Fine Element solution of 2D Laplace equation using Tensor Methods

Michael Scherbela

April 2022

This project was submitted as part of the coursework for the lecture *250085 VU Tensor methods for data science and scientific computing*. All accompanying code [3] can be found on github.

1 Introduction

Finding accurate solutions to partial differential equations (PDEs) is of high relevance for in many scientific and engineering disciplines. Examples include the Navier-Stokes equation for fluid dynamics, the Schrödinger equation for quantum physics or the Laplace equation. While for some PDEs the fundamental challenges arise from non-linearities or the high dimensionality of the solution, there are several linear, low-dimensional PDEs which can nevertheless be hard to solve for specific boundary conditions or inhomogeneities. The example to be considered in this project is the one- and two-dimensional Laplace equation

$$-\nabla^2 u(x,y) = f(x,y), \tag{1}$$

on the domain $\mathcal{D} = [0,1]^d$ with boundary conditions:

$$\begin{aligned} u(x,0) &= 0 \\ u(y,0) &= 0 \\ \partial_x u|_{(1,y)} &= 0 \\ \partial_y u|_{(x,1)} &= 0 \end{aligned}$$

When solving this PDE using Finite Element Methods (FEM), one key step is typically finding a suitable mesh of finite elements that allow accurately representation of u and f. One would like the mesh to be as fine as possible to achieve high accuracy, but this comes at high computational cost. One common route to tackle this trade-off is to use adaptive meshing algorithms that use a fine mesh in parts of the domain where u or f vary rapidly and use a coarser mesh in smoother regions of the domain. An alternative approach used in this project is the use of tensor trains to represent u and f on a very fine, virtual grid. On a very high-level the approach (exemplified on a 1D-examples) is as follows:

1. Represent all objects (e.g. u, f) on a simple, very fine, equidistant grid consisting of 2^L points
2. Reinterpret the resulting vectors as L-dimensional tensors of shape [2, ..., 2] and represent these tensors efficiently as low-rank tensor trains
3. Ensure that all algorithms (e.g. encoding f, the actual solver, post-processing of the solution, etc.) work directly on the compressed tensor-train format (they have been "tensorized") and do not require explicit full evaluation of the objects

Once all algorithms have been tensorized it is never necessary to build the full 2^L dimensional objects. Thus even though the mesh may virtually consist of 2^L many grid-points (and thus

become exponentially fine with increasing number of levels L) the actual computational cost will only scale polynomially with L.

2 Mathematical Foundations

2.1 Weak formulation of PDE

The following provides a rough, high-level motivation of the weak formulation of the PDE: Solving the original PDE $-\nabla^2 u = f$ requires the solution u to be twice differentiable and thus rules out simple ansätze like piece-wise linear functions. This can be circumvented by not solving the original PDE, but its weak formulation:

$$\begin{aligned} -\nabla^2 u &= f \\ -\int_{\mathcal{D}} u\nabla^2 u &= \int_{\mathcal{D}} uf \\ -\int_{\mathcal{D}} \nabla(u\nabla u) + \int_{\mathcal{D}} (\nabla u)^T(\nabla u) &= \int_{\mathcal{D}} uf \\ -\int_{\partial\mathcal{D}} u(\vec{n}\nabla\mathcal{D}) + \int_{\mathcal{D}} (\nabla u)^T(\nabla u) &= \int_{D} uf \\ \int_{\mathcal{D}} (\nabla u)^T(\nabla u) &= \int_{\mathcal{D}} uf \end{aligned} \tag{2}$$

We use Gauss law to transform one of the integrals over the domain into an integral over its boundary and given our boundary conditions either u or the projection of ∇u on the boundary's normal vector n will vanish. When introducing a basis the final equation can be cast in matrix form, resulting in

$$\begin{aligned} uD^T Du &= uRf \\ D^T Du &= Rf \\ Au &= Ru \end{aligned} \tag{3}$$

where D corresponds to the gradient operator and A corresponds to the laplace operator ∇^2.

2.2 Conditioning

The linear system $Au = Ru$ may be ill conditioned and thus impossible to solve numerically given only finite precision arithmetic (cf. Fig 2). This problem can be alleviated by preconditioning the linear system using a suitable matrix C:

$$B = CAC \tag{4}$$

$$Bv = CRf$$

$$u = Cv \tag{5}$$

The matrix C is chosen s.t. $B = CAC$ is well conditioned. We then solve this well conditioned system $Bv = CRf$ instead of the original system and can obtain the desired solution by multiplying with C.

2.3 Tensor-Train decompositions

Throughout the computation all objects must be represented as tensor trains (TTs). A comprehensive expossition and derivation of this topic can be found in [1]. The most relevant decompositions for this work are listed here for completeness. All tensor cores in this section are assumed to be 4-dimensional: 2 rank indices + 2 mode indices. The outer matrix shape correspond to the TT-rank dimensions, the contents of each matrix elements correspond to the mode dimensions. The

matrices corresponding to the rank dimensions will be denoted by square brackets, the matrices corresponding to mode dimensions by round parentheses. The following special products will be used:

- Mode product $\bullet$: Kronecker product with respect to TT-dimensions, matrix product with respect to mode-dimensions
- Strong Kronecker product $\bowtie$: Matrix product with respect to TT-dimensions, Kronecker product with respect to mode-dimensions

Transposition happens with respect to mode dimensions, not with respect to TT-rank-dimension. For a more thorough outline refer to [1].

