

A Modified Levenberg-Marquardt Algorithm For Tensor CP Decomposition in Image Compression

Ramin Goudarzi Karim*, Dipak Dulal†, and Carmeliza Navasca‡

*Stillman College

3601 Stillman Blvd

Tuscaloosa, Alabama, 35401, US

rkarim@stillman.edu

† ‡University of Alabama at Birmingham

1720 University Blvd

Birmingham, Alabama, 35294, US

dpdulal@uab.edu, cnavasca@uab.edu

Abstract

This paper proposed a new variant of the Levenberg-Marquardt algorithm used for Tensor Canonical Polyadic (CP) decomposition with an emphasis on image compression and reconstruction. Tensor computation, especially CP decomposition, holds significant applications in data compression and analysis. In this study, we formulate CP as a nonlinear least squares optimization problem. Then, we present an iterative Levenberg-Marquandt (LM)-based algorithm for computing the CP decomposition. Ultimately, we test the algorithm on various datasets, including randomly generated tensors and RGB images. The proposed method proves to be both efficient and effective, offering a reduced computational burden when compared to the traditional Levenberg-Marquardt technique.

Introduction

Tensor computation is essential in the domain of computational sciences. Tensors, which can be thought of as higher-order versions of matrices, offer important tools in scientific computing, computer vision, and various engineering disciplines [1–3]. Furthermore, tensor decompositions constitute essential tools for data science and machine learning applications [4, 5].

Canonical Polyadic (CP) (a.k.a. CANDECOMP/PARAFAC [6]) is a method of tensor factorization. In the simplest terms, tensor factorization involves breaking down a tensor into a product of lower-dimensional tensors. For instance, a third-order tensor can be factorized into a sum of outer products of three-dimensional vectors. The CP decomposition can be useful in many applications, including signal processing [5, 7], data compression [8], and machine learning [9], because it allows for a compressed, yet informative, representation of the data. However, finding the best CP decomposition (i.e., minimizing the difference between the original tensor and the decomposed version) can be computationally challenging, particularly for high-dimensional tensors or when the rank is unknown.

The CP decomposition represents a tensor as a summation of component tensors, each with a rank of one. In particular, for a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, the CP decomposition allows us to express it as:

$$\mathcal{X} = \sum_{r=1}^R a_r \circ b_r \circ c_r \quad (1)$$

here, $a_r \in \mathbb{R}^I$, $b_r \in \mathbb{R}^J$, and $c_r \in \mathbb{R}^K$ are vectors, and \circ denotes the outer product [4]. This decomposition allows us to break down the tensor \mathcal{X} into R simpler rank-one tensors, aiding in the processing of high-dimensional data. The smallest positive integer R for which the equation (1) holds exactly is referred to as the rank of the tensor [6]. Contrary to matrices, the computation of tensor rank poses significant challenges, see [10]. Our paper does not tackle these challenges; instead, we focus on computing the CP decomposition of a tensor with a predefined number of rank-one components. The optimization problem related to (1) can be formulated as follows:

$$\min_{a_r, b_r, c_r} \quad \frac{1}{2} \left\| \mathcal{X} - \sum_{r=1}^R a_r \circ b_r \circ c_r \right\|_F^2 \quad (2)$$

where $\|\cdot\|_F$ represent the Frobenius norm. The optimization problem (2) is ill-posed and does not have a unique solution. The ill-posedness of the problem arises from what is referred to as scaling indeterminacy in tensor literature [4]. Some other authors have employed different formulations equipped with nuclear norms to address this issue. Unlike the formulation in (2), the nuclear norm problem leads to a lower semi-continuous function, which possesses a unique solution [11, 12].

The main contributions of this paper are summarized as follows:

- We introduce an innovative iteration of the LM algorithm tailored for the CP decomposition. Our approach addresses the computational intensity typically associated with traditional methods by streamlining the process for greater efficiency.
- A comprehensive evaluation is conducted using various datasets, ranging from synthetically generated tensors to RGB images.

