

Tensor networks for scalable probabilistic modeling

Menzen, C.M.

DOI

[10.4233/uuid:de9e6c14-b0c5-43b9-a821-c8d231e1eee8](https://doi.org/10.4233/uuid:de9e6c14-b0c5-43b9-a821-c8d231e1eee8)

Publication date

2025

Document Version

Final published version

Citation (APA)

Menzen, C. M. (2025). *Tensor networks for scalable probabilistic modeling*. [Dissertation (TU Delft), Delft University of Technology]. <https://doi.org/10.4233/uuid:de9e6c14-b0c5-43b9-a821-c8d231e1eee8>

Important note

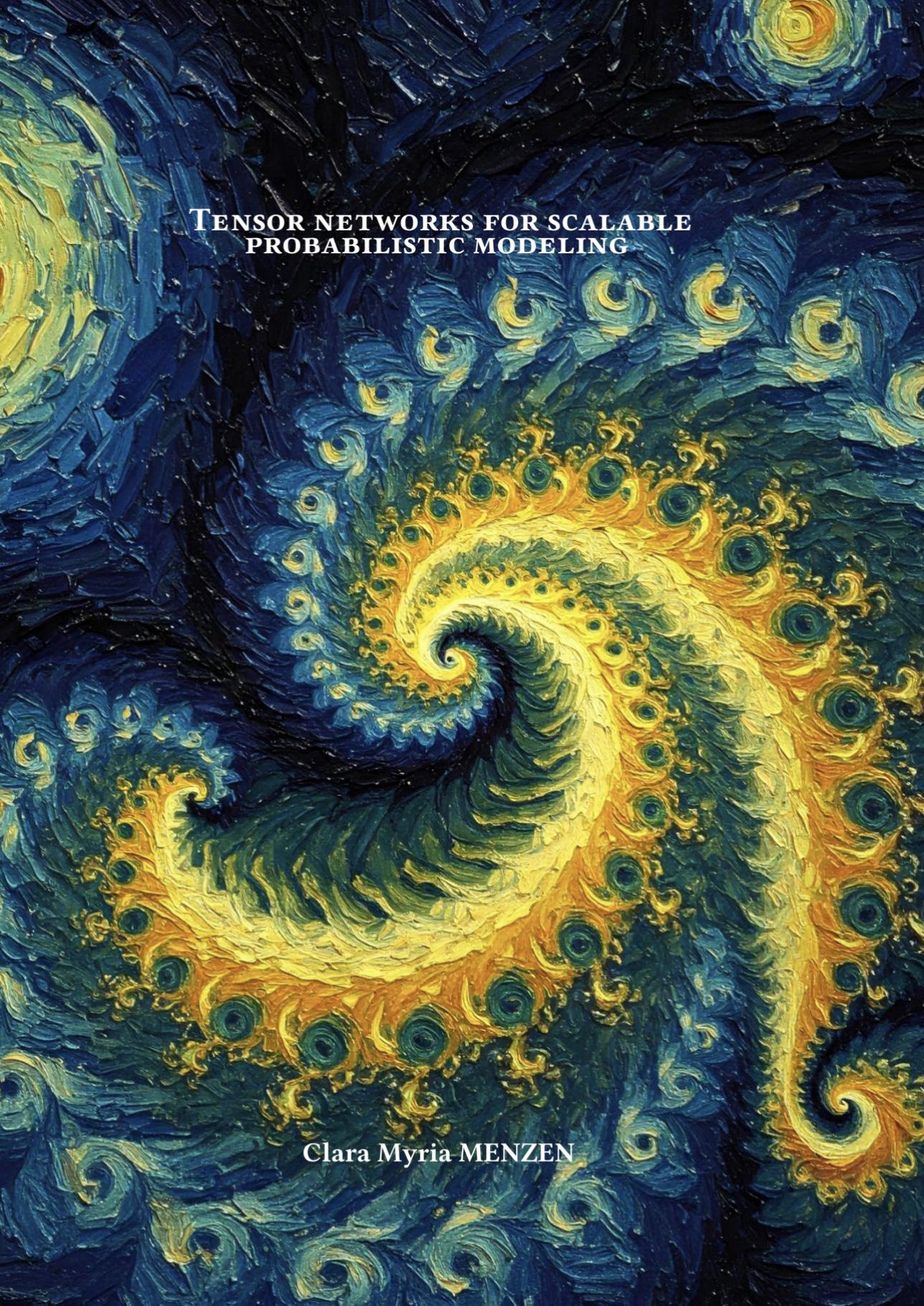
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.



**TENSOR NETWORKS FOR SCALABLE
PROBABILISTIC MODELING**

Clara Myria MENZEN

TENSOR NETWORKS FOR SCALABLE PROBABILISTIC MODELING

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board of Doctorates
to be defended publicly on Friday, November 28, 2025, at 12:30

by

Clara Myria MENZEN

Master of Science in Engineering Science,
Technische Universität Berlin, Germany,
born in Rheda-Wiedenbrück, Germany.

This dissertation has been approved by the promotor.

Composition of the doctoral committee:

Rector Magnificus,	Chairperson
prof.dr.ir. J.W. van Wingerden,	Delft University of Technology, promotor
dr. M. Kok	Delft University of Technology, promotor
dr.ir. K. Batselier	Delft University of Technology, copromotor

Independent memebers:

prof.dr.ir. G. Jongbloed,	Delft University of Technology
dr. Frans A. Oliehoek,	Delft University of Technology
Prof. Dr. M. Stoll,	Chemnitz University of Technology, Germany
Dr. F. Govaers,	Fraunhofer FKIE, Germany
prof.dr.ir. B. De Schutter,	Delft University of Technology, reserve member



Keywords: Probabilistic modeling, tensor networks, Gaussian process regression.

Cover: The cover page was generated using Gemini 2.0 Flash with the prompt: “Generate a book cover of Mandelbrot fractals in the style of Van Gogh.” The fractals represent an artistic metaphor for large-scale structures repeating at smaller scales, representing the structural assumptions and correlations underlying low-rank tensor networks. The Van Gogh style pays homage to the Netherlands, which has been a wonderful home to me during my research and beyond.

Style: TU Delft House Style, with modifications by Moritz Beller
<https://github.com/Inventitech/phd-thesis-template>

An electronic version of this dissertation is available at
<http://repository.tudelft.nl/>.

I am still in progress, and I hope that I always will be.

Michelle Obama

To Mama, Papa, Luisa and Cristian

CONTENTS

Summary	xi
Samenvatting	xiii
Acknowledgements	xvii
1 Introduction	1
1.1 Probabilistic algorithms and their scalability challenges	3
1.2 Tensor networks: A tool for scalability	6
1.2.1 From low-rank matrix decompositions to low-rank TNs	6
1.2.2 Efficient storage and computations with TNs.	8
1.3 Contributions of this thesis.	11
2 Alternating linear scheme in a Bayesian framework for low-rank tensor approximation	19
2.1 Introduction	21
2.2 Tensor basics and notation	23
2.3 Bayesian inference for low-rank tensor approximation.	28
2.4 Orthogonalization step in Bayesian framework for a TT	32
2.5 Unscented transform in TT format	38
2.6 Numerical experiments.	41
2.6.1 Convergence analysis of maximization problem	42
2.6.2 Analysis of covariance matrices	42
2.6.3 Comparison to conventional ALS.	43
2.6.4 Reconstruction of noisy image	45
2.6.5 Large-scale experiment.	47
2.7 Conclusions	49
3 Tensor network square root Kalman filter for online Gaussian process regression	55
3.1 Introduction	57
3.2 Problem Formulation.	58
3.3 Background on tensor networks	59
3.3.1 Tensor networks	59
3.4 TNSRKF	61
3.4.1 Update of weight mean.	61
3.4.2 Update of square root covariance factor	62
3.4.3 Predictions.	63
3.5 Implementation	64
3.5.1 Updating $\hat{\mathbf{w}}_t$ in TN format	64
3.5.2 Updating \mathbf{L}_t in TT format	66

3.6	Experiments	70
3.6.1	Equivalence of full-rank TNSRKf and Kalman filter	70
3.6.2	Influence of the ranks on the approximation	71
3.6.3	Comparison to TNKF for cascaded tanks benchmark data set.	72
3.7	Conclusion.	74
4	Large-scale magnetic field maps using structured kernel interpolation for Gaussian process regression	79
4.1	Introduction	81
4.2	Problem formulation	82
4.3	SKI framework	83
4.4	Large-scale magnetic field maps	84
4.5	Experiments	85
4.5.1	Accuracy analysis for growing mapping area.	85
4.5.2	Analysis of maps with divided mapping area	87
4.5.3	Large-scale map in university building	88
4.6	Conclusion.	89
5	Conclusions and future work recommendations	97
5.1	Summary of findings.	99
5.1.1	Development of Bayesian ALS algorithm.	99
5.1.2	Solving the issue of loss of positive definiteness in TN Kalman filtering	100
5.1.3	Application of structured kernel interpolation to magnetic field mapping	100
5.2	Future work recommendations.	100
	Curriculum Vitæ	103
	List of Publications	105

SUMMARY

Probabilistic or Bayesian modeling plays a fundamental role in engineering and science, providing a framework for integrating noisy measurements with predictive models through probability distributions. While probabilistic methods have many benefits, such as recursive estimation and uncertainty quantification, they often come with substantial memory and compute requirements. Computational challenges are particularly pronounced in large-scale settings, where data sets contain a high number of measurements, and for high-dimensional problems, which require exponentially many parameters to describe probability distributions. These scenarios can suffer from the curse of dimensionality, which requires exponentially growing computing resources, making conventional approaches computationally intractable.

This dissertation addresses computational challenges by leveraging tensor networks (TNs) to develop computationally efficient probabilistic algorithms. TNs, also known as tensor decompositions, extend matrix decomposition to higher dimensions by representing large multidimensional arrays, i.e., tensors, in a compact, decomposed format, defined by TN components and TN ranks. Under the assumption of low-rank structure, TNs enable efficient storage and computation, making large-scale and high-dimensional problems more tractable, even on resource-constrained hardware such as conventional laptops. The focus of this work is on scalable solutions for Bayesian estimation problems involving Gaussian distributions and exact inference, including recursive filtering and Gaussian process (GP) regression. The three main contributions of this dissertation each address a specific challenge in this domain, demonstrating how TNs and probabilistic modeling can benefit from each other.

The first contribution presents an algorithm to compute a low-rank tensor approximation in terms of a low-rank TN by solving a Bayesian inference problem. By treating the TN components as Gaussian random variables with prior distributions and dividing the overall inference problem into subproblems, the posterior distributions of the TN components are computed sequentially. This approach provides a probabilistic interpretation of the iterative algorithm, called alternating linear scheme (ALS), commonly used to compute low-rank TNs. The tensor represented by the low-rank TN can be computed from the joint posterior of all TN components, providing a mean estimate together with a measure for uncertainty quantification. The ALS in a Bayesian framework allows for the integration of prior application-specific knowledge and consideration of the measurement noise. In addition, it enables uncertainty quantification for the low-rank tensor estimate. In this context, the conventional ALS can be seen as a special case of the ALS in a Bayesian framework, where each subproblem is solved with an uninformative prior. The computational cost is in the same range as for the conventional ALS, only the memory requirements have

an additional term for the covariance matrices of the TN components that need to be stored.

The second contribution addresses the loss of positive (semi-) definiteness in TN-based Kalman filtering. While the state-of-the-art TN Kalman filter (TNKF) enables logarithmic compression of computational and storage requirements in high-dimensional recursive estimation problems, a required TN-specific operation, called rounding, can cause filter divergence. This is because covariance matrices represented as TNs can lose positive (semi-) definiteness after rounding is applied. Positive (semi-) definiteness is only guaranteed in a special case where all TN ranks are chosen to be equal to one, which limits the accuracy of the covariance matrix representation. To resolve this issue, this contribution implements the square root formulation of the Kalman filter in TN format, the tensor network square-root Kalman filter (TNSRKf). In this context, mean estimates and square root covariance matrices are represented as low-rank TNs and updated in TN format, using the ALS for each iteration of the filter. While this approach has similar computational complexity to the TNKF, it does not run into problems related to positive (semi-) definiteness, because it computes the square root covariance factors instead of the covariance matrix directly. The TN ranks of the square root covariance factors are not restricted, such that the covariance matrix can be represented with higher accuracy than in the TNKF without the risk of losing positive (semi-) definiteness. The TNSRKf is applied to online GP regression, which in this context corresponds to the measurement update of Kalman filter. While standard online GP for high-dimensional inputs using a product kernel requires an exponential amount of compute, the TNSRKf reduces the compute to be linear in the dimensionality of the problem.

The third contribution presents an application of large-scale probabilistic modeling in the field of sensor fusion: constructing scalable magnetic field maps in 3D with GP regression. Spatial variations in the ambient magnetic field, which are e.g. caused by ferromagnetic materials present in building materials, can be used for localization algorithms in indoor areas. GP regression has many benefits for modeling magnetic fields and building magnetic field maps since it allows for the incorporation of physical knowledge and provides a map estimate alongside an uncertainty which is necessary for probabilistic localization algorithms. However, GP regression in its standard setting has a cubic computational complexity with respect to the number of data points, limiting its usage to small data sets. This contribution builds on the structured kernel interpolation (SKI) framework, a scalable GP technique that approximates the kernel function with a grid-based interpolation strategy. SKI with derivatives (D-SKI) is incorporated into the scalar potential model for magnetic field modeling, enabling the efficient computation of large-scale magnetic field maps. This is achieved by exploiting Kronecker algebra in Krylov subspace methods to accelerate matrix-matrix multiplications during GP training and inference. Although using Kronecker algebra is not strictly a TN method, it is closely related, as both approaches exploit underlying structure to improve computational efficiency. In this way, large-scale magnetic field maps can be computed with a computational complexity that is linear in the number of data points.

SAMENVATTING

Probabilistische of Bayesiaanse modellen spelen een cruciale rol in de techniek en in de wetenschap, omdat ze de kansverdeling van ruisgevoelige metingen kunnen integreren in voorspellende modellen. Hoewel probabilistische methoden veel voordelen bieden, zoals recursieve schattingen en onzekerheidskwantificatie, kosten deze vaak veel geheugen en rekenkracht. De computationele uitdagingen zijn vooral uitgesproken in toepassingen waar veel data voor nodig is, en bij hoog-dimensionale problemen, waarvoor exponentieel veel parameters nodig zijn om verdelingen te beschrijven. Deze scenario's kunnen lijden onder de zogeheten “curse of dimensionality”, wat betekent dat er een exponentieel groeiende rekenkracht nodig is, waardoor traditionele benaderingen computationeel onhaalbaar worden.

Dit proefschrift pakt computationele uitdagingen aan door gebruik te maken van tensor-netwerken (TNs) om computationeel efficiënte probabilistische algoritmen te ontwikkelen. TNs, ook wel tensor decompositions genoemd, breiden matrix decomposities uit naar hogere dimensies door grote multidimensionale arrays, d.w.z. tensors, op een compacte manier weer te geven. Onder de aanname van een lage-rangstructuur maken TNs efficiënte opslag en berekeningen mogelijk, waardoor grootschalige en hoog-dimensionale problemen beter hanteerbaar worden, zelfs op hardware met beperkte middelen zoals conventionele laptops. Dit werk focust op schaalbare oplossingen voor Bayesiaanse schattingsproblemen met Gaussiaanse verdelingen en exacte inferentie, waaronder recursieve filtering en Gaussian Process (GP) regressie. De drie belangrijkste contributies van dit proefschrift behandelen elk een specifieke uitdaging binnen dit domein en tonen aan hoe TNs en probabilistisch modelleren elkaar kunnen versterken.

De eerste bijdrage presenteert een algoritme om lage-rang tensorbenaderingen te berekenen in termen van een lage-rang TN door het oplossen van een Bayesiaans inferentieprobleem. Door de TN-componenten te beschouwen als willekeurige Gaussiaanse variabelen met een prior verdeling, en het totale inferentieprobleem op te splitsen in deelproblemen, worden de posterior verdelingen van de TN-componenten sequentieel berekend. Deze benadering biedt een probabilistische interpretatie van het bekende iteratieve algoritme “alternating linear scheme”(ALS), dat vaak wordt gebruikt voor het berekenen van lage-rang TNs. De tensor die door de lage-rang TN wordt weergegeven kan worden berekend vanuit de gezamenlijke posterior van alle TN-componenten, wat zowel een schatting van het gemiddelde oplevert als een maat voor de onzekerheid. De ALS in een Bayesiaans kader maakt het mogelijk om toepassingsspecifieke voorkennis te integreren en rekening te houden met meetruis. Bovendien maakt dit onzekerheidskwantificatie mogelijk voor de lage-rang tensorbenadering. In deze context kan de conventionele ALS worden opgevat als een speciaal geval van de ALS in een Bayesiaans kader, waarin elk deelprobleem wordt opgelost met een niet-informerende prior. De computationele kosten

zijn gelijk aan die van de conventionele ALS; alleen de geheugeneisen nemen toe door de noodzaak om covariantiematrices van de TN-componenten op te slaan.

De tweede bijdrage behandelt het verlies van positieve (semi-) definitie in TN gebaseerde Kalman-filtering. Hoewel de geavanceerde TN Kalman-filter (TNKF) logaritmische compressie mogelijk maakt van reken- en opslagvereisten in hoog-dimensionale recursieve schattingsproblemen, kan een vereiste TN-specifieke bewerking, genaamd afronding, leiden tot divergentie van het filter. Dit komt doordat covariantiematrices die als TNs worden weergegeven hun positieve (semi-) definitie kunnen verliezen na het toepassen van afronding. Positieve (semi-) definitie is alleen gegarandeerd wanneer alle TN-rangen gelijk aan één worden gekozen, wat de nauwkeurigheid van de covariantiematrixrepresentatie beperkt. Om dit probleem op te lossen wordt in deze bijdrage de vierkantwortelformulering van het Kalman-filter geïmplementeerd in TN-formaat: de “Tensor Network Square-Root Kalman Filter” (TNSRKf). In dit kader worden verwachtingswaarden en vierkantwortel-covariantiematrices voorgesteld als lage-rang TNs en bij elke iteratie van het filter geüpdatet in TN-formaat met behulp van ALS. Hoewel deze aanpak vergelijkbare computationele complexiteit heeft als de TNKF, treden er geen problemen op met betrekking tot positieve (semi-) definitie, aangezien de vierkantwortelfactoren van de covariantie worden berekend in plaats van de matrix zelf. De TN-rangen van deze vierkantwortelfactoren zijn niet beperkt, zodat de covariantiematrix nauwkeuriger kan worden voorgesteld dan in de TNKF, zonder het risico op verlies van positieve (semi-) definitie. De TNSRKf wordt toegepast op online GP regressie, wat in deze context overeenkomt met de meetupdate van het Kalman-filter. Terwijl standaard online GP voor hoog-dimensionale ingangsignalen met een product kernel een exponentiële hoeveelheid berekeningen vereist, reduceert de TNSRKf dit tot een lineaire afhankelijkheid van de dimensionaliteit.

De derde bijdrage presenteert een toepassing van grootschalige probabilistische modellering op het gebied van sensorfusie: het construeren van schaalbare magnetisch-veldkaarten in 3D met behulp van GP regressie. Ruimtelijke variaties in het magnetisch veld, die bijvoorbeeld worden veroorzaakt door ferromagnetische materialen in de constructies van gebouwen, kunnen worden gebruikt voor lokalisatiealgoritmen in binnenomgevingen. GP regressie biedt veel voordelen voor het modelleren van magnetische velden en het opbouwen van magnetisch-veldkaarten, omdat het de integratie van fysische kennis mogelijk maakt en naast een kaartschatting ook een onzekerheidsmaat levert, wat essentieel is voor probabilistische lokalisatiealgoritmen. Standaard GP regressie heeft echter een computationele complexiteit die kubisch schaal met het aantal metingen, wat het gebruik beperkt tot kleine datasets. Deze bijdrage bouwt voort op het “structured kernel interpolation” (SKI) raamwerk, een schaalbare GP-techniek die de kernelfunctie benadert met een raster-gebaseerde interpolatiestrategie. SKI met afgeleiden (D-SKI) wordt geïntegreerd in het scalair potentiaalmodel voor het modelleren van magnetische velden, waardoor efficiënte berekening van grootschalige magnetisch-veldkaarten mogelijk wordt. Dit wordt bereikt door gebruik te maken van Kronecker-algebra, in combinatie met Krylov-subruimtemethoden om matrix-matrixvermenigvuldigingen tijdens GP-training en -inferentie te versnellen. Hoewel het gebruik van Kronecker-algebra niet strikt een

TN methode is, is het er nauw mee verwant, aangezien beide benaderingen gebruikmaken van onderliggende structuur om de rekenefficiëntie te verbeteren. Op deze manier kunnen grootschalige magnetisch-veldkaarten worden berekend met een computationele complexiteit die lineair schaalt met het aantal metingen.

ACKNOWLEDGEMENTS

In November 2019, I embarked on a new chapter by beginning my PhD in Delft. Having just graduated from the Technical University of Berlin, I was excited to explore a new country and dive into a new field of research. Over the following years, I was fortunate to receive incredible support, love, and guidance from many people in my life, for which I am deeply grateful. Although my PhD journey began under unexpected circumstances—less than six months in, the world went into Covid lockdown (and we all know how that story unfolded)—when I look back, what stands out most are the great moments and, above all, everything I learned along the way.

I would like to deeply thank my daily supervisors, Manon and Kim. You have been a mentor, an inspiration, and an aspiration by showing me daily how great academic research is done. Challenging me to go into detail, broaden my perspective, and push my boundaries, every meeting with you provided an opportunity to step out of my comfort zone and regain confidence by understanding what is going on. But you were more than just supervisors. In challenging times, you had an ear for me, and we always had good talks in lunch breaks or over beers after work. Manon, it was both inspiring and fun to work with you. I was fascinated by the way you find excitement in solving difficult problems, and that energy is truly contagious. I also really enjoyed our joint trips to Linköping, Fusion, and IFAC. Kim, thanks for making me appreciate linear algebra and going into a lot of detail (pun intended) in the world of tensor decompositions. Also, thanks for two of your (very different) recommendations: Ajahn Brahm's book to see the world through a Buddhist eye, and Bon Geuze Mariage parfait, which is now my favorite Belgian beer. Thank you, Jan-Willem, for being my promoter. I always really appreciated your calm presence throughout my PhD, and I always had the feeling my thoughts were heard.

Thanks to all my wonderful colleagues, because who says we cannot have a bit of fun while we are researching:

The sensor fusion squad: Frida - having you by my side all these years was a constant reminder for me to push my boundaries, because you just set the bar so high. It was great being able to learn from you, especially when it came to basis functions approximations and magnetic field maps, but also about life. Our trips to Linköping and Fusion conference together with Manon and Thomas were so much fun, even if a lot of unexpected things happened, like cancelled planes. You and Thomas were so excited about the 5k run, so I hopped on board and ended up running it for the three of us. Also, probably thanks to multiple joint karaoke sessions and the words of Gloria Gaynor, we actually did survive. ;) Thomas - my twin from another kin. I am already missing our daily walks (giros), dancing at Techno Spar, talking about a mixture of cat reels, uncertain inputs (which are like questioning the story of a nontrustworthy narrator, right ;)), Ramen, and homemade pasta. And (to be read only in a strong English or Scottish accent), I officially concur with

on the conundrum of the great outdoors, as defined by Nigel Brimsley Thornberry that we are the official royals of DCSC. Ruiyuan - even if short, thanks for the nice moments we spent together in the EWI tower office, as well as our office at 3ME, especially for giving me some Gen Z advice on the latest K-Pop bands (even if Frida was the one that first told me about BlackPink ;)). Haoyu - thanks for the best Pistachio chocolate I ever ate and for explaining cosplay to me. Mostafa - even if only for a brief time, I always appreciated your presence and quick thinking.

The tensor squad: Eva - Even if we studied alongside many years in Berlin without ever knowing each other, it was always great to have a 'PhDler' alongside who had a similar background. I really enjoyed our research symbioses, and I am so grateful for all the support I received from you over the years. Also, your passion and dedication to Green AI have inspired me to feel the same spark, and collaborating with you always felt super easy and fruitful. Frederiek - thanks discussions with innovative research perspectives. Jetze - thanks for nice talks during your time at DCSC and your support during my job search afterwards. Albert and Afra - even if we had only a short overlap, it was great seeing the Detail team growing and having fresh perspectives coming in.

DCSC fellows: Livia - *amicizia a prima vista*. I still remember the first time I saw you in the filtering and identification lecture. You immediately invited me to your house for dinner, and I always felt so welcomed by you. I just loved your presence and support during these years, I am so grateful that you were by my side in challenging times (when I needed advice) and fun times (like when we did our dance reel). Also, I will never forget your dreamy wedding and all the times we ran into each other on your honeymoon in Japan. TVTB. Coen - what can I say? Pink neon lights really did guide us. Thanks for the unforgettable experience of writing and producing a song together, and filming a music video in Tokyo. I really enjoyed having you as my fellow PhD candidate, for good and also fun talks about research, life, food, and melodies. And I'll just have to ask one more time: Is this cotton? And is this Snoop Dog? Claudia - it was great having you around in the office to talk about PhD life or life in general, and driving in the car while singing "the greatest love of all" at full volume. You always had great and kind advice, and it was super fun sharing a room with you in Yokohama. Marteen - thanks for the great and fun memories in Yokohama and throughout the entire PhD years. Suad - thanks fun times organizing the video competition together with Athina and the Christmas quiz online. Lars - thanks for the great talks and positive presents in the office. Sander - thanks for the fun talks in lunch breaks and your positive energy. Also, thanks to Leonore, Daniel, Mees, Markus, Jonas, Giulio, Emilio, Lotfi, Francesco, Tim, Rogier, David, Leila, Jean, Amr, Maarten dJ, Frederiek M., Roger, Emanuel, for being great colleagues to share moments with throughout all these past years. And last but not least, big thanks to Heleen, Francy, Marieke, Martha, and Erika.

Thanks to my family, without whom I would not be who I am.

Cris - amor. There are no words for an appropriate thank you for these past years. You were by my side every day, you knew what I was experiencing, and you helped me grow beyond myself. I am so happy that we got married on December 22nd 2024, and are now living our next chapter, a new life together in California. Te amo. Y estoy segura que un dia vamos a decir de verdad que estamos casados desde 50 años. ;) Mama, Papa - ihr seid die besten Eltern die ich mir hätte wünschen können. Ihr habt mich immer unterstützt und mein Selbstvertrauen habe ich durch euch aufgebaut. Danke für alles, ich hab euch unendlich doll lieb. Luisa - Schatz, Schwesterherz. Auch wenn wir für circa 10 Jahren auf zwei verschiedenen Kontinenten gelebt haben, habe ich dich jeden Tag an meiner Seite verspürt. Die Kraft und bedingungslose Liebe die du mir immer gibts, ist das schönste was ich mir auf dieser Welt hätte wünschen können. Wir sind eine Seele und ich bin so glücklich, dass wir jetzt wieder im gleichen Land (auch wenn in unterschiedlichen Staaten) wohnen. Danke, love you Schatz!! Danke auch an den Rest der Menzen Family: Bettina, Norbert, Ronja, Jannik, Markus, Karin, Simon, Jakob, Lasse und Nick. Liliana, Anselmo, Nira, Melina, Romina - muchísimas gracias para acamparame en esos últimos años con apoyo y amor. Soy tan feliz de poder llamar ustedes mi familia. Espero que vayamos a poder hacer mas viajes juntos en Argentina, Estados Unidos, Holanda... Los quiero mucho!

Thank to my friends, because they make life just 1000 times better.

Amici da 20 anni (and counting) Lau, Tina, Filo. Siete le persone con le quali voglio diventare vecchia, fare aperitivi a 90 anni e parlare di cibo, la vita e l'amore. Grazie per essere costanti nella mia vita e anche se viviamo in 4 paesi diversi, i nostri cuori sono sempre vicini. Marengo (Merry), le nostre storie ci hanno fatto incontrare in Italia, Berlino (nella mitica Knaackstrasse) e nel BeNe (Lux not so much). E anche se adesso chatGPT rimpiazza le amazing cover letters che mi scrivevi, in parte, non avrei potuto scrivere questa tesi senza di te (see cover page of Introduction). E' state un onore officiare il matrimonio tuo e di Antoine, vvb, soprattutto a Lola, jk :). Wendy, Paula, and Lora - my dance teammates, but also so much more. A hobby brought us together, but we found so many more reasons to spend time together, encourage and support each other to grow beyond ourselves. I started to feel at home in the Netherlands when we became friends. I am forever grateful for that. Also, thanks so much to all the other Pasito Ladies and Raquel Garrido for bringing so much joy into my life while sharing the passion for bachata, dancing, and self-expression. Nadine und Alex - meiner Berliner. Ich bin so froh das komplette Studien-Kapitel mit euch verbracht zu haben, und bin super glücklich dass wir weiterhin in unseren Leben sind. Nadine, im Studium hast du mir so viel Kraft und Selbstvertrauen gegeben. Wir haben uns gegenseitig geholfen, die Frauen zu werden, die wir heute sind – zwei Glitzer- und Tanzen-liebende Ingenieurinnen – und dafür bin ich dir sehr dankbar. Alex, wir haben in den letzten Jahren auf unterschiedliche Weisen ähnliches erlebt und unser Austausch ist immer sehr bereichernd. Was kann ich anderes sagen ausser: “Die Metamorphose zum Vollzeit-Nerd schmerzhaft aber ehrlich...”. Du weist wie es ausgeht und es hat jetzt eine komplett neue Bedeutung für uns.

My heartfelt thanks also go to all those I could not mention by name, yet who have played a part in my journey in some way.

1

INTRODUCTION

FIGARO misurando | FIGARO measuring
Cinque... dieci... venti... trenta... trentasei... quarantatre...
| Five... ten... twenty... thirty... thirtysix... fortythree

Le nozze di Figaro, Atto I, Mozart | The marriage of Figaro, Act I, Mozart

1.1 PROBABILISTIC ALGORITHMS AND THEIR SCALABILITY CHALLENGES

When NASA embarked on the Apollo program in the 1960s, the mission to land astronauts on the Moon required highly accurate, real-time onboard navigation [10]. Since space travel has a very small tolerance for error, traditional methods were inadequate. The Kalman filter [16], introduced by Rudolf E. Kalman in 1960, was identified by Stanley F. Schmidt at NASA Ames Research Center as a potential solution to the Apollo program's navigation challenges. The Kalman filter could merge noisy sensor data with predictive models based on probability distributions to provide real-time updates of the spacecraft's position and velocity. To ensure numerical stability on the limited 15-bit fixed-point Apollo Guidance Computer, James E. Potter introduced a square-root filtering method, enabling reliable computation of the Kalman filter's equations during the mission [10]. The navigation system of the Apollo program is a notable example of where probabilistic algorithms made a crucial difference. Since then, probabilistic, also called Bayesian modeling has been used in many fields of engineering and science, including learning systems [3, 13, 18, 27], inertial navigation [11], motion capture [12, 19], target tracking [1], autonomous navigation [35], and brain imaging [9].

Probabilistic modeling uses probability distributions to represent uncertainty in data and in model predictions. This allows for merging prior information and measurements, enabling, e.g. recursive or online estimation. The benefits of online estimation include the following. Firstly, the ability to process data "on the go" without relying on future data enables real-time decision-making. Secondly, retraining a model from scratch when new data becomes available can be avoided because the information is retained in prior estimates, which can be updated with new measurements. Thirdly, probabilistic modeling offers uncertainty quantification, which makes algorithms more robust and reliable and is useful, e.g. for safety-critical applications.

Figure 1.1 showcases an exemplary application of probabilistic modeling, where both the recursive property as well as uncertainty estimation are leveraged: building spatial maps from measurements recursively with Gaussian process (GP) regression [27]. Figure 1.1(a) shows simulated synthetic measurements of magnetic field anomalies computed at 2D coordinates along a spiral-shaped path, where the color corresponds to the magnitude of the anomalies. A map of the magnetic field anomalies is built from the measurements by training a GP model that maps the 2D coordinates to the measurements and computing GP predictions over the whole 2D space, as visualized in Figure 1.1(b). The map is recursively updated by treating the map from the previous time step as a prior and combining it with a new measurement. GP regression computes a predictive distribution for new inputs in terms of a mean and covariance. The GP map, therefore, consists of a magnetic field mean estimate together with uncertainty information. Figure 1.1(b) at $t = 40$ and $t = 150$ shows that close to the measurements, the predictions are more certain, and the certainty decreases with the distance to the measurement locations. Over time, when measurements are collected over a larger area (see Figure 1.1 at $t=600$), the map becomes more certain overall.

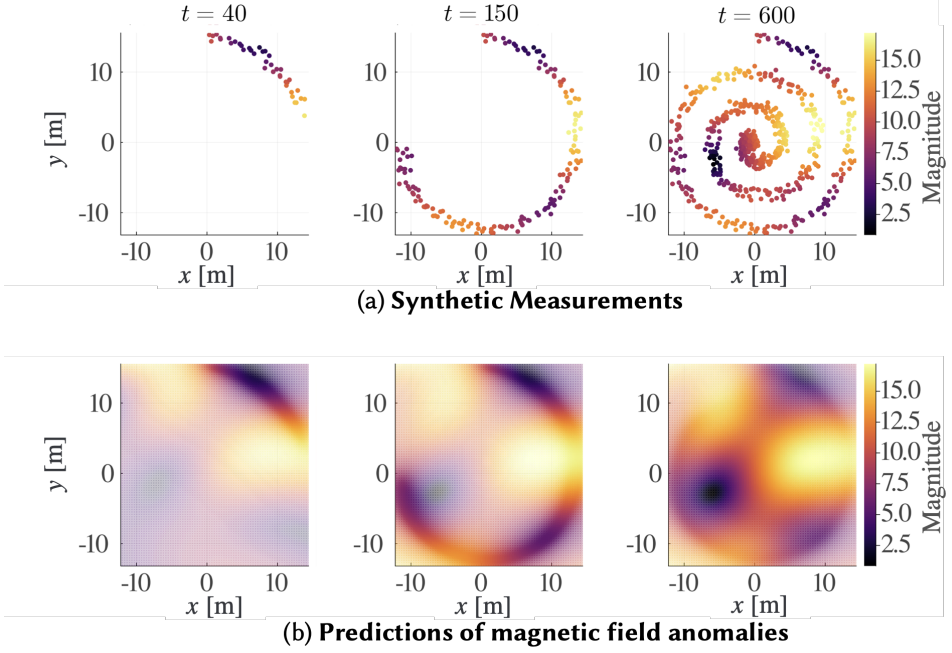


Figure 1.1: Recursive map prediction from simulated synthetic measurements of magnetic field anomalies. (a) shows the magnitude of 3D magnetic field anomaly measurements in three instances in time, where the number of available measurements grows over time. The colors correspond to the intensity of the magnetic field anomalies. (b) shows a map of the magnetic field anomalies computed by interpolating the 3D magnetic field measurements over the spatial area of interest with GP regression. The magnitude of the three magnetic field components, computed as GP predictions, is shown. The predictive mean is visualized in terms of the map's magnetic field magnitude. The predictive variance of the predictive distribution is visualized in terms of map transparency, where higher transparency means higher map uncertainty.

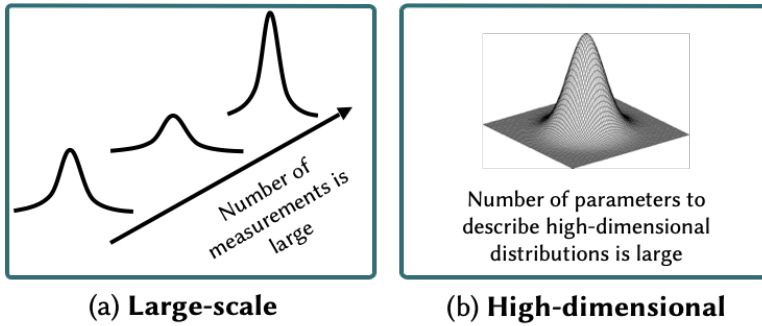


Figure 1.2: (a) Depiction of a large-scale problem where a large number of measurements N is collected and the involved distributions are estimated over time. (b) Depiction of a high-dimensional problem where distributions have exponentially many parameters caused by the dimensionality D of the problem.