2.4 Auxiliary tensor cores

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad J = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{6}$$

$$\hat{U} = \begin{bmatrix} I & J^T \\ 0 & J \end{bmatrix} \quad \hat{X} = \frac{1}{2}\begin{bmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} & \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \begin{pmatrix} 2 \\ 1 \end{pmatrix} \end{bmatrix} \quad \hat{L}_\flat = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \quad \hat{Z}_\flat = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{K_0} = \frac{1}{2}\begin{bmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{bmatrix} \quad \hat{K_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\hat{W_0} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \quad \hat{W_1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \tag{7}$$

$$\hat{V_0} = \frac{1}{4}\begin{bmatrix} \begin{pmatrix} 2 & 4 \\ 2 & 4 \end{pmatrix} & \begin{pmatrix} 0 & 2 \\ 0 & 2 \end{pmatrix} & & \\ \begin{pmatrix} 2 & 0 \\ 2 & 0 \end{pmatrix} & \begin{pmatrix} 4 & 2 \\ 4 & 2 \end{pmatrix} & & \\ \begin{pmatrix} -1 & -2 \\ 1 & 2 \end{pmatrix} & \begin{pmatrix} 0 & -1 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \\ \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix} & \begin{pmatrix} -2 & -1 \\ 2 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 1 \\ 2 & 1 \end{pmatrix} \end{bmatrix} \tag{8}$$

$$\hat{V_1} = \frac{1}{4}\begin{bmatrix} \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 1 \\ 2 & 1 \end{pmatrix} \end{bmatrix} \tag{9}$$

$$\hat{U_\flat} = \hat{U} \bullet \hat{U} \quad X_\flat = \hat{X} \bullet \hat{X}^T \tag{10}$$

2.5 Operators without preconditioning

The following matrices - represented as TT - are required to build and solve the non-preconditioned PDE in 1D: M^T maps a function given in a 2-element Legendre basis to the nodal basis of hat functions. $\hat{D}$ performs the 1st derivative in the nodal basis, A corresponds to the Laplace operator, i.e. the left-hand-side of the linear system. G is the gram matrix for functions represented in the

Legendre basis, $\hat{\Phi}$ is the mass matrix and R forms the right-hand-side of the linear system to be solved.

$$\hat{M} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \bowtie \hat{U} \bowtie \ldots \bowtie \hat{U} \bowtie \begin{bmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \\ \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} \end{bmatrix} \tag{11}$$

$$\hat{D} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \bowtie \hat{U} \bowtie \ldots \bowtie \hat{U} \bowtie \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{12}$$

$$\hat{A} = 2^{-L} \hat{D}^T \hat{D} \tag{13}$$

$$\hat{G} = \begin{bmatrix} I \end{bmatrix} \bowtie \ldots \bowtie \begin{bmatrix} I \end{bmatrix} \begin{bmatrix} \begin{pmatrix} 1 & \\ & \frac{1}{3} \end{pmatrix} \end{bmatrix} \tag{14}$$

$$\hat{\Phi} = \hat{M}^T \hat{G} \hat{M} \tag{15}$$

$$R = \hat{M}^T \hat{G} \tag{16}$$

Since $\hat{U}$ has rank 2, also $\hat{M}$ and $\hat{D}$ have TT-ranks of at most 2 and thus $\hat{A}$ and $\hat{\Phi}$ have TT-ranks of at most 4. The 2D versions of these operators can be obtained as Kronecker products of the 1D operators:

$$M = \hat{M} \otimes \hat{M} \quad D_x = \hat{D} \otimes \hat{M} \quad D_y = \hat{M} \otimes \hat{D} \quad A = \hat{A} \otimes \hat{\Phi} + \hat{\Phi} \otimes \hat{A} \tag{17}$$

2.6 Operators with preconditioning

One effective way to precondition the linear system is the so-called BPX preconditioner, which can be expressed as a low-rank TT:

$$C = \begin{bmatrix} L_\flat & L_\flat \end{bmatrix} \bowtie C_1 \bowtie \ldots C_L \bowtie \begin{bmatrix} \\ Z_\flat \end{bmatrix} \tag{18}$$

$$C_l = \begin{bmatrix} U_\flat & 2^{-l} U_\flat \\ & \frac{1}{2} X_\flat \end{bmatrix} \tag{19}$$

In this expression $L_\flat$, $Z_\flat$, $U_\flat$ and $X_\flat$ are Kronecker products of their 1D counterparts, i.e. for 2D:

$$L_\flat = \hat{L}_\flat \otimes \hat{L}_\flat \quad Z_\flat = \hat{Z}_\flat \otimes \hat{Z}_\flat \quad U_\flat = \hat{U}_\flat \otimes \hat{U}_\flat \quad X_\flat = \hat{X}_\flat \otimes \hat{X}_\flat \tag{20}$$

whereas for 1D:

$$L_\flat = \hat{L}_\flat \quad Z_\flat = \hat{Z}_\flat \quad U_\flat = \hat{U}_\flat \quad X_\flat = \hat{X}_\flat \tag{21}$$

This decomposition has ranks 2^{2D+1}, i.e. scales exponentially with the dimensionality of the problem but is independent of the refinement level L.

