Z_1 \in \operatorname{span}(X_1)$. The new approximant writes (provided the reduced system is solved exactly)

$$x = \tau(x^{(1)}, x^{(\geqslant 2)}) = X_1 A^{-1}_{\geqslant 2} b_{\geqslant 2} = \mathcal{P}_{A,X_1} x_\star,$$

and the new error may be expanded as $f = x_\star - x = (I - \mathcal{P}_{A,X_1}) x_\star$. However, note that $u = U_1 t^{(\geqslant 2)}$ with $U_1 = u^{|1\rangle} \otimes I_{n_2 \cdots n_d} \in \operatorname{span}(X_1)$ as well. In particular, $\mathcal{P}_{A,X_1} u = u$, so the error projection is established,

$$f = (I - \mathcal{P}_{A,X_1}) x_\star - (I - \mathcal{P}_{A,X_1}) u = (I - \mathcal{P}_{A,X_1}) c.$$

As soon as $\tilde{z} \in \operatorname{span}(Z_1) \subset \operatorname{span}(X_1)$, this gives

$$\frac{\mathsf{J}_{A,b}(x)}{\mathsf{J}_{A,b}(u)} = \frac{\|f\|_A^2}{\|c\|_A^2} = 1 - \frac{(c, \mathcal{P}_{A,X_1} c)_A}{(c,c)_A} = \omega^2_{z,X_1}, \tag{4.27}$$

with the *a priori* bounds

$$\omega_{z,X_1} \leqslant \omega_{z,Z_1} \leqslant \omega_{z,\tilde{z}} \leqslant \tilde{\Omega} < 1$$

for $\varepsilon < 1$, cf. (4.18).

To proceed in higher dimensions, it is enough to repeat the previous considerations recurrently. Indeed, all steps except the solution of $A_{\geqslant 2} x^{(\geqslant 2)} = b_{\geqslant 2}$ involved only structured calculations in the TT format. The reduced system (4.26) can be assembled using one-block operations as well: the "do-nothing" identity $I_{n_2 \cdots n_d}$ leaves all TT cores $A^{(2)}, \ldots, A^{(d)}$ and $b^{(2)}, \ldots, b^{(d)}$ untouched, and only the first blocks $A^{(1)}$, $b^{(1)}$ experience the projection on $x^{|1\rangle}$. This yields the following TT representation of (4.26):

$$\begin{aligned} A_{\geqslant 2} &= \tau\left(\tau(\mathsf{A}^{<2}, A^{(2)}), A^{(3)}, \ldots, A^{(d)}\right), & \mathsf{A}^{<2}_{\gamma_1} &= \left(x^{|1\rangle}\right)^* A^{(1)}_{\gamma_1} x^{|1\rangle}, \\ b_{\geqslant 2} &= \tau\left(\tau(\mathsf{b}^{<2}, b^{(2)}), b^{(3)}, \ldots, b^{(d)}\right), & \mathsf{b}^{<2}_{\beta_1} &= \left(x^{|1\rangle}\right)^* b^{(1)}_{\beta_1}, \end{aligned} \tag{4.28}$$

where $\gamma_1 = 1, \dots, r_1(A)$, $\beta_1 = 1, \dots, r_1(b)$. Note that $\mathsf{b}^{<2} \in \mathbb{C}^{r_1(x) \times r_1(b)}$ and hence $\tau(\mathsf{b}^{<2}, b^{(2)}) \in \mathbb{C}^{r_1(x) n_2 \times r_2(b)}$ is feasible. Similarly, the first TT block of the matrix $A_{\geqslant 2}$ is effectively of the one-core size, so that (4.28) is a $(d-1)$-dimensional system with the same TT ranks $r_2, \dots, r_{d-1}$ of the initial data. Moreover, as soon as the orthogonality of $x^{|1\rangle}$ is ensured, the conditioning is preserved, $\mathrm{cond}(A_{\geqslant 2}) \leqslant \mathrm{cond}(A)$.

Therefore, the reduced system may be treated using the same AMEn method, which is summarized in Alg. 10. It also gives a recurrent flavor to the convergence analysis.

In the first step, we should also account for the fact that $A_{\geqslant 2} x^{(\geqslant 2)} = b_{\geqslant 2}$ is not solved exactly, since the solution $x^{(\geqslant 2)}$ is delivered by a recursive call to the AMEn and differs from the exact one $x_\star^{(\geqslant 2)} = A_{\geqslant 2}^{-1} b_{\geqslant 2}$. We expand

$$x_\star - x = x_\star - X_1 x_\star^{(\geqslant 2)} + X_1 x_\star^{(\geqslant 2)} - X_1 x^{(\geqslant 2)} = (I - \mathcal{P}_{A,X_1}) x_\star + X_1 (x_\star^{(\geqslant 2)} - x^{(\geqslant 2)}),$$

where the second term is A–orthogonal to the first one we have already estimated nearby (4.27). It gives

$$\frac{\|x_\star - x\|_A^2}{\|x_\star - u\|_A^2} = \omega_{z,X_1}^2 + \frac{\|X_1 (x_\star^{(\geqslant 2)} - x^{(\geqslant 2)})\|_A^2}{\|x_\star - u\|_A^2}.$$

By construction, the norm equivalence holds, $\|X_1 v\|_A = \|v\|_{A_{\geqslant 2}}$. Hence the previous equation rewrites as

$$\frac{\|f\|_A^2}{\|c\|_A^2} = \omega_{z,X_1}^2 + \frac{\|x_\star^{(\geqslant 2)} - x^{(\geqslant 2)}\|_{A_{\geqslant 2}}^2}{\|c\|_A^2} \cdot \frac{\|X_1 (x_\star^{(\geqslant 2)} - t^{(\geqslant 2)})\|_A^2}{\|x_\star^{(\geqslant 2)} - t^{(\geqslant 2)}\|_{A_{\geqslant 2}}^2}, \tag{4.29}$$

where $t^{(\geqslant 2)} = t^{(2,\dots,d)}$ is the initial guess in the reduced problem. Similarly to the $d = 2$ case it holds $X_1 x^{(\geqslant 2)} = \mathcal{P}_{A,X_1} x_\star$. At the same time

$$X_1 t^{(\geqslant 2)} = \tau(x^{(1)}, t^{(2)}, \dots, t^{(d)}) = u = \mathcal{P}_{A,X_1} u,$$

since $u \in \mathrm{span}(X_1)$. We plug $X_1 (x_\star^{(\geqslant 2)} - t^{(\geqslant 2)}) = \mathcal{P}_{A,X_1}(x_\star - u) = \mathcal{P}_{A,X_1} c$ into (4.29) and compare the result with (4.27). This gives the following estimate,

$$\frac{\mathtt{J}_{A,b}(x)}{\mathtt{J}_{A,b}(u)} = \omega_{z,X_1}^2 + \frac{\|\mathcal{P}_{A,X_1} c\|_A^2}{\|c\|_A^2} \frac{\|x_\star^{(\geqslant 2)} - x^{(\geqslant 2)}\|_{A_{\geqslant 2}}^2}{\|x_\star^{(\geqslant 2)} - t^{(\geqslant 2)}\|_{A_{\geqslant 2}}^2} = \omega_{z,X_1}^2 + (1 - \omega_{z,X_1}^2) \frac{\mathtt{J}_{A_{\geqslant 2}, b_{\geqslant 2}}(x^{(\geqslant 2)})}{\mathtt{J}_{A_{\geqslant 2}, b_{\geqslant 2}}(t^{(\geqslant 2)})}. \tag{4.30}$$

The last term in (4.30) quantifies in a recurrent way, since it is returned by the same AMEn algorithm. One half-sweep of AMEn can be seen, therefore, as a sequence of embedded reduced problems $A_{\geqslant k} x^{(\geqslant k)} = b_{\geqslant k}$, where

$$\begin{aligned} A_{\geqslant k} &= X_{<k}^* A X_{<k} \in \mathbb{C}^{(r_{k-1} n_k \cdots n_d) \times (r_{k-1} n_k \cdots n_d)}, & b_{\geqslant k} &= X_{<k}^* b, \\ X_{<k} &= x^{|<k\rangle} \otimes I_{n_k \cdots n_d} \in \mathbb{C}^{(n_1 \cdots n_d) \times (r_{k-1} n_k \cdots n_d)}, & X_{<1} &= I_{n_1 \cdots n_d}. \end{aligned} \tag{4.31}$$

For each reduced problem, the one-step quantities are defined as follows.

- Initial guess $t^{(\geqslant k)} = \tau(t^{(k)}, \dots, t^{(d)})$,

- ALS-updated solution $u_k = \tau(u^{(k)}, t^{(k+1)}, \dots, t^{(d)})$,
- new solution $x^{(\geqslant k)} = \tau(x^{(k)}, \dots, x^{(d)})$, and
- exact solution $x_\star^{(\geqslant k)} = A_{\geqslant k}^{-1} b_{\geqslant k}$,
- initial error $c_k = x_\star^{(\geqslant k)} - u_k$,
- residual $z_k = b_{\geqslant k} - A_{\geqslant k} u_k \approx \tilde{z}_k = \tau(z_k^{(k)}, \dots, z_k^{(d)})$, and
- new error $f_k = x_\star^{(\geqslant k)} - x^{(\geqslant k)}$.

The enrichment is performed using the first TT block of the reduced residual,

$$x^{|k\rangle} = \begin{bmatrix} u^{|k\rangle} & z_k^{|k\rangle} \end{bmatrix}, \quad t^{\langle k+1|} = \begin{bmatrix} t^{\langle k+1|} \\ 0 \end{bmatrix}.$$

Now, we may measure the progresses in each microstep, provided by both ALS and wide SD steps,

$$\mu_k^2 = \frac{\|x_\star^{(\geqslant k)} - u_k\|_{A_{\geqslant k}}^2}{\|x_\star^{(\geqslant k)} - t^{(\geqslant k)}\|_{A_{\geqslant k}}^2} \leqslant 1, \qquad \omega_k^2 = 1 - \frac{(c_k, \mathcal{P}_{A_{\geqslant k}, X_k} c_k)_{A_{\geqslant k}}}{(c_k, c_k)_{A_{\geqslant k}}} < 1, \tag{4.32}$$

where $X_k = x^{|k\rangle} \otimes I_{n_{k+1} \dots n_d}$, after the enrichment $x^{|k\rangle} = \begin{bmatrix} u^{|k\rangle} & z_k^{|k\rangle} \end{bmatrix}$ (and orthogonalization).

Lemma 4.3.6. In definitions set above, a single iteration (half-sweep) of Alg. 10 provides the progress

$$\omega_{\text{AMEn}}^2 = \frac{\|x_\star - x\|_A^2}{\|x_\star - t\|_A^2} = \sum_{k=1}^{d-1} \omega_k^2 \prod_{j=1}^{k-1} (1 - \omega_j^2) \prod_{j=1}^{k} \mu_j^2 = \phi_{(1:d)}^2. \tag{4.33}$$

Proof. For $d = 2$ we use (4.27) and the first definition in (4.32) to establish

$$\frac{\mathtt{J}_{A,b}(x)}{\mathtt{J}_{A,b}(t)} = \frac{\mathtt{J}_{A,b}(u)}{\mathtt{J}_{A,b}(t)} \frac{\mathtt{J}_{A,b}(x)}{\mathtt{J}_{A,b}(u)} = \mu_1^2 \omega_1^2,$$

that gives the base of the recursion. Now we suppose that (4.33) holds in $d-1$ dimensions and prove it recurrently for d dimensions. From (4.30) we see that

$$\frac{\mathtt{J}_{A,b}(x)}{\mathtt{J}_{A,b}(t)} = \mu_1^2 \left(\omega_1^2 + (1 - \omega_1^2) \frac{\mathtt{J}_{A_{\geqslant 2}, b_{\geqslant 2}}(x^{(\geqslant 2)})}{\mathtt{J}_{A_{\geqslant 2}, b_{\geqslant 2}}(t^{(\geqslant 2)})} \right) = \mu_1^2 \left(\omega_1^2 + (1 - \omega_1^2) \phi_{(2:d)}^2 \right).$$

Plugging the assumption $\phi_{(2:d)}^2 = \sum_{k=2}^{d-1} \omega_k^2 \prod_{j=2}^{k-1} (1 - \omega_j^2) \prod_{j=2}^{k} \mu_j^2$ into the last equation, we obtain (4.33) and complete the proof. □

In the same way as in the steepest descent, we had to denote exact error relations as ω_k and use them to establish the *a posteriori* rate (4.33). Now we have to bound all featuring quantities, as well as the total progress by uniform *a priori* estimates less than one.

In terms of definition (4.24), $\omega_k = \omega_{z_k,X_k}$, and since $\tilde{z}_k \in \mathrm{span}(Z_k) \subset \mathrm{span}(X_k)$, the upper bound (4.25) applies as follows,

$$\omega_k \leqslant \frac{\tilde{\kappa}_k - 1}{\tilde{\kappa}_k + 1} = \tilde{\Omega}_k < 1, \qquad \tilde{\kappa}_k = \mathrm{cond}(A_{\geqslant k}) \frac{1 + \sin\theta_k}{1 - \sin\theta_k}, \tag{4.34}$$

where $\theta_k = \angle(z_k; X_k) \leqslant \angle(z_k; \tilde{z}_k)$. If the accuracy criterion $\|z_k - \tilde{z}_k\| \leqslant \varepsilon \|z_k\|$ is enforced for all approximation steps (Line 3) in Alg. 10, then $\sin\theta_k \leqslant \varepsilon$ is known *a priori*. If the rank bound criterion is applied, θ_k is estimated by the delivered relative accuracy of the approximation $\angle(z_k; \tilde{z}_k)$. Since

$$A_{\geqslant k+1} = X^*_{<k+1} A X_{<k+1} = (x^{|k\rangle} \otimes I_{n_{k+1} \cdots n_d})^* A_{\geqslant k} (x^{|k\rangle} \otimes I_{n_{k+1} \cdots n_d}), \tag{4.35}$$

the condition numbers $\mathrm{cond}(A_{\geqslant k})$ are related as follows

$$\mathrm{cond}(A) = \mathrm{cond}(A_{\geqslant 1}) \geqslant \ldots \geqslant \mathrm{cond}(A_{\geqslant k-1}) \geqslant \mathrm{cond}(A_{\geqslant k}) \geqslant \ldots \geqslant \mathrm{cond}(A_{\geqslant d}). \tag{4.36}$$

Plugging $\theta_k \leqslant \theta = \max_k \angle(z_k; \tilde{z}_k)$ and $\mathrm{cond}(A_{\geqslant k}) \leqslant \mathrm{cond}(A)$ into (4.34), we obtain

$$\omega_k \leqslant \tilde{\Omega}_k = \frac{\tilde{\kappa}_k - 1}{\tilde{\kappa}_k + 1} \leqslant \frac{\tilde{\kappa} - 1}{\tilde{\kappa} + 1} = \tilde{\Omega} < 1, \qquad \tilde{\kappa} = \mathrm{cond}(A) \frac{1 + \sin\theta}{1 - \sin\theta}, \tag{4.37}$$

which gives a uniform upper bound for all ω_k.

Theorem 4.3.7. AMEn Alg. 10 is convergent if the approximation error allowed in Line 3 satisfies $\theta = \max_{k=1,\ldots,d-1} \angle(z_k; \tilde{z}_k) < \pi/2$. The convergence rate (4.33) of a single iteration (half-sweep) is bounded from above, s.t. the following inequality holds

$$1 - \phi^2_{(1:d)} \geqslant (1 - \tilde{\Omega}^2)^{d-1}, \qquad \tilde{\Omega} = \frac{\tilde{\kappa} - 1}{\tilde{\kappa} + 1}, \qquad \tilde{\kappa} = \mathrm{cond}(A) \frac{1 + \sin\theta}{1 - \sin\theta}. \tag{4.38}$$

Proof. Note that $\phi_{(1:d)}$ in (4.33) is monotonous for $0 \leqslant \omega_k \leqslant \tilde{\Omega}$ and $0 \leqslant \mu_k \leqslant 1$, $k = 1, \ldots, d-1$. By plugging the upper bounds $\mu_k = 1$ and $\omega_k = \tilde{\Omega}$ from (4.37) into (4.33), we prove (4.38). □

Before highlighting the practical aspects of efficient implementation of the AMEn algorithm, we compare it with another type of adaptive one-site DMRG scheme.

4.3.4 Enrichment versus the 1.5-site DMRG

Both the two-site DMRG and AMEn are rank-adaptive algorithms. However, the modification of ranks is performed by different techniques. The DMRG method uses the two-dimensional separation of variables, which may return any rank up to the row or column sizes, e.g. $r'_k \leqslant r_{k-1} n_k$ or $r'_k \leqslant n_{k+1} r_{k+1}$. Thus, the rank may be multiplicatively increased by a factor n. The projection subspaces (such as the frame

matrix $X_{\neq k}$) are generally replaced by the new ones, since the SVD distributes the solution correction over all singular vectors.

In AMEn, the rank growth is *additive*, i.e. $r'_k \leqslant r_k + r_k(\tilde{z})$ due to the expansion of TT cores. The same holds for the interface matrices: after the enrichment (4.13), the k-th block is not approximated until the next sweep, so that both $u^{(k)}$ and $z^{(k)}$ belong to $\operatorname{span}(x^{(\leqslant k)})$ exactly.

This consideration will help to understand the differences between AMEn and the rank-adaptive one-site method from [240], the so-called *corrected* one-site DMRG, developed in quantum physics. The latter algorithm may be explained as follows: when the current TT block is updated (cf. Line 8 in Alg. 8), we assemble the Gram matrix $G^{(k)} = u^{|k\rangle} \left(u^{|k\rangle}\right)^*$, and perturb it with some correction, $\tilde{G}^{(k)} = G^{(k)} + aH^{(k)}$, where $a > 0$ is some (heuristic) weight, and $H^{(k)} = p^{|k\rangle} \left(p^{|k\rangle}\right)^*$ is the Gram matrix of the perturbation vector, discussed below. After that, the modified Gram matrix is decomposed back to the truncated eigenvalue decomposition, $\tilde{G}^{(k)} \approx x^{|k\rangle} \operatorname{diag}(\tilde{\lambda}) \left(x^{|k\rangle}\right)^*$, returning the (orthogonal) eigenvectors to the k-th TT block of the solution.

It is easy to see that though $\operatorname{rank}(G^{(k)}) = r_k$, and any filtering of eigenvalues in $G^{(k)} = U \operatorname{diag}(\lambda) U^*$ with a positive threshold will give $r'_k = r_k$, it is no more true for $\tilde{G}^{(k)}$, and the ε-filtering of $\tilde{\lambda}$ may return a different rank value r'_k.

For the purpose of approximation, the eigenvalue decomposition of Gram matrices is equivalent to the singular value decomposition of the *concatenated* columns of u and p,

$$\begin{bmatrix} u^{|k\rangle} & \sqrt{a} p^{|k\rangle} \end{bmatrix} \approx x^{|k\rangle} \operatorname{diag}(\sigma) V^*, \quad \sigma^2 = \tilde{\lambda}. \tag{4.39}$$

This looks very similar to the enrichment (4.13). The difference is in the approximate equality, while the enrichment, followed by the orthogonalization

$$\begin{bmatrix} u^{|k\rangle} & s^{|k\rangle} \end{bmatrix} \begin{bmatrix} t^{\langle k+1|} \\ 0 \end{bmatrix} = x^{|k\rangle} \cdot R \begin{bmatrix} t^{\langle k+1|} \\ 0 \end{bmatrix} = x^{|k\rangle} \begin{bmatrix} R_u t^{\langle k+1|} \\ 0 \end{bmatrix}, \quad \left(x^{|k\rangle}\right)^* x^{|k\rangle} = I,$$

is exact. Recalling the discussion in the beginning of this subsection, we formulate the first difference between AMEn and corrected one-side DMRG: AMEn does not affect the solution components by the enrichment, while DMRG *mixes* the information from the solution and the expansion in the updated singular vectors. Therefore, a wise tuning of the weight a is crucial: if a is small, the correction will be eliminated by the ε-thresholding in the SVD; if a is too large, the solution will be significantly perturbed.

As a correction vector, [240] suggests to use a surrogate of the Krylov vector. Denote $p = A_{\geqslant k} u_k = \tau(p^{(k)}, \ldots, p^{(d)})$, where according to (2.12), $p^{(k)} = \left[p^{(k)}_{\eta_k}(\overline{\alpha_{k-1} i_k})\right] \in \mathbb{C}^{r_{k-1} n_k \times r_k r_k(A)}$, and $\eta_k = \overline{\alpha_k, \gamma_k}$. Now the perturbation takes $p^{|k\rangle} = p^{(k)}$ directly, without the right-orthogonalization of $p^{(k+1)}, \ldots, p^{(d)}$. Though it appears to be sufficient in practical computations of the ground states of spin chains, it is possible to suggest a matrix $A_{\geqslant k}$ such that $p^{(k)}$ will give a very poor information on p, see [54] for more details.

Finally, the corrected one-site DMRG can be seen as a special case of the two-site DMRG. The left-hand side of (4.39) has the sizes $r_{k-1} n_k \times (1 + r_k(A)) r_k$. Comparing this with the two-site separation (4.12), we may introduce an additional variable $b =$

$1, \ldots, 1 + r_k(A)$, such that

$$\begin{bmatrix} u^{|k\rangle} & \sqrt{a}p^{|k\rangle} \end{bmatrix} = \left[\hat{x}^{(k)}(i_k, b)_{\alpha_{k-1},\alpha_k}\right]$$

may be seen as a superblock, and pretend that we seek a solution of the form

$$\hat{x} = [\hat{x}(i_1, \ldots, i_k, b, i_{k+1}, \ldots, i_d)] = \tau(x^{(1)}, \ldots, x^{(k-1)}, \hat{x}^{(k)}, x^{(k+1)}, \ldots, x^{(d)}). \quad (4.40)$$

The corresponding algorithm can be called a *"1.5-site"* DMRG, since the size of $\hat{x}^{(k)}$ is larger than the size of $x^{(k)}$, but smaller than the size of the two-site superblock $x^{(k,k+1)}$ for large n.

The representation (4.40) mirrors the so-called *block* TT format [48], since we may say that $\hat{x}$ encapsulates several vectors enumerated by the index b,

$$\hat{x} = [\hat{x}_b]_{b=1}^{1+r_k(A)}, \quad \text{where} \quad \hat{x}_{b+1} = \sqrt{a}p_b, \quad b = 1, \ldots, r_k(A)$$

are the perturbation vectors, and $\hat{x}_1 = x$, our sought solution.

By means of the SVD (4.39), the enumerator b may be replaced from $x^{(k)}$ to $x^{(k+1)}$ or vice versa, i.e. moved along the tensor train, changing the order of b and $i_1, \ldots, i_d$. This technique was used in [48] to store and compute several extreme eigenvectors simultaneously in the common TT format. The same procedure takes place in the corrected one-site DMRG from [240]. However, when we apply (4.39) to the set of extreme eigenvectors, they are either normalized, and the truncation error distributes fairly among the components, or scaled by physically justified weights. In the corrected one-site DMRG, neither the surrogates p are meaningful for something except the technical purposes, nor the weight a can be sensibly suggested.