Figure 1.1 is an example where the input dimension D to the GP is 2, i.e. x - and y -spatial coordinates, and the number of measurements is $N=600$. However, there are many related problems where N or D are large, referred to as large-scale and high-dimensional problems, respectively. An example of a large-scale problem is when a sensor collects data at a high sampling rate and the distributions need to be updated accordingly (see Figure 1.2 (a)). An example of a high-dimensional problem is to consider a time series where the input is a combination of multiple previous inputs and/or outputs, which requires the number of parameters to describe the distributions of interest to be large (see Figure 1.2(b)).

This thesis studies probabilistic algorithms where N and/or D are large, resulting in computational challenges. To illustrate them, consider a Bayesian estimation problem, based on [27, 33], that the problems presented in this thesis can be cast into.

We are interested in modeling a nonlinear function $f(\mathbf{x})$ by estimating parameters \mathbf{w} from a parametric model given by

$$\begin{aligned} y &= f(\mathbf{x}) + \epsilon \\ &= \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_y^2), \quad \mathbf{w} \sim \mathcal{N}(\mathbf{w}_0, \mathbf{P}_0), \end{aligned} \quad (1.1)$$

where y is a measurement, \mathbf{x} is an input to the nonlinear function $f(\mathbf{x})$, $\boldsymbol{\phi}(\mathbf{x})$ is a feature map that maps an input into a feature space of size M , and $\mathcal{N}(\cdot, \cdot)$ denotes a Gaussian distribution with a specified mean and variance. The measurement noise ϵ is zero-mean and has a noise variance σ_y^2 , and $\mathbf{w} \sim \mathcal{N}(\mathbf{w}_0, \mathbf{P}_0)$ denotes the prior distribution on the parameters with prior mean \mathbf{w}_0 and prior covariance \mathbf{P}_0 .

Given (1.1) with the Gaussian assumption of the noise and prior weights, and a set of N data points $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and $\mathbf{y} = y_1, y_2, \dots, y_N$, the posterior distribution for the weights can be computed with

$$p(\mathbf{w} | \mathbf{y}) \sim \mathcal{N}(\mathbf{w}_0 + \underbrace{\mathbf{P}_0 \boldsymbol{\Phi}^\top (\boldsymbol{\Phi} \mathbf{P}_0 \boldsymbol{\Phi}^\top + \sigma_y^2 \mathbf{I})^{-1}}_{N \times N} (\mathbf{y} - \boldsymbol{\Phi} \mathbf{w}_0), \underbrace{\mathbf{P}_0 - \mathbf{P}_0 \boldsymbol{\Phi}^\top (\boldsymbol{\Phi} \mathbf{P}_0 \boldsymbol{\Phi}^\top + \sigma_y^2 \mathbf{I})^{-1} \boldsymbol{\Phi} \mathbf{P}_0}_{N \times N}), \quad (1.2)$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{N \times M}$ contains row-wise $\boldsymbol{\phi}(\mathbf{x}_1), \boldsymbol{\phi}(\mathbf{x}_2), \dots, \boldsymbol{\phi}(\mathbf{x}_N)$. The main bottleneck of computing (1.2) is storing and inverting the $N \times N$ matrix, which has a storage and computational complexity of $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$, respectively. An alternative formulation of (1.2) after applying a matrix inversion lemma is given by

$$p(\mathbf{w} | \mathbf{y}) \sim \mathcal{N}(\mathbf{w}_0 + \underbrace{(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \sigma_y^2 \mathbf{P}_0^{-1})^{-1} \boldsymbol{\Phi}^\top}_{M \times M} (\mathbf{y} - \boldsymbol{\Phi} \mathbf{w}_0), \underbrace{\sigma_y^2 (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \sigma_y^2 \mathbf{P}_0^{-1})^{-1}}_{M \times M}), \quad (1.3)$$

where it is necessary to store and invert a matrix of size $M \times M$, such that the required storage and computational complexity are $\mathcal{O}(M^2)$ and $\mathcal{O}(M^3)$, respectively. It is possible to compute (1.3) recursively [28] at a computational cost of $\mathcal{O}(M^2)$ per recursion.

Depending on the size of M or N , either (1.2), (1.3) or both can become intractable: On the one hand, computing (1.2) can become a computational bottleneck when the number of measurements N is large. On the other hand, computing (1.3) can become unfeasible when the number of features M is large, which is typically the case when D is large. Problems

where both N and D are large are the most challenging because both formulations to solve (1.1) can become a computational bottleneck.

This thesis focuses on tensor networks as a tool to develop scalable probabilistic algorithms for large-scale and high-dimensional problems, alleviating the computational challenges mentioned in this section.

1.2 TENSOR NETWORKS: A TOOL FOR SCALABILITY

Tensors are a generalization of scalars, vectors, and matrices to higher dimensions. While, for example, a matrix is a two-dimensional array with two indices (rows and columns), tensors are represented as multi-dimensional arrays that encode data across multiple indices. When decomposing high-dimensional tensors into a network of smaller, interconnected tensors, a tensor network (TN) (interchangeably used with tensor decomposition) is formed. This can be seen as the generalization of matrix decompositions to higher dimensions. While, for example, the singular value decomposition (SVD) identifies the most significant features of a matrix, a TN captures the essential information of a multi-dimensional array. Thus, tensor decompositions can be interpreted as an SVD extended to higher dimensions. TNs provide a framework to solve large-scale and high-dimensional problems (see Figure 1.2) in an efficient way. Thus, TNs can be used to solve Bayesian estimation problems as given by (1.1) for both large N and D .

The following sections offer a high-level and accessible introduction to TN, showcasing examples that highlight why TNs are a tool for scalability. More technical and mathematical details can be found in Sections 2.2 and 3.3, and in references therein, e.g. [17] and [5]. First, the generalization of low-rank matrix decomposition to low-rank TNs is described. Then, it is shown how efficient storage and computations are enabled by low-rank TNs.

1.2.1 FROM LOW-RANK MATRIX DECOMPOSITIONS TO LOW-RANK TNs

The efficient representation of TNs enables efficient storage and computation and is thus the key characteristic that makes TNs a tool for scalability. There is, however, a key assumption that needs to be fulfilled in order to enable the scalability of TNs: the low-rank assumption [5, 30]. Before explaining what low-rank TNs are, consider again a matrix case. A truncated SVD, depicted in Figure 1.3(a), is a low-rank approximation of a matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ that considers only the dominant R singular values, where R is the rank of the matrix decomposition. The approximation of \mathbf{A} is then computed with $\mathbf{U}\mathbf{S}\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{I \times R}$, $\mathbf{V} \in \mathbb{R}^{J \times R}$ are the orthonormal matrices and $\mathbf{S} \in \mathbb{R}^{R \times R}$ contains the dominant singular values. Generalizing the matrix case, a low-rank approximation of a D th order tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_D}$ can be represented by a low-rank tensor decomposition, which has multiple so-called TN ranks. Figure 1.3(b) shows an exemplary TN architecture called tensor train (TT) decomposition [24] for $D = 3$ consisting of TN components $\mathcal{T}^{(1)} \in \mathbb{R}^{R_1 \times I_1 \times R_2}$, $\mathcal{T}^{(2)} \in \mathbb{R}^{R_2 \times I_2 \times R_3}$, $\mathcal{T}^{(3)} \in \mathbb{R}^{R_3 \times I_3 \times R_4}$ with TN ranks R_1, R_2, \dots, R_{D+1} .

A graphical notation for tensors and TNs, called Penrose graphical notation [4, 26], allows visualizing how the ranks connect the components and how \mathbf{A} and \mathcal{T} can be reconstructed from their respective matrix and tensor decomposition. As shown in Figure 1.4,

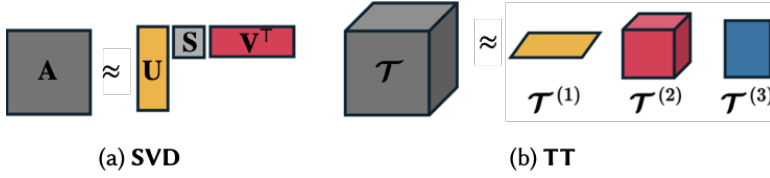


Figure 1.3: (a) Depiction of a low-rank matrix decomposition of matrix A (gray rectangle), in terms of a truncated singular value decomposition USV^T , where the yellow and red rectangles represent the orthonormal matrices U, V and the light grey square represents the matrix S containing the singular values. (b) Depiction of a low-rank tensor decomposition of 3rd order tensor \mathcal{T} , in terms of a TT with TT-cores $\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \mathcal{T}^{(3)}$.

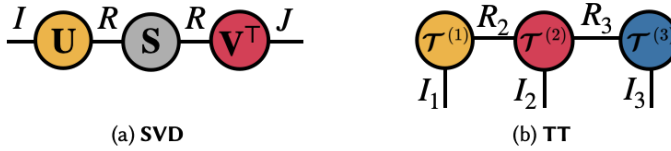


Figure 1.4: Visual depiction in terms of the Penrose graphical notation of a truncated SVD and TT of a 3rd order tensor. (a) The matrices involved in the SVD are depicted as nodes with two edges, corresponding to the row and column indices. The connected edges represent the summation over the shared index, as defined in (1.4). The free indices I and J correspond to the size of A . (b) The TT components are third-order tensors and are thus represented as a node with three edges. Note that the first and last components have only two edges because, by definition, $R_1 = R_{D+1} = 1$ and edges equal to 1 are usually not depicted. The connected edges correspond to the summation over multiple indices as defined in (1.5), and the free edges I_1, I_2, I_3 correspond to the size of \mathcal{T} .

a matrix, e.g. U , is depicted as a node with two edges, representing the row and column index, while a tensor has as many edges as its order, e.g. third order tensor $\mathcal{T}^{(2)}$ has three edges. Connected edges represent a summation, also called contraction, over indices, and free edges correspond to the size of the matrix or tensor that is represented by the matrix or tensor decomposition.

Figure 1.4(a) corresponds to the matrix-matrix-multiplications computed by summing over the shared index r of size R such that the (i, j) th entry of A is given by

$$A(i, j) = \sum_{r=1}^R U(i, r) S(r, r) V(r, j). \quad (1.4)$$

In a similar way, Figure 1.4(b) corresponds to summing over all TN ranks such that each entry of \mathcal{T} can be computed with

$$\mathcal{T}(i_1, \dots, i_d, \dots, i_D) = \sum_{r_1=1}^{R_1} \dots \sum_{r_d=1}^{R_d} \dots \sum_{r_{D+1}=1}^{R_{D+1}} \mathcal{T}^{(1)}(r_1, i_1, r_2) \dots \mathcal{T}^{(d)}(r_d, i_d, r_{d+1}) \dots \mathcal{T}^{(D)}(r_D, i_D, r_{D+1}), \quad (1.5)$$

where by definition $R_1 = R_{D+1} = 1$, such that (1.5) sums only over the TN ranks that connect the TN components.

The TN ranks, which can be considered hyperparameters, need to be computed or chosen. For matrices, the Eckart-Young theorem states that the best low-rank approximation of a matrix, in terms of minimizing the Frobenius norm or spectral norm, is given by its truncated SVD [7]. For TNs such a theorem cannot be formulated because in general, they do not have a single best rank- R approximation [6]. The choice of ranks is an important step, as it determines the accuracy of the representation. Furthermore, it needs to be considered that ranks can increase, for example, when TTs are summed together or multiplied with each other. To counteract this growth and maintain the efficiency of the TT representation, a tensor operation called TT-rounding can be applied [24]. This SVD-based algorithm sweeps through the TT components and reduces the ranks according to a specified target rank.

The impact of the ranks on storage requirements and computational complexity of a TN-based algorithm will be further discussed in the next section.

1.2.2 EFFICIENT STORAGE AND COMPUTATIONS WITH TNs

When dealing with large-scale and high-dimensional data or models, there are usually two main bottlenecks: the required storage is so large that the system exhausts its available memory, or the computations take an excessive amount of time. Even when computations are feasible, they require long computation times and powerful hardware, which in turn impact the algorithm's energy consumption and carbon footprint [29, 32]. This section explores how TNs tackle these issues by reducing storage and computational complexity, thus also leading to improved energy efficiency and a smaller carbon footprint. In fact, TNs are considered a tool for so-called Green AI [21], which is an umbrella term introduced by [29] for AI algorithms that view accuracy and efficiency as equally important, and spotlights the importance of sustainability in AI research.

The key characteristic of low-rank TNs lies in their ability to store multiple smaller tensors rather than relying on a single, large tensor. This decomposed format allows for the computation of multiple smaller, more manageable operations by distributing the computations across the TN components, see Figure 1.5.

To illustrate how TNs reduce storage and computational requirements, in the following, we discuss two examples in which we quantify storage and computational complexities with and without a TN representation. We use a TT representation, as defined in (1.5).

Suppose that we have a matrix \mathbf{A} of size $10^D \times 10^D$, where $D = 5$, as depicted in Figure 1.6(I)(a). Since the size of \mathbf{A} grows exponentially with D , it is called an exponentially large matrix with a storage complexity of $\mathcal{O}(10^D \cdot 10^D)$. Now consider a TT matrix (TTm) [23] representation of \mathbf{A} , denoted by \mathcal{A} , consisting of 5 interconnected TTm-cores, as depicted in Figure 1.6(II)(a). Each TTm-core is a 4th order tensor, $\mathcal{A}^{(d)} \in \mathbb{R}^{R_d \times 10 \times 10 \times R_{d+1}}$, $d = 1, \dots, 5$ and for this example it is assumed that all ranks are equal to R . The number of TTm-cores is the dimensionality D of the representation. The storage complexity of the TTm is $\mathcal{O}(D \cdot 10 \cdot 10 \cdot R^2)$, so linear in D as opposed to the storage complexity of \mathbf{A} , which is exponential in D . The accuracy of the representation depends on the ranks R , and, as

discussed in Section 1.2.1, efficient storage relies on the low-rank assumption.

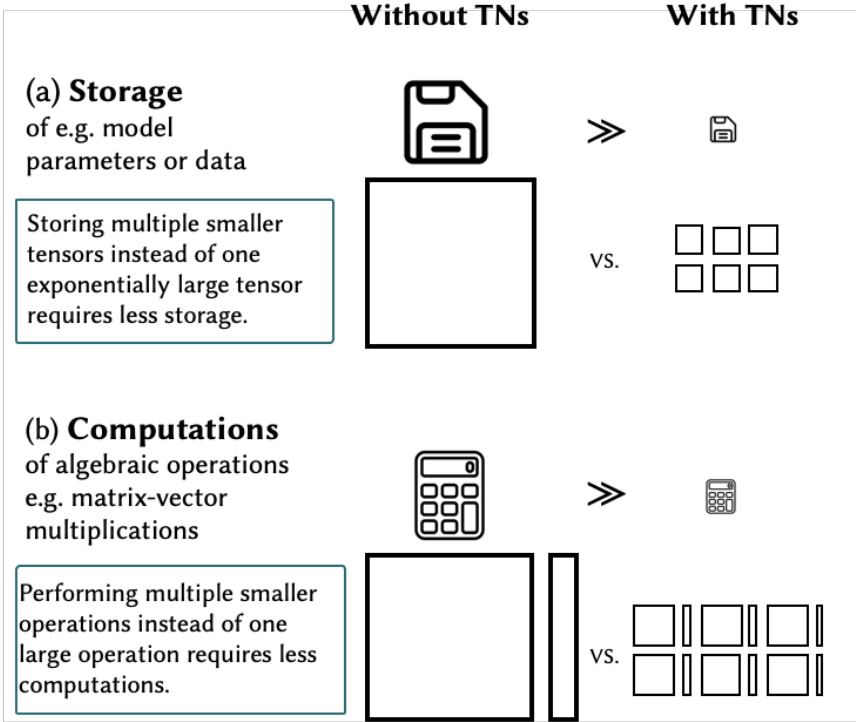


Figure 1.5: Storage and computations with and without TNs. The key characteristic that enables the storage of and computation with exponentially large tensors lies in storing multiple smaller tensors and performing multiple smaller operations, respectively.

Now consider a matrix-vector multiplication between the $10^D \times 10^D$ matrix \mathbf{A} with a $10^D \times 1$ vector \mathbf{x} , as depicted in Figure 1.6(I)(b). The computational complexity of this operation is $\mathcal{O}(10^D \cdot 10^D)$, so exponential in D . In TT format, the matrix-vector multiplication can be represented by the TN in Figure 1.6(II)(b), where \mathbf{x} is represented by a TT with TT-cores $\mathcal{X}^{(1)}, \mathcal{X}^{(2)}, \dots, \mathcal{X}^{(5)}$. Instead of summing over one shared index of size 10^D like in the case without TNs, shared indices and ranks are summed over in a sequential core-by-core fashion. The computational complexity is given by $\mathcal{O}(D \cdot 10 \cdot 10 \cdot R^4)$, which is linear in D . This allows for efficient computations under the assumption that the TT representations are low-rank. As mentioned in Section 1.2.1, after multiplying TTs with each, the ranks can increase, such that TT-rounding needs to be applied to keep the representation efficient.

In summary, the two examples above illustrate how TNs achieve what can be described as logarithmic compression, enabling both the storage of and computation with an exponentially large matrix using only linear complexity in D . This stands in contrast to the so-called


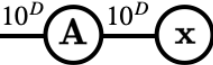
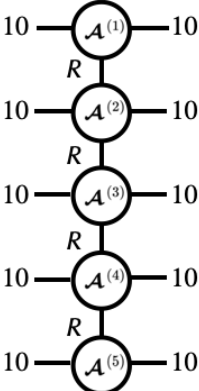
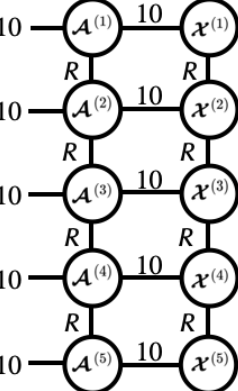
<i>Example for $D = 5$:</i>	(a) Storage complexity of a matrix of size $10^D \times 10^D$	(b) Computational complexity of a matrix-vector multiplications
(I) Without TNs	$\mathcal{O}(10^D \cdot 10^D)$ 	$\mathcal{O}(10^D \cdot 10^D)$ 
(II) With TNs	$\mathcal{O}(D \cdot 10 \cdot 10 \cdot R^2)$ 	$\mathcal{O}(D \cdot 10 \cdot 10 \cdot R^4)$ 

Figure 1.6: (a) Storage complexity of a matrix and (b) computational complexity of a matrix-vector multiplication (I) with and (II) without TNs for examples with $D = 5$.

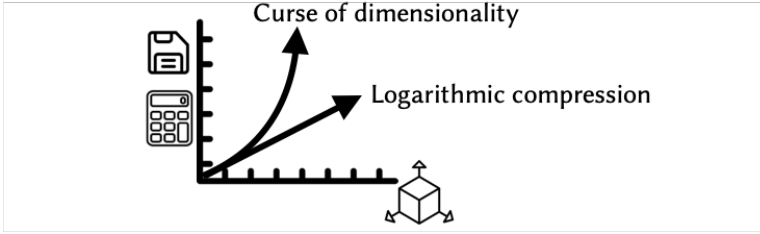


Figure 1.7: Depiction of the curse of dimensionality, where the storage (floppy disk symbol) and computational complexity (calculator symbol) grow exponentially with the dimensionality of the problem D (cube with arrows symbol) versus logarithmic compression where the complexities grow linear with D .

curse of dimensionality [25], which causes an exponential demand in computational and storage requirements as D grows (see Figure 1.7).

1.3 CONTRIBUTIONS OF THIS THESIS

The previous two sections have highlighted challenges within the research field of probabilistic modeling, particularly concerning scalability for large-scale and high-dimensional Bayesian estimation problems. In addition, TNs have been introduced as a tool for scalability and, thus, a promising approach to address these issues. All contributions involve large-scale and/or high-dimensional probabilistic estimation problems that are tackled with TNs, and the overall thesis goal is summarized as follows.

Thesis goal: Develop scalable methods for probabilistic modeling in large-scale and/or high-dimensional settings using efficient storage and computations enabled by TNs.

To achieve the thesis goal, the different chapters of this thesis present contributions in terms of papers that contribute to the overall goal. For each contribution, a context is provided, followed by highlighting the novelty of the contribution. Note that the notation in this section can be different from the notation used in the chapters for the purpose of connecting equations between contributions.

We start by applying a Bayesian framework to a well-known TN-based algorithm, the alternating least squares or alternating linear scheme (ALS), e.g. [14]. In a TN context, the ALS is an established algorithm for computing a low-rank approximation of a tensor $\mathcal{Y} \in \mathbb{R}^{I \times I \times \dots \times I}$, solving a problem given by

$$\min_{\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(d)}, \dots, \mathcal{G}^{(D)}} \|\mathbf{y} - \hat{\mathbf{y}}\|, \text{ s.t. } \hat{\mathbf{y}} \text{ being a low-rank TN} \quad (1.6)$$

with TN components $\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(d)}, \dots, \mathcal{G}^{(D)}$,

where $\mathbf{y} \in \mathbb{R}^{I^D}$ and $\hat{\mathbf{y}} \in \mathbb{R}^{I^D}$ are vectorized \mathcal{Y} and $\hat{\mathcal{Y}}$, respectively. Each TN component $\mathcal{G}^{(d)}$ is first initialized and then updated sequentially by solving a least squares problem given

by

$$\min_{\mathbf{g}^{(d)}} \|\mathbf{y} - \mathbf{U}_{\setminus d} \mathbf{g}^{(d)}\|_2^2, \quad (1.7)$$

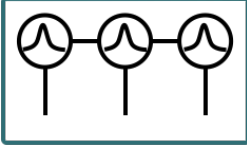
where $\mathbf{U}_{\setminus d}$ is an orthogonal matrix computed from all TN components except the d th (denoted by the subscript $\setminus d$), $\mathbf{g}^{(d)}$ is the d th vectorized TN component.

Contrary to the conventional ALS, where all subproblems (1.7) are solved without regularization, we solve a Bayesian inference problem for each update. We put a prior on each TN component $\mathbf{g}^{(d)} \sim \mathcal{N}(\mathbf{m}_0^{(d)}, \mathbf{P}_0^{(d)})$ and compute a posterior distribution for each TN component $p(\mathbf{g}^{(d)} | \{\mathbf{g}^{(i)}\}_{i \neq d}, \mathbf{y})$. Solving a Bayesian inference problem leads to turning (1.7) into a weighted least squares optimization problem given by

$$\min_{\mathbf{g}^{(d)}} \|\mathbf{y} - \mathbf{U}_{\setminus d} \mathbf{g}^{(d)}\|_{\sigma_y^{-2} \mathbf{I}}^2 + \|\mathbf{g}^{(d)}\|_{(\mathbf{P}_0^{(d)})^{-1}}^2, \quad (1.8)$$

where σ_y^2 is the variance of the measurement noise. Thus, the first contribution of this thesis can be summarized as follows.

Contribution 1



Development of a Bayesian ALS algorithm: We approach the low-rank tensor approximation problem from a Bayesian perspective by treating all TN components as Gaussian random variables with a prior mean and covariance, and solving multiple sequential Bayesian inference problems to compute the components of a TN in terms of a posterior distribution.

A central assumption in the first contribution is that the TN components are treated as Gaussian random variables such that also their posterior distributions $p(\mathbf{g}^{(d)} | \{\mathbf{g}^{(i)}\}_{i \neq d}, \mathbf{y})$ are Gaussians. We are, however, often not interested in the posterior distributions of the TN components themselves, but in the low-rank tensor estimate $\hat{\mathbf{Y}}$ and its uncertainty. A limitation is that the distribution for the tensor approximation is not Gaussian because it is reconstructed in a nonlinear way from the TN components.

One way to solve this issue is by treating only one of the TN components as stochastic and all others as deterministic when reconstructing the tensor approximation from its TN components. We explore this in [20, 22] in terms of an ALS in a Bayesian framework for the computation of the posterior model weights in (1.1). When we applied this approach in an online scenario, however, primary experiments showed that the uncertainty on the weights is highly underestimated and causes the covariance matrices in the online algorithm to lose positive definiteness.

The findings of the first contribution and [20, 22] inspired us to adopt a different perspective. Instead of treating each TN component as a probability distribution, in the second contribution, we use TNs to approximate the mean and square root covariance factors of probability distributions. We consider the recursive version of the Bayesian estimation problem (1.1), corresponding to solving the weighted least squares problem given by

$$\min_{\mathbf{w}_t} \|\mathbf{y}_t - \boldsymbol{\phi}_t^\top \mathbf{w}_t\|_{\sigma_y^{-2} \mathbf{I}}^2 + \|\mathbf{w}_t\|_{\mathbf{P}_{t-1}^{-1}}^2, \quad (1.9)$$

where y_t , ϕ_t and \mathbf{w}_t are the measurement, features, and weights at time t , and \mathbf{P}_{t-1} is the covariance matrix from the previous time step. Solving (1.9) leads to the standard equation for the measurement update of the Kalman filter [28]. As mentioned in Section 1.1, updating the weight estimate and its covariance for a high-dimensional problem results in a computational complexity that is exponential in D . In TN format, the TN Kalman filter [2] achieves logarithmic compression of computational complexity, but introduces the possibility of loss of positive (semi-) definiteness due to a tensor-specific operation, called rounding. In the second contribution, we propose a square root Kalman filter formulation in TN format that guarantees the covariance matrices to stay positive (semi-) definiteness. We use the ALS as in (1.6) to compute an estimate for the weights \mathbf{w} , as well as a square root covariance factor. In this context, the tensor that is approximated corresponds to the mean update in the Kalman filter measurement update equations. We approximate $\hat{\mathbf{w}}_t$ in terms of TT-components $\mathcal{W}_t^{(1)}, \mathcal{W}_t^{(2)}, \dots, \mathcal{W}_t^{(D)}$ by solving the optimization problem given by

$$\min_{\mathcal{W}_t^{(1)}, \mathcal{W}_t^{(2)}, \dots, \mathcal{W}_t^{(D)}} \|\mathbf{w}_t - \hat{\mathbf{w}}_t\|, \quad (1.10)$$

with $\hat{\mathbf{w}}_t = \mathbf{w}_{t-1} + \mathbf{K}_t(y_t - \phi_t^\top \hat{\mathbf{w}}_{t-1})$ s.t. being a low-rank TT,

where \mathbf{K}_t is the Kalman gain at time t . We adopt a similar strategy as in (1.10) to compute the square root covariance factor update, which allows us to recursively update the covariance matrix without running into positive definite issues like existing TN-based algorithms. The second contribution is, therefore, summarized as follows.

Contribution 2



Solving the issue of loss of positive definiteness in tensor network Kalman filtering: We propose a square root formulation of the Kalman filter in TN format by using the ALS to compute the mean and square root covariance in TN format and apply it to high-dimensional online Gaussian process regression.

The final contribution of this thesis is an application in the field of sensor fusion that tackles a large-scale magnetic field mapping problem, similar to Figure 1.1. As mentioned in Section 1.1, building magnetic field maps with GP regression provides a mean estimate of the map together with its uncertainty. Similar to [31, 34], we model the scalar potential of a magnetic field ϕ at a 3D position \mathbf{p} with a GP given by

$$\phi(\mathbf{p}) \sim \mathcal{GP}(0, \kappa(\mathbf{p}, \mathbf{p}')), \quad (1.11)$$

where the GP's mean function is set to zero and the kernel function $\kappa(\mathbf{p}, \mathbf{p}')$ is the squared exponential kernel to model only the magnetic field anomalies. The resulting derivative model is then given by

$$\mathbf{y} = -\nabla \phi(\mathbf{p}) + \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_y^2 \mathbf{I}_3), \quad (1.12)$$

where \mathbf{p} is a 3D input location, \mathbf{y} is a 3D magnetic field measurement vector, ϵ is vector-valued noise with variance σ_y^2 .

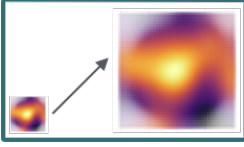
Given (1.11) and (1.12), the predictive distribution for \mathbf{f}_* , i.e. the magnetic field vector at a new location \mathbf{p}_* , is given by the standard GP prediction equations [27, Chapter 3] adapted to a derivative measurement model

$$\begin{aligned} \mathbb{E}[\mathbf{f}_*] &= \partial^2(\mathbf{K}_{*,\mathbf{f}}) \left(\partial^2(\mathbf{K}_{\mathbf{f},\mathbf{f}}) + \sigma_y^2 \mathbf{I}_{3N} \right)^{-1} \text{vec}(\mathbf{Y}^\top), \\ \mathbb{V}[\mathbf{f}_*] &= \partial^2(\mathbf{K}_{*,*}) - \partial^2(\mathbf{K}_{*,\mathbf{f}}) \left(\partial^2(\mathbf{K}_{\mathbf{f},\mathbf{f}}) + \sigma_y^2 \mathbf{I}_{3N} \right)^{-1} \partial^2(\mathbf{K}_{\mathbf{f},*}), \end{aligned} \quad (1.13)$$

where $\mathbf{Y} = [\mathbf{y}_1^\top \mathbf{y}_2^\top \dots \mathbf{y}_N^\top] \in \mathbb{R}^{N \times 3}$, and $\partial^2(\mathbf{K}_{\mathbf{f},\mathbf{f}})$, $\partial^2(\mathbf{K}_{*,\mathbf{f}})$ and $\partial^2(\mathbf{K}_{*,*})$ denote kernel matrices computed as exemplified as follows. A 3×3 block in, e.g., $\partial^2(\mathbf{K}_{*,\mathbf{f}})$ is computed with $\nabla_{\mathbf{p}_*} \kappa(\mathbf{p}_*, \mathbf{p}) \nabla_{\mathbf{p}}^\top$, where ∇ denotes the gradient that is taken w.r.t. to the position vector specified in the subscript.

In terms of computational complexities, (1.13) requires $\mathcal{O}(3N^3)$. To compute (1.13) in a more efficient way that scales linearly in N , we use a combination of a kernel approximation technique called structured kernel interpolation with derivatives [8], and Kronecker algebra to speed up matrix-vector multiplications, as described in Chapter 4, Section 4.2. Note that Kronecker algebra is not directly a TN method, but it is closely related. In fact, a TTm with all TT-ranks equal to 1 corresponds to a Kronecker product between matrices [15]. The third contribution is summarized as follows.

Contribution 3



Application of structured kernel interpolation to magnetic field mapping. We compute scalable magnetic field maps in 3D using magnetic field measurements as training data. We incorporate structured kernel interpolation with derivatives into the scalar potential model for magnetic field modeling, and use Krylov subspace methods to make GP predictions with a computational complexity that is linear in the number of data points.

REFERENCES

- [1] Yaakov Bar-Shalom and Xiao-Rong Li. *Multitarget-multisensor tracking: principles and techniques*, volume 19. YBS publishing Storrs, CT, 1995.
- [2] Kim Batselier, Zhongming Chen, and Ngai Wong. A tensor network Kalman filter with an application in recursive MIMO Volterra system identification. *Automatica*, 84:17–25, 2017.
- [3] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.
- [4] Jacob C Bridgeman and Christopher T Chubb. Hand-waving and interpretive dance: an introductory course on tensor networks. *Journal of Physics A: Mathematical and theoretical*, 50(22):223001, 2017.
- [5] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, Danilo P Mandic, et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.
- [6] Vin De Silva and Lek-Heng Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008.
- [7] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
- [8] David Eriksson, Kun Dong, Eric Lee, David Bindel, and Andrew G Wilson. Scaling Gaussian process regression with derivatives. *Advances in neural information processing systems*, 31, 2018.
- [9] Karl J Friston, Lee Harrison, and Will Penny. Dynamic causal modelling. *Neuroimage*, 19(4):1273–1302, 2003.
- [10] Mohinder S Grewal and Angus P Andrews. Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems Magazine*, 30(3):69–78, 2010.
- [11] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.
- [12] Robert Harle. A survey of indoor inertial positioning systems for pedestrians. *IEEE Communications Surveys & Tutorials*, 15(3):1281–1293, 2013.
- [13] Simon Haykin. *Kalman filtering and neural networks*. John Wiley & Sons, 2004.
- [14] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012.

- [15] Pavel Izmailov, Alexander Novikov, and Dmitry Kropotov. Scalable Gaussian processes with billions of inducing inputs via tensor train decomposition. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 726–735. PMLR, 2018.
- [16] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.
- [17] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.
- [18] Daphne Koller. Probabilistic graphical models: Principles and techniques, 2009.
- [19] Henk J Luinge and Peter H Veltink. Measuring orientation of human body segments using miniature gyroscopes and accelerometers. *Medical and Biological Engineering and Computing*, 43:273–282, 2005.
- [20] Eva Memmel, Clara Menzen, and Kim Batselier. Bayesian framework for a MIMO Volterra tensor network. *IFAC-PapersOnLine*, 56(2):7294–7299, 2023.
- [21] Eva Memmel, Clara Menzen, Jetze Schuurmans, Frederiek Wesel, and Kim Batselier. Position: Tensor networks are a valuable asset for Green AI. In *Proceedings of the International Conference on Machine Learning*, pages 35340–35353. PMLR, 2024.
- [22] Clara Menzen, Eva Memmel, Kim Batselier, and Manon Kok. Projecting basis functions with tensor networks for Gaussian process regression. *IFAC-PapersOnLine*, 56(2):7288–7293, 2023.
- [23] Ivan V Oseledets. Approximation of 2D x 2D matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145, 2010.
- [24] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.
- [25] Ivan V Oseledets and Eugene E Tyrtyshnikov. Breaking the curse of dimensionality, or how to use SVD in many dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759, 2009.
- [26] Roger Penrose. Applications of negative dimensional tensors. *Combinatorial mathematics and its applications*, 1:221–244, 1971.
- [27] Carl E Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [28] Simo Särkkä and Lennart Svensson. *Bayesian Filtering and Smoothing*, volume 17. Cambridge University Press, 2023.
- [29] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020.

- [30] Tianyi Shi and Alex Townsend. On the compressibility of tensors. *SIAM Journal on Matrix Analysis and Applications*, 42(1):275–298, 2021.
- [31] Arno Solin, Manon Kok, Niklas Wahlström, Thomas B Schön, and Simo Särkkä. Modeling and interpolation of the ambient magnetic field by Gaussian processes. *IEEE Transactions on robotics*, 34(4):1112–1127, 2018.
- [32] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243*, 2019.
- [33] Michel Verhaegen and Vincent Verdult. *Filtering and system identification: a least squares approach*. Cambridge University Press, 2007.
- [34] Niklas Wahlström, Manon Kok, Thomas B Schön, and Fredrik Gustafsson. Modeling magnetic fields using Gaussian processes. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3522–3526. IEEE, 2013.
- [35] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, and Joseph Walsh. Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors*, 21(6):2140, 2021.

2

ALTERNATING LINEAR SCHEME IN A BAYESIAN FRAMEWORK FOR LOW-RANK TENSOR APPROXIMATION

Multiway data often naturally occurs in a tensorial format which can be approximately represented by a low-rank tensor decomposition. This is useful because complexity can be significantly reduced and the treatment of large-scale data sets can be facilitated. In this contribution to the thesis, we find a low-rank representation for a given tensor by solving a Bayesian inference problem. This is achieved by dividing the overall inference problem into subproblems where we sequentially infer the posterior distribution of one tensor decomposition component at a time. This leads to a probabilistic interpretation of the well-known iterative algorithm alternating linear scheme (ALS). In this way, the consideration of measurement noise is enabled, as well as the incorporation of application-specific prior knowledge and the uncertainty quantification of the low-rank tensor estimate. To compute the low-rank tensor estimate from the posterior distributions of the tensor decomposition components, we present an algorithm that performs the unscented transform in tensor train format.

