Direct Numerical Solution of Algebraic Lyapunov Equations For Large-Scale Systems Using Quantized Tensor Trains

Michael Nip
Center for Control, Dynamical Systems, and Computation
University of California, Santa Barbara
California, USA 93106
mdnip@engineering.ucsb.edu

João P. Hespanha
Center for Control, Dynamical Systems, and Computation
University of California, Santa Barbara
California, USA 93106
hespanha@ece.ucsb.edu

Mustafa Khammash
Department of Biosystems Science and Engineering
ETH-Zürich
4058 Basel, Switzerland
mustafa.khammash@bsse.ethz.ch

***Abstract*— We present a novel method for solving high-dimensional algebraic Lyapunov equations exploiting the recently proposed Quantized Tensor Train (QTT) numerical linear algebra. A key feature of the approach is that given a prescribed error tolerance, it automatically calculates the optimal lowest rank approximation of the solution in the Frobenius norm. The low rank nature of the approximation potentially enables a *sublinear* scaling of the computational complexity with the number of states of the dynamical system. The resulting solutions appear in a new matrix tensor format which we call the *Vectorized-QTT-Matrix format*. We show the effectiveness of our method by calculating the controllability Gramians for discretized reaction-diffusion equations. We introduce an algorithm for the new tensor format of the solution for calculating the matrix-by-vector product and combine it with the celebrated Lanczos algorithm to compute the dominant eigenvalues/eigenvectors of the matrix.**

I. INTRODUCTION

Lyapunov equations play a fundamental role in characterizing the stability and input-output behavior of LTI systems. They come in two dual forms:

$$AX + XA^* = -Q \tag{1}$$

and

$$A^*X + XA = -Q.$$

In each case, $A, Q \in \mathbb{R}^{n\times n}$ are given square matrices and $X \in \mathbb{R}^{n\times n}$ is treated as an unknown. For simplicity of the exposition, we restrict our attention to (1), but the simple substitution of A^* for A yields the dual equation. Let $\mathcal{L} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ be given by

$$\mathcal{L}(X) = AX + XA^*$$

We refer to $\mathcal{L}$ as the *Lyapunov Operator*. It is a linear operator on the space of $n \times n$ matrices.

Many methods have been proposed for direct numerical solution of Lyapunov equations. Most classical methods have at best linear scaling of the computational complexity with the number of state variables [1], [2], [3].

A. Tensor Structure of Lyapunov Equations

The *Kronecker product* of an $m \times n$ matrix A and a $p \times q$ matrix B denoted $A \otimes B$ is the $mp \times nq$ block matrix given by

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}.$$

By stacking the columns of X and Q, we may rewrite (1) in terms of the Kronecker product

$$(A \otimes I + I \otimes A)X = -Q, \tag{2}$$

where I is the $n \times n$ identity matrix and we have that a matrix representation of $\mathcal{L}$ in this format is given by

$$\mathcal{L} = A \otimes I + I \otimes A. \tag{3}$$

The computational difficulty of solving the linear system in this format for large scale systems is immediately clear. For a system with n states, A, Q, and X have n^2 terms, while the matrix representation of the linear operator $\mathcal{L}$ requires n^4 terms. Even in cases where A and Q are sparse or have special structure, it is exceedingly rare that a similar assertion can be made about the solution X.

Recent work has sought to partially circumvent this problem by looking for solutions in low-rank or other low parametric formats [4], [5]. When A is stable and Q has low rank, it has been shown that the singular values of X decay exponentially [6], hence fast and accurate approximation methods that seek only to compute the first few dominant singular vectors are possible. The method we propose here exploits this idea but also expresses all the matrices and vectors involved in a compressed format achieving additional computational savings.

In the remainder of the paper, we use the recently proposed Quantized Tensor Train (QTT) format and numerical linear algebra for the low parametric representation of the vectors and matrices involved. We then exploit a QTT structured linear system solver based

on Density Matrix Renormalization Group (DMRG) methods pioneered in quantum chemistry to solve (1). While the convergence properties of the DMRG based solver are an open area of research, we find it to be highly efficient in our numerical experiments. The resulting solutions appear in a new matrix format which we call the Vectorized-QTT-Matrix format. We demonstrate the effectiveness of our method by calculating the controllability Gramians for a discretized reaction-diffusion equation. We then introduce algorithms for this new low-parametric tensor format of the solution for calculating the matrix-by-vector product and the dominant eigenvalues/eigenvectors of the matrix. In the context of controllability and observability Gramians, these correspond to the most controllable/observable modes.

II. Tensor Train representation of vectors and matrices

We propose solving Lyapunov equations and extracting useful information from their solutions by using the structured, low-parametric representation of all vectors and matrices involved in the *Tensor Train* (TT) format developed by Oseledets and Tyrtyshnikov [7], [8] (though, we remark that the TT format has been known in theoretical chemistry as Matrix Product States for at least two decades now [9]). Consider a d-dimensional $n_1 \times \ldots \times n_d$-vector $\boldsymbol{X}(i_1, \ldots, i_d)$ and assume that for the k-th dimension there is a collection of $r_{k-1} \times r_k$ matrices $X_k(i_k)$ indexed by $1 \leq i_k \leq n_k$ such that

$$\boldsymbol{X}(i_1, \ldots, i_d) = X_1(i_1) X_2(i_2) \ldots X_d(i_d). \quad (4)$$

The product of these index dependent matrices is of size $r_0 \times r_d$ so we require $r_0 = r_d = 1$. We say that $\boldsymbol{X}$ is represented in the *Tensor Train (TT) decomposition* with *TT-cores* $X_1(\cdot), \ldots, X_d(\cdot)$, where a TT-core is the family of matrices $X_k(i_k)$ parameterized by i_k. The matrix sizes $r_1, \ldots, r_{d-1}$ are referred to as the *TT-ranks* of the decomposition. For any d-dimensional vector in full format, the TT format admits a robust algorithm for the construction of a decomposition, exact or *approximate*, through the low-rank representation of a sequence of single matrices; for example, by SVD.

In particular, note that for every $k = 1, \ldots, d-1$ the decomposition (4) implies a rank-r_k representation of an *unfolding matrix* $\boldsymbol{X}^{(k)}$ which consists of the entries

$$\boldsymbol{X}^{(k)}(i_1 \ldots i_k; i_{k+1} \ldots i_d) = \boldsymbol{X}(i_1, \ldots, i_d),$$

where $i_1 \ldots i_k$ and $i_{k+1} \ldots i_d$ are treated as multi-indices. Note that the TT-ranks may be computed directly from $\boldsymbol{X}$ in full format; if the vector $\boldsymbol{X}$ is such that the unfolding matrices $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(d-1)}$ are of ranks $r_1, \ldots, r_{d-1}$ respectively, then the cores $X_1(\cdot), X_2(\cdot), \ldots, X_d(\cdot)$, satisfying (4), do exist; see Theorem 2.1 in [8]. The ranks of the unfolding matrices are the lowest possible ranks of a TT decomposition of the vector.

If a vector is given in a TT-format with sub-optimal ranks, a fast and robust rounding procedure is available based on QR and SVD. Here, rounding is understood to mean finding an approximate vector with smaller ranks close enough to satisfy a prescribed accuracy tolerance ϵ in the Frobenius norm [8].

It was discussed in [10] that the ordering of the indices plays a crucial role in determining the numerical values of the TT-ranks. Swapping indices may affect the TT-ranks significantly. This issue was discussed in the context of compressing probability density functions with a high degree of correlation in the data.

The TT representation has also been applied to multidimensional matrices. Consider a d-dimensional $(m_1 \times \ldots \times m_d) \times (n_1 \times \ldots \times n_d)$-matrix $\boldsymbol{A}(i_1, \ldots, i_d; j_1, \ldots, j_d)$ and assume that for the k-th dimension there is a collection of $r_{k-1} \times r_k$ matrices $A_k(i_k, j_k)$ indexed by (i_k, j_k) such that

$$\begin{aligned} \boldsymbol{A}(i_1, \ldots, i_d; j_1, \ldots, j_d) &= A_1(i_1, j_1) A_2(i_2, j_2) \ldots \\ &\quad \ldots A_d(i_d, j_d). \end{aligned} \quad (5)$$

We say that $\boldsymbol{A}$ is represented in the *Tensor Train-Matrix (TTM) Format*. The same definitions and properties of the TT decomposition of vectors applies to the TTM format for matrices.

Basic tensor arithmetics with vectors and matrices in the TT format, such as addition, Hadamard and dot products, multi-dimensional contraction, matrix-vector multiplication, etc. are described in detail in [8].

Note that the storage cost and complexity of many basic operations in the TT format are *linear* in the number of "physical" dimensions d and polynomial in the TT-ranks. TT methods are therefore seen as a means of lifting the so-called Curse of Dimensionality [11] in many applications [7]. We emphasize that the polynomial dependence on the TT-ranks means that it is crucial to characterize the growth of the TT-ranks whenever possible.

Since a TT decomposition of a d-dimensional tensor has $d-1$ ranks that may take different values, it is convenient to introduce an aggregate characteristic such as the *effective rank* of the TT decomposition. For an $n_1 \times \ldots \times n_d$-tensor given in a TT decomposition of ranks $r_1, \ldots, r_{d-1}$, we definite it as the positive root $r_{\text{eff}} = r$ of the quadratic equation

$$n_1 r_1 + \sum_{k=2}^{d-1} r_{k-1} n_k r_k + r_{d-1} n_d = n_1 r + \sum_{k=2}^{d-1} r\, n_k r + r\, n_d \quad (6)$$

which, for an integer r, equates the memory needed to store the given decomposition (left-hand side) and a decomposition in the same format, i.e. of an $n_1 \times \ldots \times n_d$-tensor, but with equal $d-1$ ranks $r, \ldots, r$ (right-hand side). Here, "effective" is understood with respect to storage.

A. Quantized Tensor Train Representation

In order to further reduce the complexity, the TT format can be applied to a "quantized" vector (matrix), which leads to the *Quantized Tensor Train* (QTT) format [12], [13], [14]. The idea of quantization consists of "folding" the vector (matrix) by introducing l_k "virtual" indices corresponding to the k-th original "physical" index [15], provided that the corresponding mode size n_k can be factorized as $n_k = n_{k,1} \cdot n_{k,2} \cdot \ldots \cdot n_{k,l_k}$ in terms of integral factors $n_{k,1}, \ldots, n_{k,l_k} \geq 2$. Typically, the finest possible quantization is used, i.e., $n_{k,\hat{k}} = 2$ for $\hat{k} = 1, \ldots, l_k$. For example, if $n_k = 2^{10}$, then the finest possible quantization would fold the k-th dimension into $l_k = 10$ dimensions with each corresponding index taking $n_{k,\hat{k}} = 2$ values. The folding preserves the number of entries in the vector (matrix) but makes more structure in the data accessible to the TT compression.

Once the quantization folding has been applied to all the "physical" dimensions, a TT decomposition of the quantized vector is referred to as a *QTT decomposition* of the original vector. The ranks of this TT decomposition are called *QTT-ranks* of the original vector.

If the natural ordering

$$\underbrace{i_{1,1}, \ldots, i_{1,l_1}}_{\text{1st dimension}}, \underbrace{i_{2,1}, \ldots, i_{2,l_2}}_{\text{2nd dimension}}, \ldots, \underbrace{i_{d,1}, \ldots, i_{d,l_d}}_{d\text{th dimension}} \tag{7}$$

of the "virtual" indices is used, then the QTT-ranks are ordered as follows:

$$\underbrace{r_{1,1}, \ldots, r_{1,l_1-1}}_{\text{1st dimension}}, \hat{r}_1, \underbrace{r_{2,1}, \ldots, r_{2,l_2-1}}_{\text{2nd dimension}}, \hat{r}_2, \ldots \\ \ldots, \hat{r}_{d-1}, \underbrace{r_{d,1}, \ldots, r_{d,l_d-1}}_{d\text{th dimension}},$$

where $\hat{r}_1, \ldots, \hat{r}_{d-1}$ are the TT ranks of the *original* tensor. That is, the folding preserves the TT-ranks.

As a QTT decomposition is a TT decomposition of an appropriately folded tensor, the TT arithmetics discussed previously extend naturally to the QTT format.

Quantization has been shown to be crucial in many practical applications for reducing the complexity. While a simple characterization of when a vector (matrix) will have low-rank QTT structure is unavailable QTT-rank bounds are available for many simple functions evaluated on tensor grids including trigonometric and polynomial functions [16], as well as univariate asymptotically smooth functions [17]. When a uniform QTT-rank bound is available, this implies a *logarithmic* scaling of the storage and complexity of basic arithmetics with the mode size n_k [12].

III. The structure of the Lyapunov Operator in the QTT format

When using Tensor Train formats it is important to establish bounds on the ranks whenever possible. The following result characterizes the ranks of the Lyapunov operator in terms of the ranks of $\boldsymbol{A}$.

Theorem III.1. *Suppose A has a QTT matrix decomposition with QTT ranks*

$$r_{1,1}, \ldots, r_{1,l_1-1}, \hat{r}_1, r_{2,1}, \ldots, r_{2,l_2-1}, \hat{r}_2, \ldots \\ \ldots, \hat{r}_{d-1}, r_{d,1}, \ldots, r_{d,l_d-1},$$

then the Lyapunov Operator has a QTT matrix decomposition with ranks bound from above by

$$r_{1,1}+1, \ldots, r_{1,l_1-1}+1, \hat{r}_1+1, r_{2,1}+1, \ldots \\ \ldots, r_{2,l_2-1}+1, \hat{r}_2+1, \ldots, \hat{r}_{d-1}+1, r_{d,1}+1, \ldots \\ \ldots, r_{d,l_d-1}+1, 2, r_{1,1}+1, \ldots \\ \ldots, r_{1,l_1-1}+1, \hat{r}_1+1, r_{2,1}+1, \ldots \\ \ldots, r_{2,l_2-1}+1, \hat{r}_2+1, \ldots, \hat{r}_{d-1}+1, r_{d,1}+1, \ldots \\ \ldots, r_{d,l_d-1}+1,$$

Proof: The identity matrix has a rank-1 QTT matrix decomposition. From (3) we see that $\mathcal{L}$ is the sum of two parts: $A \otimes I$ which has QTT matrix ranks

$$r_{1,1}, \ldots, r_{1,l_1-1}, \hat{r}_1, r_{2,1}, \ldots, r_{2,l_2-1}, \hat{r}_2, \ldots \\ \ldots, \hat{r}_{d-1}, r_{d,1}, \ldots, r_{d,l_d-1}, 1, \ldots, 1$$

and $I \otimes A$ which has QTT matrix ranks

$$1, \ldots, 1, r_{1,1}, \ldots, r_{1,l_1-1}, \hat{r}_1, r_{2,1}, \ldots \\ \ldots, r_{2,l_2-1}, \hat{r}_2, \ldots\ldots, \hat{r}_{d-1}, r_{d,1}, \ldots, r_{d,l_d-1},$$

The ranks of the sum are bound above by the sum of the ranks which completes the proof. □

A. Vectorized-(Q)TT-Matrix Format

Choosing the (Q)TT-Matrix format above for $\mathcal{L}$ is incompatible with using the (Q)TT-Matrix format for $\boldsymbol{X}$, even though $\boldsymbol{X}$ encodes a linear operator. Rather, the quantization and physical dimensions must be split and shuffled into a new format with index ordering

$$i_{1,1}, \ldots, i_{1,l_1}, i_{2,1}, \ldots, i_{d-1,l_{d-1}}, i_{d,1}, \ldots, i_{d,l_d}, \\ j_{1,1}, \ldots, j_{1,l_1}, j_{2,1}, \ldots, j_{d-1,l_{d-1}}, j_{d,1}, \ldots, j_{d,l_d}, \tag{8}$$

That is, the d-level matrix $\boldsymbol{X}$ is treated as a $2d$-level vector and compressed into the corresponding (Q)TT format. We refer to the matrix format with the index ordering and TT compression above as the *Vectorized-(Q)TT-Matrix* format. Note that the middle rank of this decomposition $\hat{r}_c$, where $c = \sum_{k=1}^{d} l_k$, corresponds to the (matrix) rank of the unfolding matrix

$$\boldsymbol{X}^{(c)} = \boldsymbol{X}(i_{1,1}, \ldots, i_{d,l_d}; j_{1,1}, \ldots, j_{d,l_d}).$$

That is, $\boldsymbol{X}$ as a linear operator has rank $\hat{r}_c$. Given a matrix in the Vectorized-(Q)TT-Matrix format, the rank of the corresponding linear operator is immediately known.

B. Solving Lyapunov Equations in the QTT format

For solving the QTT structured linear system corresponding to the Lyapunov Equation we propose using the Density Matrix Renormalization Group (DMRG)-based linear system solver described in [18]. Given a linear system in the (Q)TT-format, it produces an approximate solution vector also in the (Q)TT format.

The DMRG-based solver is optimization based. Given an allowed error in the residual it automatically adapts the TT-ranks and optimizes the corresponding TT-cores until the convergence criterion is reached. The automatic adaptation of the TT-ranks is essential: if the ranks are too small then there is no hope of meeting a tight error tolerance, but if they are too large the computational complexity increases. While there is no estimate of the convergence rate of the DMRG-based solver, in many practical applications, it has proven highly efficient.

IV. Computing Eigenvalues and Eigenvectors in the QTT Format

A. Vectorized-TT-Matrix-Vector Product

The most important operation in linear algebra is probably the matrix-by-vector product. Oseledets and Tyrtyshnikov proposed an algorithm for computing the matrix-by-vector product $\boldsymbol{Ax}$ when $\boldsymbol{A}$ is a d-level matrix in the TT-Matrix format and $\boldsymbol{x}$ is a d-level vector in the TT format [8]. We introduce a similar algorithm for the matrix-by-vector product when A is in the Vectorized-TT-Matrix format. For simplicity, we will use the notation of the TT format rather than QTT but the same results apply with quantized tensors as well.

Suppose that $\boldsymbol{A}$ is in the Vectorized-TT-Matrix format with decomposition

$$\boldsymbol{A}(i_1,\ldots,,i_d,j_1,\ldots,j_d) = A_{11}(i_1)\ldots A_{1d}(i_d)A_{21}(j_1)\ldots A_{2d}(j_d), \quad (9)$$

where $A_{1k}(i_k)$ and $A_{2k}(j_k)$ are $r_{1,k-1}\times r_{1,k}$ and $r_{2,k-1}\times r_{2,k}$ matrices, respectively. Suppose that $\boldsymbol{x}$ is in the TT format (4) with cores $X_k(j_k)$. The matrix-by vector product is the computation of the following sum:

$$\boldsymbol{y}(i_1,\ldots,i_d) = \sum_{j_1,\ldots,j_d} \boldsymbol{A}(i_1,\ldots,,i_d,j_1,\ldots,j_d)\boldsymbol{x}(j_1,\ldots,j_d).$$

The resulting tensor will also be in the TT-format. Indeed,

$$\begin{aligned}\boldsymbol{y}(i_1,\ldots,i_d) &= \sum_{j_1,\ldots,j_d} A_{11}(i_1)\ldots A_{1d}(i_d)A_{21}(j_1)\ldots \\ &\quad A_{2d}(j_d)\times X_1(j_1)\ldots X_d(j_d) \\ &= \sum_{j_1,\ldots,j_d} (A_{11}(i_1)\ldots A_{1d}(i_d))\ldots \\ &\times (A_{21}(j_1)\otimes X_1(j_1))\ldots(A_{2d}(j_d)\otimes X_d(j_d)) \\ &= (A_{11}(i_1)\ldots A_{1d}(i_d))\,Z_1\ldots Z_d,\end{aligned}$$

where

$$Z_k = \sum_{j_k}(A_{2k}(j_k)\otimes X_k(j_k)).$$

Taking

$$Y_k(i_k) = \begin{cases} A_{1k}(i_k), & k\neq d \\ A_{1k}(i_k)Z_1\ldots Z_d, & k=d \end{cases},$$

we have that

$$\boldsymbol{y}(i_1,\ldots,i_d) = Y_1(i_1)\ldots Y_d(i_d),$$

and the product is in the (Q)TT-format. A formal description of the algorithm is presented in Algorithm 1.

Algorithm 1 Vectorized (Q)TT-Matrix-by-Vector Product

Require: Matrix $\boldsymbol{A}$ in the Vectorized-(Q)TT-Matrix-Format with cores $A_{1k}(i_k), A_{2k}(j_k)$, and vector x in the (Q)TT-format with cores $X_k(j_k)$.
Ensure: Vector $\boldsymbol{y} = Ax$ in the TT-format with cores $Y_k(i_k)$.
  for $k=1$ to d **do**
    $Z_k = \sum_{j_k}(A_{2k}(j_k)\otimes X_k(j_k))$.
  end for
  for $k=1$ to d **do**
    if $k\neq d$ **then**
      $Y_k(i_k) = A_{1k}(i_k)$.
    else
      $Y_d(i_d) = A_{1d}(i_d)Z_1\ldots Z_d$.
    end if
  end for

Since Z_d is a column vector, evaluating $Y_d(i_d)$ reduces to the computation of matrices Z_k and evaluating $d+1$ matrix-by-vector products. The total number of arithmetic operations required is $\mathcal{O}(dnr^2)$ where n is an upper bound on all the mode sizes and r is an upper bound on all the ranks involved.

Note that the first $d-1$ cores of the resulting tensor are precisely the first $d-1$ cores of the matrix A. The d-th core is the d-th core of the matrix A post-multiplied by the matrices Y_k. For this formulation of the matrix-by-vector product, the (Q)TT-ranks of the product are exactly the (Q)TT-ranks of the original matrix A. This is in sharp contrast with the matrix-by-vector product for matrices in (Q)TT-Matrix-format where the ranks of the product are bound from above by the products of the corresponding ranks and the operation generally increases the ranks. We exploit this fact in the following section where we discuss a Lanczos-type algorithm for matrices and vectors in these formats.

B. (Q)TT Lanczos Iteration Incorporating Vectorized TT Matrix-by-Vector Product

The Lanczos iteration is a powerful method for quickly estimating dominant eigenvalues and the cor-

responding eigenvectors of Hermitian matrices, for instance, finding the most controllable/observable modes of a linear system from the controllability/observability Gramian. Given a Hermitian matrix H and a predetermined number of iterations m, it performs an efficient Krylov iteration to construct a change of basis transforming H into a tri-diagonal matrix T_{mm} that can be diagonalized efficiently, for example, using the QR algorithm. Once the eigensystem of T_{mm} is known, it is straightforward to reconstruct the dominant eigenvectors of H, see for example [19]. The Lanczos iteration is attractive computationally since the only large-scale linear operation is matrix-by-vector multiplication.

We use a version of the Lanczos iteration described in [20] for low-rank tensor formats incorporating the Vectorized Matrix-by-Vector product introduced in the previous section. A key challenge of such methods is controlling the rank through the orthogonalization step, as adding or subtracting tensors generally increases rank. The more iterations, the larger the ranks. We use the TT-rounding procedure to recompress the tensor after each orthogonalization step to help control this growth. The formal description of the procedure is listed in Algorithm 2. If the Hermitian matrix H were instead given in the (Q)TT-Matrix format, the standard (Q)TT Matrix-by-Vector product could be substituted in the proposed algorithm but the TT-rounding procedure should also be performed after every matrix-by-vector product to help control the TT-ranks.

While it can be proved that with exact arithmetic, the Lanczos iteration constructs a unitary transformation and that the eigenvalues/vectors computed are good approximations to those of the original matrix, it is well understood that when using floating point arithmetic the orthogonality may be quickly lost. An overzealous use of the TT-rounding procedure has the same effect and we observe this in our numerical experiments.

Algorithm 2 (Q)TT-Lanczos Iteration

Require: Matrix $\boldsymbol{A}$ in the Vectorized-(Q)TT-Matrix format and number of iterations m.

Ensure: Sequences of values $\{\alpha_k\}_{k=1}^m$, $\{\beta_k\}_{k=1}^m$ and Lanczos vectors $\{\boldsymbol{v}_k\}_{k=1}^m$ in the (Q)TT-format.

$\boldsymbol{v_1}$ = random (Q)TT-vector with norm 1.
$\boldsymbol{v_0}$ = zero vector.
$\beta_1 = 0$.
for $k = 1$ to m **do**
  $\boldsymbol{w_j} = A\boldsymbol{v_j}$, {Matrix-by-Vector Product}
  $\alpha_j = \boldsymbol{w_j} \cdot \boldsymbol{v_j}$,
  $\boldsymbol{w_j} = \boldsymbol{w_j} - \alpha_j \boldsymbol{v_j} - \beta_j \boldsymbol{v_{j-1}}$, {Orthogonalization}
  TT-Round $\boldsymbol{w_j}$ to accuracy ϵ,
  $\beta_{j+1} = ||\boldsymbol{w_j}||$,
  $\boldsymbol{v_{j+1}} = \boldsymbol{w}_j/\beta_{j+1}$
end for

V. Example Problem: A Discretized Reaction-Diffusion Equation

Consider the following reaction-diffusion equation on the hypercube $D = (-\pi, \pi)^d$ with point control and Dirichlet boundary conditions:

$$\begin{cases} \partial_t \boldsymbol{\psi}(x,t) = c_1 \Delta \boldsymbol{\psi}(x,t) - c_2 \boldsymbol{\psi}(x,t) + \delta(x) u(t), \\ \qquad\qquad x \in D, \\ \boldsymbol{\psi}(x,t) = 0, \quad x \in \partial D \end{cases} \tag{10}$$

where $c_1, c_2 > 0$, $u(t)$ is a control input and $\delta(x)$ is the Dirac delta function centered at zero. This equation models a d-dimensional reaction vessel in which the only reaction is spontaneous degradation. $\boldsymbol{\psi}(x,t)$ describes the concentration of a particular chemical reactant at spatial coordinate x and at time t. The control signal allows the injection or removal of the substance at the point $x = 0$.

In this section, we consider versions of (10) that have been discretized using a finite difference scheme in space on a uniform tensor grid with spacing h but no discretization in time (Method of Lines). We use a second-order central difference to approximate the Laplacian and approximate the Dirac delta as a Kronecker delta function at the nearest grid point with unit mass. Let $\hat{\boldsymbol{\psi}}(x,t)$ denote the discrete approximation of $\boldsymbol{\psi}(x,t)$. The time evolution of the discretized system is given by the finite-dimensional LTI system:

$$\partial_t \hat{\boldsymbol{\psi}}(x,t) = A\hat{\boldsymbol{\psi}}(x,t) + Bu(t), \tag{11}$$

with

$$A = \frac{c_1}{h^2}\Delta_{dd} - c_2 I, \quad B = \frac{1}{h^d}\hat{\delta}(x),$$

where Δ_{dd} is the discrete Laplacian on a rectangular grid with Dirichlet boundary conditions and $\hat{\delta}(x)$ is a vector that is zero everywhere except at the entry corresponding to the grid point closest to the origin.

Kazeev, *et al*. showed that using the finest possible quantization, Δ_{dd} has an explicit QTT representation with all QTT ranks < 4 [21]. Hence, A has QTT-ranks < 5. Also, B is rank 1 separable after quantization so that all the matrices and vectors involved have low QTT-ranks.

The controllability Gramian of the discretized system is expected to have low QTT rank for two reasons. First, the matrix B has rank one so the singular values of the Gramian can be expected to decay rapidly (the system is not very controllable). A low-rank approximation using only the first few singular values and vectors can be expected to approximate the solution well. Secondly, the first few singular vectors of the Gramian of the original system can be expected to have a high degree of regularity so that at fine levels of discretization, they can be well approximated by low order polynomials. Hence, the singular vectors of the approximate Gramian can be expected to have low QTT ranks. While it is possible to use another compression

scheme other than quantization (e.g. Galerkin projection onto tensorized Legendre polynomials), this would require *a priori* a choice of grid, polynomial orders, etc for the compression. By using the QTT numerical linear algebra, the compression is performed *automatically and on the fly*.

In the following, we implemented our proposed algorithms in MATLAB using the *TT Toolbox* implementation of the TT and QTT tensor formats, publicly available at http://spring.inm.ras.ru/osel. All calculations were performed in MATLAB 7.11.0.584 (R2010b) on a laptop with a 2.7 GHz dual-core processor with 12 GB RAM.

Our implementation relies crucially on the DMRG Linear System Solver available as **dmrg_solve2** in the *TT Toolbox*. While a rigorous convergence analysis of the DMRG solver is still missing, we find in our examples that it can be highly efficient.

A. Testing the DMRG Solver

The Controllability Gramian, W_c, of an LTI system solves the Lyapunov equation

$$AW_c + W_c A^* = -BB^*. \tag{12}$$

We use the proposed DMRG-based solver to compute a low-rank approximate Gramian $\hat{W}_c$ and compare it to a solution computed in full format using the MATLAB Control System Toolbox's **lyap** function which we treat as the true solution. In each case, we ran the DMRG-based solver until the following accuracy condition was met:

$$||\hat{W}_c - W_c||_2 < 10^{-9}$$

where $||\cdot||_2$ denotes the induced 2-norm. Practically, this required careful tuning of the DMRG-based linear system solver residual error tolerance to produce solutions of the desired accuracy (and no more). In each case, the DMRG solver was initialized with a random rank-2 QTT vector, though it is possible in practical problems to initialize in a smarter way.

Figure 1 plots the computation time required to compute the Gramian in both the full and the QTT formats for the 1-D version of the problem over a range of discretization levels. Compute time in the full format scales linearly with the number of discretization points while compute time for the QTT version does not. Instead, the QTT based algorithm depends critically on the ranks of the solution. Interestingly, on finer grids, the compute time actually decreases since the solution can be well represented with tensors having much smaller QTT-ranks. While the full format solution is faster for coarse levels of discretization, we emphasize that the *scaling* of the QTT approach is much more favorable.

Figure 2 plots the computation time required to compute the Gramian in both the full and the QTT formats where the discretization level in each dimension is kept uniform but the number of dimensions is

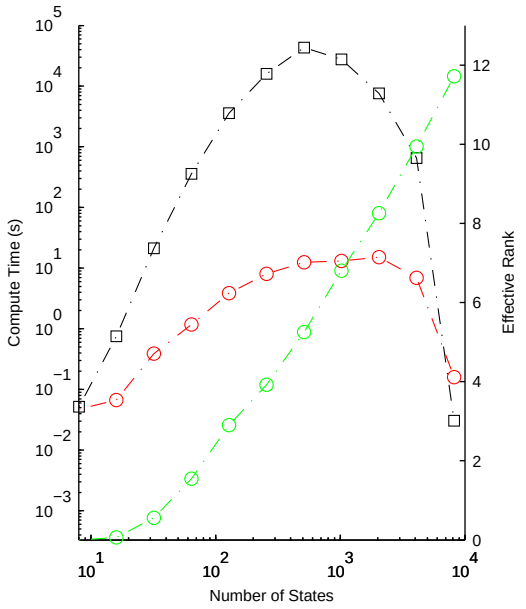


Fig. 1: DMRG Compute Time and Effective Rank vs. Number of States for discretized 1-D reaction-diffusion model. The compute time for the full format solution (green) scales linearly with the number of states, while it remains small for the proposed method (red). Black indicates the effective rank of the solution obtained by the proposed method. At finer discretizations, approximations with lower effective QTT-rank approximate the full solution to the same accuracy tolerance.

scaled. In each case, 2^4 discretization points are used in each dimension. The compute time for the full format grows rapidly with the number of dimensions while the compute time for the QTT version remains bounded. Again, the QTT based algorithm depends critically on the QTT-ranks of the solution. Three dimensions was the maximum number that our compute hardware could handle for the full format due to the exponential increase in storage requirements but dimensions as high as ten were successfully computed in the QTT format using a residual tolerance $< 10^{-9}$ in less than 5 minutes for each problem.

B. A Large Scale Problem

We next tested our proposed methods on a large-scale version of the problem. In 2-D we took $2^{10} = 1024$ grid points in each direction resulting in a discretized version of the problem with $2^{20} > 1.04$ million states. In full format, the corresponding A matrix has 2^{40} entries, though it is highly structured with only 5.24 million nonzero elements. However, even when $\boldsymbol{A}$ has a lot of sparse structure, it is rare that the Controllability Gramian inherits this structure. In this example the Gramian would require storage of 2^{40} entries or over 3 TB of storage using 32-bit floating precision. However, the rank adaptiveness of the DMRG solver combined

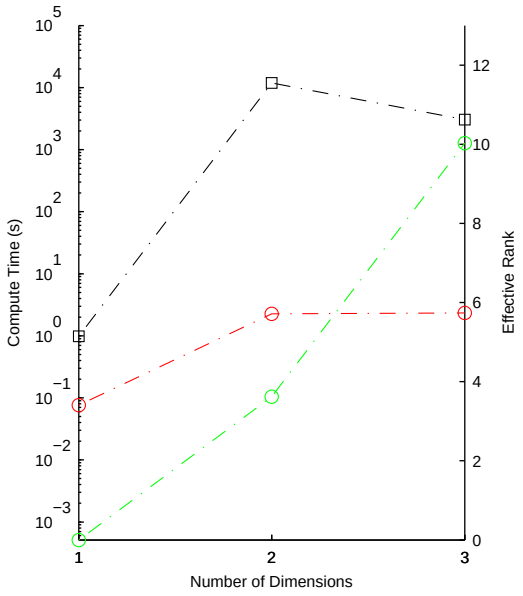


Fig. 2: DMRG Compute Time and Effective Rank vs. Number of Dimensions for discretized reaction-diffusion models where the number of physical dimensions scales. The compute time for the full format solution (green) increases rapidly with the number of dimensions, while it increases less quickly for the proposed method (red). Effective QTT rank of the approximate solution appears in black.

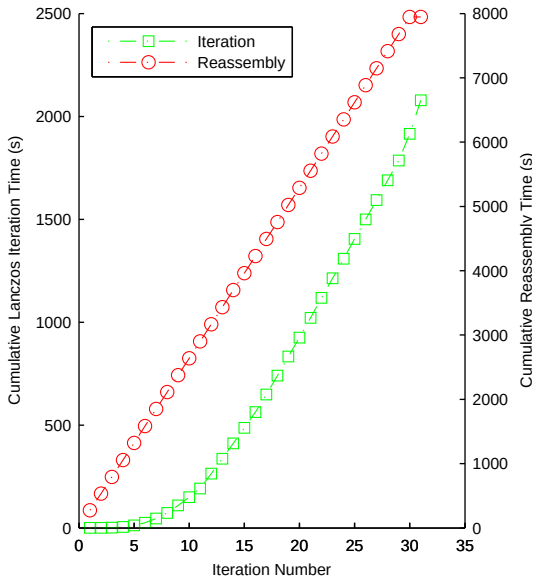


Fig. 3: Compute time for the proposed QTT Lanczos Algorithm for the large-scale problem. 30 iterations were performed.

with the high degree of QTT structure in the solution means that it is no problem for our proposed approach.

Using a residual tolerance smaller than 10^{-9} for the DMRG solver and allowing it to compute as many iterations as needed for convergence, the solver took 67.5 sec to converge to a solution in Vectorized-QTT-Matrix format with effective rank=29.09 and only 67,680 parameters needed to specify it (a compression ratio of seven orders of magnitude). The (matrix) rank of the approximate Gramian was 16.

We then used the QTT Lanczos algorithm to compute approximate dominant eigenvalues. We performed 30 iterations and accepted eigenvectors with a high degree of symmetry under exchange of the two physical dimensions due to the symmetry in the problem, see Figure 5. Figure 4 plots the eigenvalues produced by the algorithm. Figure 3 plots the cumulative compute time for the QTT Lanzcos Iteration as well as the time needed to reassemble the eigenvectors of $\hat{W}_c$ from those of T_{mm}. The total run time of the QTT Lanczos Algorithm was 2.79 hours. Orthogonality of the iterates was lost very quickly so that the algorithm produced multiple copies of these eigenvectors. The algorithm also produced several badly corrupted vectors that lacked the symmetry; especially in the last few iterations, see Figure 6. These effects are typical of the classical Lanczos Algorithm. The former problem could be mitigated using modern heuristics such as random restarts, better orthogonalization procedures, etc. and the latter issue can be resolved by performing more iterations.

VI. CONCLUSIONS AND FUTURE WORKS

We introduced a new approach to solving Lyapunov equations based on the Quantized Tensor Train numerical linear algebra. A key feature of the approach is that it adapts the matrix rank of the solution to capture the essential features of interest but no more. This parsimonious representation of the vectors and matrices involved allows substantial savings both in memory resources and overall compute time. We introduced a new QTT-Matrix format and some associated algorithms for extracting information from matrices in the format. Lastly, we demonstrated the effectiveness of the approach on a challenging numerical example.

We remark that low-parametric tensor formats and linear system solvers in these formats is an active area of research. As solver technology improves we expect a further increase in the efficiency of our approach.

As future work, we plan to compare the results reported here for the large scale problem against other approximate methods, for instance empirical Gramians from snapshot-based methods or the other low-rank methods mentioned in the introduction. Identifying practical control systems for which the proposed methods can be expected to work well is very much ongoing work.

VII. ACKNOWLEDGMENTS

This material is based upon work supported by the Institute for Collaborative Biotechnologies through

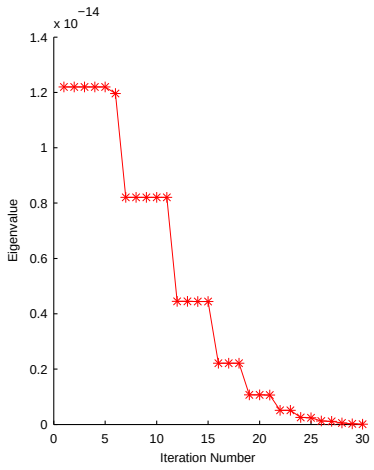
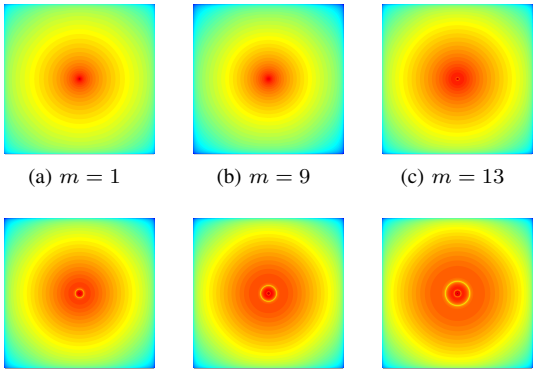


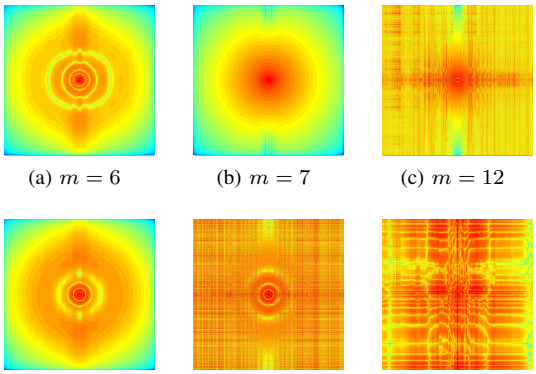
Fig. 4: Eigenvalues produced by the proposed QTT Lanczos Algorithm for the large-scale problem. 30 iterations were performed. Many eigenvalues repeat, especially in the first twenty iterations. This reflects the loss of orthogonality through the iteration process.



(a) $m = 1$ (b) $m = 9$ (c) $m = 13$

(d) $m = 16$ (e) $m = 19$ (f) $m = 22$

Fig. 5: Approximate eigenvectors of the Controllability Gramian produced by QTT Lanczos Iteration for the 2-D discretized reaction-diffusion equation with point control.



(a) $m = 6$ (b) $m = 7$ (c) $m = 12$

(d) $m = 26$ (e) $m = 28$ (f) $m = 29$

Fig. 6: Spurious eigenvectors produced by QTT Lanczos Iteration for the 2-D discretized reaction-diffusion equation with point control.

grant W911NF-09-D-0001 from the U.S. Army Research Office and by the National Science Foundation under Grants No. ECCS-0725485 and ECCS-0835847.

REFERENCES

[1] R. H. Bartels and G. W. Stewart. Solution of the matrix equation AX+XB=C. *Commun. ACM*, 15(9):820–826, September 1972.

[2] G. Golub, S. Nash, and C. Van Loan. A Hessenberg-Schur method for the problem AX+XB=C. *Automatic Control, IEEE Transactions on*, 24(6):909–913, Dec.

[3] S. J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA Journal of Numerical Analysis*, 2(3):303–323, 1982.

[4] T. Penzl. A cyclic low-rank Smith method for large sparse Lyapunov equations. *SIAM Journal on Scientific Computing*, 21(4):1401–1418, 1999.

[5] L. Grasedyck and W. Hackbusch. A multigrid method to solve large scale Sylvester equations. *SIAM Journal on Matrix Analysis and Applications*, 29(3):870–894, 2007.

[6] L. Grasedyck. Existence of a low rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation. *Numerical Linear Algebra with Applications*, 11(4):371–389, 2004.

[7] I. V. Oseledets and E. E. Tyrtyshnikov. Breaking the curse of dimensionality, or how to use SVD in many dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759, October 2009.

[8] I. V. Oseledets. Tensor Train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[9] Steven R. White. Density-matrix algorithms for quantum renormalization groups. *Phys. Rev. B*, 48(14):10345–10356, October 1993.

[10] V. Kazeev, M. Khammash, M. Nip, and C. Schwab. Direct solution of the chemical master equation using quantized tensor trains. Technical Report 2013-04, Seminar for Applied Mathematics, ETH Zürich, Switzerland, 2013.

[11] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, NJ, 1961.

[12] I. Oseledets. Approximation of matrices with logarithmic number of parameters. *Doklady Mathematics*, 80:653–654, 2009.

[13] B. N. Khoromskij. $\mathcal{O}(d \log n)$-quantics approximation of n-d tensors in high-dimensional numerical modeling. *Constructive Approximation*, 34(2):257–280, 2011. 10.1007/s00365-011-9131-1.

[14] I. V. Oseledets. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145, 2010.

[15] E. E. Tyrtyshnikov. Tensor approximations of matrices generated by asymptotically smooth functions. *Sbornik: Mathematics*, 194(5):941–954, 2003.

[16] I. V. Oseledets. Constructive representation of functions in tensor formats. *Constructive Approximation*, (37):1–18, 2013.

[17] L Grasedyck. Polynomial approximation in Hierarchical Tucker format by vector-tensorization. 2010. *Hackbusch, Tensorisation of Vectors and their Efficient Convolution, W. Hackbusch, L. Grasedyck, R. Schneider, Multilevel tensorisation (in preparation)*.

[18] Sergey V. Dolgov and Ivan V. Oseledets. Solution of linear systems and matrix inversion in the TT-format (submitted to SISC). Preprint 19, Max-Planck-Institut für Mathematik in den Naturwissenschaften, 2011.

[19] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[20] Wolfgang Hackbusch, Boris N. Khoromskij, Stefan Sauter, and Eugene E. Tyrtyshnikov. Use of tensor formats in elliptic eigenvalue problems. *Numerical Linear Algebra with Applications*, 19(1):133–151, 2012.

[21] Vladimir A. Kazeev and Boris N. Khoromskij. Low-rank explicit QTT representation of the Laplace operator and its inverse. *SIAM Journal on Matrix Analysis and Applications*, 33(3):742–758, 2012.