4.4 Practical aspects of DMRG and AMEn algorithms

Algorithms 8, 9 and 10 contain the most important steps, required for analysis and getting the idea across. In this section we focus on the implementation details that improve the performance and make the practical methods efficient. The most powerful insight is the subsequent sweeping along the tensor train, $k = 1, 2, \ldots, d$, since it allows to maintain the whole complexity linear in the problem dimension d, reusing some data and performing only local (one- or two-block) calculations.

Since many operations will involve four-dimensional tensors, we introduce a couple of new reshapes.

Definition 4.4.1. Given a tensor $X^{(k)} = \left[X^{(k)}(\alpha, i, j, \beta)\right] \in \mathbb{C}^{p\times m\times n\times q}$. Denote the following matricisations.

- Outer folding $X^{\langle k\rangle} \in \mathbb{C}^{pm\times nq}$, $\quad X^{\langle k\rangle}(\overline{\alpha i}, \overline{j\beta}) = X^{(k)}(\alpha, i, j, \beta)$.
- Inner folding $X^{\rangle k\langle} \in C^{mn\times qp}$, $\quad X^{\rangle k\langle}(\overline{ij}, \overline{\beta\alpha}) = X^{(k)}(\alpha, i, j, \beta)$.

4.4.1 Computation of local systems

In each optimization step we create and solve the local system $A_k u^{(k)} = b_k$ by (4.7). As shown in Fig. 4.3, it can be assembled from the TT cores of A, x and b in $\mathcal{O}(d)$ time for each k, while for the whole sweep the straightforward complexity is $\mathcal{O}(d^2)$.

However, only small corrections (similar to (4.35)) are required to update the local systems between microsteps $k = 1, 2, \ldots, d$. Indeed, we saw in (4.28) that one reduction step involved effectively only one TT block. Generally we may compute the left reductions $\mathsf{A}^{<k}$ as follows. We are given $\mathsf{A}^{<k} \in \mathbb{C}^{r_{k-1} \times r_{k-1} \times r_{k-1}(A)}$, and may consider its either three- or four-dimensional reshape $\mathsf{A}^{\langle <k|} = \mathsf{A}^{\langle <k\rangle} \in \mathbb{C}^{r_{k-1} \times r_{k-1} r_{k-1}(A)}$. Then compute:

$$\begin{array}{lrcll} 1. & p^{\langle k\rangle} & = & \left(x^{\langle k|}\right)^* \mathsf{A}^{\langle <k|} & \in \mathbb{C}^{n_k r_k \times r_{k-1} r_{k-1}(A)}; \\ 2. & q^{\langle k\rangle} & = & p^{\rangle k\langle} A^{\langle k\rangle} & \in \mathbb{C}^{r_k r_{k-1} \times n_k r_k(A)}; \\ 3. & \mathsf{A}^{\rangle <k+1\langle} & = & \left(x^{|k\rangle}\right)^\top q^{\rangle k\langle} & \in \mathbb{C}^{1 \cdot r_k \times r_k(A) r_k}, \end{array} \tag{4.41}$$

followed by the "inverse" reshape $\mathsf{A}^{\langle <k+1\rangle} \in \mathbb{C}^{r_k \cdot 1 \times r_k r_k(A)}$, which permutes the dimensions to the initial order, as in $\mathsf{A}^{<k}$. Note that $A^{\langle k\rangle}$ in the second line is just the reshape of the matrix TT block $A^{(k)}$ according to Def. 4.4.1. In the third line, we write explicitly a dummy dimension of size 1 to emphasize how exactly the four-dimensional reshapes are applied.

Similar recurrence takes place for the right reductions $\mathsf{A}^{>k} \in \mathbb{C}^{r_k(A) \times r_k \times r_k}$.

$$\begin{array}{lrcll} 1. & p^{\langle k\rangle} & = & x^{|k\rangle} \left(\mathsf{A}^{|>k\rangle}\right)^\top & \in \mathbb{C}^{r_{k-1} n_k \times r_k(A) r_k}; \\ 2. & q^{\langle k\rangle} & = & A^{\langle k\rangle} p^{\rangle k\langle} & \in \mathbb{C}^{r_{k-1}(A) n_k \times r_k r_{k-1}}; \\ 3. & \mathsf{A}^{\rangle >k-1\langle} & = & \bar{x}^{\langle k|} q^{\rangle k\langle} & \in \mathbb{C}^{r_{k-1} \cdot 1 \times r_{k-1} r_{k-1}(A)}, \end{array} \tag{4.42}$$

and $\mathsf{A}^{\langle >k-1\rangle} \in \mathbb{C}^{r_{k-1}(A) r_{k-1} \times 1 \cdot r_{k-1}}$ recovers the correct order of dimensions.

For the initialization purposes, we take $\mathsf{A}^{<1} = \mathsf{A}^{>d} = 1$. Now the one-site local system (4.7) may be assembled as $A_k = \tau(\mathsf{A}^{<k}, A^{(k)}, \mathsf{A}^{>k})$, and the two-site system (4.10) writes $A_{k,k+1} = \tau(\mathsf{A}^{<k}, A^{(k)}, A^{(k+1)}, \mathsf{A}^{>k+1})$. That is, Figure 4.3 of the one-site system can be detailed as shown in Fig. 4.6.

Moreover, in an iterative local solver, the product $w^{(k)} = A_k v^{(k)}$ for any vector $v^{(k)} \in \mathbb{C}^{r_{k-1} n_k r_k}$ may be computed efficiently in the following structured form, see also [51].

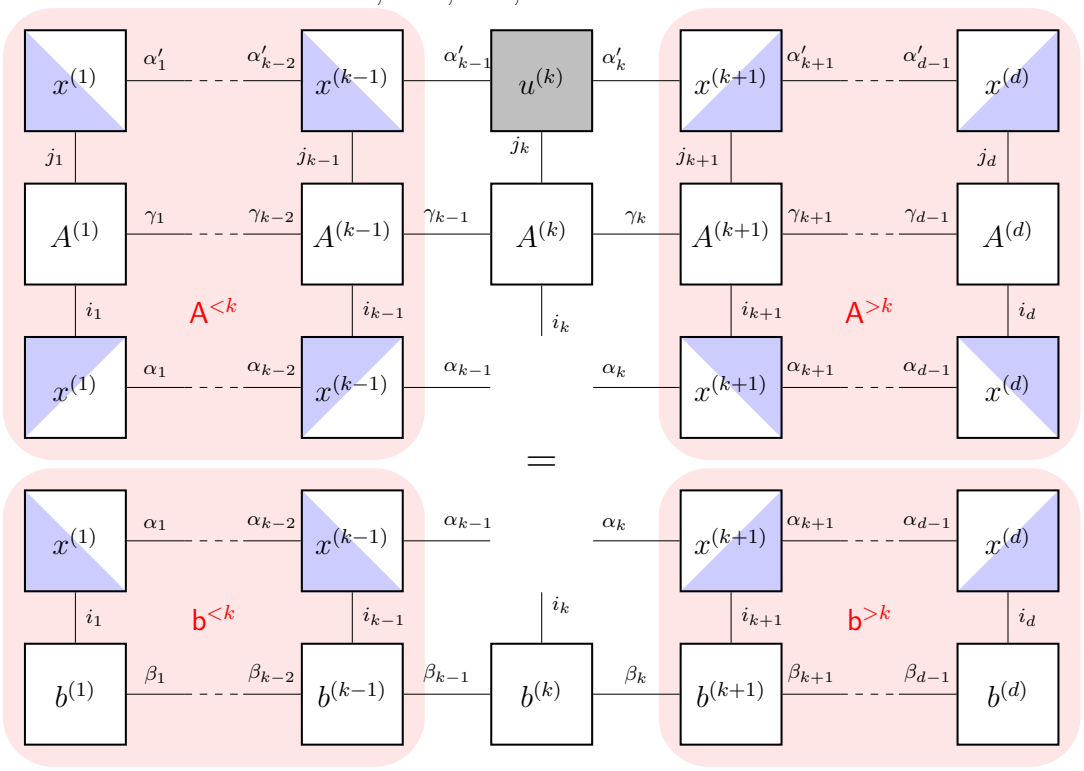
$$\begin{array}{lrcll} 1. & p^{\langle k\rangle} & = & v^{|k\rangle} \left(\mathsf{A}^{|>k\rangle}\right)^\top & \in \mathbb{C}^{r_{k-1} n_k \times r_k(A) r_k}; \\ 2. & q^{\langle k\rangle} & = & A^{\langle k\rangle} p^{\rangle k\langle} & \in \mathbb{C}^{r_{k-1}(A) n_k \times r_k r_{k-1}}; \\ 3. & w^{\langle k|} & = & \mathsf{A}^{\langle <k|} \left(q^{\rangle k\langle}\right)^\top & \in \mathbb{C}^{r_{k-1} \times n_k r_k}. \end{array} \tag{4.43}$$

The complexity of this procedure, as well as of the reductions (4.41),(4.42) can be easily estimated from the sizes of matrix products,

$$\texttt{work}\,[(4.41), (4.42), (4.43)] = \mathcal{O}(n r^3 r(A)) + \mathcal{O}(n^2 r^2 r(A)^2). \tag{4.44}$$

The second term arises from the multiplication with $A^{\langle k\rangle}$, and can be performed in $\mathcal{O}(n r^2 r(A)^2)$ operations if the TT block $A^{(k)}(i_k, j_k)$ is sparse w.r.t. the mode indices i_k, j_k. Thus, the total complexity becomes *linear* in the mode size.

Figure 4.6: The linear system $A_k u^{(k)} = b_k$ (4.7) assembled from the TT cores $A^{(k)}$, $b^{(k)}$, and the interface reductions $\mathsf{A}^{<k}$, $\mathsf{A}^{>k}$, $\mathsf{b}^{<k}$, $\mathsf{b}^{>k}$.



For the computations of the right-hand side reductions $\mathsf{b}^{<k} \in \mathbb{C}^{r_{k-1} \times r_{k-1}(b)}$, $\mathsf{b}^{>k} \in \mathbb{C}^{r_k(b) \times r_k}$ (cf. (4.28)), Equations (4.41) and (4.42) can be reused, by substituting x with b in the first lines, and skipping the second lines. In fact, $\mathsf{b}^{<k+1}$ are nothing else than the intermediate matrices s_k in the TT scalar product Algorithm 1.

4.4.2 Truncation of the solution

The AMEn algorithm increases the TT ranks of the solution in each step. Sometimes, however, it is useful to reduce them, since the wide SD may deliver a sub-optimal solution for the given ranks. To do this, we may apply the TT-SVD rounding Algorithm 4 to the solution after each microstep. Since the interface matrices are orthogonal, the SVD step may be applied at low cost: we just add the truncation of $u^{|1\rangle}$ between Lines 1 and 2 of Alg. 10. More detailed description is provided in the non-recurrent AMEn versions below.

The convergence analysis in Section 4.3.3 does not depend on this truncation. Indeed, since the residual is computed after the perturbation of the solution, we may relate the truncation error to the initial guess, similarly to Lemma 4.1.1. The error overhead $\Omega/(1-\Omega)$, as provided by Lemma 4.1.1 and Theorem 4.3.7, may look pessimistic, but the actual a posteriori convergence rate (as in Lemma 4.3.6) is usually much smaller than Ω or $\tilde{\Omega}$, and the effect of the solution truncation can be neglected.

4.4.3 Approximation of the residual: SVD method

In Line 3 of the AMEn Alg. 10, the formal complexity of the TT rounding procedure is $\mathcal{O}((d-k+1)n(rr(A)+r(b))^3)$. However, we need only one TT core of the approximate residual to insert into the enriched block. Let us see how we can write the algorithm such that the complexity of each microstep does not grow with d.

According to the TT arithmetics Sec. 2.1.5, the first TT block of the *exact* residual $z_k = b_{\geqslant k} - A_{\geqslant k} u_k$ writes as follows,

$$(\hat{z}_k^{(k)})_{\eta_k} = \begin{cases} \tau(\mathsf{b}^{<k}, b^{(k)}_{\beta_k}), & \eta_k = 1, \dots, r_k(b), \\ -\tau(\mathsf{A}^{<k}, A^{(k)}_{\gamma_k}) u^{|k\rangle}_{\alpha_k}, & \eta_k = r_k(b) + \overline{\alpha_k \gamma_k} = r_k(b)+1, \dots, r_k(b) + r_k r_k(A), \end{cases} \tag{4.45}$$

such that $z_k^{(k)} \in \mathbb{C}^{r_{k-1} n_k \times (r_k(b) + r_k r_k(A))}$, and the rest blocks read

$$\begin{aligned} \hat{z}_k^{(d)}(i_d) &= \begin{bmatrix} b^{(d)}(i_d) \\ \sum_{j_d} A^{(d)}(i_d, j_d) \otimes t^{(d)}(j_d) \end{bmatrix}, \\ \hat{z}_k^{(p)}(i_k) &= \begin{bmatrix} b^{(p)}(i_k) & \\ & \sum_{j_k} A^{(p)}(i_k, j_k) \otimes t^{(p)}(j_k) \end{bmatrix}, \quad \text{for} \quad p = k+1, \dots, d-1. \end{aligned} \tag{4.46}$$

We see that the updated solution enters only the k-th block, while all the rest ones (4.46) depend on the previous iterate t. Hence, the blocks $\hat{z}_k^{(p)} = \hat{z}^{(p)}$ are the same for all $k < p$, and may be precomputed once before the AMEn sweep. The same concerns the (right) orthogonalizations of $\hat{z}^{(p)}$, which are performed before the sweep, constituting the first part of the TT-SVD Alg. 4. As soon as all $d-1$ LQ factors are stored during the orthogonalization, they may be later multiplied with $\hat{z}_k^{(k)}$, and only one additional SVD delivers the approximate block $z_k^{(k)}$.

We summarize the whole procedure in the non-recurrent counterpart of Alg. 10, the AMEn_{SVD} Algorithm 11. The complexity of each LQ or SVD step for the residual is $\mathcal{O}(n(rr(A)+r(b))^3)$, i.e. $\mathcal{O}(nr^6)$ if the ranks of the matrix and the solution are comparable.

4.4.4 Approximation of the residual: ALS method

To reduce the complexity w.r.t. TT ranks, we can substitute the TT-SVD by the one-site DMRG (or ALS) Alg. 8, which is applied to the approximation problem $\mathtt{J}_{I,z}(\tilde{z}) = \|z - \tilde{z}\|^2 \to \min$.

The ALS algorithm, tuned for the particular form of $z = b - Au$ (4.46), can be written with $\mathcal{O}(d)$ complexity featuring additional reductions, similar to (4.41), (4.42). We assume that the initial guess for the residual is given in the TT format, $\tilde{z} = \tau(\{z^{(k)}\})$, with ranks $\boldsymbol{\rho} = (\rho_1, \dots, \rho_{d-1})$, as well as the current matrix $A = \tau(\{A^{(k)}\})$ and the solution $x = \tau(\{x^{(k)}\})$. Now we introduce $\mathsf{A}_z^{<k} \in \mathbb{C}^{\rho_{k-1} \times r_{k-1} \times r_{k-1}(A)}$, constituting parts of the projections $Z^*_{\neq k} A X_{\neq k}$, and evaluate them recurrently as follows.

$$\begin{aligned} &1.\quad p^{\langle k\rangle} &&= \left(z^{\langle k|}\right)^* \mathsf{A}_z^{\langle <k|} &&\in \mathbb{C}^{n_k \rho_k \times r_{k-1} r_{k-1}(A)}; \\ &2.\quad q^{\langle k\rangle} &&= p^{\rangle k\langle} A^{\langle k\rangle} &&\in \mathbb{C}^{\rho_k r_{k-1} \times n_k r_k(A)}; \\ &3.\quad \mathsf{A}_z^{\rangle <k+1\langle} &&= \left(x^{|k\rangle}\right)^\top q^{\rangle k\langle} &&\in \mathbb{C}^{1 \cdot r_k \times r_k(A) \rho_k}, \end{aligned} \tag{4.47}$$

Algorithm 11 $\mathrm{AMEn}_{\mathrm{svd}}$, one iteration

Require: Initial guess $t = \tau(\{t^{(k)}\})$ in the TT format (2.9), accuracy ε or rank bound r for the solution, accuracy ϵ or rank bound ρ for the residual.
Ensure: Updated vector $x = \tau(\{x^{(k)}\})$ s.t. $r' \leqslant r + \rho$
1: Copy $x^{(k)} = t^{(k)}$, $k = 1, \ldots, d$.
2: Compute the residual $z = \tau(\{\hat{z}^{(p)}\})$ by (4.46).
3: Initialize $\mathsf{A}^{>d} = \mathsf{A}^{<1} = \mathsf{b}^{>d} = \mathsf{b}^{<1} = 1$.
4: **for** $k = d, \ldots, 2$ **do** {Orthogonalization and reduction}
5: Find LQ decomposition $x^{\langle k|} = LQ$, $QQ^* = I$.
6: Replace $x^{\langle k|} := Q$, and $x^{|k-1\rangle} := x^{|k-1\rangle}L$.
7: Compute right reductions $\mathsf{A}^{>k-1}$, $\mathsf{b}^{>k-1}$ by (4.42).
8: Find LQ decomposition $\hat{z}^{\langle k|} = L_{k-1}Q$, $QQ^* = I$.
9: Replace $\hat{z}^{|k-1\rangle} := \hat{z}^{|k-1\rangle}L_{k-1}$, store L_{k-1}.
10: **end for**
11: **for** $k = 1, \ldots, d$ **do** {Optimization over TT cores}
12: Form $b_k = \tau(\mathsf{b}^{<k}, b^{(k)}, \mathsf{b}^{>k})$, and {optional} $A_k = \tau(\mathsf{A}^{<k}, A^{(k)}, \mathsf{A}^{>k})$.
13: Solve $A_k u^{(k)} = b_k$. {Take $x^{(k)}$ as an initial guess, use (4.43)}
14: Compute SVD $u^{|k\rangle} \approx U\Sigma V^*$ s.t. $\left\|u^{|k\rangle} - U\Sigma V^*\right\| \leqslant \varepsilon \left\|u^{|k\rangle}\right\|$ or $\operatorname{rank}(\Sigma) \leqslant r$, replace $u^{|k\rangle} := U\Sigma V^*$. {Optional}
15: **if** $k \neq d$ **then** {Enrichment, orthogonalization and reduction}
16: Find (4.45) and SVD $\hat{z}_k^{|k\rangle}L_k \approx z_k^{|k\rangle}\Sigma V^*$, s.t. $\left\|\hat{z}_k^{|k\rangle}L_k - z_k^{|k\rangle}\Sigma V^*\right\| \leqslant \epsilon \left\|\hat{z}_k^{|k\rangle}L_k\right\|$ or $\operatorname{rank}(\Sigma) \leqslant \rho$.
17: Expand $x^{|k\rangle} := \begin{bmatrix} U & z_k^{|k\rangle} \end{bmatrix}$, $x^{\langle k+1|} := \begin{bmatrix} x^{\langle k+1|} \\ 0 \end{bmatrix}$.
18: Find QR decomposition $x^{|k\rangle} = QR$, $Q^*Q = I$.
19: Replace $x^{|k\rangle} := Q$, and $x^{\langle k+1|} := Rx^{\langle k+1|}$.
20: Compute left reductions $\mathsf{A}^{<k+1}$, $\mathsf{b}^{<k+1}$ by (4.41).
21: **end if**
22: **end for**
23: **return** $x = \tau(x^{(1)}, \ldots, x^{(d)})$.

and the "inverse" reshape $\mathsf{A}_z^{\langle <k+1\rangle} \in \mathbb{C}^{\rho_k \cdot 1 \times r_k r_k(A)}$ permutes the dimensions to the same order as in $\mathsf{A}_z^{<k}$.

The right reductions $\mathsf{A}_z^{>k} \in \mathbb{C}^{r_k(A) \times \rho_k \times r_k}$ we compute analogously.

$$\begin{array}{lrcll} 1. & p^{\langle k\rangle} & = & x^{|k\rangle}\left(\mathsf{A}_z^{|>k\rangle}\right)^\top & \in \mathbb{C}^{r_{k-1}n_k \times r_k(A)\rho_k}; \\ 2. & q^{\langle k\rangle} & = & A^{\langle k\rangle}p^{\rangle k\langle} & \in \mathbb{C}^{r_{k-1}(A)n_k \times \rho_k r_{k-1}}; \\ 3. & \mathsf{A}_z^{\rangle >k-1\langle} & = & \bar{z}^{\langle k|}q^{\rangle k\langle} & \in \mathbb{C}^{\rho_{k-1}\cdot 1 \times r_{k-1}r_{k-1}(A)}, \end{array} \tag{4.48}$$

followed by the "inverse" reshape $\mathsf{A}_z^{\langle >k-1\rangle} \in \mathbb{C}^{r_{k-1}(A)\rho_{k-1} \times 1 \cdot r_{k-1}}$. By substituting x with b, and omitting the second lines in (4.47), (4.48), one derives the formulae for $\mathsf{b}_z^{<k} \in \mathbb{C}^{\rho_{k-1} \times r_{k-1}(b)}$ and $\mathsf{b}_z^{>k} \in \mathbb{C}^{r_k(b) \times \rho_k}$.

The microstep update in Alg 8 (Line 8) reduces to $z^{(k)} = b_{z_k}$, and taking into

account the parts (4.47), (4.48), and the block structure (4.46), writes

$$z^{(k)} = \tau(\mathsf{b}_z^{<k}, b^{(k)}, \mathsf{b}_z^{>k}) - \tau(\mathsf{A}_z^{<k}, A^{(k)}, \mathsf{A}_z^{>k})u^{(k)}, \tag{4.49}$$

where the structured computation (4.43) may be employed for the latter MatVec. Comparing the sizes of $\mathsf{A}_z, \mathsf{b}_z$ with A, b, we may estimate the complexity counterpart of (4.44),

$$\texttt{work}\,[(4.47), (4.48)] = \mathcal{O}(nr^2\rho r(A)) + \mathcal{O}(nr\rho^2 r(A)) + \mathcal{O}(n^2 r\rho r(A)^2).$$

If the TT ranks of the residual ρ_k are kept significantly smaller than the ranks of the matrix and the solution, the complexity of the ALS step is cubic w.r.t. the characteristic ranks, $\mathcal{O}(n^2r^3)$, which is essentially smaller than $\mathcal{O}(nr^6)$ of the AMEn_{svd} algorithm.

It appears that even one microstep (4.49) between the updates of x is enough to maintain a sufficiently good approximation $\tilde{z}$. It allows to conduct the AMEn for the solution $Ax = b$ and the ALS for the approximation $\tilde{z} \approx z$ simultaneously, synchronizing the steps in both methods, as shown in the AMEn_{als} Algorithm 12.

Note that the reduced residual approximation $\tilde{z}_k \approx z_k$, in particular the enrichment block $z_k^{(k)}$ in Line 16, is computed using the interface blocks $z^{(p)}$, $p = k+1, \dots, d$, shared with the global residual. This mirrors the reuse of the LQ factors in the AMEn_{svd} Alg. 11. However, as soon as the convergence of the one-site DMRG is not established globally, nothing can be rigorously said about the approximation quality of $z^{(p)}$ for either z or z_k. In practice, nevertheless, a significant speed-up overweights the lack of guarantee for this heuristics.