2.1 INTRODUCTION

Low-rank approximations of multidimensional arrays, also called tensors, have become a central tool in solving large-scale problems. The numerous applications include machine learning (e.g. tensor completion [14, 37, 44], kernel methods [5, 36] and deep learning [10, 26]), signal processing [3, 35], probabilistic modeling [21, 41], non-linear system identification [1, 13] and solving linear systems [12, 29]. An extensive overview of applications can be found, e.g., in [9].

In many applications, an observed tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ can be represented with a low-rank approximation \mathcal{Y}_{lr} , without losing the most meaningful information [8]. In the presence of uncorrelated noise \mathcal{E} , however, \mathcal{Y} loses the low-rank structure. The observed tensor can be modeled as

$$\mathcal{Y} = \mathcal{Y}_{\text{lr}}(\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_N) + \mathcal{E}, \quad \text{vec}(\mathcal{E}) \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}), \quad (2.1)$$

where \mathcal{Y}_{lr} is a low-rank tensor decomposition (TD), which is a function of TD components $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_N$. Examples of TDs are the CANDECOMP/PARAFAC (CP) decomposition [4, 15], the Tucker decomposition [38] and the tensor train (TT) decomposition [28]. We model the vectorized noise $\text{vec}(\mathcal{E})$ as Gaussian, where $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ denotes a zero mean multivariate normal distribution with covariance matrix $\sigma^2 \mathbf{I}$. The identity matrix is denoted by \mathbf{I} which in Equation (2.1) is of size $I_1 I_2 \dots I_N \times I_1 I_2 \dots I_N$.

In this work, we solve a Bayesian inference problem to seek a low-rank TD \mathcal{Y}_{lr} , given an observed noisy tensor \mathcal{Y} . In general, TDs solve an optimization problem of the form

$$\min_{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_N} \|\mathcal{Y} - \mathcal{Y}_{\text{lr}}(\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_N)\|, \quad (2.2)$$

There exist multiple methods to find a decomposition for a given tensor. The approach that we are looking at in this contribution to the thesis is the well-known iterative method alternating linear scheme (ALS). The ALS has been studied extensively and has successfully been applied to find low-rank tensor decompositions. The ALS for the CP decomposition is described in [11, 23], the Tucker decomposition is also treated in [23] and the ALS for the TT decomposition is studied in [20, 32]. The ALS optimizes the sought tensor on a manifold with fixed ranks [32, p. 1136]. Imposing the low-rank rank constraint is therefore easy to implement by choosing the ranks in advance.

The CP, Tucker and TT decomposition are all multilinear functions of all the TD components. This means that by assuming all TD components except the n th to be known, the tensor becomes a linear expression in the n th component [11, p. 4]. In the ALS all TD components are updated sequentially by making use of the TD's multilinearity. Each update step requires to solve a linear least squares problem given by

$$\min_{\mathbf{g}_n} \|\mathbf{y} - \mathbf{U}_{\setminus n} \mathbf{g}_n\|_{\text{F}}, \quad (2.3)$$

where $\mathbf{y} \in \mathbb{R}^{I_1 I_2 \dots I_N \times 1}$ and $\mathbf{g}_n \in \mathbb{R}^{K \times 1}$ denote the vectorization of \mathcal{Y} and \mathcal{G}_n , respectively, K being the number of elements in the n th TD component. The matrix $\mathbf{U}_{\setminus n} \in \mathbb{R}^{J \times K}$ is a

function off all TD components except the n th, where J is the number of elements in \mathbf{y} , and $\|\cdot\|_F$ denotes the Frobenius norm.

2

A drawback of the ALS is that it does not explicitly model the measurement noise \mathcal{E} , which in real-life applications is usually present. In this work, we model the noise by approaching the tensor decomposition in a Bayesian framework, treating all components as probability distributions. In this way, finding a low-rank TD approximation can be solved as a Bayesian inference problem: given the prior distributions of the TD components $p(\mathbf{g}_i)$ and the measurements \mathbf{y} , the posterior distribution $p(\{\mathbf{g}_i\} | \mathbf{y})$ can be found by applying Bayes' rule

$$p(\{\mathbf{g}_i\} | \mathbf{y}) = \frac{\overbrace{p(\mathbf{y}|\{\mathbf{g}_i\})}^{\text{likelihood}} \overbrace{p(\{\mathbf{g}_i\})}^{\text{prior}}}{\underbrace{p(\mathbf{y})}_{\text{evidence}}}, \quad (2.4)$$

where $\{\mathbf{g}_i\}$ denotes the collection of all TD components \mathbf{g}_i , for $i = 1, \dots, N$. We assume that the prior is Gaussian and, as in the ALS, we apply a block coordinate descent [25, p. 230]. This leads to a tractable inference for each substep.

Solving the low-rank tensor approximation problem in a Bayesian way has the following benefits. The assumptions on the measurement noise \mathcal{E} are considered and the uncertainty of each tensor decomposition component \mathbf{g}_n is quantified. Furthermore, prior knowledge can be explicitly taken into account and the resulting low-rank tensor estimate comes with a measure of uncertainty. We illustrate the benefits with numerical experiments.

Our main contribution is to approach the low-rank tensor approximation problem from a Bayesian perspective, treating all TD components as Gaussian random variables. This results in a probabilistic ALS algorithm. We ensure numerical stability by incorporating the orthogonalization step, present in the ALS algorithm for the TT decomposition, into the probabilistic framework. In addition, we propose an algorithm to approximate the mean and covariance of the low-rank tensor estimate's posterior density with the unscented transform in tensor train format. Our open-source MATLAB implementation can be found on <https://gitlab.tudelft.nl/cmmenzen/bayesian-als>.

RELATED WORK

Our work is related to inferring low-rank tensor decompositions with Bayesian methods for noisy continuous-valued multidimensional observations. While most literature considers either the CP or Tucker decomposition, our paper mainly focuses on the TT decomposition, but is also applicable to CP and Tucker. Also, in contrast to our paper, the related work mainly treats tensors with missing values. The main difference to the existing literature, however, are the modeling choices. While the existing work proposes different methods to perform approximate inference of the TD components, we make approximations that allow us to have a tractable inference. This is because we take inspiration from the ALS and each substep of the ALS has an analytical solution. Also, we assume that all TD components

are Gaussian random variables and that they are all independent. Thus, our method is preferable when these assumptions can be made for a given application.

In [30], [31] and [39] inference is performed with Gibbs sampling, using Gaussian priors for the columns of the CP decomposition's factor matrices. Variational Bayes is applied in [42] and [44]. The recovery of orthogonal factor matrices, optimizing on the Stiefel manifold with variational inference is treated by [6]. The Bayesian treatment of a low-rank Tucker decomposition for continuous data has been studied using variational inference [7, 43] and using Gibbs sampling [19]. Furthermore, an infinite Tucker decomposition based on a t -process, which is a kernel-based non-parametric Bayesian generalization of the low-rank Tucker decomposition, is proposed by [40]. The first literature about the probabilistic treatment of the tensor train decomposition using von-Mises-Fisher priors on the orthogonal cores and variational approximation with evidence lower bound is introduced by [17]. Recently, [18] published the probabilistic tensor decomposition toolbox for MATLAB, providing inference with variational Bayes and with Gibbs sampling.

2.2 TENSOR BASICS AND NOTATION

An N -way tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ is a generalization of a vector or a matrix to higher dimensions, where N is often referred to as the order of the tensor. We denote tensors by calligraphic, boldface, capital letters (e.g. \mathcal{Y}) and matrices, vectors and scalars by boldface capital (e.g. \mathbf{Y}), boldface lowercase (e.g. \mathbf{y}) and italic lower case (e.g. y) letters, respectively. To facilitate the description and computation of tensors, we use a graphical notation as depicted in Figure 2.1. The nodes represent a scalar, a vector, a matrix and an N -way tensor and edges correspond to a specific index. The number of edges is equal to how many indices need to be specified to identify one element in the object, e.g. row and column index for matrices. An identity matrix is generally denoted by \mathbf{I} . Its size is either specified in the context or as a subscript.

Often it is easier to avoid working with the tensors directly, but rather with a matricized or vectorized version of them. Therefore, we revise some useful definitions. In this context, a mode of a tensor refers to a dimension of the tensor.

Definition 2.2.1 (mode- n -unfolding [23, p. 459-460]) *The transformation of an N -way tensor into a matrix with respect to a specific mode is called the mode- n unfolding. It is denoted by*

$$\mathbf{Y}_{(n)} \in \mathbb{R}^{I_n \times I_1 \dots I_{n-1} I_{n+1} \dots I_N}.$$

The vectorization is a special case of the unfolding, denoted by the operator name $\text{vec}()$ and defined as

$$\text{vec}(\mathcal{Y}) = \mathbf{y} \in \mathbb{R}^{I_1 I_2 \dots I_N \times 1}.$$

Tensors can be multiplied with matrices defined as follows.

Definition 2.2.2 (n -mode product [23, p. 460]) *The n -mode product is defined as the multiplication of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_n \times \dots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$ in mode n , written as*

$$\mathcal{X} \times_n \mathbf{A} \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times J \times I_{n+1} \times \dots \times I_N}.$$

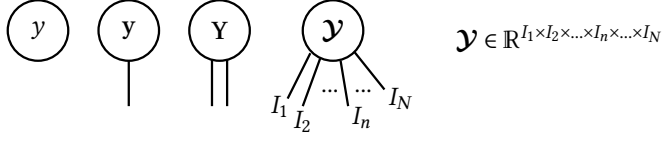


Figure 2.1: Visual depictions of a scalar, a vector, a matrix and an N -way tensor, where the nodes represent the object and the edges correspond to a specific index. The number of edges is equal to how many indices need to be specified to identify one element in the object, e.g. row and column index for matrices.

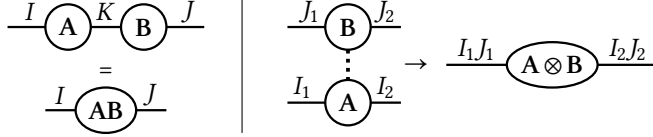


Figure 2.2: Left: Visual depictions of an index contraction between matrices A and B . Right: Visual depictions of an outer product between matrices A and B . The dotted line represents a summation over a rank-1 one connection. The resulting matrix is computed as the Kronecker product $A \otimes B$.

Element-wise, the $(i_1, i_2, \dots, i_{n-1}, j, i_{n+1}, \dots, i_N)$ -th entry of the result can be computed as

$$\sum_{i_n=1}^{I_n} \mathcal{X}(i_1, i_2, \dots, i_N) A(j, i_n).$$

Definition 2.2.3 (Kronecker product [23, p. 461]) The Kronecker product of matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{K \times L}$ is denoted by $A \otimes B$. The result is a matrix of size $(KI) \times (LJ)$ and is defined by

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1J}B \\ a_{21}B & a_{22}B & \cdots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \cdots & a_{IJ}B \end{bmatrix}.$$

Definition 2.2.4 (Kathri-Rao product [23, p. 462]) The Khatri–Rao product of matrices $A \in \mathbb{R}^{I \times K}$ and $B \in \mathbb{R}^{J \times K}$ is denoted by $A \odot B$. The result is a matrix of size $(JI) \times K$ defined by

$$A \odot B = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_K \otimes \mathbf{b}_K \end{bmatrix}.$$

The visual depictions of two important matrix operations are shown in Figure 2.2. On the left, a product between matrices $A \in \mathbb{R}^{I \times K}$ and $B \in \mathbb{R}^{K \times J}$ is shown, where the summation over the middle index K , also called contraction, is represented as an edge that connects both nodes. On the right, an outer product between matrices $A \in \mathbb{R}^{I_1 \times I_2}$ and $B \in \mathbb{R}^{J_1 \times J_2}$ is shown, where the dotted lines represent a rank-1 connection. The resulting matrix is the Kronecker product $A \otimes B \in \mathbb{R}^{I_1 J_1 \times I_2 J_2}$.

A tensor can be expressed as a function of simpler tensors that form a tensor decomposition. An extensive review about TDs can be found in [23]. The most notable are the CP decomposition, the Tucker decomposition, and the TT decomposition.

Definition 2.2.5 (CP decomposition [4, 15]) The CP decomposition consists of a set of matrices $\mathbf{G}_i \in \mathbb{R}^{I_i \times R}$, $i = 1, \dots, N$, called factor matrices and a weight vector $\boldsymbol{\lambda} \in \mathbb{R}^{R \times 1}$ that represent a given N -way tensor \mathcal{Y} . Element-wise, the (i_1, i_2, \dots, i_N) -th entry of \mathcal{Y} can be computed as

$$\sum_{r=1}^R \lambda(r) \mathbf{G}_1(i_1, r) \cdots \mathbf{G}_N(i_N, r),$$

where R denotes the rank of the decomposition.

Definition 2.2.6 (Tucker decomposition [38]) The Tucker decomposition consists of an N -way tensor $\mathcal{C} \in \mathbb{R}^{R_1 \times \dots \times R_N}$, called core tensor, and a set of matrices $\mathbf{G}_i \in \mathbb{R}^{I_i \times R_i}$, $i = 1, \dots, N$, called factor matrices, that represent a given N -way tensor \mathcal{Y} . Element-wise, the (i_1, i_2, \dots, i_N) -th entry of \mathcal{Y} can be computed as

$$\sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} \mathcal{C}(r_1, \dots, r_N) \mathbf{G}_1(i_1, r_1) \cdots \mathbf{G}_N(i_N, r_N),$$

where R_1, \dots, R_N denote the ranks of the decomposition. The factor matrices can be orthogonal, such that the Frobenius norm of the entire tensor is contained in the core tensor.

Definition 2.2.7 (The TT decomposition [28]) The tensor train decomposition consists of a set of three-way tensors $\mathcal{G}_i \in \mathbb{R}^{R_i \times I_i \times R_{i+1}}$, $i = 1, \dots, N$ called TT-cores, that represent a given N -way tensor \mathcal{Y} . Element-wise, the (i_1, i_2, \dots, i_N) -th entry of \mathcal{Y} can be computed as

$$\sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{N+1}=1}^{R_{N+1}} \mathcal{G}_1(r_1, i_1, r_2) \mathcal{G}_2(r_2, i_2, r_3) \cdots \mathcal{G}_N(r_N, i_N, r_{N+1}),$$

where R_1, \dots, R_{N+1} denote the ranks of the TT-cores and by definition $R_1 = R_{N+1} = 1$.

If the tensor is only approximately represented by a TD, then the ranks determine the accuracy of the approximation.

As mentioned in Section 2.1, to formulate the linear least squares problem for one update of the ALS, the TD's property of multi-linearity is exploited and it is expressed as $\mathbf{y} = \mathbf{U}_{\setminus n} \mathbf{g}_n$, with $\mathbf{U}_{\setminus n} \in \mathbb{R}^{J \times K}$ and $\mathbf{g}_n \in \mathbb{R}^{K \times 1}$, where J and K are the number of elements of \mathcal{Y} and \mathcal{G}_n , respectively. The following three examples describe how $\mathbf{U}_{\setminus n}$ is built for the CP decomposition, the Tucker decomposition, and the TT decomposition.

Example 2.2.8 If a tensor is represented in terms of a CP decomposition, the matrix $\mathbf{U}_{\setminus n}$ can be written as

$$\mathbf{U}_{\setminus n} = (\mathbf{G}_N \odot \cdots \odot \mathbf{G}_{n+1} \odot \mathbf{G}_{n-1} \odot \cdots \odot \mathbf{G}_1) \otimes \mathbf{I}_n \quad (2.5)$$

Note that $\mathbf{U}_{\setminus n}$ is of size $I_n I_1 \dots I_{n-1} I_{n+1} \dots I_N \times I_n R$, so the first dimension needs to be permuted in order to match $\mathbf{y} \in \mathbb{R}^{I_1 I_2 \dots I_N}$. The weight vector $\boldsymbol{\lambda}$ is absorbed into the factor matrix that is being updated. After each update, the columns of \mathbf{G}_n are normalized and the norms are stored in $\boldsymbol{\lambda}$. The CP-ALS algorithm can be found in [23, p. 471].

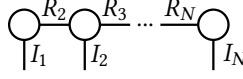
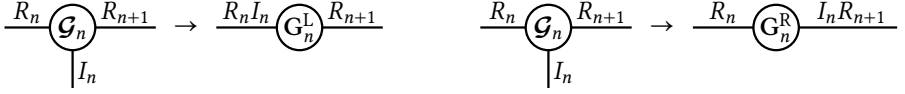
Figure 2.3: Visual depiction of a tensor train decomposition with N TT-cores.

Figure 2.4: Left: Visual depiction of a left-unfolding of a TT-core. Right: Right-unfolding of a TT-core.

Example 2.2.9 If a tensor is represented in terms of a Tucker decomposition, the matrix $\mathbf{U}_{\setminus n}$ can be written as

$$\mathbf{U}_{\setminus n} = [(\mathbf{G}_N \otimes \cdots \otimes \mathbf{G}_{n+1} \otimes \mathbf{G}_{n-1} \otimes \cdots \otimes \mathbf{G}_1) \mathbf{C}_{(n)}^\top] \otimes \mathbf{I}_{I_n}. \quad (2.6)$$

Note that $\mathbf{U}_{\setminus n}$ is of size $I_n I_1 \cdots I_{n-1} I_{n+1} \cdots I_N \times I_n R_n$, so the first dimension needs to be permuted in order to match $\mathbf{y} \in \mathbb{R}^{I_1 I_2 \cdots I_N}$. The core tensor is recomputed by solving

$$\mathbf{y} = (\mathbf{G}_N \otimes \cdots \otimes \mathbf{G}_1) \text{vec}(\mathcal{C}).$$

Example 2.2.10 If the tensor is represented in terms of a TT decomposition, the matrix $\mathbf{U}_{\setminus n}$ can be written as

$$\mathbf{U}_{\setminus n} = \mathcal{G}_{i>n} \otimes \mathbf{I}_{I_n} \otimes \mathcal{G}_{i<n}^\top \in \mathbb{R}^{I_1 I_2 \cdots I_N \times R_n I_n R_{n+1}}, \quad (2.7)$$

where $\mathcal{G}_{i<n}$ ($\mathcal{G}_{i>n}$) denotes a tensor obtained by contracting the TD components, left (right) of the n th core.

From here on, we will focus on the tensor train decomposition. We, therefore, review some of the main concepts. A tensor train can be represented by a diagram with nodes as the TT-cores and the edges as the modes of the approximated tensor. Connected edges are the summation over the ranks between two cores (Figure 2.3). To introduce a notion of orthonormality for TT-cores, a special case of Theorem 2.2.1 is used, creating unfoldings of the TT-cores defined as follows.

Definition 2.2.11 (Left- and right-unfolding [20, p. A689]) The left-unfolding \mathbf{G}_n^L and right-unfolding \mathbf{G}_n^R of a TT-core \mathcal{G}_n are the unfoldings of a core with respect to the first and last mode, respectively (Figure 2.4). Please note that the definition by [20] of the right-unfolding is the transposed version of this definition.

Definition 2.2.12 (Left-orthogonal and right-orthogonal [20, p. A689]) A TT-core \mathcal{G}_n is called left-orthogonal, if the left-unfolding \mathbf{G}_n^L satisfies

$$(\mathbf{G}_n^L)^\top \mathbf{G}_n^L = \mathbf{I}_{R_{n+1}}.$$

Analogously, a TT-core \mathcal{G}_n is called right-orthogonal, if the right-unfolding \mathbf{G}_n^R satisfies

$$\mathbf{G}_n^R (\mathbf{G}_n^R)^\top = \mathbf{I}_{R_n}.$$

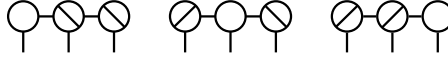


Figure 2.5: Visual depiction of tensor trains with three TT-cores in site- n -mixed-canonical form. Left: Norm in the first core and other cores left-orthogonal. Middle: Norm in the second core, first and last cores are left- and right-orthogonal, respectively. Right: Norm in the last core and other cores right-orthogonal.

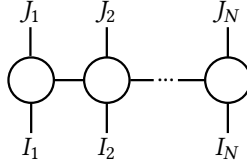


Figure 2.6: Visual depiction of a tensor train matrix. The row indices I_1, \dots, I_N point downwards, and the column indices J_1, \dots, J_N point upwards.

Definition 2.2.13 (site- n -mixed-canonical form [34, p. 113]) A tensor train is in site- n -mixed-canonical form if the TT-cores $\{\mathcal{G}_i\}_{i < n}$ are left-orthogonal and the TT-cores $\{\mathcal{G}_i\}_{i > n}$ are right-orthogonal. The n th TT-core is not orthogonal and it can be easily shown that

$$\|\mathcal{Y}\|_F = \|\mathcal{G}_n\|_F.$$

Figure 2.5 depicts different site- n -mixed-canonical forms for an exemplary three-way tensor train. On the left (right) figure, the Frobenius norm is contained in the first (last) core and all other cores are right- (left-) orthogonal, represented by the diagonal in the node.

A special case of the TT decomposition format is the tensor train matrix (Figure 2.6), which represents a large matrix in TT format. Tensor train matrices arise in the context of the unscented transform in Section 2.5.

Definition 2.2.14 (Tensor train matrix [27]) A tensor train matrix (TTm) consists of a set of four-way tensors $\mathcal{G}_i \in \mathbb{R}^{R_i \times I_i \times J_i \times R_{i+1}}$, $i = 1, \dots, N$ with $R_1 = R_{N+1} = 1$ that represents a matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$. The row and column indices are split into multiple row indices $I = I_1 \cdot I_2 \cdots I_N$ and column indices $J = J_1 \cdot J_2 \cdots J_N$, respectively and the matrix is transformed into a $2N$ -way tensor $\mathcal{Y}_A \in \mathbb{R}^{I_1 \times J_1 \times \cdots \times I_N \times J_N}$. Element-wise, the $(i_1, j_1, i_2, j_2, \dots, i_N, j_N)$ -th entry of \mathcal{Y}_A is computed as

$$\sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{N+1}=1}^{R_{N+1}} \mathcal{G}_1(r_1, i_1, j_1, r_2) \mathcal{G}_2(r_2, i_2, j_2, r_3) \cdots \mathcal{G}_N(r_N, i_N, j_N, r_{N+1}).$$

A TTm arises e.g. from an outer product between two vectors \mathbf{a} and \mathbf{b} , which corresponds to computing the product of one vector with the transpose of the other. If vector \mathbf{a} is represented by a TT with cores $\mathcal{A}_1, \dots, \mathcal{A}_N$, the resulting TTm is achieved by summing over a rank-1 connection between one of the TT-cores, e.g. the first, and vector \mathbf{b} (Figure 2.7 top). This result is a special case of the general TTm, where only one of the TT-cores has a double index. This means that only the row index is very large and therefore split into

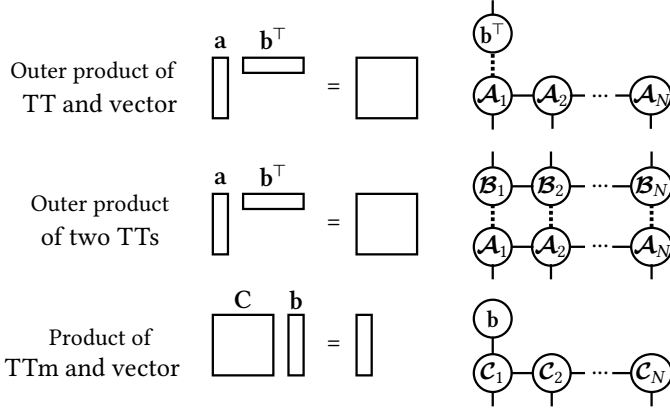


Figure 2.7: Operations with matrices and vectors. Top: Outer product between a vector \mathbf{a} , represented by a TT with cores $\mathcal{A}_1, \dots, \mathcal{A}_N$, and a vector \mathbf{b} . A rank-1 connection (dotted line) is summed over between the first TT-core and vector \mathbf{b}^T . Middle: Outer product between two vectors \mathbf{a} and \mathbf{b} represented by tensor trains with cores $\mathcal{A}_1, \dots, \mathcal{A}_N$ and $\mathcal{B}_1, \dots, \mathcal{B}_N$, respectively. A rank-1 connection is summed over between each core of the TTs. Bottom: The product between a matrix \mathbf{C} in TTm format with cores $\mathcal{C}_1, \dots, \mathcal{C}_N$ and a vector \mathbf{b} . The column index of the first TTm-core is summed over with the row index of the vector \mathbf{b} .

multiple indices, while the column index is not split. If both vectors in the outer product are represented by tensor trains with cores $\mathcal{A}_1, \dots, \mathcal{A}_N$ and $\mathcal{B}_1, \dots, \mathcal{B}_N$, respectively, then each core is summed over a rank-1 connection with the core of the other TT's transpose (Figure 2.7 middle). All cores have then a row and column indices. The product of a matrix \mathbf{C} in TTm format with cores $\mathcal{C}_1, \dots, \mathcal{C}_N$ with a vector \mathbf{b} is computed by summing over the column index of one TTm-core, e.g. the first, and the row index of the vector (Figure 2.7 bottom).

2.3 BAYESIAN INFERENCE FOR LOW-RANK TENSOR APPROXIMATION

In this section, we present a method to find a low-rank tensor decomposition using a similar strategy as in the ALS by solving a Bayesian inference problem. In this context, the vectorization of each TD component is treated as a Gaussian random variable, expressed in terms of a mean and a covariance. Generally, we denote a Gaussian probability distribution as $\mathcal{N}(\mathbf{m}, \mathbf{P})$, where \mathbf{m} is the mean and \mathbf{P} is the covariance. This section is organized as follows. First, we define the prior for the inference problem. Before computing the joint posterior distribution, we look at a simpler inference problem, stated in Theorem 2.3.1, where the posterior distribution of only one TD component is computed. Then, Theorem 2.3.4 describes the computation of the joint posterior by applying a block coordinate descent method and simplifying the inference problem to iteratively applying Theorem 2.3.1. Finally, our resulting Algorithm 1 is applied in an example.

To initialize the Bayesian inference problem, a multi-variate Gaussian prior is assigned

to every TD component

$$p(\mathbf{g}_i) = \mathcal{N}(\mathbf{m}_i^0, \mathbf{P}_i^0), \quad i = 1, \dots, N,$$

where \mathbf{m}_i^0 and \mathbf{P}_i^0 are the prior mean and covariance matrix, respectively. The TD components $\mathbf{g}_i \in \mathbb{R}^{R_i I_i R_{i+1} \times 1}$ are assumed to be statistically independent. Therefore, the joint prior distribution is given by

$$p(\{\mathbf{g}_i\}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{m}_1^0 \\ \mathbf{m}_2^0 \\ \vdots \\ \mathbf{m}_N^0 \end{bmatrix}, \begin{bmatrix} \mathbf{P}_1^0 & 0 & \dots & 0 \\ 0 & \mathbf{P}_2^0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mathbf{P}_N^0 \end{bmatrix}\right),$$

where $\{\mathbf{g}_i\}$ denotes the priors of all TD components. Because of the statistical independence, the joint prior distribution and the prior on one TD component conditioned on the other TD components, can be written as

$$p(\{\mathbf{g}_i\}) = p(\mathbf{g}_1)p(\mathbf{g}_2)\dots p(\mathbf{g}_N) \quad \text{and} \quad (2.8)$$

$$p(\mathbf{g}_n | \{\mathbf{g}_i\}_{i \neq n}) = p(\mathbf{g}_n), \quad (2.9)$$

respectively, where $\{\mathbf{g}_i\}_{i \neq n}$ denotes the collection of all TD components except the n th.

The joint posterior distribution $p(\{\mathbf{g}_i\} | \mathbf{y})$ is found by applying Bayes' rule. However, before solving this inference problem and inspired by a result described in [33, p. 29], we first look at the simpler problem to find the posterior distribution of one component, given the measurement and the other components.

Lemma 2.3.1 *Let the prior distribution $p(\mathbf{g}_n) = \mathcal{N}(\mathbf{m}_n^0, \mathbf{P}_n^0)$ and the likelihood $p(\mathbf{y} | \{\mathbf{g}_i\}) = \mathcal{N}(\mathbf{m}_y, \sigma^2 \mathbf{I})$ be Gaussian, where $\mathbf{m}_y = \mathbf{U}_{\setminus n} \mathbf{g}_n$. Further, let all TD components be statistically independent and let the TD be multilinear. Then, the posterior distribution $p(\mathbf{g}_n | \{\mathbf{g}_i\}_{i \neq n}, \mathbf{y}) = \mathcal{N}(\mathbf{m}_n^+, \mathbf{P}_n^+)$ of the n th component given the measurements and the other components is also Gaussian with mean \mathbf{m}_n^+ and covariance \mathbf{P}_n^+*

$$\mathbf{m}_n^+ = \left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}}{\sigma^2} \right]^{-1} \left[\frac{\mathbf{U}_{\setminus n}^\top \mathbf{y}}{\sigma^2} + (\mathbf{P}_n^0)^{-1} \mathbf{m}_n^0 \right] \quad (2.10)$$

$$\mathbf{P}_n^+ = \left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}}{\sigma^2} \right]^{-1}. \quad (2.11)$$

Proof 2.3.2 *The posterior distribution of one TD component conditioned on the other TD components and the measurements $p(\mathbf{g}_n | \mathbf{y}, \{\mathbf{g}_i\}_{i \neq n})$ can be found by applying Bayes' rule. Assuming that all components are statistically independent Equation (2.9) leads to*

$$p(\mathbf{g}_n | \mathbf{y}, \{\mathbf{g}_i\}_{i \neq n}) = \frac{p(\mathbf{y} | \{\mathbf{g}_i\})p(\mathbf{g}_n)}{p(\mathbf{y} | \{\mathbf{g}_i\}_{i \neq n})}. \quad (2.12)$$

Since the likelihood $p(\mathbf{y} | \{\mathbf{g}_i\})$ and prior $p(\mathbf{g}_n)$ are Gaussian, also the posterior will be Gaussian [33, p. 28-29, 209-210] with mean Equation (2.10) and covariance Equation (2.11).

Corollary 2.3.3 For $\lim \mathbf{P}_n^0 \rightarrow \infty$, Equation (2.10) reduces to the normal equations of the least squares problem and therefore the update equation of the conventional ALS

$$\mathbf{m}_n^+ = (\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n})^{-1} \mathbf{U}_{\setminus n}^\top \mathbf{y}. \quad (2.13)$$

2

Theorem 2.3.3 describes the case where there is no useful prior information available for the n th TD component. Thus, the certainty on the prior mean is zero, and $\lim \mathbf{P}_n^0 \rightarrow \infty$.

Now, we can use Theorem 2.3.1 to find the joint posterior distribution of all TD components as described in the following theorem.

Theorem 2.3.4 Let $p(\{\mathbf{g}_i\} | \mathbf{y})$ be the posterior joint distribution of all TD components given \mathbf{y} . Further, let the prior distribution $p(\mathbf{g}_n) = \mathcal{N}(\mathbf{m}_n^0, \mathbf{P}_n^0)$ of any component as well as the likelihood $p(\mathbf{y} | \{\mathbf{g}_i\}) = \mathcal{N}(\mathbf{m}_y, \sigma^2 \mathbf{I})$ be Gaussian, where the mean \mathbf{m}_y is the tensor represented by the TD, which is a nonlinear function of all the TD components. Further, let all TD components be statistically independent and let the TD be multilinear. Then, by applying block coordinate descent to find the posterior density, every step of the block coordinate descent corresponds to applying Theorem 2.3.1.

Proof 2.3.5 Bayes' rule and statistical independence Equation (2.8) gives

$$p(\{\mathbf{g}_i\} | \mathbf{y}) = \frac{p(\mathbf{y} | \{\mathbf{g}_i\}) p(\mathbf{g}_1) p(\mathbf{g}_2) \dots p(\mathbf{g}_N)}{p(\mathbf{y})}. \quad (2.14)$$

As in the conventional ALS, a block coordinate descent method is applied by conditioning the posterior distribution of one TD component on all the others. In this way, the TD components can be computed sequentially with Equation (2.12). In addition, due to the multilinearity of the TD the mean of the likelihood becomes a linear function of the n th TD component, $\mathbf{m}_y = \mathbf{U}_{\setminus n} \mathbf{g}_n$. Thus, every TD update corresponds to applying Theorem 2.3.1.

With Theorem 2.3.3, Theorem 2.3.4 gives a Bayesian interpretation of the ALS by deriving its update equation from the TD components defined as probability distributions. The following example shows how the distributions change with every update.

Example 2.3.6 (Distribution updates for a TD with three components) Assume we would like to apply Theorem 2.3.4 to find a TD with three components. First, the three TD components are initialized with a prior distribution. Then, the distributions are updated sequentially by computing the posterior with Bayes' rule, as shown in Figure 2.8. After updating the three TD components, the updates are repeated until a stopping criterion is met.

Algorithm 1 summarizes the steps of the ALS in a Bayesian framework. The mean and covariance of each TD component are sequentially updated, followed by the computation of $\mathbf{U}_{\setminus n}$ which is computed from $\{\mathbf{g}_i\}_{i \neq n}$. The stopping criterion is defined by the user, e.g. as a maximum number of iterations or the convergence of the residuals between the measurement and estimate, as used in the conventional ALS. It is also possible to consider the convergence of the TT-core's covariance matrices as a stopping criterion since these are additionally computed in the ALS in the Bayesian framework. Another possibility is to

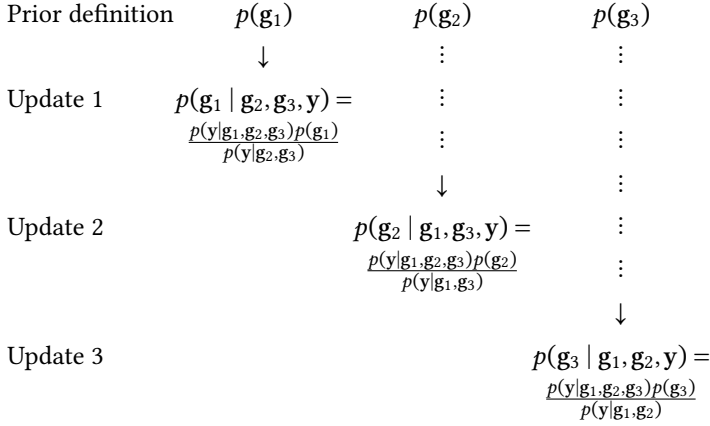


Figure 2.8: Distribution updates for example with three TD components.

Table 2.1: Computational cost per update and overall storage requirements for Algorithm 1.

TD	computational cost	storage
CP	$\mathcal{O}(R^3 I^3) + \mathcal{O}(R I^{N+1})$	$\mathcal{O}(NRI + NR^2 I^2)$
Tucker	$\mathcal{O}(R^{3N}) + \mathcal{O}(R^N I^N)$	$\mathcal{O}(NRI + R^N + NR^2 I^2 + R^{2N})$
TT	$\mathcal{O}(R^6 I^3) + \mathcal{O}(R^2 I^N)$	$\mathcal{O}(NR^2 I + NR^4 I^2)$

look at the convergence of the numerator of Bayes' rule.

For the complexity analysis, we use the following notation. The largest rank or the CP-rank is denoted by R and the largest dimension of the approximated tensor is denoted by I . The computational cost per update and the overall storage requirements are given in Table 2.1. The first term in the computational cost for CP, Tucker and TT represents the inversion that needs to be performed to compute the covariance matrix of the updated factor matrices, core tensor and TT cores, respectively. The cost for the Tucker core could be reduced, however, by incorporating an orthogonalization step, thus, avoiding the re-computation of the core tensor. The second term for all TDs is the complexity to compute $\mathbf{U}_{\setminus n}^\top \mathbf{y}$, which is dominant compared to the cost of computing $\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}$. In comparison, the conventional ALS has the same computational cost for every TD component update, and a total storage requirement of $\mathcal{O}(NRI)$ for CP, $\mathcal{O}(NRI + R^N)$ for Tucker, and $\mathcal{O}(NR^2 I)$ for TTs. The ALS in a Bayesian framework has an additional term in the storage requirements, because it computes the covariance matrix for every TD component.