This paper is structured as follows. In Section 1, we review the Levenberg-Marquardt (LM) algorithm and formulate the CP problem as a nonlinear least squares problem. Section 2 describes our proposed algorithm. Section 3 discusses numerical experiments, highlighting the performance and capabilities of our proposed method. Finally, Section 4 concludes the paper, summarizing our key findings and offering insights for potential future work.

1 Problem Formulation

The formulation of CP as a nonlinear least squares was first introduced by Paatero [13]. Subsequent authors have extended this approach to various tensor-related problems [14–16]. In this section, we reformulate the problem (2) within the nonlinear least squares framework and derive the essential tools required for the development of our algorithm.

Let R be a fixed positive integer. We introduce the vector-valued function $F : \mathbb{R}^{R(I+J+K)} \rightarrow \mathbb{R}^{IJK}$ characterized by its component functions as:

$$F_{\alpha(i_1, i_2, i_3)}(x) = x_{i_1 i_2 i_3} - \sum_{r=1}^R a_r(i_1) b_r(i_2) c_r(i_3), \quad (3)$$

where $\alpha(i_1, i_2, i_3) = i_1 + (i_2 - 1)I + (i_3 - 1)IJ$. The vector $x \in \mathbb{R}^{R(I+J+K)}$ is constructed by concatenating the vectors a_r , b_r , and c_r , resulting in:

$$x = (a_1^T \ \dots \ a_R^T \ b_1^T \ \dots \ b_R^T \ c_1^T \ \dots \ c_R^T)^T. \quad (4)$$

The factor matrices corresponding to (2), expressed in terms of the vectors used to form x , are:

$$A = (a_1 \ \dots \ a_R), \quad B = (b_1 \ \dots \ b_R), \quad C = (c_1 \ \dots \ c_R), \quad (5)$$

Therefore, the variable x is defined as the column-wise stacking of the columns of matrices A , B , and C , that is,

$$x^T = [\text{vec}(A)^T \text{vec}(B)^T \text{vec}(C)^T] \quad (6)$$

where $\text{vec}(\cdot)$ denotes the vectorization of a matrix, converting it into a column vector by stacking its columns. The problem (2) can now be reformulated as the following nonlinear least squares problem:

$$\min_{x \in \mathbb{R}^{R(I+J+K)}} \frac{1}{2} \|F(x)\|_2^2. \quad (7)$$

The reformulation enables the use of established optimization techniques for nonlinear least squares problems, offering both computational efficiency and insight into the problem's structure. There are numerous techniques available for this specific category of optimization problems [17]. In this work, we specifically opt for the LM method to address the problem (7) primarily due to specific characteristics of the Jacobian matrix associated with this problem. The following lemma details the structure of the Jacobian matrix of F for the problem (7).

Lemma 1. [18] *The Jacobian matrix, J , of the function F in (7) can be expressed as:*

$$J = (J_a \ J_b \ J_c). \quad (8)$$

Each component, J_a , J_b , and J_c , can be further decomposed as:

$$J_a = (J_a^1 \ J_a^2 \ \dots \ J_a^R), \quad J_b = (J_b^1 \ J_b^2 \ \dots \ J_b^R), \quad J_c = (J_c^1 \ J_c^2 \ \dots \ J_c^R). \quad (9)$$

The individual matrices, J_a^r , J_b^r , and J_c^r , are defined as:

$$J_a^r = -c_r \otimes b_r \otimes I, \quad J_b^r = -c_r \otimes I \otimes a_r, \quad J_c^r = -I \otimes b_r \otimes a_r, \quad (10)$$

Here \otimes represents the Kronecker product of vectors. The Jacobian matrix J is of size $Q \times P$, and, due to the appearance of the identity matrix in equation (10), it has a sparse structure. It has $3RQ$ nonzero elements. Figure 1 illustrates the Jacobian matrix for a $3 \times 4 \times 5$ tensor with an estimated rank of $R = 3$. The matrix has dimensions 60×36 . In total, there are $3 \times 3 \times 60 = 540$ non-zero elements.