4.4.5 AMEn and DMRG for the QTT-Tucker format

In the same way as in the QTT-Tucker rounding section 2.2.5, the alternating optimization methods may be written, applying their TT counterparts to the Tucker core and extended factors, cf. Alg. 5. The DMRG Algorithms 8,9, as well as the AMEn algorithms 11,12 should be only modified to return the reductions $\mathsf{A}^{<k}, \mathsf{A}^{>k}, \mathsf{b}^{<k}, \mathsf{b}^{>k}$ in addition to the solution. Indeed, suppose an optimization over the k-th extended factor (2.21) is finished. Then the last reduction $\mathsf{A}_{\gamma_k}^{f(k,<1)} = (x^{f(k)})^* A_{\gamma_k}^{f(k)} x^{f(k)} \in \mathbb{C}^{R_k \times R_k}$, $\gamma_k = 1, \dots, R_k(A)$, stands for the Tucker factor of the matrix, required for the optimization over the Tucker core. Vice versa, the reductions computed during the core update, e.g. $\mathsf{A}^{c(<k)} \in \mathbb{C}^{r_{k-1} \times r_{k-1} \times r_{k-1}(A)}$, will be used as the last TT blocks of the matrix, required for the optimization over the extended factor. The diagrammatic notations of the corresponding calculations can be seen in [44]. However, a detailed description like in Alg. 11, 12 would be too lengthy, and we prefer to omit it here.

Algorithm 12 $\mathrm{AMEn}_{\mathrm{als}}$, one iteration

Require: Initial guess $t = \tau(\{t^{(k)}\})$ in the TT format (2.9), accuracy ε or rank bound r for the solution, initial guess $\tilde{z} = \tau(\{z^{(k)}\})$ for the residual.

Ensure: Updated vectors $x = \tau(\{x^{(k)}\})$ s.t. $r' \leqslant r + \rho$, and $\tilde{z} = \tau(\{z^{(k)}\})$.

Copy $x^{(k)} = t^{(k)}$, $k = 1, \ldots, d$.
Initialize $\mathsf{A}^{>d} = \mathsf{A}^{<1} = \mathsf{b}^{>d} = \mathsf{b}^{<1} = \mathsf{A}_z^{>d} = \mathsf{A}_z^{<1} = \mathsf{b}_z^{>d} = \mathsf{b}_z^{<1} = 1$.
for $k = d, \ldots, 2$ **do** {Orthogonalization and reduction}
  Find LQ decomposition $x^{\langle k|} = LQ$, $QQ^* = I$.
  Replace $x^{\langle k|} := Q$, and $x^{|k-1\rangle} := x^{|k-1\rangle} L$.
  Find LQ decomposition $z^{\langle k|} = LQ$, $QQ^* = I$.
  Replace $z^{\langle k|} := Q$, and $z^{|k-1\rangle} := z^{|k-1\rangle} L$.
  Compute reductions $\mathsf{A}^{>k-1}$, $\mathsf{b}^{>k-1}$ by (4.42), $\mathsf{A}_z^{>k-1}$, $\mathsf{b}_z^{>k-1}$ by (4.48).
end for
for $k = 1, \ldots, d$ **do** {Optimization over TT cores}
  Form $b_k = \tau(\mathsf{b}^{<k}, b^{(k)}, \mathsf{b}^{>k})$, and {optional} $A_k = \tau(\mathsf{A}^{<k}, A^{(k)}, \mathsf{A}^{>k})$.
  Solve $A_k u^{(k)} = b_k$. {Take $x^{(k)}$ as an initial guess, use (4.43)}
  Compute SVD $u^{|k\rangle} \approx U\Sigma V^*$ s.t. $\left\| u^{|k\rangle} - U\Sigma V^* \right\| \leqslant \varepsilon \left\| u^{|k\rangle} \right\|$ or $\operatorname{rank}(\Sigma) \leqslant r$, replace $u^{|k\rangle} := U\Sigma V^*$. {Optional}
  if $k \neq d$ **then** {Enrichment, orthogonalization and reduction}
    Update $z^{(k)} = \tau(\mathsf{b}_z^{<k}, b^{(k)}, \mathsf{b}_z^{>k}) - \tau(\mathsf{A}_z^{<k}, A^{(k)}, \mathsf{A}_z^{>k}) u^{(k)}$. {Use (4.43)}
    Update $z_k^{(k)} = \tau(\mathsf{b}^{<k}, b^{(k)}, \mathsf{b}_z^{>k}) - \tau(\mathsf{A}^{<k}, A^{(k)}, \mathsf{A}_z^{>k}) u^{(k)}$. {Use (4.43)}
    Expand $x^{|k\rangle} := \begin{bmatrix} U & z_k^{|k\rangle} \end{bmatrix}$, $x^{\langle k+1|} := \begin{bmatrix} x^{\langle k+1|} \\ 0 \end{bmatrix}$.
    Find QR decomposition $x^{|k\rangle} = QR$, $Q^*Q = I$.
    Replace $x^{|k\rangle} := Q$, and $x^{\langle k+1|} := Rx^{\langle k+1|}$.
    Find QR decomposition $z^{|k\rangle} = QR$, $Q^*Q = I$.
    Replace $z^{|k\rangle} := Q$, and $z^{\langle k+1|} := Rz^{\langle k+1|}$.
    Compute reductions $\mathsf{A}^{<k+1}$, $\mathsf{b}^{<k+1}$ by (4.41), $\mathsf{A}_z^{<k+1}$, $\mathsf{b}_z^{<k+1}$ by (4.47).
  end if
end for
return $x = \tau(x^{(1)}, \ldots, x^{(d)})$, $\tilde{z} = \tau(z^{(1)}, \ldots, z^{(d)})$.

Chapter 5

Verification with applications: numerical experiments

In this chapter we present various numerical examples, demonstrating particular features of the tensor product methods and algorithms, and serving as "proofs-of-concept".

We focus mainly on the chemical master and the Fokker-Planck equations, outlined in the first part of this thesis. Some "sanity" checks, such as the solution of the high-dimensional Poisson equation, will not be presented. The reader is refereed to the corresponding papers.

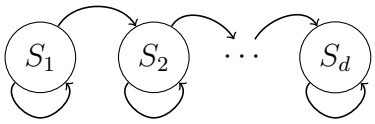
Most of the experiments are taken from the (co-)authored papers, e.g. [44, 46, 53]. Some verification tests, which were previously announced in a reduced form (due to e.g. a journal page limit), are now extended and investigate certain effects of secondary importance. Other experiments, concerning the proofs-of-concepts for particular applications, are presented with no changes.

The implementation of methods and tests is based on the MATLAB package TT-Toolbox `http://github.com/oseledets/TT-Toolbox`, co-developed by author. The TT-Toolbox provides an object-oriented framework for TT, QTT and QTT-Tucker operations, and contains both the basic algebra (additions, products and roundings, cf. Sec. 2.1.5) and advanced DMRG and AMEn algorithms, for example:

- `dmrg_solve3.m` two-site DMRG algorithm 9 for linear systems;
- `amen_solve2.m` AMEn method for linear systems (encapsulates both Alg. 11 and 12);
- `amen_mv.m` AMen method for the MatVec approximation (Alg. 12 tuned for $A = I$ and $b = Gf$, see Remark 4.2.3).

Some bottlenecks (such as the GMRES algorithm for the local system (4.7) solution) were refactored to Fortran90 and C MEX libraries to improve the performance. The computations were conducted on Linux workstations with 2.8 GHz AMD Opteron CPU at MPI MiS, Leipzig and 2.0 GHz Intel Xeon CPU at INM RAS, Moscow.

Figure 5.1: Cascade signaling network



5.1 Chemical master equation for biological networks

5.1.1 Short time cascade: comparison of methods

We start with the verification of AMEn methods, and their comparison with the DMRG on the discretized chemical master equation for a d-dimensional cascade gene regulatory network, following [53]. A cascade process occurs when adjacent genes produce proteins which influence on the expression of a succeeding gene, see Fig. 5.1. This is a typical model in genetic networks; as an example, the lytic phase of the λ-phage system [204] can be considered. In Fig. 5.1, the arrows denote feedbacks between species arising from the propensity rates w^m: an arrow going from the i-th to j-th component means that the rate of the reaction involving the j-th protein depends on the copy number of the i-th one. The number of reactions in the cascade system is $M = 2d$, according to the two classes of processes: self-dependent *destructions* (which model e.g. escapes of molecules through the cell membrane), and neighbor-driven *creations* (with some saturating propensity, e.g. Michaelis-Menten).

The stoichiometry assumes *monomolecular* reactions: all destruction reactions, labeled with $m = 1, \ldots, d$, decrease the copy number of the m-th substance by one, i.e. $\boldsymbol{z}^m = -\boldsymbol{e}_m$, the negative m-th unit vector, and the creation reactions, labeled with $m = d+1, \ldots, 2d$, increase the copy numbers by one, thus having positive unit stoichiometric vectors $\boldsymbol{z}^m = \boldsymbol{e}_{m-d}$.

As was noted, the m-th destruction propensity depends only on x_m, and its rank-1 decomposition reads

$$w^m(\boldsymbol{i}) = e(i_1) \cdots e(i_{m-1}) \cdot \check{w}^m(i_m) \cdot e(i_{m+1}) \cdots e(i_d),$$

where $e(i_k) = 1 \ \forall i_k = 0, \ldots, n_k - 1$. The corresponding part of the operator is of the Laplace form

$$A_1 = D_1 \otimes J^0 \cdots \otimes J^0 + \cdots + J^0 \otimes \cdots \otimes D_d, \quad D_m = (J^{-1} - J^0)\,\mathrm{diag}(\check{w}^m), \qquad (5.1)$$

where J^z are according to (1.10). It was already discussed that (5.1) is representable in the TT format with all ranks 2.

In the creation part, the propensities depend on the copy numbers of the previous species, i.e. $w^m = \hat{w}^m(i_{m-d-1})$, for $m = d+2, \ldots, 2d$. Thus, the second operator part sums the two-variate terms,

$$A_2 = D_1^1 \otimes J^0 \cdots \otimes J^0 + D_1^2 \otimes D_2^2 \otimes J^0 \cdots \otimes J^0 + \cdots + J^0 \cdots \otimes D_{d-1}^d \otimes D_d^d, \qquad (5.2)$$

where $D_{m-1}^m = \mathrm{diag}(\hat{w}^{m+d})$, $D_m^m = (J^1 - J^0)$, $m = 1, \ldots, d$. We may notice that (5.2) possesses the TT representation (3.11) with all ranks 3. Therefore, the total CME operator $A = A_1 + A_2$ in (1.11) is a rank-5 matrix TT format.

The particular model parameters were chosen in accordance with [108, 4, 46].

- Destruction propensities $\check{w}^m(i_m) = 0.07 \cdot i_m$, $m = 1, \ldots, d$.
- Creation propensities $\hat{w}^{d+1} = 0.7$, and the rest are $\hat{w}^m(i_{m-1-d}) = \frac{i_{m-1-d}}{5+i_{m-1-d}}$ for $m = d+2, \ldots, 2d$.
- Dimension $d = 20$ species;
- FSP box sizes (mode sizes for tensors) $n_k = n = 64$.

The non–symmetric linear system $B\psi = f$ arises after the backward Euler discretization $\psi(t + \delta t) = (I - \delta t A)^{-1}\psi(t)$, where $\psi(t)$ is the $n \times \ldots \times n$ tensor (or n^d vector) of the CME solution (1.11). Following Section 1.3, all snapshots $\psi(t_p)$, $p = 1, \ldots, N_t$, are stored simultaneously in a $n \times \ldots \times n \times N_t$ tensor $\psi = \big[\psi(p\delta t)\big]_{p=1}^{N_t}$ of dimension $d + 1$. We use the temporal preconditioning (the Euler counterpart of (1.20)), such that the right-hand side is $f = \psi(0) \otimes e$, where $e = (1, \ldots, 1)$, and the matrix writes

$$B = I_{n^d} \otimes I_{N_t} - A \otimes \delta t G_t^{-1},$$

where $G_t = \mathrm{tridiag}(-1, 1, 0) \in \mathbb{R}^{N_t \times N_t}$ is the discrete gradient, and A is the CME matrix, discussed above.

In this comparative experiment, we select a moderate time interval $T = 10$, but relatively many time steps $N_t = 2^{12}$, s.t. $\delta t = T/N_t = \mathcal{O}(10^{-3})$ is small enough. We compress all the data (both in the copy numbers state space and time) in the QTT format (see Sec. 2.2.1). The initial state is the first unit vector, $\psi(0) = (1, 0, \ldots, 0)$, which corresponds to zero copy numbers of all species with the probability one. This results in moderate QTT ranks for both B and f.

The full problem size $N_t n^d \sim 10^{40}$ makes the straightforward solution impossible. Existing techniques either leave the CME aside (the SSA method and its descendants), or employ high-dimensional techniques like Smolyak's sparse grids [108], greedy approximations [4, 3], or dynamics on tensor manifolds [122]. Here we apply the AMEn algorithm (both $\mathrm{AMEn}_{\mathrm{svd}}$ Alg. 11 and $\mathrm{AMEn}_{\mathrm{als}}$ Alg. 12) and compare it with the two-site DMRG Alg. 9, both with ($\mathrm{DMRG}_{\mathrm{rnd}}$) and without (DMRG) the random enrichment (4.13) of rank 4. For some systems with moderate dimensions and small time steps, the DMRG method can be of a good use, as was demonstrated in [126]. However, we show that the AMEn algorithm performs better for this (more difficult) problem.

We set the relative tensor truncation threshold for the solution $\varepsilon = 10^{-6}$, and track the convergence of different methods towards the reference solution, which is computed via the $\mathrm{AMEn}_{\mathrm{svd}}$ with the $\varepsilon = 10^{-9}$. The results are given in Fig. 5.2. Since $B\psi = f$ is not a symmetric positive definite system, we can apply the traditional symmetrization $B^*B\psi = B^*f$. The symmetrization squares both the condition number and the TT ranks of the matrix, and usually slows the computation. Therefore, we can also apply all algorithms directly to $B\psi = f$, dropping the theoretical support.

We observe that neither the initial system nor the symmetrized formulation are solved by the original DMRG (without the enrichment). Since it uses only the local information on the system, it returns the approximation with significantly underestimated TT ranks, which is also reflected by small CPU times. The random-enriched

Table 5.1: Errors in final time snapshots (err) and the CPU times in seconds (time) of different methods

	AMEn_{svd}		AMEn_{als}		DMRG		KSL
	B	B^*B	B	B^*B	B	B^*B	
err	8.3e-6	2.7e-5	9.2e-6	2.4e-5	9.6e-1	1.8e+0	8.4e-4
time	48.7	343	15.3	47.4	7.30	5.21	226

DMRG fails for the non-symmetric system as well, though exhibiting a slow convergence in the symmetrized formulation. This reflects the inexact steepest descent theory (4.18) to some extent: the DMRG makes a progress if the enrichment is not orthogonal to the residual, which is likely to happen with random vectors. However, the rate of such progress is usually far from optimal.

All AMEn algorithms deliver a satisfactory solution. Interestingly, the non-symmetric versions appear to be even faster and more accurate than the symmetrized counterparts. This evidences that the practical convergence rates of AMEn methods are much better than the theoretical estimates provided by the steepest descent theory. Focusing on two AMEn realizations, we note that the AMEn_{als} speeds up the computations essentially compared to the AMEn_{svd} algorithm (especially when TT ranks of the matrix are large, e.g. $r_k \simeq 40$ for B^*B), while the convergence does not deteriorate.

Since the CME problem is an ODE, we may be interested in the final time snapshot $\psi(N_t\delta t)$. As soon as we approximate all time layers in the common TT format, it is worth to check the accuracy of individual components. In Table 5.1 we show the relative Frobenius norm errors for $\psi(N_t\delta t)$ verified w.r.t. the reference solution. The observed errors are at the same level as the accuracies of the total solution ψ in Fig. 5.2.

Now we can compare the last snapshot computed by the AMEn with the result obtained by another ODE integrator. A dynamical problem may be put onto the tensor manifold via the so-called *Dirac-Frenkel* principle (see e.g. [159]): we solve the problem

$$\min_{x^{(k)}\in\mathbb{C}^{r_{k-1}\times n_k\times r_k}} \left\| \frac{d\tau(\{x^{(k)}\})}{dt} - \frac{dy}{dt} \right\|,$$

projecting the exact velocity dy/dt onto the tangent space of the TT manifold. Several theoretical issues were addressed in [175], and the numerical splitting w.r.t. the TT cores $x^{(k)}$ was proposed in [174] as the KSL scheme. The latter may be applied to our ODE if dy/dt is substituted by $M\tau(\{x^{(k)}\})$.

The KSL scheme is efficient in simulation of molecular vibrations [193]. However, it possesses the same drawback as the ALS Alg. 8: TT ranks of the solution are predefined and fixed. Moreover, we observe that the KSL delivers much larger error than the other methods, even if the TT ranks are set to the proper values. When we decrease the time step, the computational cost of the KSL grows (226 seconds for $N_t = 50$ time steps) and becomes larger than the one of AMEn. However, the error of the KSL stagnates at the level of $8 \cdot 10^{-4}$ and does not improve.

Additional study may be conducted with respect to the enrichment rank, which is a specific parameter for the AMEn. In Fig. 5.3, we track the convergence of the theoretically supported symmetrized AMEn_{svd} and the fastest non-symmetric AMEn_{als}

Table 5.2: CPU time in seconds (time), number of iterations (it), maximal TT rank of the Krylov vector $r(w)$, and relative solution error (err) of TT-GMRES Alg. 7 vs. the stopping and truncation threshold ε

ε	time	it	$r(w)$	err
10^{-2}	50.456	17	40	2.45e-02
10^{-3}	776.54	27	140	2.62e-03
10^{-4}	9514.4	37	260	2.37e-04
10^{-6}			***	

for different residual TT ranks $\boldsymbol{\rho} = (\rho, \ldots, \rho)$. We may notice that it is indeed not necessary to take excessively large TT ranks; in fact, moderate intermediate values $\rho \sim 5$ appear to be optimal in terms of the accuracy-cost ratio. This is a crucial difference with the classical iterative methods in the TT arithmetics (cf. Alg. 7), which require rather accurate high-rank approximations of the residual and Krylov vectors, otherwise stagnations may occur, according to Stat. 4.1.3. Contrarily, the AMEn methods work robustly with low-rank approximations of the residual, and a high-rank enrichment may be even slower than the rank-1 update.

The behavior of the TT-GMRES method on the problem $B\psi = f$ may be seen in Table 5.2. Algorithm 7 demonstrates a stable convergence up to the requested threshold. However, tensor ranks of Krylov vectors grow rapidly with the number of iterations, and even a few more steps may require a substantially larger CPU time. In particular, we could not conduct the TT-GMRES with the AMEn accuracy $\varepsilon = 10^{-6}$ due to memory limitations.

A comparison with the corrected one-site DMRG (not presented here; please refer to [54]) shows that there is much more freedom in the choice of the enrichment rank in the AMEn, rather than the weight in the corrected DMRG. The AMEn converges for any enrichment rank (possibly not in the optimal time), whereas the corrected one-site DMRG may significantly loose the accuracy, if the weight is chosen inappropriately.

Figure 5.2: CME example, Frobenius-norm error (top) and the residual (bottom) in different methods w.r.t. the iteration number (left) and CPU time in seconds (right). Solid line: a method is applied directly to the non-symmetric system $B\psi = f$, dashed line: a method is applied to the symmetrized system $B^*B\psi = B^*f$. The residual TT rank is fixed to $\rho = 4$. The solution is truncated using the relative Frobenius-norm threshold $\varepsilon = 10^{-6}$.

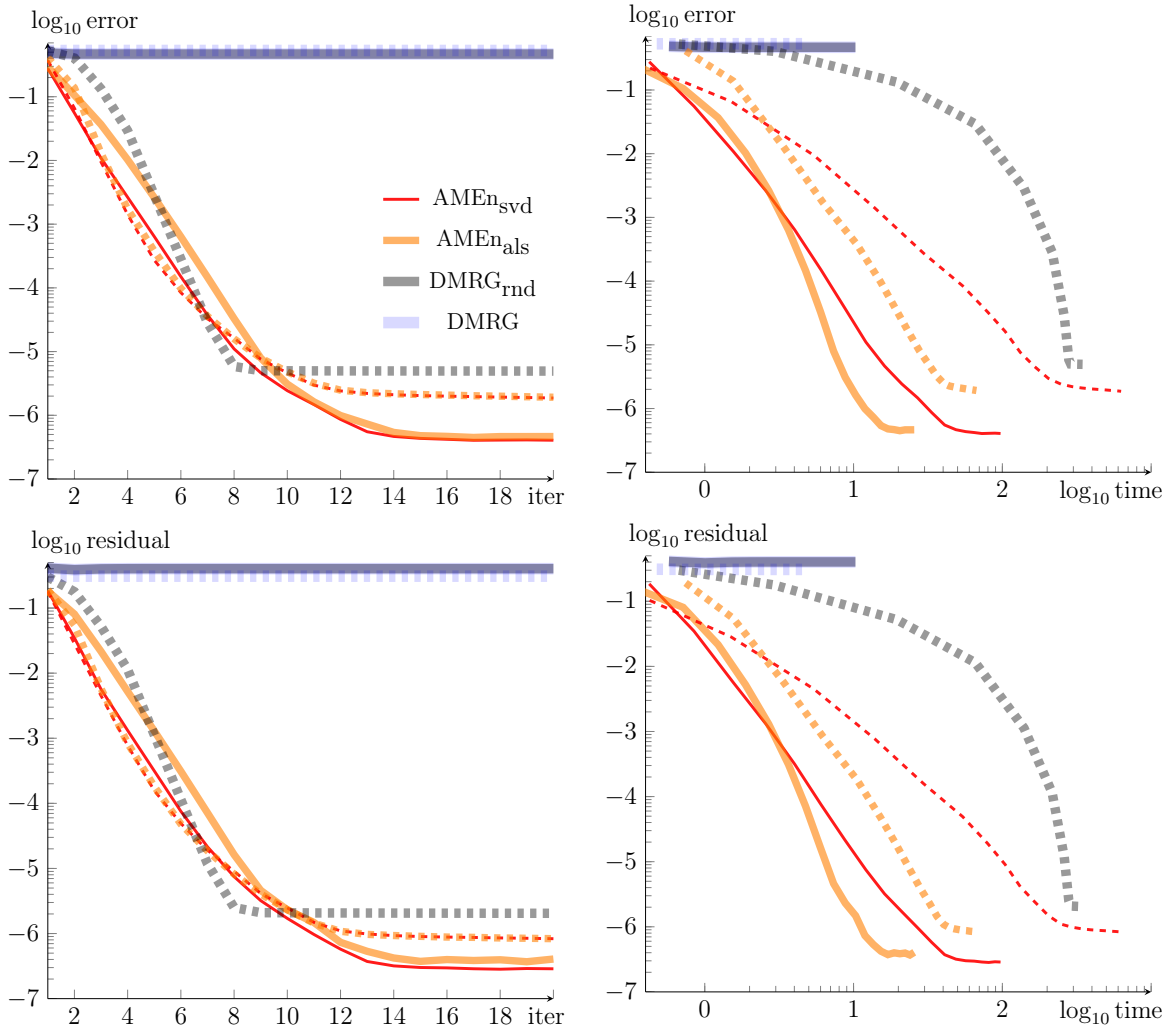
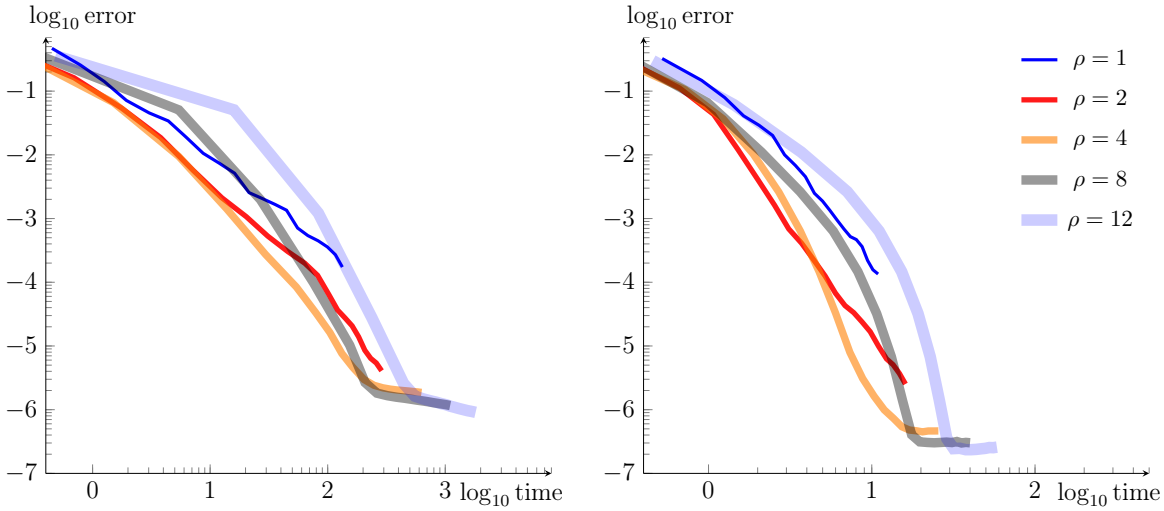


Figure 5.3: CME example, Frobenius-norm error vs. CPU time in seconds and the TT rank of the residual ρ. Left: AMEn_{svd} applied to $B^*B\psi = B^*f$, right: AMEn_{als} applied to $B\psi = f$.