Our method also opens up the possibility to recursively estimating the mean and covariance of the TD components. In case a new noisy measurement \mathbf{y} of the same underlying tensor becomes available, Algorithm 1 can be applied repeatedly with the output mean and covariance from the previous execution as the prior for the new execution.

Algorithm 1 ALS in a Bayesian framework

Require: Prior mean $\{\mathbf{m}_i^0\}$ and covariance $\{\mathbf{P}_i^0\}$, $i = 1, \dots, N$, measurement \mathbf{y} and noise variance σ^2 .

Ensure: Posterior mean $\{\mathbf{m}_i^+\}$ and covariance $\{\mathbf{P}_i^+\}$, $i = 1, \dots, N$.

- 1: Set $\{\mathbf{m}_i\} := \{\mathbf{m}_i^0\}$, $\{\mathbf{P}_i\} := \{\mathbf{P}_i^0\}$, $i = 1, \dots, N$.
- 2: **while** Stopping criterion is not true **do**
- 3: **for** $n = 1, \dots, N$ **do**
- 4: Compute $\mathbf{U}_{\setminus n}$ with Equation (2.5) for CP, Equation (2.6) for Tucker or Equation (2.7) for TT, using the mean of the TD components $\{\mathbf{m}_i\}_{i \neq n}$.
- 5: $\mathbf{P}_n^+ \leftarrow \left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}}{\sigma^2} \right]^{-1}$
- 6: $\mathbf{m}_n^+ \leftarrow \mathbf{P}_n^+ \left[\frac{\mathbf{U}_{\setminus n}^\top \mathbf{y}}{\sigma^2} + (\mathbf{P}_n^0)^{-1} \mathbf{m}_n^0 \right]$
- 7: $\mathbf{m}_n \leftarrow \mathbf{m}_n^+$, $\mathbf{P}_n \leftarrow \mathbf{P}_n^+$
- 8: **end for**
- 9: **end while**

2.4 ORTHOGONALIZATION STEP IN BAYESIAN FRAMEWORK FOR A TT

Every iteration of Algorithm 1 requires the inversion

$$\left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}}{\sigma^2} \right]^{-1}, \text{ corresponding to } [\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}]^{-1} \quad (2.15)$$

in the conventional ALS update. To avoid the propagation of numerical errors and ensure numerical stability, some ALS algorithms, e.g., the one for the TT decomposition, include an orthogonalization step after every update. In this way, the condition number of each sub-problem can not become worse than the one of the overall problem [20, p. A701]. In this section, we present how we integrate the orthogonalization procedure into the ALS in a Bayesian framework for a TT decomposition in site- n -mixed-canonical form. The same can also be applied to a Tucker decomposition with orthogonal factor matrices.

We first describe, how the orthogonalization step is performed in the conventional ALS and then how we integrate it into the ALS in a Bayesian framework. Here, we differentiate between the prior distributions of each TT-core and the initial guess for each TT-core, which initializes the conventional ALS. In the conventional ALS with orthogonalization step, the initial TT is transformed into the site-1-mixed-canonical form, where the Frobenius norm of the first TT-core corresponds to the Frobenius norm of the entire tensor train. The update is always performed on the core that contains the Frobenius norm. The procedure, therefore, requires transformations that separate the Frobenius norm from the updated TT-core and moves it to the next TT-core to be updated.

The initial TT is transformed into site-1-mixed-canonical form, by orthogonalizing the N th up to the 2nd TT-core as illustrated in Figure 2.9 for a TT with three cores. To move the Frobenius norm from the n th TT-core to the $(n-1)$ th, the n th TT-core is orthogonalized

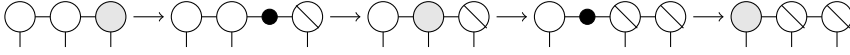


Figure 2.9: Visual depiction of a TT transformation into site-1-mixed-canonical form.

by applying the thin **QR**-decomposition on

$$(\mathbf{G}_n^{\mathbf{R}})^{\top} = \mathbf{Q}_n^{\mathbf{R}} \mathbf{R}_n^{\mathbf{R}}. \quad (2.16)$$

Then, $\mathbf{G}_n^{\mathbf{R}}$ is replaced by

$$\mathbf{G}_n^{\mathbf{R}} \leftarrow (\mathbf{Q}_n^{\mathbf{R}})^{\top} \quad (2.17)$$

and the non-orthogonal part, illustrated by the small circle in Figure 2.9, is absorbed into the $(n-1)$ th core with

$$\mathcal{G}_{n-1} \leftarrow \mathcal{G}_{n-1} \times_3 \mathbf{R}_n^{\mathbf{R}}. \quad (2.18)$$

Equations (2.16) to (2.18) are applied to the N th until the 2nd TT-core, leading to the TT in site-1-mixed-canonical form. Then, the first core is updated, followed by a transformation to move the Frobenius norm to the second core, and so on. Since the Frobenius norm moves to the right, the orthogonalization step consists of applying the thin **QR**-decomposition on the left-unfolding

$$\mathbf{G}_n^{\mathbf{L}} = \mathbf{Q}_n^{\mathbf{L}} \mathbf{R}_n^{\mathbf{L}}. \quad (2.19)$$

The n th and $(n+1)$ th core are replaced by

$$\mathbf{G}_n^{\mathbf{L}} \leftarrow \mathbf{Q}_n^{\mathbf{L}} \quad \text{and} \quad \mathcal{G}_{n+1} \leftarrow \mathcal{G}_{n+1} \times_1 \mathbf{R}_n^{\mathbf{L}}, \quad (2.20)$$

respectively. After the N th core is updated, the updating scheme goes backwards, using again Equations (2.16) to (2.18) for the orthogonalization step. When the Frobenius norm is absorbed back into the first core, one back and forth sweep of the ALS algorithm is concluded.

In the following, we describe how the transformation steps affect the distributions representing the TT-cores in the ALS in a Bayesian framework. The transformation of the random variables can be derived from Equations (2.16) to (2.20). When the Frobenius norm is moved to the left, the mean of the n th core becomes

$$\mathbf{m}_n \leftarrow \text{vec} \left((\mathbf{Q}_n^{\mathbf{R}})^{\top} \right),$$

where $(\mathbf{Q}_n^{\mathbf{R}})^{\top}$ is computed from Equation (2.17). To obtain the transformed covariance of the n th TT-core, Equation (2.16) is rewritten as

$$(\mathbf{Q}_n^{\mathbf{R}})^{\top} = (\mathbf{R}_n^{\mathbf{R}})^{-\top} \mathbf{G}_n^{\mathbf{R}}.$$

Now, the right-hand side, is vectorized by summing over a rank-1 connection between $(\mathbf{R}_n^{\mathbf{R}})^{-\top}$ and an identity matrix of size $I_n R_{n+1} \times I_n R_{n+1}$ that has a connected edge with $\mathbf{G}_n^{\mathbf{R}}$ as depicted in Figure 2.10. This leads to a transformation term

$$\mathbf{I} \otimes (\mathbf{R}_n^{\mathbf{R}})^{-\top} \quad (2.21)$$

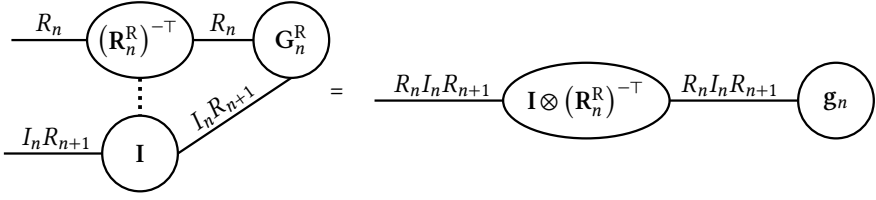


Figure 2.10: Visual depiction of how the non-orthogonal part is separated from the TD component's mean.

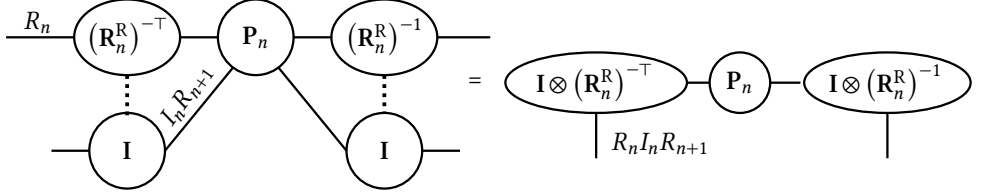


Figure 2.11: Visual depiction of how the covariance matrix is transformed in the orthogonalization step.

that orthogonalizes \mathbf{g}_n . The diagram in Figure 2.11 shows how this transformation is applied to the covariance matrix. The transformation term and its transpose are multiplied on the left and right side of \mathbf{P}_n , respectively, resulting in

$$\begin{aligned} \mathbf{P}_n &\leftarrow \left(\mathbf{I} \otimes (\mathbf{R}_n^R)^{-\top} \right) \mathbf{P}_n \left(\mathbf{I} \otimes (\mathbf{R}_n^R)^{-\top} \right)^\top \\ &= \left(\mathbf{I} \otimes (\mathbf{R}_n^R)^{-\top} \right) \mathbf{P}_n \left(\mathbf{I} \otimes (\mathbf{R}_n^R)^{-1} \right). \end{aligned}$$

The transformations of the $(n-1)$ th core to absorb the Frobenius norm, can be derived from Equation (2.18) in a similar way as explained above, resulting in a transformation term

$$\mathbf{R}_n^R \otimes \mathbf{I}. \quad (2.22)$$

When the Frobenius norm is moved to the right during the orthogonalization step, the transformations for the updated core and the next core to be updated become

$$(\mathbf{R}_n^L)^{-\top} \otimes \mathbf{I} \quad \text{and} \quad \mathbf{I} \otimes \mathbf{R}_n^L, \quad (2.23)$$

respectively. It can be easily shown that the transformations for the orthogonalization step, do not affect the statistical independence of the joint distribution of the random variables, since the transformations are performed on each variable individually. The following example shows the updating for the transformation scheme of the random variables that represent an exemplary three core TT.

Example 2.4.1 Distribution updates and orthogonalization transformations for a TT with three cores Assume we would like to apply Theorem 2.3.4 to find a TT with three cores and keep the TD in site- n -mixed- canonical form. The three TT-cores are initialized with a prior distribution and transformed such that the corresponding TT is in site-1-mixed-canonical

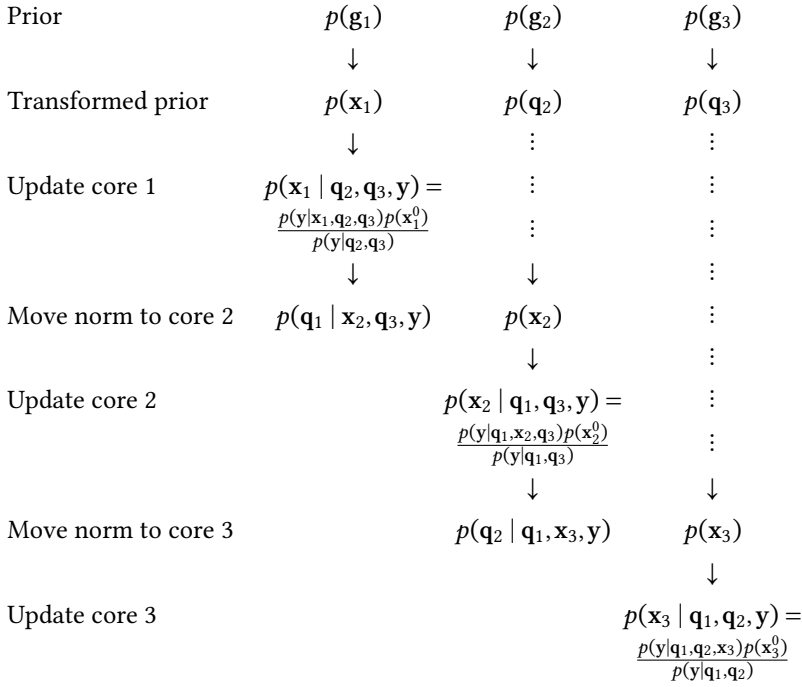


Figure 2.12: Distribution updates with orthogonalization step for example with three TT-cores.

form. The random variables that represent the orthogonal cores are denoted by \mathbf{q}_i , $i = 1, 2, 3$ and the random variable representing the TT-core that contains the Frobenius norm is denoted by \mathbf{x}_i , $i = 1, 2, 3$. After the random variables are transformed into site-1-mixed-canonical form using (2.21) and (2.22), the first core is updated followed by moving the Frobenius norm to the second core. Then the second core is updated and the Frobenius norm is moved to the last. When this half-sweep, as shown below, is completed using the transformations from (2.23), the same procedure is repeated in the opposite direction, requiring again (2.21) and (2.22). The example is depicted in Figure 2.12.

The ALS in a Bayesian framework with orthogonalization step has another difference compared to the one without orthogonalization. The update equations for the mean and covariance, Equation (2.10) and Equation (2.11), are affected by the TT decomposition being in site- n -mixed-canonical form: the matrix $\mathbf{U}_{\setminus n}$ becomes orthogonal and the update equations simplify to

$$\mathbf{m}_n^+ = \underbrace{\left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{I}}{\sigma^2} \right]^{-1}}_{\mathbf{P}_n^+} \left[\frac{\mathbf{U}_{\setminus n}^\top \mathbf{y}}{\sigma^2} + (\mathbf{P}_n^0)^{-1} \mathbf{m}_n^0 \right].$$

In this case $\mathbf{U}_{\setminus n}^\top \mathbf{y}$ corresponds to the update of the conventional ALS (2.13), due to the orthogonality of $\mathbf{U}_{\setminus n}$.

2

Algorithm 2 summarizes the ALS in a Bayesian framework with orthogonalization step for a TT decomposition. The computational cost of one update in Algorithm 2 is $\mathcal{O}(R^6 I^3)$ for the inversion and $\mathcal{O}(R^3 I)$ for the thin QR-factorization and the storage requirement is $\mathcal{O}(R^2 I + R^4 I^2)$, where R is the largest TT-rank and I is the largest dimension of the approximated tensor. The only difference compared to the conventional ALS in terms of complexity, is the additionally required storage for the covariance matrices. Thus, the number of elements of one TD component, depending on the ranks, will be the limiting factor for the computational complexity.

Algorithm 2 ALS in Bayesian framework with orthogonalization step

Require: Prior mean $\{\mathbf{m}_i^0\}$ and covariance $\{\mathbf{P}_i^0\}$, $i = 1, \dots, N$, measurement \mathbf{y} and noise variance σ^2 .

Ensure: Posterior mean $\{\mathbf{m}_i^+\}$ and covariance $\{\mathbf{P}_i^+\}$, $i = 1, \dots, N$.

- 1: Transform random variables such that the corresponding TT decomposition is in site-1-mixed-canonical form.
- 2: Set $\{\mathbf{m}_i\} := \{\mathbf{m}_i^0\}$, $\{\mathbf{P}_i\} := \{\mathbf{P}_i^0\}$, $i = 1, \dots, N$.
- 3: **while** stopping criterion is not true **do**
- 4: **for** $n = 1, \dots, N, N-1, \dots, 2$ **do**
- 5: Compute $\mathbf{U}_{\setminus n}$ with Equation (2.7) for TT using the mean of the TD components $\{\mathbf{m}_i\}_{i \neq n}$.
- 6: $\mathbf{P}_n^+ \leftarrow \left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{I}}{\sigma^2} \right]^{-1}$
- 7: $\mathbf{m}_n^+ \leftarrow \mathbf{P}_n^+ \left[\frac{\mathbf{U}_n^\top \mathbf{y}}{\sigma^2} + (\mathbf{P}_n^0)^{-1} \mathbf{m}_n^0 \right]$
- 8: **if** next core is to the right **then**
- 9: $\mathbf{m}_n^+ \leftarrow \text{vec}(\mathbf{Q}_n^L)$, with \mathbf{Q}_n^L from thin QR-factorization of \mathbf{G}_n^L
- 10: $\mathbf{P}_n^+ \leftarrow \left((\mathbf{R}_n^L)^{-\top} \otimes \mathbf{I} \right) \mathbf{P}_n^+ \left((\mathbf{R}_n^L)^{-1} \otimes \mathbf{I} \right)$
- 11: $\mathbf{m}_{n+1} \leftarrow (\mathbf{I} \otimes \mathbf{R}_n^L) \mathbf{m}_{n+1}$
- 12: $\mathbf{P}_{n+1} \leftarrow (\mathbf{I} \otimes \mathbf{R}_n^L) \mathbf{P}_{n+1} (\mathbf{I} \otimes (\mathbf{R}_n^L)^\top)$
- 13: **else if** next core is on the left **then**
- 14: $\mathbf{m}_n^+ \leftarrow \text{vec} \left((\mathbf{Q}_n^R)^\top \right)$, with \mathbf{Q}_n^R from thin QR-factorization of \mathbf{G}_n^R
- 15: $\mathbf{P}_n^+ \leftarrow \left(\mathbf{I} \otimes (\mathbf{R}_n^R)^{-\top} \right) \mathbf{P}_n^+ \left(\mathbf{I} \otimes (\mathbf{R}_n^R)^{-1} \right)$
- 16: $\mathbf{m}_{n-1} \leftarrow (\mathbf{R}_n^R \otimes \mathbf{I}) \mathbf{m}_{n-1}$
- 17: $\mathbf{P}_{n-1} \leftarrow (\mathbf{R}_n^R \otimes \mathbf{I}) \mathbf{P}_{n-1} \left((\mathbf{R}_n^R)^\top \otimes \mathbf{I} \right)$
- 18: **end if**
- 19: $\mathbf{m}_n \leftarrow \mathbf{m}_n^+$, $\mathbf{P}_n \leftarrow \mathbf{P}_n^+$
- 20: Apply the transformations of the lines 7-10 or 12-15 to $\mathbf{m}_n^0, \mathbf{P}_n^0$.
- 21: **end for**
- 22: **end while**

2.5 UNSCENTED TRANSFORM IN TT FORMAT

In Algorithm 1 and Algorithm 2 we compute the posterior distributions of the TT-cores $p(\mathbf{g}_n | \{\mathbf{g}_i\}_{i \neq n}, \mathbf{y})$. However, we are interested in computing the distribution of the low-rank tensor estimate \mathcal{G} , which is computed with a non-linear function dependent on the posterior distributions and is, therefore, not Gaussian. The unscented transform (UT) [22] can approximate the mean \mathbf{m}_{UT} and covariance \mathbf{P}_{UT} of the sought distribution. In this section, we show how we can perform the UT in TT format. In this way, the direct computation of the potentially large covariance matrix can be avoided.

Generally, the UT approximates the mean and covariance of a distribution that is a non-linear function of a known distribution $\mathbf{h} \sim \mathcal{N}(\mathbf{m}, \mathbf{P})$ with mean $\mathbf{m} \in \mathbb{R}^{M \times 1}$ and covariance $\mathbf{P} \in \mathbb{R}^{M \times M}$ [33, p. 81-84]. Firstly, $2M + 1$ sigma points are formed with

$$\mathbf{x}^{(0)} = \mathbf{m}, \quad (2.24)$$

$$\mathbf{x}^{(i)} = \mathbf{m} + \sqrt{M + \lambda} \begin{bmatrix} \sqrt{\mathbf{P}} \end{bmatrix}_i, \quad i = 1, \dots, M, \quad (2.25)$$

$$\mathbf{x}^{(i+M)} = \mathbf{m} - \sqrt{M + \lambda} \begin{bmatrix} \sqrt{\mathbf{P}} \end{bmatrix}_i, \quad i = 1, \dots, M, \quad (2.26)$$

where the square root of the covariance matrix $\sqrt{\mathbf{P}}$ corresponds to the Cholesky factor, such that $\sqrt{\mathbf{P}} \sqrt{\mathbf{P}}^\top = \mathbf{P}$, where $\begin{bmatrix} \sqrt{\mathbf{P}} \end{bmatrix}_i$ is the i -th column of that matrix. The scaling parameter λ is defined as $\lambda = \alpha^2(M + \kappa) - M$, where α and κ determine the spread of the sigma points around the mean. Secondly, the sigma points are propagated through the non-linearity and thirdly, the approximated mean \mathbf{m}_{UT} and covariance \mathbf{P}_{UT} are computed as

$$\mathbf{m}_{\text{UT}} = \sum_{i=0}^{2M} w_i^{\text{m}} \mathcal{S}^{(i)}, \quad (2.27)$$

$$\mathbf{P}_{\text{UT}} = \sum_{i=0}^{2M} w_i^{\text{P}} (\mathcal{S}^{(i)} - \mathbf{m}_{\text{UT}}) (\mathcal{S}^{(i)} - \mathbf{m}_{\text{UT}})^\top, \quad (2.28)$$

where $\mathcal{S}^{(i)}$ are the transformed sigma points. The scalars w_i^{m} and w_i^{P} denote weighting factors, defined as

$$w_0^{\text{m}} = \frac{\lambda}{M + \lambda}, \quad w_0^{\text{P}} = w_0^{\text{m}} + (1 - \alpha^2 + \beta),$$

$$w_i^{\text{m}} = w_i^{\text{m}} = w_i^{\text{P}} = w_i^{\text{P}} = \frac{1}{2(M + \lambda)}, \quad i = 1, \dots, 2M.$$

Literature suggests $\alpha = 0.001$, $\kappa = 3 - M$ [16, p. 229] and for Gaussian distributions $\beta = 2$ [33, p. 229].

In order to use the UT in TT format, the known distribution $\mathbf{h} \sim \mathcal{N}(\mathbf{m}, \mathbf{P})$ is computed from the cores' mean and covariance as

$$\mathbf{h} \sim \mathcal{N}(\mathbf{m}, \mathbf{P}) = \mathcal{N} \left(\begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}, \begin{bmatrix} \mathbf{P}_1 & 0 & \dots & 0 \\ 0 & \mathbf{P}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mathbf{P}_N \end{bmatrix} \right). \quad (2.29)$$

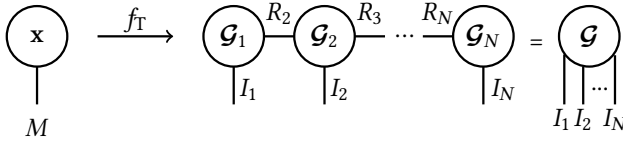


Figure 2.13: Visual depiction of the non-linear transformation from vector \mathbf{x} into a TT with cores \mathcal{G}_i , $i = 1, \dots, N$ that represents tensor \mathcal{G} .

2

The mean, consisting of the stacked vectorized cores, is of size $M \times 1$ with

$$M = \sum_{n=1}^N R_n I_n R_{n+1}, \quad R_1 = R_{N+1} = 1.$$

The covariance matrix of size $M \times M$ is block diagonal, since we assume the TT-cores to be statistically independent. The non-linear function for the UT in TT format is defined as

$$f_T : \mathbb{R}^{M \times 1} \rightarrow \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N} \text{ given by } \mathbf{x} \mapsto \mathcal{G}, \quad (2.30)$$

where \mathbf{x} is a vector of size $M \times 1$. The transformation of a vector into a tensor and is depicted in Figure 2.13.

The formation and propagation of the sigma points works as follows. The first sigma point $\mathbf{x}^{(0)}$ is the mean \mathbf{m} from Equation (2.29) and propagated through the non-linearity, it corresponds to the TT represented by the distributions determined by Algorithm 1. To facilitate later steps, the remaining sigma points from Equation (2.25) and Equation (2.26) are organized into two matrices, according to

$$\mathbf{A}_+ = \sum_{i=1}^M \mathbf{x}^{(i)} \mathbf{e}_i^\top \quad (2.31)$$

$$\mathbf{A}_- = \sum_{i=1}^M \mathbf{x}^{(M+i)} \mathbf{e}_i^\top, \quad (2.32)$$

where \mathbf{e}_i denotes a vector with zeros everywhere except a 1 at location i . In this way, the propagation through the non-linearity of all sigma points then becomes a propagation of every summand $\mathbf{x}^{(i)} \mathbf{e}_i^\top$ and $\mathbf{x}^{(M+i)} \mathbf{e}_i^\top$, respectively. The propagation is achieved by forming TT-cores from $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(M+i)}$ and summing over a rank-1 connection between the first core and the vector \mathbf{e}_i^\top as shown in Figure 2.14.

Then, all summands of $\mathbf{A}_+ \in \mathbb{R}^{M \times M}$ and $\mathbf{A}_- \in \mathbb{R}^{M \times M}$, are summed together, respectively, by stacking the cores according to [29, p. 2308], as illustrated in Figure 2.15 (left). The summation causes the ranks of the TTm to increase and a rounding procedure [28, p. 2301-2305] needs to be applied to reduce the ranks back to the required precision. Finally, the vector containing the weights $\mathbf{w}^m = \mathbf{w}^m \mathbf{1}_M$ is absorbed into the first core (Figure 2.15, right).

$$\mathbf{x}^{(i)} \mathbf{e}_j^\top = \left[\begin{array}{c|c} 0 \cdots 0 & 0 \cdots 0 \\ \vdots & \vdots \\ \vdots & \vdots \\ 0 \cdots 0 & 0 \cdots 0 \end{array} \right] \mathbf{x}^{(i)} \rightarrow \begin{array}{c} |M \\ \textcircled{\mathbf{e}_j^\top} \\ \vdots \\ \textcircled{\phantom{\mathbf{e}_j^\top}} \end{array} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \quad \begin{array}{l} i = 1 \dots 2M+1 \\ j = 1 \dots M \end{array}$$

Figure 2.14: Propagation of each sigma point as column of matrix $\mathbf{x}^{(i)} \mathbf{e}_i^\top$ through the non-linearity.

$$\sum_{j=1}^M \begin{array}{c} |M \\ \textcircled{\mathbf{e}_j^\top} \\ \vdots \\ \textcircled{\phantom{\mathbf{e}_j^\top}} \end{array} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \begin{array}{l} I_1 \\ I_2 \\ I_3 \end{array} = \begin{array}{c} |M \\ \textcircled{\phantom{\mathbf{e}_j^\top}} \\ \vdots \\ \textcircled{\phantom{\mathbf{e}_j^\top}} \end{array} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \begin{array}{l} I_1 \\ I_2 \\ I_3 \end{array} \quad \Bigg| \quad \begin{array}{c} |M \\ \textcircled{\mathbf{w}^m} \\ \vdots \\ \textcircled{\phantom{\mathbf{e}_j^\top}} \end{array} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \begin{array}{l} I_1 \\ I_2 \\ I_3 \end{array}$$

Figure 2.15: Visual depiction of Equation (2.27) as sum over sigma points (left) and absorption of the weight vector \mathbf{w}^m into the first core (right).

The computation of the covariance matrix from Equation (2.28) is divided into two steps. Firstly, \mathbf{m}_{UT} is subtracted from the sum over the sigma points (Figure 2.15, left). This is achieved by creating a matrix, where \mathbf{m}_{UT} is stacked M times next to each other. The visual depiction of this operation equals to the one in Figure 2.14 with the difference that the multiplied vector is $\mathbf{1}_M^\top$. Secondly, the result from the first step absorbs the weights into the first core, as depicted in Figure 2.16. The resulting covariance matrix \mathbf{P}_{UT} in the three-core example is a TT matrix that corresponds to a matrix of size $I_1 I_2 I_3 \times I_1 I_2 I_3$.

The computation of the approximate mean and covariance with the UT in TT format is summarized in Algorithm 3. The computational cost depends on the ranks of the TD, since M is a function of the ranks. The bottleneck of Algorithm 3 is the rounding procedure necessary after performing summations in TT format. It has a cost of $\mathcal{O}(R^3 I^2 N)$ for a TT matrix.

$$\begin{array}{c} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \textcircled{\mathbf{w}^p \mathbf{I}_M} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \\ I_1 \quad \quad \quad I_1 \\ \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \\ I_2 \quad \quad \quad I_2 \\ \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \\ I_3 \quad \quad \quad I_3 \end{array} = \begin{array}{c} \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \\ I_1 \quad \quad I_1 \\ \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \\ I_2 \quad \quad I_2 \\ \text{---} \textcircled{\phantom{\mathbf{e}_j^\top}} \text{---} \\ I_3 \quad \quad I_3 \end{array}$$

Figure 2.16: Visual depiction of Equation (2.28) with $\mathbf{w}^p \mathbf{I}_M$ as a diagonal matrix containing the weight factors on the diagonal.

Algorithm 3 Approximation of the low-rank tensor estimate's mean and covariance with the unscented transform in TT format.

Require: The mean and covariances of each TT-core $\{\mathbf{m}_i, \mathbf{P}_i\}$, $i = 1, \dots, N$ computed with the ALS in a Bayesian framework.

Ensure: The approximated mean \mathbf{m}_{UT} and covariance \mathbf{P}_{UT} in TT format of the low-rank tensor estimate's distribution.

- 1: Compute sigma point $\mathbf{x}^{(0)}$ with Equation (2.24).
 - 2: Compute remaining sigma points with Equations (2.25) and (2.26) and organize them into groups according to Equations (2.31) and (2.32).
 - 3: Propagate sigma points through Equation (2.30), where groups from step 2 are propagated as shown in Figure 2.13.
 - 4: Estimate the mean \mathbf{m}_{UT} with Equation (2.27) as shown in Figure 2.15.
 - 5: Estimate the covariance \mathbf{P}_{UT} with Equation (2.28) as shown in Figure 2.16.
-

2.6 NUMERICAL EXPERIMENTS

In this section, we present the numerical examples that test the algorithms. All experiments with exception of the last were performed with MATLAB R2020b on a Dell computer with processor Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz 2.11 GHz and 8GB of RAM. The last experiment is performed on a Lenovo computer with processor Intel(R) Core(TM) i7-10700KF CPU @ 3.80GHz 3.79 GHz and 16GB of RAM. The implementation of the experiments can be found on <https://gitlab.tudelft.nl/cmmenzen/bayesian-als>.

The first three experiments are performed with a random TT, \mathcal{G} , that represents the ground truth and has the cores

$$\mathcal{G}_{1,\text{truth}} \in \mathbb{R}^{1 \times 5 \times 3}, \quad \mathcal{G}_{2,\text{truth}} \in \mathbb{R}^{3 \times 5 \times 3} \quad \text{and} \quad \mathcal{G}_{3,\text{truth}} \in \mathbb{R}^{3 \times 5 \times 1}.$$

The entries of each TT-core are drawn from a standard normal distribution. After computing the tensor $\mathcal{Y}_{\text{truth}} \in \mathbb{R}^{5 \times 5 \times 5}$ from the TT-cores and vectorizing it, a noisy sample \mathbf{y} is formed with

$$\mathbf{y} = \mathbf{y}_{\text{truth}} + \boldsymbol{\epsilon} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}), \quad (2.33)$$

where $\mathbf{y}_{\text{truth}}$ denotes the vectorized ground truth and $\boldsymbol{\epsilon}$ is a realization of random noise. The noisy samples of the same underlying tensor formed with Equation (2.33) are uncorrelated. The covariance of the measurement noise is influenced by fixing the signal-to-noise ratio

$$\text{SNR}_{\text{dB}} = 10 \log_{10} \frac{\|\mathbf{y}\|^2}{\|\boldsymbol{\epsilon}\|^2}.$$

If not stated otherwise, the signal-to-noise ratio is set to zero dB. Some experiments use multiple noisy samples \mathbf{y} , computed from Equation (2.33). In this case, the estimate is recursively updated. Initially, the prior TT is inputted to Algorithm 1 together with a sample \mathbf{y} . After the execution of Algorithm 1, the output mean and covariance is used as a prior for the next execution together with a new sample \mathbf{y} . This recursive updating is very suitable for the ALS in a Bayesian framework, because it can deal with prior knowledge

on the TD components. For the conventional ALS, the estimate from an execution of the algorithm that computes a TT with the ALS is used as an initial TT for the next execution.

2.6.1 CONVERGENCE ANALYSIS OF MAXIMIZATION PROBLEM

In the ALS in a Bayesian framework, we solve the optimization problem given by Equation (2.2). In this context, we define the relative error between the low-rank estimate \mathbf{g} and the ground truth $\mathbf{y}_{\text{truth}}$ as

$$\varepsilon_{\text{truth}} = \frac{\|\mathbf{y}_{\text{truth}} - \mathbf{g}\|}{\|\mathbf{y}_{\text{truth}}\|}$$

and the relative error between the low-rank estimate \mathbf{g} and the noisy sample \mathbf{y} as

$$\varepsilon_{\text{meas}} = \frac{\|\mathbf{y} - \mathbf{g}\|}{\|\mathbf{y}\|}.$$

In the first experiment, we look at the errors defined above in order to analyze the convergence of Algorithm 1. In addition, we look at the evolution of the log likelihood times the prior, since from Theorem 2.3.4 it follows that the numerator of the logarithm of Equation (2.4) needs to be maximized to compute the posterior of all TD components.

In this experiment, only one noisy sample \mathbf{y} is used. The prior mean is initialized randomly and the covariance for each core is set to $200^2 \mathbf{I}$, meaning a low certainty on the prior mean. The experiment is performed 100 times with the same TT, \mathcal{G} , but with different priors. Then, the mean of the 100 results is plotted with a region of twice the standard deviation. Figure 2.17 (left) shows how both relative errors decrease rapidly and converge after approximately 5 iterations in Algorithm 1. Figure 2.17 (right) shows how the product of log likelihood and prior increases during the first approximately 6 iterations, converging to a fixed value. Both subfigures of Figure 2.17 also show how the region of twice the standard deviation from the 100 trials, becomes smaller with an increasing number of iterations. Hence, it can be concluded that Algorithm 1 converges and therefore also the optimization problem.

2.6.2 ANALYSIS OF COVARIANCE MATRICES

In the second experiment, we look at how the covariance matrix of each core changes throughout the iterations in Algorithm 2. We also examine the covariance matrix of the low-rank tensor estimate, computed with the unscented transform in TT format. The experiment is performed 100 times with the same TT, \mathcal{G} , but with different priors, as in Section 2.6.1. Then, the mean of the 100 results is plotted with a region of twice the standard deviation. Figure 2.18 shows the trace and Frobenius norm of the covariance matrix of the core that will be updated next, after the norm is moved to this core. Both the trace and Frobenius norm of each core's covariance matrix decrease and converge to a fixed value. For the first and third core, the values are smaller than for the second, because the second core has a larger number of elements. The convergence behavior is also shown in Figure 2.19, where the trace and Frobenius norm of the covariance matrix of the low-rank tensor estimate converge quickly to a fixed value. The decreasing and converging values of the trace (Figure 2.17 top) and the Frobenius norm (Figure 2.17 bottom) indicate that the

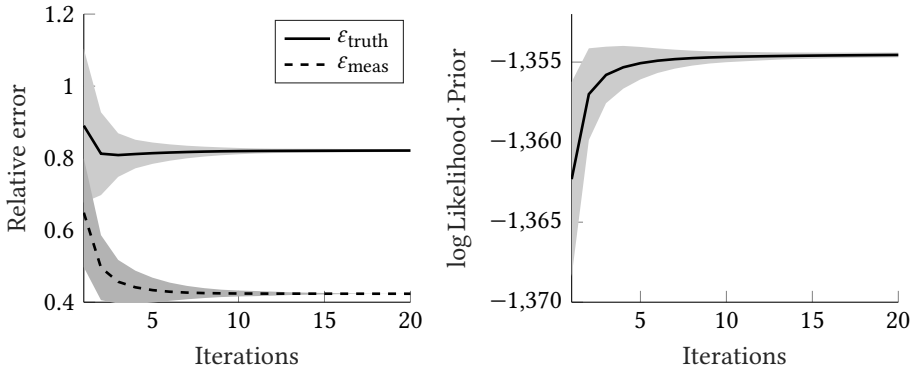


Figure 2.17: Left: Evolution of the relative errors during 20 iterations in Algorithm 1. Right: Evolution of log likelihood times the prior during 20 iterations in Algorithm 1.