Having tensorized expressions for all relevant operators (e.g. A, M, D) and the preconditioner C, one could in principle assemble the preconditioned system via matrix products. However this is not advisable for multiple reasons: First, both the matrix A as well as the matrix C are ill-conditioned. Multiplying them in finite precision can lead to amplified errors. Second, assembling CAC requires 2 matrix multiplications, each of which leads to a multiplication of the ranks. For example, assembling CAC naively for a 2D system would thus lead to very large ranks of $32 \times 32 \times 32 = 32768$. A better approach is to express the matrices $Q = MC$ and $Q' = DC$ directly as tensor trains and form the components of the linear system from these matrices. Since Q and Q' are better conditioned, numerical errors are less severe and since building $B = Q'^T Q$ only requires 1 matrix multiplication also the ranks stay substantially lower.

Following [1], the preconditioned mapping $Q = MC$ and the preconditioned derivative $Q = DC$ are given by:

$$Q = \begin{bmatrix} L_\flat & L_\flat \bowtie W \end{bmatrix} \bowtie Q_1 \bowtie \ldots \bowtie Q_L \bowtie \begin{bmatrix} \\ K \end{bmatrix} \tag{22}$$

$$Q_l = \frac{1}{2}\begin{bmatrix} U_\flat & U_\flat \bowtie W \\ & \frac{1}{2}V \end{bmatrix} \tag{23}$$

Here the tensor cores W, V and K are all Kronecker products of their respective 1D versions:

$$W = \hat{W}_0 \otimes \hat{W}_0 \quad V = \hat{V}_0 \otimes \hat{V}_0 \quad K = \hat{K}_0 \otimes \hat{K}_0 \tag{24}$$

Note that this deviates slightly from the expression given in [1], where the $U_\flat \bowtie W$ factor has an additional factor of 2^{-l}.

The tensor $Q_x = D_x C$ which computes the first derivative with respect to x in the BPX basis is given similarly by:

$$Q_x = \begin{bmatrix} L_\flat & L_\flat \bowtie W_x \end{bmatrix} \bowtie Q_{x1} \bowtie \ldots \bowtie Q_{xL} \bowtie \begin{bmatrix} \\ K_x \end{bmatrix} \tag{25}$$

$$Q_{xl} = \begin{bmatrix} U_\flat & U_\flat \bowtie W_x \\ & \frac{1}{2}V_x \end{bmatrix} \tag{26}$$

Here the tensor cores W_x, V_x and K_x are all Kronecker products of their respective 1D versions:

$$W_x = \hat{W}_1 \otimes \hat{W}_0 \quad V_x = \hat{V}_1 \otimes \hat{V}_0 \quad K_x = \hat{K}_1 \otimes \hat{K}_0 \tag{27}$$

Note that this again deviates slightly from [1], in this case by the factor of $\frac{1}{2}$ in front of V_x. Q_y can be assembled analogously by swapping the order in the Kronecker products. The tensors Q and Q_x have ranks 32 and 24 respectively.

3 Results for a 1D problem

3.1 Norms of interpolants

Before solving any system it is worthwhile to ensure that functions can be accurately represented as a TT and that norms can be accurately computed from these decompositions. As example, the following pair of functions is being used

$$u(x) = (1 - 3x + 2x^2)\sin(2\pi x)$$
$$f(x) = -u''(x) = (4 - 4\pi^2 + 12\pi^2 x - 8\pi^2 x^2)\sin(2\pi x) + (-12\pi + 16\pi x)\cos(2\pi x)$$

The solution u has the following norms:

$$||u||_{L2}^2 = \int_0^1 u^2 = -\frac{3}{16\pi^2} + \frac{3}{16\pi^4} + \frac{1}{15}$$
$$||u||_{H1}^2 = \int_0^1 (u')^2 = -\frac{1}{4\pi^2} + \frac{4\pi^2}{15} + \frac{5}{12}$$

The H1-norm can be computed in 2 distinct ways: Either directly as $u^T A u$ or alternatively as $|Du|^2$. The latter has the advantage that we only apply the ill-conditioned matrix D once instead of twice in $A = D^T D$. Calculating the norm of $|Du|$ (or any TT) can be done in a stable manner via orthogonalization. Fig. 1 shows how the error of the norm decreases exponentially with increasing refinement level. For the L2 norm it comes close to machine precision and is stable also for increasing values of L. Calculating the H1 norm from u on the other hand is not stable, because the condition number of D increases exponentially with L (see. Fig. 2). Computing the H1 norm via orthogonalization is stable for longer, because it does not square the condition number of D but the precision eventually deteriorates as well.