5.1.2 Long time cascade: full evolution history

In the previous example we confirmed the efficiency of the AMEn algorithm in a non-trivial high-dimensional problem. However, the time interval $T = 10$ is not enough to catch the stationary solution of the CME. In this and the next experiments (Sections 5.1.2, 5.1.3 and 5.1.4), we show three examples of a long dynamical evolution (such that the solution converges to the steady state) from [46] (see [43] for the improved revised version), which are more relevant to practical applications.

First, we extend the simulation of the cascade to $\hat{T} = 400$. The reaction network is the same as in the previous subsection. A couple of common differences is as follows.

- The ODE is solved via the restarted simultaneous state-time scheme from Section 1.3, see Remark 1.3.3. We perform an additional test to detect an optimal splitting length T.
- Tensor rounding and solution threshold $\varepsilon = 10^{-5}$.

As a resulting quantity (see Fig. 5.4, left), we compute the average copy numbers of all species in time,

$$\langle i_k \rangle(t) = \frac{\sum_{\boldsymbol{i}} i_k \psi(\boldsymbol{i}, t)}{\sum_{\boldsymbol{i}} \psi(\boldsymbol{i}, t)} = \frac{(\mathtt{i}_k, \psi(t))}{(\mathtt{e}, \psi(t))}, \quad k = 1, \dots, d, \tag{5.3}$$

where $e = (1, \dots, 1) \in \mathbb{R}^n$, $\mathtt{e} = e \otimes \cdots \otimes e$ is the all-ones tensor, and $\mathtt{i}_k = e \otimes \cdots \otimes e \otimes \{i_k\} \otimes e \otimes \cdots \otimes e$ is a tensor populated with all values of i_k. Note that the TT-scalar products in (5.3) are computed easily via Alg. 1.

One interesting feature of the cascade system is the delay between the equal concentration levels of different species, which can be observed in Fig. 5.4 (left). Therefore, an accurate time solution history is important to measure such delays.

Additionally, the convergence of the transient solution to the steady state is shown in Fig. 5.4 (right), which confirms that the chosen interval $\hat{T}$ is large enough to catch the stationary solution with a satisfactory confidence.

To demonstrate the performance of the QTT time discretization scheme, we present the total CPU times for different numbers of time steps N_t in each subinterval $[(q-1)T, qT]$, $q = 1, \dots, \hat{T}/T$ (Fig. 5.5, left), and the interval widths T (Fig. 5.5, right).

We see that the QTT format ensures the logarithmic grows of the computational time with the number of time steps.

The optimal length of the time intervals, leading to the fastest solution process, is the intermediate value $T = 10$. For smaller T, the solution in each interval is cheap, but the number of intervals is large. Contrarily, for large T the conditioning and TT ranks of each system are high, and it takes more time to deliver the solution.

Figure 5.4: Cascade CME example. Left: Mean copy numbers $\langle i_k \rangle(t)$. Right: Residual $\|A\psi(t)\|/\|\psi(t)\|$

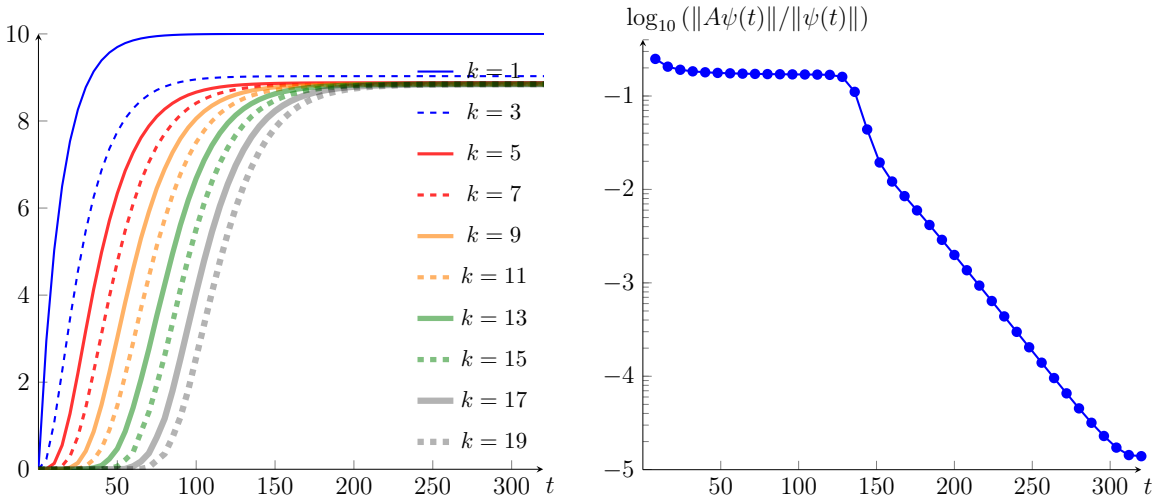


Figure 5.5: Cascade CME example, CPU time (sec.) vs. discretization parameters. Left: time vs. $\log_2(N_t)$, $T = 15$. Right: time vs. T, $N_t = 2^{14}$.

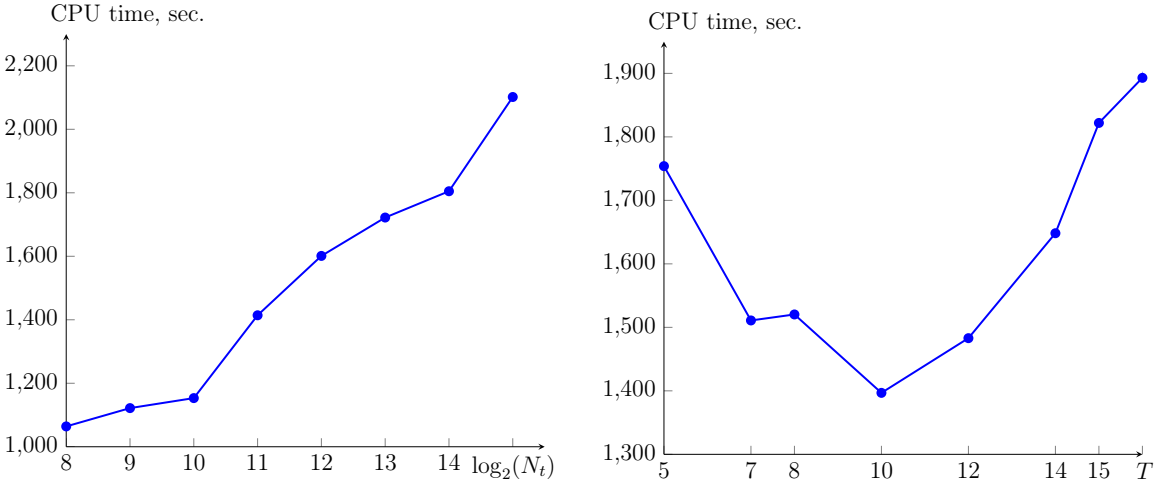
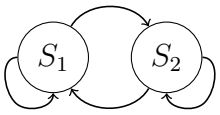


Figure 5.6: Toggle switch



5.1.3 Genetic toggle switch with a parameter

In this test, we simulate the synthetic bistable genetic toggle switch (see Fig. 5.6) developed in *Escherichia coli* [69] in presence of a parameter. The model contains $d = 2$ species and $M = 4$ reactions, specified as follows.

$$w^1(\boldsymbol{i}) = \frac{\alpha_1}{1 + i_2^\beta}, \quad \boldsymbol{z}^1 = (1, 0): \quad \text{generation of } S_1; \quad \alpha_1 = 156.25,\ \beta = 2.5.$$

$$w^2(\boldsymbol{i}) = i_1, \quad \boldsymbol{z}^2 = (-1, 0): \quad \text{destruction of } S_1.$$

$$w^3(\boldsymbol{i}) = \frac{\alpha_2}{1 + \dfrac{i_1}{(1 + y/K)^\eta}}, \quad \boldsymbol{z}^3 = (0, 1): \quad \text{generation of } S_2; \quad \alpha_2 = 15.6,\ \eta = 2.0015,\ K = 2.9618 \cdot 10^{-5}.$$

$$w^4(\boldsymbol{i}) = i_2, \quad \boldsymbol{z}^4 = (0, -1): \quad \text{destruction of } S_2.$$

According to the dominating production rate w^1, we restrict the state space to $n_1 = n_2 = n = 256$. A parameter y is the concentration of the IPTG catalyst, and is varying from 10^{-6} to 10^{-2}. The main feature of this system is the existence of the two so-called *low* (low i_2) and *high* metastable states. The probability to find the system in either of states depends on the concentration y, see Figure 5.8.

Note that y is not governed by the CME, but only enters the coefficients as a parameter, $w^m = w^m(\boldsymbol{i}, y)$, such that

$$\frac{d\psi(y,t)}{dt} = A(y)\psi(y,t) = \sum_{m=1}^{M} (\boldsymbol{J}^{\boldsymbol{z}^m} - \boldsymbol{J}^0)\,\mathrm{diag}(w^m(y))\psi(y,t) \quad \text{for each } y.$$

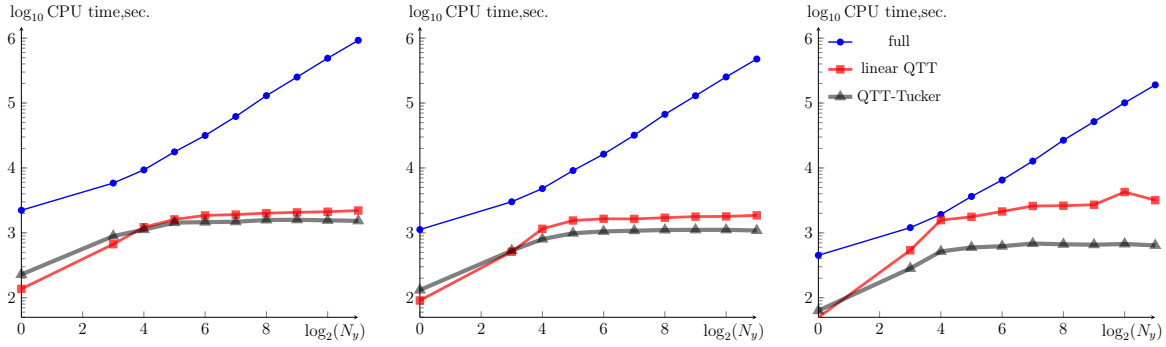
Introducing a discretization in the parameter (in this example, the collocation at the exp-uniform grid is used), we come to the block-diagonal linear system,

$$\frac{\partial\psi(t)}{\partial t} = \mathcal{A}\psi(t) = \begin{bmatrix} A(y_1) & & \\ & \ddots & \\ & & A(y_{N_y}) \end{bmatrix} \psi(t), \qquad \psi(t) = \begin{bmatrix} \psi(y_1, t) \\ \vdots \\ \psi(y_{N_y}, t) \end{bmatrix},$$

where $\{y_1, \ldots, y_{N_y}\}$ is the parametric grid, and $A(y_j)$ stands for the original CME matrix (1.11) with a fixed $y = y_j$, $j = 1, \ldots, N_y$.

If the total number of parametric points N_y is moderate, the very straightforward approach is to solve each CME equation for each y_j independently. However, if N_y is large (y can represent in fact a *multi*-parameter tuple with d_y variables, resulting in $N_y = n^{d_y}$ degrees of freedom in total), one may benefit from the tensor-structured data compression, and solve the global system at once, disregarding its block-diagonality. The cases of many parameters arise naturally in stochastic equations (e.g. [177, 61, 141]), when some coefficients may not be known exactly in advance, but only their possible ranges can be specified, or inverse problems, such as the model calibration

Figure 5.7: Toggle switch example, CPU time vs. $\log_2(N_y)$. Left: $\delta t = 1$, middle: $\delta t = 2$, right: $\delta t = 5$.



and sensitivity [222, 205]. The current example tends to the latter class, but the difference is not significant from the computational point of view.

Contrarily to the previous cascade problem, the switch simulation can be conducted in the full format without the separation of variables, since the problem size $n^2 N_y$ is feasible. Due to the diagonality in y, and sparsity in i_1, i_2, the direct elimination algorithms for sparse matrices are quite efficient for this example, and it is interesting to compare them with the tensor product techniques.

We seek for the steady state using the Euler iterations (see Section 1.3.2) in the time range $\hat{T} = 1000$, so that the stationarity accuracy is below the tensor rounding tolerance $\varepsilon = 10^{-5}$. The time step δt is varied from 1 to 5.

In the first test, we track the computational times for different parametric grid sizes and time steps δt, see Figure 5.7. As expected, the CPU time of the full format scheme demonstrates a linear growth with. N_y. For both QTT-based tensor formats the growth is logarithmic, but rather sensible TT ranks of the solution (up to 40) could yield a large contribution to the complexity. Surprisingly, the tensor scheme (AMEn_{als} applied to the inverse Euler system (1.22)) in both QTT and QTT-Tucker formats overcomes the full format solver even for one parametric point, i.e. on a two-dimensional problem of size 256^2.

When δt and N_y increase, so do the conditioning of the matrix and solution ranks. In this case, the QTT-Tucker format provides additional complexity reduction, by a factor of ca. 4 w.r.t. the linear QTT representation.

Since the tensor rounding introduces a perturbation of the magnitude $\varepsilon = 10^{-5}$, we need to check the actual error in the observable quantities. We track the point $y = 3 \cdot 10^{-5}$ (it is located in the transient region, see Fig. 5.8) by adding it explicitly to the grid, and check how the error propagates into the average copy numbers, i.e. measure

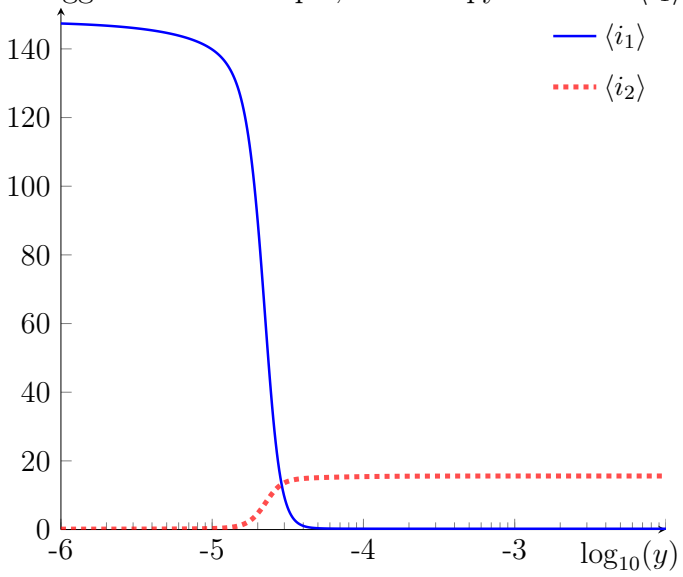
$$\mathtt{E}(i_k) = |\langle i_k \rangle - \langle i_k \rangle_{ex}| / \langle i_k \rangle_{ex}, \quad k = 1, 2, \tag{5.4}$$

where $\langle i_k \rangle_{ex}$ is taken from the simulation in the full format, see Table 5.3. For brevity, we show only the mean and maximal errors over different N_y and δt. We may observe a rather large error overhead up to 250, but still the accuracy $\mathcal{O}(10^{-3})$ is maintained, which is usually enough for phenomenological models, while the complexity is substantially reduced.

Table 5.3: Toggle switch example, mean and maximal relative errors (5.4) at $y = 3{\cdot}10^{-5}$

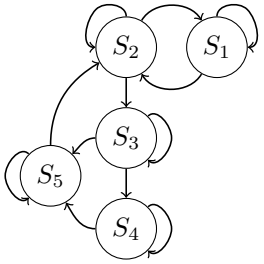
	linear QTT		QTT-Tucker	
	$\mathtt{E}(i_1)$	$\mathtt{E}(i_2)$	$\mathtt{E}(i_1)$	$\mathtt{E}(i_2)$
mean	8.8e-4	7.8e-5	5.5e-4	4.9e-5
max	2.5e-3	2.2e-4	2.1e-3	1.8e-4
$\delta t^{\max}$	1	1	1	1
$N_y^{\max}$	2^7	2^7	2^6	2^6

Figure 5.8: Toggle switch example, mean copy numbers $\langle i_1 \rangle$, $\langle i_2 \rangle$ vs. y



Finally, we plot the average copy numbers of both reacting proteins versus the concentration of the catalyst, see Figure 5.8. The quantity $\langle i_2 \rangle$ can be used also as a measure of the fraction of the cells occupying the high (or low) state. Indeed, it demonstrates the asymptotic: with y tending to ∞, the cells tend to stay in the high state. Therefore, the fraction of the high-state cells can be estimated as the normalized $\langle i_2 \rangle$. We observe a good agreement with the experimental results given in [69, Fig. 5(b)].

Figure 5.9: λ-phage



5.1.4 λ-phage

In the last example we simulate the life cycle of the bacteriophage-λ [108, 122]. The first paper [108] considers the sparse grids approach. The second one is more tensor related and uses the Dirac-Frenkel principle for the dynamical low-rank approximation in the Tucker format (DLRA). The simulation was done for a relatively small time interval ($T = 10$), which is not enough to achieve the stationary solution. Numerical treatment is difficult, since both the copy number of the second protein and the relaxation time are large ($\sim 10^4$). With the help of the QTT format, equipped with the AMEn linear solver, we were able to perform an efficient computation of the stationary solution on very large grids.

The model contains $d = 5$ species and $M = 10$ reactions, defined as follows ($\boldsymbol{e}_1, \ldots, \boldsymbol{e}_5$ are unit vectors).

$$
\begin{array}{llll}
w^1(\boldsymbol{i}) = \dfrac{a_1 b_1}{b_1 + i_2}, & \boldsymbol{z}^1 = \boldsymbol{e}_1: & \text{generation of } S_1; & a_1 = 0.5,\ b_1 = 0.12. \\
w^2(\boldsymbol{i}) = c_1 \cdot i_1, & \boldsymbol{z}^2 = -\boldsymbol{e}_1: & \text{destruction of } S_1; & c_1 = 0.0025. \\
w^3(\boldsymbol{i}) = \dfrac{(a_2 + i_5) b_2}{b_2 + i_1}, & \boldsymbol{z}^3 = \boldsymbol{e}_2: & \text{generation of } S_2; & a_2 = 1,\ b_2 = 0.6. \\
w^4(\boldsymbol{i}) = c_2 \cdot i_2, & \boldsymbol{z}^4 = -\boldsymbol{e}_2: & \text{destruction of } S_2; & c_2 = 0.0007. \\
w^5(\boldsymbol{i}) = \dfrac{a_3 b_3 i_2}{b_3 \cdot i_2 + 1}, & \boldsymbol{z}^5 = \boldsymbol{e}_3: & \text{generation of } S_3; & a_3 = 0.15,\ b_3 = 1. \\
w^6(\boldsymbol{i}) = c_3 \cdot i_3, & \boldsymbol{z}^6 = -\boldsymbol{e}_3: & \text{destruction of } S_3; & c_3 = 0.0231. \\
w^7(\boldsymbol{i}) = \dfrac{a_4 b_4 i_3}{b_4 \cdot i_3 + 1}, & \boldsymbol{z}^7 = \boldsymbol{e}_4: & \text{generation of } S_4; & a_4 = 0.3,\ b_4 = 1. \\
w^8(\boldsymbol{i}) = c_4 \cdot i_4, & \boldsymbol{z}^8 = -\boldsymbol{e}_4: & \text{destruction of } S_4; & c_4 = 0.01. \\
w^9(\boldsymbol{i}) = \dfrac{a_5 b_5 i_3}{b_5 \cdot i_3 + 1}, & \boldsymbol{z}^9 = \boldsymbol{e}_5: & \text{generation of } S_5; & a_5 = 0.3,\ b_5 = 1. \\
w^{10}(\boldsymbol{i}) = c_5 \cdot i_5, & \boldsymbol{z}^{10} = -\boldsymbol{e}_5: & \text{destruction of } S_5; & c_5 = 0.01.
\end{array}
$$

First, we investigate a short time range, $\hat{T} = 10$, and compare different methods. The state grid $16 \times 64 \times 16 \times 16 \times 16$ provides enough capacity for the FSP solution. The initial state of the CME is chosen in accordance with [122] as the following multinomial distribution,

$$\psi(\boldsymbol{i}, 0) = \frac{3!}{i_1! \cdots i_5! \cdot (3 - |i|)!} 0.05^{|i|} (1 - 5 \cdot 0.05)^{3 - |i|} \cdot \theta(3 - |i|),$$

where $|i| = i_1 + \cdots + i_5$, and $\theta(s)$ is the Heaviside function. This distribution vector can be constructed straightforwardly as a full-format $4 \times 4 \times 4 \times 4 \times 4$-tensor, since the

Heaviside function is zero if any of i_k is greater than 3. After that, the TT decomposition (with ranks 4) is computed using Alg. 4, and each factor is expanded by zeros to the desired grid size (one-dimensional operation). Finally, the TT representation is re-approximated into the QTT format.

Figures 5.11, 5.10 and Table 5.4 summarize the behavior of four techniques. Our suggested method is the AMEn Alg. 12, applied to the coupled state-time system (1.16). The tensor approximation threshold is set to $\varepsilon = 10^{-5}$, but we vary the one-step time interval T and the number of inner time steps N_t.

The second approach is the DLRA algorithm (we refer to the results reported in [122]), and the KSL scheme, considered in Sec. 5.1.1. Since the KSL scheme does not allow to adapt tensor ranks, we have to guess them a priory. We use the following strategy: QTT blocks of the initial state are augmented by zeros up to the chosen rank value (e.g. $r = 20$ in Fig. 5.11).