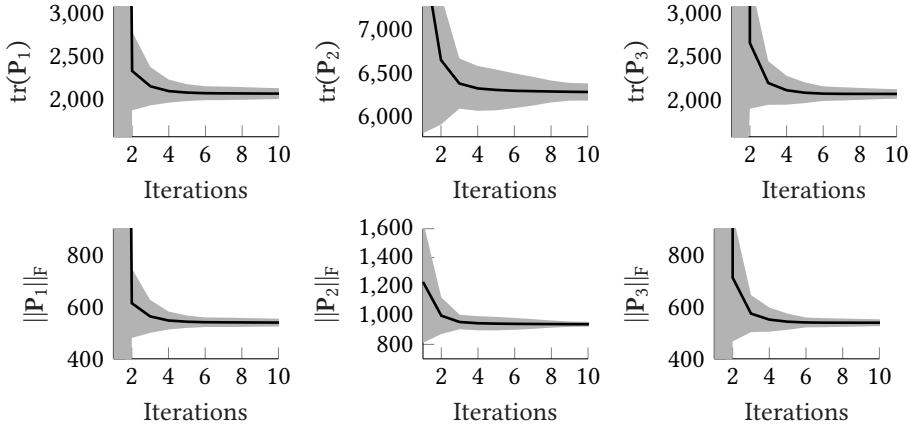


Figure 2.18: Top: Trace, bottom: Frobenius norm of covariance matrix of \mathcal{G}_1 , \mathcal{G}_2 and \mathcal{G}_3 .

uncertainty of the mean decreases and then remains constant with an increasing number of iterations. In the next experiments, we will use the information of the covariance matrices to visualize a confidence interval for our estimate.

2.6.3 COMPARISON TO CONVENTIONAL ALS

The main benefits of the ALS in a Bayesian framework are the uncertainty quantification of the low-rank tensor estimate, as well as the incorporation of prior knowledge. In the third experiment, we show the benefits by comparing the ALS in a Bayesian framework to the conventional ALS.

Figure 2.20 depicts the vectorized low-rank tensor estimate for the ALS and the mean of the ALS in a Bayesian framework's estimate with a 95% confidence interval in comparison with the ground truth. The uncertainty measure is computed from the diagonal elements of

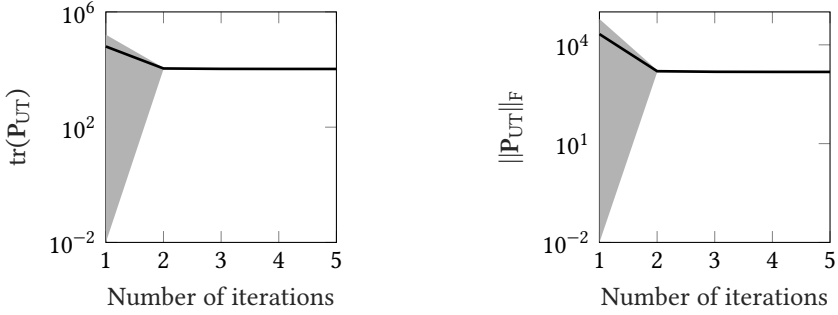


Figure 2.19: Left: Trace, right: Frobenius norm of covariance matrix of low-rank tensor estimate.

\mathbf{P}_{UT} . The top figure shows the estimate using one noisy sample \mathbf{y} for the ALS in a Bayesian framework and the bottom using 100 noisy samples. While the ALS does not improve when taking into account multiple noisy samples, the ALS in a Bayesian framework improves in two aspects. The error between the mean and the truth becomes smaller and the estimate becomes more certain. Also, in the top figure, where the uncertainty is relatively large, the ground truth almost always lies inside the confidence interval and therefore the ALS in a Bayesian framework provides more information than the conventional ALS.

Now, we analyze the influence of the prior quality on the relative error. Figure 2.21 shows the relative error $\varepsilon_{\text{truth}}$ of the ALS in a Bayesian framework for different priors. The prior mean is computed from

$$\mathbf{m}_i^0 = \mathbf{g}_{i,\text{truth}} + a \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad i = 1, 2, 3 \quad (2.34)$$

where $\mathbf{g}_{i,\text{truth}}$ denotes the vectorization of $\mathcal{G}_{i,\text{truth}}$ and a is a number that is set to values between 0 and 5. It determines how different the prior mean is from the ground truth. The prior covariance is computed from

$$\mathbf{P}_i^0 = b^2 \mathbf{I}, \quad i = 1, 2, 3 \quad (2.35)$$

by setting b to values between 0 and 5. A small value means a high certainty and a large value means a low certainty on the prior mean. Figure 2.21 shows that the error is small if the prior mean is close to the ground truth and the covariance is small. For a bad prior and a small covariance, the error is a 100 percent or larger, since a high certainty for a bad prior is assumed. For comparison, the isoline (dashed line) corresponding to the mean relative error of the conventional ALS is shown in the graph, which is almost independent of the prior information.

Figure 2.22 (left) shows the relative error of the reconstructed tensor versus the signal-to-noise ratio for a prior mean from Equation (2.34) with $a = 10^{-1}$ and prior covariance from Equation (2.35) with $b = 10^{-1}$, meaning a good prior and a high certainty on the prior. While the ALS performs poorly for high noise, the ALS in a Bayesian framework results in small relative errors. For an increasing SNR, the relative error of the ALS in a Bayesian

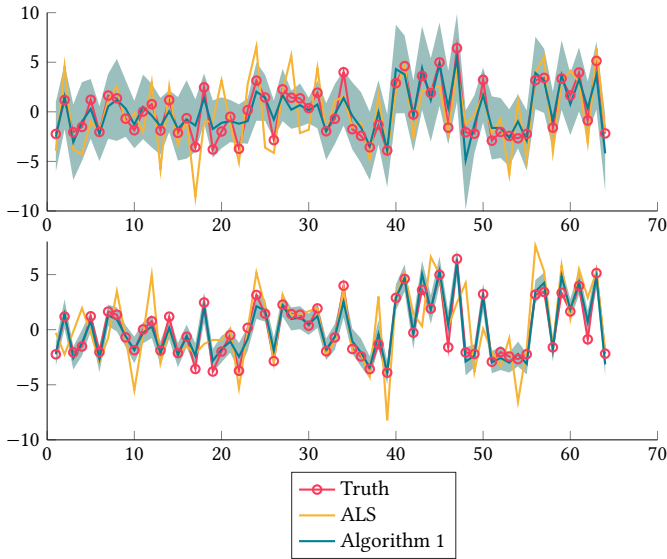


Figure 2.20: Ground truth with ALS estimate and mean of estimate from the ALS in a Bayesian framework with confidence region of 95%. Top: Estimate using one noisy sample. Bottom: Estimate using 100 noisy samples.

framework converges to the one of the ALS.

Further, Figure 2.22 (right) shows the ALS in comparison with the ALS in a Bayesian framework for multiple noisy samples. While the relative error ϵ_{truth} decreases for the ALS in a Bayesian framework, the conventional ALS does not improve when more noisy samples become available.

As shown, the ALS in a Bayesian framework gives better results if a good prior is available and it provides a measurement of the uncertainty and therefore additional valuable information. Also, if multiple noisy samples are available, ALS in a Bayesian framework significantly improves the estimate.

2.6.4 RECONSTRUCTION OF NOISY IMAGE

To test Algorithm 1 on an image processing problem, a cat image is reconstructed from an image corrupted with noise. Figure 2.23 shows the steps before applying Algorithm 1. The original image of size 256×256 pixel is reshaped into an 8-way tensor, where each mode is of dimension 4. To obtain the TT-ranks, here we use the TT-SVD algorithm [28]. It finds a TD that approximates the given tensor by setting an upper bound for the relative error. With an upper bound of 0.1, the TT-ranks, depicted in Figure 2.23 are obtained. Finally, the ground truth is computed as the vectorized contracted TT. Now, ten noisy samples are formed with Equation (2.33) with a signal-to-noise ratio of $\text{SNR}_{\text{dB}} = 0$.

Figure 2.24a) shows the original image and Figure 2.24b) the low-rank image, which

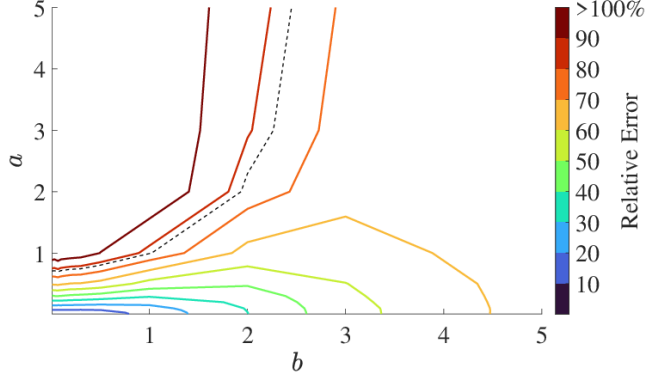


Figure 2.21: Relative error of the ALS in a Bayesian framework for different priors for $\text{SNR}_{\text{dB}} = 0$. The y-axis indicates the similarity of the prior mean to the ground truth and the x-axis indicates the certainty on the prior mean. The dashed line corresponds to the isoline corresponding to the mean error of the conventional ALS.

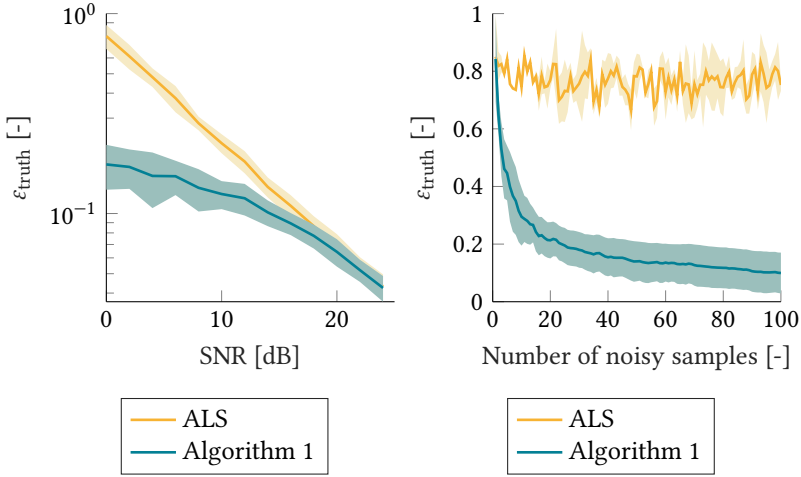


Figure 2.22: Left: Relative error $\varepsilon_{\text{truth}}$ vs. signal-to-noise ratio with prior mean from Equation (2.34) with $a = 10^{-1}$ and prior covariance from Equation (2.35) with $b = 10^{-1}$. Right: Comparison of the relative error $\varepsilon_{\text{truth}}$ between the ALS and the ALS in a Bayesian framework for different numbers of noisy samples.

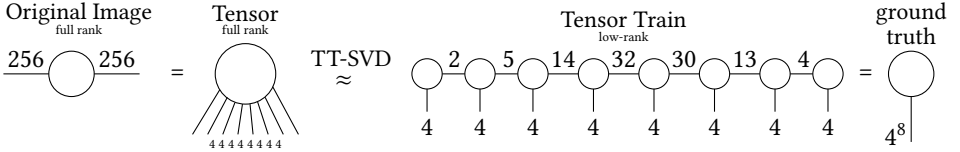


Figure 2.23: Computation of ground truth from original image: The original image of size 256×256 pixel is reshaped into an 8-way tensor, where each mode is of dimension 4. Then, the TT-SVD algorithm [28] with an upper bound for the relative error of 0.1 is applied, resulting in the depicted TT-ranks. Finally, the ground truth is obtained as the vectorized contracted TT.

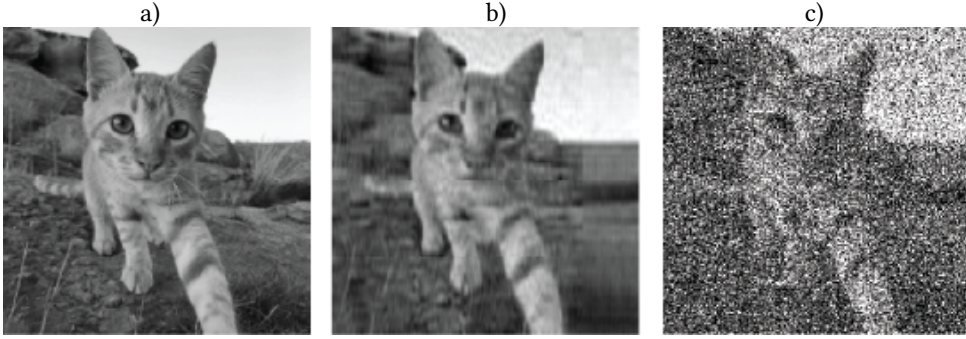


Figure 2.24: a) Original image, b) Image approximated with the TT-SVD algorithm [28] with an upper bound for the relative error of 0.1, c) One noisy sample (low-rank image corrupted with random noise).

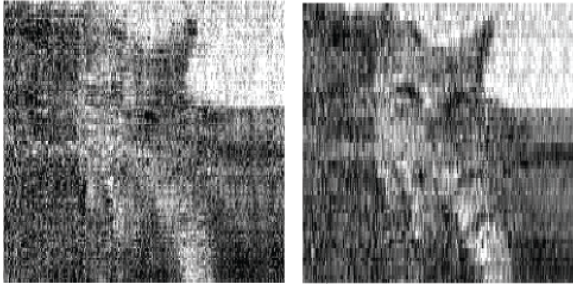


Figure 2.25: Reconstructed image with conventional ALS algorithm. Left: using one noisy sample. Right: using ten noisy samples.

is obtained by reshaping the low-rank TT from the TT-SVD into the size of the original image. Figure 2.24c) shows one exemplary noisy sample y reshaped into the dimensions of the original image. As a stopping criterion, we used the maximum number of iterations of 3. Figure 2.25 left shows the reconstruction of the image with the conventional ALS using one noisy sample and on the right using ten noisy samples. Figure 2.26 shows the reconstruction of the image inputting a random prior mean and a prior covariance on each core of $1000^2 I$ and using one and ten noisy samples. For the ALS in a Bayesian framework, it is shown that the image gets clearer with a higher number of noisy samples y , confirmed by the decreasing relative error $\varepsilon_{\text{truth}}$ from 0.3127 to 0.1478. The relative error of the conventional ALS only decreases slightly from 0.3664 to 0.3088.

2.6.5 LARGE-SCALE EXPERIMENT

In this experiment, we demonstrate that Algorithm 1 also works with larger tensors. The cat image from Section 2.6.4 in color is up-scaled via bi-cubic interpolation to obtain a $6000 \times 4000 \times 3$ tensor as depicted in Figure 2.27a). Next, we find a low-rank approximation of the image by first applying the TKPSVD algorithm [2]. The TKPSVD decomposes a

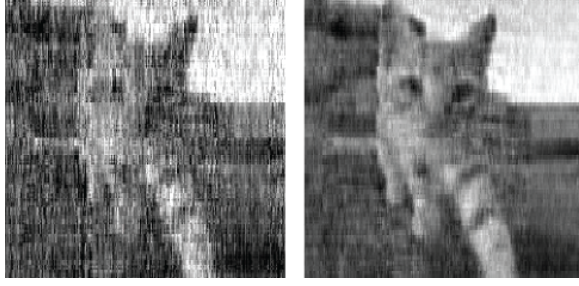


Figure 2.26: Reconstructed image with ALS in a Bayesian framework (Algorithm 1). Left: using one noisy sample. Right: using ten noisy samples.

tensor \mathcal{A} into a sum of multiple Kronecker products of N tensors $\mathcal{A}_r^{(n)}$

$$\mathcal{A} = \sum_{r=1}^R \lambda_r \mathcal{A}_r^{(N)} \otimes \dots \otimes \mathcal{A}_r^{(1)}$$

where $\lambda_r \in \mathbb{R}$. We approximate the image by taking only the term with the largest λ_r and $N = 5$,

$$\mathcal{A} \approx \lambda_{\max} \mathcal{A}_1^{(5)} \otimes \mathcal{A}_1^{(4)} \otimes \mathcal{A}_1^{(3)} \otimes \mathcal{A}_1^{(2)} \otimes \mathcal{A}_1^{(1)},$$

where $\lambda_{\max} = \lambda_1$. The resulting Kronecker products is of dimensions

$$(375 \times 250 \times 3) \otimes (2 \times 2 \times 1) \otimes (2 \times 2 \times 1) \otimes (2 \times 2 \times 1) \otimes (2 \times 2 \times 1),$$

as depicted in the top part of Figure 2.28. Secondly, $\mathcal{A}_1^{(5)} \in \mathbb{R}^{375 \times 250 \times 3}$ is further decomposed with the TT-SVD algorithm with an upper bound of the relative error of 0.08, where the dimensions are factorized as shown in the lower part of Figure 2.28. The resulting low-rank approximation of the image is shown in Figure 2.27b) and the noisy image, created with an $\text{SNR} = -22$, is shown in Figure 2.27c). We use Algorithm 1 with a random prior mean and $\mathbf{P}_i^0 = 10000^2 \mathbf{I}$. Figure 2.27d) shows the reconstructed image after 30 iterations in Algorithm 1. The main computational bottleneck is the inversion of the covariance matrix of the largest TD component (line 4 of Algorithm 1). Thus, the number of elements of a TD component, dependent on its ranks, is the limiting factor for the computational complexity. In this case, the largest TT-core has $13 \cdot 25 \cdot 18 = 5850$ elements, see second last TT-core in the lower part of Figure 2.28.

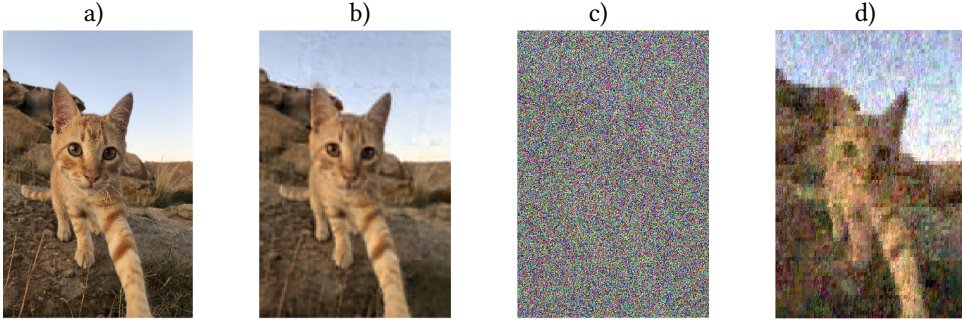


Figure 2.27: a) Original image, b) Low-rank image, c) Noisy image (low-rank image corrupted with random noise, with an $\text{SNR} = -22$), d) Reconstructed image.

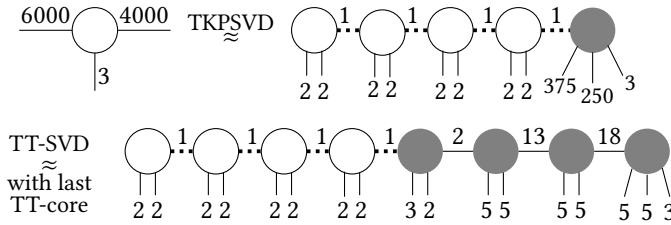


Figure 2.28: Determination of the TT-ranks by computing a low-rank decomposition with the TKPSVD and then decomposing the last TT-core (gray) further with the TT-SVD.

2.7 CONCLUSIONS

We approached the computation of low-rank tensor decomposition from a Bayesian perspective. Assuming Gaussian priors for the TD components and Gaussian measurement noise and by applying a block coordinate descent, we were able to perform a tractable inference and compute the posterior joint distribution of the TD components. This leads to a probabilistic interpretation of the ALS. The distribution of the underlying low-rank tensor was computed with the unscented transform in tensor train format. We found that the relative error of the resulting low-rank tensor approximation depends strongly on the quality of the prior distribution. In addition, our method opens up for a recursive estimation of a tensor from a sequence of noisy measurements of the same underlying tensor. If no useful prior information is available, the method gives the same result as the conventional ALS. Our method will perform worse than the conventional ALS, if a small covariance is assumed for a bad prior mean. Future work could focus on incorporating the inference of the ranks which for the ALS are fixed and therefore need to be decided beforehand. Also, the method could be extended to a non-Gaussian prior and the UT algorithm could be further developed, e.g. by parallelizing the code to make it computationally more efficient for large data sets.

REFERENCES

- [1] Kim Batselier, Zhongming Chen, and Ngai Wong. Tensor network alternating linear scheme for MIMO Volterra system. *Automatica*, 84:26–35, 2017.
- [2] Kim Batselier and Ngai Wong. A constructive arbitrary-degree Kronecker product. *Numerical Linear Algebra with Applications*, 24(5):1–17, 2017.
- [3] Cesar F Caiafa and Andrzej Cichocki. Stable, robust, and super fast reconstruction of tensors using multi-way projections. *IEEE Transactions on Signal Processing*, 63(3):780–793, 2015.
- [4] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multi-dimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.
- [5] Cong Chen, Kim Batselier, Ching-Yun Ko, and Ngai Wong. A Support Tensor Train Machine. In *Proceedings of the International Joint Conference on Neural Networks*, number July, pages 1–8. IEEE, 2019.
- [6] Lei Cheng, Yik-Chung Wu, and H Vincent Poor. Probabilistic Tensor Canonical Polyadic Decomposition With Orthogonal Factors. *IEEE Transactions on Signal Processing*, 65(3):663–676, 2017.
- [7] Wei Chu and Zoubin Ghahramani. Probabilistic models for incomplete multi-dimensional arrays. *Journal of Machine Learning Research*, 5(2006):89–96, 2009.
- [8] Andrzej Cichocki, Namgil Lee, Ivan V Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P Mandic. Tensor networks for dimensionality reduction and large-scale optimization Part 1 low-rank tensor decompositions. *Foundations and Trends in Machine Learning*, 9(4-5):249–429, 2016.
- [9] Andrzej Cichocki, Anh-Huy Phan, Qibin Zhao, Namgil Lee, Ivan V Oseledets, Masashi Sugiyama, and Danilo P Mandic. Tensor networks for dimensionality reduction and large-scale optimizations: Part 2 applications and future perspectives. *Foundations and Trends in Machine Learning*, 9(6):431–673, 2017.
- [10] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. *Journal of Machine Learning Research*, 49(June):698–728, 2016.
- [11] Pierre Comon, Xavier Luciani, and André de Almeida. Tensor Decompositions, Alternating Least Squares and other Tales. *Journal of Chemometrics*, 23:393–405, 2009.
- [12] Sergey V Dolgov and Dmitry V Savostyanov. Alternating Minimal Energy Methods for Linear Systems in Higher Dimensions. *SIAM Journal on Scientific Computing*, 36(5):A2248–A2271, 2014.
- [13] Gérard Favier, Alain Y Kibangou, and Thomas Bouilloc. Nonlinear system modeling and identification using Volterra-PARAFAC models. *International Journal of Adaptive Control and Signal Processing*, 26:30–53, 2012.

- [14] Lars Grasedyck, Melanie Kluge, and Sebastian Krämer. Variants of Alternating Least Squares Tensor Completion in the Tensor Train Format. *SIAM Journal on Scientific Computing*, 37(5):A2424–A2450, 2015.
- [15] Richard Harshman. Foundations of the PARAFAC procedure: Models and conditions for an “explanatory” multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16(10):1– 84, 1970.
- [16] Simon S Haykin. *Kalman Filtering and Neural Networks*. John Wiley & Sons, Inc., 2001.
- [17] Jasper L Hinrich and Morten Mørup. Probabilistic tensor train decomposition. In *Proceedings of the 27th European Signal Processing Conference*, 2019.
- [18] Jesper L Hinrich, Kristoffer H Madsen, and Morten Mørup. The probabilistic tensor decomposition toolbox, 2020.
- [19] Peter D Hoff. Equivariant and Scale-Free Tucker Decomposition Models. *International Society for Bayesian Analysis*, 11(3):627–648, 2016.
- [20] Sebastian Holtz and Reinhold Rohwedder, Thorsten Schneider. The Alternating Linear Scheme for Tensor Optimization in the Tensor Train Format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012.
- [21] Pavel A Izmailov, Alexander V Novikov, and Dmitry A Kropotov. Scalable Gaussian Processes with Billions of Inducing Inputs via Tensor Train Decomposition. *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 84:726–735, 2018.
- [22] Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE*, volume 92, page 1958, 2004.
- [23] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.
- [24] Clara Menzen, Manon Kok, and Kim Batselier. Alternating linear scheme in a Bayesian framework for low-rank tensor approximation. *SIAM Journal on Scientific Computing*, 44(3):A1116–A1144, 2022.
- [25] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, New York, NY, USA, second edition, 2006.
- [26] Alexander Novikov, Dmitry Podoprikin, Anton Osokin, and Dmitry Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, pages 442–450, 2015.
- [27] Ivan V Oseledets. Approximation of 2d x 2d Matrices using Tensor Decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145, 2010.
- [28] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

- [29] Ivan V Oseledets and Sergey V Dolgov. Solution of Linear Systems and Matrix Inversion in the TT-format. *SIAM Journal on Scientific Computing*, 34(5):A2718–A2739, 2012.
- [30] Piyush Rai, Yingjian Wang, and Lawrence Carin. Leveraging features and networks for probabilistic tensor decomposition. In *Proceedings of the National Conference on Artificial Intelligence*, volume 4, pages 2942–2948, 2015.
- [31] Piyush Rai, Yingjian Wang, Shengbo Guo, Gary Chen, David Dunson, and Lawrence Carin. Scalable Bayesian low-rank decomposition of incomplete multiway tensors. In *Proceedings of the 31st International Conference on Machine Learning*, volume 5, pages 3810–3820, 2014.
- [32] Thorsten Rohwedder and André Uschmajew. On local convergence of alternating schemes for optimization of convex problems in the tensor train format. *SIAM Journal on Numerical Analysis*, 51(2):1134–1162, 2013.
- [33] Simo Särkkä. *Bayesian filtering and smoothing*. Cambridge University Press, 2013.
- [34] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.
- [35] 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*, 65(13):3551–3582, 2017.
- [36] Marco Signoretto, Lieven De Lathauwer, and Johan A K Suykens. A kernel-based framework to tensorial data analysis. *Neural Networks*, 24(8):861–874, 2011.
- [37] Qingquan Song, Hancheng Ge, James Caverlee, and Xia Hu. Tensor completion algorithms in big data analytics. *ACM Transactions on Knowledge Discovery from Data*, 13(1), 2019.
- [38] Ledyard R Tucker. Some Mathematical Notes on Three-Mode Factor Analysis. *Psychometrika*, 31(3):279–311, 1966.
- [39] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *Proceedings of the 10th SIAM International Conference on Data Mining*, pages 211–222, 2010.
- [40] Zenglin Xu, Feng Yan, and Yuan Qi. Bayesian nonparametric models for multiway data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):475–487, 2015.
- [41] Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. A tensor-variate Gaussian process for classification of multidimensional structured data. *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013*, pages 1041–1047, 2013.

- [42] Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. Bayesian CP factorization of incomplete tensors with automatic rank determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1751–1763, 2015.
- [43] Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. Bayesian Sparse Tucker Models for Dimension Reduction and Tensor Completion. *ArXiv ID: 1505.02343*, pages 1–13, 2015.
- [44] Qibin Zhao, Guoxu Zhou, Liqing Zhang, Andrzej Cichocki, and Shun Ichi Amari. Bayesian Robust Tensor Factorization for Incomplete Multiway Data. *IEEE Transactions on Neural Networks and Learning Systems*, 27(4):736–748, 2016.

3

TENSOR NETWORK SQUARE ROOT KALMAN FILTER FOR ONLINE GAUSSIAN PROCESS REGRESSION

The state-of-the-art tensor network Kalman filter lifts the curse of dimensionality for high-dimensional recursive estimation problems. However, the required rounding operation can cause filter divergence due to the loss of positive definiteness of covariance matrices. We solve this issue by developing, for the first time, a tensor network square root Kalman filter and applying it to high-dimensional online Gaussian process regression. In our experiments, we demonstrate that our method is equivalent to the conventional Kalman filter when choosing a full-rank tensor network. Furthermore, we apply our method to a real-life system identification problem where we estimate 4^{14} parameters on a standard laptop. The estimated model outperforms the state-of-the-art tensor network Kalman filter in terms of prediction accuracy and uncertainty quantification.

3.1 INTRODUCTION

In a time when data-driven AI models are trained on an exponentially growing amount of data, it is crucial that the models can be adapted to newly observed data without retraining from scratch. These online or recursive settings are present in many fields, including system identification [4, 8], sensor fusion [33, 38], robotics [17, 19], and machine learning [12, 24, 35].

While Bayesian algorithms, like widely-used Gaussian processes (GPs) [25] are well-suited for an online setting, they are associated with potentially high computational costs. Standard GP regression using a batch of N observations has a cubic cost in N , i.e., $\mathcal{O}(N^3)$. The number of observations is growing in an online setting, so the cost increases each time step and can become a computational bottleneck.

There are numerous parametric approximations to address scalability in batch settings, including sparse GPs [23] and reduced-rank GPs [34], which both have a complexity of $\mathcal{O}(NM^2)$, M being the number of inducing inputs and basis function for the respective method. Structured kernel interpolation for sparse GPs [40] reduces the complexity further to $\mathcal{O}(N + DM^{1+1/D})$, D being the number of input dimensions.

Parametric approximations allow for a straightforward recursive update, where the posterior distribution from the previous time step is used as a prior for the current time step [28]. In this context, online GPs have been used, e.g., for GP state-space models [6, 27, 36], rank-reduced Kalman filtering [29] and recursive sparse GPs [35].

In this contribution to the thesis, we consider the online parametric GP model given by

$$\begin{aligned} y_t &= \boldsymbol{\phi}(\mathbf{x}_t)^\top \mathbf{w}_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma_y^2), \\ \mathbf{w}_{t-1} &\sim \mathcal{N}(\hat{\mathbf{w}}_{t-1}, \mathbf{P}_{t-1}), \end{aligned} \quad (3.1)$$

where y_t is a scalar observation at discrete time t , $\boldsymbol{\phi}(\cdot)$ are basis functions that map a D dimensional input vector \mathbf{x}_t to a feature space, $\mathbf{w}_t \in \mathbb{R}^M$ are the parameters at time t , and σ_y^2 denotes the variance of the measurement noise ϵ_t which is assumed to be i.i.d. and zero-mean Gaussian. With (3.1), the posterior distribution $p(\mathbf{w}_t | \mathbf{x}_{1:t}, \mathbf{y}_{1:t}) = \mathcal{N}(\hat{\mathbf{w}}_t, \mathbf{P}_t)$ — i.e., the distribution of \mathbf{w}_t given all inputs and measurements up until time t , $\mathbf{x}_{1:t} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t]$, $\mathbf{y}_{1:t} = [y_1, y_2, \dots, y_t]$ — is computed at each time step using the estimate $\hat{\mathbf{w}}_{t-1}$ and covariance matrix \mathbf{P}_{t-1} from the previous time step as a prior.

We consider commonly used product kernels with a feature map given by

$$\boldsymbol{\phi}(\mathbf{x}_t) = \boldsymbol{\phi}^{(1)}(\mathbf{x}_t) \otimes \dots \otimes \boldsymbol{\phi}^{(d)}(\mathbf{x}_t) \otimes \dots \otimes \boldsymbol{\phi}^{(D)}(\mathbf{x}_t), \quad (3.2)$$

where $\boldsymbol{\phi}^{(d)}(\mathbf{x}_t) \in \mathbb{R}^I$ with I being the number of basis functions in the d th dimension, and \otimes denoting a Kronecker product. The resulting number of basis functions is $M = I^D$, growing exponentially with the input dimension D . Requiring exponentially many parameters in a high-dimensional setting is a known problem, discussed in the related literature: In [36], separable kernels or a radial basis function expansion are proposed as an alternative with the disclaimer of limiting the space of functions that is possible to describe. In [35], dimensionality reduction is applied for all experiments with $D > 3$. Alternatively, several tensor network (TN)-based methods have been proposed to break this curse of dimensionality and achieve a linear computational complexity in D . In the batch setting, [3] and [39] give solutions for the squared exponential and polynomial kernel, respectively. In

the online setting, the state-of-the-art method is the tensor network Kalman filter (TNKF) [4, 5], where the Kalman filter time and measurement update are implemented in TN format.

While the TNKF lifts the curse of dimensionality, it has a significant drawback. The TNKF requires a TN-specific rounding operation [22], which can result in covariance update losing positive (semi-) definiteness [7], resulting in the divergence of the filter.

This contribution to the thesis resolves this issue by computing the square root covariance factor in tensor train (TT) format instead. Our approximation represents the $M \times M$ square root covariance factor as a tensor train matrix (TTm). This is motivated by prior square root covariance factors of product kernels having a Kronecker product structure, which corresponds to a rank-1 TTm. In addition, work by [20] and [14] approximates the covariance matrix as a rank-1 TTm. This work generalizes the rank-1 approximation to higher ranks which results in better prediction accuracy and uncertainty quantification. We call our method the tensor network square root Kalman filter (TNSRKf).

We show in experiments that the TNSRKf is equivalent to the standard Kalman filter when choosing full-rank TTs. In addition, we show how different choices of TT-ranks affect the performance of our method. Finally, we compare the TNSRKf to the TNKF in a real-life system identification problem with 4^{14} parameters and observe that, contrary to the TNKF, our method does not diverge.

3.2 PROBLEM FORMULATION

Similar to the TNKF, we build on standard equations for the measurement update of the Kalman filter, given by

$$\mathbf{S}_t = \boldsymbol{\phi}_t^\top \mathbf{P}_{t-1} \boldsymbol{\phi}_t + \sigma_y^2 \quad (3.3)$$

$$\mathbf{K}_t = \mathbf{P}_{t-1} \boldsymbol{\phi}_t \mathbf{S}_t^{-1} \quad (3.4)$$

$$\hat{\mathbf{w}}_t = \hat{\mathbf{w}}_{t-1} + \mathbf{K}_t (y_t - \boldsymbol{\phi}_t^\top \hat{\mathbf{w}}_{t-1}) \quad (3.5)$$

$$\mathbf{P}_t = (\mathbf{I}_M - \mathbf{K}_t \boldsymbol{\phi}_t^\top) \mathbf{P}_{t-1} (\mathbf{I}_M - \mathbf{K}_t \boldsymbol{\phi}_t^\top)^\top + \sigma_y^2 \mathbf{K}_t \mathbf{K}_t^\top, \quad (3.6)$$

where \mathbf{S}_t denotes the innovation covariance and \mathbf{K}_t denotes the Kalman gain. Note that for a scalar measurement, \mathbf{S}_t is a scalar and \mathbf{K}_t a vector, whereas in the case of multiple measurements per time step, they are matrices. Without the loss of generality, we present the scalar case, where, beyond the scope of this contribution to the thesis, our approach can easily be extended to vector measurements. We recursively update the posterior distribution of the parametric weights from (3.1), i.e., $p(\mathbf{w}_t | \mathbf{x}_{1:t}, \mathbf{y}_{1:t})$. For product kernels with a feature map given in (3.2), it is $\mathbf{w}_t \in \mathbb{R}^{I^D}$ and $\mathbf{P}_t \in \mathbb{R}^{I^D \times I^D}$. In this case, the Kalman filter suffers from the curse of dimensionality.

The first tensor-based Kalman filter, the TNKF [4], solved the curse of dimensionality and implements (3.3)-(3.6) in TT format, where the weights are represented as a TT and the covariance matrix as a TTm. During the updates, the algebraic operations in TT format increase the TT-ranks of the involved variables, according to [5, Lemma 2]. To counteract the rank increase and keep the algorithm efficient, the TNKF requires an additional step called TT-rounding [22]. This SVD-based operation transforms the TT or TTm to ones with smaller TT-ranks. TT-rounding can result, however, in the loss of positive (semi-) definiteness.