To investigate this ill-conditioning, one can compute the condition numbers of the relevant operators by fully evaluating the tensor trains (which is only possible for small L) and computing their spectrum of singular values. Fig. 2 shows that the matrix M which is required to calculate the L2 norm is fully stable. The matrices D and A which are required to calculate the H1 norm with and

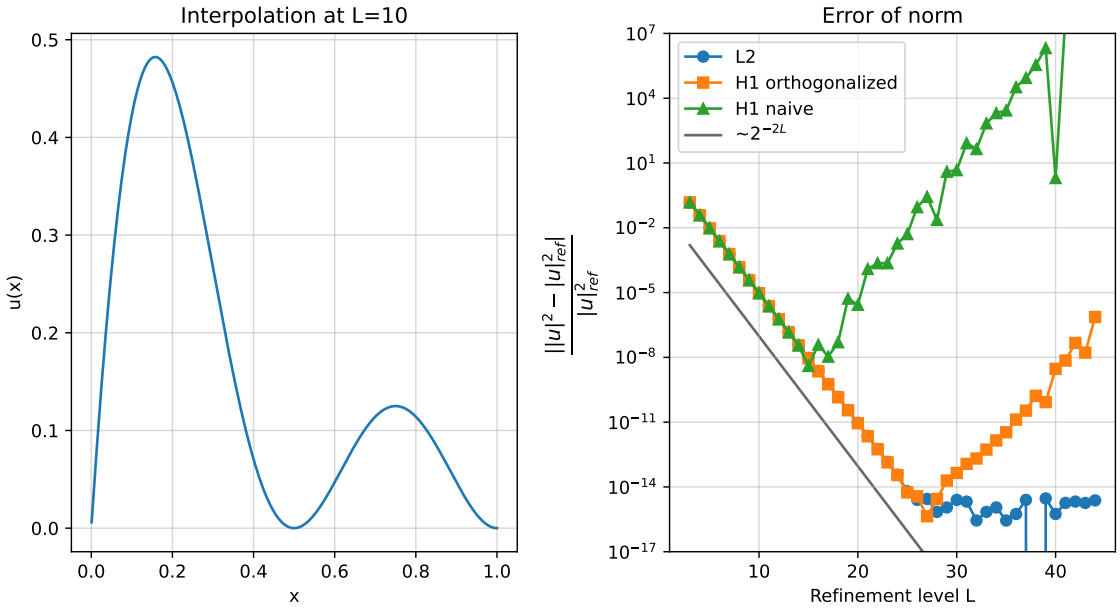

Figure 1: Error of $||u||^2$ as a function of refinement level

without orthogonalization show exponentially growing condition numbers, explaining the deterioration of the H1 norm: Small errors in u lead to exponentially large errors in Du and Au. This is not only problematic when calculating the norms, but also problematic when post-processing a solution u given in the nodal basis. Calculating the derivative or curvature of the solution is unstable and thus not possible for large L.

The preconditioned versions on the other hand do not exhibit exponentially growing condition numbers for the Laplace B or the derivative Q'. One would thus expect to see stable H1 norms when calculating these from solutions given in the BPX basis. However this comes at the price of exponentially growing condition numbers for Q and C. This could in principle problematic when solving the system since the matrix C needs to be applied twice: Once to map the RHS of the system into the BPX basis and a second time after solving the system to map the result back to the nodal basis. The resulting relative error of these mappings are in principle bounded by the condition number of C:

$$u = Cv \tag{28}$$

$$\frac{||u - u_0||}{||u_0||} = \frac{||C(v - v_0)||}{||Cv_0||} \leq \frac{||C||_{\max}||v - v_0||}{||C||_{\min}||v_0||} = \text{cond}(C)\frac{||v - v_0||}{||v_0||} \tag{29}$$

The maximum relative error in u is thus obtained when the error in v lies along an eigenvector of C with a large eigenvalue and the actual solution lies along an eigenvector of C with minimal eigenvalue. It is thus instructive to understand the nature of these eigenvectors. Fig. 3 shows that smooth functions v correspond to large eigenvalues of C whereas the smallest eigenvalues correspond to noisy high-frequency functions. Thus the worst bound will never be obtained in practice as long as the function being approximated is sufficiently smooth.

To solve the PDE in 1D, the AMEN solver [2] from the package TTPY is being used. The solver minimizes the residual by alternatingly optimizing one of the factors of the tensor train using least squares minimization. The number of sweeps was limited to 40 sweeps, the TT-rank of the solution was increased by 2 for every sweep. Fig. 4 shows that initially the accuracy increases exponentially with increasing refinement level L – both the raw as well as the preconditioned system. As the operator A becomes increasingly ill-conditioned the non-preconditioned solution requires more and more iterations to converge to a solution, and ultimately does not reach the

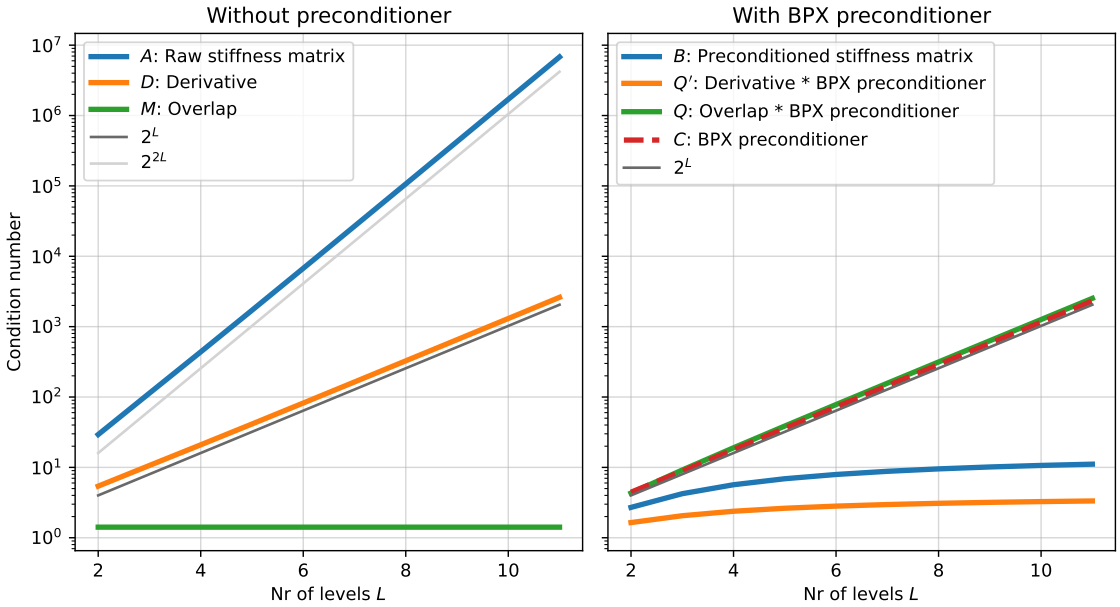

Figure 2: Condition number of various operators as a function of L for a 1D system. Left: Without preconditioning, right: With BPX preconditioner

target residual within 40 sweeps. The error of the solution then increases exponentially with L. On the contrary the BPX-preconditioned system reaches relative accuracies of 10^{-11}, relatively close to machine precision. Note that while calculating the H1-norm from a solution u given in the nodal basis is unstable (cf. Fig. 1), the H1-norm is stable when calculating it directly via the solution in the BPX basis.

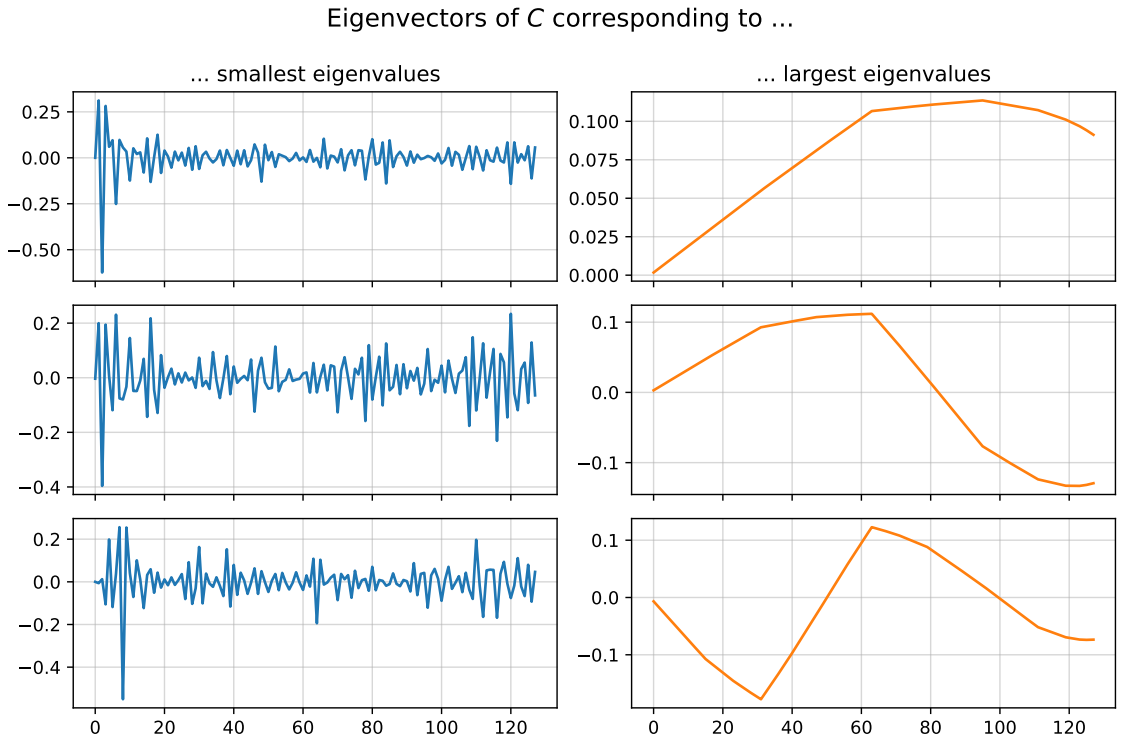

Figure 3: Eigenfunctions of C corresponding to the smallest 3 and largest 3 eigenvalues

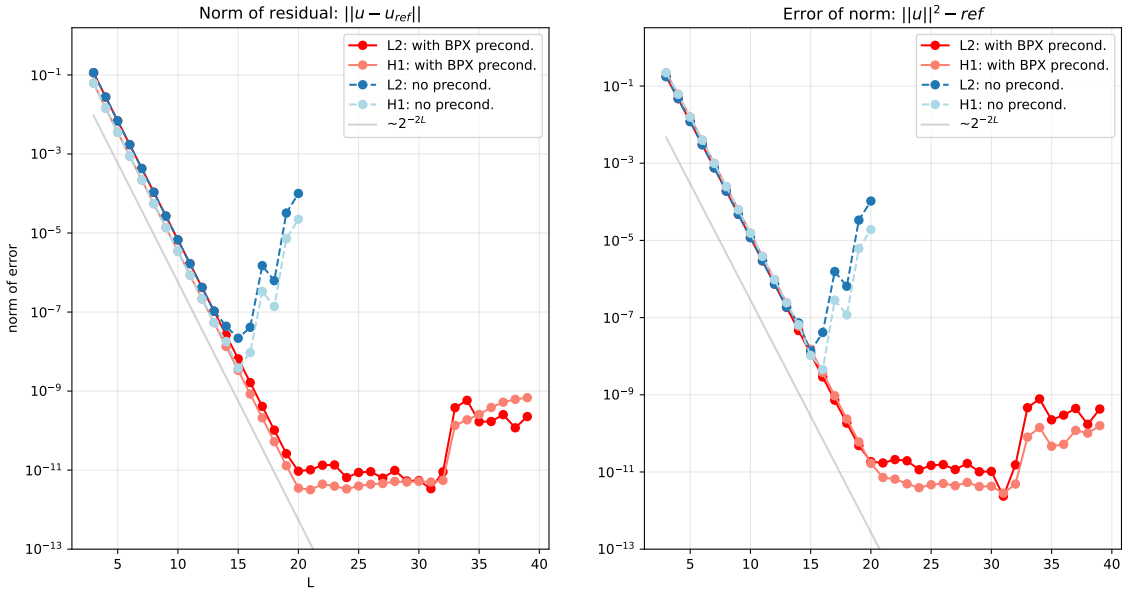

Figure 4: Accuracy of 1D solutions depending on refinement level L. Left: Norm of residual of the obtained solution vs. the direct refinement of reference solution. Right: Error in norm of obtained solution vs. the analytically computed norm

4 Results for a 2D problem

For the 2D case the experiments show analogous results. In practice the tensors required ranks are significantly larger for the 2D case than for the 1D case, resulting in larger computational cost. Similarly to the 1D case, one can at first validate that norms can be accurately calculated from a given solution. As a test function the following function is being used:

$$u(x,y) = (-x + 5x^2 - 3x^3)(1 - 3y + 2y^2)\sin(2\pi y) \tag{30}$$

This function has the following norms:

$$||u||_{L2}^2 = \int_{\mathcal{D}} u^2 = \left(-\frac{3}{16\pi^2} + \frac{3}{16\pi^4} + \frac{1}{15}\right)\frac{67}{210}$$
$$||u||_{H1}^2 = \int_{\mathcal{D}} (\nabla u)^T(\nabla u) = \frac{2144\pi^6 + 5926\pi^4 + 7245 - 9255\pi^2}{25200\pi^4}$$

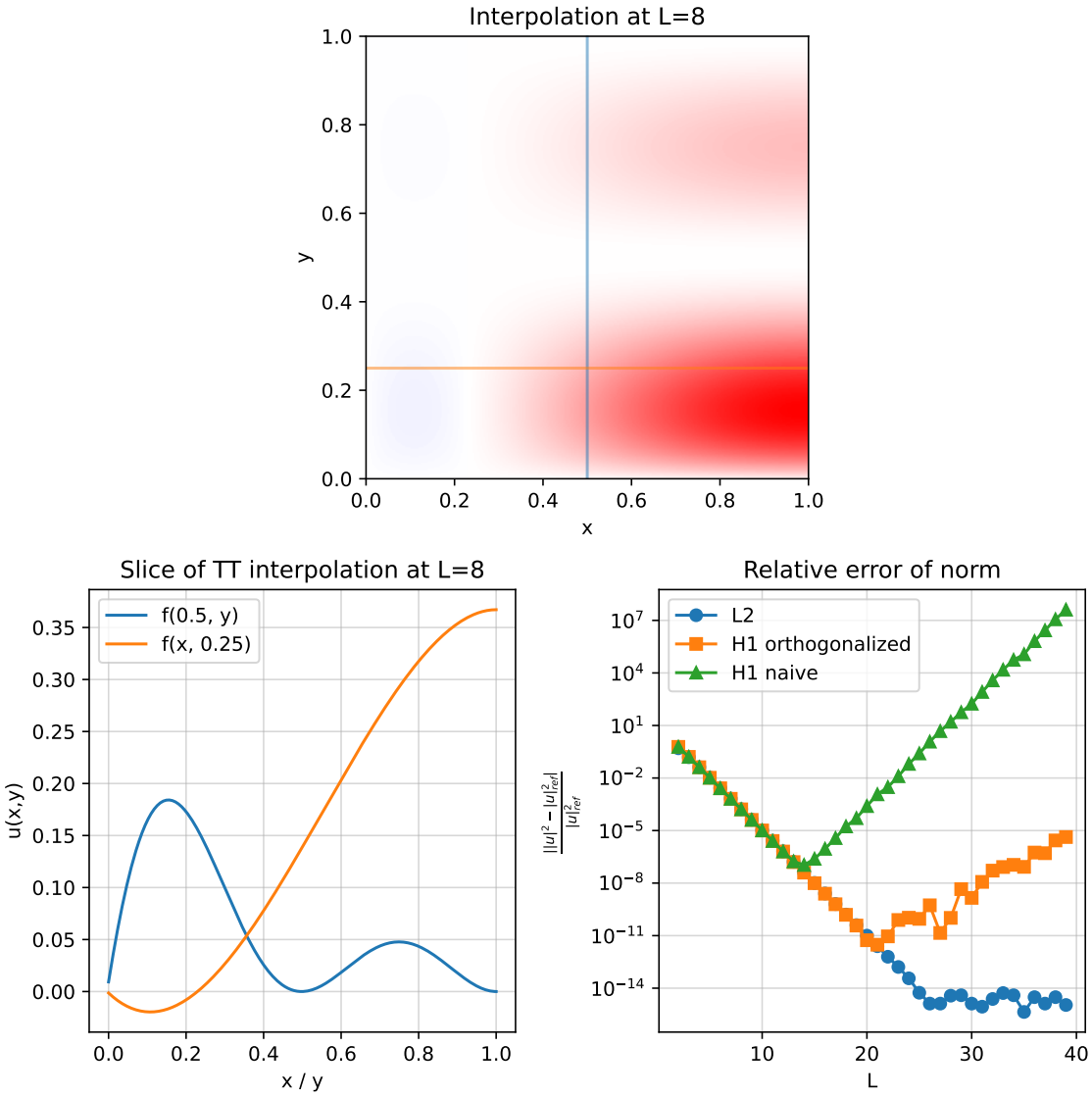

Figure 5: Accuracy of 2D solutions depending on refinement level L. Top: Visualization of example function $u(x,y)$ Left: Plot of slices of $u(x,y)$ along constant x and constant y respectively. Right: Error of norm computed from iterated refinement of u vs. the analytically computed norms

Fig. 5 shows that analogously to the 1D case, calculating the norm of a given function is stable for the L2 norm, but unstable for the H1 norm with increasing L. Similar to the 1D case there are two ways to compute the norm: Naively as $u^T Au$ or in a more stable manner as:

$$\begin{aligned}
||u||_{H1}^2 &= u^T Au \\
&= u^T (D^T \otimes M^T)(I \otimes G)(D \otimes M) + (M^T \otimes D^T)(G \otimes I)(M \otimes D)u \\
w_y &:= (I \otimes \sqrt{G})(D \otimes M) \\
w_x &:= (\sqrt{G} \otimes I)(M \otimes D) \\
u^T Au &= |w_x|^2 + |w_y|^2
\end{aligned}$$

Since the gram matrix G is diagonal and positive definite, its square root can be easily computed. The norms of w_x and w_y can again be computed in a stable manner using TT orthogonalization. Same as in the 1D case, the naive method squares the condition number leading to instability already at modest L, while orthogonalization leads to stable H1-norms up to $L \approx 20$.

Analagously to the 1D case, one can analyze the condition number of the relevant matrices as a function of L. Fig. 6 shows that the Laplace operator is stabilized by the preconditioner.

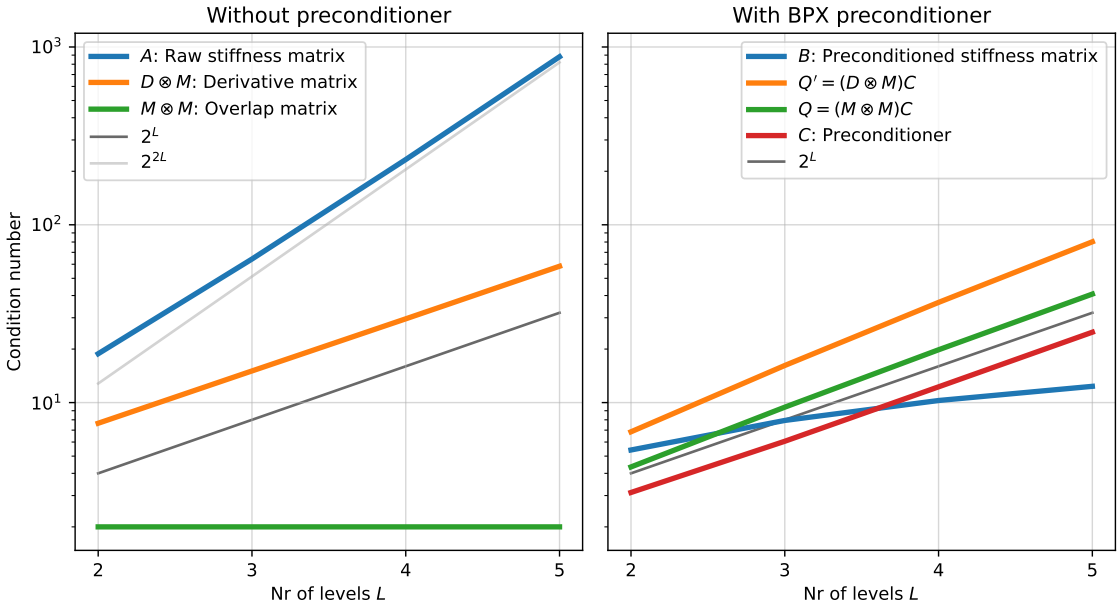

Figure 6: Condition number of various operators as a function of L for a 2D system. Left: Without preconditioning, right: With BPX preconditioner

Finally one can solve the actual PDE using the AMEN solver and investigate the solution's accuracy as a function of L. As in the 1D case, the linear system becomes unstable already at $L \approx 12$ while the preconditioned system reaches relative accuracies of $\approx 10^{-11}$. Unlike in the 1D case, the error slowly increases again at large L even for the preconditioned system. A possible explanation could be the errors introduced by the truncated TT-ranks.

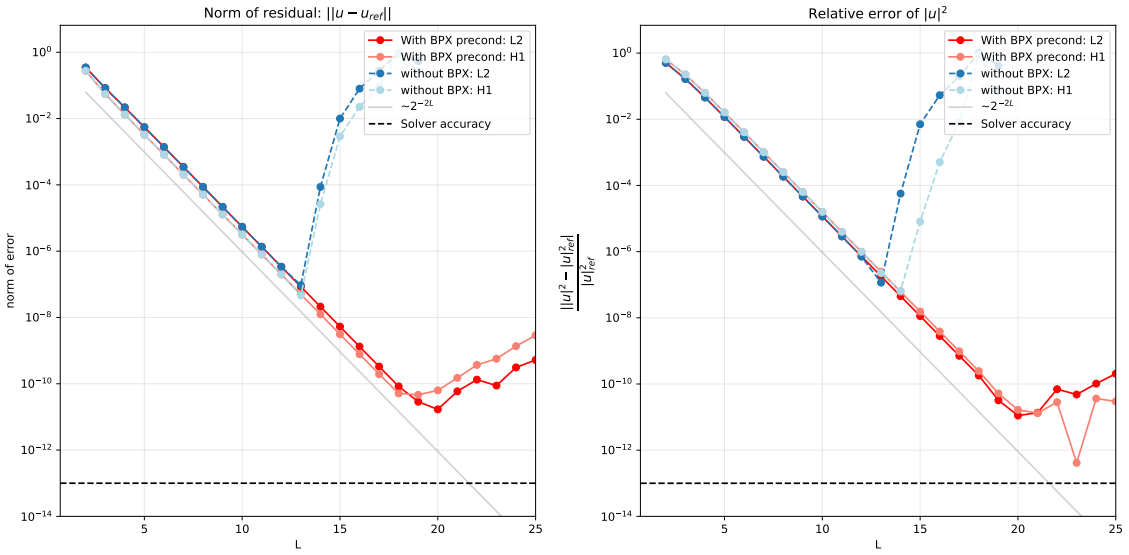

Figure 7: Accuracy of 2D solutions depending on refinement level L. Left: Norm of residual of the obtained solution vs. the direct refinement of reference solution. Right: Error in norm of obtained solution vs. the analytically computed norm

References

[1] Markus Bachmayr and Vladimir Kazeev. Stability of Low-Rank Tensor Representations and Structured Multilevel Preconditioning for Elliptic PDEs. *Foundations of Computational Mathematics*, 20(5):1175–1236, October 2020.

[2] 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, jan 2014.

[3] M. Scherbela. Laplacesolvermps. `https://github.com/MScherbela/LaplaceSolverMPS`, 2022.