The reference solution is computed in the standard vector form without approximations, using the Crank-Nicolson scheme with the time step $\tau = 10/4096$. The propagation matrices of size 2^{22} were assembled in the MATLAB `sparse` format, and the minimal residual method was employed for the implicit stage.

The qualitative correctness can be seen from the marginal distributions, shown in Figure 5.10. We see that the plots coincide with the previous results in e.g. [122].

The Frobenius-norm errors of the AMEn and KSL solutions w.r.t. the reference function, as well as the CPU times are presented in Fig. 5.11. The horizontal axis in Fig. 5.11 is the effective time step δt. Note that the same δt may correspond to different parameters, cf. $\delta t = T/N_t$ in (1.16). For example, the most left-bottom plot in the left plane of Fig. 5.11 is divided into two parts: in the solid region, we fix $N_t = 64$ and vary $T \in \{1.25, 2.5, 5, 10\}$, while in the dotted part $T = 1.25$ and N_t is varied from 2^6 to 2^{11} (both parts belong to the AMEn approach). While the time step δt is large enough, we may observe from the linear fit that the second order of approximation in the Crank-Nicolson scheme is manifested perfectly. With smaller δt, the error drops to the tensor truncation level $\mathcal{O}(\varepsilon)$. The same pattern takes place with other parameters (see $T = 5$). The error level at $T = 5$ is slightly higher, since the linear systems have larger condition numbers.

The error in the KSL scheme depends mostly on the tensor ranks of the solution, rather than the time step. Though the initial state is exactly representable in the QTT format with ranks 4, this is not the case at $t = 10$. From Fig. 5.11 and Table 5.4 we may see that even if we augment the ranks up to 20, the error level is larger than in the AMEn approach.

If we look at the complexity, from the right plane of Fig. 5.11 we observe the logarithmic dependence of the CPU time of the AMEn scheme vs. the time step. The number of operations in the KSL technique grows linearly with the number of time steps, which is also reflected in Fig. 5.11.

Finally, we note that the computations in the full representation (~ 2 hours according to Table 5.4), as well as the SSA (~ 3 hours according to [122]) are much slower. This shows a clear advantage of the AMEn tool even in a relatively small problem.

To detect the convergence to the steady state, a much larger time, $\hat{T} = \exp(10) \sim 2 \cdot 10^4$, is needed. Moreover, since the second protein develops copy number up to $4 \cdot 10^4$

Figure 5.10: Phage-λ, marginal probabilities at $t = 10$.

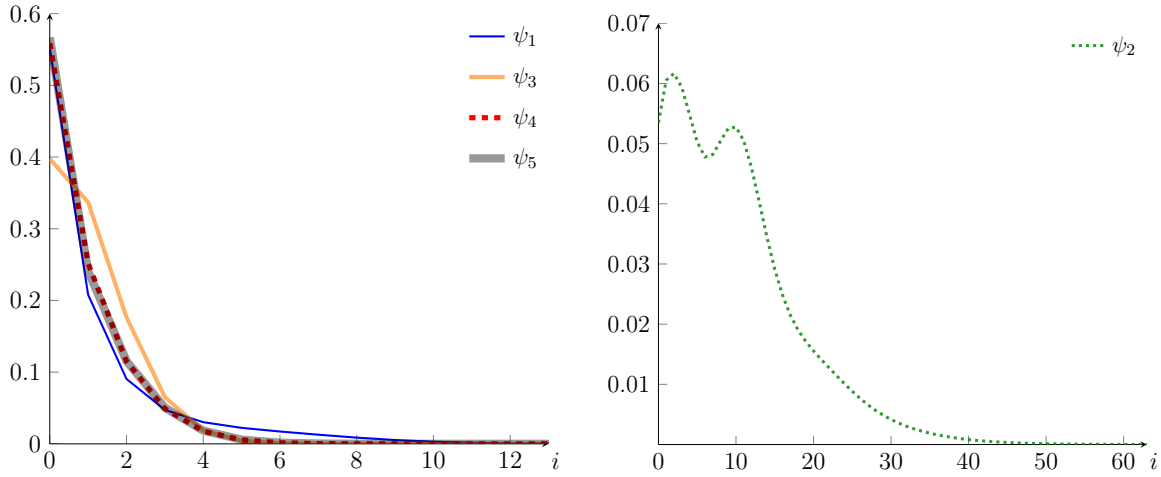


Table 5.4: Phage-λ, $\hat{T} = 10$. CPU times (sec.) and errors in different methods.

	full	KSL, $\delta t = \frac{10}{128}$			DLRA	AMEn	
	$\delta t = \frac{10}{4096}$	$r = 4$	$r = 8$	$r = 20$	from [122]	$\delta t = \frac{10}{64}$	$\delta t = \frac{1.25}{2048}$
time	6840	21.7	24.6	37.6	~ 300	22.9	32.0
$\frac{\|\psi - \psi_{full}\|}{\|\psi_{full}\|}$		1.59e-1	1.92e-2	3.92e-4	$\sim$ 3e-3	9.14e-4	2.74e-5

(see Fig. 5.12, right), the FSP box is set to $128 \times 65536 \times 64 \times 64 \times 64$, which makes the problem particularly challenging. The time interval is split via the exponential grid,

$$t_q = \exp(0.05 \cdot q), \quad q = 1, \ldots, 200.$$

In each subinterval $[t_{q-1}, t_q]$ we solve the coupled state-time system (1.16) in the QTT format with $N_t = 1024$ time steps. The convergence history and the cumulative CPU times are shown in Figure 5.12 (left) and Table 5.5 (left). We see that it takes about an hour of computational time to reveal the whole time history with the residual $\sim 10^{-7}$. Despite the large grids, in this case the the QTT-Tucker format does not outperform the linear QTT due to large core ranks, which dominate in the cubic asymptotic of the QTT-Tucker. Therefore, the QTT-Tucker solver takes about 4000 seconds.

If we are not interested in the transient processes, we may use the implicit Euler iterations (1.22) over the time points t_q. The total CPU times are shown in Table 5.5 (left), and the relative accuracies of the mean copy numbers at the final time point, compared with the fine time discretization, are shown in Table 5.5 (right). For this simulation, the QTT-Tucker format becomes preferable.

Up to the best of our knowledge, this is the first example of the direct simulation of the high-dimensional chemical master equation with significantly different temporal and spatial scales. To sum um, the presented examples reveal a promising potential of tensor product methods for accurate modeling in system biology.

Figure 5.11: Phage-λ, $\hat{T} = 10$. Error $\dfrac{\|\psi_{qtt} - \psi_{full}\|}{\|\psi_{full}\|}$ (left) and CPU time (right) vs. time interval δt.

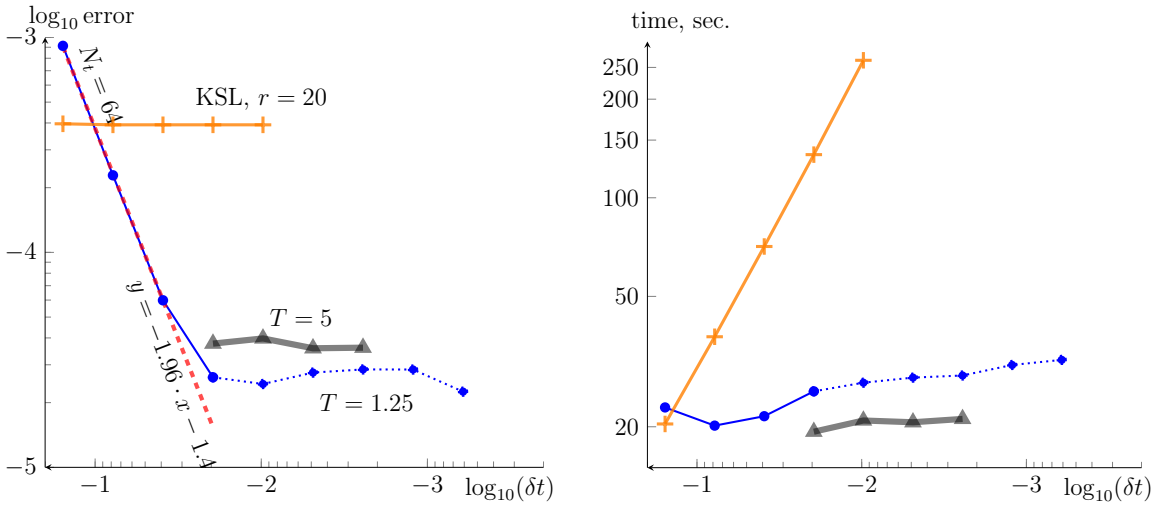
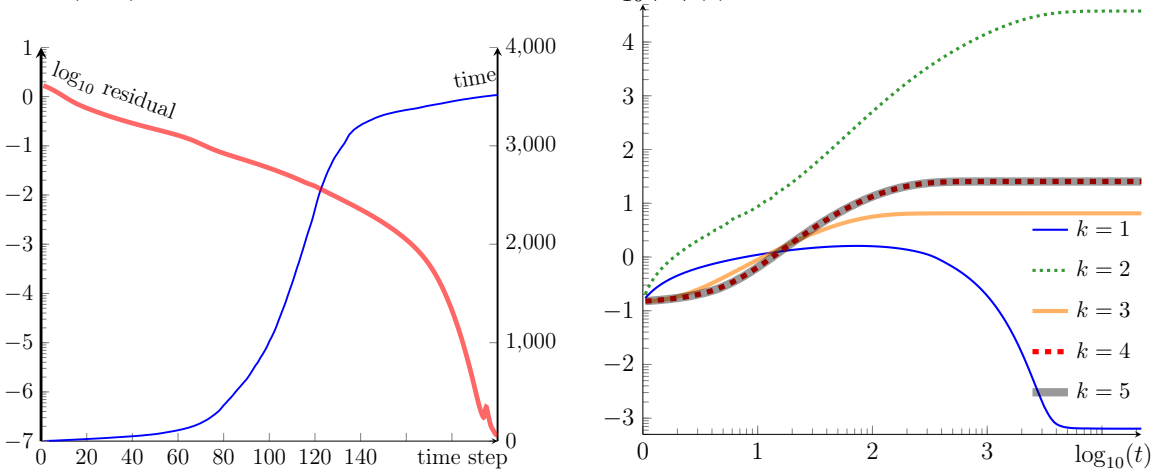


Table 5.5: Phage-λ example, large time range $\hat{T} = \exp(10)$. Left: CPU times (sec.). Right: accuracies of stationary copy numbers

	linear QTT		QTT-Tucker	
N_t	2^{10}	1	2^{10}	1
time	3500	670	4000	410

S_1	S_2	S_3	S_4	S_5
4.6e-5	1.7e-6	6.2e-8	3.1e-7	1.7e-7.

Figure 5.12: Phage-λ example. Left: residual $\|A\psi(t)\|/\|\psi(t)\|$ and the cumulative CPU time (sec.). Right: average copy numbers, $\log_{10}\langle i_k \rangle(t)$



5.2 Fokker-Planck equation for non-Newtonian fluid dynamics

In this section, we consider solution of the multi-dimensional Fokker-Planck equation for the polymeric flows, described in Section 1.1.3. Three examples involve Hookean, quasi-Hookean and FENE spring forces.

5.2.1 Hookean model

In the first example, following [47], we investigate the Hookean spring law, i.e. the Fokker-Planck equation with linear in $\boldsymbol{x}$ coefficients (1.7), and $W = 1$. We take a chain of $d = 4$ three-dimensional springs in the shear flow regime, which yields us the $dD = 12$-dimensional problem in total with the following velocity gradient,

$$\nabla_{\boldsymbol{y}} \boldsymbol{v} = \begin{bmatrix} 0 & \beta & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{5.5}$$

We will vary the shear parameter β to study its influence on the tensor ranks.

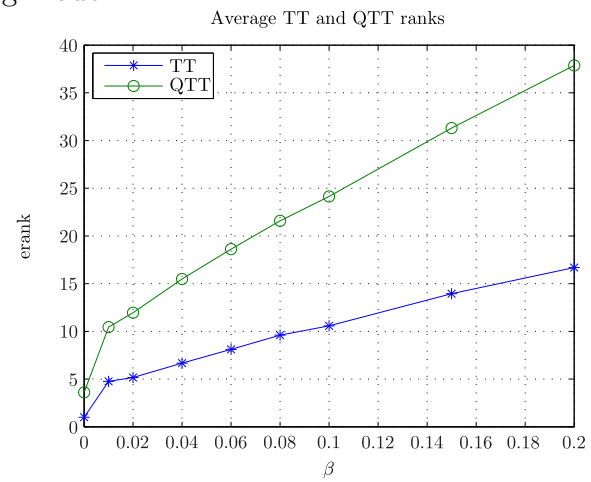
First, we discretize the Fokker-Planck equation (1.7) using the standard 3-point finite difference stencil with 256 points in each variable, and solve the dynamical problem (1.16) in the QTT format via the two-site DMRG (Alg. 9) until the steady state is reached, which requires the time interval $\hat{T} = 100$. As the initial state we choose the product of Gaussian functions with unit dispersion, $\psi(\boldsymbol{q}, 0) = \exp\left(-\frac{q_1^2+\ldots+q_{dD}^2}{2}\right)$, corresponding to the stationary solution with $\beta = 0$. The dependence of the average TT and QTT ranks of the $\varepsilon = 10^{-4}$-approximation on the shear strength β is given in Figure 5.13.

We see that the ranks grow linearly with the velocity gradient. Already for $\beta = 0.2$, the average QTT rank is equal to 38, the maximal rank (in the middle of the tensor train) is 84, and the CPU time of the whole process reaches ~ 20000 seconds. In practice, the values $\beta \gtrsim 1$ are more relevant, and additional complexity reduction techniques are necessary.

If we are only interested in the stationary solution, the Hookean model allows to construct the explicit formula (1.8). The exponential argument $V = \boldsymbol{q}^\top B \boldsymbol{q}$ admits an exact low-rank TT decomposition, see Thm. 3.2.1. To construct the stationary solution (1.8), we use the scaling and squaring method [180] to compute the pointwise exponential in the QTT or QTT-Tucker formats.

In Figures 5.14 we show the memory cells required to store the FPE solution, and the CPU times of the scaling and squaring method versus the Frobenius-norm rounding accuracy in different formats and methods. We compare the linear QTT (`1-QTT`), QTT-Tucker (`QTT-T`), and also the Extended TT (`ETT`) format, i.e. the QTT-Tucker without the quantization in Tucker factors, $n_Q = 256$, $L = 1$. The most time consuming operation is the squaring of the approximant in the scaling and squaring method. We use two approaches: computation of the exact Hadamard product, followed by the rounding Alg. 4 or 5 (`round`), and the two-site DMRG approximation (`dmrg`), see Alg. 9 and Remark 4.2.3.

Figure 5.13: Hookean 12D example. TT and QTT ranks of the stationary solution versus β in a 4-spring model.

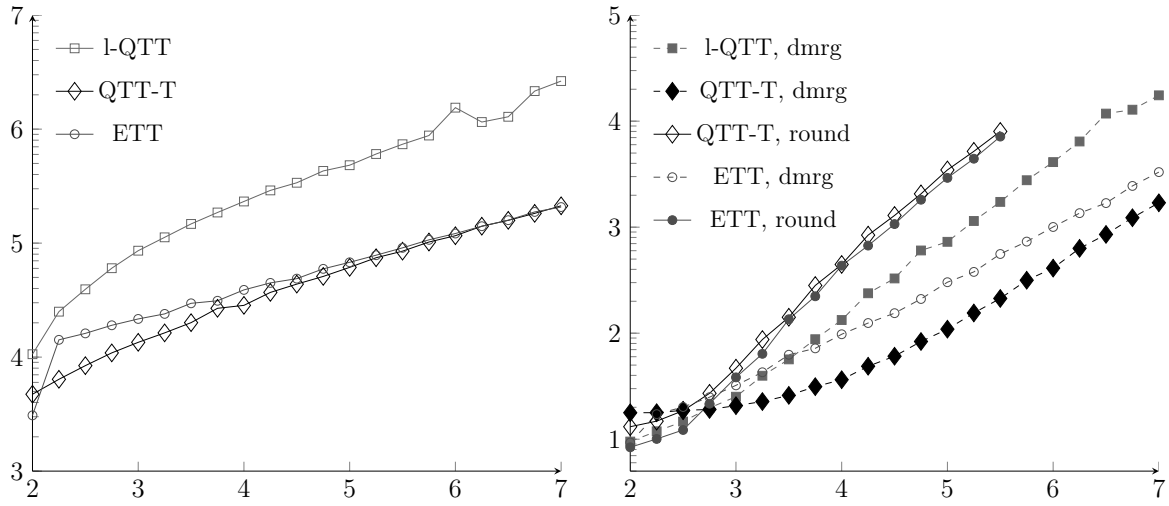


The `round` technique requires very large amount of memory, and fails with the Out-of-Memory exception at higher accuracies. Indeed, it tries to construct the exact Hadamard product, squaring the tensor ranks of the inputs. Therefore, we need $\mathcal{O}(dLr_F^4 + dr_C^6)$ storage and $\mathcal{O}(dLr_F^6 + dr_C^8)$ operations in the QTT-Tucker format. By the same reasons, we could not use the TT format without the quantization.

The QTT-Tucker `dmrg` complexity scales as $\mathcal{O}(dLr_F^4 + dr_C^6)$, and the storage as $\mathcal{O}(dLr_F^3 + dr_C^4)$, which makes the highly accurate experiments feasible. The DMRG method benefits from the quantization in all cases, though the asymptotic memory consumptions of the QTT-Tucker and ETT formats are the same. The latter phenomenon indicates that the largest tensor rank is r_C, located in the Tucker core. However, the DMRG visits the corresponding core blocks only d times, while Ld steps involve the factor blocks, and their quantization is enough to make the whole procedure faster.

Comparing different formats, we conclude that the QTT-Tucker approach could be more than 10 times faster than the methods based on the linear QTT format.

Figure 5.14: Hookean 12D example. Memory (left) and CPU time in seconds (right) vs. $-\log_{10}(\varepsilon)$.



5.2.2 Hookean + repulsive potential

Now consider a more complex model, presented in [233]: the quadratic Hookean potential perturbed by the term representing the repulsion between beads. The Fokker-Planck equation (1.4) reads

$$\frac{\partial \psi(\boldsymbol{q},t)}{\partial t} = -\sum_{k=1}^{d} \frac{\partial}{\partial \boldsymbol{q}_k} \left((\nabla_{\boldsymbol{y}} \boldsymbol{v}) \boldsymbol{q}_k - \frac{1}{4} \sum_{m=1}^{d} A_{k,m} \frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}_m} \right) \psi + \sum_{k,m=1}^{d} \frac{1}{4} A_{k,m} \frac{\partial^2}{\partial \boldsymbol{q}_k \partial \boldsymbol{q}_m} \psi,$$

where

$$V = \frac{1}{2}(|\boldsymbol{q}_1|^2 + \ldots + |\boldsymbol{q}_d|^2) + \frac{\eta}{\sigma^3} \sum_{k=1}^{d} \sum_{m=k}^{d} \exp\left(-\frac{|\boldsymbol{q}_k + \boldsymbol{q}_{k+1} + \cdots + \boldsymbol{q}_m|^2}{2\sigma^2} \right)$$

is the total potential with the particular repulsion parameters $\eta = 0.1$, $\sigma = 0.5$ in our case.

In the latter potential term, $\boldsymbol{q}_k + \cdots + \boldsymbol{q}_m$ mirrors the distance between the k-th and $m+1$-th beads. Since the exponential is stretched along the surfaces $q_k^p + \cdots + q_m^p = c$, its representation in the TT format would require extremely large ranks, and we cannot yet solve the corresponding Fokker-Planck equation for $d > 1$. It was just noted in [47] that the components of the stress tensor (1.6), computed via the Quasi Monte Carlo method in [233], do not differ significantly from the values delivered by the Hookean model (1.7), i.e. the repulsive potential is indeed a perturbation.

For $d = 1$ case, known as the *dumbbell* system (only two beads connected by one spring), the Fokker-Planck equation becomes $D = 3$-dimensional, and the repulsive potential resolves to $\exp(-\frac{|\boldsymbol{q}|^2}{2\sigma^2})$, with the rank-1 TT representation. The velocity gradient in this experiment corresponds to the shear flow (5.5) with $\beta = 1$.

We compare the simultaneous space-time discretization (1.16) with the traditional Crank-Nicolson scheme (1.15) with small time steps. We discretize the Fokker-Planck equation with finite differences and the following parameters:

- computational domain $\Omega = [-10, 10]^3$,
- time interval $\hat{T} = 10$,
- tensor rounding threshold $\varepsilon = 10^{-6}$.

We will vary the spatial L_q and temporal d_t QTT dimensions, corresponding to the univariate grid size $n = 2^{L_q}$, mesh step $h = 20/(2^{L_q}+1)$, number of time steps $N_t = 2^{d_t}$, and time step length $\delta t = 10/2^{d_t}$.

As a resulting output, we compute the viscosity components of the stress tensor (1.6),

$$\eta(t) = -\frac{\tau_{12}}{\beta}, \quad \Psi(t) = -\frac{\tau_{11} - \tau_{22}}{\beta^2}, \tag{5.6}$$

Figure 5.15: Repulsion 3D example, evolution of the stress components (5.6) in time. Left: $\eta(t)$, right: $\Psi(t)$.

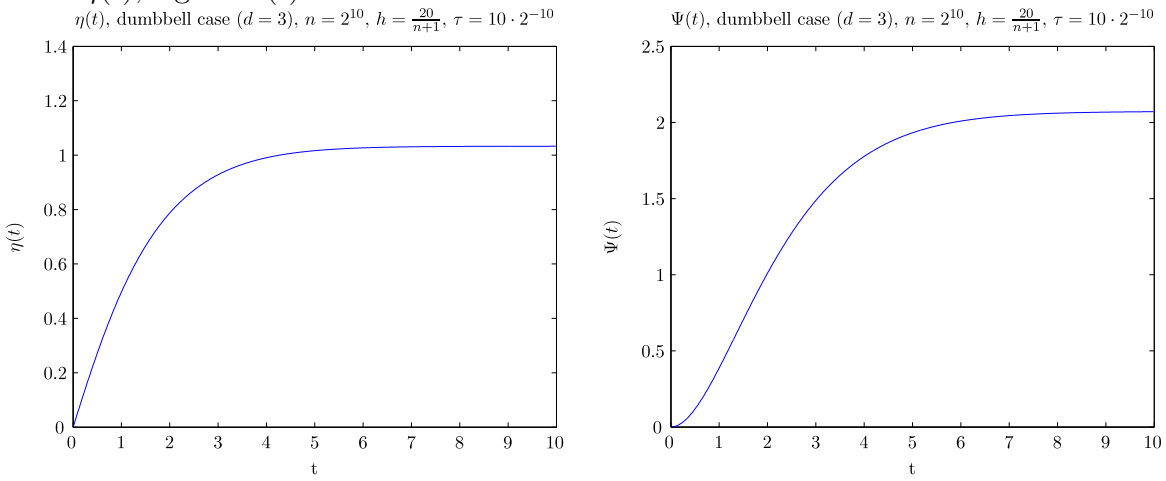


Table 5.6: Repulsion 3D example, $\eta(\hat{T})$ and $\Psi(\hat{T})$ vs. the spatial dimension L_q.