Recall that the LM scheme determines the step size h_{LM} at each iteration by solving the following linear system of equations:

$$(J^T J + \mu I) h_{LM} = -J^T F \quad (11)$$

where $\mu > 0$ is the *damping parameter*. This parameter plays a crucial role throughout the iterations. It can be interpreted as a regularization parameter. The quadratic convergence rate of the LM method, particularly when $\mu_k = \|F_k\|^\delta$, $\delta \in [1, 2]$, has been discussed in [19, 20]. Another strategy for determining the damping parameter involves the use of a *gain ratio*. This metric serves as an indicator of the accuracy with which the linear model approximates the objective function. Specifically, if the linear model at the k th iteration offers a satisfactory approximation, it may be useful to reduce the damping parameter. Conversely, if the approximation is deemed inadequate, an increase in the damping parameter might be warranted [17]. In addition to the aforementioned issue, two major computational challenges arise when employing the LM method. Firstly, the dimensionality of the Jacobian matrix can become exceedingly large, particularly for substantial datasets. Specifically, with a size of $IJK \times R(I + J + K)$, the matrix demands significant memory storage at every iteration, posing potential constraints on computational resources. To address this challenge, Tomasi and Bro [21], proposed working directly with $J^T J$ and $J^T F$ rather than evaluating J at every iteration. By leveraging the symmetric and sparse structure of $J^T J$, they optimized computational efforts and significantly reduced memory requirements. Figure 1 illustrates the $J^T J$ for a $3 \times 4 \times 5$ tensor with an estimated rank of $R = 3$. The matrix has dimensions 36×36 . In total, there are 954 nonzero elements.

The other significant challenge is the high computational requirement of calculating the entries of the Jacobian, as detailed in Lemma 1. There are several strategies to mitigate this issue. One approach is to bypass the direct evaluation of the Jacobian matrix and instead approximate it using available quasi-Newton methods. However, it is important to note that adopting this strategy would reduce the convergence rate to superlinear. Another approach, particularly effective when we are sufficiently close to the solution, is to utilize the current Jacobian J_k for the subsequent few iterations, alleviating the need for constant recomputation.

In response to the latter challenge, inspired by the methods in [22], we introduce an iterative method utilizing the LM scheme to obtain the CP decomposition of a third-order tensor. This methodology can be naturally extended to tensors of order n . At each iteration k , we use the current Jacobian J_k to compute both the LM step h_k and an approximate LM step \hat{h}_k . That is, we solve the following normal equation:

$$(J_k^T J_k + \mu_k I) \hat{h}_k = -J_k^T F(y_k), \quad y_k = x_k + h_k. \quad (12)$$

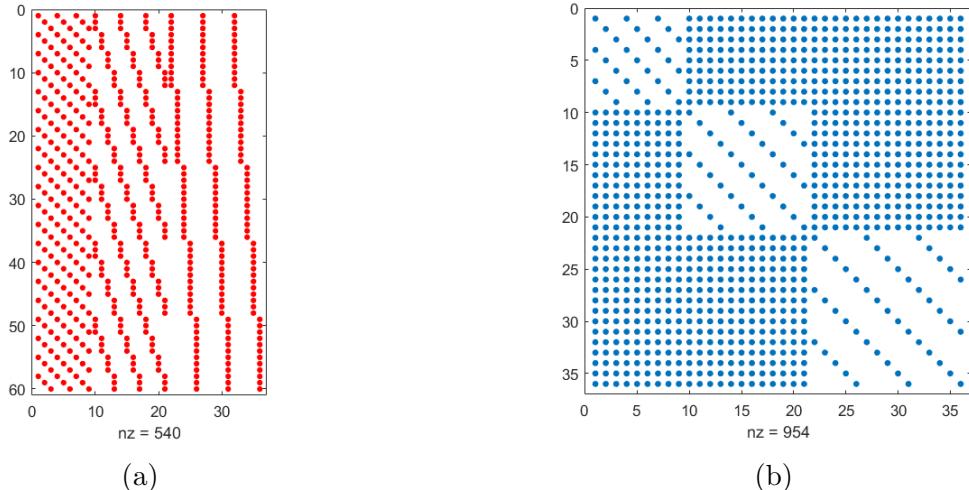


Figure 1: (a) The sparsity of the Jacobian matrix for a $3 \times 4 \times 5$ tensor with $R = 2$. The red points indicate the nonzero entries of J . (b) The symmetry and sparsity of the $J^T J$ for a $3 \times 4 \times 5$ tensor with $R = 2$. The blue points indicate the nonzero entries.

In the subsequent section, we introduce a modified LM algorithm tailored for solving the CP decomposition, which builds upon this approach.

2 A Modified Levenberg-Marquardt Algorithm for CP Decomposition

Recall that the Gauss–Newton method [23] is based on the linear approximation of the function F presented in (3), in the vicinity of x_k . That is,

$$\ell(h) = F(x_k) + J_k h. \quad (13)$$

This gives us an approximation of the nonlinear least squares problem (7) as follows:

$$\begin{aligned} L(h) &= \frac{1}{2} \ell(h)^T \ell(h) \\ &= \frac{1}{2} F(x_k)^T F(x_k) + h^T J_k^T F(x_k) + \frac{1}{2} h^T J_k^T J_k h \end{aligned} \quad (14)$$

If J_k is full rank, then the unique minimizer of $L(h)$ can be found by solving the following normal equation:

$$(J_k^T J_k) h = -J_k^T F(x_k) \quad (15)$$

However, the Jacobian matrix in Lemma 1 does not possess full rank. Specifically, the rank of this matrix is always bounded above by $P - 2R$. This issue can be addressed by introducing a damping parameter μ_k in each iteration. The damped Gauss-Newton (dGN) algorithm addresses this by solving the following normal equation in each iteration:

$$(J_k^T J_k + \mu_k I) h = -J_k^T F(x_k). \quad (16)$$

The inclusion of the damping parameter μ_k in equation (16) ensures that the matrix $J_k^T J_k + \mu_k I$ is positive definite, which consequently guarantees that the step size h is uniquely determined for each iteration. This approach can also be interpreted within the framework of trust region methods. Specifically, we are effectively solving the following constrained optimization problem at each iteration:

$$\min_h L(h) \quad \text{s.t.} \quad \|h\|_2 \leq \Delta_k, \quad (17)$$

where Δ_k is the trust region radius, analogous to μ_k , that determines the size of the region within which the model is trusted to be an accurate representation of the objective function. In order to make use of the current Jacobian J_k we also predict the next step \hat{h}_k by solving the normal equation (12) and set the new step size $s_k = h_k + \hat{h}_k$. The trial step $y_{k+1} = x_k + s_k$ is considered. In order to make use of the current Jacobian J_k , we also predict the next step \hat{h}_k by solving the normal equation (12) and set the new step size $s_k = h_k + \hat{h}_k$. The trial step $y_{k+1} = x_k + s_k$ is then considered. Note that in equation (12), there is no need to compute the Jacobian matrix at y_k . Only the function value $F(y_k)$ is required to solve (12). As a result, the cost of obtaining \hat{h}_k will be relatively inexpensive. This can be especially advantageous when the current iteration k is close enough to the solution. To test whether or not to accept the trial step, we introduce the gain ratio at every iteration:

$$\rho_k = \frac{\text{actual reduction}}{\text{predicted reduction}} = \frac{\|F(x_k)\| - \|F(x_k + s_k)\|}{\|F(x_k)\| - \|\ell(h_k)\| + \|F(y_k)\| - \|\ell(\hat{h}_k)\|} \quad (18)$$

If ρ_k is greater than a predefined threshold or is at least positive, we accept the trial step as the linear model (13) is working satisfactorily. Conversely, if ρ_k is less than the gain threshold, the linear model is not a good approximation of F , so the trial step is rejected. Consequently, we increase the damping parameter μ by multiplying it with a constant factor. Algorithm 1 summarizes the method described in this section.

tensor size	rank	LM		Modified LM		comp
		time	residual error	time	residual error	
$35 \times 25 \times 15$	40	65.8	306.4973	49.3	306.497	77
$20 \times 20 \times 12$	30	11.4	84.706	7.4	84.710	68
$28 \times 18 \times 16$	35	27	163.83	19.1	163.94	73

Table 1: LM vs Modified LM

image	rank	time(m)	residual error	comp
Rose	20\50\75	3.4\14.5\25.9	157.4\13.1\2.07	87\67\50
Leopard	20\50\80	19.5\78\178	165\83.7\53	92\79\67
Pepper	25\50\80	30.1\95.2\209	302.5\302.5\95.2\29.3	90\79\68

Table 2: **Modified LM:** time vs. error vs. compression on RGB images

Algorithm 1 Modified LM for CP decomposition

```

1: input: Initial factor matrices  $A$ ,  $B$  and  $C$ , third order tensor  $\mathcal{X}$ 
2: stopping criteria: max iteration  $N$ , error tolerance  $\epsilon$ 
3: parameters: damping parameter  $\mu > 0$ , gain ratio constant  $\gamma$ , damping parameter multiple  $\nu > 1$ 
4: initialize: evaluate the initial  $x$  and  $J$  by equations (6) and (1)
5: while stopping criteria not met do
6:   solve:  $(J_k^T J_k + \mu I) h_k = -J_k^T F(x_k)$ 
7:   set:  $y_k = x_k + h_k$ 
8:   solve:  $(J_k^T J_k + \mu I) \hat{h}_k = -J_k^T F(y_k),$ 
9:   set the trial step:  $s_k = h_k + \hat{h}_k, \quad y_{k+1} = x_k + s_k$ 
10:  compute:  $\rho_k$  as in (18)
11:  if  $\rho_k > \gamma$  then
12:    accept trial step and decrease damping parameter:

$$x_{k+1} = y_{k+1}, \quad \text{reset } \nu, \quad \mu = (1/2)\mu$$

13:  else
14:    reject trial step and increase damping parameter:

$$\mu = \nu\mu, \quad \nu = 2\nu$$

15:  end if
16: end while
17: for  $r = 1 : R$  do
18:    $A_{est}(:, r) = x((r-1)I + 1 : rI)$ 
    $B_{est}(:, r) = x(RI + (r-1)J + 1 : RI(r)J)$ 
    $C_{est}(:, r) = x(R(I+J) + (r-1)K + 1 : R(I+J) + (r)K)$ 
19: end for
20: output: three factor matrices  $A_{est}$ ,  $B_{est}$  and  $C_{est}$  of the CP decomposition.

```

3 Numerical Experiments

In this section, we evaluate and compare the performance of the proposed algorithm through a set of numerical experiments from the standard tensor-based Levenberg-Marquandt method. The computations were carried out on UAB's high-performance computing system (Cheaha), running MATLAB 2023a with a Pascalnodes node partition and 18 GB of memory per CPU with 2 GPUs and 12 CPUs per node. In

our numerical experiments, we tested two types of data: RGB images and randomly generated tensors with varying rank, mode dimensions, and other features.

To demonstrate the effectiveness of the Modified LM method, various third-order tensors of RGB images were reconstructed. In Figure 3, the Modified LM reconstructed three RGB images: rose, pepper, and leopard. The compression percentage is calculated from the ratio: $\frac{R(I+J+K)}{I \times J \times K}$ for a third order tensor of size $I \times J \times K$ of rank R . The ratio measures how much an image has been reduced in size. Clearly, the higher the compression percentage, the greater the reduction in the size of the image. In Figure 3, the images in the middle and last columns are reconstructed tensor images with varying ranks and, hence, compression percentages. The images look indistinguishable from the original images. Table 2 shows that a higher compression requires less time but with low accuracy. Similarly, Table 3 follows the same trend in Table 2 with randomly generated tensors for some specific sizes. Lastly, Table 1 compares the two algorithms, LM and modified LM, for random tensors of sizes: $35 \times 25 \times 15$ with $R = 40$, $20 \times 20 \times 12$ with $R = 30$, and $28 \times 18 \times 16$ with $R = 35$. In all three cases, CPU times for modified LM is much less than for LM. The smooth evolution of residual errors during the model execution is shown in Figure 2.