To avoid this issue, we implement the square root formulation of the Kalman filter (SRKF), as described e.g. in [11, Ch. 7], in TT format. The SRKF expresses (3.3)-(3.6) in terms of a square root decomposition $\mathbf{P}_t = \mathbf{L}_t \mathbf{L}_t^\top$, with the square root covariance factor \mathbf{L}_t given by

$$\mathbf{L}_t = [(\mathbf{I}_M - \mathbf{K}_t \boldsymbol{\phi}_t^\top) \mathbf{L}_{t-1} \quad \sigma_y \mathbf{K}_t]. \quad (3.7)$$

In each update, (3.7) is computed by concatenating two matrices, such that the number of columns of \mathbf{L}_t increases. For the next update, \mathbf{L}_t needs to be transformed back to its original size. In the SRKF, this is done by computing a thin QR-decomposition [10, p. 248] of \mathbf{L}_t given by

$$\underbrace{\mathbf{L}_t^\top}_{(M+1) \times M} = \underbrace{\mathbf{Q}_t}_{(M+1) \times M} \underbrace{\mathbf{R}_t}_{M \times M} \quad (3.8)$$

and replacing \mathbf{L}_t by \mathbf{R}_t^\top , i.e., by the transpose of \mathbf{R}_t .

The orthogonal \mathbf{Q}_t -factor can be discarded since

$$\mathbf{P}_t = \mathbf{L}_t \mathbf{L}_t^\top = \mathbf{R}_t^\top \underbrace{\mathbf{Q}_t^\top \mathbf{Q}_t}_{\mathbf{I}_M} \mathbf{R}_t = \mathbf{R}_t^\top \mathbf{R}_t. \quad (3.9)$$

In TT format, performing the QR-decomposition as in (3.8) is not possible. We solve this issue by proposing an SVD-based algorithm in TT format that truncates \mathbf{L}_t back to its original size.

3.3 BACKGROUND ON TENSOR NETWORKS

3.3.1 TENSOR NETWORKS

Tensor networks (TNs), also called tensor decompositions, are an extension of matrix decompositions to higher dimensions. There are multiple TN architectures, including the CANDECOMP/PARAFAC decomposition [16], the Tucker decomposition [37], and the tensor train (TT) decomposition [22]. In this contribution to the thesis, we focus on TTs to approximate the weight vector's mean as discussed in Section 3.3.1, and a TT matrix (TTm) [21] to approximate the square root covariance factor, as discussed in Section 3.3.1.

In this context, we denote TTs representing vectors as a lower-case bold letter, e.g. \mathbf{w}_t , and their components, called TT-cores, as capital calligraphic bold letters, e.g. $\mathcal{W}^{(d)}$. TT matrices are denoted by upper-case bold letters, e.g. \mathbf{L}_t and their corresponding TTm-cores as capital calligraphic bold letters, e.g. $\mathcal{L}^{(d)}$.

TENSOR TRAIN VECTORS

As depicted in Fig. 3.1(a), a TT vector consists of interconnected three-way tensors, called TT-cores, visualized as nodes with three edges. Each edge corresponds to an index of a TT-core and connected edges are summations over the involved indices. Each TT-core is connected by two edges, called TT-ranks, to its neighbouring TT-cores, except for the first and last TT-core, whose outer TT-ranks are by definition equal to one.

For the purpose of this contribution to the thesis, consider a TT that represents the mean of the weight vector $\mathbf{w}_t \in \mathbb{R}^M$. The TT-cores, denoted by $\mathcal{W}_t^{(1)}, \dots, \mathcal{W}_t^{(d)}, \dots, \mathcal{W}_t^{(D)}$ with $\mathcal{W}_t^{(d)} \in \mathbb{R}^{R_d \times I \times R_{d+1}}$ for $d = 1, \dots, D$, where R_d and R_{d+1} are the TT-ranks and I is the

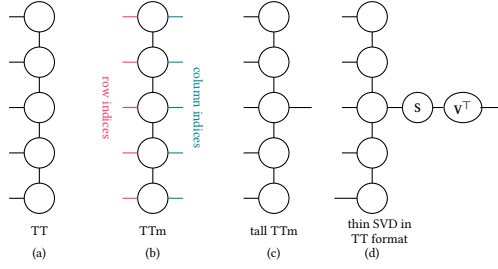


Figure 3.1: Visual depiction of tensor diagrams for a (a) TT, (b) TTm, (c) tall TTm and (d) thin SVD.

size of the non-connected edge such that $M = I^D$. By definition $R_1 = R_{D+1} = 1$. Without the loss of generality, we use TT-cores with equal TT-ranks R_w . The storage complexity of \mathbf{w}_t without TNs is $\mathcal{O}(I^D)$ and in TT format $\mathcal{O}(DIR_w^2)$, where lower TT-ranks R_w will result in more efficient representations.

An important characteristic of a TT for numerical stability is that it can be transformed into the site- d -mixed canonical format.

Definition 3.3.1 Site- d -mixed canonical format [30] A $\text{TT } \mathbf{w}_t$ in site- d -mixed canonical format is given by

$$\mathbf{w}_t = \mathbf{G}_{d,t} \mathbf{w}_t^{(d)}, \quad (3.10)$$

where $\mathbf{G}_{d,t} \in \mathbb{R}^{M \times R_w I R_w}$ is an orthogonal matrix computed from all TT-cores except the d th and $\mathbf{w}_t^{(d)} \in \mathbb{R}^{R_w I R_w}$ is the vectorization of the d th TT-core. In this format, the TT representation is linear in the d th TT-core when all other TT-cores are fixed.

TT MATRICES AND TALL TT MATRICES

A TTm consists of interconnected four-way tensors, as depicted in Fig. 3.1(b). Analogous to the TT, the TTm components and connected edges are called TTm-cores and TTm-ranks, respectively, where each TTm-core has two free edges, the row and column indices.

For the purpose of this contribution to the thesis, consider a TTm representation of the square root covariance factor $\mathbf{L}_t \in \mathbb{R}^{M \times M}$. The TTm-cores are denoted by $\mathcal{L}_t^{(1)}, \dots, \mathcal{L}_t^{(d)}, \dots, \mathcal{L}_t^{(D)}$ with $\mathcal{L}_t^{(d)} \in \mathbb{R}^{R_d \times I \times J \times R_{d+1}}$, where I and J are the number of row and column indices, indicated in Fig. 3.1(b) as red and blue edges respectively, such that $M = I^D$ and $M = J^D$. By definition, $R_1 = R_{D+1} = 1$, and for this contribution to the thesis, we generally assume that all other TTm-ranks $R_2 = \dots = R_D = R_L$ are equal. The storage complexity of \mathbf{L}_t without TNs is $\mathcal{O}(I^D \times I^D)$ and in TTm format $\mathcal{O}(DR_L^2 I J)$.

A TTm can also be written in terms of the site- d -mixed canonical format as defined in Definition 3.3.1, but it requires to be transformed into a TT first. This can be done by combining the row and column indexes into one index, which represents a kind of vectorization of the matrix represented by the TTm. Note, however, that the indices are not ordered as in conventional vectorization. A site- d -mixed canonical format of a TTm is given by

$$\text{vec}(\mathbf{L}_t) = \mathbf{H}_{d,t} \mathbf{l}_t^{(d)}, \quad (3.11)$$

where the orthogonal matrix $\mathbf{H}_{d,t} \in \mathbb{R}^{2M \times R_L I J R_L}$ is computed from all the TTm-cores but the d th, and $\mathbf{l}_t^{(d)} \in \mathbb{R}^{R_L I J R_L}$.

To recompute \mathbf{L}_t in its original size in the QR step of the SRKF (see (3.8)), here called the re-squaring step, we need a special case of a TTm, the tall TTm, as well as a thin SVD in TTm format.

Definition 3.3.2 Tall TTm [3] A tall TTm, as depicted in Figure 3.1(c), has only one TTm-core with both a row and column index, while all other TTm-cores have only row indices. Then, the TTm represents a tall matrix with many more rows than columns.

Definition 3.3.3 Thin SVD in TTm format [2] Consider a TTm in site- d -mixed canonical format, where the d th TTm-core is the one that has the column index, $\mathcal{L}^{(d)} \in \mathbb{R}^{R_L \times I \times J \times R_L}$. The SVD of $\mathcal{L}^{(d)}$ reshaped and permuted in to a matrix of size $R_L I R_L \times J$, is given by

$$\mathbf{U}^{(d)} \mathbf{S}^{(d)} (\mathbf{V}^{(d)})^\top. \quad (3.12)$$

Now replace the d th TTm-core by $\mathbf{U}^{(d)}$ reshaped and permuted back to the original TTm-core dimensions.

Then the thin SVD is given by the TTm with the replaced TT-core as the orthogonal U-factor, and $\mathbf{S}^{(d)} (\mathbf{V}^{(d)})^\top$ as the \mathbf{SV}^\top -factors, as depicted in Fig. 3.1(d).

3.4 TNSRKF

We propose our method, combining efficient TN methods with the SRKF formulation for online GP regression. More specifically, we recursively compute the posterior distribution of the parametric weights in (3.1) from the measurement update of the Kalman filter. To achieve this, we update the mean $\hat{\mathbf{w}}_t \in \mathbb{R}^M$ as a TT (Section 3.4.1), and the square root factor $\mathbf{L}_t \in \mathbb{R}^{M \times M}$ as a TTm (Section 3.4.2).

All computations are summarized in Algorithm 4, which outputs the posterior weight distributions $p(\mathbf{w}_t | \mathbf{x}_{1:t}, \mathbf{y}_{1:t}) = \mathcal{N}(\hat{\mathbf{w}}_t, \mathbf{P}_t)$, and the prediction for a test input $f_{*,t}$ in terms of a distribution $p(f_{*,t}) = \mathcal{N}(m_{*,t}, \sigma_{*,t}^2)$ with predictive mean $m_{*,t}$ and variance $\sigma_{*,t}^2$. Note that online GP regression refers to ingesting one measurement at a time and updating the weights \mathbf{w}_t recursively. Therefore, in a truly online scenario, where measurements are collected on the fly, the input to Algorithm 4 would not be a batch \mathbf{y} , but a single measurement y_t .

3.4.1 UPDATE OF WEIGHT MEAN

The mean of the weights is updated with a new measurement $y_t \in \mathbb{R}$, with (3.5). In the original tensor-based KF [4], the two terms in equation (3.5) are summed together in TT format, which increases the TT-ranks. To avoid this rank increase and application of TT-rounding, we propose solving an optimization problem to compute (3.5) instead: We apply a commonly-used optimization algorithm from the tensor community, called the alternating linear scheme (ALS) [13, 26]. The ALS computes a TT by updating one TT-core at a time while keeping all other TT-cores fixed. The optimization problem to be solved is given by

$$\begin{aligned} \min_{\mathbf{w}_t} & \|\hat{\mathbf{w}}_{t-1} + \mathbf{K}_t(y_t - \boldsymbol{\phi}_t^\top \hat{\mathbf{w}}_{t-1}) - \mathbf{w}_t\|^2 \\ \text{s.t. } & \mathbf{w}_t \text{ being a low-rank TT,} \end{aligned} \quad (3.13)$$

where $\hat{\mathbf{w}}_{t-1}$ is the estimate from the last time step, playing now the role of the prior for the current time step.

Inserting (3.10) in (3.13), thus making use of $\mathbf{G}_{d,t}$ being an orthogonal matrix (see the site- d -mixed canonical format from Definition 3.3.1), gives the optimization problem for the update of one TT-core

$$\min_{\mathbf{w}_t^{(d)}} \left\| \mathbf{G}_{d,t}^\top (\hat{\mathbf{w}}_{t-1} + \mathbf{K}_t(y_t - \boldsymbol{\phi}_t^\top \hat{\mathbf{w}}_{t-1})) - \mathbf{w}_t^{(d)} \right\|^2. \quad (3.14)$$

In one so-called sweep of the ALS, (3.14) is solved for each TT-core once. A stopping criterion for the convergence of the residual in (3.14) determines the total number of sweeps.

3.4.2 UPDATE OF SQUARE ROOT COVARIANCE FACTOR

To compute the covariance matrix with the standard covariance update in the measurement update, see (3.6), we recursively compute the square root covariance factor \mathbf{L}_t as defined in (3.7) such that $\mathbf{P}_t = \mathbf{L}_t \mathbf{L}_t^\top$. To achieve this, we use the ALS to solve (3.7) (ALS step) and then we transform \mathbf{L}_t as in (3.8) back to its original size (re-squaring step).

ALS step In this step, we use the ALS to compute a TTm representing \mathbf{L}_t . We solve the optimization problem given by

$$\min_{\mathbf{L}_t} \left\| \begin{bmatrix} (\mathbf{I}_M - \mathbf{K}_t \boldsymbol{\phi}_t^\top) \mathbf{L}_{t-1} & \sigma_y \mathbf{K}_t \end{bmatrix} - \mathbf{L}_t \right\|_F^2 \quad (3.15)$$

s.t. \mathbf{L}_t being a low-rank TTm,

where \mathbf{L}_{t-1} is the estimated square root covariance factor from time step $t-1$ now serving as the prior. The original ALS algorithm is defined for TTs, so we must adapt it for TT matrices.

For this, it is necessary to use the site- d -mixed canonical form for TT matrices, as described in Section 3.3.1 above (3.11). In addition, we need to horizontally concatenate two matrices in TTm format, which can be done by summing two matrices of size $M \times 2M$ such that (3.15) becomes

$$\min_{\mathbf{l}_t^{(d)}} \left\| \mathbf{H}_{d,t}^\top \text{vec} \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \otimes (\mathbf{I}_M - \mathbf{K}_t \boldsymbol{\phi}_t^\top) \mathbf{L}_{t-1} \right) + \mathbf{H}_{d,t}^\top \text{vec} \left(\begin{bmatrix} 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & \mathbf{0}_{M-1} \end{bmatrix} \otimes \sigma_y \mathbf{K}_t \right) - \mathbf{l}_t^{(d)} \right\|_F^2, \quad (3.16)$$

where vec denotes the vectorization of the involved TT matrices.

Re-squaring step The optimization problem given by (3.15) requires concatenating a matrix with a column vector. In TT format, this results in a TTm of size $M \times 2M$. For the TTm-cores of \mathbf{L}_t this means that one TTm-core, which we call the augmented core, is of size $R_L \times I \times 2J \times R_L$. Before serving as a prior for the next time step, a re-squaring step implementing the QR step (see (3.8)) in TN format is required to transform \mathbf{L}_t back to

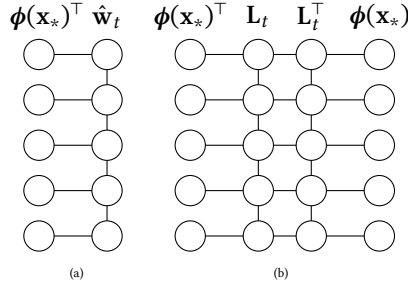


Figure 3.2: Visual depiction of (a) predictive mean and (b) predictive covariance for $D = 5$.

its original size. Since computing a QR decomposition of a TTm is not directly possible, we present an SVD-based algorithm in TN format to transform \mathbf{L}_t of size $M \times 2M$ back to $M \times M$, as described in Algorithm 5.

3.4.3 PREDICTIONS

To perform GP predictions we compute the predictive distribution for a test output $f_{*,t} = \phi(\mathbf{x}_*)^\top \mathbf{w}_t$ with mean and variance given by

$$\begin{aligned} m_{*,t} &= \phi(\mathbf{x}_*)^\top \hat{\mathbf{w}}_t \\ \sigma_{*,t}^2 &= \phi(\mathbf{x}_*)^\top \mathbf{L}_t \mathbf{L}_t^\top \phi(\mathbf{x}_*). \end{aligned} \quad (3.17)$$

Given $\hat{\mathbf{w}}_t$ as a TT and \mathbf{L}_t as a TTm, we can compute (3.17) directly in TN format without explicitly reconstructing the mean vector and square root factor. For a test input \mathbf{x}_* , Fig. 3.2 illustrates the computation of (a) the predictive mean $m_{*,t}$, (b) the predictive covariance $\sigma_{*,t}^2$. The corresponding equation to Fig. 3.2(a) is given by

$$m_{*,t} = \sum_{r^{(2)}} \cdots \sum_{r^{(D)}} \prod_d \sum_{i^{(d)}} \phi_{i^{(d)}}^{(d)}(\mathbf{x}_*) \hat{\mathbf{w}}_{r^{(d)}, i^{(d)}, r^{(d+1)}}^{(d)},$$

where the lowercase letters in the subscript, i.e. $r^{(d)}$, $r^{(d+1)}$ and $i^{(d)}$, denote the indices of size $R^{(d)} = R^{(d+1)} = R$ and $I^{(d)} = I$, respectively. Fig. 3.2(b) can be written in a similar way. Moving forward, we provide only the TN diagrams, since the equations can become lengthy.

Algorithm 4 Online GP regression in terms of SRKF in TT format (TNSRKf)

Require: Measurements $\mathbf{y} = y_1, y_2, \dots, y_N$,
 basis functions for inputs $\phi(\mathbf{x}_t)$, $t = 1, \dots, N$,
 prior $\hat{\mathbf{w}}_0$ in TN format (Lemma 3.5.2),
 prior \mathbf{L}_0 in TN format (Lemma 3.5.4),
 noise variance σ_y^2 ,
 basis functions for prediction point $\phi(\mathbf{x}_*)$.

Ensure: $p(\mathbf{w} | \mathbf{y}_{1:t}) = \mathcal{N}(\hat{\mathbf{w}}_t, \mathbf{P}_t)$,

- 1: $p(f_* | \mathbf{y}_{1:t}) = \mathcal{N}(m_{*,t}, \sigma_{*,t}^2)$, for $t = 1, \dots, N$.
- 2: Initialize $\mathbf{w}_1 = \hat{\mathbf{w}}_0$ and \mathbf{L}_1 as a random TTm in site- d -mixed canonical format.
- 3: **for** $t = 1, \dots, N$ **do**
- 4: Compute $\hat{\mathcal{W}}_t^{(1)}, \hat{\mathcal{W}}_t^{(2)}, \dots, \hat{\mathcal{W}}_t^{(D)}$ with (3.14).
- 5: Compute $\mathcal{L}_t^{(1)}, \mathcal{L}_t^{(2)}, \dots, \mathcal{L}_t^{(D)}$ with (3.16).
- 6: Save TT and TTm from step 4 and 5 as initializations for the next time step.
- 7: Re-square \mathbf{L}_t with Algorithm 5.
- 8: Compute $m_{*,t}$ with (3.17) as depicted in Fig. 3.2(a).
- 9: Compute $\sigma_{*,t}^2$ with (3.17) as depicted in Fig. 3.2(b).
- 10: **end for**

3.5 IMPLEMENTATION

In this section, we give a detailed description of the non-straightforward TN operations to update the mean estimate $\hat{\mathbf{w}}_t$ and square root covariance factor \mathbf{L}_t as described in Algorithm 4. The leading complexities of the mean and square root covariance factor update are given in Table 3.1.

3.5.1 UPDATING $\hat{\mathbf{w}}_t$ IN TN FORMAT

In the following sections, we discuss the implementation of (3.14) for the mean update (Algorithm 4, line 4), and we describe how the mean is initialized in TT format (Algorithm 4, line 2).

Table 3.1: Computational complexities for one TT-core mean and covariance update. We denote the TT-ranks of \mathbf{K}_t by $R_{\mathbf{K}}$.

Term	Complexity
$\mathbf{G}_{d,t}^\top \hat{\mathbf{w}}_{t-1}$	$\mathcal{O}(R_{\mathbf{w}}^4 I)$
$\mathbf{G}_{d,t}^\top \mathbf{K}_t (\mathbf{y}_t - \phi_t^\top \hat{\mathbf{w}}_{t-1})$ (3.20)-(3.22)	$\mathcal{O}(R_{\mathbf{w}}^2 R_{\mathbf{K}}^2 I)$ $\mathcal{O}(R_{\mathbf{L}}^4 IJ)$

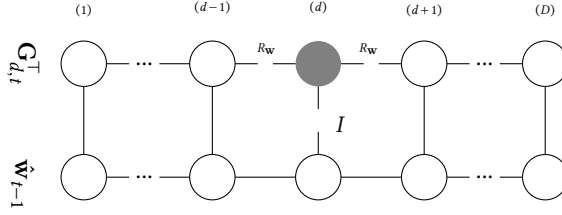


Figure 3.3: Visual depiction of computation of $\mathbf{G}_{d,t}^\top \hat{\mathbf{w}}_{t-1}$, resulting in three-way tensor of size $R_w \times I \times R_w$ (gray node). The indices are summed over from left to right, alternating between the vertical and horizontal ones.

3

IMPLEMENTATION OF $\mathbf{G}_{d,t}^\top (\hat{\mathbf{w}}_{t-1} + \mathbf{K}_t(y_t - \boldsymbol{\phi}_t^\top \hat{\mathbf{w}}_{t-1}))$

To compute the TT representing the mean estimate $\hat{\mathbf{w}}_t$, we implement the ALS to solve (3.14) (Algorithm 4, line 4).

The following example illustrates the update of one TT-core during the ALS.

Example 3.5.1 TT-core update with ALS Take a $D = 5$ dimensional weight vector in TT format with $I = 10$ basis functions in each dimension, resulting in 10^5 parameters and uniform TT-ranks of $R_2 = R_3 = R_4 = 4$. Say, we are currently updating the third TT-core $\mathcal{W}_t^{(3)} \in \mathbb{R}^{4 \times 10 \times 4}$ using

$$\underbrace{\mathbf{w}_t^{(3)}}_{160 \times 1} = \underbrace{\mathbf{G}_{3,t}^\top}_{160 \times 10^5} \left(\underbrace{\hat{\mathbf{w}}_{t-1}}_{10^5 \times 1} + \underbrace{\mathbf{K}_t}_{10^5 \times 1} \underbrace{(y_t - \boldsymbol{\phi}_t^\top \hat{\mathbf{w}}_{t-1})}_{1 \times 1} \right). \quad (3.18)$$

We first multiply over the large dimension of 10^5 in $\mathbf{G}_{3,t}^\top \hat{\mathbf{w}}_{t-1}$ and $\mathbf{G}_{3,t}^\top \mathbf{K}_t(y_t - \boldsymbol{\phi}_t^\top \hat{\mathbf{w}}_{t-1})$. In TT format, this matrix-vector multiplication is done core by core, thus avoiding the explicit multiplication. Finally, we sum two vectors of size 160.

Figure 3.3 illustrates the multiplication of $\mathbf{G}_{d,t}^\top \hat{\mathbf{w}}_{t-1}$ in TT format, resulting in a tensor $\mathcal{W}_t^{(d)}$ of size $R_w \times I \times R_w$.

The multiplication of between $\mathbf{G}_{d,t}^\top$ and $\mathbf{K}_t(y_t - \boldsymbol{\phi}_t^\top \hat{\mathbf{w}}_{t-1})$ works in the same way as depicted in Fig. 3.3, after firstly computing $\boldsymbol{\phi}_t^\top \hat{\mathbf{w}}_{t-1}$ in TN format and secondly multiplying one arbitrary TT-core of \mathbf{K}_t by the scalar $(y_t - \boldsymbol{\phi}_t^\top \hat{\mathbf{w}}_{t-1})$.

During the update of the d th TT-core, the TT is in site- d -mixed canonical format. Before updating the next TT-core, either the $(d-1)$ th or the $(d+1)$ th, the site- $(d-1)$ -mixed or site- $(d+1)$ -mixed canonical format is computed. Note that because of the recursive property, updating every TT-core once with a new measurement is usually sufficient for the residual of (3.13) to converge.

INITIALIZATION OF $\hat{\mathbf{w}}_0$ AND \mathbf{w}_1

For the first time step $t = 1$ of Algorithm 4, we choose a zero-mean assumption for the prior estimate $\hat{\mathbf{w}}_0$. The following Lemma explains how this can be implemented in TT format.

Lemma 3.5.2 Zero-mean prior in TT format [5] Consider a vector with all entries equal to zero. In TT format, such a vector is given by a TT in site- d -mixed canonical format, where the d th TT-core contains only zeros.

In addition, Algorithm 4 requires an initial guess for \mathbf{w}_1 to compute $\mathbf{G}_{d,1}$ from all TT-cores of \mathbf{w}_1 , except the d th. For this, we set $\mathbf{w}_1 = \hat{\mathbf{w}}_0$.

3.5.2 UPDATING \mathbf{L}_t IN TT FORMAT

To compute the TTm representing \mathbf{L}_t , we implement the ALS to solve (3.16) (Algorithm 4, line 5). The following example illustrates the update of one TTm-core during the ALS.

Example 3.5.3 TTm-core update with ALS Take a $D = 5$ dimensional TTm representing $\mathbf{L}_t \in \mathbb{R}^{M \times M}$, where we are currently updating the third TTm-core. We have $I = 10$ and $J = 10$, where the third TTm-core is augmented, and $R_L = 4$. We update $\mathcal{L}_t^{(3)} \in \mathbb{R}^{4 \times 10 \times 20 \times 4}$ using

$$\begin{aligned} \underbrace{\mathbf{I}_t^{(3)}}_{3200 \times 1} &= \underbrace{\mathbf{H}_{d,t}^\top}_{3200 \times 2 \cdot 10^{10}} \operatorname{vec} \left(\underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{1 \times 2} \otimes \underbrace{\mathbf{L}_{t-1}}_{10^5 \times 10^5} \right) \\ &\quad - \underbrace{\mathbf{H}_{d,t}^\top}_{3200 \times 2 \cdot 10^{10}} \operatorname{vec} \left(\underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{1 \times 2} \otimes \underbrace{\mathbf{K}_t \phi_t^\top \mathbf{L}_{t-1}}_{10^5 \times 10^5} \right) \\ &\quad + \underbrace{\mathbf{H}_{d,t}^\top}_{3200 \times 2 \cdot 10^{10}} \operatorname{vec} \left(\underbrace{\begin{bmatrix} 0 & 1 \end{bmatrix}}_{1 \times 2} \otimes \underbrace{\begin{bmatrix} 1 & \mathbf{0}_{M-1} \end{bmatrix}}_{1 \times 10^5} \otimes \underbrace{\sigma_y \mathbf{K}_t}_{10^5 \times 1} \right). \end{aligned} \quad (3.19)$$

We first multiply over the large dimension of $2 \cdot 10^{10}$ in TT format, then sum the three terms of size 3200×1 .

From Example 3.5.3, it follows that the three terms of (3.19) need to be implemented. We discuss them separately in the following sections. We distinguish between the update of the augmented TTm-core from all other ones, which result in TTm-cores of size $R_L \times I \times 2J \times R_L$ and $R_L \times I \times J \times R_L$, respectively. In the tensor diagrams (Fig. 3.4-3.6), we depict the update for the augmented TTm-core.

Before diving in, recall from (3.11) that $\mathbf{H}_{d,t}$ is computed from TTm-cores of \mathbf{L}_t , except the d th, where row and column indices are combined. In the tensor diagrams, the indices are depicted not as combined because, in practice, they are generally summed over separately. However, the vectorized format is necessary for writing down the equations in matrix form.

IMPLEMENTATION OF FIRST TERM OF (3.16)

Fig. 3.4 illustrates the computation of the augmented TTm-core in the first term of (3.16), given by

$$\mathbf{H}_{d,t}^\top \operatorname{vec} \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \otimes \mathbf{L}_{t-1} \right). \quad (3.20)$$

The column indices of \mathbf{L}_{t-1} are indicated by the round edges that are connected to the row indices of $\mathbf{H}_{d,t}^\top$. The edge containing $\mathbf{e}_1 = [1 \ 0]$ is connected to the d th TTm core of \mathbf{L}_t with a rank-1 connection, which corresponds to the Kronecker product in (3.20). The summation over the vertical and curved indices has the leading computational complexity of $\mathcal{O}(R_L^4 I J)$ per dimension. When updating all TTm-cores except the augmented TTm-core, the additional index of size 2 is summed over resulting in a tensor of size $R_L \times I \times J \times R_L$.

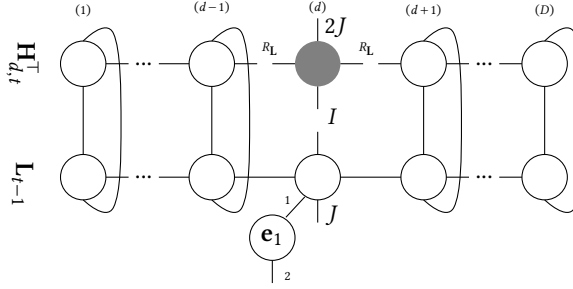


Figure 3.4: Visual depiction for computing the augmented TTm-core in (3.20) resulting in a 4-way tensor of size $R_L \times I \times 2J \times R_L$ (gray node). The combined horizontal and curved indices are summed over and alternating with the horizontal indices.

IMPLEMENTATION OF SECOND TERM OF (3.16)

Fig. 3.5 illustrates the computation of

$$\mathbf{H}_{d,t}^\top \text{vec} \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \otimes \mathbf{L}_{t-1} \mathbf{L}_{t-1}^\top \boldsymbol{\phi}_t \mathbf{S}_t^{-1} \boldsymbol{\phi}_t^\top \mathbf{L}_{t-1} \right), \quad (3.21)$$

which directly follows from the second term of (3.16). As shown, the row and column indices of $\mathbf{H}_{d,t}^\top$ are connected separately to the column and row indices of two TT matrices for \mathbf{L}_{t-1} , respectively. Like in the previous term, the edge containing $\mathbf{e}_1 = [1 \ 0]$ is connected to the augmented TTm-core of \mathbf{L}_t with a rank-1 connection, which corresponds to the Kronecker product in (3.21). The leading computational complexity of $\mathcal{O}(R_L^4 I J)$ per dimension comes from the summation over the vertical indices in the red or blue box indicated in the figure. The most efficient order of doing the computations in Fig. 3.5 was found with the visual tensor network software by [9], assuming our use case where $D > 3$ and $I, J, R_L < 10$.

IMPLEMENTATION OF THIRD TERM OF (3.16)

Fig. 3.6 illustrates the computation of

$$\mathbf{H}_{d,t}^\top \text{vec} \left(\begin{bmatrix} 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & \mathbf{0}_{M-1} \end{bmatrix} \otimes \sigma_y \mathbf{L}_t \mathbf{L}_t^\top \boldsymbol{\phi}_t \mathbf{S}_t^{-1} \right), \quad (3.22)$$

which directly follows from the third term of (3.16). The row of nodes each filled with $\mathbf{e}_1 = [1 \ \mathbf{0}_{J-1}]$ corresponds to $[1 \ \mathbf{0}_{M-1}]$ from (3.22) and their rank-1 connections to the nodes above is the second Kronecker product in (3.22), which is done dimension-wise in TT format. The node with \mathbf{e}_2 corresponds to $[0 \ 1]$ from (3.22) and its rank-1 connection is the first Kronecker product in (3.22). The summation over the vertical indices is the leading computational complexity of $\mathcal{O}(R_L^4 I J)$ per dimension.

SVD-BASED RE-SQUARING STEP IN TTm FORMAT

When computing (3.16), we double the number of columns of \mathbf{L}_t compared to \mathbf{L}_{t-1} . For the next time step, however, we need to transform \mathbf{L}_t back to its original size (Algorithm 4, line 7), otherwise its column size will grow with the iterations and slow down the algorithm. The QR step, as in (3.8), computes a full QR decomposition of \mathbf{L}_t , which cannot be done in TT format. Instead, we compute a thin SVD in TTm format (Definition 3.3.3) of \mathbf{L}_t

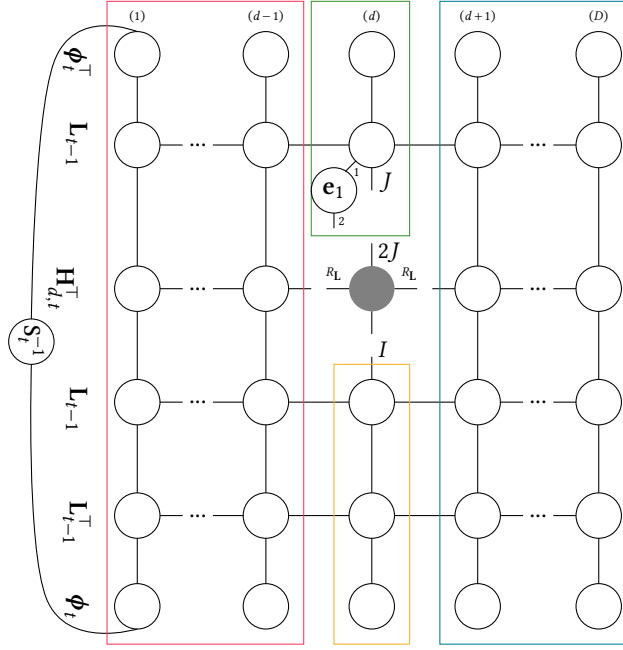


Figure 3.5: Visual depiction for computing (3.21) resulting in a 4-way tensor of size $R_L \times I \times 2J \times R_L$ (gray node). First, the indices in the red and blue boxes are summed over, then the indices between the red, yellow, and blue boxes, and finally, the ones between the red, green, and blue boxes.

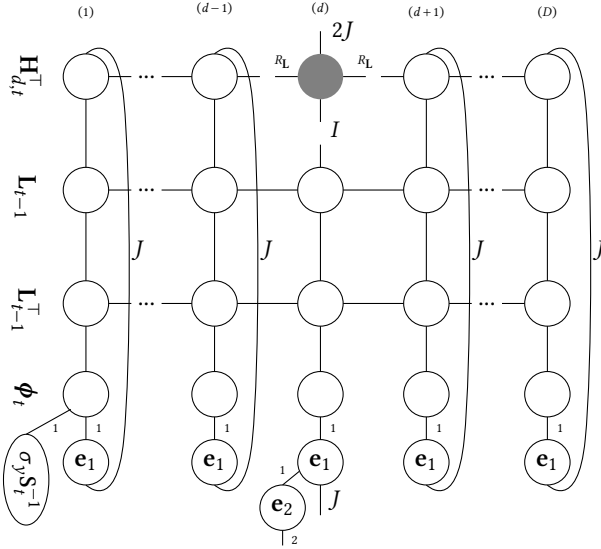


Figure 3.6: Visual depiction for computing (3.22), resulting in a 4-way tensor of size $R_L \times I \times 2J \times R_L$ (gray node). The indices are summed over from left to right by alternating between the vertical and horizontal ones.

transformed into a tall TTm (here also denoted by \mathbf{L}_t) with all row indices of size IJ , except the d th which is of size I , and the d th column index of size $2J$. The J -truncated SVD of \mathbf{L}_t is then given by

$$\underbrace{\mathbf{L}_t}_{MJ^{D-1} \times 2J} \approx \underbrace{\mathbf{U}_t \mathbf{S}_t}_{MJ^{D-1} \times J} \underbrace{\mathbf{V}_t^\top}_{J \times 2J}, \quad (3.23)$$

where $\mathbf{U}_t \mathbf{S}_t$ is the new \mathbf{L}_t and \mathbf{V}_t^\top can be discarded because of (3.9). In practice, we compute an SVD of the augmented TTm-core and truncate it back to the size of $R_L \times I \times J \times R_L$.

There is a way to make (3.23) exact. This is possible if the augmented TTm-core is of size $R_L \times I \times 2JR_L^2 \times R_L$. In this case, the SVD computed of the augmented TTm-core results in a square U-factor. Since the number of columns is doubled every measurement update, the re-squaring step can be skipped p times until $2^p = 2R_L^2$. Choosing smaller values for p reduces computational complexity at the cost of accuracy.

The SVD-based re-squaring step is described in Algorithm 5. The SVD of the reshaped and permuted augmented TTm-core is truncated for $2^p < 2R_L^2$ and exact for $2^p \geq 2R_L^2$.

Algorithm 5 SVD-based re-squaring step of covariance update

Require: TTm \mathbf{L}_t in site- d -mixed canonical format with $\mathcal{L}^{(d)} \in \mathbb{R}^{R_L \times I \times 2^p J \times R_L}$.

Ensure: TTm \mathbf{L}_t with $\mathcal{L}^{(d)} \in \mathbb{R}^{R_L \times I \times 2^{p-1} J \times R_L}$.

- 1: $\mathbf{L}^{(d)} \leftarrow$ Reshape / permute $\mathcal{L}^{(d)}$ into matrix of size $R_L I R_L \times 2^{p+1} J$.
 - 2: Compute thin SVD($\mathbf{L}^{(d)} = \mathbf{U}^{(d)} \mathbf{S}^{(d)} (\mathbf{V}^{(d)})^\top$).
 - 3: $\mathcal{L}^{(d)} \leftarrow$ Reshape / permute first $2^{p-1} J$ columns of $\mathbf{U}^{(d)} \mathbf{S}^{(d)}$ of size $R_L I R_L \times 2^{p-1} J$ into tensor of size $R_L \times I \times 2^{p-1} J \times R_L$.
-

INITIALIZATION OF \mathbf{L}_0 AND \mathbf{L}_1

At time $t = 1$, Algorithm 4 requires the prior square root covariance factor \mathbf{L}_0 in TTm format. We are considering product kernels that have priors in Kronecker format. The following Lemma describes how these types of priors can be transformed into a TTm for \mathbf{L}_0 .

Lemma 3.5.4 (Prior covariance with Kronecker structure into TTm, follows from [10, p.708]) Given a prior covariance $\mathbf{P}_0 = \mathbf{P}_0^{(1)} \otimes \mathbf{P}_0^{(2)} \otimes \dots \otimes \mathbf{P}_0^{(D)}$, the prior square root covariance in TTm format is given by a TTm with all ranks equal to 1, where the cores are given by $\mathbf{L}_0^{(1)}, \mathbf{L}_0^{(2)}, \dots, \mathbf{L}_0^{(D)}$, each reshaped into a 4-way tensor of size $1 \times I \times J \times 1$.

In addition, Algorithm 4 requires an initial guess in TTm format for $\mathbf{L}_1 \in \mathbb{R}^{M \times 2M}$. We cannot set $\mathbf{L}_1 = \mathbf{L}_0$ since the prior has TTm-ranks equal to one, and we may want higher TTm-ranks for \mathbf{L}_t . This is because the choice of the TTm-ranks of \mathbf{L}_1 determines the rank manifold on which the TTm-cores will be optimized. We initialize the TTm-cores as random samples from a zero-mean Gaussian distribution and transform the TTm into site- d -mixed canonical format, where d is the augmented TTm-core.

Table 3.2: RMSE and NLL at time $t = N$ for the full-rank setting and different choices of p in comparison to the conventional Kalman filter (KF).

Method	Setting			(RMSE) $_N$	(NLL) $_N$
KF	-			0.07873	-106.864
TNSRKf	R_w	R_L	p		
	4	16	8	0.07873	-106.864
	4	16	4	0.07879	-108.338
	4	16	2	0.07444	-157.716
	4	16	1	0.06765	-166.178

3.6 EXPERIMENTS

In this section, we show how our method works in practice by performing online GP regression on synthetic and real-life data sets. We evaluate our predictions based on the root mean square error (RMSE) for the accuracy of the mean and negative log-likelihood (NLL) for the uncertainty estimation. The metrics after t measurement updates are defined as

$$\begin{aligned}
 (\text{RMSE})_t &= \sqrt{\sum_{i=1}^{N_*} \frac{(m_{*,t,i} - y_{*,i})^2}{N_*}} \quad \text{and} \\
 (\text{NLL})_t &= 0.5 \sum_{i=1}^{N_*} \log(2\pi\sigma_{*,t,i}^2) + \frac{(m_{*,t,i} - y_{*,i})^2}{\sigma_{*,t,i}^2},
 \end{aligned} \tag{3.24}$$

where $y_{*,i}$ is the i th measurement from the test set, $m_{*,t,i}$ and $\sigma_{*,t,i}$ are the predictive mean and variance for the i th test point, and N_* is the number of test points.

First, we show the equivalence of the full-rank TNSRKf and the conventional Kalman filter. Then we show in a synthetic experiment how the choice of R_w and R_L impacts the accuracy of the approximation. Finally, we compare our method to the TNKF on a benchmark data set for nonlinear system identification.

All experiments were performed on an 11th Gen Intel(R) Core(TM) i7 processor running at 3.00 GHz with 16 GB RAM. For reproducibility of the method and the experiments, the code written in Julia programming language is freely available at <https://github.com/claraz>

3.6.1 EQUIVALENCE OF FULL-RANK TNSRKf AND KALMAN FILTER

In the first experiment, we show in which case our method is equivalent to the measurement update of the conventional Kalman filter. We generate $D = 3$ dimensional synthetic data sampled from a reduced-rank GP by [34] with a squared exponential kernel (lengthscale $\ell^2 = 0.1$ and signal variance $\sigma_f^2 = 1$), set the noise variance to $\sigma_y^2 = 0.01$ and use $I = 4$ basis functions per dimension, such that $\mathbf{P}_t \in \mathbb{R}^{64 \times 64}$. The input data lies in a cuboid given by $[-1 \ 1] \times [-1 \ 1] \times [-1 \ 1]$ and $N, N_* = 100$.

Table 3.2 shows the RMSE and NLL for test data at time $t = N$ for different choices of p . The TNSRKf is equivalent to the Kalman filter when both R_w and R_L are full-rank. In addition, p must be chosen such that the QR step, discussed in Section 3.5.2, is exact. For settings with lower values for p , the method trades off accuracy.

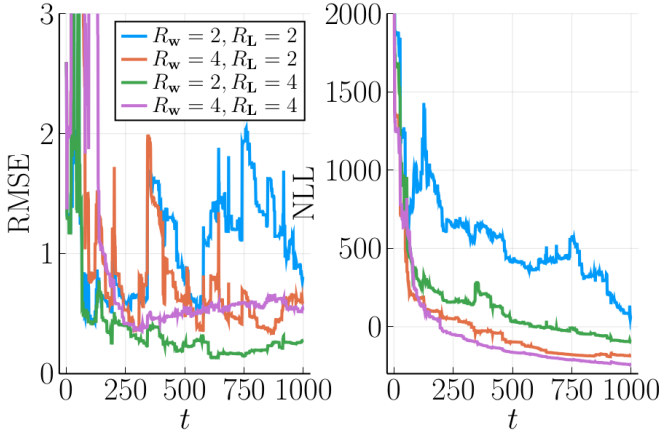


Figure 3.7: RMSE and NLL over time iterations for different combinations of R_w and R_L .

In the following sections, we look at scenarios where the Kalman filter can no longer be computed on a conventional laptop because both storage and computational time become unfeasible.

3.6.2 INFLUENCE OF THE RANKS ON THE APPROXIMATION

The choice of the TT- and TTm-ranks is not obvious and can be intricate. However, the computational budget often determines how high the ranks can be chosen. In this experiment, we use our method to make online GP predictions on synthetic data while varying the TTm-ranks of L_t , as well as the TT-ranks of w_t .

We consider the Volterra kernel, a popular choice for nonlinear system identification. It is known that the truncated Volterra series suffers from the curse of dimensionality, which was lifted in a TN setting by [3]. With the notation of this contribution to the thesis, the basis functions ϕ of parametric model (3.1) are a combination of monomials computed from the input sequence of the given problem. We generate synthetic training and testing data as described in [1], where $D = 7$ and $I = 4$ such that the number of parameters is $4^7 = 16384$. We set the SNR to 60, corresponding to $\sigma_y^2 = 6.96 \times 10^{-6}$.

Fig. 3.7 shows the RMSE and NLL on the testing data for $R_w = 2, 4$ and $R_L = 2, 4$ over time iterations of the TNSRKF. At $t = N$, the RMSE is lower for $R_w = 2$ and $R_L = 4$ than for $R_w = 4$ and $R_L = 4$. Thus, it seems that a lower value for the mean estimate represents the data better. Note that although having larger TT-ranks increases the degrees of freedom of the TT, it may not always improve the accuracy of the approximation, e.g. because higher TT-ranks can result in overfitting, while lower ranks can have a regularizing effect. The NLL is the lowest for $R_w = 4$ and $R_L = 4$, which is close to the NLL for $R_w = 4$ and $R_L = 2$. Note that the NLL for the same R_L is different for the two settings of R_w , because the NLL also depends on the difference between predicted and actual measurements, thus on the accuracy of \hat{w}_t .

This experiment showed that the choice of R_w and R_L influences the performance of the TNSRKF. Since higher values for the ranks also increase the computational complexity,

the computational budget will determine the higher limit for the ranks. In addition, an assumption with lower ranks may be fitting the data better in some cases.

3.6.3 COMPARISON TO TNKF FOR CASCADED TANKS BENCHMARK DATA SET

In this experiment, we compare our method to the TNKF on a nonlinear benchmark for system identification, the cascaded tanks data set. A detailed description can be found in [31]. The training and testing data both consist of a data set of 1024 data points each. To train our GP model, we choose lagged inputs and outputs as input to our GP, as described in [15], resulting in an input of dimensionality $D = 14$. We use a squared exponential kernel, which hyperparameters we optimize with the Gaussian process toolbox by [25], and we choose $I = 4$, such that the model has $M = 4^{14} = 268\,435\,456$ parameters.

For the comparison to the TNKF, we choose the TT-ranks for the mean to be $R_2 = R_{14} = 4$, $R_3 = \dots R_{13} = 10$, and we vary R_L and the TTm-ranks of the covariance matrix for the TNKF denoted by R_P . Fig. 3.8 and 3.9 show the RMSE and NLL over the time iterations of the respective filter. When $R_L = 1$ and $R_P = 1$, both methods perform almost the same, as visualized by the overlapping orange and blue lines. When $R_L^2 = R_P = 4$, our method improves both prediction accuracy and uncertainty estimation compared to the $R_L = 1$. On the contrary, the TNKF diverges and leaves the plotted figure area because the covariance matrix loses positive definiteness. When $R_L^2 = R_P = 16$, the TNKF shows a similar behavior, while the TNSRKf results in lower RMSEs but mostly higher NLL values. This setting shows that higher values for R_L are not always beneficial for the uncertainty estimation.

Finally, Fig. 3.10 shows the predictions with the TNSRKf on testing data after seeing 100, 200, and 922 data points. Aligned with the plot showing the RMSE and NLL, after 100 data points, the prediction is quite bad and uncertain. After 200 data points, the predictions are better and more certain and further improve after seeing the entire data set.

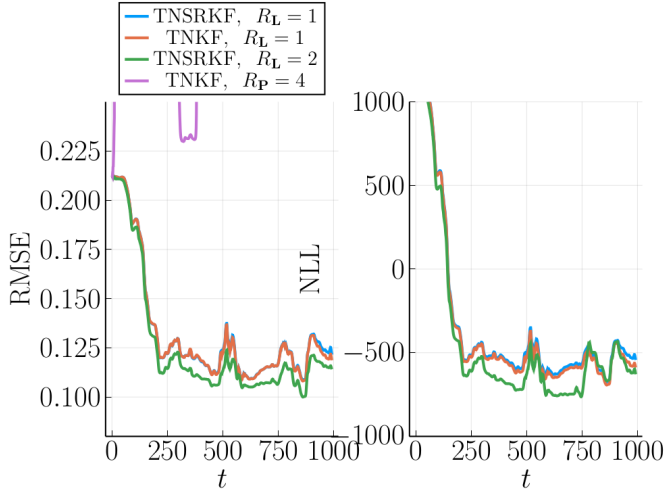


Figure 3.8: RMSE and NLL over iterations for TNKF and TNSRKf for $R_L = R_P = 1$ and $R_L = 2$, $R_P = R_L \cdot R_L = 4$. The orange and blue lines mostly overlap because both methods perform similarly for $R_L = R_P = 1$. Also, the violet curve leaves the plot window because the TNKF diverges.

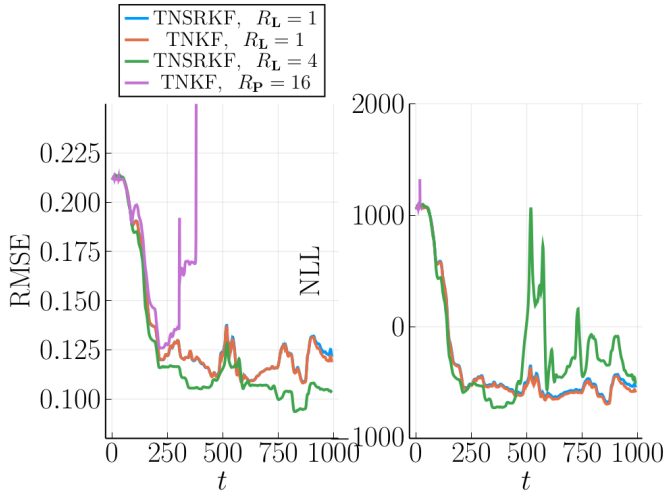


Figure 3.9: RMSE and NLL over iterations for TNKF and TNSRKf for $R_L = R_P = 1$ and $R_L = 4$, $R_P = R_L \cdot R_L = 16$. The orange and blue lines mostly overlap because both methods perform similarly for $R_L = R_P = 1$. Also, the violet curve leaves the plot window because the TNKF diverges.

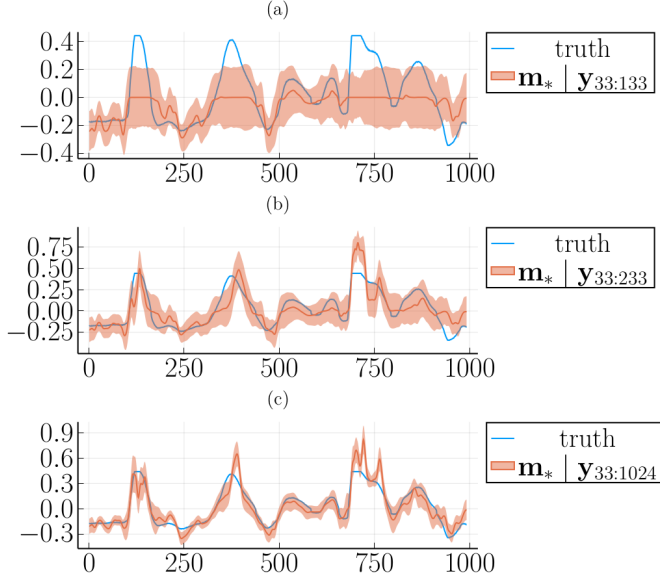


Figure 3.10: Predictions on test data with uncertainty bounds after seeing (a) 101, (b) 201, and (c) 992 data points for $R_L = 4$. The measurements start at 33 because the memory goes back 32 time steps.

3.7 CONCLUSION

In this contribution to the thesis, we presented a TT-based solution for online GP regression in terms of an SRKF. In our experiments, we show that our method is scalable to a high number of input dimensions at a reasonable computational cost such that all experiments could be run on a conventional laptop. In addition, we improve the state-of-the-art method for TN-based Kalman filter: In settings where the TNKF loses positive (semi-)definiteness and becomes numerically unstable, our method avoids this issue because we compute the square root covariance factors instead of the covariance matrix. In this way, we can choose settings for our method that achieve better accuracy than the TNKF.

A future work direction is online hyperparameter optimization. We are looking at a truly online scenario, so future data is not available. Thus, we cannot swipe over mini-batches of data multiple times like other methods, e.g. [32], to optimize hyperparameters.

Finally, there is still ongoing research to determine how to choose TT-ranks and TTm-ranks. In the synthetic experiments, we showed the impact of R_L and R_w . Generally, the TT- and TTm-ranks need to be treated as hyperparameters.

REFERENCES

- [1] Kim Batselier. Enforcing symmetry in tensor network MIMO Volterra identification. *IFAC-PapersOnLine*, 54(7):469–474, 2021.
- [2] Kim Batselier. Low-rank tensor decompositions for nonlinear system identification: A tutorial with examples. *IEEE Control Systems Magazine*, 42(1):54–74, 2022.
- [3] Kim Batselier, Zhongming Chen, and Ngai Wong. Tensor network alternating linear scheme for MIMO Volterra system identification. *Automatica*, 84:26–35, 2017.
- [4] Kim Batselier, Zhongming Chen, and Ngai Wong. A tensor network Kalman filter with an application in recursive MIMO Volterra system identification. *Automatica*, 84:17–25, 2017.
- [5] Kim Batselier, Ching-Yun Ko, and Ngai Wong. Extended Kalman filtering with low-rank tensor networks for MIMO Volterra system identification. In *Proceedings of the 2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 7148–7153, 2019.
- [6] Karl Berntorp. Online Bayesian inference and learning of Gaussian process state–space models. *Automatica*, 129:109613, 2021.
- [7] Seline J S De Rooij, Kim Batselier, and Borbala Hunyadi. Enabling large-scale probabilistic seizure detection with a tensor-network Kalman filter for LS-SVM. In *Proceedings of the 2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pages 1–5. IEEE, 2023.
- [8] Francis J Doyle, Ronald K Pearson, and Babatunde A Ogunnaike. *Identification and control using Volterra models*. Springer, 2002.
- [9] Glen Evenbly. TensorTrace: An application to contract tensor networks. *arXiv preprint arXiv:1911.02558*, 2019.
- [10] Gene H Golub and Charles F Van Loan. *Matrix Computations*. JHU press, 2013.
- [11] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.
- [12] Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal Gaussian process regression models. In *2010 IEEE International workshop on machine learning for signal processing*, pages 379–384. IEEE, 2010.
- [13] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012.
- [14] Pavel Izmailov, Alexander Novikov, and Dmitry Kropotov. Scalable Gaussian processes with billions of inducing inputs via tensor train decomposition. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 726–735. PMLR, 2018.

- [15] Ridvan Karagoz and Kim Batselier. Nonlinear system identification with regularized tensor network B-splines. *Automatica*, 122:109300, 2020.
- [16] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.
- [17] Miao Liu, Girish Chowdhary, Bruno Castra da Silva, Shih-Yuan Liu, and Jonathan P How. Gaussian processes for learning and control: A tutorial with examples. *IEEE Control Systems Magazine*, 38(5):53–86, 2018.
- [18] Clara Menzen, Manon Kok, and Kim Batselier. Tensor network square root kalman filter for online gaussian process regression. *Automatica*, *accepted for publication*, 2025.
- [19] Duy Nguyen-Tuong, Jan R Peters, and Matthias Seeger. Local Gaussian process regression for real time online model learning. *Advances in neural information processing systems*, 21, 2008.
- [20] Thomas Nickson, Tom Gunter, Chris Lloyd, Michael A Osborne, and Stephen Roberts. Blitzkriging: Kronecker-structured stochastic Gaussian processes. *arXiv preprint arXiv:1510.07965*, 2015.
- [21] Ivan V Oseledets. Approximation of 2D x 2D matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145, 2010.
- [22] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.
- [23] Joaquin Quinonero-Candela and Carl E Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.
- [24] Ananth Ranganathan, Ming-Hsuan Yang, and Jeffrey Ho. Online sparse Gaussian process regression and its applications. *IEEE Transactions on Image Processing*, 20(2):391–404, 2010.
- [25] Carl E Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [26] Thorsten Rohwedder and André Uschmajew. On local convergence of alternating schemes for optimization of convex problems in the tensor train format. *SIAM Journal on Numerical Analysis*, 51(2):1134–1162, 2013.
- [27] Simo Särkkä, Arno Solin, and Jouni Hartikainen. Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013.
- [28] Simo Särkkä and Lennart Svensson. *Bayesian Filtering and Smoothing*, volume 17. Cambridge university press, 2023.

- [29] Jonathan Schmidt, Philipp Hennig, Jörg Nick, and Filip Tronarp. The rank-reduced Kalman filter: Approximate dynamical-low-rank filtering in high dimensions. *Advances in Neural Information Processing Systems*, 36:61364–61376, 2023.
- [30] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.
- [31] Maarten Schoukens and Jean-Philippe Noël. Three benchmarks addressing open challenges in nonlinear system identification. *IFAC-PapersOnLine*, 50(1):446–451, 2017.
- [32] Manuel Schürch, Dario Azzimonti, Alessio Benavoli, and Marco Zaffalon. Recursive estimation for sparse Gaussian process regression. *Automatica*, 120:109127, 2020.
- [33] Arno Solin, Manon Kok, Niklas Wahlström, Thomas B Schön, and Simo Särkkä. Modeling and interpolation of the ambient magnetic field by Gaussian processes. *IEEE Transactions on robotics*, 34(4):1112–1127, 2018.
- [34] Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank Gaussian process regression. *Statistics and Computing*, 30:419–446, 2020.
- [35] Samuel Stanton, Wesley J Maddox, Ian Delbridge, and Andrew G Wilson. Kernel interpolation for scalable online Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 3133–3141. PMLR, 2021.
- [36] Andreas Svensson and Thomas B Schön. A flexible state–space model for learning nonlinear dynamical systems. *Automatica*, 80:189–199, 2017.
- [37] Ledyard R Tucker. Some Mathematical Notes on Three-Mode Factor Analysis. *Psychometrika*, 31(3):279–311, 1966.
- [38] Frida Viset, Rudy Helmons, and Manon Kok. An extended Kalman filter for magnetic field SLAM using Gaussian process regression. *Sensors*, 22(8):2833, 2022.
- [39] Frederiek Wesel and Kim Batselier. Large-scale learning with Fourier features and tensor decompositions. *Advances in Neural Information Processing Systems*, 34, 2021.
- [40] Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *Proceedings of the International conference on machine learning*, pages 1775–1784. PMLR, 2015.

4

LARGE-SCALE MAGNETIC FIELD MAPS USING STRUCTURED KERNEL INTERPOLATION FOR GAUSSIAN PROCESS REGRESSION

We present a mapping algorithm to compute large-scale magnetic field maps in indoor environments with approximate Gaussian process (GP) regression. Mapping the spatial variations in the ambient magnetic field can be used for localization algorithms in indoor areas. To compute such a map, GP regression is a suitable tool because it provides predictions of the magnetic field at new locations along with uncertainty quantification. Because full GP regression has a complexity that grows cubically with the number of data points, approximations for GPs have been extensively studied. In this contribution to the thesis, we build on the structured kernel interpolation (SKI) framework, speeding up inference by exploiting efficient Krylov subspace methods. More specifically, we incorporate SKI with derivatives (D-SKI) into the scalar potential model for magnetic field modeling and compute both predictive mean and covariance with a complexity that is linear in the data points. In our simulations, we show that our method achieves better accuracy than current state-of-the-art methods on magnetic field maps with a growing mapping area. In our large-scale experiments, we construct magnetic field maps from up to 40 000 three-dimensional magnetic field measurements in less than two minutes on a standard laptop.

4.1 INTRODUCTION

Indoor positioning and navigation in indoor environments is an active and challenging field of research, see e.g. [39, 40]. Since the global positioning system (GPS) does not work properly indoors, existing technologies rely on e.g. WLAN [8] or ultra-wideband [2]. In recent years, a novel and promising approach uses the spatial anomalies of the ambient magnetic field that is present indoors, see e.g. [3, 4, 7, 10, 21, 28, 29]. Probabilistic algorithms for indoor localization with magnetic field measurements use e.g. an extended Kalman filter [31] or a particle filter [26] in combination with Gaussian process (GP) regression. The motivation to use GPs [20] is the fact that they can be used to construct a magnetic field map from measurements providing a mean and uncertainty information which are both crucial for probabilistic localization algorithms. However, full GP regression becomes intractable for a large number of data points N , so existing approaches for magnetic field mapping have downsampled the data [29], made maps only using data close to a position of interest [16] or have approximated the GP kernel in terms of a number of basis functions [25]. Each of these methods has downsides, which can impact the accuracy of the map: The first two methods do not use all the data and the latter relies on a sufficient number of basis functions M_{bf} to achieve a good approximation of the kernel function [25]. Inspired by the fact that the literature about approximate GPs offers numerous other approaches for large-scale GPs that overcome the aforementioned limitations, in this contribution to the thesis, we build on the SKI framework by [35] to construct magnetic field maps. In the SKI framework, the measurements are observed through M_{ind} inducing variables, where the inducing inputs are placed on a Cartesian grid. The structure of the inducing inputs naturally enables Kronecker structure in the corresponding kernel matrix, as well as structured kernel interpolation (SKI), i.e. approximation of kernel matrices by interpolation. Exploiting Kronecker algebra and the sparsity of the interpolation matrices in Krylov subspace methods, we can compute magnetic field maps in an efficient way. This allows us to compute large-scale magnetic field maps as illustrated in Fig. 4.1 that are computationally unfeasible for full GP regression on a regular laptop. To construct the map, we use magnetometer data in combination with positions and orientations that are assumed to be known. Based on previous work by e.g. [11, 24], we model the magnetic field with the scalar potential model that allows for incorporating physical knowledge into the GP prior. In this framework, we use SKI with derivatives (D-SKI) [5] to compute the predictive means with conjugate gradients. For the predictive variance, we adapt the Lanczos Variance Estimates (LOVE) algorithm [18] to the D-SKI framework. The associated computational complexity is $\mathcal{O}((J + 2T)(3N + M_{\text{ind}}(M_{\text{ind}}^{(1)} + M_{\text{ind}}^{(2)} + M_{\text{ind}}^{(3)})))$, where J and T are chosen based on the desired accuracy of the conjugate gradient and Lanczos tridiagonalization algorithm, respectively, and $M_{\text{ind}}^{(d)}$ is the number of inducing inputs in the d th dimension.

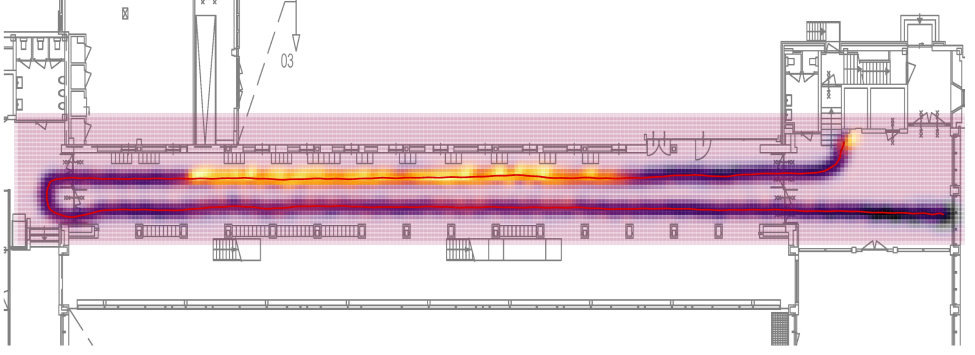


Figure 4.1: Magnitude of the magnetic field computed by constructing a map from 21 931 measurements, where darker regions correspond to a higher magnitude. The red line is the walking path, along which measurements are collected.

4

4.2 PROBLEM FORMULATION

We are interested in constructing large-scale magnetic field maps with GP regression. Similar to [24, 33], we assume the magnetic field to be curl-free and model the magnetic field measurements as the derivatives of a scalar potential φ on which we put a GP prior. Given $n = 1, \dots, N$ 3D positions \mathbf{p} at which magnetic field measurements have been collected, the GP model is given by

$$\begin{aligned} \varphi(\mathbf{p}) &\sim \mathcal{GP}(0, \kappa(\mathbf{p}, \mathbf{p}')), \\ \mathbf{y}_n &= -\nabla\varphi(\mathbf{p}_n) + \epsilon_n, \quad \epsilon_n \sim \mathcal{N}(\mathbf{0}, \sigma_y^2 \mathbf{I}_3), \end{aligned} \quad (4.1)$$

where $\mathbf{y}_n \in \mathbb{R}^3$ contains the x -, y - and z - component of the magnetic field measurement, κ is the kernel function, and σ_y^2 is the noise variance. We choose the kernel to be the squared exponential kernel, as in related literature, see e.g. [1, 12, 13, 29, 30]. The squared exponential kernel is given by

$$\kappa(\mathbf{p}, \mathbf{p}') = \sigma_f^2 \exp\left(-\frac{\|\mathbf{p} - \mathbf{p}'\|_2}{2\ell^2}\right), \quad (4.2)$$

where σ_f^2 and ℓ are the hyperparameters of the kernel, the signal variance, and the length scale.

Although we measure the Earth's magnetic field together with anomalies, we choose to only model the anomalies, because this model choice fits better into our approximation scheme. When also considering the local magnetic field as e.g. in [32], the kernel includes a linear term as well.

As the gradient operator is a linear operator given the linearity of differentiation [23], the predictive distribution of the three components of the magnetic field in a new location \mathbf{p}_* can be expressed in terms of a mean and a variance of \mathbf{f}_* given by

$$\begin{aligned} \mathbb{E}[\mathbf{f}_*] &= \partial^2(\mathbf{K}_{*,\mathbf{f}}) (\partial^2(\mathbf{K}_{\mathbf{f},\mathbf{f}}) + \sigma_y^2 \mathbf{I}_{3N})^{-1} \text{vec}(\mathbf{Y}^\top), \\ \mathbf{V}[\mathbf{f}_*] &= \partial^2(\mathbf{K}_{*,*}) - \partial^2(\mathbf{K}_{*,\mathbf{f}}) (\partial^2(\mathbf{K}_{\mathbf{f},\mathbf{f}}) + \sigma_y^2 \mathbf{I}_{3N})^{-1} \partial^2(\mathbf{K}_{\mathbf{f},*}), \end{aligned} \quad (4.3)$$

where $\mathbf{Y} = [\mathbf{y}_1^\top \mathbf{y}_2^\top \dots \mathbf{y}_N^\top] \in \mathbb{R}^{N \times 3}$ are all magnetic field measurements. The entries of $\partial^2(\mathbf{K}_{f,f})$, $\partial^2(\mathbf{K}_{*,f})$ and $\partial^2(\mathbf{K}_{*,*})$ are computed block-wise in terms of 3×3 blocks for each pair of positions with $\nabla_{\mathbf{p}} \kappa(\mathbf{p}, \mathbf{p}') \nabla_{\mathbf{p}'}^\top$, $\nabla_{\mathbf{p}} \kappa(\mathbf{p}, \mathbf{p}_*) \nabla_{\mathbf{p}_*}^\top$ and $\nabla_{\mathbf{p}_*} \kappa(\mathbf{p}_*, \mathbf{p}'_*) \nabla_{\mathbf{p}'_*}^\top$, respectively, where ∇ denotes the gradient that is taken w.r.t. to the vector specified in the subscript.

With (4.3) it is possible to predict the magnetic field in new locations. In practice, however, this is only possible for a small number of data points, since full GP regression generally scales cubically with N . In this case it even scales cubically with $3N$, because of the 3 derivatives. In this contribution to the thesis, we build on the SKI framework, described in the next section, to make large-scale magnetic field maps in an efficient way.

4.3 SKI FRAMEWORK

The SKI framework is based on sparse approximations for GP regression, using a set of M_{ind} inducing inputs $\mathbf{x}_u \in \mathbb{R}^D$. In the context of magnetic field modeling, the inducing inputs are positions in \mathbb{R}^3 . Based on the Nyström approximation [34] of the kernel, the simplest formulation of the inducing input approach is the subset of regressors (SoR), which can be implemented similarly to the predictive distribution for full GP regression using an approximation to the kernel function [19], given by

$$\kappa_{\text{SoR}}(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}, \mathbf{x}_u) \mathbf{K}_{u,u}^{-1} \kappa(\mathbf{x}_u, \mathbf{x}'), \quad (4.4)$$

where $\mathbf{K}_{u,u}$ denotes the covariance matrix of all the inducing inputs. The approximated kernel function results in new kernel matrices, which are then given by

$$\mathbf{K}_{f,f} = \mathbf{K}_{f,u} \mathbf{K}_{u,u}^{-1} \mathbf{K}_{u,f}, \quad (4.5a)$$

$$\mathbf{K}_{*,*} = \mathbf{K}_{*,u} \mathbf{K}_{u,u}^{-1} \mathbf{K}_{u,*}, \quad (4.5b)$$

$$\mathbf{K}_{*,f} = \mathbf{K}_{*,u} \mathbf{K}_{u,u}^{-1} \mathbf{K}_{u,f}. \quad (4.5c)$$

In the SKI framework [35], the inducing inputs are placed on a Cartesian grid, which is equispaced per dimension and of size $M_{\text{ind}}^{(1)} \times M_{\text{ind}}^{(2)} \times \dots \times M_{\text{ind}}^{(D)}$, for a total of $M_{\text{ind}} = \prod_{d=1}^D M_{\text{ind}}^{(d)}$ inducing inputs.

Consequently, product kernels - here the squared exponential kernel is considered - decompose over the input dimensions. Thus, $\mathbf{K}_{u,u}$ can be expressed as a Kronecker product of D matrices [22, 35], given by

$$\mathbf{K}_{u,u} = \bigotimes_{d=1}^D \mathbf{K}_{u,u}^{(d)}, \quad (4.6)$$

where $\mathbf{K}_{u,u}^{(d)}$ is computed with a squared exponential kernel having a scaled signal variance $\sigma_f^{2/D}$ [22].

In addition, in the SKI framework, the cross-covariance matrices $\mathbf{K}_{f,u}$ and $\mathbf{K}_{*,u}$ are approximated using sparse interpolation matrices, $\mathbf{W}_f \in \mathbb{R}^{N \times M}$ and $\mathbf{W}_* \in \mathbb{R}^{N_* \times M}$, such that

$$\mathbf{K}_{f,u} \approx \mathbf{W}_f \mathbf{K}_{u,u} \quad \text{and} \quad \mathbf{K}_{*,u} \approx \mathbf{W}_* \mathbf{K}_{u,u}. \quad (4.7)$$

Each row of the interpolation matrices contains 4^D interpolation weights for cubic interpolation [9] which is suggested in [35].

4.4 LARGE-SCALE MAGNETIC FIELD MAPS

Our goal is to compute magnetic field maps in 3D using magnetic field measurements as training data. In order to be able to use large data sets, we exploit mathematical formulations in the SKI framework adapted to magnetic field modeling to compute predictive means and variances in an efficient way. As the predictive distribution of the scalar potential model in (4.3) is based on the derivatives of the magnetic scalar potential, we approximate the elementwise computation $\partial^2(\cdot)$ of the kernel matrices with D-SKI [5]. Using D-SKI, $\partial^2(\cdot)$ can be simplified through differentiation of the interpolation scheme, such that it is

$$\partial^2(\mathbf{K}_{f,f}) \approx (\partial \mathbf{W}_f) \mathbf{K}_{u,u} (\partial \mathbf{W}_f)^\top, \quad (4.8a)$$

$$\partial^2(\mathbf{K}_{*,f}) \approx (\partial \mathbf{W}_*) \mathbf{K}_{u,u} (\partial \mathbf{W}_f)^\top, \quad (4.8b)$$

where $\partial \mathbf{W}_f \in \mathbb{R}^{3N \times M_{\text{ind}}}$ and $\partial \mathbf{W}_* \in \mathbb{R}^{3N_* \times M_{\text{ind}}}$. Note that the first size of the interpolation matrices is multiplied by a factor of 3 compared to (4.7) due to the three components of the magnetic field. As mentioned in the previous section, in SKI a cubic interpolation is advised [35], while in D-SKI a quintic interpolation scheme is used [5]. We use a cubic interpolation scheme for D-SKI, since preliminary experiments show that the approximation is sufficient for our application.