L_q	$\eta(\hat{T})$	$\Psi(\hat{T})$
4	**1.03**88419	**2.0**84148
5	**1.032**0981	**2.0**69552
6	**1.032**6395	**2.07**0764
7	**1.032**7829	**2.071**111
8	**1.0328**263	**2.071**209
9	**1.0328**125	**2.071**143

which are shown in Fig. 5.15.

To ensure that the finite difference scheme resolves the diffusion-convection equation properly, consider first the approximation order w.r.t. h. In Table 5.6, we vary L_q, while the temporal dimension is fixed to $d_t = 8$, corresponding to $\delta t \approx 0.04$. The relevant digits are emphasized with the boldface. The order may be detected more clearly from Fig. 5.16, where we plot the errors $\eta_{L_q}(\hat{T}) - \eta_9(\hat{T})$ and $\Psi_{L_q}(\hat{T}) - \Psi_9(\hat{T})$ vs. L_q. Linear least squares fitting reveals the numerical orders $h^{2.5}$ for η, and $h^{2.8}$ for Ψ, i.e. the approximation rate $\mathcal{O}(h^2)$ is maintained. On finer grids (cf. $\eta_8(\hat{T}) - \eta_9(\hat{T}) \approx 1.4 \cdot 10^{-5}$), the error is governed by the tensor approximation accuracy, rather than the discretization issues.

Now we present the CPU times of the step-by-step time integration via the Crank-Nicolson scheme (1.15) and the two-site DMRG Alg. 9, see Fig. 5.17. We see that the complexity is sublinear in the spatial grid size due to the QTT format ($\log \text{time} \sim L_q = \log n$). However, the growth is linear with respect to the number of time steps, as expected.

The logarithmic complexity w.r.t. the number of time steps can be achieved via the block time scheme (1.16). The interval $\hat{T} = 10$ is split into 8 equal subintervals according to Remark 1.3.3, i.e. we solve (1.16) sequentially in $[0, T], [T, 2T]$ and so on with $T = 1.25$. The computational times, required to solve the system in each

Figure 5.16: Repulsion 3D example, grid approximation of η (left) and Ψ (right) vs. h.

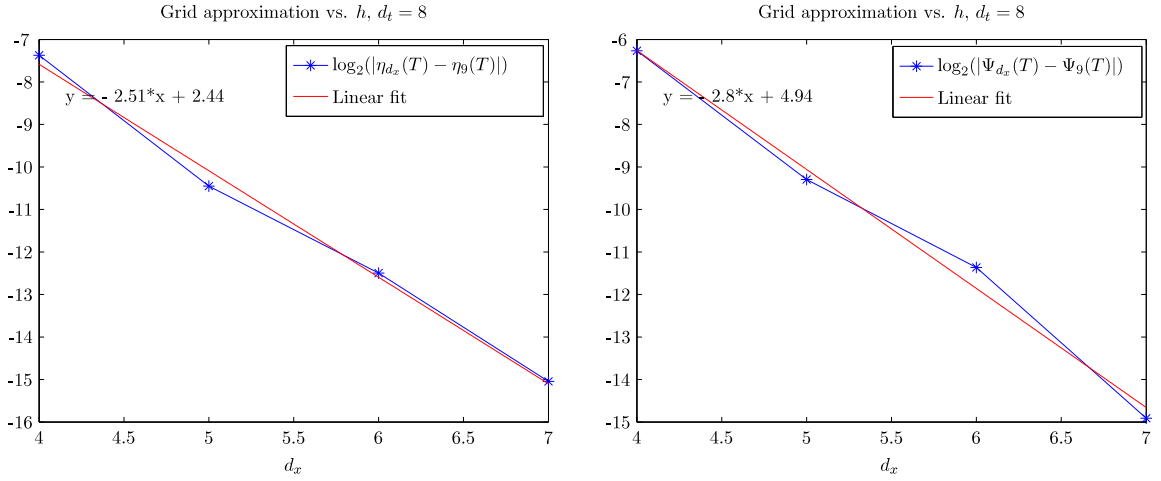
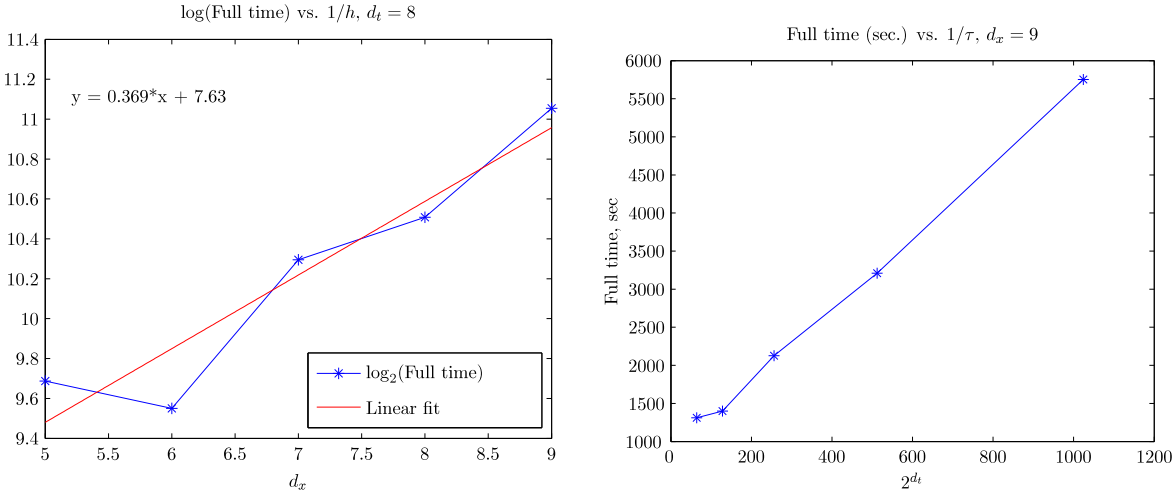


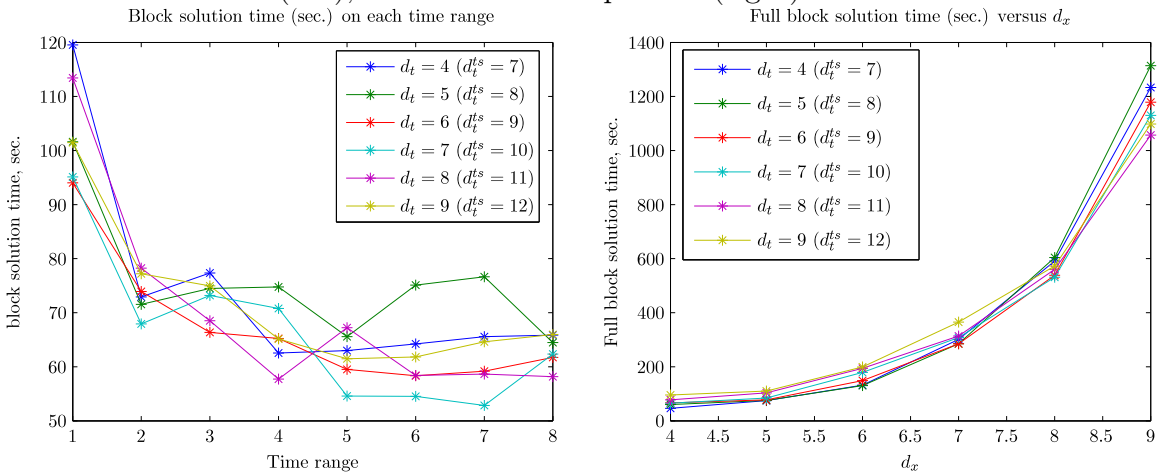
Figure 5.17: Repulsion 3D example, CPU times in seconds of the stepwise scheme (1.15). Left: log time vs. L_q. Right: time vs. $N_t = 2^{d_t}$.



subinterval are shown in Fig. 5.18 (left), and the total CPU time of the solution scheme w.r.t. the number of time steps is investigated in Fig. 5.18 (right). Note that the temporal QTT dimension d_t is defined in Fig. 5.18 for *each* subinterval; it means that the equivalent number of time steps in the previous experiment is 8 times larger. The corresponding QTT dimensions are shown in Fig. 5.18 as d_t^{ts} for convenience.

The main phenomenon that we observe in both parts of Fig. 5.18 is a very mild dependence of the complexity on the number of time steps. Surprisingly, the CPU time does not even necessarily increase with d_t: for example, the fastest computations are achieved at $d_t \sim 7$, but not the minimal value $d_t = 4$. This may be accounted for the balance between the tensor rounding and discretization errors. The exact solution is smooth, and its TT and QTT ranks are moderate according to the polynomial approximation Theorem 2.2.3. However, for small d_t, the solution is perturbed by a large discretization error, which increases the TT ranks. On finer grids the noise level falls below the tensor rounding tolerance, and the TT ranks stabilize at the proper

Figure 5.18: Repulsion 3D example, CPU times in seconds of the block scheme (1.16) on each subinterval (left), and of the whole process (right).



values.

With respect to the spatial grid size, the complexity remains sublinear, and the overall procedure overcomes significantly the previous time stepping scheme. For example, if we extrapolate Fig. 5.17 (right) to $d_t^{ts} = 12$, we may estimate the computational time of the time stepping scheme as ~ 22000 seconds, while the quantization scheme is almost 20 times faster, as we observe from Fig. 5.18.

The decrease of the CPU time on the latter subintervals (Fig. 5.18, left) is a natural consequence of the convergence towards the steady state. The initial solution occupies a wide spectral interval of the Fokker-Planck operator, but the spectral components decay with different rates in time, such that only the lowest eigenstate remains at $t \to \infty$. Therefore, the TT ranks of the solution are large in the first subintervals, and decrease near the end of the process, and so do the computational times.

5.2.3 High-dimensional FENE model

In the last example, we consider a more complicated FENE model from Section 1.1.3 for $d = 4$ springs in $D = 2$ physical space, such that the total Fokker-Planck equation is of dimension $dD = 8$. The velocity gradient, corresponding to the shear flow regime is chosen as follows,

$$\nabla_{\boldsymbol{y}} \boldsymbol{v} = \begin{bmatrix} 0 & 0.8 \\ 0 & 0 \end{bmatrix}.$$

Such a model was considered in [173, 35, 5, 53]. In particular, the latter paper consider the application of the AMEn algorithm to the linear system arising from the model (1.4) with the FENE forces, which we reconstruct here.

As was discussed in Sec. 1.1.3, the solution is posed in a product of balls, and to discretize the spatial operator, we rewrite the Fokker-Planck equation in polar coordinates and use Chebyshev spectral elements in the radial coordinates, and periodic Sinc spectral differentiation in the angular variables. We will vary the number of spectral elements in each radial direction n_r, which governs also the number of angular points as $2n_r$. With typical values $n_r \sim 20$, the solution is a tensor of size $\sim 10^{12}$. Moreover, the spectral differentiation matrices are dense, and therefore the problem is intractable in the full format. The QTT format is not efficient either, since the matrices require maximal possible QTT ranks for any reasonable accuracy. So, we use the 8-dimensional TT representation with mode sizes $n_r \times 2n_r \times \cdots \times n_r \times 2n_r$.

We are interested in the stationary solution of (1.4), and therefore use the implicit Euler iterations (1.22), rewritten as

$$(M - \delta t A)\psi(t_p) = M\psi(t_{p-1}), \quad t_p = p\delta t, \quad p = 1, \ldots, N_t, \tag{5.7}$$

where M is the mass matrix, which is not identity in this case, and A is the stiffness matrix. The time integration is performed until $\hat{T} = \delta t N_t = 20$, which is enough to approximate the steady state with a satisfactory accuracy. As the (unnormalized) initial state, we choose the Maxwellian

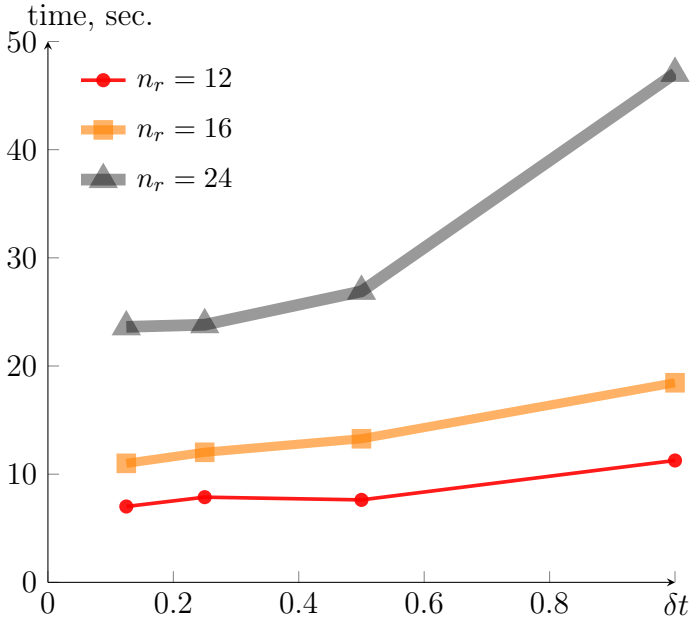
$$\psi(\boldsymbol{q}, 0) = \bigotimes_{i=1}^{4} \left(1 - \frac{|\boldsymbol{q}_i|^2}{b}\right)^{b/2} = \bigotimes_{i=1}^{4} \left(1 - \frac{r_i^2}{b}\right)^{b/2},$$

which also obeys the Fokker-Planck equation with $\nabla_{\boldsymbol{y}} \boldsymbol{v} = 0$. Note that the Maxwellian is a TT rank-1 function in the polar coordinates.

As in the previous cases, we are interested in the components of the Kramer stress tensor (1.6), which is computable from the discretized function as a scalar product in the TT format. The evolution of the stress components with time steps is presented in Fig. 5.20 (left). Notice, however, that the step size δt is rather large in our experiment, and the intermediate values in Fig. 5.20 do not approximate the exact evolution well. The main point that we observe is the qualitative correctness, in particular the growth in the beginning of the process, and stabilization at larger times. Since δt is not a physical quantity, but a parameter of the computational method, we conduct an additional investigation w.r.t. δt.

As the computational method for (5.7), we use the AMEn_{als} Algorithm 12 with the Frobenius-norm threshold $\varepsilon = 10^{-4}$, and the enrichment rank $\rho = 3$. The solution

Figure 5.19: FENE example, CPU time of one Euler iteration vs. time step and grid size



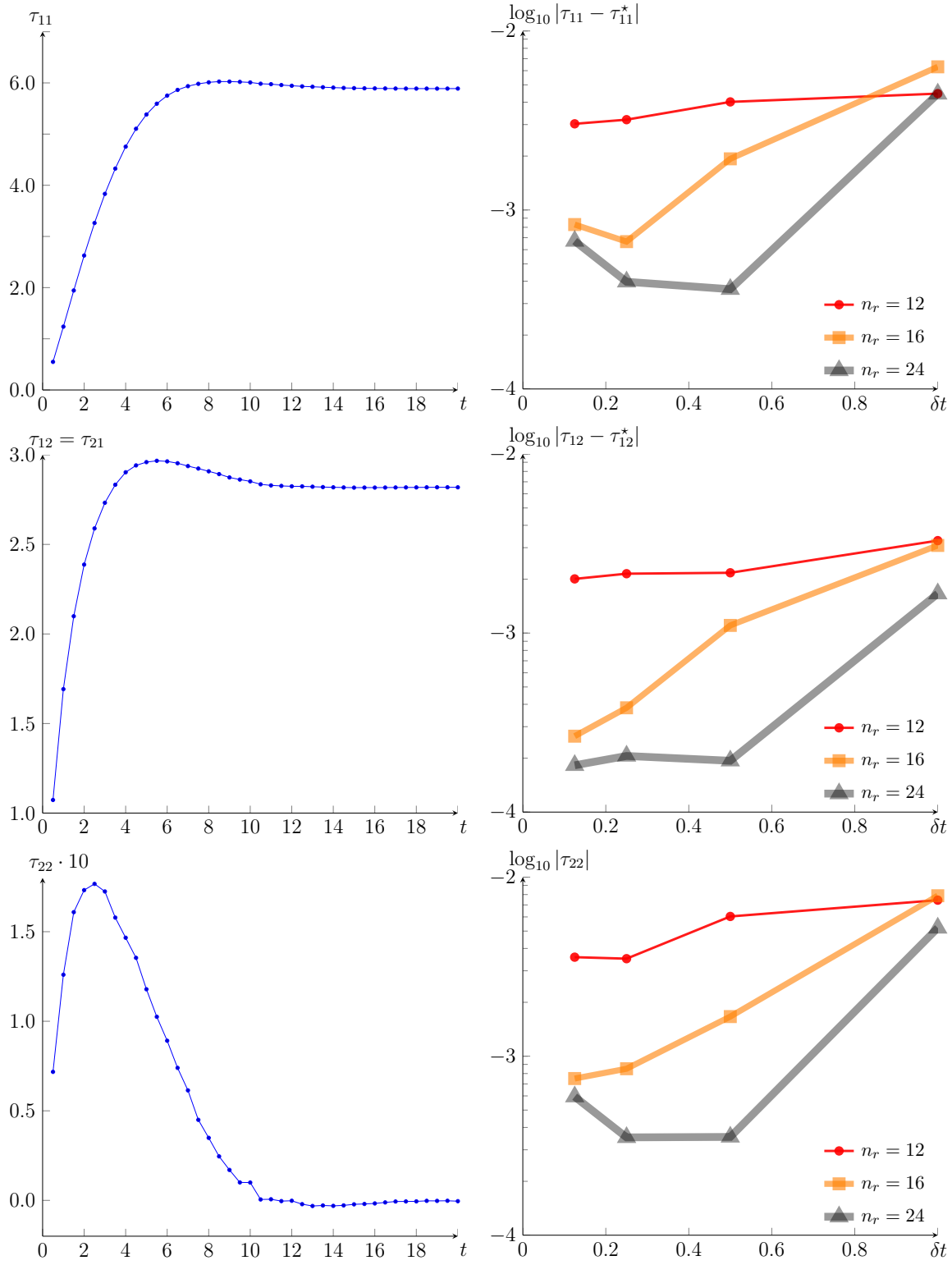
from the previous Euler step serves as a good initial guess for the AMEn algorithm. Since both the mode sizes (up to 48) and TT ranks (up to 73) in this example are relatively large, we consider only the AMEn_{als} method.

The accuracies of the stationary stress values are shown in Fig. 5.20 (right). The reference values $\tau_{11}^\star$, $\tau_{12}^\star$ were computed with the parameters $n_r = 28$ and $\varepsilon = 10^{-5}$. The exact value (from the analytical equation) of the component τ_{22} is zero, so we may use just the modulus $|\tau_{22}|$ as an in-hand measure of the accuracy.

We observe that for all δt except 1, the stress errors decay rapidly with n_r until the limit $\mathcal{O}(10^{-4})$ of the tensor approximation. This shows that accurate solutions can be computed by the tensor product methods with a reasonable cost, while the typical accuracies of greedy or Monte Carlo methods for many-spring models lie at the level $\mathcal{O}(10^{-1})$ [5, 233].

The computational times can be seen in Fig. 5.19. As expected, the complexity increases quadratically with the number of spectral elements n_r. An interesting feature is that the total CPU time is relatively stable w.r.t. the time step δt. It is easy to see that the condition number of the matrix in (5.7) grows linearly with δt, and hence we could expect a deteriorating convergence, according to Theorem 4.3.7. Contrarily, the complexity scales much milder with δt, i.e. the AMEn algorithm demonstrate a behavior, similar to direct methods. However, the quality of the initial guess is crucial: as in the previous section, the first Euler iterations are more difficult than the final ones, where the solutions at succeeding time steps are close. This may motivate attempts to relate the AMEn to Newton or Krylov methods in a future research.

Figure 5.20: FENE example. Left: evolution of the stress tensor components in time. Right: accuracy of the stationary stress tensor components vs. the time step and the grid size. From top to bottom: the components τ_{11}, τ_{12}, τ_{22}, respectively.



Conclusion

We have proposed and investigated tensor product representations and algorithms for approximation of functions and operators and linear system solution in high dimensions. Particularly intriguing high-dimensional models arise in the form of time-dependent differential equations, coming from stochastic description of dynamical systems and chemical kinetics. For a long time, direct simulation of the probability density function via the Fokker-Planck and master equations was avoided, except low-dimensional cases, due to the curse of dimensionality. General data-sparse formats, encapsulating indirectly all n^d elements, open a way to fast and accurate methods of stochastic modeling in engineering, biology, chemistry and physics.

The era of tensor product computations has begun several times in different communities. Abstract description and data fitting traces back to 1927, robust closed tensor formats to 1960's, computation of the ground state wavefunction in quantum physics, using the separable representations, was started at least in 1970's, and many brilliant ideas and practical methods emerged afterwards. In the numerical linear algebra, low-rank matrix factorizations exist, perhaps, for more than a century. However, three- and higher-dimensional tools for the solution of PDEs and other function-related purposes have appeared quite recently, and the DMRG insights from quantum physics were presented to the attention of the multilinear tensor community only in 2009–2010.

Since then, rapid development of the tensor product conception is maintained worldwide. More applications reveal more questions and difficulties, which motivate further elaboration of computational and analytical methods. Many fruitful techniques arise from the combination of several approaches. For example, the multigrid interpolation was used to supply a good initial guess for the alternating optimization in the Tucker format [151]. In the current work, following [44], the Tucker format was extended to the QTT-Tucker decomposition. Certain classes of high-dimensional functions and operators can be written explicitly in a low-rank format. There are especially elegant exact representations, for example for the bilinear form, see Chapter 3, agglomerating [44, 46, 47].

Significant progress is made in computational algorithms for time-dependent equations. A conception of evolution in time was put into the factorized representation with the aid of the curvilinear geometry. On the other hand, why cannot the time variable be considered just as another dimension, followed by the standard discretization, resulting in a linear system? We showed that this approach, combined with the QTT approximation in time variable, is indeed beneficial, and provides accurate time histories of long evolutions with the logarithmic complexity in the number of time layers [46, 47].

Such linear systems are strongly non-symmetric, and a solution technique must admit a wide generality and robustness. And such a technique was developed by bridging the DMRG optimization schemes and the classical steepest descent method [52, 53, and the current work].

The dissertation presents extensive numerical experiments with relevant examples of biological and mechanical systems. We have confirmed that the progress in applications became possible after the progress in computational methods: neither the alternating DMRG algorithms, nor the classical iterations alone are capable to deal with the suggested problems, but together they demonstrate a remarkable cooperation, smoothly delivering an accurate solution even in substantially high-dimensional cases. Up to the best of our knowledge, this is the first example when a high-dimensional tool is efficient practically, and in the same time is supported by the convergence theory.

Several directions of future work may emanate from the proposed framework. The combination of the one-site update and basis enrichment steps can be applied to other problems – first of all, naturally, to the high-dimensional eigenvalue problem. The DMRG/MPS schemes were originally developed to solve this problem for quantum many-body system, and they are implemented in several well-established numerical packages for quantum physics computations. So we have enough possibilities to compare the new methods with the state-of-the-art tools developed in the DMRG/MPS community. This work was started in [48, 54, 161, 102].