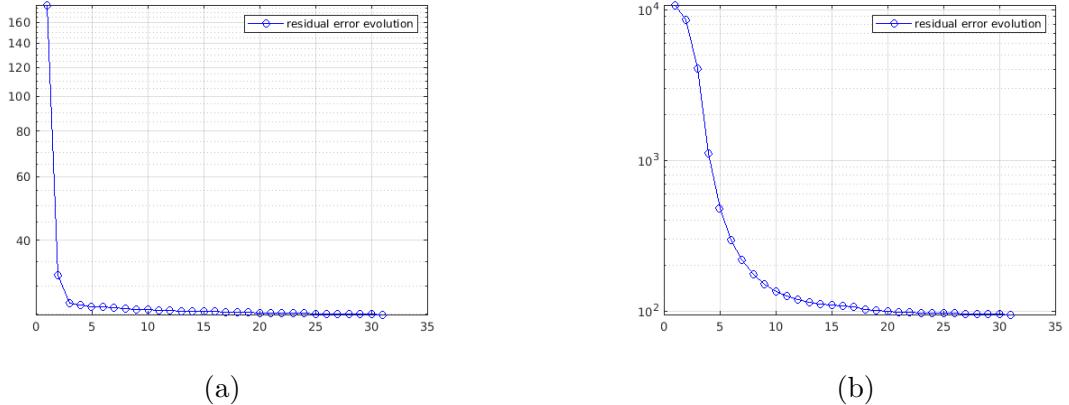


Figure 2: Modified LM: The performance of the proposed algorithm in terms of accuracy of the residual error for (a) a randomly generated tensor ($45 \times 35 \times 25$), $R=40$, 89% compression and (b) Leopard image $R = 50$, 79% compression.

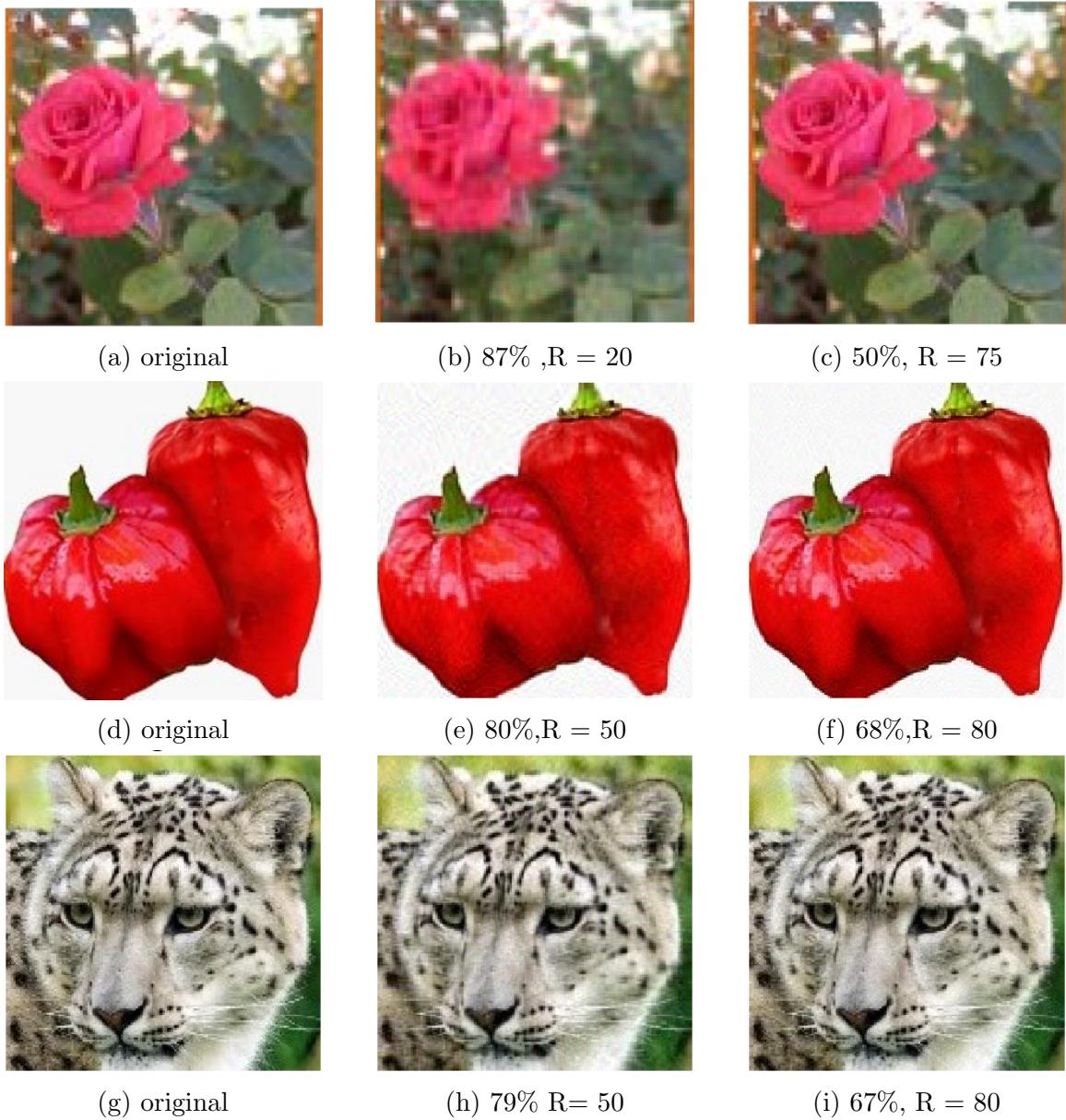


Figure 3: **Modified LM**: Rose($100 \times 100 \times 3$), Pepper ($168 \times 168 \times 3$) and Leopard($162 \times 162 \times 3$).

tensor size	rank	time cost(m)	residual error	comp
$45 \times 35 \times 20$	25\40\60	1.9\3.5\5.9	21.7\19.5\16.7	91\87\81
$45 \times 35 \times 25$	25\40\60	2.6\4.9\7.5	24.8\22.8\20.08	93\89\84
$45 \times 35 \times 30$	25\40\60	3\5.4\9.1	27.7\25.6\22.0	94\91\86

Table 3: **Modified LM**: time vs. error vs. compression on randomly generated third order tensors

4 Conclusion

In this work, we proposed a new method for reconstructing higher-order low rank tensors in the canonical polyadic decomposition format. We build on the success of the Levenberg-Marquardt approach on approximating a well-scaled solution of a nonlinear least-squares problem. However, there are drawbacks and challenges in the LM method. Our approach is a modified LM which decreases the computational demand on the calculation of the Jacobian and objective function evaluation. From numerical experiments, the modified LM performed well in compressing RGB images and randomly generated tensors while keeping the tensor rank low. The modified LM has significantly lower CPU times in all experimental cases compared to the LM. In conclusion, a modified LM provides sensible compression for low-rank CP reconstruction of tensors with the same order of accuracy as the LM method. This is achieved while leveraging a noticeable reduction in CPU time cost. In our future outlook, we will add sampling methods to tackle massively high mode dimensions in high-order tensors and will provide theoretical convergence results.

References

- [1] Boris N Khoromskij, “Tensors-structured numerical methods in scientific computing: Survey on recent advances,” *Chemometrics and Intelligent Laboratory Systems*, vol. 110, no. 1, pp. 1–19, 2012.
- [2] Lars Grasedyck, Daniel Kressner, and Christine Tobler, “A literature survey of low-rank tensor approximation techniques,” *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, 2013.
- [3] Ramin Goudarzi Karim, Guimu Guo, Da Yan, and Carmeliza Navasca, “Accurate tensor decomposition with simultaneous rank approximation for surveillance videos,” in *2020 54th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2020, pp. 842–846.
- [4] Tamara G Kolda and Brett W Bader, “Tensor decompositions and applications,” *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.
- [5] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos, “Tensor decomposition for signal processing and machine learning,” *IEEE Transactions on signal processing*, vol. 65, no. 13, pp. 3551–3582, 2017.
- [6] Frank L Hitchcock, “The expression of a tensor or a polyadic as a sum of products,” *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.
- [7] Jiahua Jiang, Fatoumata Sanogo, and Carmeliza Navasca, “Low-cp-rank tensor completion via practical regularization,” *Journal of Scientific Computing*, vol. 91, no. 1, pp. 18, 2022.
- [8] Miguel A Veganzones, Jeremy E Cohen, Rodrigo Cabral Farias, Jocelyn Chanussot, and Pierre Comon, “Nonnegative tensor cp decomposition of hyperspectral data,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 5, pp. 2577–2588, 2015.
- [9] Mingyi Zhou, Yipeng Liu, Zhen Long, Longxi Chen, and Ce Zhu, “Tensor rank learning in cp decomposition via convolutional neural network,” *Signal Processing: Image Communication*, vol. 73, pp. 12–21, 2019.
- [10] Christopher J Hillar and Lek-Heng Lim, “Most tensor problems are np-hard,” *Journal of the ACM (JACM)*, vol. 60, no. 6, pp. 1–39, 2013.

- [11] Shmuel Friedland and Lek-Heng Lim, “Nuclear norm of higher-order tensors,” *Mathematics of Computation*, vol. 87, no. 311, pp. 1255–1281, 2018.
- [12] Ming Yuan and Cun-Hui Zhang, “On tensor completion via nuclear norm minimization,” *Foundations of Computational Mathematics*, vol. 16, no. 4, pp. 1031–1068, 2016.
- [13] Pentti Paatero, “A weighted non-negative least squares algorithm for three-way ‘parafac’ factor analysis,” *Chemometrics and Intelligent Laboratory Systems*, vol. 38, no. 2, pp. 223–242, 1997.
- [14] Giorgio Tomasi and Rasmus Bro, “Parafac and missing values,” *Chemometrics and Intelligent Laboratory Systems*, vol. 75, no. 2, pp. 163–180, 2005.
- [15] Jinyao Zhao, Xuejuan Zhang, and Jinling Zhao, “A levenberg-marquardt method for tensor approximation,” *Symmetry*, vol. 15, no. 3, pp. 694, 2023.
- [16] Shih Yu Chang, Hsiao-Chun Wu, Yen-Cheng Kuan, and Yiyan Wu, “Tensor levenberg-marquardt algorithm for multi-relational traffic prediction,” *IEEE Transactions on Vehicular Technology*, 2023.
- [17] Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff, “Methods for non-linear least squares problems,” *lecture notes*, 2004.
- [18] Evrim Acar, Daniel M Dunlavy, and Tamara G Kolda, “A scalable optimization approach for fitting canonical tensor decompositions,” *Journal of chemometrics*, vol. 25, no. 2, pp. 67–86, 2011.
- [19] Jin-yan Fan and Ya-xiang Yuan, “On the quadratic convergence of the levenberg-marquardt method without nonsingularity assumption,” *Computing*, vol. 74, pp. 23–39, 2005.
- [20] Jinyan Fan and Jianyu Pan, “A note on the levenberg–marquardt parameter,” *Applied Mathematics and Computation*, vol. 207, no. 2, pp. 351–359, 2009.
- [21] Giorgio Tomasi and Rasmus Bro, “A comparison of algorithms for fitting the parafac model,” *Computational Statistics & Data Analysis*, vol. 50, no. 7, pp. 1700–1734, 2006.
- [22] Jinyan Fan, Jianchao Huang, and Jianyu Pan, “An adaptive multi-step levenberg–marquardt method,” *Journal of Scientific Computing*, vol. 78, pp. 531–548, 2019.
- [23] Edwin KP Chong, Wu-Sheng Lu, and Stanislaw H Żak, *An introduction to optimization*, John Wiley & Sons, 2023.