The predictive distribution in a new location $\mathbf{p}_* \in \mathbb{R}^3$ of the scalar potential model for magnetic field modeling using D-SKI is given by

$$\mathbf{E}[\mathbf{f}_*] = (\partial \mathbf{W}_*) \mathbf{K}_{u,u} (\partial \mathbf{W}_f)^\top \mathbf{A}^{-1} \text{vec}(\mathbf{Y}^\top) \quad (4.9a)$$

$$\mathbf{V}[\mathbf{f}_*] = (\partial \mathbf{W}_*) \mathbf{K}_{u,u} (\partial \mathbf{W}_*)^\top - (\partial \mathbf{W}_*) \mathbf{C} (\partial \mathbf{W}_*)^\top \quad (4.9b)$$

The matrices \mathbf{A} and \mathbf{C} only depend on the training data and are defined as

$$\mathbf{A} := (\partial \mathbf{W}_f) \mathbf{K}_{u,u} (\partial \mathbf{W}_f)^\top + \sigma_y^2 \mathbf{I}_{3N}, \quad (4.10a)$$

$$\mathbf{C} := \mathbf{K}_{u,u} (\partial \mathbf{W}_f)^\top \mathbf{A}^{-1} (\partial \mathbf{W}_f) \mathbf{K}_{u,u}. \quad (4.10b)$$

A naive computation of the predictive mean and variance with (4.9a) and (4.9b) would require an inverse of a $3N \times 3N$ matrix. Inducing inputs on a grid and kernel interpolation, however, enable efficient computations via Krylov subspace methods. The key to efficient computation is not to construct the matrices involved in (4.9a) and (4.9b) explicitly but to keep them in terms of a factorized format of 3 smaller matrices, one for each input dimension. Based on [5], we use preconditioned conjugate gradient to find the solution $\boldsymbol{\alpha}$ to the linear system given by

$$\mathbf{A} \boldsymbol{\alpha} = \text{vec}(\mathbf{Y}^\top). \quad (4.11)$$

While in D-SKI fast matrix-vector-multiplications (MVMs) are computed via FFT, we compute them like in [35] via Kronecker MVMs as described in [36]. In this way, the predictive mean of a 3D map is computed with a computational complexity of

$$\mathcal{O} \left(J \left(3N + M_{\text{ind}} \sum_{d=1}^3 M_{\text{ind}}^{(d)} \right) \right) = \mathcal{O}_{\text{ind}}, \quad (4.12)$$

where J is the number of iterations in the conjugate gradient. To compute the predictive variance of the magnetic field map, we build on the Lanczos Variance Estimates approach

by [18], which is based on the Lanczos tridiagonalization algorithm. We use the Lanczos Variance Estimates within the D-SKI framework to find a low-rank approximation for \mathbf{A} given by

$$\mathbf{A} \approx \mathbf{Q}_T \mathbf{T}_T \mathbf{Q}_T^\top, \quad (4.13)$$

where $\mathbf{Q}_T \in \mathbb{R}^{3N \times T}$ contains T orthonormal vectors corresponding to the first T leading eigenvalues and $\mathbf{T}_T \in \mathbb{R}^{T \times T}$ has a tridiagonal structure [6]. Once an approximation of \mathbf{A} is found, \mathbf{C} from (4.10b) can be computed as

$$\begin{aligned} \mathbf{C} &= \mathbf{K}_{\mathbf{u},\mathbf{u}} (\partial \mathbf{W}_f)^\top \mathbf{A}^{-1} (\partial \mathbf{W}_f) \mathbf{K}_{\mathbf{u},\mathbf{u}} \\ &\approx \mathbf{K}_{\mathbf{u},\mathbf{u}} (\partial \mathbf{W}_f)^\top \mathbf{Q}_T \mathbf{T}_T^{-1} \mathbf{Q}_T^\top (\partial \mathbf{W}_f) \mathbf{K}_{\mathbf{u},\mathbf{u}}. \end{aligned} \quad (4.14)$$

As described in [18], we again exploit Kronecker algebra to compute the MVMs in (4.14) efficiently. The computation of \mathbf{C} has an associated computational complexity of $\mathcal{O}(2T(3N + M_{\text{ind}}(M_{\text{ind}}^{(1)} + M_{\text{ind}}^{(2)} + M_{\text{ind}}^{(3)})))$ for magnetic field modeling [18]. In numerical implementations, the complexity is higher due to the full reorthogonalization required for the Lanczos tridiagonalization algorithm, scaling linearly with the number of training points N and Lanczos iterations T .

Alternatively, the predictive variance could also be computed using conjugate gradient. However, computing \mathbf{C} by solving a linear system to find $\mathbf{A}^{-1} (\partial \mathbf{W}_f) \mathbf{K}_{\mathbf{u},\mathbf{u}}$ needs to be done for each column of $(\partial \mathbf{W}_f) \mathbf{K}_{\mathbf{u},\mathbf{u}}$ sequentially, which is not particularly efficient. In [37], the variance is stochastically estimated by drawing samples from the predictive distributions [17]. While this approach can reduce the computational complexity associated with the computation of the predictive variances, it introduces significant accuracy losses.

Once $\boldsymbol{\alpha}$ and \mathbf{C} are computed, predictions in new locations can be computed with (4.9a) and (4.9b) again by exploiting Kronecker algebra and the structure of the interpolation matrices.

4.5 EXPERIMENTS

In our experiments, we first compare our method to existing ones in simulations with synthetic data, then show our method's scalability with large-scale magnetometer data collected with a motion capture suit. All computations are done on a 2016 HP ZBook Studio G3 laptop (Intel Core i7 @ 2.60 GHz, 8GB RAM).

4.5.1 ACCURACY ANALYSIS FOR GROWING MAPPING AREA

In the first simulation, we compare the accuracy of magnetic field maps computed with our and existing methods. For this, we create a synthetic data set of 6000 data points, representing a magnetic field map. Each input is a random 3D vector lying in a box confined by $[-20, 20] \times [-20, 20] \times [0.01, 0.01]$ and the corresponding output is sampled from a GP prior with a curl-free kernel with hyperparameters, length scale, signal variance, and noise variance, $[\ell, \sigma_f^2, \sigma_y^2] = [2, 1, 0.01]$ which is equivalent to drawing samples from model (4.1) [32].

We compute two maps with different sizes: area 1 of size 20×20 and area 2 of size 40×40 . For area 1, a subset of the 6000 data points is used, laying in the white square shown in Fig. 4.2, and for area 2, all data is used. We divide the data points in each area

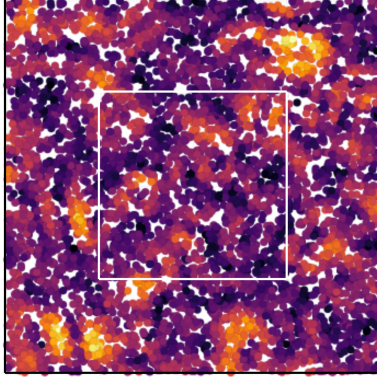


Figure 4.2: Synthetic data with white and black squares denoting area 1 of size 20×20 ($N \approx 1526$) and area 2 of size 40×40 ($N = 6000$). The scattered dots are data points and the color corresponds to the magnitude of the magnetic field.

4

into 80% training data and 20% testing data. With the testing data, we compute root mean square errors (RMSEs) as a metric for accuracy.

We compare our method to a GP where we downsample the data, as well as to the basis function approach by [25], showing how the area size impacts the accuracy of the map for different M_{ind} , N_{dwn} and M_{bf} in the respective methods. Since the domain on which basis functions are computed needs to be a bit bigger than the mapping area [25], we add twice the length scale in each dimension.

To be able to compare all methods, we compute the computational complexity of our method \mathcal{O}_{ind} as defined in (4.12), and impose this complexity as the computational budget for the other methods. The complexity of the approach of downsampling the data is equal to the cubic complexity of full GP. The complexity of the basis function approach is linear in the number of data points and quadratic in the number of basis functions. By equating the complexities

$$\mathcal{O}_{\text{ind}} = \mathcal{O}(M_{\text{bf}}^2 3N) = \mathcal{O}(N_{\text{dwn}}^3), \quad (4.15)$$

and solving for M_{bf} and N_{dwn} , those numbers can be used in the corresponding methods.

For each of the two area sizes, we have six different settings, where we vary the number of inducing inputs, i.e., $M_{\text{ind}}^{(1)} = M_{\text{ind}}^{(2)} = [10, 20, 40, 80, 100, 200]$ and $M_{\text{ind}}^{(3)} = 5$. The number of iterations in the conjugate gradient, J , is based on the tolerance for accuracy. Table 4.1 summarizes all settings used in the simulations. The first two rows are the total number of inducing inputs M_{ind} and the other rows are the number of basis functions and downsampled data points that result from the imposed computational budget \mathcal{O}_{ind} . We run the simulation 100 times, where each time new data is sampled. An example of the data is illustrated in Fig. 4.2.

Fig. 4.3 shows the RMSEs on the testing data computed with the three methods for area 1 (solid line) and area 2 (dash-dotted line). The mean and standard deviation of the RMSE from 100 runs are plotted. The horizontal axis denotes the 6 different settings as described in Table 4.1. The mean RMSE of the full GP is given for area 2 as a reference (dashed black line), for area 1 the RMSE is very similar and therefore not shown. The

figure shows that for each method, the RMSEs are larger for the 40×40 area than for the 20×20 area. This implies that a larger M_{ind} , N_{down} , and M_{bf} are required for a larger area to achieve low RMSE. Also, all methods converge to the RMSE of the full GP in the limit. Comparing our methods to the other two, for both areas our method has lower RMSEs. In addition, the difference in RMSEs between our methods and the other methods is more significant for the larger area. The main takeaway of this simulation is that our method has better accuracy than the other methods when the computational budget \mathcal{O}_{ind} is imposed for all methods. It follows that the computational cost to compute a map of a specified accuracy is lower for our approach compared to the others. Especially when computing large-scale maps, this becomes important: Since the number of basis functions needed to approximate the kernel function sufficiently is known to scale with the domain size [25], a trade-off between accuracy and computational cost needs to be made. A similar trade-off is necessary for downsampling the data, because more data points are required for larger areas.

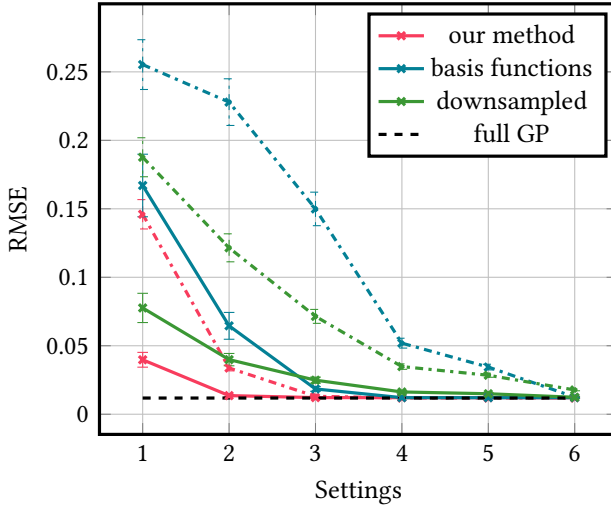


Figure 4.3: RMSE for area 1 (solid line) and 2 (dash-dotted line). For reference, the RMSE of full GP is given for area 2 only, because the other value is very similar. The horizontal axis are the six different settings described in Table 4.1. Mean and standard deviation are plotted for 100 runs of the simulation.

4.5.2 ANALYSIS OF MAPS WITH DIVIDED MAPPING AREA

As mentioned in the previous section, the basis function approach requires a large number of basis functions for growing mapping areas. To lower the computational complexity, an alternative approach for computing large-scale maps with basis functions is dividing the area into smaller areas for each of which a GP approximation with basis functions is made [11]. To train each smaller map, not only training data from the mapping area is used, but also training data in close proximity to that area. In a second simulation, we show that this strategy may result in inconsistencies at boundaries. We use synthetic data sampled from a GP prior with a curl-free kernel and with hyperparameters $[\ell, \sigma_f^2, \sigma_y^2] = [5, 1, 0.01]$, divide

Table 4.1: M_{ind} , M_{bf} and N_{dwn} used in the 6 settings of the simulation. The first and second rows for basis function approach and downsampled data are the values for the area 1 and 2, respectively. Value ranges for 100 simulations.

	Setting	1	2	3	4	5	6
M_{ind}	Area 1	500	2K	8K	32K	50K	200K
	Area 2	500	2K	8K	32K	50K	200K
M_{bf}	Area 1	35 - 36	97 - 98	262 - 266	726 - 738	1012 - 1027	2843 - 2887
	Area 2	15 - 16	57 - 58	152 - 153	420 - 422	584 - 587	1635 - 1646
N_{dwn}	Area 1	55 - 56	107 - 108	208 - 210	411 - 415	512 - 518	1020 - 1030
	Area 2	49 - 50	120 - 121	231 - 232	455 - 457	567 - 569	1126 - 1131

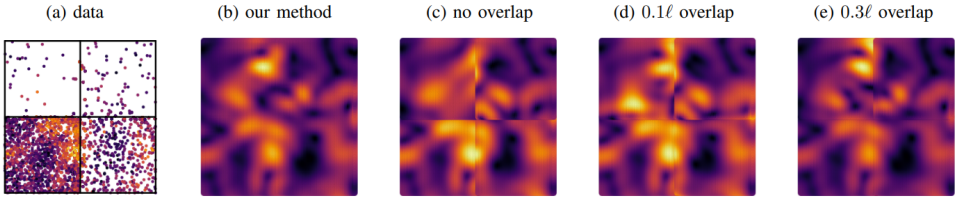


Figure 4.4: Magnetic field data (a). Predictions with our method for the whole area (b). Predictions with basis functions in smaller areas separately using an overlap of 0, 0.1ℓ , and 0.3ℓ for the training data, respectively (c)-(e).

the data into four regions and downsample the data by different factors in every region, as shown in Fig. 4.4 (a). The reason for it is to analyze how the amount of data points in a neighboring area influences the inconsistencies at the boundaries. For training each map, we use data points in each area as well as data from an overlap of size $0\ell, 0.1\ell, 0.3\ell$ to the neighboring areas. The domain on which we compute basis functions is then the size of the mapping area plus the overlap plus twice the length scale. The result of the reconstruction is shown in Fig. 4.4 (c)-(e). The figure shows that for no overlap, there are visible inconsistencies in the mean on the magnetic field prediction. For 0.1ℓ and 0.3ℓ , the inconsistencies are smaller but still present. In addition, the inconsistencies are more visible at the boundaries of areas with fewer data. As a comparison, a map reconstructed with our method is shown in Fig. 4.4 (b), where the mapping area is not divided.

4.5.3 LARGE-SCALE MAP IN UNIVERSITY BUILDING

For our experiments, we use data collected with a motion capture suit (Xsens MVN Link [15]). The suit contains 17 inertial measurement units (IMUs) equipped with magnetometers tightly attached to segments all over the body. The IMUs provide accelerometer, gyroscope, and magnetometer data with position and orientation data in the navigation frame at a maximum sample rate of 240Hz. The magnetometers have been calibrated using software available with the suit, such that after calibration the norm of the undisturbed Earth's magnetic field is 1 [38]. In a pre-processing step, the data is first rotated to the global frame defined by the magnetic North pole, and second, the mean is subtracted from the x -, y -, and z -component, since we only model the anomalies of the magnetic field. We use the data collected from one IMU that is located at the pelvis. In the first large-scale

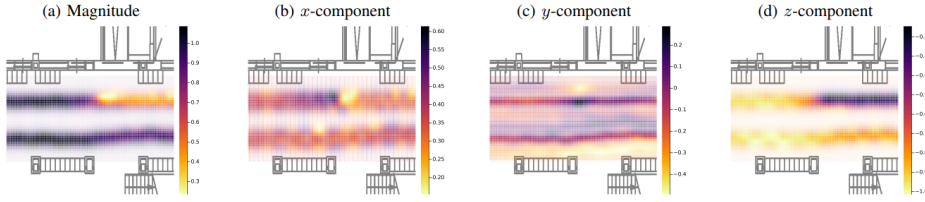


Figure 4.5: Magnetic field maps for a smaller region of map in Fig. 4.1, located on the left part of the hallway. In the z -components, the lockers are strongly visible, indicating that the metal in the lockers mainly disturbs the magnetic field in that direction. Transparency indicates the certainty of the prediction.

experiment, the magnetic field of one of the university hallway wings at the TU Delft is computed based on $N = 21\,931$ magnetic field measurements. The area of interest for this experiment is a rectangle bounded by $[-34\text{ m}, 34\text{ m}] \times [-5.25\text{ m}, 5.25\text{ m}]$ located at a height of approximately 1 m. The magnetic field is estimated with our method using an inducing point grid of size $400 \times 40 \times 4$, for a total of 64 000 inducing points. The number of inducing points per dimension is chosen such that several inducing points are present per characteristic length scale. The hyperparameters are trained on a subset of the data by minimizing the log marginal likelihood with the curl-free kernel, resulting in $\ell = 0.5\text{ m}$, $\sigma_f = 0.2$ and $\sigma_y = 0.01$. Fig. 4.1 shows the magnitude of the magnetic field predictions, computed from the three components. Darker regions in the figure correspond to a higher magnitude of the magnetic field. Since there are metallic lockers located in the hallway, we expected a strong magnetic anomaly, which is visible in the figure. Fig. 4.5 shows a smaller section of the magnetic field map, in terms of its magnitude, as well as its x -, y - and z -component. The magnetic disturbance caused by the lockers is mostly visible in the z -components, as shown in the upper right part of Fig. 4.5 (d). The transparency in the figure indicates the certainty of the map.

Regarding computational time, computing the map with 21 931 measurements took approximately 1 min for training and 18 s for testing. In a second experiment with 41 383 data points, the training took approximately 97 s for training and 18 s for testing. While with full GP the map would not be feasible to compute on our laptop, our method scales very well: The computing time approximately only doubling when doubling the data points, thus is approximately linear in N .

4.6 CONCLUSION

In this contribution to the thesis, we described an algorithm to efficiently compute large-scale magnetic field maps using approximate Gaussian process regression. We used inducing inputs on a grid and structured kernel interpolation with derivative to compute the predictive mean and an algorithm based on Lanczos tridiagonalization to compute variance estimates. We compared our method to existing methods in simulations and showed its scalability in large-scale experiments. There are multiple directions for future work. The kernel in the scalar potential model can be complemented by a linear kernel as in [24, 33] in order to also model the underlying Earth's magnetic field. The linear kernel, however, is not a product kernel and can thus not be decomposed as a Kronecker product. When

using the equivalent curl-free model instead, the kernel consists of a constant term and a curl-free kernel [32], where the constant term can again be decomposed as a Kronecker product. In this way, a model considering both the Earth's magnetic field and the spatial anomalies can be used in the D-SKI framework. In addition, the presented method can be extended to online mapping to enable its use in e.g. simultaneous localization and mapping (SLAM) algorithms. The SKI framework for online GPs has been described in [27] and can be adapted to magnetic field mapping.

ACKNOWLEDGMENT

We would like to thank Fabian Girrba from Movella Technologies for post-processing data collected with the motion capture suit.

REFERENCES

- [1] Naoki Akai and Koichi Ozaki. Gaussian processes for magnetic map-based localization in large-scale indoor environments. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4459–4464. IEEE, 2015.
- [2] Abdulrahman Alarifi, AbdulMalik Al-Salman, Mansour Alsaleh, Ahmad Alnafessah, Suheer Al-Hadhrani, Mai A Al-Ammar, and Hend S Al-Khalifa. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, 16(5):707, 2016.
- [3] Gennadii Berkovich, Dmitry Churikov, Jacques Georgy, and Chris Goodall. Coursa venue: Indoor navigation platform using fusion of inertial sensors with magnetic and radio fingerprinting. In *Proceedings of the 2019 22th International Conference on Information Fusion (FUSION)*, pages 1–6. IEEE, 2019.
- [4] Jaewoo Chung, Matt Donahoe, Chris Schmandt, Ig-Jae Kim, Pedram Razavai, and Micaela Wiseman. Indoor location sensing using geo-magnetism. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, pages 141–154, 2011.
- [5] David Eriksson, Kun Dong, Eric Lee, David Bindel, and Andrew G Wilson. Scaling Gaussian process regression with derivatives. *Advances in neural information processing systems*, 31, 2018.
- [6] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.
- [7] Janne Haverinen and Anssi Kemppainen. Global indoor self-localization based on the ambient magnetic field. *Robotics and Autonomous Systems*, 57(10):1028–1035, 2009.
- [8] Suining He and Shueng-Han G Chan. Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons. *IEEE Communications Surveys & Tutorials*, 18(1):466–490, 2015.
- [9] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153–1160, 1981.
- [10] Seong-Eun Kim, Yong Kim, Jihyun Yoon, and Eung Sun Kim. Indoor positioning system using geomagnetic anomalies for smartphones. In *Proceedings of the 2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–5. IEEE, 2012.
- [11] Manon Kok and Arno Solin. Scalable magnetic field SLAM in 3D using Gaussian process maps. In *Proceedings of the 2018 21st international conference on information fusion (FUSION)*, pages 1353–1360. IEEE, 2018.
- [12] Taylor N Lee and Aaron J Canciani. MagSLAM: Aerial simultaneous localization and mapping using Earth’s magnetic anomaly field. *Navigation*, 67(1):95–107, 2020.

- [13] Yuqi Li, Zhe He, John Nielsen, and Gérard Lachapelle. Using Wi-Fi/magnetometers for indoor location and personal navigation. In *Proceedings of the 2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7. IEEE, 2015.
- [14] Clara Menzen, Marnix Fetter, and Manon Kok. Large-scale magnetic field maps using structured kernel interpolation for gaussian process regression. In *Proceedings of the 2023 26th International Conference on Information Fusion (FUSION)*, pages 1–7. IEEE, 2023.
- [15] Movella Inc. MVN Link. [Online; accessed March, 2023], <https://www.movella.com/products/motion-capture/xsens-mvn-link>.
- [16] Duy Nguyen-Tuong, Jan Peters, and Matthias Seeger. Local Gaussian process regression for real-time online model learning. *Advances in neural information processing systems*, 21, 2008.
- [17] George Papandreou and Alan L Yuille. Efficient variational inference in large-scale Bayesian compressed sensing. In *Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1332–1339. IEEE, 2011.
- [18] Geoff Pleiss, Jacob Gardner, Kilian Weinberger, and Andrew G Wilson. Constant-time predictive distributions for Gaussian processes. In *Proceedings of the International Conference on Machine Learning*, pages 4114–4123. PMLR, 2018.
- [19] Joaquin Quinonero-Candela and Carl E Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.
- [20] Carl E Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (GPML) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010.
- [21] Patrick Robertson, Martin Frassl, Michael Angermann, Marek Doniec, Brian J Julian, Maria Garcia Puyol, Mohammed Khider, Michael Lichtenstern, and Luigi Bruno. Simultaneous localization and mapping for pedestrians using distortions of the local magnetic field intensity in large indoor environments. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation*, pages 1–10. IEEE, 2013.
- [22] Yunus Saatçi. *Scalable inference for structured Gaussian process models*. PhD thesis, University of Cambridge, 2012.
- [23] Simo Särkkä. Linear operators and stochastic partial differential equations in Gaussian process regression. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 151–158. Springer, 2011.
- [24] Arno Solin, Manon Kok, Niklas Wahlström, Thomas B Schön, and Simo Särkkä. Modeling and interpolation of the ambient magnetic field by Gaussian processes. *IEEE Transactions on Robotics*, 34(4):1112–1127, 2018.

- [25] Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank Gaussian process regression. *Statistics and Computing*, 30(2):419–446, 2020.
- [26] Arno Solin, Simo Särkkä, Juho Kannala, and Esa Rahtu. Terrain navigation in the magnetic landscape: Particle filtering for indoor positioning. In *Proceedings of the 2016 European Navigation Conference (ENC)*, pages 1–9. IEEE, 2016.
- [27] Samuel Stanton, Wesley Maddox, Ian Delbridge, and Andrew G Wilson. Kernel interpolation for scalable online Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 3133–3141. PMLR, 2021.
- [28] Joaquín Torres-Sospedra, David Rambla, Raul Montoliu, Oscar Belmonte, and Joaquín Huerta. Ujiindoorloc-mag: A new database for magnetic field-based localization problems. In *Proceedings of the 2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10. IEEE, 2015.
- [29] Ilari Vallivaara, Janne Haverinen, Anssi Kemppainen, and Juha Röning. Simultaneous localization and mapping using ambient magnetic field. In *Proceedings of the 2010 IEEE Conference on Multisensor Fusion and Integration*, pages 14–19. IEEE, 2010.
- [30] Ilari Vallivaara, Janne Haverinen, Anssi Kemppainen, and Juha Röning. Magnetic field-based SLAM method for solving the localization problem in mobile robot floor-cleaning task. In *Proceedings of the 2011 15th International Conference on Advanced Robotics (ICAR)*, pages 198–203. IEEE, 2011.
- [31] Frida Viset, Rudy Helmons, and Manon Kok. An extended Kalman filter for magnetic field SLAM using Gaussian process regression. *Sensors*, 22(8):2833, 2022.
- [32] Niklas Wahlström. *Modeling of magnetic fields and extended objects for localization applications*. PhD thesis, Linköping University, 2015.
- [33] Niklas Wahlström, Manon Kok, Thomas B Schön, and Fredrik Gustafsson. Modeling magnetic fields using Gaussian processes. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3522–3526. IEEE, 2013.
- [34] Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.
- [35] Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *Proceedings of the International conference on machine learning*, pages 1775–1784. PMLR, 2015.
- [36] Andrew G Wilson. *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes*. PhD thesis, University of Cambridge, 2014.
- [37] Andrew G Wilson, Christoph Dann, and Hannes Nickisch. Thoughts on massively scalable Gaussian processes. *arXiv preprint arXiv:1511.01870*, 2015.

- [38] Xsens. Interpreting magnetic field data represented as an arbitrary unit (a.u.). *Xsens knowledge base*, Aug 2022. [Online; accessed November 14, 2022], <https://base.xsens.com/s/article/Interpreting-magnetic-field-data-represented-as-an-arbitrary-unit-a-u>.
- [39] Ali Yassin, Youssef Nasser, Mariette Awad, Ahmed Al-Dubai, Ran Liu, Chau Yuen, Ronald Raulefs, and Elias Aboutanios. Recent advances in indoor localization: A survey on theoretical approaches and applications. *IEEE Communications Surveys & Tutorials*, 19(2):1327–1346, 2016.
- [40] Faheem Zafari, Athanasios Gkelias, and Kin K Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys & Tutorials*, 21(3):2568–2599, 2019.

5

CONCLUSIONS AND FUTURE WORK RECOMMENDATIONS

*Coll'astuzia, coll'arguzia | With cunning, with wit
Col giudizio, col criterio | With judgment, with criterion
Si potrebbe ... Il fatto è serio: | You could ... The fact is serious:
Ma, credete, si farà. | But, believe, it will be done.
Se tutto il codice deessi volgere. | If all the codex I must turn.
Se tutto l'indice dovessi leggere | If all the index I should read
Con un equivoco, con un sinonimo | With a misunderstanding, with a synonym
Qualche garbuglio si troverà. | Some confusion will be found.*

Le nozze di Figaro, Atto I, Mozart | The marriage of Figaro, Act I, Mozart

5.1 SUMMARY OF FINDINGS

This thesis studied scalable methods for probabilistic modeling with Gaussian distributions and tractable inference, using efficient storage and computations enabled by TNs. In this context, three contributions were presented (see Figure 5.1), each addressing challenges in large-scale and/or high-dimensional settings. They are related to solving recursive least squares or weighted least squares problems, providing solutions to Bayesian estimation problems such as those outlined in Section (1.1). This chapter summarizes the contributions of this thesis and points out in which way they have made advances compared to the state-of-the-art.

Generally, all algorithms presented in this thesis are developed with a focus on both accuracy and efficiency metrics such that all experiments can be conducted on conventional laptops in a reasonable time. This approach avoids the use of external hardware or cloud computing while also limiting energy consumption.

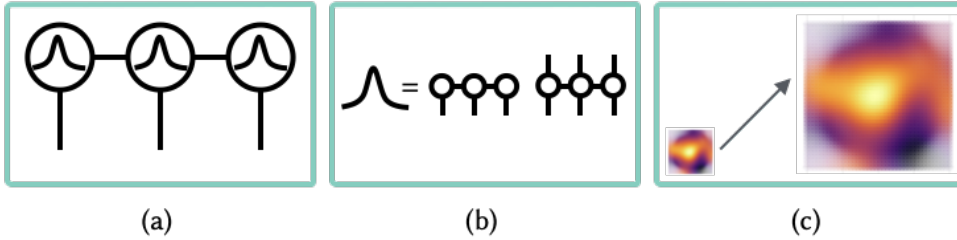


Figure 5.1: Visual summary of three main contributions of this thesis. (a) TT-cores as Gaussian distributions for the development of a Bayesian ALS algorithm, (b) Representing the mean and covariance matrix of a Gaussian distribution as TTs for solving the issue of loss of positive definiteness in TN Kalman filtering, (c) Building scalable magnetic field maps with Gaussian process regression by applying structured kernel interpolation.

5.1.1 DEVELOPMENT OF BAYESIAN ALS ALGORITHM

In the first contribution, we approach the computation of low-rank TNs from a Bayesian perspective. We compute a posterior joint distribution of the TN components by solving multiple Bayesian inference problems sequentially to infer the posterior distribution of each individual TN component. Thus, we introduce a Bayesian framework for the ALS to compute low-rank tensor approximations, extending the state-of-the-art in the following way. The conventional ALS updates each TN component by solving an unregularized least squares problem. This can be interpreted as solving a Bayesian inference problem with an uninformative prior on the TN components. The ALS in a Bayesian framework overcomes this limitation by making it possible to incorporate application-specific prior knowledge for each TN component. In addition, the measurement noise is explicitly considered, and the joint posterior distribution of the TN components offers uncertainty quantification for the low-rank tensor estimate. Regarding complexity, the ALS in a Bayesian framework has the same computational complexity as the conventional ALS, there is only an additional storage cost for storing the covariance matrices of the TN components.

5.1.2 SOLVING THE ISSUE OF LOSS OF POSITIVE DEFINITENESS IN TN KALMAN FILTERING

The state-of-the-art algorithm for Kalman filtering in a TN format, the TNKF, logarithmically compresses the complexity of filtering problems that are computationally untractable in their conventional form. However, the TNKF can become numerically unstable, because it can suffer from loss of positive (semi-) definiteness of covariance matrices involved in the filter. The only setting in which positive (semi-) definiteness is guaranteed is when all ranks of the TTm that represents the covariance matrix are chosen to be one, limiting the accuracy of the approximation. In the second contribution of this thesis, the TNSRKf, we solve this issue by computing square root covariance factors instead of the covariance matrix directly. While the computational cost of the TNSRKf and TNKF are in a similar range, the TNSRKf has the advantage of being more flexible. The TNSRKf allows for choosing higher TN ranks than one for the square-root coefficient factor without the loss of positive (semi-) definiteness. In this way, the covariance matrix is also approximated with higher TN ranks and can result in a better approximation.

5

5.1.3 APPLICATION OF STRUCTURED KERNEL INTERPOLATION TO MAGNETIC FIELD MAPPING

The third contribution of this thesis consists of computing scalable magnetic field maps using approximate GP regression. Computing GP predictions efficiently is achieved by using structured kernel interpolation with derivatives, in combination with Kronecker algebra used in Krylov subspace methods. State-of-the-art methods for computing large-scale magnetic field maps are based on, e.g., downsampling the magnetic field measurements or approximating the GP kernel with a reduced number of basis functions. The limitation of these methods is that the efficiency gain, compared to full GP regression, can result in a significant accuracy loss. A motivation for exploring structured kernel interpolation for magnetic field mapping was that it is a popular approach for large-scale GP problems, but it had not been applied in this context yet. The third contribution showed how maps can be computed in a more efficient way compared to the state-of-the-art, such that more accurate maps can be computed with a limited computational budget.

5.2 FUTURE WORK RECOMMENDATIONS

Based on the conclusions of the contributions of this thesis, this section summarizes possible future work directions, inspired by the findings and future work of the individual contributions.

1. In the context of contribution 2, one extension of the presented work is incorporating a time update into the TNSRKf and applying it to high-dimensional state estimation, where the state can be, e.g., a large field. This can be achieved by, e.g. using a spatio-temporal GP model as described in [1].
2. Representing kernel matrices with TT matrices is challenging due to potential loss of positive (semi-) definiteness, such that in contribution 2, it is proposed to represent the square root covariance factor instead. A future work direction is to investigate if

there are other ways to incorporate positive (semi-) definiteness into TT matrices, e.g. by optimizing on a manifold of positive (semi-) definite TT matrices.

3. In the context of contribution 2, another option is to approximate the covariance matrix with a low-rank matrix computed from truncated square-root covariance factors. These tall matrices can be represented by tall TT matrices, where only one TTm-core has a column index larger than one. First experiments showed, that the rank of the matrix represented by the tall TTm is limited by the choice of the TTm-ranks. A future work direction is to investigate why this is the case and how low-rank matrices can be represented by tall low-rank TT matrices.
4. In contribution 2, the hyperparameters of the squared exponential kernel were trained before deploying the recursive algorithm. In contribution 3, the hyperparameters were trained on a subset of the entire dataset. A possible future research direction is to focus on truly online scenarios, where hyperparameters are computed without the knowledge of future data, in large-scale and high-dimensional settings.
5. In contribution 1, TN components were represented as Gaussian distributions, and the low-rank tensor approximation problem was solved with exact inference. A future work direction could be to consider non-Gaussian distributions and approximate inference.
6. In contribution 3, scalable magnetic field maps were computed with SKI using a batch of measurements. Future work could be to apply an online SKI algorithm such that this mapping strategy can be incorporated into, e.g., a simultaneous localization and mapping (SLAM) algorithm. In addition, other approximate GP methods could be explored for efficient magnetic field mapping.
7. In contribution 3, one sensor of a motion capture suit was used to build a scalable magnetic field map. Future work could be to explore how multiple sensors on the suit can be leveraged. It can be investigated, e.g. how having multiple magnetometers located on body segments in a flexible array kind of fashion can improve map quality and localization accuracy, or how different sensors revisiting the same positions and forming loop closures can be beneficial.
8. All three contributions use synthetically generated data for conducting controlled experiments that illustrate a specific point. For high-dimensional problems, however, it is challenging to sample data that has some realistic meaning. Synthetic data generation in high dimensions could be another future work direction.

REFERENCES

- [1] Simo Särkkä, Arno Solin, and Jouni Hartikainen. Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, 2013.

CURRICULUM VITÆ

Clara Myria MENZEN


18-03-1992 Born in Rheda-Wiedenbrück, Germany.

EDUCATION

- 2019–2024 PhD. in Control Engineering
Delft Center for Systems and Control, Delft University of Technology,
Delft, the Netherlands
Thesis: Tensor networks for scalable probabilistic modeling
Promoters: prof. dr. ir. Jan-Willem Wingerden,
dr. Manon Kok,
dr. ir. Kim Batselier.
- 2016–2019 M.Sc. Engineering Science
Technical University of Berlin, Germany
Erasmus exchange at Polytechnic University of Catalonia, Barcelona, Spain.
- 2012–2016 B.Sc. Energy and Process Engineering
Technical University of Berlin, Germany
Erasmus exchange at Swiss Federal Institute of Technology, Zurich, Switzerland.
- 1998–2011 German School of Milan, Italy (2003–2011)
Ratsgymnasium Wiedenbrück, Germany (2002–2003)
Parkschule Rheda, Germany (1998–2002)

LIST OF PUBLICATIONS

1. *Clara Menzen, Manon Kok, Kim Batselier* (2020) Alternating linear scheme in a Bayesian framework for low-rank tensor approximation, *SIAM Journal on Scientific Computing*, 44(3), A1116-A1144.
2. *Clara Menzen, Manon Kok, Kim Batselier* (2025) Tensor network square root Kalman filter for online Gaussian process regression, *Automatica*, accepted for publication.
3. *Clara Menzen, Marnix Fetter, Manon Kok* (2023) Large-scale magnetic field maps using structured kernel interpolation for Gaussian process regression, *Proceedings of the 26th International Conference on Information Fusion (FUSION)*, Charleston, South Carolina, USA, (pp. 1-7). IEEE.
4. *Eva Memmel, Clara Menzen, Jetze Schuurmans, Frederiek Wesel, Kim Batselier* (2024) Position: Tensor Networks are a Valuable Asset for Green AI, *Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235.
5. *Clara Menzen, Eva Memmel, Kim Batselier, Manon Kok* (2023) Projecting basis functions with tensor networks for Gaussian process regression, *Proceedings of the 22nd IFAC World Congress*, Yokohama, Japan, *IFAC-PapersOnLine*, 56(2), 7288-7293.
6. *Eva Memmel, Clara Menzen, Kim Batselier* (2023) Bayesian Framework for a MIMO Volterra Tensor Network, *Proceedings of the 22nd IFAC World Congress*, Yokohama, Japan, *IFAC-PapersOnLine* 56(2), 7294-7299.

 Included in this thesis.