Theoretical understanding of tensor product methods is still far from being complete. For example, there is a certain mismatch between the theoretical convergence estimates for the AMEn algorithm, which are at the level of the one-step steepest descent algorithm, and the practical convergence pattern, which looks more like the one of the GMRES. This may inspire us to look for possible connections with the iterative methods of Krylov and Newton type, and more thorough analysis of the alternating projection schemes themselves. The use of preconditioners should also be discussed, see [145, 141, 164, 161] for example. Still under the question are the properties of the temporal evolution methods on manifolds generated by tensor formats. There are significant recent advances [175, 174], and we may expect them to continue in future.

To conclude, it is never possible to study and understand the nature completely. We look forward to solving more high-dimensional problems, and are sure that they will bring new comprehension of the advantages and drawbacks of the proposed methods, and open even more directions for a further research.

Bibliography

[1] J. A. Acebrón, M. P. Busico, P. Lanucara, and R. Spigler, *Domain Decomposition Solution of Elliptic Boundary-Value Problems via Monte Carlo and Quasi-Monte Carlo Methods*, SIAM J. Sci. Comput., 27 (2005), pp. 440–457.

[2] I. Affleck, T. Kennedy, E. H. Lieb, and H. Tasaki, *Rigorous results on valence-bond ground states in antiferromagnets*, Phys. Rev. Lett., 59 (1987), pp. 799–802.

[3] I. Alfaro, D. Gonzalez, F. Bordeu, A. Leygue, A. Ammar, E. Cueto, and F. Chinesta, *Real-time in silico experiments on gene regulatory networks and surgery simulation on handheld devices*, Journal of Computational Surgery, 1 (2014).

[4] A. Ammar, E. Cueto, and F. Chinesta, *Reduction of the chemical master equation for gene regulatory networks using proper generalized decompositions*, Int. J. Numer. Meth. Biomed. Engng., 28 (2012), pp. 960–973.

[5] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings, *A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids*, Journal of Non-Newtonian Fluid Mechanics, 139 (2006), pp. 153 – 176.

[6] R. Andreev and C. Tobler, *Multilevel preconditioning and low rank tensor iteration for space-time simultaneous discretizations of parabolic PDEs*, Tech. Rep. 16, SAM, ETH Zürich, 2012.

[7] A. Arkin, J. Ross, and H. McAdams, *Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected Escherichia coli cells*, Genetics, 149 (1998), pp. 1633–1648.

[8] B. W. Bader and T. G. Kolda, *Efficient MATLAB computations with sparse and factored tensors*, SIAM J. Sci. Comput., 30 (2007), pp. 205–231.

[9] J. Ballani and L. Grasedyck, *A projection method to solve linear systems in tensor format*, Numerical Linear Algebra with Applications, 20 (2013), pp. 27–43.

[10] J. Ballani and L. Grasedyck, *Tree adaptive approximation in the hierarchical tensor format*, RWTH preprint 141, 2013.

[11] J. Ballani, L. Grasedyck, and M. Kluge, *Black box approximation of tensors in hierarchical Tucker format*, Linear Alg. Appl., 428 (2013), pp. 639–657.

[12] A. Barth, C. Schwab, and N. Zollinger, *Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients*, Numerische Mathematik, 119 (2011), pp. 123–161.

[13] R. J. Bartlett and M. Musiał, *Coupled-cluster theory in quantum chemistry*, Reviews of Modern Physics, 79 (2007), p. 291.

[14] F. L. Bauer and A. S. Householder, *Some inequalities involving the euclidian condition of a matrix*, Numerische Mathematik, 2 (1960), pp. 308–311.

[15] M. Bebendorf, *Adaptive cross approximation of multivariate functions*, Constructive approximation, 34 (2011), pp. 149–179.

[16] R. E. Bellman, *Dynamic programming*, Princeton University Press, 1957.

[17] P. Benner and T. Breiten, *Low rank methods for a class of generalized lyapunov equations and related issues*, Numerische Mathematik, 124 (2013), pp. 441–470.

[18] P. Benner, P. Kürschner, and J. Saak, *Self-generating and efficient shift parameters in ADI methods for large Lyapunov and Sylvester equations*, MPI Magdeburg Preprint 13-18, 2013.

[19] P. Benner, A. Onwunta, and M. Stoll, *Low rank solution of unsteady diffusion equations with stochastic coefficients*, MPI Preprint 13, 2013.

[20] S. Bernstein, *Lecons sur les propriétés extrémales et la meilleure approximation des fonctions analytiques d´une variable réelle*, Paris: Gauthier-Villars, 1926.

[21] C. Bertoglio and B. N. Khoromskij, *Low-rank quadrature-based tensor approximation of the Galerkin projected Newton/Yukawa kernels*, Computer Phys. Comm., 183 (2012), pp. 904–912.

[22] G. Beylkin, G. Fann, Z. Gan, R. Harrison, and T. Yanai, *Multiresolution quantum chemistry: basic theory and initial applications*, J. Chem. Phys, 121 (2004), pp. 11587–11598.

[23] G. Beylkin and M. J. Mohlenkamp, *Numerical operator calculus in higher dimensions*, Proc. Nat. Acad. Sci. USA, 99 (2002), pp. 10246–10251.

[24] G. Beylkin and M. J. Mohlenkamp, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., 26 (2005), pp. 2133–2159.

[25] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk, *Convergence rates for greedy algorithms in reduced basis methods*, SIAM J. Math. Anal., 43 (2011), pp. 1457–1472.

[26] R. Bird, C. Curtiss, R. Armstrong, and O. Hassager, *Dynamics of Polymeric Liquids, Kinetic Theory*, Wiley, 1987.

[27] T. Breiten, V. Simoncini, and M. Stoll, *Fast iterative solvers for fractional differential equations*, MPI Magdeburg Preprint 14-02, 2014.

[28] R. Bro, *PARAFAC: Tutorial and applications*, Chemometrics and Intelligent Lab. Syst., 38 (1997), pp. 149–171.

[29] M. Buhmann, *Multivariate cardinal interpolation with radial-basis functions*, Constr. Approx., 6 (1990), pp. 225–255.

[30] M. Buhmann, *Radial basis functions*, Acta Numerica, 9 (2000), pp. 1–38.

[31] H.-J. Bungatrz and M. Griebel, *Sparse grids*, Acta Numerica, 13 (2004), pp. 147–269.

[32] B. R. Butchart, *An explicit solution to the fokker-planck equation for an ordinary differential equation*, International Journal of Control, 1 (1965), pp. 201–208.

[33] E. Cancés, V. Ehrlacher, and T. Leliévre, *Convergence of a greedy algorithm for high-dimensional convex nonlinear problems*, Mathematical Models and Methods in Applied Sciences, 21 (2011), pp. 2433–2467.

[34] J. D. Caroll and J. J. Chang, *Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart–Young decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[35] C. Chauviére and A. Lozinski, *Simulation of dilute polymer solutions using a Fokker-Planck equation*, Computers & Fluids, 33 (2004), pp. 687–696.

[36] S. R. Chinnamsetty, M. Espig, W. Hackbusch, B. N. Khoromskij, and H. J. Flad, *Tensor product approximation with optimal rank in quantum chemistry*, J. Chem. Phys., 127 (2007), pp. 84–110.

[37] A. Cohen, R. DeVore, and C. Schwab, *Convergence rates of best N-term Galerkin approximations for a class of elliptic sPDEs*, Found. Comput. Math, 10 (2010), pp. 615–646.

[38] P. Comon, *Tensor decomposition: state of the art and applications*, in IMA Conf. Math. in Sig. Proc., Warwick, UK, 2000.

[39] L. de Lathauwer, B. de Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.

[40] L. de Lathauwer, B. de Moor, and J. Vandewalle, *On best rank-1 and rank-($R_1, R_2, ..., R_N$) approximation of high-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342.

[41] L. De Lathauwer and J. Vandewalle, *Dimensionality reduction in higher-order signal processing and rank-$(R_1, R_2, ..., R_N)$ reduction in multilinear algebra*, Linear Algebra Appl., 391 (2004), pp. 31–55.

[42] V. de Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127.

[43] S. Dolgov and B. Khoromskij, *Simultaneous state-time approximation of the chemical master equation using tensor product formats*, arXiv 1311.3143 (to appear in NLAA, 2014), 2013.

[44] S. Dolgov and B. Khoromskij, *Two-level QTT-Tucker format for optimized tensor calculus*, SIAM J. on Matrix An. Appl., 34 (2013), pp. 593–623.

[45] S. V. Dolgov, *TT-GMRES: solution to a linear system in the structured tensor format*, Russ. J. Numer. Anal. Math. Modelling, 28 (2013), pp. 149–172.

[46] S. V. Dolgov and B. N. Khoromskij, *Tensor-product approach to global time-space-parametric discretization of chemical master equation*, Preprint 68, MPI MIS, 2012.

[47] S. V. Dolgov, B. N. Khoromskij, and I. V. Oseledets, *Fast solution of multi-dimensional parabolic problems in the tensor train/quantized tensor train–format with initial application to the Fokker-Planck equation*, SIAM J. Sci. Comput., 34 (2012), pp. A3016–A3038.

[48] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov, *Computation of extreme eigenvalues in higher dimensions using block tensor train format*, Computer Phys. Comm., 185 (2014), pp. 1207–1216.

[49] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and E. E. Tyrtyshnikov, *Low-rank tensor structure of solutions to elliptic problems with jumping coefficients*, J. Comput. Math., 30 (2012), pp. 14–23.

[50] S. V. Dolgov, B. N. Khoromskij, and D. V. Savostyanov, *Superfast Fourier transform using QTT approximation*, J. Fourier Anal. Appl., 18 (2012), pp. 915–953.

[51] S. V. Dolgov and I. V. Oseledets, *Solution of linear systems and matrix inversion in the TT-format*, SIAM J. Sci. Comput., 34 (2012), pp. A2718–A2739.

[52] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions. Part I: SPD systems*, arXiv preprint 1301.6068, 2013.

[53] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions. Part II: Faster algorithm and application to nonsymmetric systems*, arXiv preprint 1304.1222, 2013.

[54] S. V. Dolgov and D. V. Savostyanov, *Corrected one-site density matrix renormalization group and alternating minimal energy algorithm*, in Proc. ENUMATH 2013, accepted, 2014.

[55] I. Domanov, *Study of Canonical Polyadic Decomposition of Higher-Order Tensors*, PhD thesis, 2013.

[56] G. W. F. Drake, *High precision theory of atomic helium*, Physica Scripta, 1999 (1999), p. 83.

[57] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition*, SIAM J Comput, 36 (2006), pp. 184–206.

[58] T. H. Dunning Jr, *Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen*, The Journal of Chemical Physics, 90 (1989), p. 1007.

[59] M. Espig, L. Grasedyck, and W. Hackbusch, *Black box low tensor rank approximation using fibre-crosses*, Constr. Appr., 30 (2009), pp. 557–597.

[60] M. Espig and W. Hackbusch, *A regularized Newton method for the efficient approximation of tensors represented in the canonical tensor format*, Numer. Math., 122 (2012), pp. 489–525.

[61] M. Espig, W. Hackbusch, A. Litvinenko, H. Matthies, and E. Zander, *Efficient analysis of high dimensional data in tensor formats*, in Sparse Grids and Applications, Springer, 2013, pp. 31–56.

[62] M. Fannes, B. Nachtergaele, and R. Werner, *Finitely correlated states on quantum spin chains*, Comm. Math. Phys., 144 (1992), pp. 443–490.

[63] M. Fannes, B. Nachtergaele, and R. F. Werner, *Ground states of VBS models on Cayley trees*, J. Stat. Phys., 66 (1992), pp. 939–973.

[64] L. E. Figueroa and E. Süli, *Greedy approximation of high-dimensional ornstein–uhlenbeck operators*, Foundations of Computational Mathematics, 12 (2012), pp. 573–623.

[65] G. S. Fishman, *Monte Carlo: concepts, algorithms, and applications*, vol. 1196, Springer New York, 1996.

[66] H.-J. Flad, B. N. Khoromskij, D. V. Savostyanov, and E. E. Tyrtyshnikov, *Verification of the cross 3D algorithm on quantum chemistry data*, Rus. J. Numer. Anal. Math. Model., 23 (2008), pp. 329–344.

[67] A. D. Fokker, *Die mittlere Energie rotierender elektrischer Dipole im Strahlungsfeld*, Annalen der Physik, 348 (1914), pp. 810–820.

[68] J. Garcke, M. Griebel, and M. Thess, *Data mining with sparse grids*, Computing, 67 (2001), pp. 225–253.

[69] T. Gardner, C. Cantor, and J. Collins, *Construction of a genetic toggle switch in Escherichia coli*, Nature, 403 (2000), pp. 339–342.

[70] L. Gauckler and H. Yserentant, *Regularity and approximability of the solutions to the chemical master equation*, Matheon Preprint 1010, 2013.

[71] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *$\mathcal{H}$-Matrix approximation for the operator exponential with applications*, Numerische Mathematik, 92 (2002), pp. 83–111.

[72] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *Tensor-product approximation to the inverse and related operators in high-dimensional elliptic problems*, Computing, (2005), pp. 131–157.

[73] I. P. Gavrilyuk and B. N. Khoromskij, *Quantized-TT-Cayley transform for computing the dynamics and the spectrum of high-dimensional Hamiltonians*, Comput. Methods in Appl. Math., 11 (2011), pp. 273–290.

[74] D. Gillespie, *A general method for numerically simulating the stochastic time evolution of coupled chemical reactions*, J Comput. Phys., 22 (1976), pp. 403–434.

[75] D. Gillespie, *Approximate accelerated stochastic simulation of chemically reacting systems*, The Journal of Chemical Physics, 115 (2001), p. 1716.

[76] D. Gillespie, *The chemical langevin and fokker-planck equations for the reversible isomerization reaction*, The Journal of Physical Chemistry A, 106 (2002), pp. 5063–5071.

[77] D. T. Gillespie, *A rigorous derivation of the chemical master equation*, Physica A: Statistical Mechanics and its Applications, 188 (1992), pp. 404 – 425.

[78] D. T. Gillespie, *The chemical langevin equation*, The Journal of Chemical Physics, 113 (2000), pp. 297–306.

[79] L. Giraldi, A. Nouy, and G. Legrain, *Low-rank approximate inverse for preconditioning tensor-structured linear systems*, arXiv preprint 1304.6004, 2013.

[80] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205–224.

[81] G. Golub and C. Van Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, MD, 1996.

[82] S. A. Goreinov, I. V. Oseledets, and D. V. Savostyanov, *Wedderburn rank reduction and Krylov subspace method for tensor approximation. Part 1: Tucker case*, SIAM J. Sci. Comput., 34 (2012), pp. A1–A27.

[83] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *How to find a good submatrix*, Research Report 08-10, ICM HKBU, Kowloon Tong, Hong Kong, 2008.

[84] S. A. Goreinov and E. E. Tyrtyshnikov, *The maximal-volume concept in approximation by low-rank matrices*, Contemporary Mathematics, 208 (2001), pp. 47–51.

[85] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *Pseudo–skeleton approximations of matrices*, Reports of Russian Academy of Sciences, 342 (1995), pp. 151–152.

[86] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *A theory of pseudo–skeleton approximations*, Linear Algebra Appl., 261 (1997), pp. 1–21.

[87] S. A. Goreinov, N. L. Zamarashkin, and E. E. Tyrtyshnikov, *Pseudo–skeleton approximations by matrices of maximum volume*, Mathematical Notes, 62 (1997), pp. 515–519.

[88] J. Goutsias, *Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems*, The Journal of chemical physics, 122 (2005), p. 184102.

[89] L. Grasedyck, *Existence and computation of low Kronecker-rank approximations for large systems in tensor product structure*, Computing, 72 (2004), pp. 247–265.

[90] L. Grasedyck, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2029–2054.

[91] L. Grasedyck, *Polynomial approximation in hierarchical Tucker format by vector-tensorization*, DFG-SPP1324 Preprint 43, Philipps-Univ., Marburg, 2010.

[92] L. Grasedyck and W. Hackbusch, *An introduction to hierarchical ($\mathcal{H}$-) and TT-rank of tensors with examples*, Comput. Meth. Appl. Math., 3 (2011), pp. 291–304.

[93] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitteilungen, 36 (2013), pp. 53–78.

[94] M. Griebel, *Sparse grids and related approximation schemes for higher dimensional problems*, SFB 611, 2005.

[95] M. Griebel and H. Harbrecht, *Approximation of bi-variate functions: singular value decomposition versus sparse grids*, IMA Journal of Numerical Analysis, (2013).

[96] M. Griebel and D. Oeltz, *A sparse grid space-time discretization scheme for parabolic problems*, Computing, 81 (2007), pp. 1–34.

[97] A. Gupta and M. Khammash, *Determining the long-term behavior of cell populations: A new procedure for detecting ergodicity in large stochastic reaction networks*, arXiv 1312.2879, 2013.

[98] W. Hackbusch, *Tensor spaces and numerical tensor calculus*, Springer–Verlag, Berlin, 2012.

[99] W. Hackbusch and D. Braess, *Approximation of $\frac{1}{x}$ by exponential sums in $[1, \infty]$*, IMA J. Numer. Anal., 25 (2005), pp. 685–697.

[100] W. Hackbusch and B. N. Khoromskij, *Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. I. Separable approximation of multi-variate functions*, Computing, 76 (2006), pp. 177–202.

[101] W. Hackbusch and B. N. Khoromskij, *Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. II. HKT representation of certain operators*, Computing, 76 (2006), pp. 203–225.

[102] W. Hackbusch, B. N. Khoromskij, S. A. Sauter, and E. E. Tyrtyshnikov, *Use of tensor formats in elliptic eigenvalue problems*, Numer. Linear Algebra Appl., 19 (2012), pp. 133–151.

[103] W. Hackbusch, B. N. Khoromskij, and E. E. Tyrtyshnikov, *Hierarchical Kronecker tensor-product approximations*, J. Numer. Math., 13 (2005), pp. 119–156.

[104] W. Hackbusch, B. N. Khoromskij, and E. E. Tyrtyshnikov, *Approximate iterations for structured matrices*, Numer. Mathematik, 109 (2008), pp. 365–383.

[105] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, J. Fourier Anal. Appl., 15 (2009), pp. 706–722.

[106] R. A. Harshman, *Foundations of the PARAFAC procedure: models and conditions for an explanatory multimodal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.

[107] W. K. Hastings, *Monte Carlo sampling methods using Markov chains and their applications*, Biometrika, 57 (1970), pp. 97–109.

[108] M. Hegland, C. Burden, L. Santoso, S. MacNamara, and H. Booth, *A solver for the stochastic master equation applied to gene regulatory networks*, Journal of Computational and Applied Mathematics, 205 (2007), pp. 708 – 724.

[109] M. Hegland and J. Garcke, *On the numerical solution of the chemical master equation with sums of rank one tensors*, ANZIAM, 52 (2011), pp. C628–C643.

[110] T. Helgaker, P. Jørgensen, J. Olsen, and M. A. Ratner, *Molecular electronic-structure theory*, Physics Today, 54 (2001), p. 52.

[111] A. Hellander and P. Lötstedt, *Hybrid method for the chemical master equation*, Journal of Computational Physics, 227 (2007), pp. 100–122.

[112] M. Hemberg and M. Barahona, *Perfect sampling of the master equation for gene regulatory networks*, Biophysical journal, 93 (2007), pp. 401–410.

[113] F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, J. Math. Phys, 6 (1927), pp. 164–189.

[114] H. J. Hogben, M. Krzystyniak, G. T. P. Charnock, P. J. Hore, and I. Kuprov, *Spinach — A software library for simulation of spin dynamics in large spin systems*, J Magn. Reson., 208 (2011), pp. 179–194.

[115] S. Holtz, T. Rohwedder, and R. Schneider, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713.

[116] T. Huckle and K. Waldherr, *Subspace iteration method in terms of matrix product states*, Proc. Appl. Math. Mech., 12 (2012), pp. 641–642.

[117] T. Huckle, K. Waldherr, and T. Schulte-Herbrüggen, *Computations in quantum tensor networks*, Linear Algebra Appl., 438 (2013), pp. 750–781.

[118] I. Ibraghimov, *Application of the three-way decomposition for matrix compression*, Numer. Linear Algebra Appl., 9 (2002), pp. 551–565.

[119] M. Ishteva, L. de Lathauwer, P. A. Absil, and S. van Huffel, *Differential-geometric Newton method for the best rank-(r_1, r_2, r_3) approximation of tensors*, Numerical Algorithms, 51 (2009), pp. 179–194.

[120] T. Jahnke, *An adaptive wavelet method for the chemical master equation*, SIAM J. Sci. Comput., 31 (2010), p. 4373.

[121] T. Jahnke, *On reduced models for the chemical master equation*, Multiscale Modeling and Simulation, 9 (2011), pp. 1646–1676.

[122] T. Jahnke and W. Huisinga, *A dynamical low-rank approach to the chemical master equation*, Bulletin of Mathematical Biology, 70 (2008), pp. 2283–2302.

[123] T. Jahnke and M. Kreim, *Error bound for piecewise deterministic processes modeling stochastic reaction systems*, Multiscale Modeling and Simulation, 10 (2012), pp. 1119–1147.

[124] E. Jeckelmann, *Dynamical density–matrix renormalization–group method*, Phys. Rev. B, 66 (2002), p. 045114.

[125] L. V. Kantorovich, *Funktsionallniy analyz i prikladnaya matematika*, Uspehi Mat. Nauk, 3 (1945), pp. 89–185.

[126] V. Kazeev, M. Khammash, M. Nip, and C. Schwab, *Direct solution of the Chemical Master Equation using Quantized Tensor Trains*, PLOS Computational Biology, (2014).

[127] V. Kazeev, B. Khoromskij, and E. Tyrtyshnikov, *Multilevel Toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity*, SIAM J. Sci. Comp., 35 (2013), pp. A1511–A1536.

[128] V. Kazeev, O. Reichmann, and C. Schwab, *hp-DG-QTT solution of high-dimensional degenerate diffusion equations*, Tech. Report 2012-11, ETH SAM, Zürich, 2012.

[129] V. Kazeev, O. Reichmann, and C. Schwab, *Low-rank tensor structure of linear diffusion operators in the TT and QTT formats*, Linear Algebra and its Applications, 438 (2013), pp. 4204–4221.

[130] V. Kazeev and C. Schwab, *Tensor approximation of stationary distributions of chemical reaction networks*, Research Report 18, SAM, ETH Zürich, 2013.

[131] V. A. Kazeev and B. N. Khoromskij, *Low-rank explicit QTT representation of the Laplace operator and its inverse*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 742–758.

[132] V. A. Kazeev and I. V. Oseledets, *The tensor structure of a class of adaptive algebraic wavelet transforms*, Preprint 2013-28, ETH SAM, Zürich, 2013.

[133] V. A. Kazeev and E. E. Tyrtyshnikov, *Structure of the Hessian matrix and an economical implementation of Newton's method in the problem of canonical approximation of tensors*, Comput. Math. Math. Phys., 50 (2010), pp. 927–945.

[134] R. B. Kellogg, *An alternating direction method for operator equations*, SIAM, 12 (1964), pp. 848–854.

[135] R. Keunings, *Micro-macro methods for the multiscale simulation of viscoelastic flow using molecular models of kinetic theory*, Rheology Reviews, (2004), pp. 67–98.

[136] V. Khoromskaia, *Computation of the Hartree-Fock exchange by tensor-structured methods*, Comput. Methd. Appl. Math., 10 (2008).

[137] V. Khoromskaia, *Numerical solution of the Hartree-Fock equation by multilevel tensor-structured methods*, PhD thesis, TU Berlin, 2010.

[138] V. Khoromskaia, *Black-box Hartree–Fock solver by tensor numerical methods*, Computational Methods in Applied Mathematics, 14 (2014), pp. 89–111.

[139] V. Khoromskaia and B. N. Khoromskij, *Grid-based lattice summation of electrostatic potentials by low-rank tensor approximation*, Preprint 116, MPI MIS, 2013.

[140] V. Khoromskaia, B. N. Khoromskij, and R. Schneider, *Tensor-structured factorized calculation of two-electron integrals in a general basis*, SIAM J. Sci. Comput., 35 (2013), pp. A987–A1010.

[141] B. Khoromskij and C. Schwab, *Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs*, SIAM J. Sci. Comp., 33 (2011), pp. 1–25.

[142] B. N. Khoromskij, *Structured rank-$(r_1, \dots, r_d)$ decomposition of function-related operators in $\mathbb{R}^d$*, Comput. Meth. Appl. Math, 6 (2006), pp. 194–220.

[143] B. N. Khoromskij, *On tensor approximation of Green iterations for Kohn-Sham equations*, Computing and visualization in science, 11 (2008), pp. 259–271.

[144] B. N. Khoromskij, *$\mathcal{O}(d \log n)$-Quantics approximation of n-d tensors in high-dimensional numerical modeling*, Preprint 55, MPI MIS, Leipzig, 2009.

[145] B. N. Khoromskij, *Tensor-structured preconditioners and approximate inverse of elliptic operators in $\mathbb{R}^d$*, Constr. Approx, (2009), pp. 599–620.

[146] B. N. Khoromskij, *Fast and accurate tensor approximation of multivariate convolution with linear scaling in dimension*, J. Comp. Appl. Math., 234 (2010), pp. 3122–3139.

[147] B. N. Khoromskij, *Introduction to tensor numerical methods in scientific computing*, Preprint, Lecture Notes 06-2011, University of Zürich, 2010.

[148] B. N. Khoromskij, *$\mathcal{O}(d \log n)$–Quantics approximation of N–d tensors in high-dimensional numerical modeling*, Constr. Appr., 34 (2011), pp. 257–280.

[149] B. N. Khoromskij, *Tensor-structured numerical methods in scientific computing: Survey on recent advances*, Chemometr. Intell. Lab. Syst., 110 (2012), pp. 1–19.

[150] B. N. Khoromskij and V. Khoromskaia, *Low rank Tucker-type tensor approximation to classical potentials*, Central European journal of mathematics, 5 (2007), pp. 523–550.

[151] B. N. Khoromskij and V. Khoromskaia, *Multigrid accelerated tensor approximation of function related multidimensional arrays*, SIAM J. Sci. Comput., 31 (2009), pp. 3002–3026.

[152] B. N. Khoromskij, V. Khoromskaia, S. R. Chinnamsetty, and H.-J. Flad, *Tensor decomposition in electronic structure calculations on 3D Cartesian grids*, J. Comput. Phys., 228 (2009), pp. 5749–5762.

[153] B. N. Khoromskij, V. Khoromskaia, and H.-J. Flad., *Numerical solution of the Hartree–Fock equation in multilevel tensor-structured format*, SIAM J. Sci. Comput., 33 (2011), pp. 45–65.

[154] B. N. Khoromskij and S. Miao, *Superfast wavelet transform using QTT approximation. I: Haar wavelets*, Preprint MPI MIS, Leipzig 103, 2013.

[155] B. N. Khoromskij and I. V. Oseledets, *DMRG+QTT approach to computation of the ground state for the molecular Schrödinger operator*, Preprint 69, MPI MIS, Leipzig, 2010.

[156] B. N. Khoromskij and I. V. Oseledets, *Quantics-TT collocation approximation of parameter-dependent and stochastic elliptic PDEs*, Comput. Meth. Appl. Math., 10 (2010), pp. 376–394.

[157] B. N. Khoromskij and I. V. Oseledets, *QTT-approximation of elliptic solution operators in higher dimensions*, Rus. J. Numer. Anal. Math. Model., 26 (2011), pp. 303–322.

[158] A. Klümper, A. Schadschneider, and J. Zittartz, *Matrix product ground states for one-dimensional spin-1 quantum antiferromagnets*, Europhys. Lett., 24 (1993), pp. 293–297.

[159] O. Koch and C. Lubich, *Dynamical tensor approximation*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2360–2375.

[160] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.

[161] D. Kressner, M. Steinlechner, and A. Uschmajew, *Low-rank tensor methods with subspace correction for symmetric eigenvalue problems*, MATHICSE preprint 40.2013, EPFL, Lausanne, 2013.

[162] D. Kressner and C. Tobler, *Krylov subspace methods for linear systems with tensor product structure*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1688–1714.

[163] D. Kressner and C. Tobler, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 273–290.

[164] D. Kressner and C. Tobler, *Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems*, Computational Methods in Applied Mathematics, 11 (2011), pp. 363–381.

[165] P. Kroonenberg and J. de Leeuw, *Principal component analysis of three-mode data by means of alternating least squares algorithms*, Psychometrika, 45 (1980), pp. 69–97.

[166] S. Kühn, *Hierarchische Tensordarstellung*, PhD thesis, Uni. Leipzig, Faculty of Mathematics and Informatics, 2012.

[167] I. Kuprov, N. Wagner-Rundell, and P. J. Hore, *Polynomially scaling spin dynamics simulation algorithm based on adaptive state-space restriction*, J Magn. Reson., 189 (2007), pp. 241–250.

[168] C. Le Bris, T. Leliévre, and Y. Maday, *Results and questions on a nonlinear approximation approach for solving high-dimensional partial differential equations*, Constr. Approx., 30 (2009), pp. 621–651.

[169] O. S. Lebedeva, *Block tensor conjugate gradient-type method for Rayleigh quotient minimization in two-dimensional case*, Comput. Math. Math. Phys., 50 (2010), pp. 749–765.

[170] O. S. Lebedeva, *Tensor conjugate-gradient-type method for Rayleigh quotient minimization in block QTT-format*, Russ. J. Numer. Anal. Math. Modelling, 26 (2011), p. 465–489.

[171] C. Lécot and F. E. Khettabi, *Quasi-Monte Carlo Simulation of Diffusion*, Journal of Complexity, 15 (1999), pp. 342 – 359.

[172] O. Legeza, T. Rohwedder, R. Schneider, and S. Szalay, *Tensor product approximation DMRG and coupled cluster method in quantum chemistry*, arXiv preprint 1310.2736, 2013.

[173] A. Lozinski and C. Chauviére, *A fast solver for Fokker-Planck equation applied to viscoelastic flows calculations: 2D FENE model*, Journal of Computational Physics, 189 (2003), pp. 607 – 625.

[174] C. Lubich and I. V. Oseledets, *A projector-splitting integrator for dynamical low-rank approximation*, BIT, 54 (2014), pp. 171–188.

[175] C. Lubich, T. Rohwedder, R. Schneider, and B. Vandereycken, *Dynamical approximation by hierarchical Tucker and tensor-train tensors*, SIAM J. Matrix. Anal. Appl., 34 (2013), pp. 470–494.

[176] T. Mach, *Computing inner eigenvalues of matrices in tensor train matrix format*, in Numerical Mathematics and Advanced Applications 2011, Springer Berlin Heidelberg, 2013, pp. 781–788.

[177] H. Matthies and A. Keese, *Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations*, Computer Methods in Applied Mechanics and Engineering, 194 (2005), pp. 1295–1331.

[178] N. Metropolis and S. Ulam, *The monte carlo method*, Journal of the American statistical association, 44 (1949), pp. 335–341.

[179] H.-D. Meyer, F. Gatti, and G. A. Worth, *Multidimensional Quantum Dynamics: MCTDH Theory and Applications*, Wiley-VCH, Weinheim, 2009.

[180] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Review, 45 (2003), pp. 3–49.

[181] B. Munsky and M. Khammash, *The finite state projection algorithm for the solution of the chemical master equation*, The Journal of chemical physics, 124 (2006), p. 044104.

[182] H. Munthe-Kaas, *The convergence rate of inexact preconditioned steepest descent algorithm for solving linear systems*, Numerical Analysis Report NA-87-04, Stanford University, 1987.

[183] N. R. Nené and A. Zaikin, *Decision making in noisy bistable systems with time-dependent asymmetry*, Phys. Rev. E, 87 (2013), p. 012715.

[184] H. Niederreiter, *Quasi–Monte Carlo methods and pseudo–random numbers*, Bull. AMS, 84 (1978), pp. 957–1041.

[185] Y. Notay, *Convergence analysis of inexact rayleigh quotient iteration*, SIAM J. on Matrix An. Appl., 24 (2003), pp. 627–644.

[186] A. Nouy, *A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations*, Computer Methods in Applied Mechanics and Engineering, 199 (2010), pp. 1603–1626.

[187] I. V. Oseledets, *Lower bounds for separable approximations of the Hilbert kernel*, Mat. Sb., 198 (2007), pp. 137–144.

[188] I. V. Oseledets, *Compact matrix form of the d-dimensional tensor decomposition*, Preprint 2009-01, INM RAS, Moscow, 2009.

[189] I. V. Oseledets, *Approximation of $2^d \times 2^d$ matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2130–2145.

[190] I. V. Oseledets, *DMRG approach to fast linear algebra in the TT–format*, Comput. Meth. Appl. Math., 11 (2011), pp. 382–393.

[191] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.

[192] I. V. Oseledets, *Constructive representation of functions in low-rank tensor formats*, Constr. Appr., 37 (2013), pp. 1–18.

[193] I. V. Oseledets, B. N. Khoromskij, and R. Schneider, *Efficient time-stepping scheme for dynamics on TT-manifolds*, Preprint 24, MPI MIS, 2012.

[194] I. V. Oseledets, D. V. Savostianov, and E. E. Tyrtyshnikov, *Tucker dimensionality reduction of three-dimensional arrays in linear time*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 939–956.

[195] I. V. Oseledets, D. V. Savostyanov, and E. E. Tyrtyshnikov, *Linear algebra for tensor problems*, Computing, 85 (2009), pp. 169–188.

[196] I. V. Oseledets, D. V. Savostyanov, and E. E. Tyrtyshnikov, *Cross approximation in tensor electron density computations*, Numer. Linear Algebra Appl., 17 (2010), pp. 935–952.

[197] I. V. Oseledets and E. E. Tyrtyshnikov, *Breaking the curse of dimensionality, or how to use SVD in many dimensions*, SIAM J. Sci. Comput., 31 (2009), pp. 3744–3759.

[198] I. V. Oseledets and E. E. Tyrtyshnikov, *Tensor tree decomposition does not need a tree*, Preprint (Submitted to Linear Algebra Appl) 2009-04, INM RAS, Moscow, 2009.

[199] I. V. Oseledets and E. E. Tyrtyshnikov, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88.

[200] S. Östlund and S. Rommer, *Thermodynamic limit of density matrix renormalization*, Phys. Rev. Lett., 75 (1995), pp. 3537–3540.

[201] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, *Matrix product state representations*, Quantum Info. Comput., 7 (2007), pp. 401–430.

[202] I. Pižorn and F. Verstraete, *Variational Numerical Renormalization Group: Bridging the gap between NRG and Density Matrix Renormalization Group*, Phys. Rev. Lett., 108 (2012).

[203] M. Planck, Sitzber. Preuss. Akad. Wiss., (1917), p. 324.

[204] M. Ptashne, *A genetic switch: λ-phage and higher organisms*, Wiley-Blackwell, 1992.

[205] H. Rabitz, M. Kramer, and D. Dacol, *Sensitivity analysis in chemical kinetics*, Annual review of physical chemistry, 34 (1983), pp. 419–461.

[206] H. Risken, *The Fokker-Planck Equation: Methods of Solutions and Applications, 2nd ed.*, Springer Verlag, Berlin, Heidelberg, 1989.

[207] T. Rohwedder and A. Uschmajew, *On local convergence of alternating schemes for optimization of convex problems in the tensor train format*, SIAM J. Num. Anal., 51 (2013), pp. 1134–1162.

[208] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, 2003.

[209] B. Savas and L. Eldén, *Krylov-type methods for tensor computations I*, Linear Algebra and its Applications, 438 (2013), pp. 891–918.

[210] D. V. Savostyanov, *Polilinear approximation of matrices and integral equations*, PhD thesis, INM RAS, Moscow, 2006. (in Russian).

[211] D. V. Savostyanov, *Fast revealing of mode ranks of tensor in canonical form*, Numer. Math. Theor. Meth. Appl., 2 (2009), pp. 439–444.

[212] D. V. Savostyanov, *Quasioptimality of maximum–volume cross interpolation of tensors*, arXiv preprint 1305.1818, 2013.

[213] D. V. Savostyanov and I. V. Oseledets, *Fast adaptive interpolation of multi-dimensional arrays in tensor train format*, in Proceedings of 7th International Workshop on Multidimensional Systems (nDS), IEEE, 2011.

[214] J. Schneider, *Error estimates for two–dimensional cross approximation*, J. Approx. Theory, 162 (2010), pp. 1685–1700.

[215] R. Schneider and A. Uschmajew, *Approximation rates for the hierarchical tensor format in periodic sobolev spaces*, Journal of Complexity, (2013).

[216] U. Schollwöck, *The density–matrix renormalization group*, Rev. Mod. Phys., 77 (2005), pp. 259–315.

[217] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Annals of Physics, 326 (2011), pp. 96–192.

[218] D. Schötzau, *hp-DGFEM for parabolic evolution problems. Applications to diffusion and viscous incompressible fluid flow*, PhD thesis, ETH, Zürich, 1999.

[219] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477.

[220] I. Sloan and H. Wozniakowski, *When are quasi-Monte Carlo algorithms efficient for high dimensional integrals*, J. of Complexity, 14 (1998), pp. 1–33.

[221] S. A. Smolyak, *Quadrature and interpolation formulas for tensor products of certain class of functions*, Dokl. Akad. Nauk SSSR, 148 (1963), pp. 1042–1053. Transl.: Soviet Math. Dokl. 4:240-243, 1963.

[222] S. N. Sreenath, C. Kwang-Hyun, and P. Wellstead, *Modelling the dynamics of signalling pathways*, Essays Biochemistry, 45 (2008), pp. 1–28.

[223] R. Steuer, *Effects of stochasticity in models of the cell cycle: from quantized cycle times to noise-induced oscillations*, Journal of theoretical biology, 228 (2004), pp. 293–301.

[224] M. Stoll and T. Breiten, *A low-rank in time approach to pde-constrained optimization*, MPI Preprint 08, 2013.

[225] E. Tadmor, *The exponential accuracy of Fourier and Chebychev differencing methods*, SIAM J. Numer. Anal., 23 (1986), pp. 1–23.

[226] V. Temlyakov, *Greedy Approximation*, Cambridge University Press, 2011.

[227] L. N. Trefethen, *Spectral methods in MATLAB*, SIAM, Philadelphia, 2000.

[228] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.

[229] E. E. Tyrtyshnikov, *Incomplete cross approximation in the mosaic–skeleton method*, Computing, 64 (2000), pp. 367–380.

[230] E. E. Tyrtyshnikov, *Tensor approximations of matrices generated by asymptotically smooth functions*, Sbornik: Mathematics, 194 (2003), pp. 941–954.

[231] E. E. Tyrtyshnikov, *Kronecker-product approximations for some function-related matrices*, Linear Algebra Appl., 379 (2004), pp. 423–437.

[232] N. G. van Kampen, *Stochastic processes in physics and chemistry*, North Holland, Amsterdam, 1981.

[233] G. Venkiteswaran and M. Junk, *A QMC approach for high dimensional Fokker-Planck equations modelling polymeric liquids*, Math. Comput. Simul., 68 (2005), pp. 43–56.

[234] G. Vidal, *Efficient classical simulation of slightly entangled quantum computations*, Phys. Rev. Lett., 91 (2003), p. 147902.

[235] T. von Petersdorff and C. Schwab, *Numerical solution of parabolic equations in high dimensions*, ESAIM: Mathematical Modelling and Numerical Analysis, 38 (2004), pp. 93–127.

[236] A. Weichselbaum, F. Verstraete, U. Schollwöck, J. I. Cirac, and J. von Delft, *Variational matrix-product-state approach to quantum impurity models*, Phys. Rev. B, 80 (2009), p. 165117.

[237] S. R. White, *Density matrix formulation for quantum renormalization groups*, Phys. Rev. Lett., 69 (1992), pp. 2863–2866.

[238] S. R. White, *Density-matrix algorithms for quantum renormalization groups*, Phys. Rev. B, 48 (1993), pp. 10345–10356.

[239] S. R. White, *Spin gaps in a frustrated Heisenberg model for CaV4O9*, Phys. Rev. Lett., 77 (1996), pp. 3633–3636.

[240] S. R. White, *Density matrix renormalization group algorithms with a single center site*, Phys. Rev. B, 72 (2005), p. 180403.

[241] K. G. Wilson, *The renormalization group: Critical phenomena and the Kondo problem*, Rev. Mod. Phys., 47 (1975), pp. 773–840.

List of notations

	Dimensions, sizes and indices
d	dimension of a tensor, number of coordinates in an equation.
$n_k \leqslant n$	mode size, range of the k-th index in a tensor, $k = 1, \ldots, d$.
i_k, j_k	k-th index in an initial d-dimensional tensor.
$\boldsymbol{i}$, $\boldsymbol{j}$	Multi-index of the initial tensor, $\boldsymbol{i} = (i_1, \ldots, i_d)$.
$\overline{i_1, \ldots, i_d}$	Equivalent multi-index, emphasizing the vectorization, $\overline{i_1, \ldots, i_d} = i_1 + (i_2 - 1)n_1 + \cdots + (i_d - 1)n_1 \cdots n_{d-1}$.
$A \otimes B$	Kronecker product: $C = A \otimes B = \left[C_{\overline{ik},\overline{jm}}\right] = [AB_{k,m}] = [A_{i,j}B_{k,m}]$.
α_k, $\boldsymbol{\alpha}$	rank index between k-th and $(k+1)$-th TT blocks. $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{d-1})$.
$r_k \leqslant r$	TT rank, range of α_k. A tensor of origin x is pointed as $r_k(x)$.
γ_k, $\boldsymbol{\gamma}$	Tucker rank index. Alternatively: rank index in the matrix TT.
$R_k \leqslant R$	Tucker rank, range of γ_k.
	Default tensors and their roles
x	general "vector" tensor, discrete multi-variate function, solution.
A	general "matrix" tensor, discrete operator, matrix in a lin. system.
b	right-hand side in a linear system, "vector" tensor.
z, $\tilde{z}$	(approximate) residual in a linear system, $\tilde{z} \approx z = b - Ax$.
ψ	a "vector" tensor as solution to the Fokker-Planck/master equations.
	Blocks (cores) of tensor formats
$x^{(k)}$, $A^{(k)}$	TT blocks of a vector x, resp. matrix A. Generally are tensors $\left[x^{(k)}_{\alpha_{k-1},\alpha_k}(i_k)\right] \in \mathbb{C}^{r_{k-1} \times n_k \times r_k}$, $\left[A^{(k)}_{\alpha_{k-1},\alpha_k}(i_k, j_k)\right] \in \mathbb{C}^{r_{k-1} \times n_k \times m_k \times r_k}$. In products of form $Mx^{(k)}$, reads also as a vector, $x^{(k)} \in \mathbb{C}^{r_{k-1} n_k r_k}$.
$x^{(c)}$, $x^{c(k)}$	Tucker core, $x^{(c)} \in \mathbb{C}^{R_1 \times \cdots \times R_d}$, and its TT block, $x^{c(k)} \in \mathbb{C}^{r_{k-1} \times R_k \times r_k}$.
$x^{f(k,l)}$	Tucker or QTT-Tucker factors, $x^{f(k)} \in \mathbb{C}^{n_k \times R_k}$, $x^{f(k,l)} \in \mathbb{C}^{r_{k,l} \times n_{k,l} \times r_{k,l-1}}$.
	Maps and reshapes of format blocks
$\tau(\{x^{(k)}\})$	Tensor train map, $\tau(x^{(p)}, \ldots, x^{(q)}) \in \mathbb{C}^{r_{p-1} \times n_p \cdots n_q \times r_q}$. (Def. 2.1.11)
$x^{(\leqslant k)}$, $x^{(\geqslant k)}$	Interface TT chunks, $x^{(\leqslant k)} = \tau(x^{(1)}, \ldots, x^{(k)})$, $x^{(\geqslant k)} = \tau(x^{(k)}, \ldots, x^{(d)})$.
$X_{<k}$, $X_{\neq k}$	Frame matrices, $X_{<k} = x^{(<k)} \otimes I_{n_k \cdots n_d}$, $X_{\neq k} = x^{(<k)} \otimes I_{n_k} \otimes \left(x^{(>k)}\right)^\top$.
$x^{\|k\rangle}$	Left-folded reshape of a 3-index tensor, $x^{\|k\rangle} \in \mathbb{C}^{r_{k-1} n_k \times r_k}$. (Def. 2.1.12)
$x^{\langle k\|}$	Right-folded reshape of a 3-index tensor, $x^{\langle k\|} \in \mathbb{C}^{r_{k-1} \times n_k r_k}$. (Def. 2.1.12)
$x^{\|k\|}$	Center-folded reshape of a 3-index tensor, $x^{\|k\|} \in \mathbb{C}^{n_k \times r_{k-1} r_k}$. (Def. 2.1.12)
$A^{\langle k\rangle}$	Outer-folded reshape of a 4-index tensor, $A^{\langle k\rangle} \in \mathbb{C}^{r_{k-1} n_k \times m_k r_k}$. (Def. 4.4.1)
$A^{\rangle k\langle}$	Inner-folded reshape of a 4-index tensor, $A^{\rangle k\langle} \in \mathbb{C}^{n_k m_k \times r_k r_{k-1}}$. (Def. 4.4.1)
$\mathsf{A}^{<k}$, $\mathsf{A}^{>k}$	Matrix projections onto interfaces, e.g. $\mathsf{A}^{<k}_{\gamma_{k-1}} = \left(x^{(<k)}\right)^* A^{(<k)}_{\gamma_{k-1}} x^{(<k)}$.
$\mathsf{b}^{<k}$, $\mathsf{b}^{>k}$	Vector projections onto interfaces, e.g. $\mathsf{b}^{<k} = \left(x^{(<k)}\right)^* b^{(<k)}$. (Sec. 4.4)
	Quantities related to PDEs and time schemes
q_k, $\boldsymbol{q}$	coordinate variables, $\boldsymbol{q} = (q_1, \ldots, q_d)$.
t	time variable.
δt	effective time step in a numerical scheme.
N_t	number of time steps.
T	"large" time step (interval) in the simultaneous space-time scheme.
$\hat{T}$	time interval of the whole dynamics, i.e. $t \in [0, \hat{T}]$.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialen oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den 03.04.2014

. . . . . .
(Sergey Dolgov)

Daten zum Autor

Name:	Sergey Dolgov
Geburtsdatum, -ort:	19.03.1988, Sankt Petersburg
09.2005 – 06.2009	B. Sc. Applied Mathematics and Physics, Moscow Institute of Physics and Technology
09.2009 – 06.2011	M. Sc. Applied Mathematics and Physics, Moscow Institute of Physics and Technology
06.2011 – 08.2014	Doktorand am Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig