Skoltech
Skolkovo Institute of Science and Technology
Skolkovo Institute of Science and Technology

STATISTICAL INFERENCE AND MACHINE LEARNING IN NUMERICAL LINEAR ALGEBRA

Doctoral Thesis

by

VLADIMIR FANASKOV

DOCTORAL PROGRAM IN MATHEMATICS AND MECHANICS

Supervisor
Associate Professor Aslan Kasimov

Moscow – 2022

I hereby declare that the work presented in this thesis was carried out by myself at Skolkovo Institute of Science and Technology, Moscow, except where due acknowledgement is made, and has not been submitted for any other degree.

Vladimir Fanaskov

Aslan Kasimov

Abstract

We explore how data-driven approaches can help to enhance iterative methods used to solve linear problems. More specifically, we are interested in linear problem $\boldsymbol{Ax} = \boldsymbol{b}$ where $\boldsymbol{A}$ is a large sparse square matrix typically originated from finite difference or finite element discretization of a partial differential equation. Partial differential equations that we consider are mainly relevant to the different subfields of computational continuum mechanics that include mass and heat transfer, resolution of boundary layers, a vibration of thin plates, propagation of acoustic waves, the flow of incompressible fluid, etc. On the side of data science, we use Bayesian data analysis (Part I) and machine learning (Part II). In the first part, we study three topics: belief propagation algorithms, probabilistic projection methods, probabilistic uncertainty quantification for deterministic algorithms. In the second part, we consider the automatic construction of solvers and preconditioners by unsupervised training, neural network architecture for multigrid methods, and online optimization of iterative methods with the help of reinforcement learning. All our approaches are illustrated on a large set of physically relevant linear problems. The results demonstrate that data-driven approaches can be superior to classical methods from numerical linear algebra.

Publications

[Fan21a] Vladimir Fanaskov. "Neural Multigrid Architectures". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533736.

[Fan21b] Vladimir Fanaskov. "Uncertainty calibration for probabilistic projection methods". In: *Statistics and Computing* 31.5 (2021), pp. 1–17. DOI: https://doi.org/10.1007/s11222-021-10031-9. URL: https://link.springer.com/article/10.1007/s11222-021-10031-9.

[OF21] Ivan Oseledets and Vladimir Fanaskov. "Direct optimization of BPX preconditioners". In: *Journal of Computational and Applied Mathematics* (2021), p. 113811. DOI: https://doi.org/10.1016/j.cam.2021.113811. URL: https://www.sciencedirect.com/science/article/pii/S0377042721004337.

[Fan22] Vladimir Fanaskov. "Gaussian belief propagation solvers for nonsymmetric systems of linear equations". SIAM Journal on Scientific Computing. 2022. URL: https://epubs.siam.org/doi/abs/10.1137/19M1275139.

Conference presentations

- V. Fanaskov. “Neural multigrid architectures”, International Joint Conference on Neural Networks, 2021
- V. Fanaskov. “Iterative Methods in the Multi-Armed Bandit Setting”, SIAM Conference on Applied Linear Algebra, 2021
- V. Fanaskov. “Using Dynamic Mode Decomposition to speed up Approximate Bayesian Computing for PDE-based generative models”, Machine Learning Summer School, 2019
- V. Fanaskov. “Reinforced linear algebra”, Gen-Y, 2019

Contents

Chapter 1

Introduction

1.1 What this thesis is about

When one needs to solve a large sparse linear system, i.e., to find $\boldsymbol{x}$ such that

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{A} \in \mathbb{R}^{N\times N},\ \boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^{N}, \tag{1.1}$$

the best option often is to resort to iterative techniques [Saa20], [Hac16].

The important class of iterative methods is given by fixed-point iterations with linear contraction mapping. The method is defined by the pair of matrices $\boldsymbol{N}$ and $\boldsymbol{M}$ and the so-called first normal form [Hac16, Equation (2.8)] of these iterations reads

$$\boldsymbol{x}^{n+1} = \boldsymbol{M}\boldsymbol{x}^{n} + \boldsymbol{N}\boldsymbol{b},\ \boldsymbol{M} = \boldsymbol{I} - \boldsymbol{N}\boldsymbol{A}. \tag{1.2}$$

The latter condition ensures that the fixed point is given by the exact solution of Eq. (1.1). We also require that $\|\boldsymbol{M}\| < 1$ in some norm, for iterations defined in Eq. (1.2) to be convergent.

Properly designed iterative methods offer many benefits including

1. $O(N)$ memory footprint;
2. possibility to exploit initial guess;
3. $O(N)$ computational complexity per iteration;
4. opportunity to stop iterations and use approximate solution.

However, there are two principal questions regarding Eq. (1.2) and other iterative methods:

1. How to construct an efficient iterative method?

2. How to decide when the approximation $\boldsymbol{x}^n$ is accurate enough to stop the iterations?

These two questions are not novel and were extensively studied in the literature. The main goal of the present thesis is to research them from the perspective of Bayesian statistics and machine learning.

The structure of the rest of the introduction is as follows. In Section 1.2 we provide examples of classical iterative methods. The performance of novel iterative methods introduced in the thesis is compared against those well-established classical techniques throughout the thesis. Section 1.3 contains examples of sparse linear problems Eq. (1.1) later used for tests of iterative methods. In the last Section 1.4 we sketch the structure and the results of the thesis along with the methods from Bayesian statistics and machine learning that we use.

1.2 Classical iterative methods

The assortment of iterative methods is vast. Here we briefly describe three major classes of those methods: relaxation, projection, and multigrid.

1.2.1 Relaxation methods

To describe relaxation methods we use second normal form [Hac16, Equation (2.10)] of iterations:

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \boldsymbol{N}\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^n\right). \tag{1.3}$$

Observe, that if we take $\boldsymbol{N} = \boldsymbol{A}^{-1}$, the iterative method converges after a single iteration. So, the one way to construct good iterative method is to pick $\boldsymbol{N} \simeq \boldsymbol{A}^{-1}$. Relaxation methods use fairly unsophisticated approximations to $\boldsymbol{A}^{-1}$ [Saa20, p. 4.1]:

Modified Richardson — $\boldsymbol{N} = \theta\boldsymbol{I}$, for some $\theta \in \mathbb{R}$.

Jacobi — $\boldsymbol{N} = \boldsymbol{D}^{-1}$, where $\boldsymbol{D}$ is a diagonal part of $\boldsymbol{A}$.

Gauss-Seidel — $\boldsymbol{N} = \boldsymbol{\Delta}^{-1}$, where $\boldsymbol{\Delta}$ is the upper or the lower triangular part of $\boldsymbol{A}$.

We also define successive over-relaxation method (SOR):

$$\boldsymbol{x}^{n+1} = \omega\boldsymbol{x}^{\mathrm{GS}} + (1-\omega)\boldsymbol{x}^n,\ \boldsymbol{x}^{\mathrm{GS}} = \boldsymbol{x}^n + \boldsymbol{\Delta}^{-1}\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^n\right),\ \omega \in \mathbb{R}. \tag{1.4}$$

It is also possible to define block variants of relaxation methods [Saa20, p. 4.1.1]. In this case, the matrix $\boldsymbol{A}$ is partitioned on submatrices, and the methods above use subblocks, e.g., the Jacobi method uses diagonal blocks in place of the diagonal of the matrix $\boldsymbol{A}$.

1.2.2 Projection methods

The other way to approximately solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{A} \in \mathbb{R}^{n\times n}$ is to start from the initial guess $\boldsymbol{x}_0$, choose two subspaces $\mathcal{K}, \mathcal{L}$ spanned by columns of matrices $\boldsymbol{V}, \boldsymbol{W} \in \mathbb{R}^{n\times m}$, $m \leq n$ and enforce Petrov–Galerkin condition: $\widetilde{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{\delta},\ \boldsymbol{\delta} \in \mathcal{K},\ \boldsymbol{b} - \boldsymbol{A}\widetilde{\boldsymbol{x}} \perp \mathcal{L}$. For suitably chosen subspaces, the new approximation reads

$$\widetilde{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{V}\left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0\right). \tag{1.5}$$

Different choices of $\boldsymbol{V}, \boldsymbol{W}$ lead to different projection methods, amongst which are conjugate gradient algorithm, generalized minimum residual method, and others [Saa03].

Approximation Eq. (1.5) makes sense only if $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}$ is invertible. Two most widely used cases are $\boldsymbol{W} = \boldsymbol{V}$ for $\boldsymbol{A} > 0$ and $\boldsymbol{W} = \boldsymbol{A}\boldsymbol{V}$ for general nonsingular matrix $\boldsymbol{A}$ [Saa03, Propostion 5.1].

When $\boldsymbol{W}$ is chosen as explained above, projection method is completely specified by the choice of $\boldsymbol{V}$. The most powerful and robust projection methods are constructed based on Krylov subspace $\mathcal{K}_m\left(\boldsymbol{A}, \boldsymbol{r}_0\right) = \mathsf{span}\left(\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{r}_0\right)$, where $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$ and $\boldsymbol{x}_0$ is an initial guess. Namely, columns of $\boldsymbol{V}$ are chosen in such a way that $\mathcal{K}_m\left(\boldsymbol{A}, \boldsymbol{r}_0\right) = \mathsf{span}(\boldsymbol{V})$.

So one can show that GMRES [Saa03, Algorithm 6.9] is the projection method with $\mathcal{K}_m\left(\boldsymbol{A}, \boldsymbol{r}_0\right) = \mathsf{span}(\boldsymbol{V})$ and $\boldsymbol{W} = \boldsymbol{A}\boldsymbol{V}$, whereas conjugate gradient [Saa03, Algorithm 6.16] is the projection method with $\mathcal{K}_m\left(\boldsymbol{A}, \boldsymbol{r}_0\right) = \mathsf{span}(\boldsymbol{V})$ and $\boldsymbol{W} = \boldsymbol{V}$. Other projection methods can be constructed based on different choices of $\boldsymbol{W}$ and $\boldsymbol{V}$ (see, for example, [Saa03, Chapter 7]).

1.2.3 Multigrid

The most straightforward view on the geometric multigrid is to describe it as an acceleration scheme for classical iterative methods. For completeness, we briefly recall the main ideas.

The multigrid consists of four essential elements: a projection operator $\mathbf{I}_V^{V'} : V \to V'$ (V, V' are linear spaces) that reduces the number of degrees of freedom, an interpolation operator $\mathbf{I}_{V'}^{V} : V' \to V$ that acts in the "inverse" way, a smoothing

operator $\mathbf{S}_V : V \to V$ which is usually a classical relaxation method, and a set of linear operators $\mathbf{A}_{V'}$ that approximate $\mathbf{A}$ on coarse spaces $V^{'}$. What we describe next is a two-grid cycle.

- For the current approximation $\mathbf{x}^n$ of solutions of $\mathbf{Ax} = \mathbf{b}$, one performs several relaxation steps $\overline{\mathbf{x}} = \mathbf{S}^{\nu}\left(\mathbf{x}^n, \boldsymbol{b}, \boldsymbol{A}\right)$.
- Then, based on properties of $\mathbf{S}$, the linear space $V^{'}$ and the transfer operator $\mathbf{I}_V^{V^{'}} : V \to V^{'}$ are constructed. The purpose of this space is to represent the residual $\mathbf{r} = \mathbf{b} - \mathbf{A}\overline{\mathbf{x}}$ and an error $\mathbf{e} = \mathbf{x}_{\text{exact}} - \overline{\mathbf{x}}$ accurately using fewer degrees of freedom $\left|V^{'}\right| < |V|$.
- Having the space $V^{'}$, one constructs an operator $\mathbf{A}^{'}$ that approximates $\mathbf{A}$ and solves the error equation $\mathbf{A}^{'}\mathbf{e}^{'} = \mathbf{I}_V^{V^{'}}\mathbf{r}$.
- The error, after projection back to V, gives the next approximation to the exact solution, $\mathbf{x}^{n+1} = \mathbf{S}^{\mu}\left(\mathbf{x}^n + \mathbf{I}_{V^{'}}^{V}\mathbf{e}^{'}, \boldsymbol{b}, \boldsymbol{A}\right)$.

The multigrid utilizes a two-grid cycle to solve the error equation $\mathbf{A}^{'}\mathbf{e}^{'} = \mathbf{I}_V^{V^{'}}\mathbf{r}$ itself. It produces the chain of spaces (grids in the geometric setup), projection operators that allow moving between them, and a set of approximate linear operators. For more details, we refer the reader to other resources: a simple introduction to geometric multigrid can be found in [Saa03, ch. 13], for the algebraic multigrid a recent review [XZ17], physical considerations about algebraic multigrid can be found in the introduction of [RSB11], and among other books on the subject, [TOS00] provides a comprehensive introduction for practitioners.

1.3 Model equations

Here we gather partial differential equations that are frequently used as test problems. The section includes the description of discretization as well as background information such as applications of given equation and numerical difficulties associated with it.

1.3.1 Finite difference discretization

We use uniform 1D grid (or mesh) inside the interval $x \in [0, 1]$:

$$M_l^{1\text{D}} = \left\{x_j^l = j/2^l : j = 0, 1, \dots\ , 2^l - 1, 2^l\right\}, \tag{1.6}$$

where grid spacing is $h = 2^{-l}$. In 2D grid is build as a direct product of two 1D meshed:

$$M_l^{2\mathrm{D}} = M_l^{1\mathrm{D}} \otimes M_l^{1\mathrm{D}} = \left\{ \left(x_j^l, y_k^l\right) = \left(j/2^l, k/2^l\right) : j,k = 0,1,\dots,2^l-1,2^l \right\}. \quad (1.7)$$

The finite difference approximation on the uniform grid is defined either with Taylor series [LeV07, Chapter 1] or using polynomial interpolation [CF13, Section 11.2]. To give an example of finite difference approximation, consider function $u(x,y)$ with at least two derivative and 2D grid with fixed l. Second order finite difference approximation to Laplace operator reads:

$$-\left(\partial_x^2 + \partial_y^2\right) u(x,y)\big|_{x=x_i,y=y_j} \simeq \frac{2u_{i,j} - u_{i+1,j} - u_{i,j+1} - u_{i-1,j} - u_{i,j-1}}{h^2}, \quad (1.8)$$

where $u_{i,j} = u(x_i, y_j)$. For more extended discussion of finite difference method consult [Col12], [LeV07], [Ise09a].

1.3.2 Finite element discretization

For finite element discretization we define meshes the same way as for finite difference method (see (1.6), (1.7)). For a given 1D mesh with fixed l, we define a set of basis functions $\phi_i^l(x) = \phi^l(x - x_i), i = 0,\dots,2^l$, which are rescaled and translated copies of a tent function $\phi^l(x) = \left(1 + x/2^l\right) \mathrm{Ind}\left[-1/2^l \le x \le 0\right] + \left(1 - x/2^l\right) \mathrm{Ind}\left[0 < x \le 1/2^l\right]$, where $\mathrm{Ind}\,[x]$ is 1 if x holds and 0 otherwise. Basis functions $\left\{\phi_i^L(x) : i = 0,\dots,2^L\right\}$ are used to perform standard finite element discretization [Cia02] for $x \in [0,1]$.

For example, consider second-order boundary value problem

$$-\frac{d}{dx}\left(g(x)\frac{d}{dx}u(x)\right) = f(x),\ x \in [0,1],\ u(0) = u(1) = 0. \quad (1.9)$$

Given mesh M_l we consider the following approximation for $u(x)$:

$$u(x) \simeq \hat{u}(x) = \sum_{j=1}^{2^l-1} \phi_i^l(x) u_i. \quad (1.10)$$

To find unknown coefficients u_i we enforce PDE in a weak form:

$$\int_{x_{i-1}}^{x_{i+1}} dx g(x) \frac{d}{dx}\phi_i^l(x) \frac{d}{dx}\hat{u}(x) = \int_{x_{i-1}}^{x_{i+1}} dx \phi_i^l(x) f(x). \quad (1.11)$$

Petrov-Galerkin condition Eq. (1.11) allows us to form system of linear equations that can be used to find u_i.

For higher dimensions, we use M_l and ϕ_i^l that are direct products of unidimensional meshes and basis functions. More details on finite element method can be found in [Cia02], [ZTZ13].

1.3.3 Poisson equation

Poisson equation appears in a variety of contexts, from continuum mechanics [PTA12, Sections 4.3, 5.1] to electrodynamics [Jac99, Section 1.7]. It is also a standard test equation for multilevel solvers and preconditioners [TOS00, Section 1.4]. The continuum boundary value problem reads

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} = f(x),\ x, y \in [0,1]^2,\ u(x,y)|_{\partial\Gamma} = 0, \tag{1.12}$$

here Γ represents a domain, and $\partial\Gamma$ is a boundary. We use second-order finite difference discretization in $D = 1$ and $D = 2$ given by stencils

$$s_{P(3)\mathrm{FD}} = \frac{1}{h^2}\begin{bmatrix} -1 & 2 & -1 \end{bmatrix}, \tag{1.13}$$

$$s_{P(5)\mathrm{FD}} = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}, \tag{1.14}$$

Beside that we also consider finite element discretization:

$$s_{P(3)\mathrm{FEM}} = \frac{1}{h}\begin{bmatrix} -1 & 2 & -1 \end{bmatrix}, \tag{1.15}$$

$$s_{P(9)\mathrm{FEM}} = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \tag{1.16}$$

We also employ a high order compact scheme known as Mehrstellen [Col12, Table VI]. Mehrstellen discretization corresponds to the stencil

$$s_{P(M)} = \frac{1}{6h^2}\begin{bmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{bmatrix}, \tag{1.17}$$

which can be used to construct a fourth and sixth-order accurate approximation to the Poisson equation if boundary conditions and right-hand side are sufficiently smooth [Ros75].

Finally, for some solvers we consider four-order discretization with nine-point stencil (see [TOS00, Section 5.4])

$$s_{P(9)} = \frac{1}{12h^2} \begin{bmatrix} & & 1 & & \\ & & -16 & & \\ 1 & -16 & 60 & -16 & 1 \\ & & -16 & & \\ & & 1 & & \end{bmatrix}. \tag{1.18}$$

1.3.4 Mixed derivative

Another problem of interest is a Poisson equation with mixed derivative

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} - 2\tau \frac{\partial^2 u(x,y)}{\partial x \partial y} = f(x),\ x,y \in [0,1]^2,\ u(x,y)|_{\partial\Gamma} = 0. \tag{1.19}$$

For $|\tau| > 1$ the equation becomes hyperbolic, which poses challenges to standard multigrid components (see [TOS00, Section 7.6]). For this problem we use only standard bilinear finite element and second-order finite difference approximations.

1.3.5 Anisotropic problems

Anisotropic version of Poisson equation

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \epsilon \frac{\partial^2 u(x,y)}{\partial y^2} = f(x),\ x,y \in [0,1]^2,\ u(x,y)|_{\partial\Gamma} = 0, \tag{1.20}$$

arises naturally in computational fluid dynamics when a refined or stretched grid is used to resolve a boundary layer, shock, or some other singularity [Lis17, Chapter 4], [TOS00, Section 5.1.2]. Parameter ϵ can also be related to the anisotropy of the physical system. For example, a crystal's permittivity can depend on the direction [New05, Chapter 9], so electrostatic boundary-value problems lead to an anisotropic Poisson equation. We also consider anisotropy along different direction

$$-\epsilon \frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} = f(x),\ x,y \in [0,1]^2,\ u(x,y)|_{\partial\Gamma} = 0, \tag{1.21}$$

and anisotropy that varies in space

$$-g_1(x,y)\frac{\partial^2 u(x,y)}{\partial x^2} - g_2(x,y)\frac{\partial^2 u(x,y)}{\partial y^2} = f(x),\ x,y \in [0,1]^2,\ u(x,y)|_{\partial\Gamma} = 0,$$
$$g_i(x,y) = \begin{cases} (2-i)\sigma + (i-1),\ \text{if } (x-0.5)(y-0.5) \geq 0; \\ (i-1)\sigma + (2-i),\ \text{if } (x-0.5)(y-0.5) < 0. \end{cases} \tag{1.22}$$

Again standard finite element and finite difference discretizations are used.

1.3.6 Helmholtz equation

Helmholtz equation

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} - k^2 u(x,y) = f(x),\ x,y \in [0,1]^2,\ u(x,y)|_{\partial\Gamma} = 0, \tag{1.23}$$

appears in the context of wave propagation problems [Erl08, Section 2.1]. For example, the Helmholtz equation needs to be solved at each time step in the semi-implicit discretization of governing equation of non-hydrostatic weather prediction models [Ste+03, Section 4.1].

Because of the term $-k^2 u(x,y)$, bilinear finite element discretization can result in an indefinite matrix, especially for large k. However, the value of k can not be arbitrary on a given grid because of the pollution problem [BS97]. More precisely, unless $k^2 h$ is sufficiently small, the solution to a discrete problem is of no use because it does not approximate an exact solution. Given that we usually fix $k^2 h = \text{const}$ and chose h to have reasonable number of wavelength per h and avoid indefinite linear problems.

1.3.7 Convection-diffusion problems

When convective transport is present, the original diffusion equation needs to be modified as follows

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} + v_x u(x,y) + v_y u(x,y) = f(x,y),\ x,y \in [0,1]^2,\ u(x,y)|_{\partial\Gamma} = 0. \tag{1.24}$$

The presence of v_x and v_y results in nonsymmetric matrix. Since we employ bilinear finite element or standard finite difference discretization both of which are centered difference approximation, the stability restriction is given by Peclet condition $\max(|v_x|, |v_y|) \leq 2/h$.

1.3.8 Biharmonic equation

The only fourth-order equation we consider is biharmonic:

$$\begin{aligned} &\frac{\partial^4}{\partial x^4}u(x,y) + 2\frac{\partial^2}{\partial x \partial y}u(x,y) + \frac{\partial^4}{\partial y^4}u(x,y) = f(x,y), \\ &x, y \in [0,1]^2\,,\ \ u(x,y)|_{\partial\Gamma} = 0,\ \ \partial_n u(x,y)|_{\partial\Gamma} = 0, \end{aligned} \tag{1.25}$$

here ∂_n is a derivative along the normal direction to the boundary $\partial\Gamma$. Applications of the Biharmonic equation include a description of fluid flows [CLL04], vibrating plates, Chladni figures [GK12], gravitation theory, and quantum mechanics [MLH18, Introduction]. To discretize this equation, we use centered second-order finite difference approximation given by a 13 point stencil

$$s_{\text{BH}} = \begin{bmatrix} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & 20 & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{bmatrix}, \tag{1.26}$$

which should be modified appropriately near the boundaries [TCK92, Section 4] (see also [GM79] and [Bra66]).

1.3.9 Diffusion with discontinuous coefficients

In some situations, diffusion coefficient $a(x,y)$ in equation

$$-\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial u(x,y)}{\partial x}\right) - \frac{\partial}{\partial y}\left(a(x,y)\frac{\partial u(x,y)}{\partial y}\right) = f(x,y),\ x,y \in [0,1]^2\,,\ u(x,y)|_{\partial\Gamma} = 0, \tag{1.27}$$

is discontinuous along some curve or surface inside the computational domain. For example, this is the case in reservoir simulation [TOS00, Section 7.7.1], and the description of the neutron diffusion [Alc+81]. For our experiments, we take

$$a(x,y) = g(x) + g(y),\ g(x) = \sigma^{-1}\mathsf{Ind}\left[x < 1/2\right] + \sigma\mathsf{Ind}\left[x \geq 1/2\right], \tag{1.28}$$

where σ is a parameter that controls the magnitude of the jump. The discretization in use is, again, finite element method.

1.3.10 Implicit scheme for the heat equation

We also consider "algebraic" equation that comes from the trapezoidal discretization (in time) of the heat equation

$$\begin{aligned} &\frac{\partial u(x,y,t)}{\partial t} = \frac{\partial^2 u(x,y,t)}{\partial x^2} + \frac{\partial^2 u(x,y,t)}{\partial y^2},\ x,y \in [0,1]^2,\ t \in [0,+\infty), \\ &u(x,y,t)|_{t=0} = \phi(x,y),\ \ u(x,y)|_{\partial\Gamma} = 0. \end{aligned} \tag{1.29}$$

Let A be a matrix that corresponds to a spatial FEM discretization of the right-hand side operator. It results in a system of ordinary differential equations

$$\frac{du_i(t)}{dt} = \sum_j A_{ij} u_j(t),\ \ u_i(0) = \phi_i. \tag{1.30}$$

Application of the trapezoidal rule leads to an unconditionally stable iteration

$$\sum_j \left(I - \frac{\widetilde{\mu}}{2} A \right)_{ij} u_j^{n+1} = \sum_j \left(I + \frac{\widetilde{\mu}}{2} A \right)_{ij} u_j^n \tag{1.31}$$

known as Crank-Nicolson scheme [Ise09b, Section 16.4]. Here $\widetilde{\mu} = \Delta t$ is related to the Courant number $\mu = \Delta t / \Delta x^2$. Since matrix $\left(I - \frac{\widetilde{\mu}}{2} A \right)$ is symmetric positive definite for $\widetilde{\mu} \geq 0$ that needs to be inverted during each iteration, we test our preconditioner on this problem.

1.4 Outline of the thesis

Here we describe the results of the dissertation in a non-technical way with an emphasis on concepts rather than technical details and their relations to two questions raised in Section 1.1.

1.4.1 Gaussian belief propagation

In this part, we describe a construction of iterative techniques based on statistical inference.

More precisely, this is done by first observing that marginal distributions of certain multivariate normal distributions (with parameters depending on $\boldsymbol{A}$, $\boldsymbol{b}$) can be used to find the solution to Eq. (1.1). After that one can apply statistical algorithms to approximate marginal distributions. So the techniques from statistics induce iterative method Eq. (1.2). These ideas are summarized is [Bic08], [She+08].

The problem is the described technique only works for symmetric positive definite matrices $\boldsymbol{A}$. Our contribution (see [Fan22]) is the generalization of this approach to non-symmetric matrices. Along the way, we also prove the sufficient convergence condition, build block-version of the same algorithm and define a multigrid method based on proposed iterations.

This chapter demonstrates that statistical inference can supply novel iterative methods competitive with classical iterations.

1.4.2 Probabilistic projection methods

This part deals with the second question, i.e., how to decide when to stop iterations. The question is answered based on a probabilistic approach.

Namely, the uncertainty about the true solution on iteration step n is summarized by a probability density over the approximate solution on this step. One of the benefits of expressing uncertainty as the probability is the fact, that uncertainties from other sources (for example, uncertainty about parameters of PDE) can be combined with the ones that originated from the iterative approximation of the exact solution.

The method is based on the relation between projection method Eq. (1.5) and parameters of the conditional distribution of certain multivariate normal model [Hen15], [Coc+19a], [Bar+19].

The main downside of the articles mentioned above is that the normal models proposed there are either unable to reproduce the projection method Eq. (1.5) or lead to uncertainty that is poorly calibrated. Our contribution (see [Fan21b]) is the corrected multivariate normal model with fairly good uncertainty calibration. We also demonstrate the application of the proposed approach to PDE-constrained optimization.

The results of this part indicate that statistical techniques can be used to provide probabilistic estimation of error for iterative method. As such, probabilistic projection methods can in principle be used to derive stopping criteria for iterative implementation of projection methods Eq. (1.5).

1.4.3 Hidden representation

This chapter continues the topic of probabilistic error analysis for iterative methods. Here we propose an original scheme to quantify error for a large class of numerical methods using probability theory.

The main idea is to concoct transformations with two properties. First, they preserve the exact solution (since the solution is the same we can interpret these transformations as the change of the representation for the same problem, hence the name "hidden representation"). Second, they do not preserve the approximate solution. We show that it is fairly simple to construct such transformations. In fact, almost any change of representation possesses the desired properties.

When the transformations are found, one parametrizes them and puts some probability distribution over the parameters. By construction, this scheme guarantees convergence of measure to Dirac delta function when the approximate solution converges to the exact one.

The idea of hidden representation is demonstrated on a large set of scientific problems and developed in detail for iterative methods. This way we construct the instationary probabilistic Richardson method. This method returns probability distribution over the approximate solution which can be used to analyze the error of approximation and derive stopping criteria. We also show how well-calibrated uncertainty of probabilistic Richardson method can be used to speed up the iterative technique.

1.4.4 Black-box optimization of BPX preconditioners

Here we return to the construction of iterative methods. This time we focus on preconditioners, i.e., on numerically attractive approximations to $\boldsymbol{N} = \boldsymbol{A}^{-1}$ with good regularity properties.

To approach the problem, we combine modified version of multilevel preconditioners [BPX90] (known as BPX preconditioners) with stochastic gradient-based optimization of suitably introduced loss function that approximates condition number of matrix $\boldsymbol{NA}$.

The techniques introduced in this section show how machine learning techniques (stochastic optimization, unsupervised training) allow for a black-box construction of optimal preconditioners. Comparison of learned preconditioners with BPX preconditioners shows that machine learning is a valuable tool for the construction of iterative techniques.

The chapter is based on the article [OF21]

1.4.5 Neural multigrid architectures

In this chapter, we continue to develop unsupervised learning techniques for the construction of iterative methods.

This time we introduce the special architecture that can emulate geometric multigrid. As explained in Section 1.2.3, multigrid consists of a few components that introduce a significant leeway in the whole scheme. The introduced architecture along with the stochastic optimization technique allows tuning these components simultaneously to produce a multigrid solver with a faster convergence rate.

In this chapter, we also study the generalization capabilities of the proposed architecture and show that our neural networks generalize much better than the other related architectures.

The material of this chapter is based on the article [Fan21a].

1.4.6 Relaxation methods in the multi-armed bandit setting

Two previous chapters emphasize static picture, i.e., the solver is constructed during the "offline stage" and later applied to the desired problem. In the present section, this restriction is lifted and the solver adapts itself to the problem simultaneously to the construction of an approximate solution.

This is achieved by combining online optimization techniques from reinforcement learning with the classical estimation of the spectral radius of the error propagation matrix.

The capabilities of this approach are demonstrated for relaxation techniques (SOR, Richardson) and geometric multigrid method.

Part I

Statistical inference

Chapter 2

Linear problems and statistical inference

Statistical inference is a set of techniques to determine free parameters of statistical model based on data. As an example consider classical regression model

$$p\left(y|x,\boldsymbol{\beta}\right)=\mathcal{N}\left(\boldsymbol{\beta}^{T}\boldsymbol{\phi}(x),\sigma^{2}\right), \tag{2.1}$$

where variance σ^2 is known, x is explanatory variable and $\boldsymbol{\phi}(x)$ is a vector of "features" (for example, we can take $\boldsymbol{\phi}(x)^T = \begin{pmatrix}1 & x\end{pmatrix}$), and $\boldsymbol{\beta}$ is a conformable vector of parameters subject to inference. For fixed $\boldsymbol{\beta}$, regression model prescribes relation between variables y and x. On the other hand if the relation between y and x is known at $N \geq 1$ points, we can ask for $\boldsymbol{\beta}$ that support available data. One way to uniquely fix parameters $\boldsymbol{\beta}$ is a maximal likelihood principle

$$\boldsymbol{\beta}^{\star}=\arg\max_{\boldsymbol{\beta}}\prod_{i=1}^{N}p\left(y_{i}|x_{i},\boldsymbol{\beta}\right). \tag{2.2}$$

It is well known that maximal likelihood solution $\boldsymbol{\beta}^\star$ will be the same as the one obtained with ordinary least squares [TB97, Lecture 11].

One can consider ordinary list squares as a form of interpolation and justifies the technique without statistics or probability theory. However, probabilistic reinterpretation allows for substantial generalizations. The first natural step is to perform a fully Bayesian analysis. For that one can introduce prior distribution for parameters $\boldsymbol{\beta}$ and find posterior given available data [Gel+13, Chapter 14]. The posterior distribution for $\boldsymbol{\beta}$ contains more information compared to point estimate. That includes,

for example, correlations between different components of $\boldsymbol{\beta}$ and the spread of probability density function for each component. Next, one can select different probabilistic model for likelihood, prior and hyperprior. The choice of probabilistic models can have dramatic consequences. For example, if we take Student's t-distribution to model likelihood, we gain robustness against outliers [Gel+13, Section 17.6]. So the behaviour of the model with Student's likelihood differs substantially from the normal likelihood. Prior distribution is important too. Normal distribution for each component of $\boldsymbol{\beta}$ with shared variance corresponds to L_2 regularization, whereas Cauchy prior provides L_1 regularization [Tib96], [Bis06a, Chapter 3]. The later choice promotes sparsity, so one can select only a few most important component of $\boldsymbol{\beta}$ and nullify all the rest. Again, the Cauchy prior leads to quite different inference compared to the normal prior.

As we have seen on the example of ordinary least squares, probabilistic model can enrich underlying deterministic method with useful features, that are not apparent in the original formulation. The goal of this chapter is a similar reinterpretation of linear problem $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ as statistical inference for suitably chosen probabilistic model. For now suppose that we have a probabilistic model that can be used to perform inference on the solution of linear problem. What can we gain from the reinterpretation?

First, we can apply Bayesian analysis and obtain a distribution over solutions of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. This distribution can be used to perform probabilistic error analysis. We can also apply statistical decision theory, and combine the uncertainty about exact solution with other uncertainties in a meaningful way. More details can be found in contributions on probabilistic numerical methods [HOG15], [OS19], [Coc+19b].

Second, researchers in statistics developed a large set of algorithms to perform exact or approximate inference. To name a few, variational message passing [WBJ05], [Bis06a, Section 10.4.1], belief propagation [YFW03], [Bis06a, Section 8.4.1], expectation propagation [Min13]. So using the duality between statistical inference and the solution of linear problems we can explore the set of novel algorithm unknown to numerical linear algebra community.

Both of these routes are explored in the present part. In Chapter 3 we start by extending belief propagation (and generalized belief propagation) to non-symmetric linear systems and use it as a smoother in multigrid scheme. Next in Chapter 4 we consider probabilistic interpretation for general projection methods. Here we introduce novel prior distribution that results in reasonable posterior and can be used to calibrate uncertainty. At the end, in Chapter 5, we explore a completely new way to introduce probabilistic model for given numerical algorithm. Our approach relies on symmetry transformation and indifference principle. This allows us to

introduce probabilistic model in a more natural way without resorting to "epistemic uncertainty".

Chapter 3

Gaussian belief propagation

3.1 Linear problems and multivariate normal distribution

A basic problem of numerical linear algebra is to solve a linear equation $\mathbf{Ax} = \mathbf{b}$ with an invertible matrix $\mathbf{A}$. The textbook technique is the LU decomposition, equivalent to Gaussian elimination [GV12, ch. 3]. However, when $\mathbf{A}$ is large and sparse, algorithms that exploit sparsity are used instead of direct elimination [DRS16]. Among iterative methods for sparse systems, one can mention classical relaxation techniques such as Gauss-Seidel (GS), Jacobi, Richardson, and projection methods such as conjugate gradient, GMRES, biconjugate gradient, and others [SV01; Saa20].

An easy way to understand the projection methods is to reformulate the original equation as an optimization problem [She+94]. For example, for a symmetric positive-definite matrix $\mathbf{A}$, one has

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} \left(\frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{2} - \mathbf{x}^T \mathbf{b} \right). \tag{3.1}$$

Such a reformulation allows one to apply new techniques and leads to methods of steepest descent, conjugate directions, and cheap and efficient conjugate gradient [HS+52].

Another reformulation of the problem is known, but is less explored. It also goes back to Gauss and his version of elimination. To derive an LU solution of $\mathbf{Ax} = \mathbf{b}$, one can consider the probability density function $p(\mathbf{x})$ of multivariate normal distribution

$$p(\mathbf{x}) \sim \exp \left(-\frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{2} + \mathbf{b}^T \mathbf{x} \right). \tag{3.2}$$

We can consider the first component x_1 of $\mathbf{x}$ and integrate it out in Eq. (3.2) (a process called "marginalization"). The resulting marginal distribution for the remaining components $x_2, x_3, \ldots$ is again multivariate normal, but with the covariance matrix given by the Schur complement of A_{11}[1] and the mean vector modified accordingly, i.e.

$$\mathbf{A}_{22} \leftarrow \mathbf{A}_{22} - \frac{\mathbf{A}_{21}\mathbf{A}_{12}}{A_{11}}, \ \mathbf{b}_2 \leftarrow \mathbf{b}_2 - \frac{\mathbf{A}_{21}b_1}{A_{11}}. \tag{3.3}$$

It is well known that the LU decomposition consists of the very same steps [Ste98]. When x_1 is not a scalar, but a subset of variables, marginalization of multivariate normal distribution results in a block LU decomposition.

Thus, the most popular direct technique for the solution of linear equations with dense matrices is intimately connected with the marginalization problem, which belongs to the class of inference problems. Recently, many other intriguing connections between statistical inference and linear algebra have been pointed out. For instance, in [Coc+19a], [Hen15], and [Bar+18], the authors provide a method to recover the Petrov-Galerkin condition from the Bayesian update and construct a Bayesian version of the conjugate gradients. Paper [Owh17] constructs a state-of-the-art multigrid solver using game theory and statistical inference. These works demonstrate that ideas from statistical inference allow for new and useful insights into problems of linear algebra. It is thus reasonable to explore how other inference algorithms are translated to the realm of numerical linear algebra. Among them are expectation propagation [Min01], Markov chain Monte Carlo, mean field, other variational Bayesian approximations [Bis06b, chapters 8, 10], [WJ+08], and belief propagation with its generalized counterparts. The latter two are the focus of this chapter.

We start to study the reformulation of linear problem as inference problem (3.2) in Section 3.2 with a discussion of general belief propagation algorithm. Next, in Section 3.3 we continue with a specialized version of the algorithm tailored to multivariate normal models, its connection with elimination, Monte Carlo linear algebra and the extension to non-symmetric matrices. Later, in Section 3.4 we extend the discussion on block version. In Section 3.5 we analyse Gaussian belief propagation in the context of multigrid. Section 3.6 contains comparison of belief propagation with other well-established projection and relaxation techniques.

[1] In the article we use boldface for matrices or matrix blocks, and regular font for scalar values and matrix components. In this case A_{11} is an element of the matrix $\mathbf{A}$ in the first row and the first column, and $\mathbf{A}_{22}$ is a square matrix that contains all elements of $\mathbf{A}$ excluding the first row and the first column.

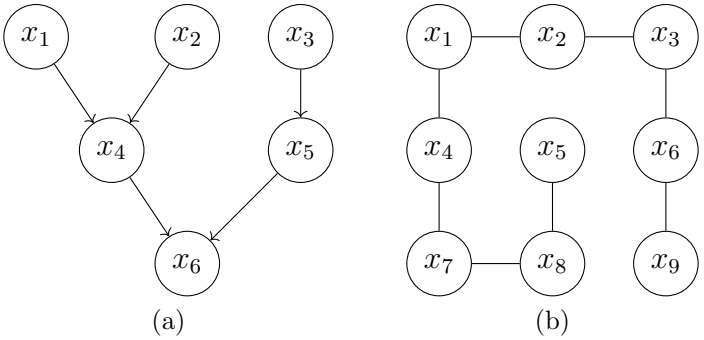


Figure 3.1: (a) – Bayesian network, (b) – Markov random field. Bayesian network specifies conditional dependence between variables and Markov random field restricts the functional form of the distribution. Details can be found in Section 3.2.

3.2 Belief propagation

Most probabilistic models can be completely specified by a joint distribution of random variables (see examples in [Gel+14, Chapters 5, 15, 16]). It is often the case that the joint distribution is factorized on groups of independent or conditionally independent random variables. A convenient way to specify this kind of independence using graphs is called graphical model. Two most widely used instances of graphical models are Bayesian networks (directed graphs) and Markov random field (undirected graphs).

An example of Bayesian network is in Fig. 3.1a. The directed graph, specifies the following functional form of joint distribution

$$p(x_1, \ldots, x_6) = p(x_6|x_4, x_5)p(x_5|x_3)p(x_4|x_1, x_2)p(x_1)p(x_2)p(x_3). \tag{3.4}$$

So we can see that Bayesian network specifies conditional dependence among random variable. It is known that arbitrary Bayesian network can also be represented as a Markov random field [KF09, Chapter 4]. So in what is following we consider only the later. More details about Bayesian networks can be found, for example, in [Bis06b, Chapter 8], [Pea88, Chapter 3], [KF09, Chapter 3].

An example of Markov random field is depicted in Fig. 3.1b. The undirected graph itself only restricts the functional form of the joint distribution. To completely specify the joint distribution one needs to supply the set of non-negative integrable function that define functional dependence between variables $\psi_{ij}(x_i, x_j)$ and dependence on individual variables $\phi_i(x_i)$. Given these functions we can write joint probability

density function in the following form:

$$\begin{aligned} p(x_1,\dots,x_8) =& \psi_{14}(x_1,x_4)\psi_{12}(x_1,x_2)\psi_{23}(x_2,x_3)\psi_{36}(x_3,x_6)\psi_{69}(x_6,x_9) \\ & \psi_{58}(x_5,x_8)\psi_{78}(x_7,x_8)\psi_{47}(x_4,x_7)\prod_{i=1}^{9}\phi_i(x_i). \end{aligned} \tag{3.5}$$

For a general pairwise Markov random field the situation is analogous. First, we need to define a graph Γ. The graph Γ is the set of edges $\mathcal{E}$ and vertices $\mathcal{V}$. Each vertex i corresponds to the random variable x_i (discrete or continuous), and each edge corresponds to interactions between variables. The set of non-negative integrable functions $\{\phi_i, \psi_{ij}\}$ together with the graph Γ completely specifies the form of the probability density function of a pairwise Markov random field

$$p(\boldsymbol{x}) = \frac{1}{Z}\prod_{i\in\mathcal{V}}\phi_i(x_i)\prod_{(ij)\in\mathcal{E}}\psi_{ij}(x_i,x_j) \equiv \frac{1}{Z}\exp\left(-E(\boldsymbol{x})\right). \tag{3.6}$$

Here, Z is a normalization constant known in statistical physics as a partition function [Set21, Chapter 6]. The second equation in (3.6), that is, the Boltzmann (or Gibbs) distribution [Set21, Chapter 6], should be considered as a definition of energy $E(\mathbf{x})$. Since all functions ψ and ϕ are non-negative, it is always possible to introduce $E(\mathbf{x})$.[2]

Markov random fields are used to model a wide class of system. In physics all lattice models including Ising, Potts, six-vertex, Heisenberg chains can be presented in form (3.6) where x_i are often discrete variables [Set21, Chapter 8]. Markov random field can be used to perform image de-noising [Bis06b, Chapter 8.3.3], extracting messages from error-correction codes [YFW03, Section 1.4]. Object matching, edge detection and other tasks in computer vision can also be performed using Markov random field [Li09].

When probabilistic model is specified, a common task in the inference process is a computation of a partial distribution (or a marginalization) $p_r(\mathbf{x}_r) = \sum_{\mathbf{x}\setminus\mathbf{x}_r} p(\mathbf{x})$ possibly given values of some other variables $p_r(\mathbf{x}_r|\mathbf{x}_o = \mathbf{y}) = \sum_{\mathbf{x}\setminus\mathbf{x}_r} p(\mathbf{x}|\mathbf{x}_o = \mathbf{y})$ (integrals replace sums if $\mathbf{x} \in \mathbb{R}^N$). The marginalization is, in principle, straightforward operation. However, as exemplified in [Bis06b, Chapter 8.4.1], a naive summation for a chain with N sites (Fig. 3.1b shows the chain with $N = 9$ sites) where each random variable has K states requires $O(K^N)$ operations. This cost can be dropped to

[2]Note, that the parallel to statistical physics is rather deep and heavily exploited in graphical models. The main tools are Bethe approximation to free energy, Kikuchi approximation, cluster variation method (see [Pel05], [YFW01b], [YFW05]).

$O(NK^2)$ if one exploits the structure of the underlying graph. Typically, one wants to obtain marginal distribution for a set of variables (see [YFW03, Section 1.1] for a medical example). Good news is all marginal distributions $p_i(x_i)$ can be found simultaneously with the same cost $O(NK^2)$, using belief propagation, introduced by Pearl [Pea88].

Pearl's algorithm operates with local messages that spread from node to node along the graph edges, and beliefs (approximate or exact marginals) are computed as a normalized product of all incoming messages after the convergence. More precisely, belief propagation consists of (i) the message update rule

$$m_{ij}(x_j) \leftarrow \sum_{x_i} \phi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in N(i)\setminus j} m_{ki}(x_i), \tag{3.7}$$

where m_{ij} is a message from node i to node j and $N(i)$ is the set of neighbors[3] of the node i, and (ii) the formula for marginals

$$b_i(x_i) \sim \phi_i(x_i) \prod_{k \in N(i)} m_{ki}(x_i). \tag{3.8}$$

Despite the fact that (3.7) and (3.8) can be justified only when graph Γ has no loops (see [Pea88]) [4], belief propagation was successfully applied on networks with loops [FM98]. The resulting algorithm is sometimes called loopy belief propagation [MWJ13].

With that we close the short reminder on graphical models and belief propagation. More specialised version of graphical models and the belief propagation algorithms will be used in the later sections to solve large sparse linear systems.

3.3 Gaussian belief propagation

3.3.1 Message update rules for Gauss-Markov models

Gauss-Markov model is a Markov random field with multivariate normal joint distribution (3.6). More formally, given any $\boldsymbol{A} > 0$ and conformable vector $\boldsymbol{b} \in \mathbf{R}^N$

[3] Here we work with a pairwise Markov random field, so there is a underlying undirected graph $\Gamma = \{\mathcal{V}, \mathcal{E}\}$. For such graph, the number of neighbors for node i is defined as $N(i) = \{j : \ j \in \mathcal{V}, \exists (i,j) = (j,i) \in \mathcal{E}\}$.

[4] We are going to show how rules (3.7) can be derived for Gauss-Markov model in the next section.

we construct undirected graph Γ with the set of vertices $\mathcal{V}$ corresponding to components of $b_i, i = 1, \ldots, N$, and the set of edges $\mathcal{E}$ corresponding to nonzero elements $A_{ij}, i \neq j$. More specifically, edge e_{ij} from vertex i to vertex $j \neq i$ appears in $\mathcal{E}$ iff $A_{ij} \neq 0$. Having this graph we can write a joint probability density function:

$$\begin{aligned} p(\boldsymbol{x}) &= \frac{1}{Z} \prod_{i \in \mathcal{V}} \exp\left(-\frac{1}{2} A_{ii} x_i^2 + b_i x_i\right) \prod_{(ij) \in \mathcal{E}} \exp\left(-A_{ij} x_i x_j\right) \\ &= \frac{1}{Z} \exp\left(-\frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{2} + \mathbf{b}^T \mathbf{x}\right) = \mathcal{N}\left(\mathbf{x} | \underbrace{\mathbf{A}^{-1}\mathbf{b}}_{\text{mean}}, \underbrace{\mathbf{A}^{-1}}_{\substack{\text{covariance} \\ \text{matrix}}}\right). \end{aligned} \tag{3.9}$$

For the Gauss-Markov model, marginal distributions are known explicitly. For individual components of the vector $\mathbf{x}$, which is distributed according to (3.9), one can obtain distributions in closed form

$$p_i(x_i) = \mathcal{N}\left(x_i | \left(\mathbf{A}^{-1}\mathbf{b}\right)_i, \left(\mathbf{A}^{-1}\right)_{ii}\right) \equiv \mathcal{N}\left(x_i | \mu_i, \beta_i\right). \tag{3.10}$$

From (3.10) we can see that mean values stacked in vector $\boldsymbol{\mu}$ provide solution for linear problem $\boldsymbol{A}\boldsymbol{\mu} = \boldsymbol{b}$. Since belief propagation can be used to obtain marginal distributions it was proposed to use Perl's algorithm to solve linear problems [Bic08], [She+08].

Although, for continuous random variables the problem of marginalization and the algorithm of belief propagation are harder in general, it is not the case for the normal distribution. Namely, for the Gauss-Markov model, one can parameterize messages in the form of the normal distribution

$$m_{ji}(x_i) \sim \exp\left(-\frac{\Lambda_{ji}\left(x_i - \mu_{ji}\right)^2}{2}\right), \tag{3.11}$$

and derive message update rule directly from (3.7) using message parametrization and definition of ψ and ϕ given in (3.9). However, to gain additional intuition we use only (3.8) and consistency condition. Consider two variables x_i and x_j, such that $A_{ij} \neq 0$. According to (3.8) the joint probability density function is multivariate normal

$$\begin{aligned} &b\left(x_i, x_j\right) = \mathcal{N}\left(\left.\begin{pmatrix} x_i \\ x_j \end{pmatrix}\right| \boldsymbol{m}, \boldsymbol{\Sigma}\right), \ \boldsymbol{\Sigma}^{-1}\boldsymbol{m} = \begin{pmatrix} b_i + \sum_{k \in N(i) \setminus j} \Lambda_{ki}\mu_{ki} \\ b_j + \sum_{k \in N(j) \setminus i} \Lambda_{kj}\mu_{kj} \end{pmatrix}, \\ &\boldsymbol{\Sigma}^{-1} = \begin{pmatrix} A_{ii} + \sum_{k \in N(i) \setminus j} \Lambda_{ki} & A_{ij} \\ A_{ji} & A_{jj} + \sum_{k \in N(j) \setminus i} \Lambda_{kj} \end{pmatrix}. \end{aligned} \tag{3.12}$$

Joint probability density (3.12) also prescribes probability density functions for individual variables:

$$b(x_i) = \mathcal{N}\left(x_i | m_{11}, \Sigma_{11}\right), \; b(x_j) = \mathcal{N}\left(x_j | m_{22}, \Sigma_{22}\right). \tag{3.13}$$

On the other hand, we can compute probability density function for x_j applying (3.8) directly, which gives

$$b(x_j) = \mathcal{N}\left(x_j | m, \Sigma\right), \; \frac{m}{\Sigma} = b_j + \sum_{k \in N(i)} \Lambda_{kj}\mu_{kj}, \; \Sigma = A_{jj} + \sum_{k \in N(j)} \Lambda_{kj}. \tag{3.14}$$

Since $b(x_j)$ is the same in (3.13) and (3.14) we conclude that $m_{22} = m$ and $\Sigma = \Sigma_{22}$. We see that (3.13) does not contain Λ_{ij}, μ_{ij} as well as Λ_{ji}, μ_{ji}, so from (3.14) we can find, for express, Λ_{ji} and μ_{ji} as a function of other messages and interpret this relation as an update rule. Namely, we supplement messages with a superscript $\Lambda_{ji}^{(k)}$, $\mu_{ji}^{(k)}$ that correspond to iteration k and from consistency condition $m_{22} = m$, $\Sigma = \Sigma_{22}$ find the following update rules:

$$\begin{aligned}
\mu_{ji}^{(n+1)} &= \frac{b_j + \sum\limits_{k \in N(j) \setminus i} \Lambda_{kj}^{(n)} \mu_{kj}^{(n)}}{A_{ji}}, \; \Lambda_{ji}^{(n+1)} = -\frac{A_{ij}A_{ji}}{A_{jj} + \sum\limits_{k \in N(j) \setminus i} \Lambda_{kj}^{(n)}}; \\
\mu_{i}^{(n)} &= \frac{b_i + \sum\limits_{j \in N(i)} \Lambda_{ji}^{(n)} \mu_{ji}^{(n)}}{A_{ii} + \sum\limits_{j \in N(i)} \Lambda_{ji}^{(n)}}, \; \beta_i^{(n)} = A_{ii} + \sum_{j \in N(i)} \Lambda_{ji}^{(n)}.
\end{aligned} \tag{3.15}$$

From the derivation of the update rules it is clear that the only role of messages is to enforce consistency condition, that is, to make sure that $b(x_i, x_j) = \int dx_i b(x_i, x_j)$.

These update rules correspond to the fully parallel schedule such that at the current iteration step, each node sends messages to all its neighbors based on messages received at the previous step. Equations for the mean and precision should be put to use only after saturation according to some criteria, for example $|\mu^{(n+1)} - \mu^{(n)}| \leq$ tolerance, and the same for $\mathbf{\Lambda}$. Rules (3.15) are collectively known as Gaussian belief propagation (GaBP).

As we already mentionned, belief propagation was designed to give an exact answer if Γ has no loops. In the presence of loops, the result appears to be approximate if delivered at all. In the case of GaBP, the situation is more optimistic. We briefly recall some useful facts about GaBP that we discuss later in more detail. If GaBP converges on the graph of arbitrary topology, the means are exact, but variances

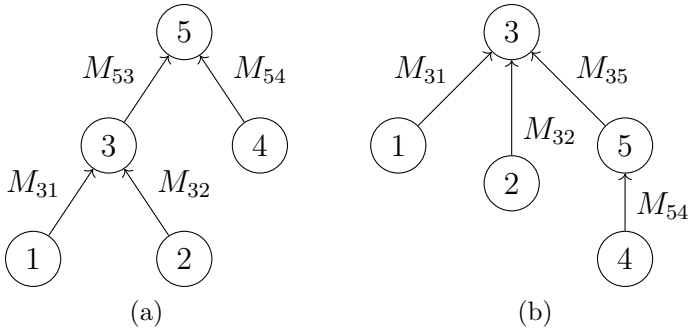


Figure 3.2: Both (a) and (b) sketch the graph, corresponding to the matrix $\mathbf{A}$ from equation (3.16). We use M_{ji} to represent the pair of messages (Λ_{ji}, μ_{ji}) (see equation (3.19)) from the node i to j. Figures (a) and (b) show different order of elimination. For example, in case of (a) one first exclude x_1 and x_2 from the equation for x_3 and then solve resulting equation to obtain x_5.

can be incorrect [WF00]. The best sufficient condition for convergence of the Gauss-Markov model with symmetric positive-definite matrix can be found in [MJW06], we discuss it later in greater detail. The fixed point of GaBP is unique [Hes04]. On the tree, GaBP is equivalent to the Gaussian elimination [PK04].

In the derivation of GaBP rules we relied on the definition of the Markov random field (3.9) which is valid only for $\boldsymbol{A} > 0$. This fact makes extension of GaBP on non-symmetric case conceptually nontrivial. The way to go is to relate GaBP rules with Gaussian elimination that can be applied to arbitrary matrix. This is done in the next section.

3.3.2 Elimination perspective

The connection of GaBP with Gaussian elimination was explored in [PK04]. Ideas of this section are similar in spirit, but the presentation is more straightforward and after appropriate modifications applies to non-symmetric matrices as well.

To illustrate the main ideas, consider a linear problem $\boldsymbol{Ax} = \boldsymbol{b}$ with the matrix and right-hand side defined as

$$\mathbf{A} = \begin{pmatrix} A_{11} & 0 & A_{13} & 0 & 0 \\ 0 & A_{22} & A_{23} & 0 & 0 \\ A_{31} & A_{32} & A_{33} & 0 & A_{35} \\ 0 & 0 & 0 & A_{44} & A_{45} \\ 0 & 0 & A_{53} & A_{54} & A_{55} \end{pmatrix}, \ \mathbf{b} \in \mathbb{R}^5. \tag{3.16}$$

For simplicity, require that $\boldsymbol{A}$ be positive definite and that all elements of $\boldsymbol{A}$, not explicitly indicated as zeros, are nonzero.

To obtain GaBP rules, we introduce a graph of the matrix (3.16) the same way as this has been done in Section 3.3.1.

Suppose we want to calculate variable x_5. To do it, we exclude variables x_1, x_2 from the equation for x_3 and then eliminate variables x_3, x_4 from the equation for x_5

$$\underbrace{\left(A_{33}-\frac{A_{31}A_{13}}{A_{11}}-\frac{A_{32}A_{23}}{A_{22}}\right)}_{=\widetilde{A}_{33}}x_3+A_{35}x_5=\underbrace{b_3-\frac{A_{31}b_1}{A_{11}}-\frac{A_{32}b_2}{A_{22}}}_{=\widetilde{b}_3}; \tag{3.17a}$$

$$\left(A_{55}-\frac{A_{53}A_{35}}{\widetilde{A}_{33}}-\frac{A_{54}A_{45}}{A_{44}}\right)x_5=b_5-\frac{A_{53}\widetilde{b}_3}{\widetilde{A}_{33}}-\frac{A_{54}b_4}{A_{44}}. \tag{3.17b}$$

Fig. 3.2a captures this particular elimination order. In the same vein, to find x_3 one may follow the order presented in Fig. 3.2b. The resulting equations are

$$\underbrace{\left(A_{55}-\frac{A_{54}A_{45}}{A_{44}}\right)}_{=\widetilde{A}_{55}}x_5+A_{53}x_3=\underbrace{b_5-\frac{A_{53}b_4}{A_{44}}}_{=\widetilde{b}_5}; \tag{3.18a}$$

$$\left(A_{33}-\frac{A_{35}A_{53}}{\widetilde{A}_{55}}-\frac{A_{31}A_{13}}{A_{11}}-\frac{A_{32}A_{23}}{A_{22}}\right)x_3=b_3-\frac{A_{31}b_1}{A_{11}}-\frac{A_{32}b_2}{A_{22}}-\frac{A_{35}\widetilde{b}_5}{\widetilde{A}_{55}}. \tag{3.18b}$$

From these calculations, one can make two observations:

1. In the course of elimination one successively changes the diagonal elements A_{jj} and the right-hand side b_j.

2. The exclusion schemes in figures 3.2a and 3.2b share the same computations. For example, terms $A_{31}A_{13}/A_{11}$, $A_{32}A_{23}/A_{22}$ and $A_{54}A_{45}/A_{44}$ appear on the way to equation (3.18b) as well as to (3.17b). It would be more advantageous to reuse the same computations, not to redo them each time one needs to eliminate a variable.

The first observation suggests that one can introduce corrections to the diagonal terms and b_j, that come from the elimination of variable i. For the sake of convenience, we denote them Λ_{ij} and $\mu_{ij}\Lambda_{ij}$, respectively. For example, equation (3.17a) becomes

$$\left(A_{33}+\Lambda_{13}+\Lambda_{23}\right)x_3+A_{35}x_5=b_3+\Lambda_{23}\mu_{23}+\Lambda_{13}\mu_{13}. \tag{3.19}$$

Since corrections are the same for any order of elimination, to reuse them, we can regard Λ_{ij} and μ_{ij} as a message that node i sends to node j along the edge of the graph. Once computed, these messages are in use in expressions like (3.19) and (3.20). To complete rewriting the elimination in terms of messages, one needs to introduce the rules to update messages when a new variable is excluded. To derive the rules, we rewrite equation (3.17b) using the definition of messages

$$(A_{55} + \Lambda_{35} + \Lambda_{45})\, x_5 = b_5 + \Lambda_{35}\mu_{35} + \Lambda_{45}\mu_{45}. \tag{3.20}$$

Now, the elimination step that leads from (3.17a) to (3.17b) can be written in the form of a message update rule using the definition of Λ_{35}, $\Lambda_{35}\mu_{35}$ and (3.19):

$$\Lambda_{53} = -\frac{A_{35}A_{53}}{A_{33} + \Lambda_{31} + \Lambda_{32}},\ \mu_{53} = \frac{b_3 + \Lambda_{31}\mu_{31} + \Lambda_{32}\mu_{32}}{A_{35}}. \tag{3.21}$$

It is easy to see that one needs to accumulate all messages from neighbors of i except for j to send the message from node i to node j. The solution can be computed from all incoming messages as follows

$$x_j = \frac{b_j + \sum\limits_{k \in \text{neighbours of } j} \Lambda_{kj}\mu_{kj}}{A_{jj} + \sum\limits_{k \in \text{neighbours of } j} \Lambda_{kj}}. \tag{3.22}$$

Note that (3.18) and (3.17) have exactly this form. Equations for the update of messages that we deduced in this section coincide with the GaBP update rules given by (3.15), which are derived from the probabilistic perspective below.

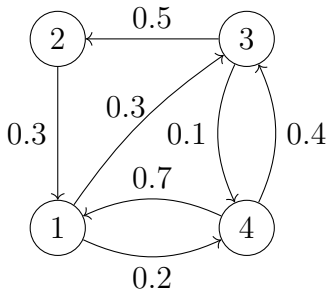
To summarize, the GaBP rules can be understood as a scheme that propagates messages on the graph, corresponding to the matrix of the linear system under consideration. These messages, namely Λ_{ji} and μ_{ji}, represent the corrections to the diagonal terms of matrix $\mathbf{A}$ and to the right-hand side $\mathbf{b}$, resulting from the elimination of variable x_i from the j-th equation, $A_{jj}x_j + A_{ji}x_i + \cdots = b_j$.

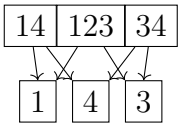
Unlike the discussion in Section 3.3.1, all reasoning above can be applied to non-symmetric matrices as well. This is done in the next section.

3.3.3 Gaussian belief propagation for non-symmetric linear systems

As was explained in the previous section, the GaBP rules can be understood as corrections to the right-hand side and the diagonal elements of the matrix under successive elimination of variables. It means that in principle, one can apply the

$$A = \begin{pmatrix} 6 & 0.3 & 0 & 0.7 \\ 0 & 5 & 0.5 & 0 \\ 0.3 & 0 & 7 & 0.4 \\ 0.2 & 0 & 0.1 & 6 \end{pmatrix}$$





(a) (b) (c)

Figure 3.3: The figure exemplifies the correspondence between nonsymmetric matrix A given in (a), a directed graph in (b); (c) contains a region graph corresponding to the partition $\mathcal{P} = \{\{1,2,3\},\{1,4\},\{3,4\}\}$ (see Section 3.4). Note that the direction of edge e_{ij} corresponds to the entry A_{ji}. In Algorithm 1 we use graph to represent a sparsity pattern of A and do not use directed weighted graphs. Weights on this figure are given for the sake of illustration.

rules to solve at least some nonsymmetric systems. However, there are two problems specific to nonsymmetric case. First, it is possible to have $A_{ij} = 0$ and $A_{ji} \neq 0$. In this case, rules (3.15) lead to singularity as A_{ij} appears in the denominator. Since parametrization of messages is not unique both from the elimination and probabilistic perspectives, it is possible to define new set of messages $\widetilde{\mathbf{\Lambda}}$ and $\mathbf{m}$ as follows $m_{ji}^{(n)} \equiv \mu_{ji}^{(n)} \Lambda_{ji}^{(n)}$, $\widetilde{\Lambda}_{ji}^{(n)} \equiv \Lambda_{ji}^{(n)} / A_{ji}$. Note that this reparametrization has a problem in that it is not one-to-one if $A_{ji} = 0$. However, quick look at the equations (3.17), (3.18) makes clear that indeed it is possible to define messages in that way. That is, if $A_{jk} = 0$, one does not need to eliminate x_k from the second equation so the message Λ_{kj} is indeed zero.

Second, the undirected graph used to derive rules (3.15) is unsuitable for nonsymmetric matrices. To establish rules for given nonsymmetric matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, we construct a directed graph with N vertices $v \in \mathcal{V}$ corresponding to the variables $x_1, \ldots, x_N$ and the set of directed edges $\mathcal{E}$. The edge pointing from the vertex j to the vertex i belongs to the set of edges iff $A_{ij} \neq 0$, i.e. $A_{ij} \neq 0 \Leftrightarrow e_{ji} \in \mathcal{E}$. The example is given in Fig. 3.3. We also define a set of in-neighbors and out-neighbors as $N_{\text{in}}(i) = \{k : k \in \mathcal{V}, e_{ki} \in \mathcal{E}\}$ and $N_{\text{out}}(i) = \{k : k \in \mathcal{V}, e_{ik} \in \mathcal{E}\}$. For example, $N_{\text{in}}(1) = \{2,4\}$ and $N_{\text{out}}(1) = \{3,4\}$ for the matrix in Fig. 3.3.

The direction of edges is chosen to coincide with a flow of messages. Indeed, from Section 3.3.2 we known that for variable x_i the elimination of variable x_j can

Algorithm 1 GaBP for a nonsymmetric linear system.

Input: invertible matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$.
Output: approximate solution $\boldsymbol{x}$.

Form directed graph $G = \{\mathcal{V}, \mathcal{E}\}$ based on $\mathbf{A}$, initialize $\widetilde{\boldsymbol{\Lambda}} = 0$, $\boldsymbol{m} = 0$.
while not converge **do**
  for $j \in \mathcal{V}$ **do**
    $m = b_j$, $\Lambda = A_{jj}$
    for $k \in N_{\text{in}}(j)$ **do**
      $m = m + m_{kj}$, $\Lambda = \Lambda + \widetilde{\Lambda}_{kj} A_{kj}$
    end for
    $x_j = m/\Lambda$
    for $k \in N_{\text{out}}(j)$ **do**
      $\widetilde{\Lambda}_{jk} = -A_{kj} / \left(\Lambda - \widetilde{\Lambda}_{kj} A_{kj}\right)$, $m_{jk} = \widetilde{\Lambda}_{jk} \left(m - m_{kj}\right)$
    end for
  end for
end while

be regarded as a message sent from j to i that contains A_{ij}, which, according to our definitions, is related to the directed edge e_{ji}. This observations allows us to put forward a modified form of update rule:

$$m_{ji}^{(n+1)} = \widetilde{\Lambda}_{ji}^{(n+1)} \left(b_j + \sum_{k \in N_{\text{in}}(j) \setminus i} m_{kj}^{(n)} \right), \; \widetilde{\Lambda}_{ji}^{(n+1)} = - \frac{A_{ij}}{A_{jj} + \sum\limits_{k \in N_{\text{in}}(j) \setminus i} \widetilde{\Lambda}_{kj}^{(n)} A_{kj}}, \quad (3.23)$$

and the expression for the approximate solution

$$\mu_i^{(n)} = \frac{b_i + \sum\limits_{j \in N_{\text{in}}(i)} m_{ji}^{(n)}}{A_{ii} + \sum\limits_{j \in N_{\text{in}}(i)} \widetilde{\Lambda}_{ji}^{(n)} A_{ji}}, \; \beta_i^{(n)} = A_{ii} + \sum_{j \in N_{\text{in}}(i)} \widetilde{\Lambda}_{ji}^{(n)} A_{ji}. \quad (3.24)$$

Note, that $m_{ji}^{(n+1)}$ in (3.23) corresponds to the edge (j, i) of directed graph Γ. The form of update that put more emphasis on the structure of the underlying directed graph appears in Algorithm 1.

Algorithm 1 is sequential, but can run in parallel after some modifications. Parallel version can be discussed in two context.

The first parallel version is related to the order of message update. In Algorithm 1 index j in the first for-loop (line 5) also defines the iteration number (n) in rules (3.15). That is, vertex $k = 2$ receives messages from $j = 1$ if there is an edge $(1, 2)$. Saying another way, messages are used immediately after they are updated. Instead of the "in-place" update of messages we can write the new messages $\widetilde{\mathbf{\Lambda}}^{(n+1)}$, $\boldsymbol{m}^{(n+1)}$ in different containers. The resulting algorithm is related to Algorithm 1 as Jacobi iteration is related to Gauss-Seidel iteration.

The other option is to consider parallel version of Algorithm 1 with no modification. To do that, one needs to use specific techniques for parallel computations on graphs. Relevant details can be found in [Gon+12]. We further discuss the parallel properties of solvers in Section 3.6.1.

The stopping criteria, not specified in Algorithm 1, can be chosen in many different ways. For example, it is possible to use some norm of residual, $e_1 \equiv \left\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(n)}\right\|$, or $e_2 \equiv \max(e_3, e_4)$,, where $e_3 \equiv \left\|\boldsymbol{m}^{(n+1)} - \boldsymbol{m}^{(n)}\right\|$, $e_4 \equiv \left\|\widetilde{\mathbf{\Lambda}}^{(n+1)} - \widetilde{\mathbf{\Lambda}}^{(n)}\right\|$ and stop when $e_i \leq$ tolerance, $i = 1, \ldots, 4$.

Note that the update of $\widetilde{\mathbf{\Lambda}}$ decouples from the one for $\boldsymbol{m}$. So it is possible to construct an algorithm that computes only messages $\widetilde{\mathbf{\Lambda}}$ and returns diagonal elements for the inverse matrix. Later, these messages can be used in the course of all successive iterations if one resorts to the error correction scheme. We discuss how the algorithm of this kind can be utilized to decrease the number of floating point operations in the context of a multigrid scheme.

Two classical theorems from GaBP theory, summarized below, can be readily established for nonsymmetric matrices.

Theorem 3.3.1. *If there is $N \in \mathbb{N}$ such that $m_{ij}^{(N+k)} = m_{ij}^{(N)}$, $\widetilde{\Lambda}_{ij}^{(N+k)} = \widetilde{\Lambda}_{ij}^{(N)}$ for all $e_{ij} \in \mathcal{E}$ and for any $k \in \mathbb{N}$, then $\mu_i^{(N+k)} = \mu_i^{(N)} = (\mathbf{A}^{-1}\mathbf{b})_i$.*

The analogous result for symmetric matrices first appeared in [WF00]. In Section 10.1, we show how to extend the proof for the nonsymmetric case.

Theorem 3.3.2. *If $A_{ii} \neq 0$ $\forall i$, $\left|\widetilde{R}\right|_{ij} := (1 - \delta_{ij}) \frac{|A_{ij}|}{|A_{ii}|}$[5], and $\rho(|\widetilde{\mathbf{R}}|) < 1$, then the Algorithm 1 converges to the solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ for arbitrary $\mathbf{b}$.*

Sufficient condition for symmetric positive-definite matrices was established in [MJW06]. Section 10.2 contains the proof with necessary modifications that holds for nonsymmetric matrices. It should be noted that we do not provide a necessary and sufficient condition for GaBP convergence or the convergence rate. The reason

[5]Here $\left|\widetilde{R}\right|_{ij}$ are matrix elements of the matrix $|\widetilde{\mathbf{R}}|$ and δ_{ij} are elements of diagonal matrix.

is that unlike classical linear iterative methods like Gauss-Seidel and Jacobi GaBP can not be presented in a form $\boldsymbol{x}^{(n+1)} = \boldsymbol{M}\boldsymbol{x}^{(n)} + \boldsymbol{N}\boldsymbol{b}$. We will see in Section 3.5 that GaBP modifies matrices $\boldsymbol{M}$ and $\boldsymbol{N}$. Namely, GaBP can be presented as $\boldsymbol{x}^{(n)} = \boldsymbol{M}^{(n)}\boldsymbol{x}^{(0)} + \boldsymbol{N}^{(n)}\boldsymbol{b}$, where $\boldsymbol{M}^{(n)}$ and $\boldsymbol{N}^{(n)}$ depend on entries of $\boldsymbol{A}$, iteration n and do not depend on $\boldsymbol{x}^{(0)}$. As such, standard tools for analysis of iterative method can not be applied. We further discuss the matter in Section 3.5.

To make connections with the classical theory of iterative methods, we give another less general sufficient condition.

Corollary 3.3.1. *If* $\mathbf{A}$ *is an* M*-matrix (defined in the proof below, see also [Saa03, Definition 1.30, Theorem 1.31]), Algorithm 1 converges to the solution* $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ *for any* $\mathbf{b}$*.*

Proof. $\mathbf{A}$ is an M-matrix if $\rho(\mathbf{I} - \mathbf{D}^{-1}\mathbf{A}) < 1$, $A_{ij} \leq 0,\ i \neq j$ and $A_{ii} > 0$, where $\mathbf{D}$ is a diagonal of $\mathbf{A}$. It means that $\widetilde{\mathbf{R}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} = \left|\widetilde{\mathbf{R}}\right|$ and $\rho\left(\left|\widetilde{\mathbf{R}}\right|\right) < 1$. □

3.3.4 Statistical interpretation of belief propagation for non-symmetric linear systems

We showed that it is possible to generalize the belief-propagation algorithm to non-symmetric linear systems. The problem is that we lost the probabilistic interpretation in the process. Indeed, with non-symmetric matrix $\boldsymbol{A}$ we can not appeal to the Gauss-Markov models because the covariance matrix should be symmetric positive-definite. Here we discuss how to regain probabilistic interpretation from the Monte Carlo method for matrix inversion described in [FL50].

The algorithm described in [FL50] works as follows. First, one picks values v_{ij}, and probabilities $p_{ij} \geq 0$ such that $v_{ij}p_{ij} = \delta_{ij} - A_{ij}$, $\sum_j p_{ij} < 1$. Next, simulate a random walk on a graph corresponding to A_{ij}. We start from vertex i and moves to vertex $j \neq i$ with probability p_{ij}, and vertex $j = i$ with probability $1 - \sum_j p_{ij}$. The weight of the walk is a random variable z:

$$z = \begin{cases} 0, \text{ end vertex is not } j; \\ v_{ii_1}v_{i_1i_2}\dots v_{i_kj} \big/ \left(1 - \sum_i p_{ji}\right), \text{ for walk } i \to i_1 \to \dots \to i_k \to j. \end{cases} \tag{3.25}$$

It is shown in [FL50] that mean value of z is $\left((I - A)^{-1}\right)_{ij}$ if eigenvalues of $I - A$ lays inside the unit circle on the complex plain.

In the same way, one can relate the i-th component of the solution of a linear system with the random walk on a graph.

Now, the message update rules that we derived were inspired by the relation between Gaussian elimination and GaBP but the formal proof relays on the connection between parameters of the messages and walks on graphs. So to regain a probabilistic interpretation of belief propagation, one needs to show

1. the relation between z and messages;

2. that z[6] is normally distributed at least for large enough walks.

The first part is somewhat technical and closely related to proofs of GaBP convergence from this chapter. The second part is non-trivial, but we expect it to hold from results similar to the central limit theorem. Currently, we do not have formal proof and leave it to further research.

3.4 Generalized Gaussian belief propagation

Many different schemes that extend belief propagation and GaBP have been developed [YFW03], [EGG12], [SWW04], [Min01], [EMK06]. In this section we are going to consider generalized belief propagation proposed in [YFW03] and subsequently developed in [YFW01a], [YFW01b] ,[YFW05]. This new algorithm is significantly more accurate [YFW05, Fig. 15] than Pearl's algorithm, but at the same time it can be computationally costly. The goal of the section is to show how particular versions of generalized GaBP can be used as a solver for non-symmetric linear systems.

3.4.1 Set-decompositions and the region graph

Generalized GaBP is a generalization of GaBP in the same sense in which the block Jacobi method is a generalization of Jacobi iteration (see Section 1.2.1 for a discussion of block relaxation method). In case of GaBP the analog of blocks is given by a set-decomposition. Namely, having variable set $S = \{1, \ldots, N\}$ we define a collection of subsets $\mathcal{P} = \{S_i : i = 1, \ldots, p\}$ such that $S_i \subset S$, and $\cup_{i=1}^{p} S_i = S$. In principle, GaBP can be applied with a sufficiently general set-decomposition, but the rules for message update becomes cumbersome (see [YFW05] for details). Because of that we consider only partitions that obeys the following rules:

1. For arbitrary $k = 1, \ldots, p$ there is no $l = 1, \ldots, p$, $l \neq k$ such that $S_k \subset S_l$.

[6] Or related random variable, for example, corresponding to the sum of closed walks on the graph.

2. For arbitrary $i = 1, \ldots, p$, $j = 1, \ldots, p$ the intersection is not in the set decomposition $S_i \cap S_j \notin \mathcal{P}$.

3. If $A_{ij} \neq 0$ there is a subset in $\mathcal{P}$ that contain both i and j, i.e., $\exists k : \{i, j\} \subset S_k$.

4. The intersection of two arbitrary subsets $\alpha_{jk} \equiv S_j \cap S_k$ can not be a subset of any other distinct intersection α_{lm}, i.e., $\forall j = 1 \ldots, p$, $\forall k = 1, \ldots, p$ there is no $l = 1, \ldots, p$, $m = 1, \ldots, p$ such that $(S_j \cap S_k) \subset (S_l \cap S_m)$ given that among S_j, S_k, S_l, S_m there are at least three distinct subsets.

A set-decomposition is called admissible if it obeys the rules above. We postpone the intuitive explanation of these technical requirements till the next section where the consistency condition will be introduced.

To exemplify the application of rules above, we consider five set-decompositions for the matrix given in Fig. 3.3:

$$\begin{aligned}
&\mathcal{P}_1 = \{\{1,2,3\},\{1,4\},\{3,4\},\{1,3\}\},\ \mathcal{P}_2 = \{\{1,2,3\},\{1,3,4\},\{1,3\}\},\\
&\mathcal{P}_3 = \{\{1,2\},\{1,4\},\{3,4\},\{2,3\}\},\ \mathcal{P}_4 = \{\{1,2\},\{1,3\},\{2,3,4\},\{1,3,4\}\},\\
&\mathcal{P}_5 = \{\{1,2,3\},\{1,3,4\}\}.
\end{aligned} \tag{3.26}$$

Set-decomposition $\mathcal{P}_1$ breaks the first rule, because $\{1,3\} \subset \{1,2,3\}$. If one removes set $\{1,3\}$ the decomposition becomes valid. The second set-decomposition $\mathcal{P}_2$ is not apt because the intersection of the first and the second sets gives the third set. Set-decomposition $\mathcal{P}_3$ is not valid according to the third criteria, because it does not contain a set that includes $\{1,3\}$ as a subset, despite the fact that $A_{31} \neq 0$. The fourth set decomposition $\mathcal{P}_4$ violates the rule four, because $\{1,3\} \cap \{2,3,4\} = \{3\} \subset \{3,4\} = \{1,3,4\} \cap \{2,3,4\}$. The last decomposition $\mathcal{P}_5$ is admissible. In a more general versions of GaBP described in [YFW05] all set-decompositions excluding $\mathcal{P}_2$ can be used.

Besides set-decomposition we need to define a region graph. For a given admissible set-decomposition $\mathcal{P}$ we define a collection of all intersections of subsets from $\mathcal{P}$, i.e., $\mathcal{L}_{\mathcal{P}} = \{S_1 \cap S_2 : S_1 \neq S_2, S_1 \in \mathcal{P}, S_2 \in \mathcal{P}\}$. Next, we define a set of vertices $\mathcal{V}_{\mathcal{R}}$ such that there is a one-to-one correspondence between each set from $\mathcal{P} \cup \mathcal{L}_{\mathcal{P}}$ and a given vertex, i.e., if $v \in \mathcal{V}_{\mathcal{R}}$ there is $S \in \mathcal{P} \cup \mathcal{L}_{\mathcal{P}}$ such that $v \sim S$. Finally, we define a set of directed edges $\mathcal{E}_{\mathcal{R}}$ such that $e_{ij} \in \mathcal{E}_{\mathcal{R}}$ iff $S_j \subset S_i$. So, we call a directed graph $\mathcal{R} = \{\mathcal{V}_{\mathcal{R}}, \mathcal{E}_{\mathcal{R}}\}$ a region graph of a given admissible set-decomposition. The region graph is illustrated in Fig. 3.3c.

For convenience we also define two additional sets:

1. $P(i) \equiv \{v_j : v_j \in \mathcal{V}_\mathcal{R}, e_{ji} \in \mathcal{E}_\mathcal{R}\}$ is a set of parents of vertex $i \in \mathcal{V}_\mathcal{R}$.

2. $C(i) \equiv \{v_j : v_j \in \mathcal{V}_\mathcal{R}, e_{ij} \in \mathcal{E}_\mathcal{R}\}$ is a set of child of vertex $i \in \mathcal{V}_\mathcal{R}$.

For example, for a region graph in Fig. 3.3c we can find $C(1) = \{\emptyset\}$, $P(1) = \{\{1,4\},\{1,2,3\}\}$, $C(123) = \{\{1\},\{3\}\}$.

We use the same notation for the variable sets:

1. $P(a) \equiv \{b : a \neq b, a \subset b, b \in \mathcal{P} \cup \mathcal{L}_\mathcal{P}\}$.

2. $C(a) \equiv \{b : a \neq b, b \subset a, b \in \mathcal{P} \cup \mathcal{L}_\mathcal{P}\}$.

Lastly, for a variable set $b \in \mathcal{P} \cup \mathcal{L}_\mathcal{P}$ we define a block matrix $\boldsymbol{A}_b : (\boldsymbol{A}_b)_{ij} = A_{ij},\ i,j \in b$. So, for the other variable set c we can find $(\boldsymbol{A}_b)_c = \boldsymbol{A}_{b\cap c}$. The last expression is defined iff $b \cap c \neq \emptyset$. The same notation is used for vectors.

3.4.2 Message update rules for the generalized Gaussian belief propagation

In this section we suppose that a pairwise Markov random field (3.9) is given along with its graph $\Gamma = \{\mathcal{V}, \mathcal{E}\}$ (see Section 3.3.1). Besides that we use an arbitrary admissible set-decomposition $\mathcal{P}$.

As we discussed in Section 3.3.1, messages for GaBP enforces consistency, i.e., ensures that $b(x_j) = \int dx_i b(x_i, x_j)$. Based on this idea the messages are defined for a generalized GaBP (see [YFW05, equation 114]). Namely, variables in the set a receives message from variables in the set b iff $a \subset b$, because in this case we should have $b(\boldsymbol{x}_a) = \int d\boldsymbol{x}_{b\setminus a} b(\boldsymbol{x}_b)$. So, messages propagate along edges of region graph defined in the previous section. Following the case of GaBP we parametrize messages from b to a as multivariate normal variables

$$m_{ba}(\boldsymbol{x}_a) \sim \exp\left(-\frac{(\boldsymbol{x}_a - \boldsymbol{\mu}_{ba})^T \boldsymbol{\Lambda}_{ba} (\boldsymbol{x}_a - \boldsymbol{\mu}_{ba})}{2}\right). \tag{3.27}$$

Each message is uniquely defined by the precision matrix $\boldsymbol{\Lambda}_{ba}$ and the mean vector $\boldsymbol{\mu}_{ba}$.

We also define a belief for a set of variables a from the intersection $a \in \mathcal{L}_\mathcal{P}$ as

$$b(\boldsymbol{x}_a) \sim \prod_{i\in a} \phi_i(x_i) \prod_{i,j\in a,\ (i,j)\in\mathcal{E}} \psi_i(x_i, x_j) \prod_{b\in P(a)} m_{ba}(\boldsymbol{x}_a), \tag{3.28}$$

and a belief for a set of variables c from the set-decomposition $f \in \mathcal{P}$:

$$b(\boldsymbol{x}_f) \sim \prod_{i \in f} \phi_i(x_i) \prod_{i,j \in f,\ (i,j) \in \mathcal{E}} \psi_i(x_i, x_j) \prod_{d \in P(a) \setminus f,\ a \in C(f)} m_{da}(\boldsymbol{x}_a). \tag{3.29}$$

Note that variables f does not receive messages directly because there is no d such that $c \subset d$. However, because f and d may contain joint a, belief $b(\boldsymbol{x}_f)$ depends on messages sent from d to a.

Suppose $a \subset f$, then a consistency condition is simply $b(\boldsymbol{x}_a) = \int d\boldsymbol{x}_{f \setminus a} b(\boldsymbol{x}_f)$. Since all variables are multivariate normal, we can derive a compact expression for belief (3.28):

$$b(\boldsymbol{x}_a) = \mathcal{N}\left(\boldsymbol{x}_a \,|\boldsymbol{\Sigma}_a \boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a\right),\ \boldsymbol{\Sigma}_a^{-1} = \boldsymbol{A}_a + \sum_{d \in P(a)} \boldsymbol{\Lambda}_{da},\ \boldsymbol{\mu}_a = \boldsymbol{b}_a + \sum_{d \in P(a)} \boldsymbol{\Lambda}_{da} \boldsymbol{\mu}_{da}. \tag{3.30}$$

The expression for (3.29) is slightly more complex:

$$\begin{aligned} &b(\boldsymbol{x}_f) = \mathcal{N}\left(\boldsymbol{x}_f \,\middle|\boldsymbol{\Sigma}_f \boldsymbol{\mu}_f, \boldsymbol{\Sigma}_f\right), \\ &\forall a \in C(f) : \left(\boldsymbol{\Sigma}_f^{-1}\right)_a = \boldsymbol{A}_{f \cap a} + \sum_{d \in P(a) \setminus f} \boldsymbol{\Lambda}_{da},\ \boldsymbol{\mu}_{f \cap a} = \boldsymbol{b}_{f \cap a} + \sum_{d \in P(a) \setminus f} \boldsymbol{\Lambda}_{da} \boldsymbol{\mu}_{da} \\ &\left(\boldsymbol{\Sigma}_f^{-1}\right)_{f \setminus \cup_{a \in C(f)} a} = \boldsymbol{A}_{f \setminus \cup_{a \in C(f)} a},\ \boldsymbol{\mu}_{f \setminus \cup_{a \in C(f)} a} = \boldsymbol{b}_{f \setminus \cup_{a \in C(f)} a}. \end{aligned} \tag{3.31}$$

From the standard identities for multivariate normal distribution we can conclude that the consistency condition gives $\left(\boldsymbol{\Sigma}_f\right)_a = \boldsymbol{\Sigma}_a$ and $\left(\boldsymbol{\Sigma}_f \boldsymbol{\mu}_f\right)_a = \boldsymbol{\Sigma}_a \boldsymbol{\mu}_a$. So, introducing the iteration index n, we obtain the following message update rules

$$\begin{aligned} \boldsymbol{\Lambda}_{fa}^{(n+1)} &= \boldsymbol{\Lambda}_{fa}^{(n)} + \left(\left(\boldsymbol{\Sigma}_f^{(n)}\right)_a\right)^{-1} - \boldsymbol{\Lambda}_a^{(n)}, \\ \boldsymbol{\Lambda}_{fa}^{(n+1)} \boldsymbol{\mu}_{fa}^{(n+1)} &= \boldsymbol{\Lambda}_{fa}^{(n)} \boldsymbol{\mu}_{fa}^{(n)} + \left(\left(\boldsymbol{\Sigma}_f^{(n)}\right)_a\right)^{-1} \left(\boldsymbol{\Sigma}_f^{(n)} \boldsymbol{\mu}_f^{(n)}\right)_a - \boldsymbol{\mu}_a^{(n)}, \end{aligned} \tag{3.32}$$

where $\boldsymbol{\Lambda}_a^{(n)} = \left(\boldsymbol{\Sigma}_a^{(n)}\right)^{-1}$, $\boldsymbol{\Sigma}_a^{(n)}$ with $\boldsymbol{\mu}_a^{(n)}$ are defined as in (3.30), and $\boldsymbol{\Sigma}_f^{(n)}$ with $\boldsymbol{\mu}_f^{(n)}$ are as in (3.31).

We want to emphasize three points about message update rules (3.32). First, in case $\mathcal{P}$ consists of all sets $\{i, j\} : A_{ij} \neq 0$, rules (3.32) reduces to (3.15) (see [YFW03]). Second, to compute updated messages one needs to perform three potentially computationally intensive operations: solve linear system $\boldsymbol{\Sigma}_f^{(n)} \boldsymbol{\mu}_f^{(n)}$, find a diagonal subblock of the inverse matrix $\left(\boldsymbol{\Sigma}_f^{(n)}\right)_a$ and invert this subblock. To still

have a complexity $O(N)$ per iteration we must restrict ourselves with a particular partitions for which matrices $\boldsymbol{\Sigma}_f$ has simple structure that enables us to perform above operations with reasonable computational complexity. In what is following we will chose f in such a way that $\boldsymbol{\Sigma}_f^{(n)}$ is a tridiagonal matrix. Third, the validity of the presented rules for nonsymmetric linear problems does not follow from the derivation above. In the two following sections we will show that (3.32) can be applied to the nonsymmetric problems as well.

On this stage it is appropriate to clarify the meaning of requirements for set-decomposition given in Section 3.4.1. The first requirement simply reduces the number of unnecessary operations. When we computed a distribution $p(\boldsymbol{x}_f)$ of larger set f we do not need to send a message to the subset u, because the computation of $p(\boldsymbol{x}_u)$ is trivial. The same is true for the second condition, because we obtain a distribution for the intersection of two sets as a byproduct of rules (3.32). The third condition ensures that partition captures the whole structure of $\boldsymbol{A}$. Suppose the third condition fails for some pair $i, j : \ A_{ij} \neq 0$. This would mean that A_{ij} never appears in rules (3.32), so we can not guaranty that marginal distribution are correct. The last condition allows us to obtain compact update rules (3.32) (see [YFW05] for more general result).

3.4.3 Elimination perspective

To motivate the applicability of (3.32) to a nonsymmetric system we consider the following block matrix, right-hand side vector, and the solution vector

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & 0 \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \boldsymbol{A}_{23} \\ 0 & \boldsymbol{A}_{32} & \boldsymbol{A}_{33} \end{pmatrix}, \ \boldsymbol{b} = \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \end{pmatrix}, \ \boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \end{pmatrix}. \tag{3.33}$$

If $\boldsymbol{A}_{33}$ is invertible we can exclude $\boldsymbol{x}_3$ from the second equation and obtain the modified system

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \widetilde{\boldsymbol{A}}_{22} \end{pmatrix}, \ \boldsymbol{b} = \begin{pmatrix} \boldsymbol{b}_1 \\ \widetilde{\boldsymbol{b}}_2 \end{pmatrix}, \ \widetilde{\boldsymbol{A}}_{22} = \boldsymbol{A}_{22} - \boldsymbol{A}_{23}\boldsymbol{A}_{33}^{-1}\boldsymbol{A}_{32}, \ \widetilde{\boldsymbol{b}}_2 = \boldsymbol{b}_2 - \boldsymbol{A}_{23}\boldsymbol{A}_{33}^{-1}\boldsymbol{b}_3. \tag{3.34}$$

Now, let α and β be variable sets, such that $\boldsymbol{x}_\alpha = \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 \end{pmatrix}^T$ and $\boldsymbol{x}_\beta = \begin{pmatrix} \boldsymbol{x}_2 & \boldsymbol{x}_3 \end{pmatrix}^T$. It is easy to find the first messages from β to $\beta \cap \alpha$ are

$$\begin{aligned} \boldsymbol{\Lambda}_{\beta\ \beta\cap\alpha}\boldsymbol{\mu}_{\beta\ \beta\cap\alpha} &= \left(\boldsymbol{A}_{22} - \boldsymbol{A}_{23}\boldsymbol{A}_{33}^{-1}\boldsymbol{A}_{32}\right)\left(\boldsymbol{B}^{-1}\begin{pmatrix} \boldsymbol{b}_2, \\ \boldsymbol{b}_3 \end{pmatrix}\right)_{\beta\cap\alpha} - \boldsymbol{b}_2 = -\boldsymbol{A}_{23}\boldsymbol{A}_{33}^{-1}\boldsymbol{b}_3 \\ \boldsymbol{\Lambda}_{\beta\ \beta\cap\alpha} &= \boldsymbol{B}_{\beta\cap\alpha}^{-1} - \boldsymbol{A}_{22} = -\boldsymbol{A}_{23}\boldsymbol{A}_{33}^{-1}\boldsymbol{A}_{32}, \ \boldsymbol{B} = \begin{pmatrix} \boldsymbol{A}_{22} & \boldsymbol{A}_{23} \\ \boldsymbol{A}_{32} & \boldsymbol{A}_{33} \end{pmatrix}, \end{aligned} \tag{3.35}$$

Algorithm 2 Generalized GaBP for a nonsymmetric linear system.

1: **Input:** invertible matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, admissible set-decomposition $\boldsymbol{P}$.
2: **Output:** approximate solution $\boldsymbol{x}$.

3: Form region graph $\mathcal{R} = \{\mathcal{V}_\mathcal{R}, \mathcal{E}_\mathcal{R}\}$ based on $\boldsymbol{A}$ and $\boldsymbol{P}$, initialize $\boldsymbol{\Lambda}_e = 0$, $\boldsymbol{\mu}_e = 0$ for all $e \in \mathcal{E}_\mathcal{R}$.
4: **while** not converge **do**
5: **for** $v \in \mathcal{V}_\mathcal{R}$, $C(v) \neq \emptyset$ **do**
6: $\boldsymbol{m}_v = \boldsymbol{b}_v$, $\boldsymbol{\Lambda}_v = \boldsymbol{A}_v$
7: **for** $u \in C(v)$ **do**
8: $\boldsymbol{m}_u = \boldsymbol{b}_u$, $\boldsymbol{\Lambda}_u = \boldsymbol{A}_u$
9: **for** $y \in P(u) \backslash v$ **do**
10: $\boldsymbol{m}_u = \boldsymbol{m}_u + \boldsymbol{m}_{yu}$, $\boldsymbol{\Lambda}_u = \boldsymbol{\Lambda}_u + \boldsymbol{\Lambda}_{yu}$
11: **end for**
12: $(\boldsymbol{m}_v)_u = (\boldsymbol{m}_v)_u + \boldsymbol{m}_u$, $(\boldsymbol{\Lambda}_v)_u = (\boldsymbol{\Lambda}_v)_u + \boldsymbol{\Lambda}_u$
13: **end for**
14: $\boldsymbol{x}_v = \boldsymbol{\Lambda}_v^{-1} \boldsymbol{m}_v$, $\boldsymbol{\Lambda}_v = \boldsymbol{\Lambda}_v^{-1}$
15: **for** $u \in C(v)$ **do**
16: $\boldsymbol{\Lambda}_{vu} = ((\boldsymbol{\Lambda}_v)_u)^{-1} - \boldsymbol{\Lambda}_u$, $\boldsymbol{m}_{vu} = ((\boldsymbol{\Lambda}_v)_u)^{-1} (\boldsymbol{x}_v)_u - \boldsymbol{m}_u$
17: **end for**
18: **end for**
19: **end while**

where we use zero starting messages, equation (3.32) and standard formulæ for block matrix inverse.

As we can see, messages coincides with corrections that the second equation receives when the third variable is eliminated. Since Gaussian elimination works for non-symmetric matrices, the results of this section suggests that we can use generalized GaBP for non-symmetric matrices.

3.4.4 Generalized Gaussian belief propagation for nonsymmetric linear systems

Based on considerations above we formally define Algorithm 2. A few comments are in order. First, line 14 of Algorithm 2 contains $\boldsymbol{\Lambda}_v^{-1}$ which should not be considered literally. Because only diagonal subblocks of $\boldsymbol{\Lambda}_v^{-1}$ are used later, it is preferable to compute and store only them. Second, Algorithm 2 is sequential. The most nat-

ural parallelization strategy is to compute $\boldsymbol{m}_u$ such that $C(u) = \emptyset$ and after that update messages from $v : C(v) \neq \emptyset$. We use this strategy in line GaBP smoothers in Section 3.6. Third, the computational complexity of Algorithm 2 strongly depends on set-decomposition $\mathcal{P}$ and matrix $\boldsymbol{A}$. We provide estimates for a concrete partition in Section 3.6.1.

The Algorithm 2 can be justified theoretically on the basis of the following two theorems.

Theorem 3.4.1. *If there is $N \in \mathbb{N}$ such that $\mathbf{m}_e^{(N+k)} = \mathbf{m}_e^{(N)}$, $\boldsymbol{\Lambda}_e^{(N+k)} = \boldsymbol{\Lambda}_e^{(N)}$ for all $e \in \mathcal{E}_{\mathcal{R}}$ and for any $k \in \mathbb{N}$, then for each set $v : C(v) \neq \emptyset$ we obtain $\boldsymbol{\Lambda}_v^{-1}\boldsymbol{m}_v = (\mathbf{A}^{-1}\mathbf{b})_v$ (see Algorithm 2 for details).*

That is, the steady state of the message flow, if it exists, corresponds to the exact solution. The proof is given in Section 10.3.

For the sufficient condition for convergence we need to construct a particular block matrix based on set-partition $\mathcal{P}$ and matrix $\boldsymbol{A}$. First, we define a set of variable subsets

$$F \equiv \mathcal{L} \cup \left\{ S \backslash \cup_{a \in C(S)} a : S \in \mathcal{P} \right\}. \tag{3.36}$$

Using F, we form a partition of the matrix $\mathbf{A}$ and the right-hand-side vector $\mathbf{b}$

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{ii} & \mathbf{A}_{ij} & \dots \\ \mathbf{A}_{ji} & \mathbf{A}_{jj} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \mathbf{b}_i \\ \mathbf{b}_j \\ \vdots \end{pmatrix}, \tag{3.37}$$

where each diagonal block corresponds to the element of the set F. We also define

$$\widetilde{\mathbf{A}}_{ij} = \mathbf{A}_{ii}^{-1}\mathbf{A}_{ij} \equiv \mathbf{I}_{ij} - \widetilde{\mathbf{R}}_{ij}; \ \left\|\widetilde{\mathbf{R}}\right\|_{ij} \equiv \left\|\widetilde{\mathbf{R}}_{ij}\right\|, \ \widetilde{\mathbf{b}}_i = \mathbf{A}_{ii}^{-1}\mathbf{b}_i. \tag{3.38}$$

Note that the first equality after the semicolon in the preceding equation contains a definition of matrix $\left\|\widetilde{\mathbf{R}}\right\|$, which depends on the operator norm $\|\cdot\|$ (see [HJ13, ch. 5, Definition 5.6.3]).

The following statement gives sufficient conditions for convergence.

Theorem 3.4.2. *If for matrix* (3.37) *which is based on partition* (3.36) $\det \mathbf{A}_{ii} \neq 0 \ \forall i$ *and* $\rho\left(\left\|\widetilde{\mathbf{R}}\right\|\right) < 1$ *in some operator norm, then two-layer generalized GaBP (Algorithm 2) converges to the exact solution* $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

The proof of this theorems appears in Section 10.4. From the second part of the argument in Section 10.4.2, one can deduce the following

Corollary 3.4.1. *Generalized GaBP (Algorithm 2) converges whenever GaBP converges (Theorem 3.3.2) and all submatrices corresponding to the large blocks are invertible (see equations* (3.36), (3.37)*).*

The opposite does not hold. For example, consider a matrix

$$\mathbf{A} = \begin{pmatrix} 10 & 1.5 & 2 & 2 & 0 & 2 & 0 \\ 2 & 4 & 2.5 & 0 & 2 & 0 & 0 \\ 2 & 3 & 5 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 10 & 0.5 & 1 & 0 \\ 0 & 2 & 0 & 0.5 & 5 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 & 7 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{pmatrix}, \tag{3.39}$$

$\mathbf{A}_{11} \in \mathbb{R}^{3\times 3},\ \mathbf{A}_{22} \in \mathbb{R}^{2\times 2},\ \mathbf{A}_{33} \in \mathbb{R}^{2\times 2}.$

In this case, the spectral radius of matrix $\left|\widetilde{\mathbf{R}}\right|$ defined in Theorem 3.3.2 equals ~ 1.03 and GaBP diverges[7]. On the other hand, the spectral radius of $\left\|\widetilde{\mathbf{R}}\right\|$ defined by (3.38) and the partition given in (3.39) is smaller than one in l_∞ and spectral norms [HJ13, Examples 5.6.5, 5.6.6] (see https://github.com/VLSF/GaBPsolvers for further details).

3.5 Gaussian belief propagation as a smoother for multigrid method

As explained in Section 1.2.3, where an introduction to multigrid method is given, the smoother should be a mapping $\mathbf{S} : \mathbf{x}^n \to \mathbf{x}^{n+1}$. Although GaBP is not of this form, one can use an error correction scheme as explained in Algorithm 3. The part "Apply k sweeps of Algorithm 1" means that one should replace a while-loop with for-loop specifying k iterations. In the next two subsections we analyze properties of GaBP in the error-correction regime (Algorithm 3) and estimate its computational complexity.

[7]Note that the divergence of GaBP does not follow from $\left|\widetilde{\mathbf{R}}\right| > 1$ as Theorem 3.3.2 provides only sufficient conditions. For this particular case, the pathological behavior of GaBP follows from the numerical experiment (see https://github.com/VLSF/GaBPsolvers for details).

Algorithm 3 GaBP as a smoother.

Compute a residual $\mathbf{r}^n = \mathbf{b} - \mathbf{A}\mathbf{x}^n$.
Apply k sweeps of Algorithm 1 or 2 to the linear system $\mathbf{A}\mathbf{e} = \mathbf{r}^n$.
Perform an error correction $\mathbf{x}^{n+1} = \mathbf{x}^n + \mathbf{e}^\mu$.

3.5.1 Gaussian belief propagation in the error correction scheme

As we discussed in Section 3.3.3 GaBP modifies error propagation matrix $\boldsymbol{M}$ which makes the analysis of convergence nontrivial. Because in Algorithm 3 we performs a fixed predetermined number of sweeps, it is possible to exclude messages and find explicit representation of GaBP smoother. For convenience we write $S^{(k)}_{\text{GaBP}}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{A})$ as a shortcut for k iterations of Algorithm 1 applied as a smoother Algorithm 3. Besides sequential version of GaBP (Algorithm 1) we also consider a parallel version. In the parallel variant we compute all incoming messages simultaneously and do not transverse the graph node by node. The difference between sequential and parallel GaBP is similar to the difference between multiplicative and additive solvers. For the parallel version of Algorithm 1 the following result holds

Lemma 3.5.1. *Parallel GaBP with $k = 1, 2$ iterations is equivalent to the following linear iteration*

$$S^{(k)}_{\textit{parallel GaBP}}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{A}) = \boldsymbol{x} + \boldsymbol{D}(\boldsymbol{B}^{(k)}\boldsymbol{A})^{-1}\boldsymbol{B}^{(k)}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) \tag{3.40}$$

where $\boldsymbol{D}(\boldsymbol{B}^{(k)}\boldsymbol{A})$ is a diagonal part of $\boldsymbol{B}^{(k)}A$, $\boldsymbol{B}^{(1)} = \boldsymbol{I}$, and $\boldsymbol{B}^{(2)} = 2\boldsymbol{I} - \boldsymbol{A}\boldsymbol{D}(\boldsymbol{A})^{-1}$.

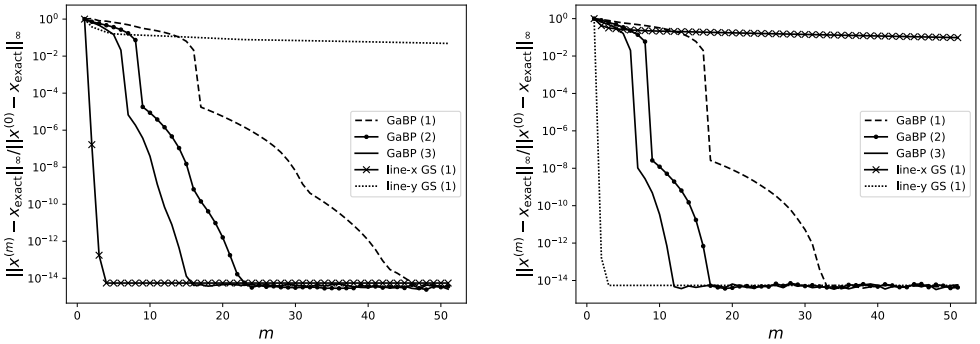
We see that $k = 1, 2$ sweeps of parallel GaBP coincides with preconditioned Jacobi iteration where matrix $\boldsymbol{B}^{(k)}$ serves as preconditioner. It is certainly possible to exclude messages for $k > 2$ but the resulting smoother can not be represented in a compact form. From the result Lemma 3.5.1 and the standard multigrid theory [TOS00, Sections 2.1.2 and 4.3] we can conclude that smoothing properties of parallel GaBP are poor. For the sequential version of Algorithm 1 the following result holds

Lemma 3.5.2. *Sequential GaBP with $k = 1$ iteration is equivalent to the following linear iteration*

$$S^{(1)}_{\textit{sequential GaBP}}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{A}) = \boldsymbol{x} + \boldsymbol{L}(\boldsymbol{C})^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}), \tag{3.41}$$

where $\boldsymbol{L}(\boldsymbol{C})$ is a lower-triangular (diagonal is included) part of $\boldsymbol{C}$, $C_{ij} = A_{ij}$ for $i \neq j$, and the diagonal elements of $\boldsymbol{C}$ are defined recursively as

$$i = 1: \; C_{11} = A_{11}; \; i > 1: \; C_{ii} = A_{ii} - \sum_{j<i} A_{ij}A_{ji}/C_{jj}. \tag{3.42}$$



(a) (b)

Figure 3.4: Convergence histories for different anisotropies: (a) Second derivative along y is multiplied by $\epsilon = 10^{-5}$ (see (1.20)) (b) Second derivative along x is multiplied by $\epsilon = 10^{-8}$ (see (1.21)). In both cases, the fine grid consists of $(2^5 - 1)$ points, and the coarsest grid consists of $(2^3 - 1)$ points along each coordinate line. GaBP (k), $k = 1, 2, 3$ refers to Algorithm 3 with k sweeps, line-x GS (1) and line-y GS (1) are block Gauss-Seidel smoother. We can see that depending on the direction of anisotropy either line-x GS (1) or line-y GS (1) stagnates, whereas GaBP remains efficient.

So one sweep of sequential GaBP resembles Gauss-Seidel iteration but with modified lower part of matrix $\boldsymbol{A}$. Note, that (3.41) can not be regarded as a Gauss-Seidel method with left preconditioner because the residual is not modified. It is also easy to see that the diagonal of $\boldsymbol{C}$ is exactly the same as the diagonal of $\boldsymbol{U}$ in incomplete LU decomposition with zero fill-in (ILU(0) in what is following) [CP15]. As we will see momentarily, this form of $\boldsymbol{C}$ explains a robustness of GaBP.

To observe practical implications of the difference between GaBP and Gauss-Seidel, we consider the matrix following from standard finite difference approximation of the elliptic boundary value problems (1.20) and (1.21).

We solve the resulting linear problem with geometric multigrid with one presmoothing and one postsmoothing steps (see Section 3.6 for the detailed description of the multigrid used). We use two two types of smoothers: line Gauss-Seidel smoothers [TOS00, p. 5.1.3], and GaBP. In Fig. 3.4a we can see the results for equation (1.20) with $\epsilon = 10^{-5}$, and in Fig. 3.4b the results for equation (1.21) with $\epsilon = 10^{-8}$ are given. There are three thing we want to stress. First, from results in Fig. 3.4 we can see that GaBP is robust, unlike line Gauss-Seidel smoothers the performance of which depends on the direction of anisotropy. Second, with the increase of anisotropy, con-

vergence of GaBP improves. This can be explained as follows. Recall, that GaBP coincides with Gaussian elimination when the graph of a matrix is a tree. When we increase anisotropy, lines decouple from each other, which removes loops from the graph, so GaBP becomes an exact solver. Third, from convergence histories of GaBP we can observe a number of iterations m corresponding to the sharp drop of the relative error. Namely, $m \simeq 16$, $m \simeq 8$, $m \simeq 5$ for GaBP (1), GaBP (2), GaBP (3) respectively. To explain this behavior we recall, that for anisotropic equation under consideration GaBP essentially performs Gaussian elimination. As such, after the first presmoothing step of GaBP (1) we obtain an almost exact solution in $\sqrt{N}$ points corresponding to variables $u(x_1, 1-h)$ or $u(1-h, x_2)$ (depending on the direction of the anisotropy), where h is a grid spacing and N is a total number of unknown. After the first postsmoothing step of GaBP (1) we will recover $u(x_1, 1-2h)$ or $u(1-2h, x_2)$ almost exactly. Since the grid consists of $2^5 - 1$ points in each direction, elimination recovers almost correct solution after $m \simeq [(2^5 - 1)/2] = 16$ iteration. The same analysis works for GaBP (2) and GaBP (3). From this behavior we expect that the convergence rate of GaBP will deteriorate with the increase of number of variables N, and will improve with the increase of the anisotropy. We further study the anisotropic problem in Section 3.6.

The results obtained in this section suggest a way to analyze the smoothing properties of GaBP. The strategy is to, first, exclude messages and present GaBP in the form of standard smoother $\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \boldsymbol{N}\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(n)}\right)$. After that, local (or rigorous) Fourier analysis can be performed the same way as it is done for other linear iteration (see, for example, [TOS00, Chapter 4, Section 7.5]). Having the assurance that GaBP smoother can be analyzed by standard means, we do not pursue this topic further.

3.5.2 Reducing computational complexity

The number of floating point operations per iteration for Algorithm 1 and Algorithm 2 depends on the graph of the matrix $\boldsymbol{A}$ and the set-decomposition $\mathcal{P}$. As an example we compute the number of floating point operations in Algorithm 1 in case $\boldsymbol{A}$ corresponds to the operator with the five-point stencil, i.e., the standard second order discretization of Laplace operator. For convenience, we split Algorithm 1 (sequential version) into three parts:

- Accumulation stage. Λ and m are computed (line 8).
- Update stage. New messages $\widetilde{\boldsymbol{\Lambda}}$ and $\boldsymbol{m}$ are constructed from the previous ones (line 12).

Solver	Computational complexity per N	
	five-point stencil	nine-point stencil
GaBP	$12m + 12$	$24m + 20$
Jacobi	$9m$	$17m$
Chebyshev $m \geq 2$	$16(m-2) + 25$	$24(m-2) + 41$
line GaBP	$22m + 11$	$22m + 19$
Gauss-Seidel	$9m$	$17m$
alternating zebra GS	$26m$	$42m$
ILU(0)	$9m + 11$	$17m + 19$

Table 3.1: Table contains computational complexity of GaBP (error correction scheme) with precomputed $\mathbf{\Lambda}$ messages, and other relaxation schemes. Number of sweeps in m, number of variables in N, matrix $\boldsymbol{A}$ corresponds to five-point or nine-point stencil. Details on solvers can be found in Section 3.5.2, Section 3.6.1 and Table 3.2.

- Termination stage. The final answer m/Σ is obtained (line 10).

We also neglect all effects from boundaries. Under these assumptions, the number of floating point operations for the single sweep GaBP is

$$\#_{\text{GaBP(1)}} = \underbrace{10N}_{\mathbf{r}=\mathbf{b}-\mathbf{A}\mathbf{x}_0} + \underbrace{2N + 4N}_{\text{accumulate}} + \underbrace{4N + 6N}_{\text{update}} + \underbrace{N}_{\text{terminate}} + \underbrace{N}_{+\mathbf{x}_0} = 28N. \tag{3.43}$$

Here we take into account that we do not need to receive messages from nodes that we have not visited yet, nor send messages to already visited nodes.

In the context of multigrid, it is important to have a cheap smoother, so to reduce computational complexity we precompute all required messages $\widetilde{\mathbf{\Lambda}}$, which depend only on the matrix $\boldsymbol{A}$ and not on the right-hand-side vector. The number of operations is modified as follows

$$\#_{\text{GaBP(1)}} = \underbrace{10N}_{\mathbf{r}=\mathbf{b}-\mathbf{A}\mathbf{x}_0} + \underbrace{2N}_{\text{accumulate}} + \underbrace{4N}_{\text{update}} + \underbrace{N}_{\text{terminate}} + \underbrace{N}_{+\mathbf{x}_0} = 18N. \tag{3.44}$$

Here we used the fact that some messages are absent or need not be computed. In Table 3.1, where computational complexity for the solvers is listed, we compute number of operations for the worst case scenario.

3.6 Numerical examples

We consider linear systems of equations arising from finite difference and bilinear finite element discretizations of elliptic equations with smooth coefficients in two space dimensions. Model equations are given in Section 1.3. The multigrid solver is constructed as follows. The finest uniform grid along x direction contain $2^J + 1$ points $\mathcal{G}_J = \left\{ih : h = 2^{-J}, i = 0, \ldots, 2^J\right\}$. The coarsening scheme is such that the next grid $\mathcal{G}_{J-1}$ contains each second point. The full grid is a direct product of two unidimensional grids. The coarsest grid always contains 25 points in total. We use LU as a solver on the coarsest grid. As transfer operators we choose full weighting restriction [TOS00, eq. 2.3.3] and bilinear interpolation [TOS00, eq. 2.3.7]. We always use a V-cycle. The number of sweeps for pre- and postsmoothing steps is the same an given in brackets, for example, "ILU(0) (2)" refers to two sweeps of incomplete LU. Properties of smoothers are collected in Table 3.1, Table 3.2.

We also list results for multigrid used as a preconditioner. We test CG and GMRES.

As a performance measure for multigrid we use $\hat{\rho} = \sqrt[\hat{N}]{\left\|e^{(\hat{N})}\right\|_\infty / \left\|e^{(0)}\right\|_\infty}$, where $e^{(\hat{N})}$ is an error on iteration $\hat{N}$, which is chosen such that $\hat{\rho}^{\hat{N}} \leq 10^{-10}$. The same $\hat{N}$ is used to compare the efficiency of preconditioners.

Following [Hac16, (2.31a)] we define effective amount of work $\mathsf{Eff} \equiv N_{\mathsf{solver}} N_{\mathsf{eff}}$, where N_{solver} estimates the number of floating-point operations per N performed by solver, and N_{eff} is a reference number of iterations chosen as $N_{\mathsf{eff}} \equiv -1 / \log_{10} \hat{\rho}$ for a solver and $N_{\mathsf{eff}} \equiv \hat{N}$ for a preconditioner.[8] For multigrid $N_{\mathsf{solver}} \equiv \sum_{k=0}^{\infty} (2N_s + \delta) / 4^k = 4(2N_s + \delta) / 3$, where δ (21 for five-point stencil, 29 for nine-point stencil) is the number of operations needed for error-correction, projection, interpolation, and N_s can be found in Table 3.1. For stand-alone solvers we use $N_{\mathsf{solver}} = N_s / m$.

Table 3.3, Table 3.4, Table 3.5, Table 3.6, Table 3.7 contain results (spectral radius $\rho \in [0, 1)$, which is located in the upper half of each cell, and the effective amount of work which is situated in the lower half of each cell) for multigrid used as a stand-alone solver (the left part of the table) as well as a preconditioner (the right part of the table). Table 3.8 contains results for stand-alone solvers.

Solver	Independent points	Reference
parallel GaBP	N	Algorithm 1
Jacobi, Chebyshev	N	[Saa03, Eq. (4.5)], [Bra+15, Algorithm 3.1]
red-black schemes	$N/2$	[TOS00, equation (2.1.11), Remark (2.1.1)]
four color schemes	$N/4$	[TOS00, Remark (5.4.5)]
line GaBP	$\sqrt{N}$	Algorithm 2, Section 3.6.1
alternating zebra GS	$\sqrt{N}/2$	[TOS00, Section 5.1, 5.2]
sequential GaBP, GS	$\leq \sqrt{N}$	Algorithm 1, [Saa03, Eq. (4.8)]
ILU(0)	$\leq \sqrt{N}$	[CP15, Algorithm 1], [TOS00, Section 7.5]

Table 3.2: In this table we collect degree of parallelism of smoothers. First column contain the names of the solvers. Second column lists number of operations that can be performed in parallel for a linear operator with five-point stencil on the square 2D grid with N points in total (see Section 3.6.1 for details). The last column contain a reference, where the precise information about the smoother can be found.

3.6.1 Notes about solvers and smoothers

The computational complexity of solvers is given in Table 3.1. The degree of parallelism is estimated in Table 3.2. To exemplify how these numbers are computed consider a Gauss-Seidel iteration for a linear operator with five-point stencil on square grid with N points in total. For each sweep we start by obtaining the value of variable $(i, j) = (1, 1)$. After that, we can compute variables $(i + 1, j)$, $(i, j + 1)$. On the next step we can obtain variables with coordinates $(i + 2, j)$, $(i + 1, j + 1)$, $(i, j + 2)$, and so on. This means at best we have $\sqrt{N}$ number of points we can process in parallel.

We would like to add that there are various parallel versions of classical algorithms. For example, it is possible to construct a parallel version of Gauss-Seidel method [Ada+03] and cyclic reduction [GG97] for a parallel solution of tridiagonal linear systems (e.g., in application to line smoothers).

Polynomial smoothers require $\lambda_{\max}$ and $\lambda_{\min}$. Following [Ada+03] we use $\lambda_{\max} = 1.1\rho(\boldsymbol{A})$ and $\lambda_{\min} = 0.3\rho(\boldsymbol{A})$ where $\rho(\boldsymbol{A})$ is a maximal eigenvalue estimated with projection method.

Line GaBP is an example of generalized GaBP smoother. The set-decomposition

[8]Note, that our definition of Eff is suitable for comparison of solvers with solvers and preconditioners with preconditioners. It is meaningless to use Eff (as it is defined here) to compare solver with preconditioner.

	$\hat{\rho}$ Eff				$\hat{N}$ Eff			
smoother$/J$	4	5	6	7	4	5	6	7
Gauss-Seidel (3)	.79 977	.94 3721	.98 11397	.99 22911	20 1468	40 2908	310 22348	150 10828
GaBP (3)	.03 102	.08 142	.32 315	.6 703	10 1308	10 1308	40 5148	90 11548
zebra line GS (2)	0 0	0 0	0 0	0 0	2 305	2 305	4 583	4 583
line GaBP (2)	0 0	0 0	0 0	0 0	5 761	5 761	6 908	3 468
red-black GS (3)	.78 927	.93 3173	.98 11397	.99 22911	20 1468	40 2908	120 8668	340 24508
red-black GaBP (3)	.48 489	.84 2060	.95 7003	.97 11793	20 2588	40 5148	80 10268	200 25628
ILU(0) (2)	0 0	0 0	0 0	0 0	4 337	2 183	6 492	4 337
Chebyshev (4)	.85 2550	.95 8080	.97 13607	.97 13607	30 4588	60 9148	170 25868	>500 >76028
Chebyshev (8)	.75 2807	.92 9684	.96 19780	.97 26509	20 6481	40 12935	90 29068	300 96828

Table 3.3: Poisson equation (1.12), finite difference, conjugate gradient.

which defines this solver consists of all horizontal and vertical lines. For example, on grid with 3 points in each direction the set-decomposition used in Algorithm 2 is $\mathcal{P} = \{\{1,2,3\},\{4,5,6\},\{7,8,9\},\{1,4,7\},\{2,5,8\},\{3,6,9\}\}$.

3.6.2 Summary of results

Poisson equation

The results for this classical test problem (1.12) are given in Table 3.3. We can see that red-black GaBP has better convergence rate than other solvers. The closes two are zebra line Gauss-Seidel and ILU(0). From the comparison of the effective amount of work, we can conclude that red-black Gauss-Seidel is slightly more efficient than red-black GaBP.

Large mixed derivative

In this section we discuss (1.19) with $\tau = 1 - \delta/2$. When $\delta \geq 0$ is small, the equation almost losses the ellipticity which presents challenges for classical multigrid components. From the convergence rates presented in Table 3.4 we can see that sequential GaBP versions are more robust. Indeed, for GaBP and four-color GaBP the convergence rates are better than the ones for Gauss-Seidel and four-color Gauss-Seidel. These results become less pronounced for larger grids. Comparing Eff, we conclude that GaBP is more efficient than Gauss-Seidel, and four-color GaBP is more efficient on the smallest grid but loses its advantage on larger grids. It is also interesting that

	$\hat{\rho}$ Eff				$\hat{N}$ Eff			
smoother$/J$	4	5	6	7	4	5	6	7
Gauss-Seidel (2)	.52 455	.73 946	.85 1832	.92 3572	20 1852	20 1852	30 2759	50 4572
GaBP (2)	.34 470	.54 822	.75 1761	.85 3117	20 3665	20 3665	30 5479	50 9105
zebra line GS (2)	.27 462	.52 925	.72 1841	.82 3048	13 2951	20 4519	20 4519	40 8999
line GaBP (2)	.54 772	.73 1512	.83 2554	.89 4084	20 3399	20 3399	30 5079	50 8439
four-color GS (2)	.56 514	.71 870	.83 1598	.89 2555	19 1761	20 1852	30 2759	50 4572
four-color GaBP (2)	.31 433	.57 901	.75 1761	.85 3117	10 1852	20 3665	30 5479	40 7292
ILU(0) (2)	.13 203	.4 452	.6 811	.78 1668	10 1452	17 2441	20 2865	30 4279
Chebyshev (4)	.49 891	.72 1935	.84 3645	.88 4971	20 4785	20 4785	40 9532	50 11905
Chebyshev (8)	.32 1075	.6 2398	.76 4464	.85 7537	10 4972	20 9905	30 14839	40 19772

Table 3.4: Mixed derivative (1.19) with $\tau = 1 - \delta/2$, $\delta = 0.01$, finite difference, restarted GMRES (10).

line GaBP seems to be preferable over zebra line GS as a preconditioner for $J = 7$ even though it is inferior as a solver. Overall, for this problem all smoothers perform poorly. Incomplete LU shows the best convergence rate, but Chebyshev (8) is more preferable because of the excellent parallelelism.

Boundary layer

For small ϵ in (1.24) with $v_x = v_y = 1/\epsilon$ we have advection-dominated diffusion. The results are presented in Table 3.5. All smoothers are reasonable for this problem. We can see that sequential GaBP produces a slightly better convergence rate than Gauss-Seidel. The same is true for line versions because line GaBP is more parallel than line Gauss-Seidel. Overall, it is reasonable to use either Chebyshev or any of the red-black smoothers for this problem. Incomplete LU, having the smallest Eff, is also an option but looks less attractive because of the poor parallelism.

Anisotropic Poisson

Here we continue the study of anisotropic problem Eq. (1.21) that we mentioned in Section 3.5.1. The standard approach is to use semi-coarsening, but we will continue to use full-coarsening to investigate the effect of the smoother.

For this problem ILU(0) and line smoothers are essentially the exact solvers, so they produce almost correct solution after a single iteration. This is indicated by $\hat{\rho} = 0$, Eff $= 0$ in Table 3.6 where the convergence rates for this problem are

	$\hat{\rho}$ Eff				$\hat{N}$ Eff			
smoother/J	4	5	6	7	4	5	6	7
Gauss-Seidel (2)	.02 45	.05 58	.06 62	.07 66	9 460	10 508	10 508	10 508
GaBP (2)	.007 58	.03 81	.04 89	.05 95	10 988	10 988	10 988	10 988
zebra line GS (2)	.004 70	.01 83	.02 98	.02 98	10 1415	16 2247	9 1276	9 1276
line GaBP (2)	.02 103	.05 134	.06 143	.06 143	10 1495	10 1495	10 1495	9 1348
red-black GS (2)	.06 62	.06 62	.06 62	.05 58	20 988	20 988	10 508	13 652
red-black GaBP (2)	.05 95	.05 95	.05 95	.05 95	10 988	11 1084	20 1948	11 1084
ILU(0) (2)	.0006 33	.005 46	.01 53	.01 53	5 415	7 569	20 1575	10 801
Chebyshev (4)	.16 226	.07 156	.08 164	.1 180	10 1548	10 1548	10 1548	10 1548
Chebyshev (8)	.02 206	.02 206	.03 230	.03 230	10 3255	10 3255	10 3255	10 3255

Table 3.5: Boundary layer (1.24) with $v_x = v_y = \epsilon^{-1}$, $\epsilon = 0.05$, finite difference, restarted GMRES (10).

	$\hat{\rho}$ Eff				$\hat{N}$ Eff			
smoother/J	4	5	6	7	4	5	6	7
Gauss-Seidel (3)	.79 977	.94 3721	.98 11397	.99 22911	20 1468	40 2908	310 22348	150 10828
GaBP (3)	.03 102	.08 142	.32 315	.6 703	10 1308	10 1308	40 5148	90 11548
zebra line GS (2)	0 0	0 0	0 0	0 0	2 305	2 305	4 583	4 583
line GaBP (2)	0 0	0 0	0 0	0 0	5 761	5 761	6 908	3 468
red-black GS (3)	.78 927	.93 3173	.98 11397	.99 22911	20 1468	40 2908	120 8668	340 24508
red-black GaBP (3)	.48 489	.84 2060	.95 7003	.97 11793	20 2588	40 5148	80 10268	200 25628
ILU(0) (2)	0 0	0 0	0 0	0 0	4 337	2 183	6 492	4 337
Chebyshev (4)	.85 2550	.95 8080	.97 13607	.97 13607	30 4588	60 9148	170 25868	>500 >76028
Chebyshev (8)	.75 2807	.92 9684	.96 19780	.97 26509	20 6481	40 12935	90 29068	300 96828

Table 3.6: Anisotropic Poisson equation Eq. (1.21) with $\epsilon = 10^{-6}$, finite difference, restarted GMRES (10)

	$\hat{\rho}$ Eff				$\hat{N}$ Eff			
smoother$\big/J$	4	5	6	7	4	5	6	7
Gauss-Seidel (2)	.04 93	.1 129	.23 203	.54 483	6 583	7 673	8 764	10 945
GaBP (2)	.03 144	.08 201	.21 325	.49 710	6 1127	7 1308	8 1489	10 1852
zebra line GS (2)	.03 172	.06 215	.2 376	.56 1043	5 1159	6 1383	7 1607	9 2055
line GaBP (2)	.23 324	.23 324	.23 324	.52 728	9 1551	9 1551	10 1719	11 1887
four-color GS (2)	.02 76	.07 112	.22 197	.55 498	5 492	6 583	7 673	9 855
four-color GaBP (2)	.02 129	.07 190	.21 325	.49 710	5 945	8 1489	9 1671	12 2215
ILU(0) (2)	.013 95	.06 147	.14 211	.49 581	4 604	5 745	5 745	8 1169
Chebyshev (4)	.03 181	.09 264	.16 347	.52 972	5 1225	6 1463	7 1700	9 2175
Chebyshev (8)	.016 296	.07 461	.18 714	.54 1988	5 2505	6 2999	7 3492	8 3985

Table 3.7: Helmholtz equation (1.23) with $k^2h = 0.1$, finite element, conjugate gradient

summarized. GaBP clearly presents the best alternative among the other solvers, at least when the grid size is not too large. We can see that, as was expected, the convergence rate deteriorates when J increases. Nevertheless, this deterioration is less pronounced than the decline of $\hat{\rho}$ for other points smoothers. Analysis of Eff shows that among the "exact" solvers line GaBP seems to be preferable as a preconditioner. Again, ILU(0) has the smallest Eff but can not be effectively run in parallel.

Helmholtz

Here we consider finite elements diszretization of Helmholtz equation (1.23) with fixed k^2h. The condition $k^2h = \text{const}$ means a fixed number of wavelength per grid spacing, this allows us to avoid the pollution effect [BS97]. Table 3.7 indicates that for the largest grid ILU(0), four-color GaBP and GaBP has the same (smallest) convergence rate. Among color smoothers (and overall), Gauss-Seidel has the smallest Eff. It is also the best preconditioner. Comparison of line smoothers reveals that GaBP is more efficient than Gauss-Seidel both as a solver and a preconditioner.

Testing GaBP as a stand-alone solver

To compare GaBP with classical iteration we use two equations. The first one is a Mehrstellen discretization (1.17). The second equation is the anisotropic equation with space-dependent coefficients (1.22).

The results for both equations are given in Table 3.8.

Convergence rates for Mehrstellen discretization indicate that four-color Gauss-Seidel is the most cost-efficient solver. One can see that GaBP solvers result in a faster convergence rate for the smaller grids. However, the increased cost of GaBP solvers makes them less preferable. Interestingly, on the finest grid line GaBP seems to be more efficient than zebra line GS.

For the anisotropic problem, the most efficient solver is either line GaBP or zebra line GS. The latter one has smaller Eff, but since parallel properties of line GaBP are superior, this difference is mitigated, and line GaBP is at least as efficient as line GS. We also would like to mention that sequential GaBP outperforms sequential Gauss-Seidel on the finest grid. The same is true for red-black versions, although the difference is negligible.

Discussion

We have seen that for all test problems, GaBP smoothers results in solvers with smaller or similar $\hat{\rho}$ compared to classical smoothers. However, GaBP can be less efficient because the presence of messages leads to an increased number of floating-point operations. Two other trends are evident. First, GaBP has advantages in situations when there are weakly linked small clusters of variables. Second, for large grids, GaBP smoothers resemble the classical ones. The situation is less clear for the generalized GaBP solvers and smoothers because their performance depends on the set-decomposition $\mathcal{P}$.

Set-decomposition for generalized GaBP is not a well-studied subject. Contributions [YFW05], [Wel04] contain some general recommendations on the construction of set-decomposition. The technique of embedded trees [SWW04] is also related to the choice of good set-decomposition. To the best of our knowledge, the literature completely lacks results on the influence of set-decomposition on the quality of generalized GaBP smoother. We leave this topic for further research.

	$\hat{\rho}_{\mathsf{Eff}}$, Mehrstellen				$\hat{\rho}_{\mathsf{Eff}}$, Poisson			
smoother$/J$	3	4	5	6	3	4	5	6
Jacobi	.89 336	.94 633	.96 959	.96 959	.91 220	.96 508	.98 1026	.98 1026
Gauss-Seidel	.81 186	.92 469	.95 763	.95 763	.84 119	.94 335	.96 508	.98 1026
GaBP (2)	.76 243	.89 573	.94 1079	.95 1302	.71 121	.86 275	.94 670	.95 808
zebra line GS	.70 271	.87 694	.94 1563	.95 1885	.70 168	.71 175	.80 268	.88 468
line GaBP (2)	.83 389	.92 870	.94 1172	.96 1777	.71 185	.82 319	.89 543	.92 759
color GS	.81 186	.92 469	.94 633	.96 959	.84 119	.93 286	.96 508	.98 1026
color GaBP (2)	.76 243	.91 708	.94 1079	.96 1636	.70 116	.86 275	.94 670	.96 1015

Table 3.8: Mehrstellen discretization (1.17) and anisotropic Poisson equation (1.22) with $\epsilon = 10^{-3}$, finite difference.

Chapter 4

Probabilistic projection methods

4.1 Projection methods and statistical inference

Projection methods briefly described in Section 1.2.2 is versatile and reliable way to construct approximate solutions. Projection methods are known to be related to the best approximation with orthogonal polynomials (symmetric Lanczos algorithm [Saa03, Section 6.6.2]), Gaussian elimination (conjugate direction method [She+94, Section 7.2]) and minimization problem (conjugate gradient method [Hac16, Section 10.1.4], [She+94, Section 3]). These connections are important because they can be used to gain additional understanding, estimate convergence, construct new proofs.

Recently additional interpretation emerged. A series of papers starting with the work on probabilistic reconstruction of quasi-Newton methods [HK13] led to Bayesian projection methods [Hen15], [Coc+19a], [Bar+19]. By probabilistic projection method we mean a construction of appropriate prior distribution $p(\boldsymbol{x})$ and information operator $\boldsymbol{I}(\boldsymbol{A}, \boldsymbol{b})$ in such a way that the posterior distribution $p\left(\boldsymbol{x}|\boldsymbol{I}(\boldsymbol{A}, \boldsymbol{b})\right)$ reproduces the result of a given projection method as it's mean, and reflects uncertainty about the true solution $\boldsymbol{A}^{-1}\boldsymbol{b}$ in a meaningful way (see Section 4.2.2 for a discussion on what constitutes a well-calibrated posterior distribution). The program of probabilistic reconstruction was successful to some extent. In particular, in [Coc+19a] and [Bar+19], the authors proved the following general result:

Theorem 4.1.1. *Let* $\det \boldsymbol{A} \neq 0$, $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_0, \boldsymbol{\Sigma}_0)$ *and* $\boldsymbol{y}_m = \boldsymbol{S}_m^T \boldsymbol{A}\boldsymbol{x}$, *where* $\boldsymbol{S}_m \in \mathbb{R}^{n\times m}, m \leq n$ *is a full-rank matrix. The mean of conditional distribution* $p(\boldsymbol{x}|\boldsymbol{y}_m = \boldsymbol{S}_m^T\boldsymbol{b}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_m, \boldsymbol{\Sigma}_m)$ *reproduces projection method* (1.5) *for three choices of prior distribution and search directions* $\boldsymbol{S}_m$*:*

1. $\boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T$ *and* $\boldsymbol{S}_m = \boldsymbol{W}$ *result in* $\boldsymbol{x}_m = \widetilde{\boldsymbol{x}}$, $\boldsymbol{\Sigma}_m = 0$*;*

2. *In case* $\boldsymbol{A}$ *is symmetric positive definite, the choice* $\boldsymbol{\Sigma}_0 = \boldsymbol{A}^{-1}$, $\boldsymbol{S}_m = \boldsymbol{V}$ *results in* $\boldsymbol{x}_m = \widetilde{\boldsymbol{x}}|_{\boldsymbol{W}=\boldsymbol{V}}$, $\boldsymbol{\Sigma}_m = \boldsymbol{A}^{-1} - \boldsymbol{V}\left(\boldsymbol{V}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{V}^T$;

3. $\boldsymbol{\Sigma}_0 = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}$, $\boldsymbol{S}_m = \boldsymbol{A}\boldsymbol{V}$ *result in* $\boldsymbol{x}_m = \widetilde{\boldsymbol{x}}|_{\boldsymbol{W}=\boldsymbol{A}\boldsymbol{V}}$, $\boldsymbol{\Sigma}_m = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1} - \boldsymbol{V}\left(\left(\boldsymbol{A}\boldsymbol{V}\right)^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{V}^T$.

We can see that $\boldsymbol{S}_m$ and $\boldsymbol{\Sigma}_0$ can be chosen in such a way that the mean of posterior distribution reproduces the wide class of projection methods. Although the correctness of mean vectors is reassuring, the main novelty of the probabilistic projection methods in the case of multivariate normal model is a covariance matrix that encodes the uncertainty about the solution. Unfortunately, all three posterior covariance matrices are not apt for different reasons. The first option leads to trivial uncertainty, while the other two are too expensive to compute because posterior covariances contains unknown $\boldsymbol{A}^{-1}$. Moreover, as shown in [Bar+19] and [Coc+19a], posterior distributions of the last two choices are poorly calibrated for Krylov subspace methods. Further examination of priors reveals that they do not have free parameters, which renders uncertainty calibration impossible.

In the present chapter we address these problems. Namely, we propose an extension of the covariance matrix $\boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T$ that maintains the same mean of conditional distribution, but introduces a nontrivial covariance $\boldsymbol{\Sigma}_m$. The main idea behind our construction stems from the observation made in [Bar+19], that the first prior distribution is a probability density of random variable $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{v}$, were $p(\boldsymbol{v}) = \mathcal{N}(\boldsymbol{v}|0, \boldsymbol{I})$. Perhaps it is not surprising that the posterior uncertainty is trivial, since the prior distribution puts no probability mass on the part of space where a projection method is not allowed to operate. Naturally, we seek a prior of the form $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{v} + \boldsymbol{Y}\boldsymbol{y}$, $p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}|0, \boldsymbol{I})$, and restrict $\boldsymbol{Y}$ to have meaningful mean and posterior covariance matrix.

In Section 4.2.1, we completely characterize all possible choices of $\boldsymbol{Y}$. Section 4.2.2 contains a discussion of uncertainty calibration for abstract projection methods. A practical inexpensive construction of covariance matrix in terms of projectors is presented in Section 4.2.3. In Section 4.3 we argue that realistic Krylov subspace methods elude rigorous probabilistic interpretation. Given the popularity of Krylov subspace methods, we explain how uncertainty can be calibrated for them in Section 4.3.1. In Section 4.3.2 we compare our approach with the related ones, including recently introduced in [Rei+20], [WH20], as well as [Coc+19a] and [Bar+19]. In Section 4.4 we perform a comparative study of different uncertainty calibration procedures on a several test problems that include a large family of small dense matrices, large and medium sparse matrices from SuiteSparse Matrix Col-

lection `https://sparse.tamu.edu`, a finite-difference discretization of biharmonic equation and a PDE-constrained optimization problem.

4.2 Fixing prior distribution

4.2.1 General form of prior distribution

In this section, we establish a sufficiently general form of $\boldsymbol{\Sigma}_0$ that leads to nontrivial uncertainty for probabilistic projection methods. We start by stating three lemmas and then gather all results in Theorem 4.2.1.

Lemma 4.2.1. *Let $\boldsymbol{V}$ and $\boldsymbol{W}$ lead to a well-defined projection method* (1.5)*, $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_0, \boldsymbol{\Sigma}_0)$, $\boldsymbol{y}_m = \boldsymbol{S}_m^T \boldsymbol{A}\boldsymbol{x}$, $p(\boldsymbol{x}|\boldsymbol{y}_m = \boldsymbol{S}_m^T\boldsymbol{b}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_m, \boldsymbol{\Sigma}_m)$. If we take covariance matrix $\boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{\Psi}$ and search directions $\boldsymbol{S}_m = \boldsymbol{W}$, where $\boldsymbol{\Psi}$ satisfies $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{\Psi} = 0$, $\boldsymbol{\Psi} \geq 0$, the resulting mean and covariance matrix are $\boldsymbol{x}_m = \widetilde{\boldsymbol{x}}$ from* (1.5) *and $\boldsymbol{\Sigma}_m = \boldsymbol{\Psi}$.*

As the following result shows, matrix $\boldsymbol{\Psi}$ exists under mild conditions.

Lemma 4.2.2. *For invertible $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and full-rank $\boldsymbol{W} \in \mathbb{R}^{n\times m}$, $m \leq n$, there exists a full-rank $\boldsymbol{Y} \in \mathbb{R}^{n\times k}$, $k \leq n - m$ for which $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{Y} = 0$. As such, we can take $\boldsymbol{\Psi} = \boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T$ for any conformable $\boldsymbol{G} > 0$.*

Next, we show that $\boldsymbol{\Psi}$ can be chosen to have $\boldsymbol{\Sigma}_0 > 0$, given $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}$ is invertible. To demonstrate that we need to prove that for a well-defined projection method it is always possible to supplement m vectors $\boldsymbol{V}_{\star i}$ with $n - m$ vectors $\boldsymbol{Y}_{n-m}$ to form a basis for $\mathbb{R}^n$. Indeed, if this is the case, $\boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T > 0$ since it is clearly positive semidefinite for any $\boldsymbol{G} > 0$, and there is no $\boldsymbol{x}$ such that $\boldsymbol{x}^T\boldsymbol{\Sigma}_0\boldsymbol{x} = 0$ because $\mathsf{Range}\,(\boldsymbol{V}) \cup \mathsf{Range}\,(\boldsymbol{Y}) = \mathbb{R}^n$.

Lemma 4.2.3. *If $\boldsymbol{V}$ and $\boldsymbol{W}$ lead to a well-defined projection method* (1.5)*, m linearly independent vectors $\boldsymbol{V}_{\star i}$ along with $n - m$ linearly independent $\boldsymbol{Y}_{\star i} : \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{Y} = 0$ form basis for $\mathbb{R}^n$.*

We summarize all results of this section in the following statement:

Theorem 4.2.1. *Let the following be true:*

1. *Matrix $\boldsymbol{A}$ is invertible, $\boldsymbol{W}, \boldsymbol{V} \in \mathbb{R}^{n\times m}$ are full-rank matrices, and* $\det \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V} \neq 0$*;*

2. *Solution of* $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ *is a normal random variable with probability density function* $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_0, \boldsymbol{\Sigma}_0)$*;*

3. *Covariance matrix* $\boldsymbol{\Sigma}_0$ *has a form* $\boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T$*, where* $\mathsf{Range}\,(\boldsymbol{Y}) = \mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$ *and* $\boldsymbol{G} \geq 0$*;*

4. *Random variable* $\boldsymbol{y} = \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x}$ *represents information available to a projection method.*

Then under these conditions $p(\boldsymbol{x}|\boldsymbol{y} = \boldsymbol{W}^T\boldsymbol{b}) = \mathcal{N}(\boldsymbol{x}|\widetilde{\boldsymbol{x}}, \boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T)$*, where* $\widetilde{\boldsymbol{x}}$ *is defined by* (1.5)*.*

The proposed covariance matrix has a clear geometric meaning. It is easy to see that $\boldsymbol{x}$ from Theorem 4.2.1 can be represented as a sum of two independent random variables $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{v} + \boldsymbol{Y}\boldsymbol{G}^{1/2}\boldsymbol{y}$, where $p(\boldsymbol{v}) = \mathcal{N}\left(\boldsymbol{v}|0, \boldsymbol{I}\right)$ and $p(\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{v}|0, \boldsymbol{I}\right)$. So, the part $\boldsymbol{V}\boldsymbol{V}^T$ corresponds to the vector that is sampled from $\mathsf{Range}(\boldsymbol{V})$, whereas the second part $\boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T$ accounts for the subspace $\mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A})$ in accordance with Petrov-Galerkin condition $\boldsymbol{W}^T(\boldsymbol{b} - \boldsymbol{A}\widetilde{\boldsymbol{x}}) = \boldsymbol{W}^T\boldsymbol{A}(\boldsymbol{A}^{-1}\boldsymbol{b} - \boldsymbol{x}) = 0$. Thanks to Lemma 4.2.3 we known that sampling $\boldsymbol{x}$ we can reproduce any vector from $\mathbb{R}^n$, so prior distribution is suitable for an arbitrary right-hand side. Adjusting $\boldsymbol{G} \geq 0$ we can control how $\boldsymbol{x}$ is distributed in $\mathsf{Range}\,(\boldsymbol{Y})$ (see Lemma 4.2.4 for a quantitative result). On the other hand it is not possible to control the distribution inside $\mathsf{Range}\,(\boldsymbol{V})$. This does not pose any problem, since as a result of projection process, the solution vector is completely defined within subspace $\mathsf{Range}\,(\boldsymbol{V})$.

4.2.2 Uncertainty calibration for abstract projection methods

To be useful in practical applications (for example, in probabilistic decision theory, sensitivity analysis and others) probability density function produced by probabilistic projection methods should be meaningfully related to the actual error. In [Coc+19a] authors propose a statistical criterion for uncertainty calibration: "When the UQ is well-calibrated, we could consider $\mathbf{x}^\star$ [the solution $\boldsymbol{A}^{-1}\boldsymbol{b}$] as plausibly being drawn from the posterior distribution $\mathcal{N}(\boldsymbol{x}_m, \boldsymbol{\Sigma}_m)$." Based on this statements authors suggest a test statistic $Z(\boldsymbol{x}^\star) \equiv \|\boldsymbol{x}^\star - \widetilde{\boldsymbol{x}}\|^2_{\boldsymbol{\Sigma}_m^\dagger} \sim \chi^2_{n-m}$. In what follows we refer to $Z(\boldsymbol{x}^\star)$ as $Z-$statistic. We now show that, according to this definition, the prior proposed in Theorem 4.2.1 provides a perfect uncertainty calibration.

Theorem 4.2.2. *Let* $\boldsymbol{x}^\star = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{v} + \boldsymbol{Y}\boldsymbol{G}^{1/2}\boldsymbol{y}$*, where* $\boldsymbol{v}$ *and* $\boldsymbol{y}$ *are independent random variables,* $\boldsymbol{v}$ *has arbitrary distribution and* $p(\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{y}|0, \boldsymbol{I}\right)$*. Under conditions of Theorem 4.2.1, a posterior distribution is well-calibrated:*

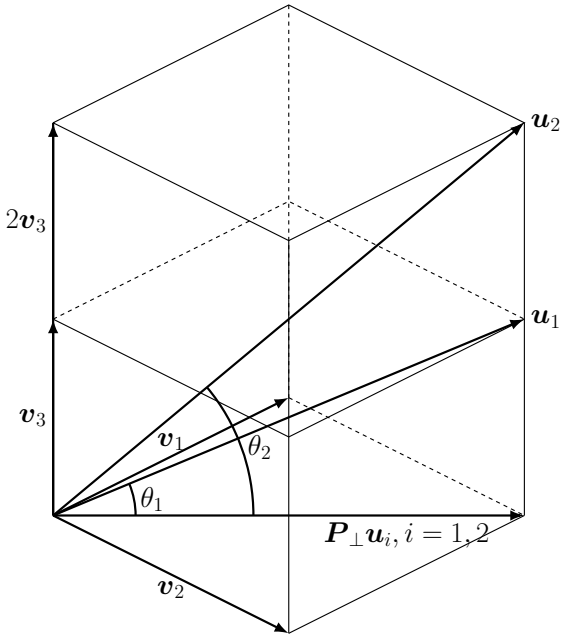


Figure 4.1: The figure demonstrates how the acute angle $\theta_i,\ i = 1, 2$ between subspace spanned by $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ and $\boldsymbol{u}_i,\ i = 1, 2$ depend on the vector $\boldsymbol{u}_i = \boldsymbol{v}_1 + \boldsymbol{v}_2 + i\boldsymbol{v}_3$. The angles can be computed as $\cos(\theta_i) = \boldsymbol{u}_i^T \boldsymbol{P}_\perp \boldsymbol{u}_i / \boldsymbol{u}_i^T \boldsymbol{u}_i$. Lemma 4.2.4 is a probabilistic counterpart of this situation. Namely, by rescaling eigenvectors of covariance matrix one can influence the distribution of the angle between the error and a given subspace.

1. $p\left(\boldsymbol{x} | \boldsymbol{y} = \boldsymbol{W}^T \left(\boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{A}\boldsymbol{V}\boldsymbol{v}_0\right)\right) = p(\boldsymbol{x}^\star | \boldsymbol{v} = \boldsymbol{v}_0)$
2. $\|\boldsymbol{x}^\star - \widetilde{\boldsymbol{x}}\|^2_{\left(\boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T\right)^\dagger} \sim \chi^2_{n-m}$

Note, that this result is also correct for all priors proposed in Theorem 4.1.1. This is because all methods are fully Bayesian when $\boldsymbol{W}$ and $\boldsymbol{V}$ do not depend on $\boldsymbol{x}$. As we discuss in Section 4.3, this is not true for Krylov subspace methods like CG and GMRES.

Having a well-calibrated posterior probability, we turn to the choice of a prior distribution. Since with $\boldsymbol{G}$, we can always perform a change of basis in $\mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$; we consider it to be fixed and describe how the rescaling of basis vectors influences an error vector.

Lemma 4.2.4. *Let in addition to conditions of Theorem 4.2.1 columns of matrix* $\boldsymbol{Y}$ *be orthonormal, and the exact solution be* $\boldsymbol{x}^\star = \boldsymbol{x}_0 + \boldsymbol{V}\delta_1 + \boldsymbol{Y}\boldsymbol{G}^{1/2}\boldsymbol{\delta}$*, where*

$\boldsymbol{\delta}_1, \boldsymbol{\delta}$ *are standard multivariate normal random variables. The choice* $\boldsymbol{G} = s^2 \boldsymbol{I}_{p\times p} \oplus \boldsymbol{I}_{(n-m-p)\times(n-m-p)}$, $s \in \mathbb{R}$ *leads to* $\cos(\theta) = 1/\left(1 + \frac{n-m-p}{s^2 p} z\right)$, *where* θ *is an acute angle between the error* $\widetilde{\boldsymbol{e}} = \boldsymbol{x}^\star - \widetilde{\boldsymbol{x}}$ *and* $\mathsf{span}\left\{\boldsymbol{Y}_{\star i} : i = 1, \ldots, p\right\}$; z *is* F*-distributed with numerator* $n-m-p$ *and denominator* p *(see Fig. 4.1 for geometric interpretation).*

With this result we can easily construct probabilistic bounds. For example, identity $P\left(\cos(\theta) \geq 1 - \epsilon\right) = P\left(z \leq s^2 p\epsilon / \left((1-\epsilon)(n-m-p)\right)\right)$ allows to choose s that guaranties $\widetilde{\boldsymbol{e}}$ to be located within a p-dimensional subspace with prescribed probability.

4.2.3 Construction of covariance matrices

So far, we discussed only a general form of a covariance matrix. The most straightforward way to construct it explicitly is to compute a basis for $\mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$ with SVD and choose positive semidefinite $\boldsymbol{G}$ according to some criteria. This can be problematic for two reasons. First, SVD incurs additional $O\left(nm^2\right)$ floating-point operations [TB97]. Depending on the situation, this can be manageable. The second and more serious problem is that we need to store a dense $n \times (n-m)$ matrix. Iterative methods are useful only when $\boldsymbol{A}$ is sparse and large, so as a rule, we do not have the luxury to store $(n-m)$ vectors forming a basis for $\mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$. The following result resolves these issues.

Theorem 4.2.3. *Let conditions of Theorem 4.2.1 be fulfilled. For* $\boldsymbol{P}_1 = \boldsymbol{I} - \boldsymbol{V}\left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T\boldsymbol{A}$ *the following statements are true:*

1. *Matrix* $\boldsymbol{P}_1$ *is a projection operator.*
2. $\mathsf{Range}(\boldsymbol{P}_1) = \mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$
3. *General form of covariance matrix from Lemma 4.2.2 is* $\boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{P}_1\boldsymbol{G}\boldsymbol{P}_1^T$, $\boldsymbol{G} \geq 0$.

We would like to point out that it is natural to use projector $\boldsymbol{P}_1$ to quantify uncertainty. It is known from general theory of iterative methods (see [Hac16, Chapter 2]) that linear iteration of the form $\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \boldsymbol{N}[\boldsymbol{A}]r^{(n)} \equiv \boldsymbol{M}[\boldsymbol{A}]\boldsymbol{x}^{(n)} + \boldsymbol{N}[\boldsymbol{A}]\boldsymbol{b}$, where $\boldsymbol{M}[\boldsymbol{A}]$ and $\boldsymbol{N}[\boldsymbol{A}]$ are matrices depending on $\boldsymbol{A}$ such that the consistency condition $\boldsymbol{M}[\boldsymbol{A}] + \boldsymbol{N}[\boldsymbol{A}]\boldsymbol{A} = \boldsymbol{I}$ holds. In our case $\boldsymbol{N}[\boldsymbol{A}] = \boldsymbol{V}\left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T$ approximates $\boldsymbol{A}^{-1}$, and $\boldsymbol{P}_1 = \boldsymbol{M}[\boldsymbol{A}] = \boldsymbol{I} - \boldsymbol{N}[\boldsymbol{A}]\boldsymbol{A}$ quantifies how well this is done.

To compute projection operator $\boldsymbol{P}_1$ from Theorem 4.2.3, one need not perform more complex operations that are required for projection method itself: matrices $\boldsymbol{W}$

and $\boldsymbol{V}$ are available as a byproduct of Arnoldi or Lanczos processes and $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}$ usually has a special form (Hessenberg or tridiagonal). Moreover, to store $\boldsymbol{P}_1$, we need to keep matrices $\boldsymbol{W}$, $\boldsymbol{V}$, and $\left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}$, that is $2nm + m^2$ floating-point numbers in the worst case, which is much better than $n^2 - mn$ in situations when $m \ll n$.

Covariance matrix in Theorem 4.2.3 contains projection operator $\boldsymbol{P}_1$ which is not orthogonal. Later we will see that orthogonal projectors are more suitable in the context of statistical inference, so we formulate a result similar to Theorem 4.2.3 but with an orthogonal projector.

Theorem 4.2.4. *Let $\boldsymbol{P}_2 = \boldsymbol{Y}\left(\boldsymbol{Y}^T\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^T$, where columns of $\boldsymbol{Y}$ are $k = n - m$ linearly independent vectors from $\mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$. If $\boldsymbol{W}$ and $\boldsymbol{V}$ result in a well-defined projection method, the following is true:*

1. *$\boldsymbol{P}_2$ is an orthogonal projector on $\mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$.*
2. *Covariance $\boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{P}_2\boldsymbol{G}\boldsymbol{P}_2^T$, $\boldsymbol{G} \geq 0$, leads to posterior $\mathcal{N}(\cdot|\widetilde{\boldsymbol{x}}, \boldsymbol{P}_2\boldsymbol{G}\boldsymbol{P}_2^T)$, under linear observations and conditions defined in Theorem 4.2.1.*

Note that to compute $\boldsymbol{P}_2$ one need no explicitly form the orthonormal basis for $\mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$, which is not feasible in typical practical situations when $n \gg 1$ and $m \ll n$. In place of that, one can use $\widetilde{\boldsymbol{Y}} \in \mathbb{R}^{n\times m}$ with columns such that $\mathsf{Range}\left(\widetilde{\boldsymbol{Y}}\right) = \mathsf{Range}\left(\boldsymbol{A}^T\boldsymbol{W}\right)$. Since $\mathsf{Range}\left(\boldsymbol{A}^T\boldsymbol{W}\right) \perp \mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$ we conclude that $\boldsymbol{P}_2 = \boldsymbol{I} - \widetilde{\boldsymbol{Y}}\left(\widetilde{\boldsymbol{Y}}^T\widetilde{\boldsymbol{Y}}\right)^{-1}\widetilde{\boldsymbol{Y}}^T$. Unlike $\boldsymbol{Y}$, computation of $\widetilde{\boldsymbol{Y}}$ is feasible. Moreover, for some projection method $\widetilde{\boldsymbol{Y}}$ can be available as a byproduct of the method itself. For example, vectors from $\mathsf{Range}\left(\boldsymbol{A}^T\boldsymbol{W}\right)$ are available in case of Lanczos biorthoganolization (see [Saa03, Subsection 7.2]). These vectors are discarded when only the solution of the linear system is of interest, however as we see from Theorem 4.2.4 they can be used to construct a covariance matrix. Conjugate gradient iteration provides the other example. In this case $\boldsymbol{A} > 0$ and $\boldsymbol{W} = \boldsymbol{V}$, so the residuals can be used to form orthonormal basis for $\mathsf{Range}\left(\boldsymbol{A}\boldsymbol{V}\right)$.

4.3 Difficulties with probabilistic projection methods

The validity of Theorem 4.1.1 and Bayesian conjugate gradient Method proposed in [Coc+19a], as well as all results of the present paper, depend on the assumption that

the joint distribution of x and $\boldsymbol{y}_m$ is a multivariate normal. This fact can be shown via computation of characteristic function if search directions $\boldsymbol{S}_m$ and prior covariance matrix $\boldsymbol{\Sigma}_0$ are independent of $\boldsymbol{x}$. When Krylov subspace $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{b})$ is used to build $\boldsymbol{S}_m$, as it is done in almost all Krylov subspace methods, information $\boldsymbol{y}_m$ becomes a nonlinear function of $\boldsymbol{x}$, and the joint distribution of $\boldsymbol{y}_m$ and $\boldsymbol{x}$ is not a multivariate normal. This implies that algorithms based on Theorem 4.1.1 and Bayesian conjugate gradient cannot stand as probabilistic Krylov subspace methods. Moreover, even when $\boldsymbol{S}_m$ is unrelated to $\boldsymbol{x}$, as in the Lanczos biorthogonalisation algorithm, $\boldsymbol{V}$, that still depends on $\boldsymbol{x}$, is not allowed to appear in prior covariance matrix $\boldsymbol{\Sigma}_0$. These restrictions render probabilistic Krylov projection methods incorrect. We can think of three possible solutions to this problem.

The first solution is to focus on projection methods that do not use $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{b})$ to construct approximate solution. For example, a two-grid operator in the Algebraic Multigrid (AMG) framework has the same form as a projection method (1.5), given $\boldsymbol{V}$ is a matrix of interpolation operator and $\boldsymbol{W}$ is a matrix of restriction operator. The same is true for Gauss-Seidel method, which is equivalent to the sequence of projection steps with $\mathcal{L} = \mathcal{K} = \mathsf{span}\{\boldsymbol{e}_i\}$ repeated for $i = 1, \ldots, n$ until convergence.

Another way is to use Arnoldi or Lancsoz processes to build basis in $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{\rho})$, where $\boldsymbol{\rho}$ is independent of $\boldsymbol{b}$. For this kind of projection processes, probabilistic methods are rigorously justified. On the downside, there are few theoretical results and estimations available from numerical linear algebra. One can also expect a deterioration of the convergence rate. In addition to that, memory-friendly algorithms like Conjugate Gradient should be rederived (if this is possible at all), because they explicitly rely on the fact that the first search direction is parallel to an initial residual vector.

Finally, it is possible to apply the results obtained under the assumption that $\boldsymbol{W}$ and $\boldsymbol{V}$ are independent of $\boldsymbol{x}$ to actual Krylov subspace methods and try to tune prior probability to get well-calibrated uncertainty. We consider this option in the next section.

4.3.1 Uncertainty calibration for Krylov subspace methods

For Krylov subspace methods, uncertainty is poorly calibrated. In the present section we put forward a statistical procedure that allows us to adjust a single scalar parameter in such a way, that $Z-$statistic as well as $S-$statistic (to be defined) are well-calibrated.

Before the main results we prove the following supplementary lemma.

Lemma 4.3.1. *Let $p(s|\alpha,\beta) = \mathsf{IG}(s|\alpha,\beta)$ be the inverse-gamma distribution, and $p(\boldsymbol{x}|s,\boldsymbol{\Sigma},\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, s\boldsymbol{\Sigma})$, $\boldsymbol{\Sigma} \geq 0$, then*

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{\Sigma},\boldsymbol{\mu},\alpha,\beta) &= \int d\boldsymbol{x}\ p(\boldsymbol{x}|s,\boldsymbol{\Sigma},\boldsymbol{\mu})p(s|\alpha,\beta) \\ &= \mathsf{St}_{2\alpha}\left(\boldsymbol{x}\,\middle|\,\boldsymbol{\mu}, \frac{\beta}{\alpha}\boldsymbol{\Sigma}\right). \end{aligned} \tag{4.1}$$

The first result is based on the rescaling of the full covariance matrix from Theorem 4.2.1 as proposed in [Coc+19a].

Lemma 4.3.2. *Let conditions of Theorem 4.2.1 be fulfilled. For covariance matrix $\boldsymbol{\Sigma}_0 = s\left(\boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{\Psi}\right)$, $s > 0$, $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{\Psi} = 0$, $\boldsymbol{\Psi} \geq 0$; and prior $p(s|\alpha,\beta) = \mathsf{IG}(s|\alpha,\beta)$ the following is true:*

1. *Probability density function $p\left(s|\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}\right)$ is the inverse-gamma distribution with $\widetilde{\alpha} = \alpha + m/2$, $\widetilde{\beta} = \beta + \boldsymbol{\delta}^T\boldsymbol{\delta}/2$, $\boldsymbol{\delta} = \left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0)$.*
2. *Predictive distribution for $\boldsymbol{x}|\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}$ is multivariate Student distribution $\mathsf{St}_{2\widetilde{\alpha}}\left(\boldsymbol{x}|\widetilde{\boldsymbol{x}}, \frac{\widetilde{\beta}}{\widetilde{\alpha}}\boldsymbol{\Psi}\right)$.*

Lemma 4.3.2 is straightforward from the point of view of the implementation, because approximate solution (1.5) is $\widetilde{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{\delta}$, where $\boldsymbol{\delta} = \left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0)$, scalar $\|\boldsymbol{\delta}\|_2^2$, required for uncertainty calibration, can be readily computed for arbitrary projection method. Common factor s appears in Lemma 4.3.2 because if we take $\boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T + s\boldsymbol{\Psi}$, posterior distribution for the scale $p\left(s|\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}\right)$ coincides with $\mathsf{IG}(s|\alpha,\beta)$, that is available information is insufficient to fix the scale. Since a scale of an error can be completely unrelated to the L_2 norm of projection of $\boldsymbol{A}^{-1}\boldsymbol{b}$ on $\boldsymbol{V}$, additional information can be valuable to tune s. This is explored in the following result.

Lemma 4.3.3. *Let conditions of Theorem 4.2.4 be fulfilled and $\boldsymbol{G} = s\boldsymbol{I}$ for $s > 0$, so $\boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T + s\boldsymbol{P}_2$, the solution is a multivariate normal variable $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_0,\boldsymbol{\Sigma}_0)$. For a prior distribution $p(s|\alpha,\beta) = \mathsf{IG}(s|\alpha,\beta)$ and i.i.d. observations $\boldsymbol{X}_{\star i}$, $i = 1,\ldots,k$ of random variable $\boldsymbol{P}_1(\boldsymbol{x} - \boldsymbol{x}_0)$ (here $\boldsymbol{P}_1$ is as in Theorem 4.2.3) the following is true:*

1. *Posterior distribution of $s|\boldsymbol{X}$ is $\mathsf{IG}(s|\widetilde{\alpha},\widetilde{\beta})$, where $\widetilde{\alpha} = \alpha + k(n-m)/2$, $\widetilde{\beta} = \beta + \mathsf{tr}\left(\boldsymbol{X}^T\boldsymbol{X}\right)/2$.*

2. Predictive distribution of $\boldsymbol{x}|\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}, \boldsymbol{X}$ *is* $\mathsf{St}_{2\widetilde{\alpha}}\left(\boldsymbol{x}\,\middle|\,\widetilde{\boldsymbol{x}}, \frac{\widetilde{\beta}}{\widetilde{\alpha}}\boldsymbol{P}_2\right)$.

The reason why we take $\boldsymbol{P}_1\boldsymbol{x}$ as an additional observation to fix the scale is that an exact solution has a representation $\boldsymbol{x}^\star = (\boldsymbol{I} - \boldsymbol{P}_1)\boldsymbol{x}^\star + \boldsymbol{P}_1\boldsymbol{x}^\star$. If $\boldsymbol{x}_0 = 0$ the first term $(\boldsymbol{I} - \boldsymbol{P}_1)\boldsymbol{x}^\star = \widetilde{\boldsymbol{x}}$, so $\boldsymbol{P}_1\boldsymbol{x}^\star$ is an error. To collect independent sample $\boldsymbol{x}_P$ we need to run the same projection method second time, starting from a sample $\boldsymbol{x}^\star$ from a prior distribution that we presume to be available. As a result, application of Lemma 4.3.3 doubles (for $k = 1$) numerical costs of any projection method. This is summarized in Algorithm 4.

Algorithm 4 Uncertainty calibration.

Input: distributions for exact solution $p(\boldsymbol{x}^\star)$, $\boldsymbol{x}^\star \in \mathbb{R}^n$; a projection method $\boldsymbol{V}, \boldsymbol{W} \leftarrow \mathsf{Proj}\,(\boldsymbol{A}, \boldsymbol{b}, m)$; a number of search directions m; parameters of inverse-gamma distribution α, β; a number of observations k; statistic either S or Z.

Output: modified parameters of inverse-gamma distribution $\widetilde{\alpha}, \widetilde{\beta}$.

$\widetilde{\alpha} = \alpha + k(n-m)/2$, $\widetilde{\beta} = \beta$
for $i = 1 : k$ **do**
  $\boldsymbol{x}^\star \sim p(\boldsymbol{x}^\star)$, $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}^\star$
  $\boldsymbol{V}, \boldsymbol{W} \leftarrow \mathsf{Proj}\,(\boldsymbol{A}, \boldsymbol{b}, m)$
  $\boldsymbol{x}^\star \leftarrow \left(\boldsymbol{I} - \boldsymbol{V}\left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T\boldsymbol{A}\right)\boldsymbol{x}^\star$
  if statistic $= Z$ **then**
    $\delta = \left(\boldsymbol{x}^\star\right)^T\left(\boldsymbol{x}^\star\right)$
  end if
  if statistic $= S$ **then**
    $\delta = \left(\boldsymbol{x}^\star\right)^T\boldsymbol{A}\left(\boldsymbol{x}^\star\right)$
  end if
  $\widetilde{\beta} = \widetilde{\beta} + \delta/2$
end for

Note, that prior from Lemma 4.3.3 leads to simple form of $Z-$statistic $Z(\boldsymbol{x}^\star) = \|\boldsymbol{x}^\star - \boldsymbol{x}_m\|_{\Sigma_m^\dagger}$, where $\boldsymbol{x}^\star = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{\delta}_1 + s^{1/2}\boldsymbol{P}_2\boldsymbol{\delta}_2$ is an exact solution, $\boldsymbol{\Sigma}_m^\dagger$ and $\boldsymbol{x}_m$ are posterior covariance matrix and posterior mean vector respectively and $\boldsymbol{\delta}_i$, $i = 1, 2$ are standard multivariate normal random variables. Indeed, because $\boldsymbol{P}_2$ is an orthogonal projector $\boldsymbol{P}_2^\dagger = \boldsymbol{P}_2$. Moreover, an error $\boldsymbol{x}^\star - \boldsymbol{x}_m$ belongs to $\mathsf{Range}(\boldsymbol{P}_2) = \mathsf{Range}(\boldsymbol{P}_1)$ which follows from the fact that $\boldsymbol{x}^\star - \boldsymbol{x}_m = \boldsymbol{P}_1(\boldsymbol{x}^\star - \boldsymbol{x}_0)$. So we can conclude that test statistic is simply a squared L_2 norm of the error $\|\boldsymbol{x}^\star - \boldsymbol{x}_m\|_2^2$. In

light of this observation, Algorithm 4 simply samples an error from a known $\boldsymbol{x}^\star$ and use its squared L_2 norm to estimate an error for a given right-hand side $\boldsymbol{b}$ for which the exact solution is unknown.

Both Lemma 4.3.2 and Lemma 4.3.3 are designed for test $Z-$statistic. Recently [Rei+20] propose a different test statistic $S(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{x}_m)^T \boldsymbol{A} (\boldsymbol{x} - \boldsymbol{x}_m)$, where $\boldsymbol{x}$ is drawn from a posterior distribution given linear observations as in Theorem 4.2.1. In what is following we call this random variable $S-$statistic. To calibrate the scale for $S-$statistic we use the following result.

Lemma 4.3.4. *Let $\boldsymbol{A} > 0$, $\boldsymbol{W} = \boldsymbol{V}$, columns of $\boldsymbol{Y}$ in Theorem 4.2.1 are $\boldsymbol{A}-$orthogonal, i.e., $\boldsymbol{Y}^T \boldsymbol{A} \boldsymbol{Y} = \boldsymbol{I}$ and $\boldsymbol{G} = s\boldsymbol{I}$, $s > 0$. Let $\boldsymbol{Z}_{\star i}, i = a, \ldots, k$ be a set of i.i.d. observations of random variable $\boldsymbol{A}^{1/2}\boldsymbol{P}_1(\boldsymbol{x} - \boldsymbol{x}_0)$ (here $\boldsymbol{P}_1$ is as in Theorem 4.2.3). For the prior distribution $p(s) = \mathsf{IG}(s|\alpha, \beta)$ under condition of Theorem 4.2.1 the following is true:*

1. *Posterior distribution of $s|\boldsymbol{Z}$ is $\mathsf{IG}\left(s|\widetilde{\alpha}, \widetilde{\beta}\right)$, $\widetilde{\alpha} = \alpha + k(n-m)/2$, $\widetilde{\beta} = \beta + \mathsf{tr}\left(\boldsymbol{Z}^T\boldsymbol{Z}\right)/2$.*
2. *Predictive distribution of $\boldsymbol{x}|\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}, \boldsymbol{Z}$ is $\mathsf{St}_{2\widetilde{\alpha}}\left(\boldsymbol{x}\,\middle|\,\widetilde{\boldsymbol{x}}, \frac{\widetilde{\beta}}{\widetilde{\alpha}}\boldsymbol{Y}\boldsymbol{Y}^T\right)$.*

The uncertainty calibration is summarized in Algorithm 4. As explained in the next result, the covariance matrix from Lemma 4.3.4 leads to simple $S-$statistic.

Lemma 4.3.5. *Under conditions of Lemma 4.3.4 distribution of $S(\boldsymbol{x}) = (\boldsymbol{x} - \widetilde{\boldsymbol{x}})\boldsymbol{A}(\boldsymbol{x} - \widetilde{\boldsymbol{x}})$ is the same as distribution of $s\chi^2_{n-m}$.*

Note, that $\boldsymbol{z}^T\boldsymbol{z}$ from Lemma 4.3.4 is an independent sample from $\boldsymbol{e}^T\boldsymbol{A}\boldsymbol{e}$, where $\boldsymbol{e}$ is a current error vector. So, if $\alpha = \beta = 0$, mean value of s is approximately $\sum_{i=1}^k \boldsymbol{e}_i^T\boldsymbol{A}\boldsymbol{e}_i/(k(n-m))$, so $S-$statistic takes a form

$$S(\boldsymbol{x}) \simeq \frac{1}{k}\left(\sum_{i=1}^k \boldsymbol{e}_i^T\boldsymbol{A}\boldsymbol{e}_i\right)\frac{\chi^2_{n-m}}{(n-m)}. \tag{4.2}$$

Since $\mathbb{E}\left[\chi^2_{n-m}\right] = n - m$ we can expect that $S-$statistic is well calibrated.

Comparison of uncertainty calibration provided by Lemma 4.3.2, Lemma 4.3.3 and Lemma 4.3.4 appears in Section 4.4.

Algorithm 5 Conjugate gradient.

Input: positive definite matrix $\boldsymbol{A}$, right-hand side $\boldsymbol{b}$, initial guess $\boldsymbol{x}$, number of sweeps m.
Output: approximate solution $\boldsymbol{x}$.

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
$\boldsymbol{v}_1 = \boldsymbol{r}_0$
for $i = 1 : m$ **do**
  $\eta_i = \boldsymbol{v}_i^T \boldsymbol{A} \boldsymbol{v}_i$
  $\gamma_i = \boldsymbol{r}_{i-1}^T \boldsymbol{r}_{i-1} / \eta_i$
  $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \gamma_i \boldsymbol{v}_i$
  $\boldsymbol{r}_i = \boldsymbol{r}_{i-1} - \gamma_i \boldsymbol{A} \boldsymbol{v}_i$
  $\delta_i = \boldsymbol{r}_i^T \boldsymbol{r}_i / \boldsymbol{r}_{i-1}^T \boldsymbol{r}_{i-1}$
  $\boldsymbol{v}_{i+1} = \boldsymbol{r}_i + \delta_i \boldsymbol{v}_i$
end for
$\boldsymbol{x} = \boldsymbol{x}_m$

4.3.2 Comparison with [Rei+20]

In recent contribution [Rei+20], authors explore related ideas to the construction of probabilistic projection methods. In this section we show that the covariance matrix introduced in [Rei+20, Definition 3.1] corresponds to a particular choice of $\boldsymbol{Y}$ and $\boldsymbol{G}$ in Theorem 4.2.1, we formulate a conjecture about optimality of the low-rank posterior in [Rei+20], and comment on uncertainty calibration adopted in [Rei+20].

Covariance matrix

In [Rei+20] authors propose to use the following covariance matrix:

$$\boldsymbol{\Sigma}_0 = \sum_{i=1}^{m+d} \left(\gamma_i \left\| \boldsymbol{r}_{i-1} \right\|_2^2 \right) \widetilde{\boldsymbol{v}}_i \left(\widetilde{\boldsymbol{v}}_i \right)^T, \ \widetilde{\boldsymbol{v}}_i = \boldsymbol{v}_i / \sqrt{\eta_i} \tag{4.3}$$

where η_i, $\boldsymbol{v}_i$, $\boldsymbol{r}_i$ are as in Algorithm 5, and $d \ll n - m$ is a small number of additional CG iterations used to calibrate uncertainty. For this covariance matrix they show that posterior covariance after projection on the first m search directions reads

$$\boldsymbol{\Sigma}_m = \sum_{i=m+1}^{m+d} \left(\gamma_i \left\| \boldsymbol{r}_{i-1} \right\|_2^2 \right) \widetilde{\boldsymbol{v}}_i \left(\widetilde{\boldsymbol{v}}_i \right)^T. \tag{4.4}$$

We are going to show that covariance matrix (4.3) is in line with Theorem 4.2.1. We start with the following supplementary result.

Lemma 4.3.6. *Mean vector $\widetilde{\boldsymbol{x}}$ in Theorem 4.2.1 does not depend on the choice of bases in subspaces* $\mathsf{Range}\,(\boldsymbol{V})$, $\mathsf{Range}\,(\boldsymbol{W})$.

Now, we show that the following result holds.

Theorem 4.3.1. *For $\boldsymbol{A} > 0$ let $\boldsymbol{Y}$, $\boldsymbol{G}$ and $\boldsymbol{V} = \boldsymbol{W}$ be chosen as follows. Columns of $\boldsymbol{Y} \in \mathbb{R}^{n\times(n-m)}$ are search directions $\widetilde{\boldsymbol{v}}_i = \boldsymbol{v}_i/\sqrt{\eta_i}$, $i = m+1,\ldots,n$, matrix $\boldsymbol{G} \in \mathbb{R}^{(n-m)\times(n-m)}$ is diagonal with elements $G_{ii} = \gamma_i \left\|r_{i-1}\right\|_2^2$, $i = m+1,\ldots,n$, where v_i and η, $\boldsymbol{r}_i$, γ_i are defined by Algorithm 5. Columns of matrix V form a basis for Krylov subspace $\mathcal{K}_m(\boldsymbol{A},\boldsymbol{r}_0) = \mathsf{Span}\left\{\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0,\ldots,\boldsymbol{A}^{m-1}\boldsymbol{r}_0\right\}$.*

Let the solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ be a normal random variable with probability density function $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_0, \boldsymbol{\Sigma}_0)$, where $\boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T$.

Under this condition the mean of posterior distribution $p(\boldsymbol{x}|\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b})$ coincides with projection method (1.5) *(and with the output of Algorithm 5 in exact arithmetic), and the covariance matrix is $\boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T = \sum_{i=m+1}^{n}\left(\gamma_i\left\|\boldsymbol{r}_{i-1}\right\|_2^2\right)\widetilde{\boldsymbol{v}}_i\left(\widetilde{\boldsymbol{v}}_i\right)^T$.*

From Theorem 4.3.1 we can conclude that the covariance from [Rei+20] can be considered as a special case of general result given in Theorem 4.2.1.

Before the comparison on uncertainty calibration we want to discuss a low-rank approximation (4.4) to a full-rank matrix $\boldsymbol{\Sigma}_m$ from Theorem 4.3.1. Is it the "best" low-rank approximation? We believe, that in some sense it is. To motivate this we start with a supplementary statement.

Lemma 4.3.7. *For $\boldsymbol{A} > 0$ we define the following operator norm $\|\boldsymbol{B}\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} \equiv \sup_{\boldsymbol{x}} \|\boldsymbol{B}\boldsymbol{x}\|_{\boldsymbol{A}} \,/\, \|\boldsymbol{x}\|_{\boldsymbol{A}^{-1}}$. If $\boldsymbol{B} = \sum_{j=1}^{K} d_j \boldsymbol{u}_j\boldsymbol{u}_j^T$ where $d_1 \geq d_2 \geq \cdots \geq d_K > 0, K \leq n$ and $\boldsymbol{u}_j^T\boldsymbol{A}\boldsymbol{u}_k = \delta_{jk}$, the operator norm of $\boldsymbol{B}$ is $\|\boldsymbol{B}\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} = d_1$.*

Next we extend a well-known optimal low-rank approximation result on norm $\|\cdot\|_{\boldsymbol{A},\boldsymbol{A}^{-1}}$.

Lemma 4.3.8. *Let B be the same as in Lemma 4.3.7, and $\boldsymbol{B}_m = \sum_{j=1}^{m} d_j\boldsymbol{u}_j\boldsymbol{u}_j^T$, $m < K$. Then*

$$\|\boldsymbol{B} - \boldsymbol{B}_m\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} = \inf_{\mathsf{rank}\boldsymbol{C}\leq m} \|\boldsymbol{B} - \boldsymbol{C}\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} = d_{m+1}.$$

Lemma 4.3.8 implies that approximation (4.4) is optimal (best $d-$rank approximation) in $\|\cdot\|_{\boldsymbol{A},\boldsymbol{A}^{-1}}$ norm if $\gamma_i\left\|\boldsymbol{r}_{i-1}\right\|_2^2, i = 1,\ldots,n$ form a non-increasing sequence. Unfortunately, this is not the case, because $\|\boldsymbol{r}_i\|$ can increase in the course of iterations.

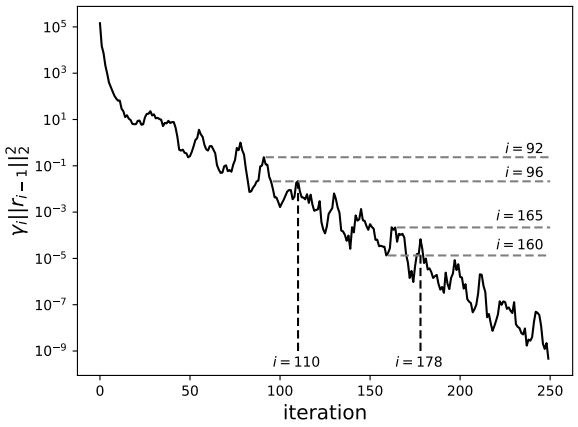


Figure 4.2: Figure demonstrates $\gamma_i \left\|\boldsymbol{r}_{i-1}\right\|_2^2$ for matrix bcsstm07 from SuiteSparse matrix collection.

However, because $\|\boldsymbol{r}_i\| \to 0$ in exact arithmetic, it seems, we still can obtain an optimal low rank approximation for an appropriate choice of d in (4.4). This is exemplified in Fig. 4.2. Evidently, if $m = 91$ for $d \leq 4$ we obtain optimal $d-$rank approximation to the whole covariance matrix $\boldsymbol{\Sigma}_m$ from Theorem 4.3.1. However if we take $4 < d < 8$ we achieve no improvement over $d = 4$ because the next peak $i = 110$ has larger $\gamma_i \left\|\boldsymbol{r}_{i-1}\right\|_2^2$. Less favourable situation occurs when $m = 159$. In this case all $d < 5$ does not result in optimal $d-$rank approximation, and $d = 5$ gives an optimal $1-$rank approximation. Based on these observations we formulate the following conjecture.

Conjecture 4.3.1. *For almost any positive definite matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, for any iteration $m \leq n$, there is a $d(m) \ll n$ and $r(d) \leq d(m)$ such that a covariance matrix $\boldsymbol{\Sigma}_m = \sum_{i=m+1}^{m+d(m)} \left(\gamma_i \left\|\boldsymbol{r}_{i-1}\right\|_2^2\right) \widetilde{\boldsymbol{v}}_i \left(\widetilde{\boldsymbol{v}}_i\right)^T$ is an optimal $r(d)-$rank approximation to the full covariance matrix $\widetilde{\boldsymbol{\Sigma}}_m = \sum_{i=m+1}^{n} \left(\gamma_i \left\|\boldsymbol{r}_{i-1}\right\|_2^2\right) \widetilde{\boldsymbol{v}}_i \left(\widetilde{\boldsymbol{v}}_i\right)^T$ with respect to the operator norm $\|\cdot\|_{\boldsymbol{A},\boldsymbol{A}^{-1}}$.*

Note, that the conjecture, if correct, merely ensures an optimality of approximation (4.4) to the full covariance matrix. Conjecture 4.3.1 does not tell whether the full matrix is optimal for uncertainty quantification in some (yet undefined) sense.

Comparison of uncertainty calibration

Unlike previous works [Bar+19], [Coc+19a] in article [Rei+20] authors focus on $\boldsymbol{A}$-norm of error. For this choice it is easy to construct an underestimate for an error

$\|\boldsymbol{x}^\star - \boldsymbol{x}_m\|_{\boldsymbol{A}}^2$ using information, available as a byproduct of Algorithm 5. Namely, this is done by the following expression [Rei+20, p. 4.1], [HS+52, Theorem 5:3]

$$\|\boldsymbol{x}^\star - \boldsymbol{x}_m\|_{\boldsymbol{A}}^2 - \|\boldsymbol{x}^\star - \boldsymbol{x}_{m+d}\|_{\boldsymbol{A}}^2 = \sum_{m+1}^{m+d} \gamma_i \|\boldsymbol{r}_{i-1}\|_2^2 , \tag{4.5}$$

from which we conclude that

$$\|\boldsymbol{x}^\star - \boldsymbol{x}_m\|_{\boldsymbol{A}}^2 \geq \sum_{m+1}^{m+d} \gamma_i \|\boldsymbol{r}_{i-1}\|_2^2 . \tag{4.6}$$

The advantage of a posterior covariance matrix defined by (4.4) is that to compute it one needs to perform a few additional iterations of conjugate gradient and store $\boldsymbol{A}$-orthogonal directions v_i and scales $\gamma_i \|\boldsymbol{r}_{i-1}\|_2^2$. So the estimation of $\boldsymbol{\Sigma}_m$ is cheap and justified by (4.6). However, in our opinion there are several disadvantages. First, even when $\|\boldsymbol{e}_i\|_{\boldsymbol{A}}$ is small $\|\boldsymbol{e}_i\|$ can remain large in the subspace corresponding to small eigenvalues of $\boldsymbol{A}$. Second, (4.6) provides only underestimate, which can be misleading in case of slow convergence (see Fig. 4.5 for an example of this behavior for biharmonic equation).

Our approach to uncertainty calibration is based on Lemma 4.3.4 and Algorithm 4 with $\mathsf{statistic} = S$. Algorithm 4 simply performs an additional run of a projection method (conjugate gradient in this case) for a known $\boldsymbol{x}^\star$, and records $\|\boldsymbol{x}^\star - \boldsymbol{x}_m\|_{\boldsymbol{A}}^2$. This norm is then used as an estimation for an error with a target right-hand side $\boldsymbol{b}$ for which $\boldsymbol{x}^\star$ is unknown. We will see that this approach leads to more reasonable S-statistic. The obvious disadvantage is a much higher cost of uncertainty calibration. However, our approach can be cheaper in case one needs to solve a set of linear equation with different right-hand sides and the same matrix $\boldsymbol{A}$ (Section 4.4 contain a relevant example).

4.4 Numerical examples

Julia [Bez+17] code that reproduces experiments in this section is available at `https://github.com/VLSF/BayesKrylov`.

4.4.1 Comparison with [Bar+19]

To assess the uncertainty calibration, we compare theoretical distributions for test statistics with empirical probability density functions averaged over many matrices.

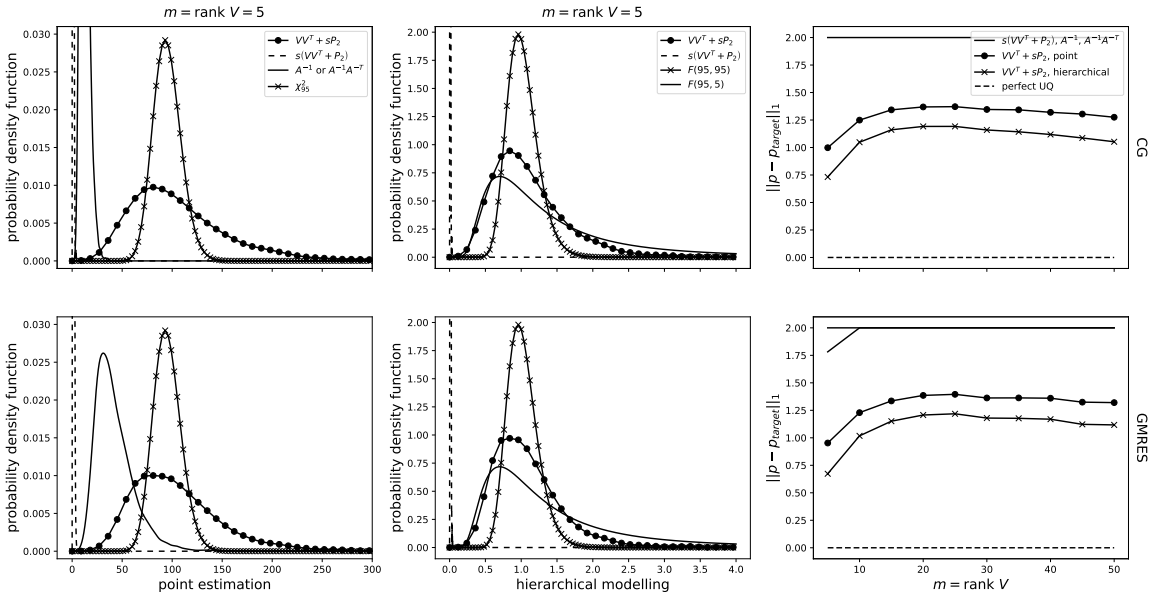


Figure 4.3: Figures demonstrate theoretical test statistics and empirical distributions for different prior distributions. Common legends for each column appear in the first row. The legend provides specifications of covariance matrices. For example, $s\left(\boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{P}_2\right)$ refers to posterior described in Lemma 4.3.2 with $\boldsymbol{\Psi} = \boldsymbol{P}_2$. The first two columns contain point estimation and hierarchical modelling for five projection steps. The first row presents results related to the conjugate gradient method and the second to GMRES. In the last column we show how L_1 norm of the difference between empirical p_e and target p_t (χ^2 or F as explained in Section 4.4) distributions changes with the number of projection steps. Perfect uncertainty calibration corresponds to zero value of discrepancy. The worst possible mismatch corresponds to L_1 norm of the error equals two. Overall we can see that the method proposed in Lemma 4.3.3 provides a reasonable uncertainty for both projection processes.

Note, that unlike $S-$statistic, $Z-$statistic for perfectly calibrated uncertainty does not depend on the matrix for both point estimation and hierarchical modelling. This makes averaging over $\boldsymbol{A}$ legitimate. Details of this procedure are summarized in Algorithm 6.

Algorithm 6 UQ assessment.

Input: Distributions for matrix $p(\boldsymbol{A})$; exact solution $p(\boldsymbol{x}^\star)$; number of search directions m; projection method $\boldsymbol{V}, \boldsymbol{W} \leftarrow \mathsf{Proj}\left(\boldsymbol{A}, \boldsymbol{b}, m\right)$,; number of samples N; statistics $p(z) \leftarrow \mathsf{Stat}(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N)$; number of matrices M.
Output: test statistic.

---

for $i = \overline{1, M}$ **do**
  $\boldsymbol{A}_i \sim p(\boldsymbol{A})$
  for $j = \overline{1, N}$ **do**
    $\boldsymbol{x}_j^\star \sim p(\boldsymbol{x}^\star)$
    $\boldsymbol{b}_{ij} = \boldsymbol{A}_i \boldsymbol{x}_j^\star$
    $\boldsymbol{V}_{ij}, \boldsymbol{W}_{ij} \leftarrow \mathsf{Proj}\left(\boldsymbol{A}_i, \boldsymbol{b}_{ij}, m\right)$
    $\widetilde{\boldsymbol{x}}_{ij} = \boldsymbol{V}_{ij} \left(\boldsymbol{W}_{ij}^T \boldsymbol{A}_i \boldsymbol{V}_{ij}\right)^{-1} \boldsymbol{W}_{ij}^T \boldsymbol{b}_{ij}$
    $\boldsymbol{e}_{ij} \leftarrow \widetilde{\boldsymbol{x}}_{ij} - \boldsymbol{x}_j^\star$
  end for
end for
$p(z) \leftarrow \mathsf{Stat}(\boldsymbol{e}_{11}, \ldots, \boldsymbol{e}_{NN})$

Details on components of Algorithm 6 are as follows:

$p(\boldsymbol{A})$: To draw symmetric positive definite matrices $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$ we sample stacked eigenvectors $\boldsymbol{U}$ from uniform distribution over $O(n)$, and eigenvalues from exponential distribution with scale $\widetilde{s}$.

$p(\boldsymbol{x}^\star)$: As a distribution of exact solution we take standard multivariate normal $\mathcal{N}\left(\cdot|0, \boldsymbol{I}\right)$ as in [Coc+19a].

Proj : Two projection processes are used. The first one with $\boldsymbol{W} = \boldsymbol{V} = \left(\widetilde{\boldsymbol{b}} \quad \boldsymbol{A}\widetilde{\boldsymbol{b}} \quad \cdots \quad \boldsymbol{A}^{m-1}\widetilde{\boldsymbol{b}}\right)$, $\widetilde{\boldsymbol{b}} = \boldsymbol{b} / \left\|b\right\|_2$ is equivalent to conjugate gradient in exact arithmetic. The second one with $\boldsymbol{W} = \boldsymbol{A}\boldsymbol{V}$, and $\boldsymbol{V} = \left(\widetilde{\boldsymbol{b}} \quad \boldsymbol{A}\widetilde{\boldsymbol{b}} \quad \cdots \quad \boldsymbol{A}^{m-1}\widetilde{\boldsymbol{b}}\right)$, $\widetilde{\boldsymbol{b}} = \boldsymbol{b} / \left\|\boldsymbol{b}\right\|_2$ is equivalent to GMRES under the same condition.

Stat : For distribution $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\mathsf{rank}(\boldsymbol{\Sigma}) = n - m$ test statistic is $z = (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^\dagger (\boldsymbol{x} - \boldsymbol{\mu}) \sim \chi^2_{n-m}$, and for multivariate Student distribution $\mathsf{St}_\nu(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, test statistic is $z = (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^\dagger (\boldsymbol{x} - \boldsymbol{\mu}) / (n - m) \sim F(n - m, \nu)$.

In all experiments, the size of the problem is $n = 100$, the scale is $\widetilde{s} = 10$, number of matrices $M = 500$, number of samples is $N = 20$. We also take $\boldsymbol{G} = \boldsymbol{I}$, $\alpha = \beta = 0$[1] in both Lemma 4.3.2 and Lemma 4.3.3, and use Algorithm 4 with $\mathsf{statistic} = Z$ and $k = 1$, i.e., a single additional sample, to calibrate uncertainty. Results of Lemma 4.3.2 and Lemma 4.3.3 are used in two regimes. The first one is point estimation. In this case parameters $\widetilde{\alpha}$, $\widetilde{\beta}$ of inverse-gamma distribution are used to find a mean value $\mathbb{E}[s] = \beta/(\alpha - 1)$, and this mean value is used as a scale in covariance matrix $s\boldsymbol{P}_2$. As a result, the statistic $Z(x)$ is compared with $\sim \chi^2_{n-m}$. The second one is a hierarchical modelling. In this case s is marginalized (as in second parts of Lemma 4.3.2 and Lemma 4.3.3) and the resulting statistic $Z(x)$ is compared with $F(n - m, 2\widetilde{\alpha})$. More precisely, according to Lemma 4.3.2 for prior with covariance matrix $s(\boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{P}_2)$ and no additional observations the target distribution is $F(n-m, 2\alpha+m) = F(n-m, m)$, whereas Lemma 4.3.3 implies that for prior with covariance matrix $\boldsymbol{V}\boldsymbol{V}^T + s\boldsymbol{P}_2$ and k additional observations (see Algorithm 4) we should use $F(n - m, 2\alpha + k(n - m)) = F(n - m, k(n - m))$ as a target distribution.

As a distance between distributions we choose standard L_1 norm $d(p_1, p_2) \equiv \int dx\, |p_1(x) - p_2(x)|$ approximated by central Riemann sum. Probability density is computed with RBF kernel density estimator.

The results are presented in Fig. 4.3 ($k = 1$ in Algorithm 4) and Fig. 4.4 ($k = 1, 5, 25$ in Algorithm 4). From data presented on Fig. 4.3 it follows, that covariance matrices $\boldsymbol{A}^{-1}$, $\boldsymbol{A}^{-1}\boldsymbol{A}^{-T}$ and $s(\boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{P}_2)$ (Lemma 4.3.2 with $\boldsymbol{\Psi} = \boldsymbol{P}_2$) fail to provide meaningful uncertainty calibration. The only reasonably tuned variant is given by covariance $\boldsymbol{V}\boldsymbol{V}^T + s\boldsymbol{P}_2$ (Lemma 4.3.3), where s is fixed with additional observation $\boldsymbol{P}\boldsymbol{x}$. We can also see that the hierarchical modelling is marginally better than the point estimation. Fig. 4.4 describes how uncertainty calibration depends on the number of observations k. We can see that when k increases, the calibration for point estimation slightly improves, whereas the increase in k leads to the degradation of uncertainty calibration for the hierarchical modelling. Nowhere the convergence to theoretical distribution is observed when k is increased. This pathological behaviour supports the discussion in Section 4.3, where we state that probabilistic projection methods in they current form are unsuitable for Krylov subspace methods.

[1]Note that the choice $\alpha = \beta = 0$ leads to the improper prior. In the present case the posterior distribution is always proper, so noninformative prior seems harmless. Moreover, s is a scale parameter so $p(s) \propto s^{-1}$ is a reasonable choice (see [Gel+13, Section 2.8]).

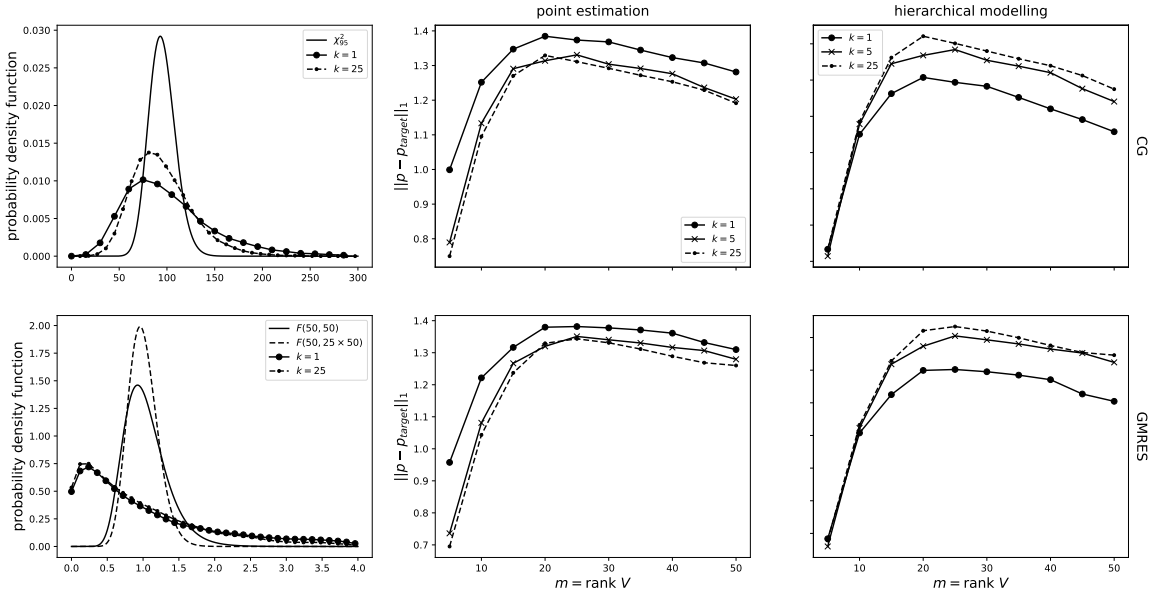


Figure 4.4: Figures summarize the dependence of proposed uncertainty calibration (Algorithm 4) on the number of additional observations k. First row corresponds to results for conjugate gradient iteration and the second row – for GMRES iteration. The second and the third columns, which contain point estimation and hierarchical modelling, respectively, share common legends that appeared in the first row. Graphs in these last two columns show how L_1 norm of the difference between empirical p_e and target p_t (χ^2 or F as explained in Section 4.4) distributions changes with the number of projection steps for $k = 1, 5, 25$ additional observations in Algorithm 4. Figures in the first column allow for visual inspection of empirical and target distributions for Z-statistic. Namely, for CG, we sketch the probability density function of Z-statistic for point estimation in the first row (the target distribution is χ^2), whereas the second row contains the same quantity but for hierarchical modelling (the target distribution is F). We can see that for point estimation, additional observations marginally improve uncertainty calibration, whereas, for hierarchical modelling, the situation is reversed. We conclude that, first, it makes little sense to use $k > 1$ for the chosen family of linear systems. Second, such behaviour clearly indicates that the chosen statistical model is inadequate for Krylov subspace methods.

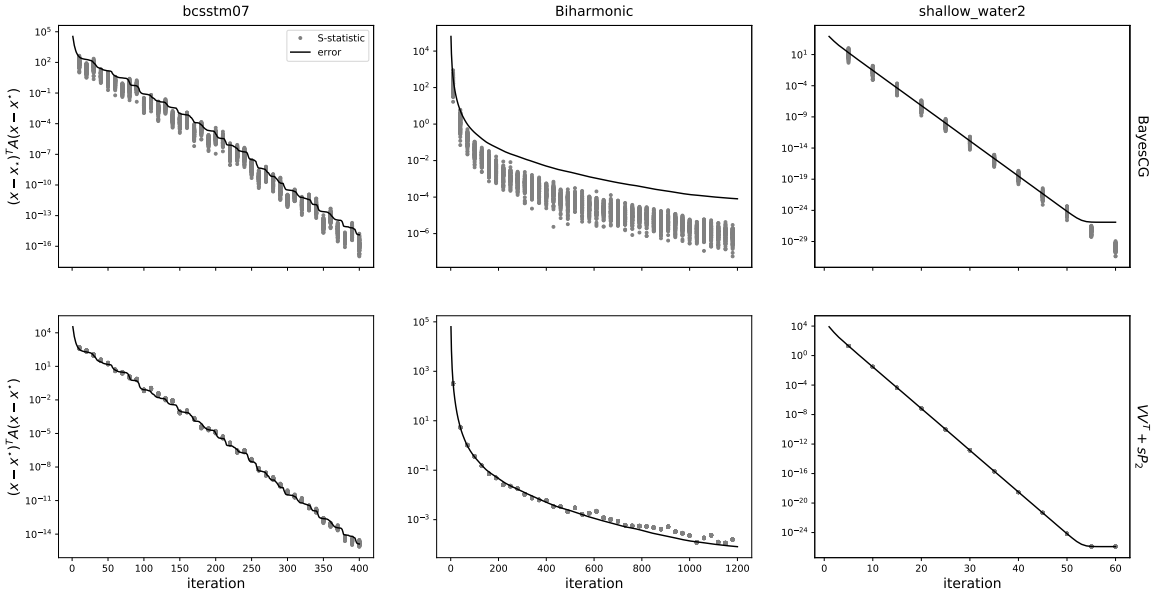


Figure 4.5: Figures demonstrate exact error $\boldsymbol{e}_m^T \boldsymbol{A} \boldsymbol{e}_m$ on iteration m, and samples from $S-$statistic for three matrices. First row corresponds to uncertainty calibration proposed in [Rei+20]. Second row shows samples from $S-$statistic calibrated according to Algorithm 4 with $\mathsf{statistic} = S$. We can see that the statistical uncertainty calibration proposed in this article leads to better uncertainty in all three cases.

4.4.2 Comparison with [Rei+20]

In this case, we use Algorithm 4 with $\mathsf{statistic} = S$ and $k = 1$. Note, that because for large m the effect of rounding errors is significant, we use conjugate gradient to compute projection operator P_1. If one computes P_1 as in Algorithm 4, it gives an underestimation of error for large m, because in this case methods based on projection method (1.5) converge much faster than the conjugate gradient as defined in Algorithm 5.

For a given matrix $\boldsymbol{A} > 0$ we compare uncertainty calibration as follows. For method described in [Rei+20] we sample $\boldsymbol{\delta}$ from $\mathcal{N}(\cdot|0, \boldsymbol{\Sigma}_m)$, where $\boldsymbol{\Sigma}_m$ is a posterior covariance matrix (4.4) and plot $l = 100$ samples from $S-$statistic $\boldsymbol{\delta}^T \boldsymbol{A} \boldsymbol{\delta}$ for m in regular intervals (each 10 or each 20 iterations). For our approach we use Algorithm 4 with $\mathsf{statistic} = S$ and $k = 1$, take $\mathbb{E}[s] = \widetilde{\beta}/(\widetilde{\alpha} - 1)$ and sample from $\mathbb{E}[s]\chi^2_{n-m}$, which is equivalent to $S-$statistic as explained in Lemma 4.3.5. Results for test problems can be found in Fig. 4.5. Overall, we can see that our approach leads to much

better uncertainty calibration in all cases. The price for it is much more expensive uncertainty calibration than the one adopted in [Rei+20]. Results for individual matrices are discussed below.

We use three positive definite matrices $\boldsymbol{A}$:

bcsstm07

The first example is a symmetric positive definite $n = 420$ matrix from SuiteSparse Matrix Collection: `https://sparse.tamu.edu/HB/bcsstm07`.

From the first column of Fig. 4.5 we can conclude that the method of [Rei+20] leads to underestimation for approximately an order of magnitude for each iteration. Our approach gives almost exact error estimation in this case.

Biharmonic equation

For the second test problem we take biharmonic equation (1.3.8). Results are in the second column of Fig. 4.5. The condition number is large and the convergence is extremely slow. As a result, uncertainty calibration from [Rei+20] is poor. For example at $m = 1200$ the exact error norm is about $\simeq 10^{-3}$, whereas an estimation is $\simeq 10^{-6}$. Our statistical uncertainty calibration results in a mild overestimation of the exact error, which is better than the uncertainty from [Rei+20].

shallow_water2

The third example is symmetric positive definite $n = 81920$ matrix from SuiteSparse matrix collection: `https://sparse.tamu.edu/MaxPlanck/shallow_water2`.

Last column of Fig. 4.5 provides a summary of results. The convergence is good and for all practical purposes both our approach and the method from [Rei+20] provide a reasonable estimation of error. The only difference is that our approach leads to smaller variance of the test statistic.

4.4.3 Uncertainty quantification for PDE-constraint optimization

As a last example we consider an optimal heating problem. Consider a diffusive heat transfer [PTA12, Section 5.1.3] from four point heat sources with unit heat fluxes in

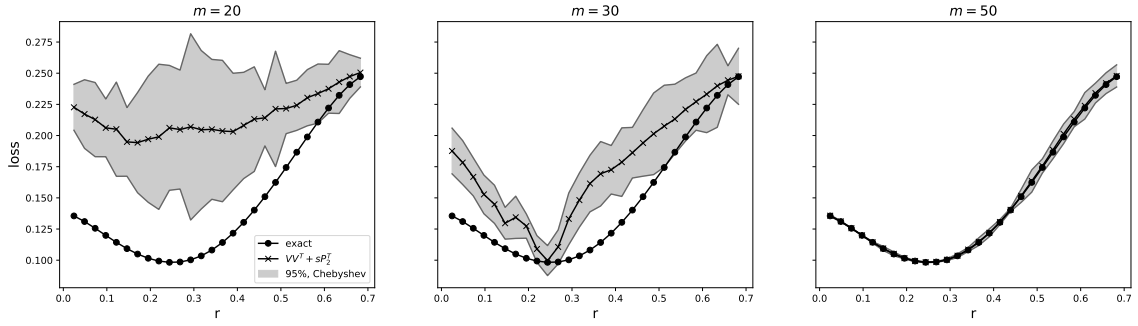


Figure 4.6: Figures demonstrate comparison of exact loss function (4.10) with an estimation obtained from probabilistic projection method from Theorem 4.2.4 with $\boldsymbol{W} = \boldsymbol{V} = \left(\widetilde{\boldsymbol{b}} \quad \boldsymbol{A}\widetilde{\boldsymbol{b}} \quad \cdots \quad \boldsymbol{A}^{m-1}\widetilde{\boldsymbol{b}}\right)$, $\widetilde{\boldsymbol{b}} = \boldsymbol{b}/\left\|\boldsymbol{b}\right\|_2$ for $m = 20,\ 30,\ 50$. Shaded region is enclosed by curves $\mu_m(r) \pm \sigma_m(r)$, where $\mu_m(r)$ is an approximate mean value of $\mathcal{L}(r)$ and $\sigma_m(r)^2$ is approximate variance, both estimated using 30 samples from the posterior distribution specified in Theorem 4.2.4.

simple geometry

$$\begin{aligned} &-\frac{\partial^2 T(x,y)}{\partial x^2} - \frac{\partial^2 T(x,y)}{\partial y^2} = \sum_{i=1}^{4} \delta(x - x_i)\delta(y - y_i), \\ &x, y \in \Gamma \equiv [0,1]^2,\, T(x,y)|_{\partial\Gamma} = 0, \end{aligned} \tag{4.7}$$

where x_i, y_i are located in vertices of the square:

$$\begin{aligned} x_1 &= r\cos\left(\pi/4\right),\ y_1 = r\sin\left(\pi/4\right); \\ x_2 &= -r\cos\left(\pi/4\right),\ y_2 = r\sin\left(\pi/4\right); \\ x_3 &= -r\cos\left(\pi/4\right),\ y_3 = -r\sin\left(\pi/4\right); \\ x_4 &= r\cos\left(\pi/4\right),\ y_4 = -r\sin\left(\pi/4\right). \end{aligned} \tag{4.8}$$

We consider the following PDE-constrained optimization problem

$$\min_r \int dxdy \left(T(x,y) - T_{\mathsf{target}}\right)^2 \text{ s.t. } T(x,y) \text{ solves (4.7)}. \tag{4.9}$$

Physically, the solution to the problem (4.9) is a distribution of sources that results in a smallest deviation of temperature field from the target temperature.

We use the finite element method (see Section 1.3.2) to discretise equation (4.7). As a result we obtain linear problem $\boldsymbol{A}\boldsymbol{t} = \boldsymbol{b}(r)$. A discrete counterpart of the

continuous PDE-constrained optimization problem (4.9) reads

$$\min_{r} \frac{1}{\left(2^L - 1\right)^2} \left\| \boldsymbol{t} - T_{\mathsf{target}} \right\|_2^2 \text{ s.t. } \boldsymbol{A}\boldsymbol{t} = \boldsymbol{b}(r). \tag{4.10}$$

To test the uncertainty calibration, we approximate a solution of linear system using probabilistic projection method with $\boldsymbol{W} = \boldsymbol{V} = \left(\widetilde{\boldsymbol{b}} \quad \boldsymbol{A}\widetilde{\boldsymbol{b}} \quad \cdots \quad \boldsymbol{A}^{m-1}\widetilde{\boldsymbol{b}}\right)$, $\widetilde{\boldsymbol{b}} = \boldsymbol{b} / \left\|\boldsymbol{b}\right\|_2$ and sample $\widetilde{\boldsymbol{t}}$ from the posterior distribution. This procedure turns loss function $\mathcal{L}(r)$ into a random variable.

The resulting uncertainty and the loss function are depicted in Fig. 4.6. We take $L = 6$, so the size of the matrix is $n = 3969$, $T_{\mathsf{target}} = 0.5$, and access three approximate solutions using $\mathsf{rank}(\boldsymbol{V}) \equiv m = 20,\ 30,\ 50$. In each case we retrieve 30 samples from $\mathcal{L}(r)$ and estimate mean $\mu_m(r)$ and variance $\sigma_m^2(r)$. The shaded region in Fig. 4.6 lies in-between curves $\mu_m(r) \pm 5\sigma_m(r)$. According to the Chebyshev inequality it contains a given sample from $\mathcal{L}(r)$ with probability 0.96. In addition to $\mu_m(r)$ and variance $\sigma_m^2(r)$, Fig. 4.6 contains an "exact" loss function obtained from (4.10), where linear system is solved with LU decomposition. Note, that since for all r the same linear system is solved, we perform the uncertainty calibration (using $\mathbb{E}[s]$ from Lemma 4.3.3) only once. So, the present example demonstrates that our uncertainty calibration can be cheaper than the one, proposed in [Rei+20].

From Fig. 4.6 we can see that the uncertainty calibration is not ideal. For example, in the case $m = 30$ the exact value of $\mathcal{L}(r)$ is confidently rejected for $r \leq 0.2$ and $0.3 \leq r < 0.5$, the same is true for $m = 20$ for $r \leq 0.5$. Despite this fact, we argue that the present uncertainty is useful. Observe, that for $m = 20$ the largest value $\sigma_{20}(r)$ resides in the region that corresponds to the smallest value of the exact loss. This fact can be exploited as follows. A natural way to perform a PDE-constrained optimization is to fit a surrogate model [PWG18, Section 5], using multifidelity Gaussian process (see [KO00] for a well-known example of a multifidelity model). The most widely used exploration rules (see [Sha+15, Section IV]) are directly related to the variance $\sigma(r)$, which contains σ_m. To exemplify, the well known principle coined "optimism in the face of uncertainty" (see [LS20, Section 7.1]) used in the construction of UCB exploration rules, prescribes to choose the next point according to $\arg\min\left(\mu_m(r) - \sigma_m(r)\right)$. As such, with the present uncertainty calibration Gaussian process favours a correct region for the further exploration.

Chapter 5

Hidden representation

5.1 Probability, uncertainty and numerical methods

Probability theory provides natural way to make qualitative statement about uncertain outcome of any experiment for which a complete specification of relevant conditions that can influence the outcome are not available. As it has been realized by some researches already in 1970, the experiment can be computational as well [OS19]. By computational experiment we mean arbitrary algorithm that provides an approximate answer to a given numerical task. So, if we treat numerical experiments as being uncertain, the goal is to provide probability distribution over outcomes, in place of a single approximation (known as point estimation in statistical literature). The role of probability distribution is to capture uncertainty about the approximation made. So, the goal is not unlike a usual error estimation that should accompany any reasonable algorithm (see [Ske86], [CF13] for the examples).

Numerical methods that output distribution in place of a point estimation were coined probabilistic numerical methods (PN) [HOG15]. The primary goal of the PN methods is not to speed up computations or approximate some object of interest, but rather to quantify epistemic uncertainty that comes from discretization or the early stopping of the algorithm [HOG15], [Coc+19b]. By now, PN algorithms are available for numerical integration [Bri+19], solution of ODEs [SDH14], numerical linear algebra [Bar+18], optimization [HK13], PDE solution [Owh15], and other problems (see [OS19; Coc+19b] for a more detailed account).

In recent work [Coc+19b] authors provided a formal definition of PN method. Among other things, this work discusses the update of the distribution, consistency

when different PN methods are combined and general existence results. The approach that is advocated in [Coc+19b] is to postulate random nature of the desired answer from the start. For example, one should formally treat number $\int_a^b dx f(x)$ as random even if $f(x)$ is fixed completely known (that is, one provides a way to compute $f(x)$ for all relevant x with any prescribed tolerance) deterministic function. As a justification for this approach authors [Coc+19b] rely on specific interpretation of probability as epistemic uncertainty.

In the present section we propose completely different approach that can turn the output of arbitrary black-box function into the distribution. Our approach is based on symmetry transformations and indifference argument. Namely, we find transformations of input data that preserve an exact answer to a given problem and simultaneously disturb the output of a given algorithm. Next we introduce probability distribution over these transformations using indifference argument. As a result we obtain probability distribution over output of arbitrary deterministic algorithm. The scheme is fully described in Section 5.2. Examples of distribution obtained this way appear in Section 5.2.1. In Section 5.3 we apply proposed approach to general linear iterative methods. And in Section 5.4 we construct am inexpensive variational approximation to a preconditioned Richardson iteration.

5.2 Uncertainty is in the representation

This section contains a general description of the procedure that allows us to introduce a subjective distribution over the input of numerical schemes, which in turn can transform the broad class of classical numerical methods into PN algorithms. As the first illustration of definitions to appear shortly, we will use systems of linear equations.

Let $\boldsymbol{A} \in \mathbb{R}^{K\times K}$, $\det \boldsymbol{A} \neq 0$, $\boldsymbol{b} \in \mathbb{R}^K$ and

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{5.1}$$

is a linear problem we want to solve. In principle, the solution is known

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}, \tag{5.2}$$

and can be made explicit via, for example, singular value decomposition. However, in some situations, one is forced to use an approximate solution, for instance, the generic projection method of the form

$$\begin{aligned} &\widetilde{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{S}\left(\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{S}\right)^{-1}\boldsymbol{R}^T\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0\right), \\ &\boldsymbol{R} \in \mathbb{R}^{K\times L},\ \boldsymbol{S} \in \mathbb{R}^{K\times L},\ \det\left(\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{S}\right) \neq 0. \end{aligned} \tag{5.3}$$

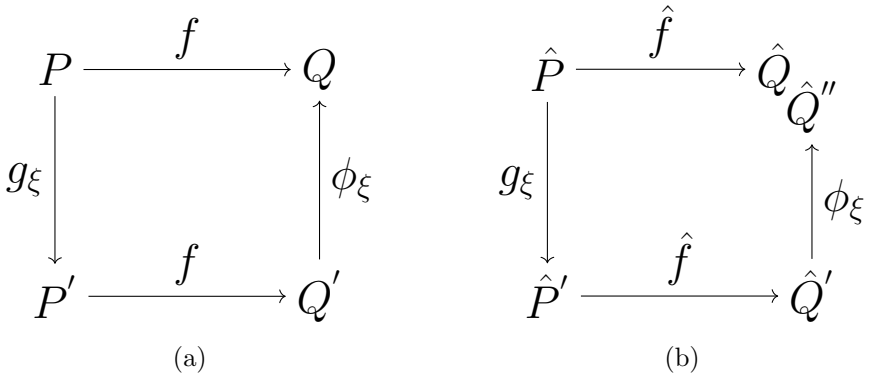


Figure 5.1: On (a) is a commutative diagram that represents an invariance of the exact problem with respect to the pair of transformations g_η, ϕ_η. Whereas (b) exemplifies the situation in which the approximation $\hat{f}$ (i.e., a particular numerical method) fails to be invariant under the same pair of transformations. See Section 5.2 for the detailed explanation.

Concrete expressions for $\boldsymbol{R}$ and $\boldsymbol{S}$ depend on the matrix $\boldsymbol{A}$ and the projection method. Usually, columns of $\boldsymbol{R}$ and $\boldsymbol{S}$ form a basis for some Krylov subspace. All projection methods also involve the inversion of the matrix, but since $M \ll N$, the complexity of the problem is lower.

Now, we are ready to introduce a few concepts that we need for the subjective distribution over the space of solutions.

1. Let P be all information needed to define a particular problem.

   For (5.1) one can see that $P = (\boldsymbol{A}, \boldsymbol{b})$.

2. The exact solution for a particular problem instance (or representation) P in explicit or implicit form is denoted by $f(P) = Q$.

   For the system of linear equations, the implicit form is given by (5.2). The explicit form would be, for example, the condensation-based Cramer's rule [HA12], or the Gaussian elimination with pivoting via LU decomposition [TB97, Lecture 20].

3. We also define a family of functions $g_\xi : P \to P'$, parameterized by ξ, that transform one problem instance to another within a chosen class of problems.

   For (5.1) we can consider transformation $g_{\boldsymbol{U},\boldsymbol{V}} : (\boldsymbol{A}, \boldsymbol{b}) \to \left(\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{U}, \boldsymbol{V}^{-1}\boldsymbol{b}\right)$, where $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{K\times K}, \det \boldsymbol{U} \neq 0, \det \boldsymbol{V} \neq 0$.

4. Now, we introduce another transformation $\phi_\xi : Q \to Q'$ over the solutions $f(P)$ that combined with g_ξ restores the solution to the initial problem, that is $f(P) = \phi_\xi(f(g_\xi(P)))$. For the transformation $g_{\boldsymbol{U},\boldsymbol{V}} : (\boldsymbol{A}, \boldsymbol{b}) \to \left(\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{U}, \boldsymbol{V}^{-1}\boldsymbol{b}\right)$ the complementary one is $\phi_{\boldsymbol{U},\boldsymbol{V}} : f(P) \to \boldsymbol{U} f(P)$.

Fig. 5.1a illustrates the commutative diagram that contains elements we have just described. In words, to apply our approach, one first needs to find a family of transformations that leave the problem invariant.

Next, we define a set of analogous concepts for the numerical algorithm employed to solve the problem.

- Let $\hat{P}$ be the information used by the algorithm.

  For the projection method (5.3), one has $\mathcal{Y} = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}_0)$.

- We let $\hat{f}\left(\hat{P}\right) = \hat{Q}$ to denote the numerical solution, i.e. the approximation to the solution of the problem: $\hat{P} \simeq P \Rightarrow \hat{f}\left(\hat{P}\right) \simeq f(P)$.

  In the context of the projection method, (5.3) is an example of an approximation to (5.2).

Note that it is often possible to find such a pair of transformations g_ξ, ϕ_ξ that the exact solution P remains invariant, whereas approximate one $\hat{P}$ is not. More precisely in previously defined terms, the condition reads

$$Q_\xi \equiv \phi_\xi(f(g_\xi(P))) = f(P) \text{ but } \hat{Q}_\xi \equiv \phi_\xi\left(\hat{f}\left(g_\xi\left(\hat{P}\right)\right)\right) \neq \hat{f}\left(\hat{P}\right). \tag{5.4}$$

The same idea is represented schematically on Fig. 5.1.

Since, by construction, g_ξ and ϕ_ξ change the representation of the problem P and do not change the solution $f(P)$ which is the only thing of interest at the end of the day, one can introduce a distribution over the set of transformations $p(\xi)$. Should (5.4) be satisfied, $p(\xi)$ in turn induces a pair of distributions

$$p(Q_\xi) = \delta(Q_\xi - f(P)) \text{ and } p\left(\hat{Q}_\xi\right) \neq \delta\left(\hat{Q}_\xi - \hat{f}\left(\hat{P}\right)\right), \tag{5.5}$$

the first of which is trivial, as one expects, given this is a completely deterministic quantity, and the second one is nontrivial by construction of symmetry transformations.

Regardless of the choice of a particular representation $\hat{P}$, the same algorithm $\hat{f}$ is applied to solve the problem. Since information about symmetry transformations

g_ξ and ϕ_ξ is not accessible to algorithm $\hat{f}$, we propose to call the presented scheme *the method of hidden representation* and use the term *hidden representation* for the corresponding prior distribution.

To completely specify the hidden representation, distribution $p(\xi)$ and the choice of functions g_ξ, ϕ_ξ if there are many, should be chosen based on an application at hand. At the present stage, we do not have a general recipe about how to achieve a reasonable uncertainty quantification for an arbitrary problem. However, in Section 5.3, Section 5.4 we construct a hidden representation and calibrate uncertainty for iterations that solve linear problem (5.1). Section 5.2.1 also contains several examples of g_ξ, ϕ_ξ for a number of well-known numerical methods.

Also note that by construction, if it is possible to describe ξ as a measurable space, the consistency and convergence for a method $\hat{f}$ ensures the concentration of the probability measure $p\left(Q_\xi\right)$. As such, the method is asymptotically consistent for any well-defined distribution.

5.2.1 Transformation-based examples

In this section, we demonstrate that it is relatively easy to find suitable symmetry transformations for a large class of numerical problems. Empirically computed distributions and experiments that demonstrate the concentration of measure for the algorithms in this section can be found in https://github.com/VLSF/HiddenRepresentation. The same repository also contains examples of the consequences of hidden representation for conjugate gradient and "not-a-knot" cubic splines which are not included here.

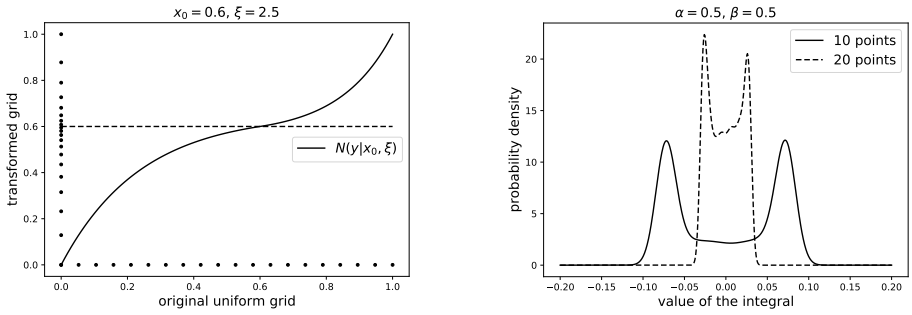
Trapezoidal rule

A definite integral of the form

$$\mathcal{Q}\left(a, b, f(x)\right) \equiv \int\limits_a^b f(x) dx, \tag{5.6}$$

can be approximated numerically by, for instance, trapezoidal rule [Tai94]

$$Q\left(a, b, f(x), N\right) = \frac{b-a}{N}\left(\sum_{i=1}^{N-1} f\left(a + (b-a)\frac{i}{N}\right) + \frac{f(a)+f(b)}{2}\right). \tag{5.7}$$



(a) (b)

Figure 5.2: (a) – two-parameter grid transformation defined by (5.10); (b) – the resulting distribution (kernel estimation of the probability density function) of the value of the integral.

The original problem is invariant under the invertible change of variables $x = g_\xi(y)$ in a sense that

$$\int_a^b f(x)dx = \int_{g_\xi^{-1}(a)}^{g_\xi^{-1}(b)} f\left(g_\xi\left(y\right)\right)\left|\partial_y g_\xi(y)\right| dy. \tag{5.8}$$

However, it is quiet obvious that, in general, approximation (5.7) is not invariant under the same transformation

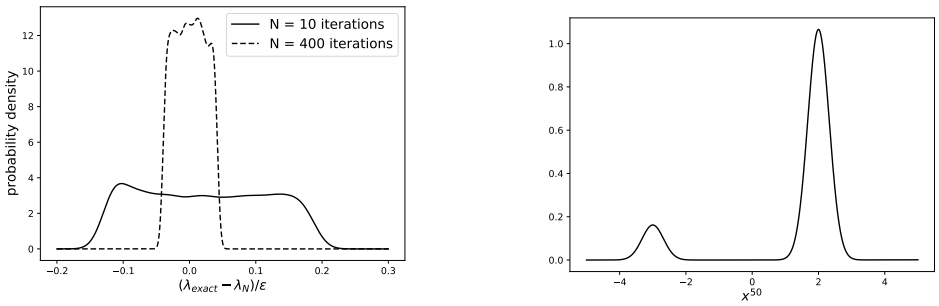
$$Q\left(a, b, f(x), N\right) \neq Q\left(g_\xi^{-1}(a), g_\xi^{-1}(b), \left|\partial_x g_\xi(x)\right| f\left(g_\xi(x)\right), N\right). \tag{5.9}$$

It means that the distribution over the parameter ξ will induce distribution over (5.7).

As an example of parametric transformation, we consider the popular grid stretching method [Lis17, ch. 4]. Without loss of generality, we can assume that the function is defined on the interval $[0, 1]$. The following two-parameter grid transformation concentrates nodes near the point x_0

$$N\left(y|x_0, \xi\right) = \begin{cases} x_0\left(1 - E\left(1 - \frac{y}{x_0}\middle|\, \xi\right)\right), \text{if } y \in [0, x_0]; \\ x_0 + (1 - x_0)\, E\left(\frac{y - x_0}{1 - x_0}\middle|\, \xi\right), \text{if } y \in [x_0, 1], \end{cases} \qquad E\left(y|\xi\right) = \frac{e^{\xi y} - 1}{e^\xi - 1}, \tag{5.10}$$

where ξ controls the degree of grid concentration. The example of the transformed grid appears in Fig. 5.2a.



(a) (b)

Figure 5.3: (a) – probability density function for the difference between the exact leading eigenvalue and the approximated one rescaled by ϵ (see (5.16) and (5.17) for the problem and similarity transformation respectively), (b) – probability density function for x^{50} from (5.18) applied to (5.19).

As an integrand, we take the function

$$f(x) = \left(x - \frac{1}{2}\right)\sqrt{\frac{20}{\pi}}\exp\left(-40\left(x - \frac{1}{2}\right)^2\right), \tag{5.11}$$

and, keeping $\xi = 10$, draw x_0 from a Beta distribution

$$p(x_0|\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}. \tag{5.12}$$

As we discussed earlier, the result of numerical integration is not invariant under the transformation (5.10). As a consequence, the probability density of the value of the integral under the prior distribution (5.12) is nontrivial. The kernel density estimation is shown in Fig. 5.2b. Since Beta distribution is broad for chosen values of hyperparameters, the resulting grid either resolves the region where the function is negative or the region where the function is positive, hence there are two distinct peaks. The probability density function for a trapezoidal rule with 20 points also shows that peaks become less prominent if one refines the grid. One can expect the distribution to converge to delta function as we further increase the number of points.

Leading eigenvalue by power iteration

The power iteration is an iterative method that allows approximating a leading, i.e., of the largest absolute value, eigenvalue and correspondent eigenvector. For conver-

gence conditions and the rate of convergence, see, for instance, [Saa20, Algorithm 4.1]. The iteration scheme starts from the initially chosen vector $\boldsymbol{v}_0$ and successively computes updates using the following two formulae

$$\boldsymbol{v}_n = \boldsymbol{A}\boldsymbol{v}_{n-1} / \left\|\boldsymbol{A}\boldsymbol{v}_{n-1}\right\|_\infty , \tag{5.13}$$

$$\lambda_n = \frac{(\boldsymbol{v}_n, \boldsymbol{A}\boldsymbol{v}_n)}{(\boldsymbol{v}_n, \boldsymbol{v}_n)}. \tag{5.14}$$

We will use power iteration to find only the leading eigenvalue. Let $\lambda_{N_{\text{it}}}(\boldsymbol{A}, \boldsymbol{v}_0)$ correspond to the N_{it} steps of power iteration (5.14) starting from $\boldsymbol{v}_0$. The spectrum of matrix $\boldsymbol{A}$ remains the same under the similarity transformation

$$\left(\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V}\right)\left(\boldsymbol{V}^{-1}\boldsymbol{x}\right) = \lambda\left(\boldsymbol{V}^{-1}\boldsymbol{x}\right). \tag{5.15}$$

However, it is easy to see that the scalar product $(\boldsymbol{x}, \boldsymbol{x})$ is not invariant under the change of basis with $\boldsymbol{V}^{-1}$ unless $\boldsymbol{V}$ is an orthogonal matrix. Hence, if one chooses the arbitrary distribution over invertible matrices (at least some of which are not orthogonal) $p_v(\boldsymbol{V})$, it induces the probability distribution over the leading eigenvalues computed with (5.14).

As an example, consider a toy problem

$$\boldsymbol{A} = \boldsymbol{v}\boldsymbol{v}^T + (1+\epsilon)\boldsymbol{u}\boldsymbol{u}^T, \boldsymbol{v} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}, \boldsymbol{u} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}. \tag{5.16}$$

We apply transformed (5.15) power iteration (5.14) to the problem (5.16), using

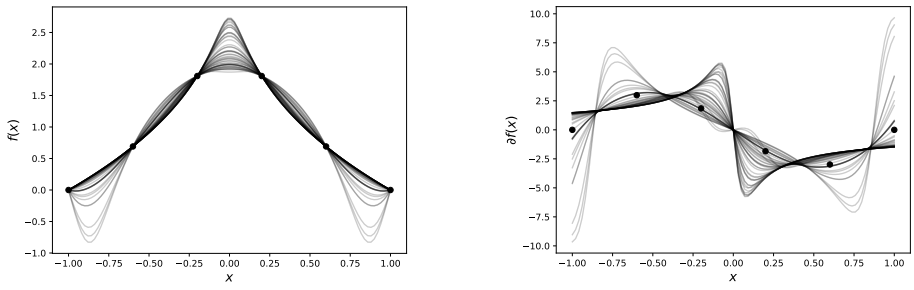
$$\boldsymbol{V} = \begin{pmatrix}a & 0\\0 & 1\end{pmatrix}\begin{pmatrix}\cos\phi & -\sin\phi\\\sin\phi & \cos\phi\end{pmatrix}, \; p(\phi) = \text{Ind}\left[\phi \in \left[0, \frac{\pi}{2}\right]\right]. \tag{5.17}$$

So the transformation performs rotation on a random angle and the rescaling along the new first axis. The result for $\epsilon = 10^{-2}$, $a = 1/2$ appears in Fig. 5.3a. One can see the measure concentration as the number of iteration increases.

Newton's method

Newton's method is a widely known technique for the solution of nonlinear algebraic equations of the form $g(x) = 0$. For a scalar equation, the single iteration reads

$$x^{n+1} = x^n - \frac{g(x^n)}{g'(x^n)}. \tag{5.18}$$



(a) (b)

Figure 5.4: Figures provide an example of the hidden representation applied to the interpolation problem. In (a) one can see curves that result from (5.22) applied to (5.24), and in (b) a result for the derivative of the Lagrange polynomial, i.e., the application of (5.23) to the same problem (5.24). Solid dots correspond to the true values of the function and derivative.

It is always possible to transform the initial equation using any function $f_\xi(x)$ (one-parameter family with different functions corresponding to different ξ) that is defined over the range of g, $f_\xi(x) = 0 \Leftrightarrow x = 0$ and $f^{'}(x) \neq 0 \, \forall x$. Under this condition any root of the original equation $g(x) = 0$ is also a root of the modified equation $f_\xi(g(x)) = 0$. So it is clear that any distribution over ξ does not affect the exact solution.

Nevertheless, it does affect the solution after a finite number of iterations. As an example, we consider

$$g(x) = (x-2)(x+3), \;\; f_\xi(y) = \frac{e^{\xi y} - 1}{e^{\xi} - 1}, \;\; p(\xi) = N(0, 0.1), \tag{5.19}$$

where $\mathcal{N}(0, 0.1)$ stands for the Normal distribution with zero mean and $\sigma = 0.1$. Fig. 5.3b shows the distribution over x^{50}.

Interpolation and differentiation

Suppose one is given a value of some function in K distinct points

$$\{f(x_i)\}_{i=\overline{1,K}} \tag{5.20}$$

and is asked to restore missing values of the function in-between. One solution of this simple interpolation problem is to use the Lagrange polynomial

$$\hat{f}(x) = \sum_{i=1}^{K} f(x_i) p_i(x),\ p_i(x) = \prod_{k \neq i} \frac{x - x_k}{x_i - x_k}. \tag{5.21}$$

Now, suppose we knew the values of a function at all points of the interval of interest, i.e. $f(x)$. In this case, the trivial transformation with arbitrary invertible function $g_\xi^{-1}\left(g_\xi\left(f(x)\right)\right)$ changes nothing. In particular, the original data (5.20) remain intact. On the other hand, this transformation leads to the modification of (5.21) that reads

$$\hat{f}_\xi(x) \equiv g_\xi^{-1}\left(\sum_{i=1}^{K} g_\xi\left(f\left(x_i\right)\right) p_i(x)\right) \neq \hat{f}(x). \tag{5.22}$$

Again, the distribution over ξ results in the distribution over $\hat{f}_\xi(x)$. Since the problem of numerical differentiation is intertwined with the problem of interpolation, the ordinary chain rule

$$\frac{d}{dx} f(x) = \frac{d}{dx} g_\xi\left(f(x)\right) \Big/ \left.\frac{d}{dy} g_\xi\left(y\right)\right|_{y=f(x)}, \tag{5.23}$$

allows one to compute a derivative of $f(x)$ from the interpolation of the function $g_\xi\left(f(x_i)\right)$. Figs. 5.4a and 5.4b show samples from the resulting distribution over (5.22) with

$$f(x) = 1 + \cos\left(\pi x\right),\ g_\xi(y) = \exp\left(-\xi y\right),\ p(\xi) = \mathrm{Ind}\left[\xi \in [-1, 3]\right], \tag{5.24}$$

for the Lagrange interpolation through 10 known values of the function on the uniform grid on $[-1, 1]$.

5.2.2 Examples based on the hidden subgrid dynamics

Here we consider simplified approach, where the only one transformation is required. Code for these examples can be found in the same repository.

Polynomial approximation

Suppose we are presented with a function $f(x)$ on the interval $[x, x + \Delta x]$. One of the simplest way to approximate the function is to use first-order polynomial approximation

$$[f]_{\mathsf{approximate}}(\xi) = \frac{1}{\Delta x} f(x)(x + \Delta x - \xi) + \frac{1}{\Delta x} f(x + \Delta x)(\xi - x). \tag{5.25}$$

Now we can alter the continuous approximation problem by adding and subtracting known function $g(x)$. This gives us expression $[f-g]_{\mathsf{approximate}}(\xi)+g(x)$. In is clear that if approximation step is exact, function $g(x)$ does not matter, so the transformation we described can be considered as a symmetry of continuous problem. Now, the trick is to choose $g(x)$ in such a way, that $[f-g]_{\mathsf{approximate}}(\xi)=[f]_{\mathsf{approximate}}(\xi)$. This can be easily done based on classical error analysis. For example, we can take $g(x)=\epsilon(x+\Delta x-\xi)(\xi-x)f^{''}(x)/2$ (see, for example, [Hol16, Theorem 5.2]), which is the reminder for the Lagrange interpolation multiplied by parameter ϵ. With this choice of $g(x)$ we obtain the following approximation

$$\begin{aligned}[f]_{\mathsf{approximate}}(\xi)=\frac{1}{\Delta x}f(x)(x+\Delta x-\xi)+\frac{1}{\Delta x}f(x+\Delta x)(\xi-x)\\ -\epsilon(x+\Delta x-\xi)(\xi-x)f^{''}(x)/2.\end{aligned} \tag{5.26}$$

Since function $g(x)$ is identically zero on the grid with spacing Δx, we call this construction "hidden subgrid dynamics". Now, if one takes ϵ to be random variable, Eq. (5.26) becomes random function which approximate sufficiently smooth function $f(x)$ arbitrary well when $\Delta x\to 0$.

In Fig. 5.5 we provide the example of polynomial approximation for the function $f(x)=\exp(\sin(3\pi x))+\exp(-\cos(\pi x))$. For this example we take $\epsilon\simeq\mathcal{N}(0,\sigma^2),\sigma^2=0.2$, and as $f''(x)$ we use the exact derivative (example with approximate derivative appears in Section 5.2.2). As we can see, the uncertainty provided by the method is quite reasonable. The main reason for that is the connection with classical error estimation. It is interesting to note that random function in Fig. 5.5 are not smooth. This problem can be fixed if one resorts to a higher order method or Hermite interpolation.

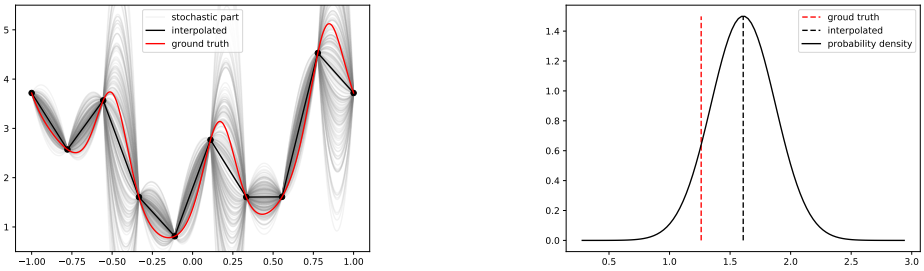
Differentiation and integration

Once the polynomial approximation is known, it can be used to define integration and differentiation. The derivative of 5.26 reads

$$\left[f^{'}\right]_{\mathsf{approximate}}(\xi)=\frac{1}{\Delta x}\left(f(x+\Delta x)-f(x)\right)-\epsilon\frac{\Delta x}{2}f^{''}(x), \tag{5.27}$$

and the integration yields

$$\int_x^{x+\Delta x}d\xi\,[f]_{\mathsf{approximate}}(\xi)=\frac{\Delta x}{2}\left(f(x)+f(x+\Delta x)\right)-\epsilon\frac{\Delta x^3}{12}f^{''}(x). \tag{5.28}$$



(a) (b)

Figure 5.5: Interpolation Eq. (5.26) applied to the function $f(x) = \exp(\sin(3\pi x)) + \exp(-\cos(\pi x))$. In (a) one can see the random functions, piecewise linear interpolation and the original function, computed on the fine mesh. In (b) the distribution for $[f]_{\mathsf{approximate}}(4/9)$ is given as well as the exact value of the function.

Example of the resulting distributions can be found in Fig. 5.6. Two comments are in order. First, again, the uncertainty is reasonable. Second, the distribution for approximate $\int_{-1}^{1} dx\, f(x)$ is clearly well approximated by a normal density. This can be expected from central limit theorem.

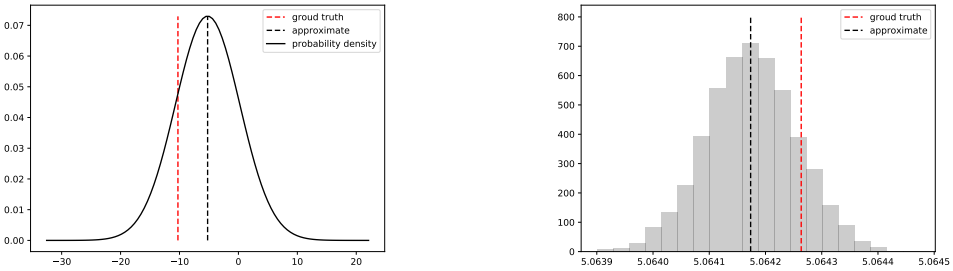
Solution of the initial value problem

Again, polynomial interpolation allows us to derive a scheme for the solution of the boundary value problem:

$$\frac{1}{\Delta t}\left(f(t+\Delta t) - f(t)\right) = u(f(t),t) + \epsilon\frac{\Delta t}{2}f''(t). \tag{5.29}$$

Note that even when ϵ is a random variable, Eq. (5.29) is not stochastic differential equation. Even if ϵ has stable distribution (normal, for example), the power of Δt is wrong. For Eq. (5.29) to be stochastic, one need to have equation like $df = \Delta t^{1/2}\epsilon$ [Ste13, Section 2.1]. The fact that Eq. (5.29) is not stochastic differential equation is an advantageous. The desirable property of random discretization is to have diminishing effect of randomness when $\Delta t \to 0$. In contrast to that, stochastic differential equation remain stochastic when $\Delta t \to 0$.

To use equation Eq. (5.29) we need to introduce approximation for $f''(t)$. When one-sided first-order finite difference approximation is used we obtain the scheme



(a) (b)

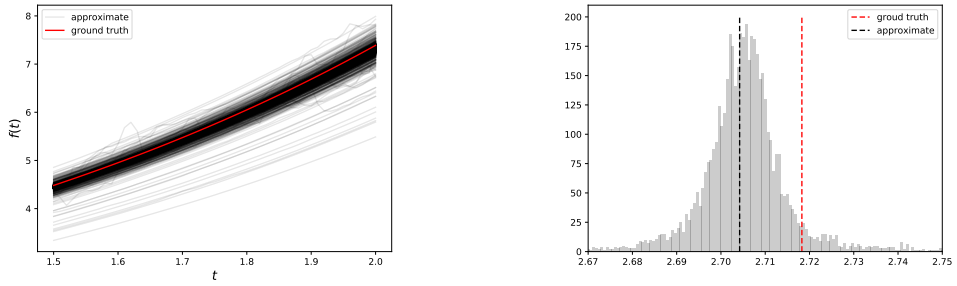
Figure 5.6: Differentiation Eq. (5.27) and integration Eq. (5.28) applied to the function $f(x) = \exp(\sin(3\pi x)) + \exp(-\cos(\pi x))$. In (a) one can see the value if the derivative as well as the distribution at $x = 4/9$. In (b) one can find the exact value of the integral $\int_{-1}^{1} dx\, f(x)$ and the distribution of approximate values.

$$f_{i+1} = f_i + \Delta t\, u(f_i, t_i) + + \frac{\epsilon}{2}\left(f_{i-2} - 2f_{i-1} + f_i\right), f_0 = f(0). \tag{5.30}$$

We apply this scheme to the problem $\dot{f}(t) = f(t), f(0) = 1$. The results can be found in Fig. 5.7. This time the uncertainty is clearly poorly calibrated, but still there is non-vanishing overlap between the probability density and the value of exact solution. Overall, we can see that the proposed method is biased. It tends to produce underestimation to the exact solution.

5.3 Iterative methods for sparse linear systems

Results from the previous section show that the symmetry itself is not difficult to find. However, the resulting distribution is intractable in most cases. Of course, the distribution can be computed using some sampling technique. However, it does not seem to make much sense to use an exhaustive sampling to quantify uncertainty which results from the finite amount of computation or approximation made for basic numerical tasks such as integration or the solution of linear systems. In place of uncertainty quantification, the very same computational resources can be more reasonably used to improve the numerical approximation itself. After the introduction of hidden representation for linear problems, we address this issue in Section 5.3.1. The variational approximation allows one to reduce the numerical cost in comparison



(a) (b)

Figure 5.7: Finite difference approximation Eq. (5.30) applied to the problem $\dot{f}(t) = f(t), f(0) = 1$. In (a) random trajectories are given. In (b) the histogram for $t = 1$ is given.

to sampling. Since the resulting algorithm (given in Section 5.4) is still not efficient from the point of view of storage and numerical complexity, we propose further modifications in Section 5.4.1. For all iterations, necessary and sufficient (or sufficient) conditions for measure concentration along with the "uncertainty quantification" results are given in Section 5.4.2. Section 5.4.4 explains the formal way to extend the uncertainty quantification results derived for probabilistic instationary Richardson to an arbitrary iterative method.

Consider the following transformation of the linear problem (5.1):

$$\widetilde{\boldsymbol{x}} = \boldsymbol{U}^{-1}\boldsymbol{x}, \ \widetilde{\boldsymbol{A}} = \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{U}, \ \widetilde{\boldsymbol{b}} = \boldsymbol{V}^{-1}\boldsymbol{b}, \tag{5.31}$$

where $\det \boldsymbol{V} \neq 0$ and $\det \boldsymbol{U} \neq 0$. If one solves the resulting linear system

$$\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{b}}, \tag{5.32}$$

and then switches back to the original basis, the result will coincide with (5.2), i.e.

$$\boldsymbol{U}\widetilde{\boldsymbol{A}}^{-1}\widetilde{\boldsymbol{b}} = \boldsymbol{A}^{-1}\boldsymbol{b}. \tag{5.33}$$

Thus, (5.31) and (5.33) correspond to the symmetry transformations g_ξ and ϕ_ξ, respectively.

Now, as an approximate solution to (5.32), we consider an arbitrary iterative scheme, for example, the multigrid or conjugate gradient method. It is well known

[Hac16, Sections 2.2.1 and 2.2.2] that a general iterative method can be presented in its first or second normal form:

$$\widetilde{\boldsymbol{x}}^{(n+1)} = \widetilde{\boldsymbol{M}}\widetilde{\boldsymbol{x}}^{(n)} + \widetilde{\boldsymbol{N}}\widetilde{\boldsymbol{b}}, \tag{5.34}$$

$$\widetilde{\boldsymbol{x}}^{(n+1)} = \widetilde{\boldsymbol{x}}^{(n)} + \widetilde{\boldsymbol{N}}\widetilde{\boldsymbol{r}}^{(n)},\ \widetilde{\boldsymbol{r}}^{(n)} = \widetilde{\boldsymbol{b}} - \widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{x}}^{(n)}; \tag{5.35}$$

$$\det \widetilde{\boldsymbol{N}} \neq 0,\ \boldsymbol{I} = \widetilde{\boldsymbol{M}} + \widetilde{\boldsymbol{N}}\widetilde{\boldsymbol{A}}. \tag{5.36}$$

The last line assures that the steady state is unique and coincides with the solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, i.e., the method is consistent. As a rule, matrices $\widetilde{\boldsymbol{M}}$ and $\widetilde{\boldsymbol{N}}$ are nonlinear functions of $\widetilde{\boldsymbol{A}}$, so the wide class of distributions over $\boldsymbol{U}$ and $\boldsymbol{V}$ produce a nontrivial distribution over $\boldsymbol{x}^{n+1}$ the same way as we saw in Section 5.2.1. To see the effect of the change of bases explicitly, we will write (5.34) and the resulting $\boldsymbol{x}^{n+1}$ for Jacobi iteration.

Define $\boldsymbol{D}(\boldsymbol{A})$ to be a matrix functions that extract the diagonal part, i.e. $D(A)_{ij} \equiv A_{ij}\delta_{ij}$. The matrix form of Jacobi iteration reads

$$\widetilde{\boldsymbol{x}}^{n+1} = \left(\boldsymbol{I} - \left(\boldsymbol{D}\left(\widetilde{\boldsymbol{A}}\right)\right)^{-1}\widetilde{\boldsymbol{A}}\right)\widetilde{\boldsymbol{x}}^{n} + \left(\boldsymbol{D}\left(\widetilde{\boldsymbol{A}}\right)\right)^{-1}\widetilde{\boldsymbol{b}}. \tag{5.37}$$

Transformation to the original bases gives

$$\boldsymbol{x}^{n+1} = \left(\boldsymbol{I} - \boldsymbol{U}\left(\boldsymbol{D}\left(\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{U}\right)\right)^{-1}\boldsymbol{V}^{-1}\boldsymbol{A}\right)\boldsymbol{x}^{n} + \boldsymbol{U}\left(\boldsymbol{D}\left(\boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{U}\right)\right)^{-1}\boldsymbol{V}^{-1}\boldsymbol{b}. \tag{5.38}$$

It is evident that $\boldsymbol{U}$ and $\boldsymbol{V}$ are explicitly presented in the expression for $\boldsymbol{x}^{n+1}$ unless both of them are diagonal matrices. In the latter case, (5.38) reduces to the ordinary Jacobi iteration [Saa20, Section 4.1].

We regard formulae like (5.38) obtained from (5.34) as a law for $\boldsymbol{x}^{n+1}$ given $\boldsymbol{x}^{n}$, i.e., should $p\left(\boldsymbol{U},\boldsymbol{V}\right)$ be the probability density function of $\boldsymbol{U}$ and $\boldsymbol{V}$, the conditional probability $p\left(\boldsymbol{x}^{n+1}|\boldsymbol{x}^{n}\right)$ is

$$p\left(\boldsymbol{x}^{n+1}|\boldsymbol{x}^{n}\right) = \int d\boldsymbol{U}d\boldsymbol{V}p\left(\boldsymbol{U},\boldsymbol{V}\right)\delta\left(\boldsymbol{x}^{(n+1)} - \boldsymbol{U}\widetilde{\boldsymbol{M}}\boldsymbol{U}^{-1}\boldsymbol{x}^{(n)} - \boldsymbol{U}\widetilde{\boldsymbol{N}}\boldsymbol{V}^{-1}\boldsymbol{b}\right), \tag{5.39}$$

where $\widetilde{\boldsymbol{M}}$ and $\widetilde{\boldsymbol{N}}$ are matrices that were used in the solution of (5.32) and then transformed back to the original basis in a full analogy with Jacobi iteration (5.38).

Usually, the iterative method (5.34) is applied repeatedly until some convergence criterion is met. Considered from this perspective, the method of hidden representation allows translating an arbitrary iterative method to the random process with discrete-time. For such situations, one needs to define a filtration (for example, see

[CE15, Section 3.1]) and establish convergence in probability in some form. It is likely that the proof of this kind will include tail bounds for eigenvalues like ones discussed in [Tro19, Section 1.1.1] combined with necessary and sufficient conditions for convergence of the underlying iterative method. We choose a different route to follow: we construct a variational approximation based on a multivariate normal distribution for $\boldsymbol{x}^n$, derive updates for the mean vector and covariance matrix, and prove measure concentration for these deterministic update rules. This is explained in detail in the next two subsections.

5.3.1 Variational approximation

Let $\boldsymbol{U}_n$ and $\boldsymbol{V}_n$ be random matrices that perform transformations during the step n and $p_n\left(\boldsymbol{U}_n, \boldsymbol{V}_n\right)$ be a joint probability density function. We also define iteration matrices from (5.34) after two symmetry transformations:

$$\boldsymbol{M}_n \equiv \boldsymbol{U}_n \widetilde{\boldsymbol{M}}\left(\boldsymbol{U}_n, \boldsymbol{V}_n, \boldsymbol{A}\right) \boldsymbol{U}_n^{-1}, \;\; \boldsymbol{N}_n \equiv \boldsymbol{U}_n \widetilde{\boldsymbol{N}}\left(\boldsymbol{U}_n, \boldsymbol{V}_n, \boldsymbol{A}\right) \boldsymbol{V}_n^{-1}. \tag{5.40}$$

The law of $\boldsymbol{x}^{n+1}$ given $\boldsymbol{x}^n$ becomes

$$\boldsymbol{x}^{n+1} = \boldsymbol{M}_n \boldsymbol{x}^n + \boldsymbol{N}_n \boldsymbol{b}. \tag{5.41}$$

The quantity of interest that contains all the relevant information about random process is a probability density function for the approximate solution $\boldsymbol{x}^n$ given the deterministic starting point, i.e. the distribution $p\left(\boldsymbol{x}^n | \boldsymbol{x}^0\right)$. In what follows, we drop $\boldsymbol{x}^0$ and simply write $p\left(\boldsymbol{x}^n\right)$. Since in general this distribution is not easy to obtain from transition probabilities (5.39), we resort to the approximation of the form

$$p\left(\boldsymbol{x}^n\right) = \mathcal{N}\left(\boldsymbol{x}^n | \boldsymbol{\mu}^n, \boldsymbol{\Sigma}^n\right), \tag{5.42}$$

where the right hand side stands for the probability density function of the multivariate normal distribution with mean $\boldsymbol{\mu}^n$ and covariance matrix $\boldsymbol{\Sigma}^n$. To find the connection between parameters of the distributions in successive iteration steps, one can use

$$p\left(\boldsymbol{x}^{n+1}\right) = \int d\boldsymbol{x}^n p\left(\boldsymbol{x}^{n+1} | \boldsymbol{x}^n\right) p\left(\boldsymbol{x}^n\right), \tag{5.43}$$

and the fact that (5.42) is fully specified by its first two moments. These two facts allow us to relate $\boldsymbol{\mu}^{n+1}$ with $\boldsymbol{\mu}^n$:

$$\begin{aligned}\boldsymbol{\mu}^{n+1} &\equiv \int d\boldsymbol{x}^{n+1} p\left(\boldsymbol{x}^{n+1}\right) \boldsymbol{x}^{n+1} = \\ &= \int d\boldsymbol{U}_n d\boldsymbol{V}_n d\boldsymbol{x}^n d\boldsymbol{x}^{n+1} p\left(\boldsymbol{x}^n\right) p_n\left(\boldsymbol{U}_n, \boldsymbol{V}_n\right) \delta\left(\boldsymbol{x}^{n+1} - \boldsymbol{M}_n \boldsymbol{x}^n - \boldsymbol{N}_n \boldsymbol{b}\right) \boldsymbol{x}^{n+1} \\ &= \mathbb{E}_{\boldsymbol{U}_n, \boldsymbol{V}_n}\left[\boldsymbol{M}_n\right] \boldsymbol{\mu}^n + \mathbb{E}_{\boldsymbol{U}_n, \boldsymbol{V}_n}\left[\boldsymbol{N}_n\right] \boldsymbol{b}. \end{aligned} \quad (5.44)$$

The same reasoning leads to the update for covariance matrix:

$$\begin{aligned}\boldsymbol{\Sigma}^{n+1} &\equiv \int d\boldsymbol{x}^{n+1} p\left(\boldsymbol{x}^{n+1}\right) \boldsymbol{x}^{n+1} \left(\boldsymbol{x}^{n+1}\right)^T - \boldsymbol{\mu}^{n+1} \left(\boldsymbol{\mu}^{n+1}\right)^T = \\ &= \mathbb{E}_{\boldsymbol{U}_n, \boldsymbol{V}_n}\left[\boldsymbol{M}_n \boldsymbol{\Sigma}^n \boldsymbol{M}_n^T + \left(\boldsymbol{M}_n \boldsymbol{\mu}^n + \boldsymbol{N}_n \boldsymbol{b}\right) \left(\boldsymbol{M}_n \boldsymbol{\mu}^n + \boldsymbol{N}_n \boldsymbol{b}\right)^T\right] - \boldsymbol{\mu}^{n+1} \left(\boldsymbol{\mu}^{n+1}\right)^T. \end{aligned} \quad (5.45)$$

Together with the initial delta distribution $\boldsymbol{\mu}^0 \equiv \boldsymbol{x}^0$, $\boldsymbol{\Sigma}^0 = 0$ that provides initial conditions, (5.44) and (5.45) constitute a closed iterative scheme that defines multivariate normal distributions over each subsequent approximation to exact solution.

The precise form of the updates (5.44), (5.45) depends on the iterative method M, N and the set of joint distributions $p_n\left(\boldsymbol{U}_n, \boldsymbol{V}_n\right)$. To obtain an explicit form, one needs to specify them, and then compute (or approximate) expected values of some nonlinear functions of $\boldsymbol{U}_n, \boldsymbol{V}_n$. Even though this problem is somewhat easier than the computation of the probability density function $p\left(\boldsymbol{x}^{n+1}\right)$ directly from $p_n\left(\boldsymbol{U}_n, \boldsymbol{V}_n\right)$ and (5.41), it is still often not possible to obtain the closed form of the updates because (5.41) is a nonlinear matrix function with respect to $\boldsymbol{U}_n$ and $\boldsymbol{V}_n$. In the next section, we consider the Richardson iteration and the set of matrix normal distributions $p_n\left(\boldsymbol{U}_n, \boldsymbol{V}_n\right)$ for which update rules (5.44), (5.45) can be written in an explicit form.

5.4 Probabilistic instationary Richardson iteration

In this section, we confine our attention to the Richardson iteration, for which $\widetilde{\boldsymbol{N}} = \boldsymbol{I}$, so the law for x^{n+1} given x^n is

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \boldsymbol{U}_n \left(\boldsymbol{V}_n\right)^{-1} \left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^n\right). \quad (5.46)$$

We also choose $\boldsymbol{U}_n \equiv \boldsymbol{I}$ and require that $(\boldsymbol{V}_n)^{-1}$ have a normal distribution (see [GN18, Definition 2.2.1]), i.e.

$$p_n\left((\boldsymbol{V}_n)^{-1}\right) = \mathcal{N}\left((\boldsymbol{V}_n)^{-1}\,\middle|\,\boldsymbol{R}_n, \widetilde{\boldsymbol{\Sigma}}_n \otimes \widetilde{\boldsymbol{\Psi}}_n\right), \tag{5.47}$$

where $\boldsymbol{R}_n$ and $\widetilde{\boldsymbol{\Sigma}}_n \otimes \widetilde{\boldsymbol{\Psi}}_n$ are mean and covariance, respectively, and $\otimes$ stands for Kronecker product. The following proposition characterises probabilistic Richardson iteration.

Proposition 5.4.1. *Let $\boldsymbol{U}_n$ and $\boldsymbol{V}_n$ be independent, $\boldsymbol{U}_n \equiv \boldsymbol{I}$ $\forall n$, the law of $\boldsymbol{x}^{n+1}$ given $\boldsymbol{x}^n$ is fixed by* (5.46)*, probability density function of $(\boldsymbol{V}_n)^{-1}$ is given by* (5.47)*, then* (5.44) *becomes*

$$\boldsymbol{\mu}^{n+1} = \boldsymbol{\mu}^n + \boldsymbol{R}_n\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{\mu}^n\right) = \left(\boldsymbol{I} - \boldsymbol{R}_n\boldsymbol{A}\right)\boldsymbol{\mu}^n + \boldsymbol{R}_n\boldsymbol{b}, \tag{5.48}$$

and (5.45) *leads to*

$$\boldsymbol{\Sigma}^{n+1} = \left(\boldsymbol{I} - \boldsymbol{R}_n\boldsymbol{A}\right)\boldsymbol{\Sigma}^n\left(\boldsymbol{I} - \boldsymbol{R}_n\boldsymbol{A}\right)^T + \widetilde{\boldsymbol{\Sigma}}_n\left(\mathsf{tr}\left(\boldsymbol{\Sigma}^n\boldsymbol{A}^T\widetilde{\boldsymbol{\Psi}}_n\boldsymbol{A}\right) + \left(\boldsymbol{r}^n\right)^T\widetilde{\boldsymbol{\Psi}}_n\boldsymbol{r}^n\right), \tag{5.49}$$

where $\boldsymbol{r}^n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{\mu}^n$.

Updates (5.48), (5.49) should be supplemented with initial conditions $\boldsymbol{\Sigma}^0 = 0$, $\boldsymbol{\mu}^0 = \boldsymbol{x}^0$ and an explicit definition of $\boldsymbol{R}_n$, $\widetilde{\boldsymbol{\Psi}}_n$, $\widetilde{\boldsymbol{\Sigma}}_n$ for each step. We discuss different choices in the sections that follow.

We point out here a few immediate consequences of the form of (5.48), (5.49). First, the update for mean (5.48) coincides with the instationary preconditioned Richardson iteration with a new preconditioner for each step. Second, iteration (5.49) preserves positive definiteness, as stated in the following proposition.

Proposition 5.4.2. *If $\boldsymbol{\Sigma}^0 = 0$, $\boldsymbol{\mu}^0 = \boldsymbol{x}^0$, $\widetilde{\boldsymbol{\Psi}}_n > 0$, $\widetilde{\boldsymbol{\Sigma}}_n > 0$, then for the updates defined by* (5.48), (5.49)*, $\boldsymbol{\Sigma}^n > 0$ for all $n \geq 1$. In addition, if at least one matrix is positive semidefinite, i.e., $\widetilde{\boldsymbol{\Psi}}_n \geq 0$ or $\widetilde{\boldsymbol{\Sigma}}_n \geq 0$, or both, than $\boldsymbol{\Sigma}_n \geq 0$ for all $n \geq 1$.*

Finally, one can formulate the following uncertainty quantification result valid for a finite iteration step.

Proposition 5.4.3. *For positive definite matrices $\widetilde{\boldsymbol{\Psi}}_n > 0$, $\widetilde{\boldsymbol{\Sigma}}_n > 0$, the following inequality holds:*

$$\frac{\mathsf{tr}\left(\boldsymbol{\Sigma}^{n+1}\left(\boldsymbol{\Sigma}^1\right)^{-1}\right)}{\mathsf{tr}\left(\widetilde{\boldsymbol{\Sigma}}_n\left(\widetilde{\boldsymbol{\Sigma}}_0\right)^{-1}\right)} > \frac{\left(\boldsymbol{r}^n\right)^T\widetilde{\boldsymbol{\Psi}}_n\boldsymbol{r}^n}{\left(\boldsymbol{r}^0\right)^T\widetilde{\boldsymbol{\Psi}}_0\boldsymbol{r}^0}. \tag{5.50}$$

Asymptotic results are formulated in Section 5.4.2.

Algorithm 7 Low-rank probabilistic Richardson.

Given $\boldsymbol{\mu}^n$, $\boldsymbol{\Sigma}^n = \sum_{i=1}^{L} \lambda_i^n \boldsymbol{u}_i^n \left(\boldsymbol{u}_i^n\right)^T$, and the linear system $\{\boldsymbol{A}, \boldsymbol{b}\}$.
Choose $\widetilde{\boldsymbol{\Sigma}}_n \equiv \sum_{i=1}^{L_n} \sigma_i^n \boldsymbol{v}_i^n \left(\boldsymbol{v}_i^n\right)^T$, $\widetilde{\boldsymbol{\Psi}}_n \equiv \boldsymbol{\psi}_n^T \boldsymbol{\psi}_n$ and $\boldsymbol{R}_n$.

$\boldsymbol{r}^n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{\mu}^n$
$\boldsymbol{\mu}^{n+1} = \boldsymbol{\mu}^n + \boldsymbol{R}_n \boldsymbol{r}^n$

$\beta^n = \left\| \boldsymbol{\Psi}_n \boldsymbol{r}^n \right\|_2^2$
$\alpha^n = 0$
for $i = \overline{1, L}$ **do**
  $\widetilde{\boldsymbol{u}}_i^n = \boldsymbol{A}\boldsymbol{u}_i^n$
  $\alpha^n \leftarrow \alpha^n + \lambda_i^n \left\| \boldsymbol{\Psi}_n \widetilde{\boldsymbol{u}}_i^n \right\|_2^2$
  $\widetilde{\boldsymbol{u}}_i^n \leftarrow \boldsymbol{u}_i^n - \boldsymbol{R}_n \widetilde{\boldsymbol{u}}_i^n$
end for
$\boldsymbol{V} = \begin{pmatrix} | & & | & | & & | \\ \widetilde{\boldsymbol{u}}_1^n & \dots & \widetilde{\boldsymbol{u}}_L^n & \boldsymbol{v}_1^n & \dots & \boldsymbol{v}_{L_n}^n \\ | & & | & | & & | \end{pmatrix}$; $\{\boldsymbol{Q}, \boldsymbol{R}\} \leftarrow \mathsf{QR}\left(\boldsymbol{V}\right)$
$\boldsymbol{D} = \begin{pmatrix} \mathrm{diag}\left(\{\lambda_i^n\}\right) & 0 \\ 0 & \mathrm{diag}\left(\{\left(\alpha^n + \beta^n\right)\sigma_i^n\}\right) \end{pmatrix}$; $\boldsymbol{D} \leftarrow \boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T$
$\left\{ \{\widetilde{v}_i\}_{i=\overline{1, L+L_n}}, \left\{\widetilde{\lambda}_i\right\}_{i=\overline{1, L+L_n}} \right\} \leftarrow$ Spectral decomposition (D)
Pick $\boldsymbol{M}$ leading eigenvalues $\left\{\lambda_i^{n+1}\right\} = \left\{\widetilde{\lambda}_i\right\}_{i=\overline{1, L}}$.
Keep corresponding eigenvectors $\{\widetilde{\boldsymbol{v}}_i\}_{i=\overline{1, L}}$.
$\begin{pmatrix} | & & | \\ \boldsymbol{u}_1^{n+1} & \dots & \boldsymbol{u}_L^{n+1} \\ | & & | \end{pmatrix} = \boldsymbol{Q} \begin{pmatrix} | & & | \\ \widetilde{\boldsymbol{v}}_1 & \dots & \widetilde{\boldsymbol{v}}_L \\ | & & | \end{pmatrix}$
$\boldsymbol{\Sigma}^{n+1} = \sum_{i=1}^{L} \lambda_i^{n+1} \boldsymbol{u}_i^{n+1} \left(\boldsymbol{u}_i^{n+1}\right)^T$

5.4.1 The covariance matrix is intractable

The usual situation for an iterative method to be an algorithm of choice is when A is large and sparse. Under these circumstances, there are two common arguments against naive LU. The first one is that the $O\left(K^3\right)$ operations are needed to solve the problem with K degrees of freedom. The second one is the presence of fill-in: in practically interesting problems, K can be so large that it is impossible to store $\sim K^2$ elements for lower and upper triangular factors of the decomposition.

One can raise the same objections against the update rule for covariance matrix (5.49). First, the explicit product of matrices is present in (5.49) which corresponds to the roughly $O(K^3)$ operations per iteration. Second, even if $\widetilde{\boldsymbol{\Sigma}}_n$ and R_n are sparse, after a few iterations $\boldsymbol{\Sigma}^n$ becomes dense. We propose two modifications to make (5.49) tractable. Throughout this section we take $\widetilde{\boldsymbol{\Psi}}_n = I$ for brevity, see Section 5.4.3 for the algorithm with arbitrary $\widetilde{\boldsymbol{\Psi}}_n$.

The first one is to restrict the rank of the update and consider matrix $\widetilde{\boldsymbol{\Sigma}}_n$ of a special form. For arbitrary matrix $\boldsymbol{B}$, we define $[\boldsymbol{B}]_L$ to be the best approximation in L_2 operator norm to B among all matrices of rank up to L. Taking $\widetilde{\boldsymbol{\Sigma}}_n = \sum_{i=1}^{M_n} \sigma_i^n \boldsymbol{u}_i^n \left(\boldsymbol{u}_i^n\right)^T$ we introduce the low-rank approximate update for the covariance matrix

$$\boldsymbol{\Sigma}^{n+1} = \left[\left(\boldsymbol{I} - \boldsymbol{R}_n \boldsymbol{A}\right) \boldsymbol{\Sigma}^n \left(\boldsymbol{I} - \boldsymbol{R}_n \boldsymbol{A}\right)^T + \widetilde{\boldsymbol{\Sigma}}_n \left(\mathsf{tr}\left(\boldsymbol{A} \boldsymbol{\Sigma}^n \boldsymbol{A}^T \right) + \|\boldsymbol{r}_n\|_2^2 \right) \right]_L . \tag{5.51}$$

For this form of update it is convenient to store L eigenpairs of $\boldsymbol{\Sigma}^n$ and update them during each iteration. This can be done either via Bunch–Nielsen–Sorensen algorithm [GE94], or through QR factorization of $\{\boldsymbol{u}_i^n, \boldsymbol{v}_i^n\}$ (see Section 5.4.3 for details), where $\boldsymbol{u}_i^n$ and $\boldsymbol{v}_i^n$ are eigenvectors of $\widetilde{\boldsymbol{\Sigma}}_n$ and $\boldsymbol{\Sigma}^n$ respectively. The cost of both options is roughly $O\left(\left(L_n + L\right)^2 N\right)$ for $L \ll K$, $L_n \ll K$, and in addition to $\boldsymbol{\mu}^n$ one needs to store $LK + L$ scalars.

The other possibility is to consider the diagonal approximation. For arbitrary square matrix $\boldsymbol{B}$, we define $D(\boldsymbol{B})_{ij} = B_{ij}\delta_{ij}$. Taking $\boldsymbol{D}\left(\widetilde{\boldsymbol{\Sigma}}_n\right) = \widetilde{\boldsymbol{\Sigma}}_n$ we introduce the following approximate update for the covariance matrix

$$\boldsymbol{\Sigma}^{n+1} = \boldsymbol{D}\left(\left(\boldsymbol{I} - \boldsymbol{R}_n \boldsymbol{A}\right) \boldsymbol{\Sigma}^n \left(\boldsymbol{I} - \boldsymbol{R}_n \boldsymbol{A}\right)^T \right) + \widetilde{\boldsymbol{\Sigma}}_n \left(\mathsf{tr}\left(\boldsymbol{A} \boldsymbol{\Sigma}^n \boldsymbol{A}^T \right) + \|\boldsymbol{r}_n\|_2^2 \right) . \tag{5.52}$$

In case $\left(\boldsymbol{I} - \boldsymbol{R}_n \boldsymbol{A}\right)$ and $\boldsymbol{A}$ are sparse matrices, the cost of update is $O(K)$. If any one of them is dense, this form of update becomes computationally expensive. Also, the diagonal of $\boldsymbol{\Sigma}^n$, i.e. N additional scalars, needs to be stored. This modification is less interesting, because it provides poor directional information. We do not consider this update rule in what follows.

Results of this section put some restrictions on possible form of $\widetilde{\boldsymbol{\Sigma}}_n$ and $\widetilde{\boldsymbol{\Psi}}_n$. In the next section, we further restrict (see Proposition 5.4.4 – Proposition 5.4.7) covariance matrices to ensure convergence of iteration. Yet, even after that, the scheme remains flexible enough to permit quantification of uncertainty. The discussion on concrete choices of $\widetilde{\boldsymbol{\Sigma}}_n$ and $\widetilde{\boldsymbol{\Psi}}_n$ appears in Section 5.4.2.

5.4.2 Concentration of measure and alignment

In this section, we characterize the asymptotic behavior of probabilistic Richardson iteration. Since, according to our interpretation, one cannot control the input covariance matrix $\boldsymbol{\Sigma}^0 \equiv 0$, in the following proposition, we consider $\widetilde{\boldsymbol{\Sigma}}_0$ and $\boldsymbol{\mu}^n$ as a starting point of the stationary algorithm.

Proposition 5.4.4. *Let in* (5.48), (5.49) $\forall n \geq 1$ $\widetilde{\boldsymbol{\Sigma}}_n \equiv \widetilde{\boldsymbol{\Sigma}}$, *and* $\forall n$ $\widetilde{\boldsymbol{\Psi}}_n \equiv \widetilde{\boldsymbol{\Psi}}$, $\boldsymbol{R}_n \equiv \boldsymbol{R}$, $\det \boldsymbol{R} \neq 0$, *then the iterative method given by* (5.48), (5.49) *converges to* $\boldsymbol{\Sigma}^n = 0$, $\boldsymbol{\mu}^n = \boldsymbol{A}^{-1}b$ *for the arbitrary input* $\boldsymbol{b}$, $\widetilde{\boldsymbol{\Sigma}}_0$, $\boldsymbol{\mu}^0 \equiv \boldsymbol{x}^0$ *iff*

$$\begin{aligned} &\rho\left(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{A}\right) < 1, \\ &\rho\left(\left(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{A}\right) \otimes \left(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{A}\right) + \mathsf{vec}\left(\widetilde{\boldsymbol{\Sigma}}\right) \mathsf{vec}\left(\boldsymbol{A}^T \widetilde{\Psi} \boldsymbol{A}\right)^T\right) < 1. \end{aligned} \tag{5.53}$$

This proposition mimics classical necessary and sufficient convergence results available in the literature (the definition of convergence [Hac16, Definition 2.6], and the theorem on the convergence of linear iterative methods [Hac16, Theorem 2.16]).

The stationary iterations can be too restrictive from the point of view of uncertainty quantification. The following result gives a sufficient condition for convergence of the instationary iterative scheme.

Proposition 5.4.5. *If* $\lim\limits_{n\to\infty} \boldsymbol{R}_n = \boldsymbol{R}$, *and* $\lim\limits_{n\to\infty} \widetilde{\boldsymbol{\Sigma}}_n = \widetilde{\boldsymbol{\Sigma}}$, $\lim\limits_{n\to\infty} \widetilde{\boldsymbol{\Psi}}_n = \widetilde{\boldsymbol{\Psi}}$:

$$\begin{aligned} &\rho\left(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{A}\right) < 1, \\ &\rho\left(\left(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{A}\right) \otimes \left(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{A}\right) + \mathsf{vec}\left(\widetilde{\boldsymbol{\Sigma}}\right) \mathsf{vec}\left(\boldsymbol{A}^T \widetilde{\boldsymbol{\Psi}} \boldsymbol{A}\right)^T\right) < 1, \end{aligned} \tag{5.54}$$

the iterative method given by (5.48), (5.49) *converges to* $\boldsymbol{\Sigma}^n = 0$, $\boldsymbol{\mu}^n = \boldsymbol{A}^{-1}\boldsymbol{b}$ *for arbitrary input* $\boldsymbol{b}$, $\widetilde{\boldsymbol{\Sigma}}_0$, $\boldsymbol{\mu}^0 \equiv \boldsymbol{x}^0$.

In both Proposition 5.4.4, Proposition 5.4.5 the first lines of equations (5.53), (5.54) represents standard requirements familiar from the analysis of classical iterative methods. The second line of (5.53) or (5.54) is a consequence of the distribution over representations of the problem: variance matrices have a direct impact on the distribution of the spectrum of $\widetilde{\boldsymbol{A}}$, hence affect the convergence. In general, it is more convenient to have just one convergence criterion to check. The following proposition gives a practical way to drop the second condition in Proposition 5.4.5.

Proposition 5.4.6. *Provided the following conditions are met*

$$\begin{aligned} &\lim_{n\to\infty} \boldsymbol{R}_n = \boldsymbol{R} : \rho\left(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{A}\right) < 1, \\ &\lim_{k\to\infty} \widetilde{\boldsymbol{\Sigma}}_k = 0 \ \left(or \ \lim_{k\to\infty} \widetilde{\boldsymbol{\Psi}}_k = 0\right), \\ &\exists \boldsymbol{C} : \forall n \left\|\widetilde{\boldsymbol{\Psi}}_n\right\| < \boldsymbol{C} \ \left(or \ \exists \boldsymbol{C} : \forall n \left\|\widetilde{\boldsymbol{\Sigma}}_n\right\| < \boldsymbol{C}\right); \end{aligned} \tag{5.55}$$

the iterative method given by (5.48), (5.49) *converges to* $\boldsymbol{\Sigma}^n = 0$, $\boldsymbol{\mu}^n = \boldsymbol{A}^{-1}b$ *for arbitrary input* $\boldsymbol{b}$, $\widetilde{\boldsymbol{\Sigma}}_0$, $\boldsymbol{\mu}^0 \equiv \boldsymbol{x}^0$.

That is, if the uncertainty in the choice of representation vanishes in the course of iterations, and the underlying linear method converges, the probabilistic scheme converges as well. This condition provides a very convenient way to guarantee that the scheme converges, and at the same time, one keeps full control over uncertainty in the non-asymptotic regime.

All statements so far characterize the original iteration scheme, which we found to be impractical because the covariance matrix becomes dense. The next proposition elaborates on the relationship between the convergence of the approximate scheme and the exact one.

Proposition 5.4.7. *If* $\widetilde{\boldsymbol{\Psi}}_n$, $\widetilde{\boldsymbol{\Sigma}}_n$, $\boldsymbol{R}_n$ *are chosen in a way that iterations* (5.48), (5.49) *converge, the low-rank iterations* (5.51) *induced by the same sequence of matrices converge as well.*

One of the interesting features of the probabilistic numerical methods in linear algebra is that besides the scalar proxies of uncertainty like $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^n\|_2^2$ they provide directional information through the eigenvectors of covariance matrix $\boldsymbol{\Sigma}^n$. The simple alignment result can be found below.

Proposition 5.4.8. *Let* $\boldsymbol{R}_n \equiv \boldsymbol{R}$, *the iteration matrix is symmetric* $\boldsymbol{M} \equiv \boldsymbol{I} - \boldsymbol{R}\boldsymbol{A}$. *We represent* $\boldsymbol{M}$ *using spectral theorem* $\boldsymbol{M} = \boldsymbol{S}\boldsymbol{D}\boldsymbol{S}^T$, *denote i-th column of* $\boldsymbol{S}$ *as* $\boldsymbol{S}_{i\star}$, *and express the covariance matrix in basis* $\{\boldsymbol{S}_{i\star}\}_i$: $\boldsymbol{\Sigma}^n \equiv \sum_{i,j} \sigma_{ij}^n \boldsymbol{S}_{i\star}(\boldsymbol{S}_{j\star})^T$. *Then for instationary iterations with* $\widetilde{\boldsymbol{\Sigma}}_0 = \widetilde{\boldsymbol{\Sigma}}, \det\left(\widetilde{\boldsymbol{\Sigma}}\right) \neq 0$, $\widetilde{\boldsymbol{\Psi}}_0 = \widetilde{\boldsymbol{\Psi}}, \det\left(\widetilde{\boldsymbol{\Psi}}\right) \neq 0$, *and* $\widetilde{\boldsymbol{\Sigma}}_n = \widetilde{\boldsymbol{\Psi}}_n = 0$ *for* $n \geq 1$, *one has*

$$|\lambda_k| \leq |\lambda_i|, \ |\lambda_l| < |\lambda_i| \Longrightarrow \lim_{n\to\infty} \frac{\sigma_{kl}^n}{\sigma_{ii}^n} = 0. \tag{5.56}$$

So in the absence of control ($\widetilde{\boldsymbol{\Sigma}}_n = \widetilde{\boldsymbol{\Psi}}_n = 0$ for $n \geq 1$), eigenvectors of $\boldsymbol{\Sigma}^n$ align with the subspace, corresponding to the leading eigenvectors of the iteration matrix.

This is reasonable behavior, taking into account that the components of the error lying in the same subspace have the slowest rate of decay asymptotically. In the presence of control, one can actively influence the direction of uncertainty by the suitable choice of $\widetilde{\boldsymbol{\Sigma}}$ and $\widetilde{\boldsymbol{\Psi}}$ on each stage.

5.4.3 Algorithm

We put forward the general algorithm corresponding to (5.51) with arbitrary $\widetilde{\boldsymbol{\Psi}}_n$, $\widetilde{\boldsymbol{\Sigma}}_n$ and $\boldsymbol{R}_n$. As discussed before, iterative methods are considered in the context of large sparse linear systems, so in the construction of the algorithm we follow two principles:

- Matrix-matrix products are forbidden.
- It is not possible to store more than $O(K)$ floats.

To fulfill the first requirement we demand $\widetilde{\boldsymbol{\Sigma}}$ and $\widetilde{\boldsymbol{\Psi}}$ to be of special form

$$\widetilde{\boldsymbol{\Sigma}}_n \equiv \sum_{i=1}^{L_n} \sigma_i^n \boldsymbol{v}_i^n \left(\boldsymbol{v}_i^n\right)^T, \;\; \widetilde{\boldsymbol{\Psi}}_n \equiv \boldsymbol{\psi}_n^T \boldsymbol{\psi}_n, \tag{5.57}$$

where $\boldsymbol{\psi}_n$ is a matrix of the same shape as $\widetilde{\boldsymbol{\Psi}}_n$, and $\{\sigma_i^n, \boldsymbol{v}_i^n\}$ is an eigenpair of $\widetilde{\boldsymbol{\Sigma}}_n$. In addition, we store $\boldsymbol{\Sigma}^n$ in the same form as $\widetilde{\boldsymbol{\Sigma}}_n$.

The second requirement is addressed by the low-rank approximation (5.51), the convergence of which is characterised in Proposition 5.4.7.

he body of Algorithm 7 provides the set of instructions for the update of a multivariate normal distribution, corresponding to the current iteration step. We would like to make a few comments. First, in the description of the algorithm we put two additional horizontal lines (one of which separates mean update from the update of the covariance matrix) to improve readability. Second, even though we write expressions like $\boldsymbol{\Sigma}^n = \sum_{i=1}^{L} \lambda_i^n \boldsymbol{u}_i^n \left(\boldsymbol{u}_i^n\right)^T$, one should not compute them explicitly, in place they are expected to be stored like a set of vectors $\{\boldsymbol{u}_i^n\}$ and real numbers $\{\lambda_i^n\}$. Also, Algorithm 7 contains two decomposition: QR (line 12) that involve $L_n + L$ vectors from $\mathbb{R}^K$, and eigendecomposition (line 14) of the matrix from $\mathbb{R}^{(L_n+L)\times(L_n+L)}$. The first operation is $O(K\left(L_n + L\right)^2)$ and the second is $O((L_n + L)^3)$. Given that $L \ll K$, $L_n \ll K$, both of these operations are tractable for large sparse matrices.

The discussion about the precise choice of $\widetilde{\boldsymbol{\Psi}}_n$, $\widetilde{\boldsymbol{\Sigma}}_n$ can be found in Section 5.4.5, but the development is at a preliminary stage: we are not in a position to give precise statements on the uncertainty quantification for finite n, even though we do provide some heuristics.

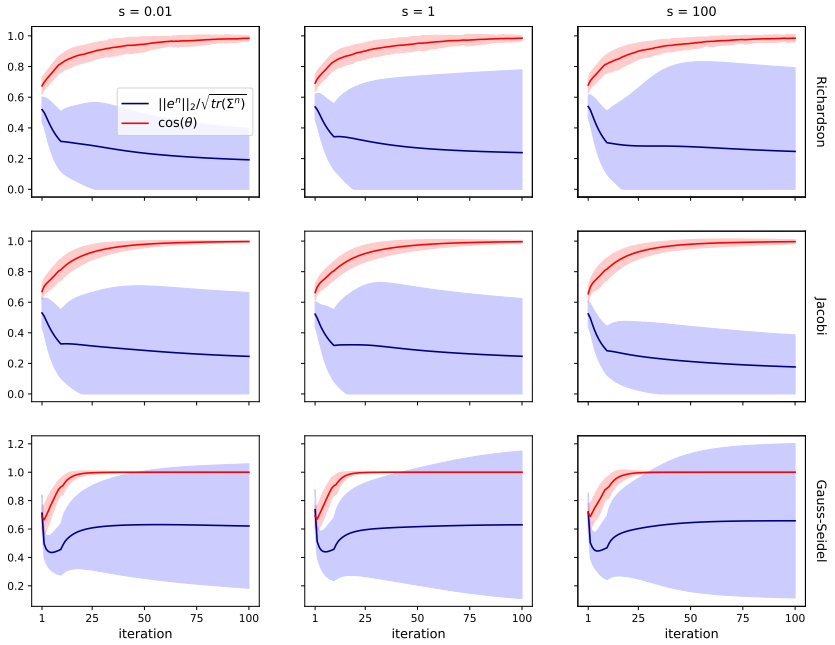


Figure 5.8: Results of uncertainty calibration (5.63) for random matrices (5.64). Thick lines represent empirical mean, shaded region covers (empirical mean) $\pm$ $\sqrt{\text{(empirical variance)}}$. Details are given after (5.64).

5.4.4 Connection with other iterative methods

It is known, that any iterative method can be seen as a transformed (preconditioned) version of the Richardson scheme [Hac16, Proposition 5.44]. Namely, if one compares the second normal form of arbitrary iteration $\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \boldsymbol{N}_{\text{it}}\boldsymbol{r}^n$ with the Richardson scheme $\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \boldsymbol{r}^n$, it is evident that they coincide in case one applies the later to the transformed system $\boldsymbol{NAx} = \boldsymbol{Nb}$. Since all linear iterations can be presented in the second normal form, it is possible to interpret the stationary Richardson scheme with $\boldsymbol{R}_n \equiv \boldsymbol{R} = \boldsymbol{N}_{\text{it}}$ as a probabilistic counterpart of a given iterative method.

For example, consider the splitting of $\boldsymbol{A}$ into the diagonal $\boldsymbol{D}$, the upper triangular $-\boldsymbol{E}$, and the lower triangular $-\boldsymbol{F}$ parts, i.e., $\boldsymbol{A} \equiv \boldsymbol{D} - \boldsymbol{E} - \boldsymbol{F}$. Then, the iteration matrices of Gauss-Seidel and Jacobi methods are defined by

$$\boldsymbol{N}_{\text{GS}} = (\boldsymbol{D} - \boldsymbol{E})^{-1}, \quad \boldsymbol{N}_{\text{Jacobi}} = \boldsymbol{D}^{-1}. \tag{5.58}$$

Therefore, one can run Algorithm 7 with $\boldsymbol{R}_n \equiv \boldsymbol{R} = \boldsymbol{N}_{\text{GS}}$ or $\boldsymbol{R}_n \equiv \boldsymbol{R} = \boldsymbol{N}_{\text{GS}}$ and call the resulting iteration "probabilistic Gauss-Seidel" or "probabilistic Jacobi"

methods. The same idea can be used to obtain "probabilistic multigrid" or "probabilistic projection methods".

However, note that probabilistic algorithms obtained from the Algorithm 7 in the way just described, do not coincide with the ones stemming from the hidden representation applied for these methods (for example, this should be evident for the Jacobi iteration given by (5.38)). Nevertheless, for relaxation methods or restarted projection methods (restarted FOM, one-dimensional methods such as steepest descent, minimal residual, etc.), these tactics produce a controllable distribution over approximate solution. On the other hand, methods like conjugate gradient or GMRES update N_n for each iteration and obtain solution from the same starting vector, so they are effectively "one-shot methods", i.e., if one tries to transform them as Gauss-Seidel or Jacobi, they will correspond to the single iteration and the resulting covariance matrix will be trivial $\boldsymbol{\Sigma}^1 = \widetilde{\boldsymbol{\Sigma}}_0 \left(\boldsymbol{r}^0\right)^T \widetilde{\boldsymbol{\Psi}}_0 \boldsymbol{r}^0$. To get a nontrivial result for GMRES or conjugate gradient, one needs to obtain an explicit form of (5.44), (5.45) or other variational approximation directly for these iterations.

5.4.5 Calibration of the uncertainty

Note that in this and the following section, we use rescaled errors and residuals, i.e., $\|\boldsymbol{e}^n\| \leftarrow \|\boldsymbol{e}^n\| \,/\, \|\boldsymbol{e}^0\|$ and $\|\boldsymbol{r}^n\| \leftarrow \|\boldsymbol{r}^n\| \,/\, \|\boldsymbol{r}^0\|$.

To measure how well uncertainty is calibrated, we introduce two geometric criteria:

- The trace of the covariance matrix should provide the correct scale, i.e., the extent to which the obtained solution deviates from the exact. So the first measure is
$$c \equiv \|\boldsymbol{e}^n\|_2 \left(\sqrt{\operatorname{tr}\left(\boldsymbol{\Sigma}^n\right)}\right)^{-1}, \tag{5.59}$$
where $c > 1$ corresponds to the underestimation of the L_2 norm of error, $c < 1$ – to overestimation, and $c = 1$ – to the correct scale.

- The error should lie in the space of eigenvectors of the covariance matrix. More precisely, let $\{\boldsymbol{u}_i^n\}_{i=\overline{1,L}}$ be the set of eigenvectors of $\boldsymbol{\Sigma}^n$. We define a projector on the row space of the covariance matrix
$$\boldsymbol{V} \equiv \begin{pmatrix} | & & | \\ \boldsymbol{u}_1^n & \dots & \boldsymbol{u}_L^n \\ | & & | \end{pmatrix}, \; \boldsymbol{P}_{\boldsymbol{V}} \equiv \boldsymbol{V}\boldsymbol{V}^T. \tag{5.60}$$

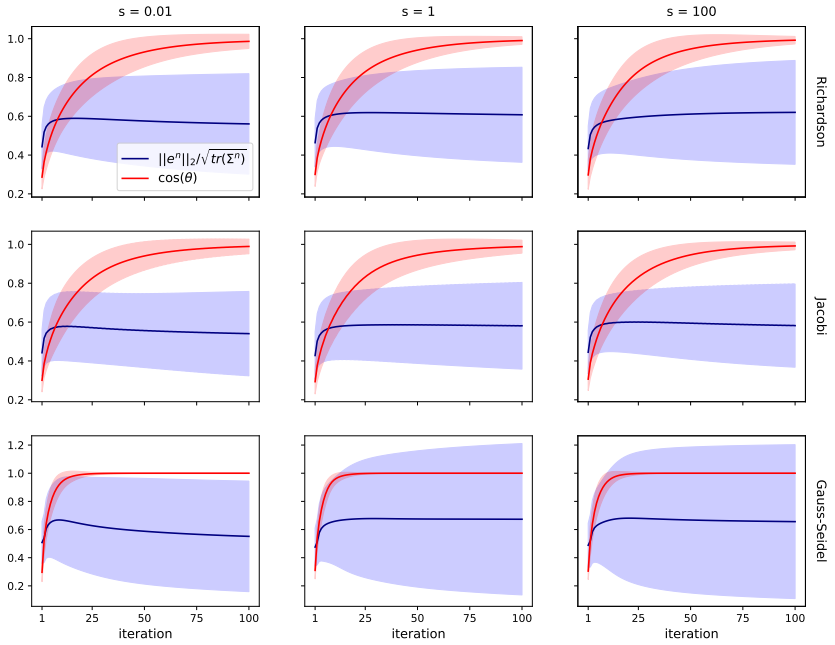


Figure 5.9: Results of uncertainty calibration (5.62) for random matrices (5.64). Thick lines represent empirical mean, shaded region covers (empirical mean) $\pm$ $\sqrt{\text{(empirical variance)}}$. Details are given after (5.64).

Now, a good way to measure how well the error can be represented by L available vectors is

$$\cos(\theta) \equiv \left|(\boldsymbol{e}^n)^T \boldsymbol{P}_{\boldsymbol{V}} \boldsymbol{e}^n\right| / \|\boldsymbol{e}^n\|_2^2 . \tag{5.61}$$

The closer $\cos(\theta)$ to 1, the better span $\left(\{\boldsymbol{u}_i^n\}_{i=\overline{1,L}}\right)$ is aligned with the error vector. Note, that for $L \sim K$ this requirement is trivial, but for $L \ll K$ it is hard to expect to have $\cos(\theta) \sim 1$ unless the vectors are tuned in a special way.

In Section 5.4.2, we proved a few asymptotic results. However, for the design of a well-calibration method the transient dynamics is more relevant. To avoid a complicated pseudospectrum analysis, we simplify the iterations for $\boldsymbol{\Sigma}^n$ by taking $\widetilde{\boldsymbol{\Sigma}}_n = \widetilde{\boldsymbol{\Psi}}_n = 0$, $n \geq L$ starting from some small iteration number L. Proposition 5.4.8 guarantees that in this situation the cosine of the acute angle between subspace spanned by eigenvectors of covariance matrix $\boldsymbol{\Sigma}^n$ and error vector $\boldsymbol{e}^n$ approaches zero for large n.

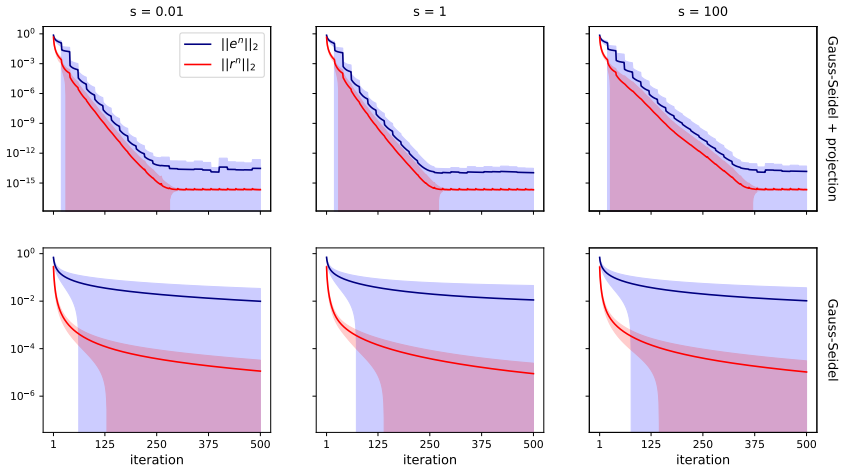


Figure 5.10: Acceleration of Gauss-Seidel iteration by repeated projections on the low-dimensional subspace of covariance matrix eigenvectors. Projection step is applied each 20 iterations. Average convergence factor for top row $\simeq 0.89$, and for bottom row $\simeq 0.99$. Matrices are drawn from (5.64), uncertainty calibration is (5.63). Thick lines represent the empirical mean, shaded regions cover (empirical mean) $\pm \sqrt{\text{(empirical variance)}}$. The dimension of the problem, number of samples and other details are given after (5.64).

The first scheme that we propose reads

$$\widetilde{\boldsymbol{\Sigma}}_0 = \sum_{i=1}^{L} \boldsymbol{\sigma}_i \boldsymbol{\sigma}_i^T, \;\; \widetilde{\boldsymbol{\Psi}}_0 = \alpha \boldsymbol{I}; \;\; \widetilde{\boldsymbol{\Sigma}}_j = \widetilde{\boldsymbol{\Psi}}_j = 0, \;\; \forall j > 0, \tag{5.62}$$

where $\boldsymbol{\sigma}_i = \boldsymbol{x}_i / \|\boldsymbol{x}_i\|_2$ and each component of x_i is drawn from standard normal distribution. The choice of α is discussed below.

For the second scheme, we take $\widetilde{\boldsymbol{\Psi}}_n \sim \boldsymbol{A}^{-T}\boldsymbol{A}^{-1}$ and approximate unknown term $\|\boldsymbol{e}^n\|_2^2$ by the rescaled residual:

$$\begin{aligned} &\widetilde{\boldsymbol{\Sigma}}_j = \sum_{i=1}^{L} \boldsymbol{r}^i \left(\boldsymbol{r}^i\right)^T / \left\|\boldsymbol{r}^i\right\|_2^2, \;\; j \le L; \;\; \widetilde{\boldsymbol{\Sigma}}_j = 0, \;\; \forall j > L; \\ &\boldsymbol{\Sigma}^{n+1} = \left[(\boldsymbol{I} - \boldsymbol{R}\boldsymbol{A}) \, \boldsymbol{\Sigma}^n \, (\boldsymbol{I} - \boldsymbol{R}\boldsymbol{A})^T + \frac{\widetilde{\boldsymbol{\Sigma}}_n}{\operatorname{rank} \boldsymbol{\Sigma}^{n+1}} \left(\operatorname{tr} \left(\boldsymbol{\Sigma}^n \right) + \alpha \left\| \boldsymbol{r}^n \right\|_2^2 \right) \right]_L . \end{aligned} \tag{5.63}$$

To fix α for both methods, we run each of them on homogeneous problem $\boldsymbol{A}\boldsymbol{x} = 0$ starting from a random initial guess $\boldsymbol{x}_0$. Since we know the exact solution of the

homogeneous equation, we can explicitly compute the error and estimate α from a few iterations. In experiments below, we use three iterations for (5.62) and 13 iterations for (5.63).

To evaluate the uncertainty quantification schemes (5.62) and (5.63) we sample random matrices

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^T, \tag{5.64}$$

where $\boldsymbol{Q}$ is uniformly distributed over $O(N)$, $D_{ij} = \delta_{ij} d_i$ and each d_i have p.d.f. $f(d) = s \exp(-d/s)$. Numerical results are presented on Section 5.4.5 and Section 5.4.3 for (5.62) and (5.63) respectively. The dimension of each problem is 100, for scales of exponential distribution $s \in \{0.01,\ 1,\ 100\}$ three classical iteration methods are used. Jacobi and Richardson are applied with the optimal relaxation parameter. To produce each subfigure, we average over 100 runs, rank L of the covariance matrix $\boldsymbol{\Sigma}^n$ is 10 in all experiments

Overall we can see that even simple uncertainty calibration schemes result in conservative estimation of L_2 error norm and correctly approximated low-dimensional subspace that captures error direction. These results persist for various condition numbers, iterative methods, and scales of A_{ij}. Scheme (5.62) results in better prediction of error norm, whereas (5.63) provides better directional information.

5.4.6 Acceleration of iteration by projection

As a rule, error estimations can be used to improve accuracy [Ske86]. Both uncertainty calibration strategies (5.62) and (5.63) estimate low-dimensional subspace $\operatorname{span}\{x : (\boldsymbol{I} - \boldsymbol{R}\boldsymbol{A})\,\boldsymbol{x} \simeq 0\}$ which can accurately approximate $\boldsymbol{e}^n$ for large n. So to exploit this approximation we resort to ordinary error correction scheme. That is, we perform a few iterations of basic iterative method and then use eigenvectors of covariance matrix to find the best error vector in the least square sense. Note that multigrid follows a similar strategy, but in place of a small number of dense vectors, a large number of sparse vectors is used on par with recursion. Section 5.4.5 shows the results of an error-correction scheme. As we can clearly see from this example, the other end of the probabilistic Richardson iteration is the acceleration of an underlying iterative method. Better uncertainty quantification will lead to a faster algorithm.

Part II

Machine learning

Chapter 6

Linear problems and machine learning

In this part we study how machine learning can aid numerical linear algebra. We can think of two principal objections one may raise against the use of machine learning in this context.

The first objection is that machine learning is typically applied in the situation when the problem at hands can not be formalized well in familiar mathematical terms, so the construction of the algorithm is not obvious. To give an example, object recognition problem (cats, dogs, cars, digits, etc) is notoriously difficult to formalize, yet neural networks manage to achieve human and superhuman performance on certain task of this kind [KSH12], [CMS12], [Rus+15]. In contrast to the hard-to-formalized problems, linear problem $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is formalized perfectly well. Moreover, well-known classical algorithm that can solve this problem (for example, Gaussian elimination) are readily available. So why use machine learning?

The problem is that direct methods are typically scales as $O(N^3)$, where N is a number of unknown. On the other hand iterative methods can potentially have $O(N)$ complexity for sparse linear problems. The problem is, the construction of efficient iterative method requires a specification of error propagation matrix $\boldsymbol{M}$ (see Section 1.2.1) with small spectral radius or a good preconditioner for projection methods (see Section 1.2.2). Since there are no general recipes on the construction of those objects, researchers proposed a set of heuristics of different generality (geometric and algebraic multigrid Section 1.2.3, filterring, circulant and other preconditioners Section 7.1.2). On the other hand, for any given error propagation matrix $\boldsymbol{M}$ or preconditioner one can estimate the efficiency of the resulting iterative method. Performance measures of this kind (see Section 7.1.1, Section 7.1.2) can

serve as loss functions. This mean, machine learning approaches like optimization of hyperperameters [Li+17], Bayesian optimization [Sha+15], [Fra18], online learning and optimization [Sha+11], k-armed bandits [LS20] become relevant. Moreover, as was shown by [Gre+19a], [Luz+20] not merely the optimization and heuristic search but also the generalization is possible.

The second objection is that machine learning itself relies heavily on numerical linear algebra. Examples being PCA [Shl14], Gaussian processes [Ras03], ordinary least squares [Bis06a, Chapter 3], EM for Gaussian mixtures [Bis06a, Chapter 9], etc. That is, algorithm of numerical linear algebra are in some sense more elementary. So it is not obvious that one can benefit from machine learning when it comes to underlying linear algebra algorithms.

The second objection forces us to give up on attempts to use machine learning to solve linear problem. In place of that we use machine learning to construct an algorithm, i.e., solver, itself. The example of a general endeavour of this kind would be program synthesis [GPS+17]. Since our goal is more modest, we typically form a parametric family of solvers or preconditioners and are trying to learn optimal parameters with machine learning approaches. This way, machine learning is used on the preparation step. So ones the algorithm is prepared, it can be used to solve linear problem, or family of linear problem with to additional computations compare to standard iterative methods.

The rest of the present part provides more examples of how machine learning can be used in numerical linear algebra. Namely, we start in Chapter 7 with a general explanation of unsupervised training suitable for the construction of solvers and preconditioners for numerical linear algebra. The whole endeavor is based on the introduction of appropriate stochastic losses and the minimization of them with gradient-based methods. That is, we apply classical approaches from machine learning to the construction of preconditioners. The models for preconditioners under the study are given by parametric families of generalized BPC multilevel preconditioners with overall architectures resembling U-Net. Next, in Chapter 8 we study the connection between multigrid and neural networks. In this chapter, we apply techniques from Chapter 7 to access the generalization capabilities of proposed approaches. That is, we show that one can train (perform optimization) on problems with a small number of unknowns and later use the same method on problems with a large number of unknowns without the loss in performance. Finally, in Chapter 9 we turns to the online setup. Namely, we show that it is possible to improve the iterative method on the flight using auxiliary information available as a byproduct of iterations. This is done with a help of k-armed bandits and Bayesian optimization.

Chapter 7

Black-box optimization of BPX preconditioners

7.1 Automatic construction of preconditioners and solvers

In this part, we consider two optimization problems. The first one is the optimization of a parametric family of preconditioners for a modified Richardson method applied to the matrix $\boldsymbol{A} : \boldsymbol{A}^T + \boldsymbol{A} > 0$, that is,

$$\omega_{\text{opt}},\ \theta_{\text{opt}} = \arg\min_{\omega,\theta} \rho\left(\boldsymbol{I} - \theta \boldsymbol{B}(\boldsymbol{A}, \omega)\right), \tag{7.1}$$

where $\boldsymbol{B}(\boldsymbol{A}, \omega) = \boldsymbol{B}(\omega)\boldsymbol{A}$ (or $\boldsymbol{B}(\boldsymbol{A}, \omega) = \boldsymbol{B}(\omega)\boldsymbol{A}\boldsymbol{B}(\omega)$) is a family of linear systems preconditioned from the left (or in a symmetric fashion), ρ is a spectral radius, and ω is a set of real numbers. Problem (7.1) corresponds to a direct optimization of asymptotic convergence speed of a preconditioned Richardson method [Hac16, Section 2.2.5]. It is known that arbitrary linear iterative method can be considered as a preconditioned Richardson iteration [Hac16, Proposition 5.44.]. So problem (7.1) can be seen an optimization of convergence speed of arbitrary linear iteration.

The second related problem is the optimization of the condition number

$$\omega_{\text{opt}} = \arg\min_{\omega} \lambda_{\max}\left(\boldsymbol{B}(\boldsymbol{A}, \omega)\right) / \lambda_{\min}\left(\boldsymbol{B}(\boldsymbol{A}, \omega)\right), \tag{7.2}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the smallest and the largest eigenvalues, and $\boldsymbol{A}$ is symmetric positive definite.

In both problems we follow the approach adopted in [KDO20] and further generalized in [Gre+19b], [Luz+20]. That is, we introduce a stochastic loss function

that approximates an objective function – spectral radius or a condition number – and perform a direct gradient-based optimization. The details can be found in Section 7.1.1 and Section 7.1.2.

For $\boldsymbol{B}(\omega)$ we use a modified BPX [BPX90] preconditioner. General multilevel preconditioner operates on a chain of linear spaces $V_1 \subset V_2 \subset \cdots \subset V_L$, where $V_l,\ 1 \leq l \leq L$ is formed as a linear combination of the set of functions $\phi_k^l(x),\ k = 1, \ldots, N_l$. In the context of a finite element method, $\phi_k^l(x)$ is a tent function located at vertex k of a grid with the diameter of a cell $\simeq$ const 2^{-l} (grid corresponding to V_{l+1} is constructed from l-th grid by, for example, subdivision of coarse triangulation, see i.e. [Zha92, Section 2]). BPX preconditioners were developed for an elliptic problem

$$-\sum_{i,j=1}^{D} \frac{\partial}{\partial x_i} a_{ij}(x) \frac{\partial}{\partial x_j} u(x) = f(x), \tag{7.3}$$

with homogeneous Dirichlet boundary conditions and uniformly symmetric positive definite $a_{ij}(x)$. For equation (7.3) and a nested set of finite element spaces span $\left\{\phi_k^l :\ k = 1, \ldots, N_l\right\}$, original BPX and preconditioner reads

$$B_{\mathrm{BPX}}(\omega)[v] = \sum_{l=1}^{L} \sum_{k=1}^{N_l} \left(v, \phi_k^l\right) \phi_k^l, \tag{7.4}$$

where $(\psi, \chi) = \int \psi(x)\chi(x)dx$ is a L_2 scalar product. To introduce additional parameters to BPX preconditioner and later use them in optimization of condition number we replace tent function with empirical basis functions $\widetilde{\phi}_k^l,\ l = 1, \ldots, L-1$ and introduce scalars $\widetilde{\alpha}_l,\ l = 1, \ldots, L-1$ that weight contributions from individual spaces V_l, that is

$$B_{\mathrm{BPX}}(\omega)[v] = \sum_{l=1}^{L} \widetilde{\alpha}_l \left(\sum_k \left(v, \widetilde{\phi}_k^l\right) \widetilde{\phi}_k^l \right). \tag{7.5}$$

The details of the parametrisation and more convenient form of preconditioners (7.5) are given in Section 7.2.

Together $\widetilde{\phi}_k^l$ and $\widetilde{\alpha}_l$ form a set of parameters ω in problems (7.1), (7.2). The results of the optimization can be found in Section 7.3.

7.1.1 Direct optimization of spectral radius

Problem (7.1) can be viewed in the context of a general search for better linear iterative methods. As explained in Section 1.2.1, an arbitrary consistent iterative

method can be written in a form

$$\boldsymbol{x}^{n+1} = \boldsymbol{M}(\omega, \boldsymbol{A})\boldsymbol{x}^n + \boldsymbol{N}(\omega, \boldsymbol{A})\boldsymbol{b},\ \boldsymbol{I} - \boldsymbol{M}(\omega, \boldsymbol{A}) = \boldsymbol{N}(\omega, \boldsymbol{A})\boldsymbol{A}. \tag{7.6}$$

The efficiency of the method can be characterised by spectral radius $\rho\left(\boldsymbol{M}(\omega, \boldsymbol{A})\right)$, because it quantifies an asymptotic convergence rate in a following sense. Let $\boldsymbol{e}^n$ be an error vector on step n, $\|\cdot\|$ is arbitrary norm and $\rho_{m+k,m} = \left(\left\|\boldsymbol{e}^{m+k}\right\| / \left\|\boldsymbol{e}^m\right\|\right)^{1/k}$ is a geometric mean of a one-step error reduction factor $\rho_{m+1,m}$. It is known that $\lim_{k\to\infty} \max_{\boldsymbol{x}_0} \left\{\rho_{m+k,m}(\boldsymbol{x}_0)\right\} = \rho(\boldsymbol{M}(\omega, \boldsymbol{A}))$ (see [Hac94, Remark 2.22]). That is, $\rho\left(\boldsymbol{M}(\omega, \boldsymbol{A})\right)$ characterises a geometric mean of an error reduction per iteration in the worst case. Because of that it is a custom to use $\rho\left(\boldsymbol{M}(\omega, \boldsymbol{A})\right)$ as an objective function. For example, classical schemes like SOR and instationary Richardson iteration were optimized analytically [Had00], [Hac94, chapters 4, 8] and numerically [Man78], [Rei66], to achieve better $\rho\left(\boldsymbol{M}(\omega, \boldsymbol{A})\right)$. More modern attempts include optimization of multigrid with local Fourier analysis [Bro+20] and directly [SKK19], [Luz+20], [Gre+19b], [KDO20].

To apply gradient-based optimization to (7.1) we need a differentiable approximation to the spectral radius. We consider three options.

The first one is an approximation of $\rho(A)$ by Gelfand formula [Koz09] $\rho(\boldsymbol{A}) = \lim_{k\to\infty} \left\|\boldsymbol{A}^k\right\|^{1/k}$ combined with a stochastic trace approximation [AT11]:

$$\begin{gathered}\rho(\boldsymbol{A}) \simeq \rho_1(\boldsymbol{A}, k, N_{\text{batch}}) \equiv \left(\frac{1}{N_{\text{batch}}} \sum_{j=1}^{N_{\text{batch}}} \left\|\boldsymbol{A}^k \boldsymbol{z}_j\right\|_2^2\right)^{1/2k}, \\ \forall j : \mathbb{P}\left((\boldsymbol{z}_j)_i = \pm 1\right) = 1/2,\ \forall i, j :\ \boldsymbol{z}_i, \boldsymbol{z}_j \text{ are independent.}\end{gathered} \tag{7.7}$$

More details about this approach can be found in [KDO20].

The second option is based on $\rho(\boldsymbol{A}) = \lim_{k\to\infty} \left(\left\|\boldsymbol{e}^{m+k}\right\| / \left\|\boldsymbol{e}^m\right\|\right)^{1/k}$, $\boldsymbol{e}^{m+l} = \boldsymbol{A}^l \boldsymbol{e}^m$, see [Hac94, Remark 2.22 (b)] for details. This gives us another approximation

$$\rho(\boldsymbol{A}) \simeq \rho_2\left(\boldsymbol{A}, k\right) \equiv \left(\left\|\boldsymbol{A}^k \boldsymbol{z}\right\|_2 / \left\|\boldsymbol{z}\right\|_2\right)^{1/k},\ (\boldsymbol{z}_i)_j \sim \mathcal{N}(0, 1). \tag{7.8}$$

Approximation (7.8) does not contain averaging, but we can introduce N_{batch} the same way as in (7.7). That gives us the following the last approximation

$$\begin{gathered}\rho(\boldsymbol{A}) \simeq \rho_3\left(\boldsymbol{A}, k, N_{\text{batch}}\right) \equiv \frac{1}{N_{\text{batch}}} \sum_{j=1}^{N_{\text{batch}}} \left(\left\|\boldsymbol{A}^k \boldsymbol{z}_j\right\|_2 / \left\|\boldsymbol{z}_j\right\|_2\right)^{1/k}, \\ \forall j : (\boldsymbol{z}_j)_i \sim \mathcal{N}(0, 1),\ \forall i, j :\ \boldsymbol{z}_i, \boldsymbol{z}_j \text{ are independent.}\end{gathered} \tag{7.9}$$

Algorithm 8 Minimization of L_1 (7.10).

Input: matrix $\boldsymbol{A} > 0$, parametric family of preconditioners $\boldsymbol{B}(\boldsymbol{A}, \omega) : \boldsymbol{B}(\boldsymbol{A}, \omega) > 0$, stochastic gradient-based optimizer $\omega \leftarrow O\left(\omega, \partial_\omega \left(\text{loss function}\right)\right)$ (f.e., ADAM, [KB14]), batch size N_{batch}, number of matrix-vector products k, number of epochs N_{epochs}, number of iterations for inner loop N_{inner}, estimator of the spectral radius $m \in \{1, 2, 3\}$.

---

for $i = 1 : N_{\text{epochs}}$ **do**
  for $j = 1 : N_{\text{inner}}$ **do**
    $\rho_m, \partial_\theta \rho_m \leftarrow \text{AD } \rho_m \left(\boldsymbol{I} - \theta \boldsymbol{B}(\boldsymbol{A}, \omega), k, N_{\text{batch}}\right)$ // AD – automatic differentiation
    $\theta \leftarrow O\left(\theta, \partial_\theta \rho_m\right)$
  end for
  $L_1, \partial_\omega L_1 \leftarrow \text{AD } \rho_m \left(\boldsymbol{I} - \theta \boldsymbol{B}(\boldsymbol{A}, \omega), k, N_{\text{batch}}\right)$
  $\omega \leftarrow O\left(\omega, \partial_\omega L_1\right)$
end for

The resulting loss will measure how well matrix $\boldsymbol{A}$ damps nonzero initial vectors on average. We observed that introduction of $N_{\text{batch}} > 1$ in (7.9) leads to better convergence.

With approximations $\rho_i \left(\boldsymbol{A}, k, N_{\text{batch}}\right),\ i = 1, 2, 3$ we can use forward mode automatic differentiation [RLP16] and standard optimizers [Goo+16, Section 8.3] to solve problem (7.1). The resulting algorithm coincides with Algorithm 8 with $N_{\text{inner}} = 1$.

7.1.2 Direct optimization of spectral condition number

Unlike problem (7.1) the optimization of the condition number is not straightforward. The main problem is the presence of $\lambda_{\min}$ which is not readily available. The standard way to resolve this issue is to substitute spectral radius with more amenable loss. For example, objective functions $\|\boldsymbol{R} - \boldsymbol{A}\|$ and $\left\|\boldsymbol{I} - \boldsymbol{R}^{-1}\boldsymbol{A}\right\|$ (here $\boldsymbol{R}$ is an easy invertible approximation to $\boldsymbol{A}$) were used to construct optimal circulant [Cha88], [Tyr92], [Str86] and sparse approximate inverse [GH97], [CS98] preconditioners. It is known that for nonsymmetric matrices optimization of $\left\|\boldsymbol{I} - \boldsymbol{R}^{-1}\boldsymbol{A}\right\|$ can fail to deliver good preconditioner [CS94]. The same is true for symmetric positive definite matrices as illustrated on Fig. 7.1.

For symmetric positive definite matrices, one can construct a loss function that leads to a direct minimization of the spectral condition number. It is well known

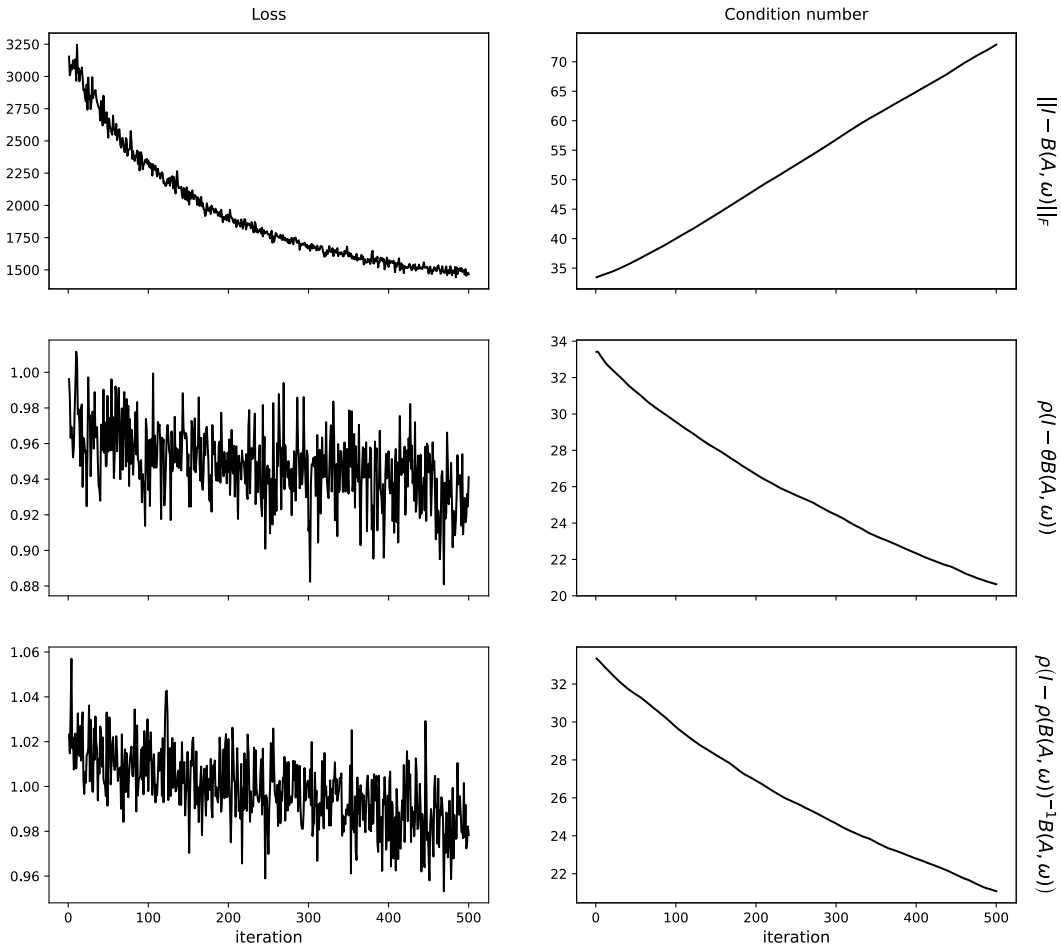


Figure 7.1: Comparison of three loss functions. The first column shows how the value of the loss function changes in the course of iterations, graphs in the second column demonstrate an evolution of condition number. The first row corresponds to the Frobenius norm $\|\boldsymbol{I} - \boldsymbol{B}(\omega)\boldsymbol{A}\|$ used as a loss function, the second row shows minimization of L_1 by Algorithm 8 ($N_{\text{inner}} = 1$), the last row shows minimization of L_2 by Algorithm 9. For the last two cases, we used (7.7) to approximate spectral radius. It is clear that the decrease of both losses L_1 and L_2 lead to a smaller spectral condition number, whereas smaller Frobenius norm does not lead to a better spectral condition number. In all cases we use modified BPX preconditioner (7.15) as $B(\omega)$ and FEM discretization (see Section 7.2) of Poisson equation Section 1.3.3 in $D = 1$.

that for arbitrary positive definite matrix $\boldsymbol{C}$, optimal spectral radius of $\boldsymbol{I} - \theta\boldsymbol{C}$ is $(\lambda_{\max}(\boldsymbol{C}) - \lambda_{\min}(\boldsymbol{C})) \,/\, (\lambda_{\max}(\boldsymbol{C}) + \lambda_{\min}(\boldsymbol{C}))$. Using this fact, we can consider the following loss function

$$L_1(\omega) = \rho\left(\boldsymbol{I} - \theta_{\text{opt}}(\omega)\boldsymbol{B}(\boldsymbol{A}, \omega)\right), \;\; \theta_{\text{opt}}(\omega) = \arg\min_{\theta} \rho\left(\boldsymbol{I} - \theta\boldsymbol{B}(\boldsymbol{A}, \omega)\right). \tag{7.10}$$

Evidently, the minimization of (7.10) is equivalent to the minimization of the expression $(\kappa(\boldsymbol{B}(\boldsymbol{A}, \omega)) - 1) \,/\, (\kappa(\boldsymbol{B}(\boldsymbol{A}, \omega)) + 1)$, where κ is the spectral condition number. That means we constructed an optimization problem equivalent to (7.2) but without $\lambda_{\min}$. A procedure for minimization of loss (7.10) is summarised in Algorithm 8. The inner loop finds θ_{opt} for each ω and the outer loop optimizes ω. If an inner loop is reduced to a single iteration as it is done in many other situations (for example, generalized policy iteration [SB98, Section 4.6], and full approximation scheme [TOS00, Section 5.3.1] follow the same pattern), we obtain an algorithm that minimizes spectral radius for modified Richardson scheme.

Another equivalent loss function is

$$L_2(\omega) = \rho\left(\boldsymbol{I} - \frac{1}{\rho(\boldsymbol{B}(\boldsymbol{A}, \omega))}\boldsymbol{B}(\boldsymbol{A}, \omega)\right). \tag{7.11}$$

Indeed, $\rho\left(\boldsymbol{I} - \boldsymbol{B}(\boldsymbol{A}, \omega)/\rho(\boldsymbol{B}(\boldsymbol{A}, \omega))\right) = 1 - \lambda_{\min}(\boldsymbol{B}(\boldsymbol{A}, \omega))/\lambda_{\max}(\boldsymbol{B}(\boldsymbol{A}, \omega))$, which means that a minimization of (7.11) is equivalent to minimization of $1 - 1/\kappa(\boldsymbol{B}(\boldsymbol{A}, \omega))$. Gradient-based optimization can be applied to (7.11) directly, but we can exploit a special structure of the problem to shorten the computation graph. Using a chain rule we get

$$\begin{aligned}\frac{\partial}{\partial\omega_i}\rho\left(\boldsymbol{I} - \frac{1}{\rho(\boldsymbol{B}(\boldsymbol{A}, \omega))}\boldsymbol{B}(\boldsymbol{A}, \omega)\right) &= \left(\frac{\partial}{\partial\omega_i}\rho\left(\boldsymbol{I} - \theta\boldsymbol{B}(\boldsymbol{A}, \omega)\right)\right)\Big|_{\theta=\rho(\boldsymbol{B}(\boldsymbol{A},\omega))^{-1}} \\ - \left(\theta^2\frac{\partial}{\partial\theta}\rho\left(\boldsymbol{I} - \theta\boldsymbol{B}(\boldsymbol{A}, \omega)\right)\right)\Big|_{\theta=\rho(\boldsymbol{B}(\boldsymbol{A},\omega))^{-1}} &\frac{\partial}{\partial\omega_i}\rho(\boldsymbol{B}(\boldsymbol{A}, \omega)).\end{aligned} \tag{7.12}$$

This leads to Algorithm 9. The performance of these two loss function is illustrated on Fig. 7.1. In our experiments, we find little difference between Algorithm 8 and Algorithm 9. Because of that, we mainly use Algorithm 8, which requires a single computation of a gradient with respect to ω. However, unlike L_1 loss function L_2 is defined in terms of ρ in closed form, i.e., without an additional optimization problem, so it can be more advantageous in situations when a family of preconditioners is learned for a set of related linear equations, as it is done in [Gre+19b] for the multigrid solver.

We summarize the results of this section in the following statement.

Algorithm 9 Minimization of L_2 (7.11).

Input: symmetric positive definite matrix $\boldsymbol{A} > 0$, parametric family of preconditioners $\boldsymbol{B}(\boldsymbol{A}, \omega) : \boldsymbol{B}(\boldsymbol{A}, \omega) > 0$, stochastic gradient-based optimizer $\omega \leftarrow O\left(\omega, \partial_\omega \left(\text{loss function}\right)\right)$ (f.e., ADAM, [KB14]), batch size N_{batch}, number of matrix-vector products k, number of epochs N_{epochs}, estimator of the spectral radius $m \in \{1, 2, 3\}$.

---

for $i = 1 : N_{\text{epochs}}$ **do**
  $\theta \leftarrow 1/\rho_1\left(\boldsymbol{B}(\boldsymbol{A}, \omega), k, N_{\text{batch}}\right)$
  $\rho_m, \partial_\omega \rho_m \leftarrow \text{AD } \rho_m\left(\boldsymbol{B}(\boldsymbol{A}, \omega), k, N_{\text{batch}}\right)$ // AD – automatic differentiation
  $L_2, \partial_\omega L_2, \partial_\theta L_2 \leftarrow \text{AD } \rho_m\left(\boldsymbol{I} - \theta \boldsymbol{B}(\boldsymbol{A}, \omega), k, N_{\text{batch}}\right)$
  $\omega \leftarrow O\left(\omega, \partial_\omega \rho_m - \theta^2 \partial_\theta L_2 \partial_\omega L_2\right)$
end for

Proposition 7.1.1. *Let $\boldsymbol{A} > 0$ and $\boldsymbol{B}(\omega) > 0$ for all ω. For left $\boldsymbol{B}(\boldsymbol{A}, \omega) = \boldsymbol{B}(\omega)\boldsymbol{A}$, symmetric $\boldsymbol{B}(\boldsymbol{A}, \omega) = \boldsymbol{B}(\omega)\boldsymbol{A}\boldsymbol{B}(\omega)$ and right $\boldsymbol{B}(\boldsymbol{A}, \omega) = \boldsymbol{A}\boldsymbol{B}(\omega)$ preconditioners the following three optimization problems are equivalent:*

- $\min_\omega \rho\left(\boldsymbol{I} - \theta_{opt}(\omega)\boldsymbol{B}(\boldsymbol{A}, \omega)\right)$, *where* $\theta_{opt}(\omega) = \arg\min_\theta \rho\left(\boldsymbol{I} - \theta \boldsymbol{B}(\boldsymbol{A}, \omega)\right)$ – *loss function* (7.10)
- $\min_\omega \rho\left(\boldsymbol{I} - \boldsymbol{B}(\boldsymbol{A}, \omega)/\rho(\boldsymbol{B}(\boldsymbol{A}, \omega))\right)$ – *loss function* (7.11)
- $\min_\omega \left(\lambda_{\max}(\boldsymbol{B}(\boldsymbol{A}, \omega))/\lambda_{\min}(\boldsymbol{B}(\boldsymbol{A}, \omega))\right)$

7.2 Modified BPX preconditioners

We already specified algorithms that can be used to optimize condition number (optimization problem (7.2)). In this section, we describe a parametric family of positive definite preconditioners that we use in optimization.

To obtain a convenient form of BPX preconditioner, we introduce a hierarchy of meshes

$$M_l = \left\{x_j^l = j/2^l : j = 0, 1, \ldots, 2^l - 1, 2^l\right\}, \; l = 1, \ldots, L \tag{7.13}$$

such that each next mesh contains a previous one, that is, $M_l \subset M_{l+1}$. For each mesh, we define a set of basis functions $\phi_i^l(x) = \phi^l(x - x_i), i = 0, \ldots, 2^l$, which are rescaled and translated copies of a tent function $\phi^l(x) = \left(1 + x/2^l\right) \text{Ind}\left[-1/2^l \le x \le 0\right] + \left(1 - x/2^l\right) \text{Ind}\left[0 < x \le 1/2^l\right]$, where $\text{Ind}\left[x\right]$ is 1 if x holds and 0 otherwise. Basis

functions $\left\{\phi_i^L(x) : i = 0, \ldots, 2^L\right\}$ are used to perform standard finite element discretization [Cia02] of elliptic problem (7.3) for $x \in [0, 1]$. For higher dimensions, we use M_l and ϕ_i^l that are direct products of unidimensional meshes and basis functions.

In article [BK20], authors show that for equation (7.3) in $D = 1$ with uniform Dirichlet boundary condition at $x = 0$ and uniform Neumann boundary condition at $x = 1$ discretized as we just described, BPX preconditioner has the following form

$$\begin{gathered}\boldsymbol{\mathcal{B}} = \sum_{k=1}^{L} \alpha_k \boldsymbol{B}_k^L \boldsymbol{B}_L^k,\ \boldsymbol{B}_l^L = \boldsymbol{I}_l \otimes \boldsymbol{\eta}_{L-l} + \boldsymbol{S}_l \otimes \left(\boldsymbol{\xi}_{L-l} - \boldsymbol{\eta}_{L-l}\right), \boldsymbol{B}_L^l = \left(\boldsymbol{B}_l^L\right)^T,\ \alpha_k = 1 \\ (\boldsymbol{\eta}_k)_i = i/2^k, (\boldsymbol{\xi}_k)_i = 1,\ (\boldsymbol{S}_l)_{ij} = \delta_{ij+1}, (\boldsymbol{I}_l)_{ij} = \delta_{ij},\ i, j = 1,\ \ldots\ ,\ 2^l.\end{gathered} \tag{7.14}$$

If $D = 2$ matrices $\boldsymbol{B}_L^k$ are replaced with $\boldsymbol{B}_L^k \otimes \boldsymbol{B}_L^k$ and α_k are with ratio of grid spacings h_L/h_k. The proof of the optimality of symmetric preconditioner (7.14) can be found in [BK20, Appendix A].

It is easy to see that components of $\boldsymbol{\eta}_{L-l}$ and $\boldsymbol{\xi}_{L-l} - \boldsymbol{\eta}_{L-l}$ contains scalar products $\left(\phi^L, \phi^l\right)$. Using this observation, one can extend (7.14) on other boundary conditions:

Proposition 7.2.1. *For equation* (7.3) *in* $D = 1$ *discretized with linear finite elements, symmetric BPX preconditioner has a form* $\boldsymbol{\mathcal{B}} = \sum_{k=1}^{L} \alpha_k \boldsymbol{B}_k^L \boldsymbol{B}_L^k$, *where matrices* $\boldsymbol{B}_L^k$ *depend on boundary conditions as follows:*

- *Dirichlet-Neumann:* $\boldsymbol{B}_l^L = \boldsymbol{I}_l \otimes \boldsymbol{\eta}_{L-l} + \boldsymbol{S}_l \otimes \left(\boldsymbol{\xi}_{L-l} - \boldsymbol{\eta}_{L-l}\right)$;
- *Neumann-Dirichlet:* $\boldsymbol{B}_l^L = \boldsymbol{I}_l \otimes \boldsymbol{\eta}_{L-l}^r + (\boldsymbol{S}_l)^T \otimes \left(\boldsymbol{\xi}_{L-l} - \boldsymbol{\eta}_{L-l}^r\right),\ (\boldsymbol{\eta}_k^r)_i = (\boldsymbol{\eta}_k)_{2^k - i + 1}$;
- *Neumann-Neumann:* $\boldsymbol{B}_l^L = \begin{pmatrix} 1 & 0_{1\times 2^l} \\ \boldsymbol{e}_l \otimes \left(\boldsymbol{\xi}_{L-l} - \boldsymbol{\eta}_{L-l}\right) & \boldsymbol{I}_l \otimes \boldsymbol{\eta}_{L-l} + \boldsymbol{S}_l \otimes \left(\boldsymbol{\xi}_{L-l} - \boldsymbol{\eta}_{L-l}\right) \end{pmatrix}$;
- *Dirichlet-Dirichlet:* $\boldsymbol{B}_l^L = \left[\boldsymbol{I}_l \otimes \boldsymbol{\eta}_{L-l} + \boldsymbol{S}_l \otimes \left(\boldsymbol{\xi}_{L-l} - \boldsymbol{\eta}_{L-l}\right)\right]_{\textit{last row and column are removed}}$.

All boundary conditions are uniform and vectors $\boldsymbol{\xi}_{L-l}, \boldsymbol{\eta}_{L-l}$ *are defined as in* (7.14).

Based on (7.14) and Proposition 7.2.1, we put forward the following parametrization

$$\widetilde{\boldsymbol{\mathcal{B}}} = \sum_{k=1}^{L} (\widetilde{\alpha}_k)^2\, \widetilde{\boldsymbol{B}}_k^L \widetilde{\boldsymbol{B}}_L^k,\ \widetilde{\boldsymbol{B}}_l^L = \boldsymbol{I}_l \otimes \widetilde{\boldsymbol{\eta}}_{L-l} + \boldsymbol{S}_l \otimes \widetilde{\boldsymbol{\xi}}_{L-l}, \left(\widetilde{\boldsymbol{\xi}}_{L-l}\right)_{2^l} = 0,\ \widetilde{\eta}_0 = 1,\ \widetilde{\alpha}_L = 1, \tag{7.15}$$

where $\widetilde{\boldsymbol{\alpha}}_k, \widetilde{\boldsymbol{\eta}}_{L-k}$ and $\widetilde{\boldsymbol{\xi}}_{L-k}$ are free parameters that correspond to ω in Algorithm 8 and Algorithm 9. Chosen parametrization differs from (7.14) in two respects. First, we use $\widetilde{\boldsymbol{\xi}}_{L-k}$ in place of $\widetilde{\boldsymbol{\xi}}_{L-k} - \widetilde{\boldsymbol{\eta}}_{L-k}$. Since both $\widetilde{\boldsymbol{\eta}}_{L-k}$ and $\widetilde{\boldsymbol{\xi}}_{L-k}$ are free parameters, both options lead to the same family of preconditioners. Second, we use $(\widetilde{\alpha}_k)^2$ in place of $\widetilde{\alpha}_k$. This choice among with conditions $\widetilde{\boldsymbol{\eta}}_0 = 1$ and $\widetilde{\alpha}_L = 1$ guarantee that $\widetilde{\boldsymbol{\mathcal{B}}}$ is positive definite regardless of the choice of other parameters. Indeed, $\widetilde{\boldsymbol{\mathcal{B}}}$ has a form $\boldsymbol{I} + \sum_{k=1}^{L-1} (\widetilde{\alpha}_k)^2 \left(\widetilde{\boldsymbol{B}}_L^k\right)^T \widetilde{\boldsymbol{B}}_L^k$, that is, the sum of positive definite and positive semidefinite matrices. Because of that, conditions of Proposition 7.1.1 apply and we can use parametric family (7.15) to optimize condition number with Algorithm 8 and Algorithm 9. The last condition $\left(\widetilde{\boldsymbol{\xi}}_{L-l}\right)_{2^l} = 0$ ensures that basis functions on level l have the same support as the ordinary tent functions.

7.3 Numerical examples

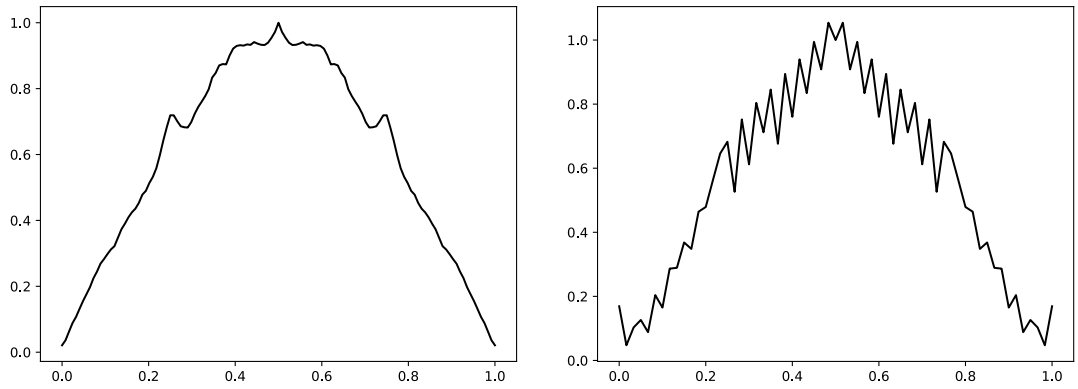
Here we present the results of the optimization for a set of test problems. For all equations we use a symmetric form of both BPX (7.14) and modified BPX (7.15) preconditioners. To access the results of optimization we list three related numbers: $\rho = \lambda_{\max}\left(\boldsymbol{I} - \theta_{\text{opt}}\boldsymbol{BAB}\right)$ – a spectral radius of the optimal Richardson iteration for a given preconditioner, $\kappa = \lambda_{\max}(\boldsymbol{BAB})/\lambda_{\min}(\boldsymbol{BAB})$ – spectral condition number, and N – the number of iteration needed to drop an error by 0.1 with the optimal modified Richardson iteration in an arbitrary chosen norm, i.e., $\left\|\boldsymbol{e}^{n+N}\right\| / \left\|\boldsymbol{e}^n\right\| \leq 0.1$. The number of iterations N is computed as $\lceil -1/\log_{10}\rho \rceil$, where $\lceil\cdot\rceil$ is the ceiling function. [1]

In all cases, we use Dirichlet boundary conditions. Value of L fixes the number of points along each direction to be $2^L - 1$.

For all examples we employed Algorithm 8 with the loss function (7.9) ($N_{\text{batch}} = 10$, $k = 10$), ADAM optimizer [KB14], $N_{\text{epoch}} = 500$, $N_{\text{inner}} = 1$. Initial parameters $\widetilde{\alpha}, \widetilde{\boldsymbol{\eta}}, \widetilde{\boldsymbol{\xi}}$ of the modified BPX preconditioner (7.15) were chosen such that the resulting matrix $\widetilde{\boldsymbol{\mathcal{B}}}$ coincides with the BPX preconditioner (7.14).

All algorithms were implemented in Julia [Bez+17] and available in a public repository `https://github.com/VLSF/neuralBPX`. In Fig. 7.3, Fig. 7.4, Fig. 7.5 one can find $\rho = \lambda_{\max}\left(I - \theta_{\text{opt}}BAB\right)$ – a spectral radius of optimal Richardson iteration for a given preconditioner, $\kappa = \lambda_{\max}(BAB)/\lambda_{\min}(BAB)$ – spectral condition number,

[1] This definition of N guarantees $\left\|\boldsymbol{e}^{n+N}\right\| / \left\|\boldsymbol{e}^n\right\| \leq 0.1$ for normal iteration matrix $\boldsymbol{M}(\omega, \boldsymbol{A})$. If $\boldsymbol{M}(\omega, \boldsymbol{A})$ is not normal, N holds as an estimation (see the discussion in Section 7.1.1 after equation (7.6)).



(a) Basis function (b) Basis function

Figure 7.2: Optimal basis functions for (a) – 1D Poisson equation (1.12), and (b) – biharmonic equation (1.25)

and N – the number of iteration needed to drop an error by 0.1 in the arbitrary norm, i.e., $\left\|e^{n+N}\right\| / \left\|e^{n}\right\| \leq 0.1$. Examples of optimal basis functions are in Fig. 7.2.

7.3.1 Poisson equation

We can see on Table 7.3a that for the 2D Poisson equation optimization successfully decreases the condition number. Moreover, it seems to grow slower compared to the original BPX preconditioner as the number of points increases [2]. To assess the contribution of the optimized basis functions, we perform additional optimization in $D = 1$ with fixed basis functions. Results, given in Table 7.3c, indicate that optimization of the basis function leads to twice as small spectral radius compare to the situation when only scales are being optimized. The basis function itself is depicted in Fig. 7.2a. We can see that it is self-similar and seems to be well defined (in a sense that a subsampled basis function for $L_1 > L_2$ is a good basis function for L_2). We can deduce that this function is a limit of some subdivision scheme [Rio92], but we could not reliably define subdivision weights from our numerical experiments.

[2] To estimate the growth rate we fit data using ordinary least squares with the model $\kappa(L) = c_1 + c_2 L$. For BPX preconditioner $(c_1, c_2) = (0.792, 1.196)$, and for the optimized BPX $(c_1, c_2) = (1.164, 0.264)$.

7.3.2 Helmholtz equation

Table 7.4b contains the results for Helmholtz equation (1.23) with k^2h equal to 0.01 and 0.1. The results are similar to the one for the Poisson equation. However, if we further increase the number of points or k, the resulting matrix becomes indefinite, and the optimization breaks down. That means that with our approach, we cannot construct preconditioners for the Helmholtz equation. It is known that preconditioners for the Helmholtz equation significantly differ from preconditioners for Poisson-like equations (see [Erl08] for the review), so this result is not surprising.

7.3.3 Anisotrpoic Poisson equation

Table 7.4a contains the results for anisotropic Poisson equation (1.20) with ϵ equal to 10 and 100. To cope with the anisotropy, we apply semicoarsening [TOS00, Section 5.1]. Without semicoarsening a "projector" on the grid $M_k \times M_k$ (M_k is as in (7.13)) reads $\widetilde{B}_L^k \otimes \widetilde{B}_L^k$. For semicoarsening the hierarchy of grids is modified, that is, in place of $M_k \times M_k$ we project on $M_{\min(k-s,0)} \times M_k$, where s quantifies the extent to which the grid along one direction is denser than a grid in the other direction. With this modification, a preconditioner itself takes a form

$$\widetilde{\mathcal{B}}_s = \sum_{k=1}^{L} (\widetilde{\alpha}_k)^2 \left(\widetilde{B}^L_{\min(k-s,0)} \otimes \widetilde{B}^L_k \right) \left(\widetilde{B}_L^{\min(k-s,0)} \otimes \widetilde{B}_L^k \right). \tag{7.16}$$

As a result, the coarsening is delayed for y because $\epsilon > 1$ in (1.20), i.e., y is a direction of the strong coupling. Note that in Table 7.4a we compare (7.16) with original BPX preconditioner. If semicoarsening is applied to the BPX preconditioner, the weights α_k need to be modified. Original weights α_k combined with semicoarsening lead to worse performance. We can see that the optimization was able to fix the weights correctly. Moreover, comparing to semicoarsening applied in the context of filtering preconditioners [TCK92] we were able to perform more aggressive coarsening, i.e., to decrease the number of floating-point operations.

7.3.4 Biharmonic equation

Results for the biharmonic equation are given in Table 7.4c. We can see that the BPX preconditioner is relatively inefficient. It was able to substantially decrease the condition number compared to the original matrix (this condition number is not listed), but still, the condition number is large and grows like $\kappa_{L+1} \simeq 4\kappa_L$. Condition number for the optimized BPX preconditioner is not only smaller but grows like

$\kappa_{L+1} \simeq 2\kappa_L$. The basis function on Fig. 7.2b does not seem to be stable in this case. Authors in [TCK92] were managed to obtain a better preconditioner for the biharmonic equation using larger filters. The same applies to the case of multigrid solvers, where orders of interpolation n_i and restriction n_r operators should fulfill $n_i + n_r > n_l$ [TOS00, Remark 2.7.1], where n_l is the order of the linear operator (4 in the case of biharmonic equation). Given that, we can suggest that by increasing the basis function's support, one can achieve a better condition number. We will study this elsewhere.

7.3.5 Convection-diffusion equation

Convection-diffusion equation leads to a non-symmetric matrix. Because of this, we do not list spectral condition number in Table 7.3b. Here optimization results in about twice as efficient solver, but the improvement becomes less pronounced for larger convection coefficient values.

7.3.6 Diffusion with discontinuous coefficients

Because neither BPX nor modified BPX account for the variation of coefficients, we used a rescaled version of preconditioner

$$\widetilde{\mathcal{B}}_r = \sum_{k=1}^{L} (\widetilde{\alpha}_k)^2 \, \widetilde{B}_k^L D \left(B_L^k A B_k^L \right)^{-1/2} \widetilde{B}_L^k, \tag{7.17}$$

where $D(\cdot)$ denotes the diagonal part of the matrix. For the original BPX preconditioner we again insert a diagonal part in-between "projectors" and use α_k as in (7.14). Results are given in Table 7.5c. It is evident that it is enough to recover the correct scales $\widetilde{\alpha}_k$. This was achieved by optimization which produces a good preconditioner regardless of scale.

The other option would be to perform a Jacobi preconditioning step

$$\boldsymbol{A} \to \boldsymbol{D}(\boldsymbol{A})^{-1/2} \boldsymbol{A} \boldsymbol{D}(\boldsymbol{A})^{-1/2} \tag{7.18}$$

as explained in [Bre97, discussion after equation (5.2)] and (in relation to diffusion with discontinuous coefficients) in [Wat15, Section 3.1]. If this kind of rescaling is performed, BPX becomes a reasonable preconditioner, and optimization leads to results similar to the observed ones for the Poisson equation.

	Bilinear FEM						Mehrstellen					
	BPX			optimized BPX			BPX			optimized BPX		
L	ρ	κ	N	ρ	κ	N	ρ	κ	N	ρ	κ	N
3	.621	4.277	5	.314	1.915	2	.62	4.269	5	.427	2.488	3
4	.701	5.678	7	.386	2.259	3	.7	5.678	7	.448	2.621	3
5	.746	6.867	8	.432	2.523	3	.746	6.867	8	.454	2.666	3
6	.774	7.866	10	.46	2.706	3	.774	7.867	10	.472	2.791	4

(a) Poisson 2D, (1.12)

	$v_x = -v_y = 1/h$				$v_x = -v_y = 2/h$				$v_x = -v_y = 3/h$			
	BPX		optimized BPX		BPX		optimized BPX		BPX		optimized BPX	
L	ρ_3	N	ρ_3	N	ρ_3	N	ρ_3	N	ρ_3	N	ρ_3	N
3	.629	5	.398	3	.787	10	.574	5	.855	15	.693	7
4	.741	8	.554	4	.830	13	.711	7	.869	17	.743	8
5	.797	11	.649	6	.864	16	.737	8	.872	17	.785	10
6	.829	13	.690	7	.874	18	.743	8	.874	18	.792	10

(b) Convection-diffusion, (1.24)

	BPX			ϕ_i are fixed optimized BPX			optimized BPX		
L	ρ	κ	N	ρ	κ	N	ρ	κ	N
3	.611	4.138	5	.483	2.866	4	.332	1.994	3
4	.696	5.58	7	.554	3.484	4	.357	2.109	3
5	.744	6.81	8	.599	3.983	5	.367	2.159	3
6	.774	7.845	9	.629	4.389	5	.37	2.174	3
7	.794	8.718	10	.651	4.724	6	.373	2.19	3
8	.809	9.456	11	.667	5.003	6	.377	2.21	3

(c) Poisson 1D, (1.12)

Table 7.3: Comparison of three metrics for classical BPX preconditioners and optimized BPX preconditioners for selected equations. Spectral radius of error propagation matrix $\boldsymbol{I} - \boldsymbol{N}\boldsymbol{A}$ ($\boldsymbol{N}$ is BPX preconditioner $\boldsymbol{A}$ is a matrix of the original linear operators) denoted by ρ, condition number of preconditioned matrix κ, and number of iterations needed to drop and error by a factor of 10.

	$\epsilon = 10$						$\epsilon = 100$					
	BPX			$s = 1$ optimized BPX			BPX			$s = 2$ optimized BPX		
L	ρ	κ	N	ρ	κ	N	ρ	κ	N	ρ	κ	N
3	.919	23.7	28	.592	3.9	5	.974	75.5	87	.679	5.235	6
4	.956	44.1	51	.625	4.339	5	.991	216.1	249	.704	5.763	7
5	.967	60.3	70	.653	4.766	6	.996	468.4	540	.71	5.9	7
6	.973	72.4	84	.68	5.25	6	.997	753.1	867	.754	7.145	9

(a) Anisotropic Poisson, (1.20)

	$k^2h = 0.01$						$k^2h = 0.1$					
	BPX			optimized BPX			BPX			optimized BPX		
L	ρ	κ	N	ρ	κ	N	ρ	κ	N	ρ	κ	N
3	.621	4.277	5	.316	1.922	2	.621	4.277	5	.316	1.922	2
4	.701	5.678	7	.385	2.254	3	.701	5.678	7	.385	2.254	3
5	.746	6.867	8	.431	2.515	3	.746	6.867	8	.431	2.515	3
6	.774	7.866	10	.457	2.685	3	.774	7.866	10	.457	2.685	3

(b) Helmholtz, (1.23)

	BPX			optimized BPX		
L	ρ	κ	N	ρ	κ	N
3	.878	15.367	18	.846	11.984	14
4	.96	48.717	57	.878	15.33	18
5	.988	167.576	193	.899	18.9	22
6	.997	617.095	711	.945	35.073	41

(c) Biharmonic, (1.25)

Table 7.4: Comparison of three metrics for classical BPX preconditioners and optimized BPX preconditioners for selected equations. Spectral radius of error propagation matrix $\boldsymbol{I} - \boldsymbol{N}\boldsymbol{A}$ ($\boldsymbol{N}$ is BPX preconditioner $\boldsymbol{A}$ is a matrix of the original linear operators) denoted by ρ, condition number of preconditioned matrix κ, and number of iterations needed to drop and error by a factor of 10.

	$\tau = 0.5$						$\tau = 0.9$					
	BPX			optimized BPX			BPX			optimized BPX		
L	ρ	κ	N	ρ	κ	N	ρ	κ	N	ρ	κ	N
3	.68	5.255	6	.45	2.638	3	.817	9.93	12	.697	5.599	7
4	.751	7.044	9	.511	3.086	4	.89	17.24	20	.814	9.781	12
5	.79	8.51	10	.553	3.479	4	.922	24.787	29	.864	13.752	16
6	.813	9.685	12	.577	3.731	5	.935	29.933	35	.894	17.811	21

(a) Mixed derivative, (1.19)

	$\widetilde{\mu} = h/2$						$\widetilde{\mu} = 2/h$					
	BPX			optimized BPX			BPX			optimized BPX		
L	ρ	κ	N	ρ	κ	N	ρ	κ	N	ρ	κ	N
3	.908	20.7	24	.067	1.144	1	.641	4.57	6	.307	1.89	2
4	.979	94.2	109	.033	1.069	1	.729	6.38	8	.391	2.29	3
5	.995	407.7	470	.016	1.033	1	.789	8.47	10	.505	2.92	4
6	.99	1702.6	1961	.017	1.031	1	.845	11.88	14	.709	5.88	7

(b) Crank-Nicolson, (1.31)

	$\sigma = 10$						$\sigma = 100$					
	BPX (r)			optimized BPX (r)			BPX (r)			optimized BPX (r)		
L	ρ	κ	N	ρ	κ	N	ρ	κ	N	ρ	κ	N
3	.727	6.3	8	.657	4.8	6	.753	7.1	9	.614	4.2	5
4	.898	18.5	22	.616	4.2	5	.912	21.8	26	.692	5.5	7
5	.964	54.8	64	.652	4.8	6	.97	66.6	77	.739	6.7	8
6	.986	145.2	168	.744	6.8	8	.989	187.0	216	.809	9.5	11

(c) Discontinuous diffusion, (1.27)

Table 7.5: Comparison of three metrics for classical BPX preconditioners and optimized BPX preconditioners for selected equations. Spectral radius of error propagation matrix $\boldsymbol{I} - \boldsymbol{N}\boldsymbol{A}$ ($\boldsymbol{N}$ is BPX preconditioner $\boldsymbol{A}$ is a matrix of the original linear operators) denoted by ρ, condition number of preconditioned matrix κ, and number of iterations needed to drop and error by a factor of 10.

7.3.7 Mixed derivative

Results can be found in Table 7.5a. We can see that optimization is better for smaller values of τ, but when τ becomes closer to one, optimization deteriorates.

7.3.8 Implicit scheme for heat equation

Results are in Table 7.5b. We study problem (1.31) in two regimes. The first one corresponds to small time steps $\widetilde{\mu} = h/2$ used when the transient dynamic is of interest. In this case $I - \left(\widetilde{\mu}/2\right) A \simeq I$ so the preconditioner is not needed. As a result, BPX applied in a naive manner increases the condition number. The alternative solution would be to apply BPX preconditioner to the second matrix only, i.e., $I - \left(\widetilde{\mu}/2\right) BAB$, which solves this problem. However, the goal was to access the optimization, so we keep this experiment. In the other regime $\widetilde{\mu} = 2/h$ and one is interested in steady-state. In this situation, optimization again helps to decrease the spectral condition number. The last regime related to the elliptic equation with a linear source (different sign compare to the Helmholtz equation) for which a robust preconditioner was constructed in [GO95] with the help of a sophisticated subspace splitting technique.

Chapter 8

Neural multigrid architectures

8.1 Multigrid and neural networks

As explained in Section 1.2.3 multigrid is not a concrete solver but rather a set of techniques, universal in some sense, used to construct linear solvers. This fact paired with an excellent performance of multigrid on a wide set of problems leads to the spread of multigrid-inspired algorithms in other fields of science, including data science. The two evident examples are U-Net [RFB15] and Res-Net [He+16]. The former based on the repeated convolutions and downsampling operators followed by the set of convolutions with upsampling, while the later is based on the error-correction scheme – the idea that the error equation can be easier to solve in some circumstances.

The adoption of multigrid in data-science followed by the tendency to use data-driven approaches in the construction of multigrid solvers.[1] In particular, in [Gre+19a] authors proposed to use deep Res-Net to learn the part of projection operator with smoother being fixed in a fully unsupervised manner. Later authors of [Luz+20] extended the results into algebraic setting using graph neural networks. These articles firmly demonstrate that machine learning is a valuable tool for the construction of multigrid solvers.[2] In both [Gre+19a] and [Luz+20] authors use neural network as

[1] Other approaches to the automatic constriction of multigrid exist. Related areas include Bootstrap AMG [Bra+11], optimization based on local Fourier analysis [WJ05], and evolutionary computation [SKK19].

[2] Arguably, in both cases scalable solvers are obtained in part because authors utilize ready-made coarsening strategies and robust smoother. Namely, in [Gre+19a] a strategy from algebraic multigrid (AMG) is used to restore the solution on the fine grid in such a way that $b - Ax = 0$ for red points in a red-black pattern, and in [Luz+20] authors completely rely on AMG coarsening strategy.

an external tool and do not exploit the fact that multigrid itself can be considered as a form of neural network. In [Hsi+19] authors perform at attempt to construct linear solver based on unmodified U-Net. In this part we extend ideas from [Hsi+19] providing a convenient matrix-free multigrid architecture that contains only convolution layers.

The rest of the section is organized as follows. Section 8.2 contains the explanation how basic operations related to multigrid can be considered as layers in the neural network. The loss function that we use for unsupervised training is described in Section 8.3. Further in Section 8.4 we explain that it is impossible to keep both fixed architecture and a fast convergence speed of an iterative method. This result rules out approaches like [Hsi+19] that rely on fixed architecture. To resolve the issue, in Section 8.5 we propose architectures with the serialization of layers. The other architectures that roughly correspond [Hsi+19] and [KDO20] are described in the same section. All architectures are compared in Section 8.6.

Our implementation based on TensorFlow [Aba+16] as well as all results presented in this section are available in https://github.com/VLSF/nmg.

8.2 Matrix-free multigrid architecture

To describe our architecture, we need to introduce a few matrices. For each level $k \geq 1$ we use $\boldsymbol{A}_k, \boldsymbol{P}_k$ to describe matrix of linear operator and the restriction matrix. If the Pertov-Galerking condition is used the following relation holds $\boldsymbol{A}_{k+1} = \boldsymbol{P}_k \boldsymbol{A}_k \boldsymbol{P}_k^T$. For level $k = 1$, matrix $\boldsymbol{A}_1$ should be given either explicitly or as a linear operator, i.e., the black-box function that computes $\boldsymbol{A}_1 \boldsymbol{x}$ for any given $\boldsymbol{x}$ suffices.

On each level k we need to implement two-grid cycle as neural network. There are four operations we need to consider: computation of residual $\boldsymbol{r}_k \equiv \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x}_k$, restriction $\boldsymbol{P}\boldsymbol{r}_k$, prolongation (interpolation) $\boldsymbol{P}^T \boldsymbol{e}_k$, and smoothing $\boldsymbol{N}_k \boldsymbol{r}_k$.

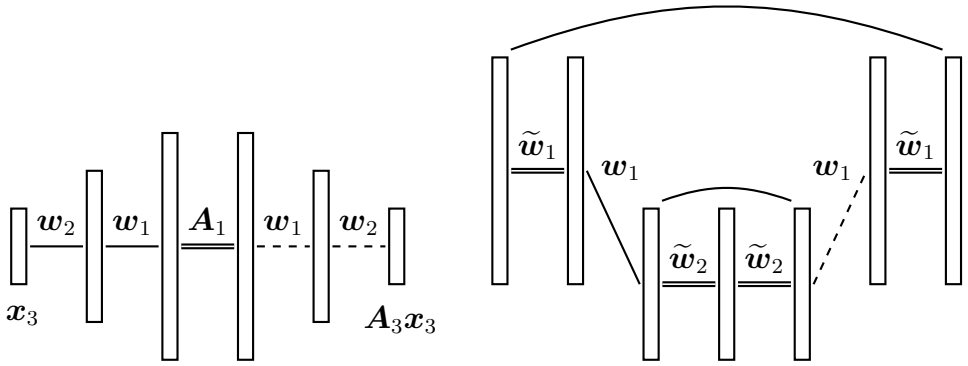
The simplest operations are restriction and prolongation that can be considered convolution with at least one stride > 1, and a transpose to this operation.

Computation of the residual is straightforward too. Because $\boldsymbol{A}_{k+1} = \boldsymbol{P}_k \boldsymbol{A}_k \boldsymbol{P}_k^T$, any product can be computed recursively:

$$\boldsymbol{A}_m \boldsymbol{x}_m = \left(\prod_{l=m-1,\ldots,1} \boldsymbol{P}_l \right) \boldsymbol{A}_1 \left(\prod_{l=1,\ldots,m-1} \boldsymbol{P}_l^T \right) \boldsymbol{x}_m. \tag{8.1}$$

This procedure is illustrated for $k = 3$ on Fig. 8.1a.

The situation with smoothers is less straightforward. Not all smoothers can be considered in a matrix-free framework. For example, Gauss-Seidel smoother explic-



(a) Computation of $\boldsymbol{A}$ (b) U-Net architecture

Figure 8.1: (a) – $\boldsymbol{A}_{i+1} = \boldsymbol{P}_i \boldsymbol{A}_i \boldsymbol{P}_i^T$, product $\boldsymbol{A}_3\boldsymbol{x}_3$ can be computed as a set of convolutions (dashed lines), and transposed convolutions (solid lines), and an application of operator on the fine grid (double line); $\boldsymbol{w}_i$ corresponds to convolution kernels; (b) – U-Net architecture with two layers, convolutions with all strides equal 1 are denoted by double lines (they correspond to smoothing in multigrid architecture), the single line represents convolution with at least one stride > 1, dashed line is a transpose to this operation, a curved line is a skip connection (copy and add).

itly requires a lower triangular part of the matrix, which can be hard to extract. However, there is a family of smoothers, known as polynomial smoothers [Ada+03, Section 3], that are better suited for our purposes. Polynomial smoothers take a form

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + p(\boldsymbol{A}) \left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(n)} \right), \; p(\boldsymbol{A}) = \sum_{i=0}^{D} \alpha_i \boldsymbol{A}^i, \tag{8.2}$$

where $\alpha_i, \; i = 0, \ldots, D$ are parameters of the smoother chosen based on matrix $\boldsymbol{A}$. Since (8.2) contains only vector-matrix products, we can apply the smoother using (8.1).

As a rule, polynomial smoothers are applied to the matrix with 1 on diagonal, i.e., the diagonal rescaling $\boldsymbol{D}(\boldsymbol{A})^{-1}$ is explicitly introduced. We hide this additional factor in convolution operation.

Algorithm 10 specifies the smoothers that we use. In lines 2 and 5, $\boldsymbol{A}_k(x)$ uses kernels $\boldsymbol{w}_i$, fine-grid operator $\boldsymbol{A}$ and should be computed as in (8.1), $\text{conv}_{\widetilde{\boldsymbol{w}}_{i-1}}$ in line 4 should preserve the size of the input vector, so all strides equal 1.

To summarize, for a given level k, we implement a two-grid cycle as a convolutional neural network with the following adjustments:

Algorithm 10 Polynomial smoothing.

Input: $\boldsymbol{x}_k$, $\boldsymbol{b}_k$, kernels $\boldsymbol{w}_i$, $i = k-1, \ldots, 1$ corresponding to convolutions $\boldsymbol{P}_l$, $l = k-1, \ldots, 1$, fine grid operator $\boldsymbol{A}$, kernels $\widetilde{\boldsymbol{w}}_j$, $j = 0, \ldots, D$ that are used to compute $p(\boldsymbol{A})$.

$r \leftarrow \boldsymbol{b}_k - \boldsymbol{A}_k(\boldsymbol{x})$
for $i = 1 : (D+1)$ **do**
  $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_k + \mathrm{conv}_{\widetilde{\boldsymbol{w}}_{i-1}}(\boldsymbol{r})$
  $\boldsymbol{r} \leftarrow \boldsymbol{A}_k(\boldsymbol{r})$
end for

1. $\boldsymbol{w}_k$, $\boldsymbol{s}_k$ are kernel and strides (at least one stride should be > 1) that implement convolution and transposed convolution that corresponds to $\boldsymbol{P}$ and $\boldsymbol{P}^T$;
2. $\widetilde{\boldsymbol{w}}_k^{(i)}, i = 0, \ldots, D$ are kernels that are used in Algorithm 10 that constitutes the pre- and post-smoothing steps in the two-grid cycle;
3. all convolutions are with zero biases and without nonlinearities,
4. any matrix-vector product is computed according to (8.1) (see also Fig. 8.1a).

The whole multigrid architecture can be constructed by recursive application of two-grid layers. To imitate matrix inversion on the coarsest grid, we use a few additional convolutions.

The presence of residuals makes it hard to draw the resulting architecture. But, in general, a neural network that imitates multigrid resembles U-Net [Hsi+19]. The one crucial difference is that U-Net contains only one ascending and one descending branches, whereas our architecture contains an additional Λ-shaped network at each place where the residual $\boldsymbol{b}_k - \boldsymbol{A}_k\boldsymbol{x}_k$ is needed.

Our approach offers the following advantages:

- Currently, no major machine learning framework supports sparse-sparse matrix multiplication, so $\boldsymbol{PAP}^T$ can not be computed efficiently.
- The architecture is agnostic to the sparsity pattern of $\boldsymbol{A}$ and $\boldsymbol{P}$ so that they can be changed easily. This can be especially useful when graph neural networks are used to learn the coarsening strategy.
- Since the network consists of convolution layers, one can benefit from using GPU.

- There is a one-to-one correspondence between some multigrid schemes and proposed architecture. This improves interpretability.

The main disadvantage is the additional operations we need to perform to compute $\boldsymbol{A}_k\boldsymbol{x}_k$. However, on modern GPUs, training on matrices with $n = (2^5 - 1)^2$ takes a few minutes, so the overhead seems to be justified by the overall convenience of the architecture.

8.3 Loss function and training

As explained in Section 7.1.1 spectral radius of the error propagation matrix is a good target for optimization. We ones again use stochastic estimation of spectral radius

$$\rho(\boldsymbol{B}) \simeq \rho_1(\boldsymbol{B}, k, N_{\text{batch}}) \equiv \left(\frac{1}{N_{\text{batch}}} \sum_{j=1}^{N_{\text{batch}}} \left\| \boldsymbol{B}^k \boldsymbol{z}_j \right\|_2^2 \right)^{1/2k}, \tag{8.3}$$

where $\boldsymbol{B}$ is an arbitrary matrix, and each $\boldsymbol{z}_j,\ j = 1, \ldots, N_{\text{batch}}$ is a random vector with components i.i.d. according to Rademacher distribution. In all our experiments we use $k = N_{\text{batch}} = 10$.

Since Eq. (8.3) contains only a repeated application of matrix $\boldsymbol{B}$ it is enough to have a black-box implementation of $\boldsymbol{B}\boldsymbol{z}$. So the optimization with Eq. (8.3) is straightforward.

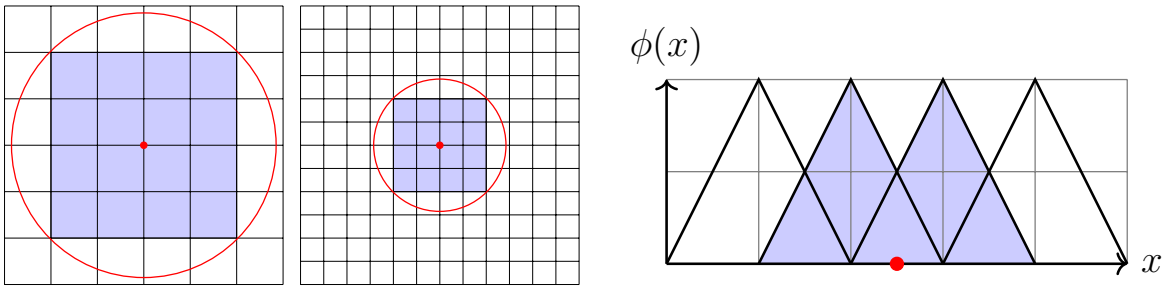
However, we are not interested in the optimization per se. The optimization was already performed, for example, in [KDO20]. The difference between training and optimization is that the former includes some form of generalization (or interpolation, or extrapolation) whereas the later does not [Goo+16, Chapter 5]. We can think of three situations that constitute training of linear solver:

1. Suppose there is a systematic way to produce related linear systems $\boldsymbol{A}_h$ based on parameter h, where matrices $\boldsymbol{A}_h$ differ by size.[3] In this case it is reasonable to train on small matrices and test on larger matrices. The training scheme is useful if the spectral radius remains small when the size of the matrix increases.

2. Suppose there is a systematic way to produce related linear systems $\boldsymbol{A}_\epsilon$ based on parameter ϵ, where matrices $\boldsymbol{A}_\epsilon$ have the same size.[4] In this case we train

[3] The discretization of PDEs is a relevant example. In this case h is a grid spacing.

[4] Here the example may be the set of heat conductivity problems (1.27) on a fixed mesh, where $k(x, y)$ is a random function, that is conductive properties of the medium change from problem to problem.



(a) Receptive field and refinement

(b) Locality of the right-hand side

Figure 8.2: (a) – the figure shows how the receptive field of fixed architecture with only local layers (shaded) changes after refinement. Since convolutions are performed on discrete data, information from the dot in the middle can spread over the smaller region (enclosed by the circle) in physical space. This limits the ability to generalize for a neural network with fixed architecture; (b) – when delta-function is presented as a right-hand side of a continuous problem, the weak form results in a sparse right-hand side because only a small number of (shaded) tent functions feels the presence of the source (denoted by a point).

on $\boldsymbol{A}_{\epsilon_i},\ i = 1, \ldots, K$ for some small K and hope that spectral radius remains small for unseen ϵ.

3. Combination of the above.

We will concentrate on the first scenario in what is following.

8.4 Restriction on architecture for linear iterative methods

Standard machine learning pipeline consists of choosing an appropriate architecture, training (supervised or unsupervised) with a given loss function, and applying trained model to unseen data [Goo+16, Chapter 11]. In this section, we argue that this approach is insufficient for training[5] specific neural networks if we are to use them as iterative methods.

To make an argument, we consider the following boundary value problem:

$$\Delta u(x, y, z) = -\delta(x)\delta(y)\delta(z), \quad u(x, y, z)|_{x^2+y^2+z^2=R^2} = 0, \tag{8.4}$$

[5]As defined in Section 8.3.

J	LMG	s1MG(rs)	s1MG(s)	s3MG(s)	U-Net	fMG
3	0.11	0.047	0.046	0.028	0.50	0.031
4	0.13	0.051	0.050	0.035	0.54	0.034
5	0.15	0.057	0.058	0.041	0.58	0.037
6	0.16	0.19	0.066	0.050	0.92	0.37
7	0.17	0.58	0.073	0.059	–	0.80
8	0.19	–	0.080	0.065	–	–
9	0.20	–	0.088	0.073	–	–
10	0.21	–	0.094	0.083	–	–
11	0.23	–	0.10	0.092	–	–

Figure 8.3: 5-point stencil, Poisson equation (1.14), $\rho\left(\boldsymbol{I}-\boldsymbol{\mathcal{N}}\boldsymbol{A}\right)$.

that is, a $3D$ Poisson equation with a point source at the origin, considered inside a sphere of radius R. The solution is easily obtained from Green function [Jac99, Section 1.10]

$$u(x,y,z)=\frac{1}{4\pi}\left(\frac{1}{\sqrt{x^2+y^2+z^2}}-\frac{1}{R}\right). \tag{8.5}$$

One way to solve (8.4) numerically is to use finite element method (see Section 1.3.2). For a suitable defined mesh (for example the mesh as in Fig. 8.2a can be used), we introduce a set of piecewise linear functions $\phi_i(x,y,z)$ that possess cardinality property: $\phi_i(x_j,y_j,z_j)=\delta_{ij}$, where (x_j,y_j,z_j) is a fixed grid point. The solution is approximated as $u(x,y,z)=\sum_i \phi_i(x,y,z)u_i$, and PDE is enforced in a weak form by Petrov-Galerkin condition $\int dxdydz\ \phi_i(x,y,z)\left(\Delta u(x,y,z)+\delta(x)\delta(y)\delta(z)\right)=0$. That gives us a system of linear equations with sparse matrix and sparse right-hand side. The sparsity of the right-hand side is illustrated by Fig. 8.2b.

Let $\mathcal{U}$ be a space of functions on a finite $3D$ grid with spacing $\simeq H$, and $\mathcal{N}$ be a linear neural network $\mathcal{U}\xrightarrow{\mathcal{N}}\mathcal{U}$ that acts like linear operator on space $\mathcal{U}$. Let $u_k\in\mathcal{U}$ be a function equals 1 at point k and 0 at all other points. Because the grid is finite, it is possible to find a minimal radius R_k such that all nonzero elements of $\mathcal{N}\left(u_k\right)$ are inside the ball with radius R_k centered at point k. We define the radius of influence of a given network $\mathcal{N}$ as

$$r_H(\mathcal{N})=\max_k R_k.$$

The example of this radius is given in Fig. 8.2a for convolution with 5×5 kernel.

Now, if refinement is performed and the architecture of the network does not contain dense layers, the radius of influence shrinks as explained in the same Fig. 8.2a.

J	LMG	s1MG(rs)	s1MG(s)	s3MG(s)	U-Net	fMG
3	0.16	0.058	0.069	0.042	0.53	0.030
4	0.22	0.063	0.073	0.041	0.58	0.031
5	0.25	0.070	0.079	0.049	0.62	0.038
6	0.28	0.081	0.088	0.086	–	0.54
7	0.30	0.33	0.096	0.11	–	0.89
8	0.32	0.82	0.10	0.12	–	–
9	0.35	–	0.11	0.13	–	–
10	0.37	–	0.12	0.14	–	–
11	0.40	–	0.13	0.15	–	–

Figure 8.4: 9-point stencil, Poisson equation (1.18), $\rho\left(\boldsymbol{I} - \boldsymbol{\mathcal{N}}\boldsymbol{A}\right)$.

This fact is used to prove the following statement.

Proposition 8.4.1. *Let $\boldsymbol{A}_H$ be a matrix of linear problem* (8.4) *obtained using finite element method on a given grid with spacing $\simeq H$. Let $\mathcal{N}$ be a neural network, that consists on finite number of (local) convolutional layers*[6]*, and used as $\boldsymbol{N}$ in linear iterative method $\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \boldsymbol{N}\boldsymbol{r}^n$. Suppose that the network has been trained to provide a good convergence for grid H, that is, $\rho(\boldsymbol{I} - \boldsymbol{\mathcal{N}}_H\boldsymbol{A}_H) = \epsilon \ll 1$. It is always possible to find a grid with spacing $\simeq h < H$ such that $\rho(\boldsymbol{I} - \boldsymbol{\mathcal{N}}_h\boldsymbol{A}_h)$ is arbitrary close to* 1*.*

The proposition above holds for networks that consist of (local) convolutional layers. We exclude networks with dense and nonlocal layers because they require $\simeq O(N^2)$ (N is a number of inputs) flops, which is unacceptable for iterative methods. On the other hand, convolutional neural networks require $\simeq O(N)$ flops and can be applied on grids with different sizes and geometries.

Our "radius of influence" is similar to the "domain of dependence" used to analyze convergence of numerical methods [LeV07, Section 10.7]. Also, there is an evident parallel with CFL condition [CFL67].

It is easy to see that the "no-go" result below immediately follows from Proposition 8.4.1.

Corollary 8.4.1. *Let $\boldsymbol{A}_H$ be a matrix of linear problem* (8.4) *obtained using finite element method on a given grid with spacing $\simeq H$. Let $\boldsymbol{\mathcal{N}}$ be a neural network, that consists of finite number of (local) convolutional layers. It is not possible to*

[6]We exclude nonlocal convolutions based on graph Laplacian as in [Bru+14].

J	LMG	s1MG(rs)	s1MG(s)	s3MG(s)	U-Net	fMG
3	0.073	0.030	0.036	0.022	0.40	0.017
4	0.094	0.034	0.041	0.028	0.44	0.019
5	0.11	0.041	0.048	0.02	0.60	0.022
6	0.12	0.13	0.058	0.042	0.96	0.52
7	0.13	0.40	0.065	0.051	–	0.99
8	0.14	0.78	0.071	0.057	–	–
9	0.15	0.99	0.077	0.063	–	–
10	0.16	–	0.083	0.069	–	–
11	0.17	–	0.090	0.076	–	–

Figure 8.5: Mehrstellen discretization (1.17), $\rho\left(\boldsymbol{I} - \boldsymbol{\mathcal{N}}\boldsymbol{A}\right)$.

have $\|\boldsymbol{I} - \boldsymbol{\mathcal{N}}_h\boldsymbol{A}_h\| \leq \epsilon \ll 1$*, with* ϵ *independent on* $h < H$*. In other words, it is impossible to uniformly approximate inverses to* $\boldsymbol{A}_h$ *using fixed architecture with local layers.*

Proof. Since $\rho(\boldsymbol{I} - \boldsymbol{\mathcal{N}}_h\boldsymbol{A}_h) \leq \|\boldsymbol{I} - \boldsymbol{\mathcal{N}}_h\boldsymbol{A}_h\|$ for any matrix norm, the statement can be proven by contradiction. □

Despite the fact that we showed the result for a particular elliptic equation with a particular right-hand side, the main fact that we used is that $\boldsymbol{A}_h^{-1}$ is nonlocal. For example, $\boldsymbol{A}_h$ from Proposition 8.4.1 is equivalent to $\boldsymbol{\mathcal{N}}$ with a single convolutional layer. Nonlocality of $\boldsymbol{A}_h^{-1}$ is shared by all elliptic equations so propositions like Proposition 8.4.1 holds for them too.

We would like to add that, it is known that neural network can approximate arbitrary nonlinear operator [CC95]. The corollary above does not contradict this result because it is restricted to neural networks with a fixed number of layers.

The results of this section implies that fixed architectures like [Hsi+19] are not going to work. This problem is addressed in the next section.

8.5 Architectures and the baseline solver

According to the result in the previous section, it is necessary to enlarge the network when we refine the grid. The simplest strategy is a serialization of layers. By serialization, we mean that an additional layer uses parameters from a previous layer. Here we formulate a few concrete architectures that we are going to compare in Section 8.6.

J	LMG	s1MG(rs)	s1MG(s)	s3MG(s)	U-Net	fMG
3	0.23	0.071	0.076	0.047	0.65	0.060
4	0.29	0.076	0.085	0.048	0.70	0.067
5	0.31	0.084	0.093	0.054	0.76	0.075
6	0.33	0.31	0.10	0.066	–	0.39
7	0.36	0.74	0.11	0.089	–	0.74
8	0.38	–	0.12	0.10	–	–
9	0.41	–	0.13	0.11	–	–
10	0.44	–	0.15	0.12	–	–
11	0.47	–	0.16	0.13	–	–

Figure 8.6: Anisotropic Poisson equation (1.21) with $\epsilon = 2$, $\rho\left(\boldsymbol{I} - \boldsymbol{\mathcal{N}}\boldsymbol{A}\right)$.

8.5.1 LMG

As a baseline model we use multigrid with linear interpolation and two pre-smoothing and two post-smoothing Jacobi sweeps with $\omega = 4/5$ (this ω is optimal for five-point discretization of Poisson equation in $2D$ [TOS00, Section 2.1.2]). Linear interpolation means that $\boldsymbol{P}$ corresponds to convolution with strides $(2, 2)$ with the kernel

$$k_{\text{linear}} = \frac{1}{2}\begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}. \tag{8.6}$$

8.5.2 s1MG(rs)

The name of the model derived from the fact that it is a neural multigrid (MG) architecture with a single serialized layer (s1), which contain adjustable restriction and smoothing operators (rs), with weights w and $\widetilde{w}$ respectively. To have the same number of floating-point operations as a baseline model, we use smoothing (Algorithm 10) with $D = 0$. Both $\boldsymbol{w}$ and $\widetilde{\boldsymbol{w}}$ represents kernels of sizes 3×3, which initially coincide with linear interpolation (8.6). Convolutional layer with kernel $\boldsymbol{w}$ has strides $(2, 2)$, and the layer with kernel $\widetilde{\boldsymbol{w}}$ has strides $(1, 1)$. For this model, we use exact matrix inversion as a coarse-grid correction. This is possible because we can always stack enough layers to have a single unknown on a coarse grid for considered model problems.

J	LMG	s1MG(rs)	s1MG(s)	s3MG(s)	U-Net	fMG
3	0.59	0.40	0.44	0.42	0.91	0.44
4	0.73	0.42	0.47	0.42	0.99	0.47
5	0.81	0.49	0.53	0.49	–	0.53
6	0.88	–	0.59	0.53	–	–
7	0.94	–	0.63	0.57	–	–
8	–	–	0.68	0.61	–	–
9	–	–	0.73	0.65	–	–
10	–	–	0.78	0.70	–	–
11	–	–	0.83	0.75	–	–

Figure 8.7: Anisotropic Poisson equation (1.21) with $\epsilon = 10$, $\rho\left(\boldsymbol{I} - \boldsymbol{\mathcal{N}}\boldsymbol{A}\right)$.

8.5.3 s1MG(s)

This model is the same as the previous one but with two differences. First, the restriction operator is fixed to be linear interpolation (8.6), and only the smoothing operator is learned. Second, we explicitly incorporate diagonal rescaling with $\boldsymbol{D}(\boldsymbol{A})^{-1}$ on each level. This is possible because restriction operators are fixed, so all diagonal can be computed in advance.

8.5.4 s3MG(s)

The model is the same as a previous one, but now we train three distinct layers with $\widetilde{\boldsymbol{w}}_1$, $\widetilde{\boldsymbol{w}}_2$, $\widetilde{\boldsymbol{w}}_3$. The serialization is performed as follows:

$$\begin{aligned} &\text{layer } 1: \ \widetilde{\boldsymbol{w}}_1; \ \text{layer } 2: \ \widetilde{\boldsymbol{w}}_2; \ \text{layer } 3: \ \widetilde{\boldsymbol{w}}_3; \\ &\text{layer } 4: \ \widetilde{\boldsymbol{w}}_1; \ \text{layer } 5: \ \widetilde{\boldsymbol{w}}_2; \ \text{layer } 6: \ \widetilde{\boldsymbol{w}}_3; \\ &\text{layer } 7: \ \widetilde{\boldsymbol{w}}_1; \ \ldots \end{aligned} \tag{8.7}$$

8.5.5 U-Net

This is an attempt to reproduce results from [Hsi+19].[7] We use architecture presented in Fig. 8.1b, but with 5 layers. U-Net is used as N in linear iteration (8.6). Parameters for all layers are distinct, and no serialization is performed.

[7] Which is nontrivial because the code is absent and the architecture of the model is unspecified. We cannot also use results from the article because they are scarce, and authors measure performance relative to the multigrid method, which they did not bother to describe in detail.

J	LMG	s1MG(rs)	s1MG(s)	s3MG(s)	U-Net	fMG
3	0.11	0.040	0.049	0.029	0.49	0.043
4	0.14	0.049	0.056	0.032	0.54	0.048
5	0.15	0.061	0.065	0.037	0.58	0.054
6	0.17	0.27	0.073	0.045	0.88	0.34
7	0.18	0.67	0.081	0.052	–	0.78
8	0.19	–	0.088	0.062	–	0.98
9	0.21	–	0.095	0.072	–	–
10	0.22	–	0.10	0.083	–	–
11	0.24	–	0.11	0.092	–	–

Figure 8.8: Equation with mixed derivative (1.19) with $\tau = 1/4$, $\rho\left(\boldsymbol{I} - \boldsymbol{\mathcal{N}}\boldsymbol{A}\right)$.

8.5.6 fMG

This is another model without serialization. We use 5 layers with distinct restriction $\boldsymbol{w}$ and smoothing $\widetilde{\boldsymbol{w}}$ operators. Two convolutions are used as a coarse grid correction. All kernels are initialised as bilinear interpolation (8.6).

8.6 Numerical examples

For all discrete equations we perform a rescaling of linear operator as follows

$$\boldsymbol{A} \to \boldsymbol{D}(\boldsymbol{A})^{-1/2}\boldsymbol{A}\boldsymbol{D}(\boldsymbol{A})^{-1/2}. \tag{8.8}$$

8.6.1 Poisson equation

Results are presented in Fig. 8.3, Fig. 8.4, Fig. 8.5. For all discretizations of the Poisson equation, we can see that architectures U-Net, fMG, and s1MG(rs) fail to provide a good solver for $J \geq 6$.

Presumably, the spectral radius of error propagation matrices corresponding to U-Net and fMG architectures deteriorates because neural networks have fixed sizes.

This explanation does not work for s1MG(rs) because of the serialization performed. We can conjure that because both restriction and smoothing operators are optimized, s1MG(rs) is getting tuned to the spectrum of the matrix with $n = (2^5-1)^2$, since the spectrum changes when J increases, the solver ceases to be efficient. It is evident from other examples that the naive serialization does not seem to work when both restriction and smoothing operators are optimized.

J	LMG	s1MG(rs)	s1MG(s)	s3MG(s)	U-Net	fMG
3	0.24	0.097	0.15	0.055	0.51	0.062
4	0.35	0.11	0.16	0.071	0.56	0.069
5	0.41	0.12	0.19	0.087	0.61	0.082
6	0.46	0.24	0.23	0.19	0.79	0.33
7	0.50	–	0.25	0.25	0.99	0.72
8	0.54	–	0.28	0.28	–	0.98
9	0.59	–	0.31	0.30	–	–
10	0.63	–	0.33	0.33	–	–
11	0.68	–	0.36	0.35	–	–

Figure 8.9: Equation with mixed derivative (1.19) with $\tau = 3/4$, $\rho\left(\boldsymbol{I} - \boldsymbol{\mathcal{N}}\boldsymbol{A}\right)$.

The only two solvers (besides a baseline model) that retain their efficiency are s1MG(s) and s3MG(s). The latter is better than the former for five-point and Mehrstellen discretizations, but for the long stencil s1MG(s) is superior.

We can conclude that for the Poisson equation, U-Net is the weakest model, fMG and s1MG(rs) fail to generalize on the test set, and both s1MG(s) and s3MG(s) can generalize and outperform a baseline model on a test set.

8.6.2 Anisotropic Poisson equation

Fig. 8.6, Fig. 8.7 contain the results. For equation (1.21), the trend is largely the same. That is, fMG, s1MG(rs) and U-Net lose their efficiency, s1MG(s) and s3MG(s) are robust and outperform a baseline model.

It is instructive to discuss results for anisotropic equation with $\epsilon = 10$. First, we can see that all solvers are relatively inefficient. The reason is a full coarsening that we applied. If one uses semicoarsening instead, the results would be the same as for the isotropic Poisson equation. Because of the full coarsening, the U-Net solver fails already on a train set. If one further increases ϵ, our networks would not be able to provide efficient solvers unless strides are chosen appropriately. If strides and sizes of filters are considered as hyperparameters, it should be possible to apply Bayesian optimization [Sha+15], reinforcement learning [Li+18], or genetic programming [SKK19] to construct optimal solver.

	stencil size = 4		stencil size = 5		stencil size = 6	
J	s1MG(s)	s3MG(s)	s1MG(s)	s3MG(s)	s1MG(s)	s3MG(s)
3	0.033	0.025	0.031	0.026	0.019	0.020
4	0.034	0.023	0.030	0.024	0.025	0.020
5	0.039	0.026	0.035	0.027	0.032	0.022
6	0.051	0.037	0.051	0.044	0.040	0.036
7	0.063	0.047	0.063	0.057	0.047	0.051
8	0.073	0.055	0.074	0.065	0.053	0.061
9	0.083	0.065	0.082	0.071	0.059	0.066
10	0.092	0.072	0.092	0.078	0.064	0.072
11	0.10	0.079	0.10	0.085	0.069	0.078

Figure 8.10: The effect of smoother's stencil size on the spectral radius of error propagation matrix $\rho\left(\boldsymbol{I} - \boldsymbol{\mathcal{N}}\boldsymbol{A}\right)$ for 5-point stencil Poisson equation (1.14).

8.6.3 Mixed derivative

Results are given in Fig. 8.8, Fig. 8.9. Equation with mixed derivative changes type from elliptic to hyperbolic when τ crosses 1. It is interesting to look at how our models behave when τ approach 1.

For $\tau = 1/4$ architectures s1MG(s), s3MG(s) produces more efficient solvers than the standard multigrid with two Jacobi sweeps. On the other hand, U-Net is of no use even on the test set, and fMG and s1MG(rs) deteriorate rapidly for $J > 5$.

We can see that for $\tau = 3/4$ s3MG(s) performs substantially better than s1MG(s) on the train set. However, on the test set, it results in only a marginally smaller spectral radius. This means that the training and serialization strategies are not ideal. It should be possible to use additional parameters of s3MG(s) more efficiently. Other architectures behave similarly to the case $\tau = 1/4$.

8.6.4 Influence of smoother's stencil size

Our implementation of operator-based multigrid is sufficiently flexible to perform experiments with multigrid components of variable sizes. To demonstrate this, we show the learning results for multigrids with progressively large stencils of smoothing operators.

In Fig. 8.10 one can find results for Poisson equation with 5-point stencil (1.14). In all the experiments, we used 400 loss evaluations. As we can see, the increase in the stencil size for relaxation operator clearly improves convergence rate. For

	stencil size = 4		stencil size = 5		stencil size = 6	
J	s1MG(s)	s3MG(s)	s1MG(s)	s3MG(s)	s1MG(s)	s3MG(s)
3	0.26	0.30	0.32	0.22	0.18	0.16
4	0.28	0.34	0.46	0.38	0.30	0.23
5	0.36	0.39	0.52	0.45	0.37	0.31
6	0.41	–	0.56	0.52	0.50	0.45
7	0.44	–	0.61	0.55	0.80	0.66
8	0.48	–	0.64	0.59	0.90	0.88
9	0.51	–	0.69	0.63	0.99	–
10	0.55	–	0.74	0.68	–	–
11	0.59	–	0.79	0.72	–	–

Figure 8.11: The effect of smoother's stencil size on the spectral radius of error propagation matrix $\rho\left(\boldsymbol{I} - \boldsymbol{\mathcal{N}}\boldsymbol{A}\right)$ for anisotropic Poisson equation (1.21) with $\epsilon = 10$.

s1MG(s) we have 30% improvement and for s3MG(s) we observe 15% improvement on grid with $J = 11$.

In Fig. 8.11 one can find results for anisotropic Poisson equation (1.21) with $\epsilon = 10$. Here we can observe a few peculiar features. First, the improvement for s1MG(s) is still about 30% whereas for s3MG(s) we observe very little improvement. Second, the models are seized to be robust: both s1MG(s) and s3MG(s) fail to generalize for some stencils (the former one for stencil size 6, and the latter one for stencil sizes 4 and 6).

We also should note that we performed a set of experiments with more aggressive coarsening. In all cases tried, we failed to train a neural multigrid solver that converges and generalizes.

Chapter 9

Relaxation methods in the multi-armed bandit setting

9.1 Adaptive linear solvers

An introduction of successive over-relaxation (see Eq. (1.4)) [You54] was a decisive moment for the numerical linear algebra community. It was believed before that an efficient linear solver is hard or impossible to construct on a digital computer.[1] To efficiently apply SOR to a given problem, it is crucial to have an accurate approximation to the optimal over-relaxation parameter ω. Research on the topic splits into two directions. The first one is to derive the convenient form of ω for a restricted class of matrices [Had00]. The other opportunity is to start with some ω and gradually move toward optimal value based on the efficiency of iteration [Rei66], [UNK94].

The main deficiency of the first approach is a scarcity of general results. Even

[1] Quote from David Young [You87]: "Not too long after I began my work, Sir Richard Southwell visited Birkhoff at Harvard. One day when he, Birkhoff, and I were together, I told him what I was trying to do. As near as I can recall, his words were "any attempt to mechanize relaxation methods would be a waste of time." This was somewhat discouraging, but my propensity for making numerical errors was so strong that I knew that I would never be able to solve significant problems except by machines. Thus, though discouraged, I continued to work."

Quote from Radii Fedorenko has the same vibes [Fed62]: "Our method has been described as relaxational for the following reason. In the literature, a relaxation method is usually described thus: the calculator, having obtained an approximate solution of the system and having computed its discrepancy, changes the approximate solution in such a way that the discrepancy is made to vanish where it is large without at the same time being increased at other points; naturally, this procedure is the more successful, the more experienced the calculator. Evidently, the "experience" of the calculator here consists of skill in solving "by eye" a system with a discrepancy as the right-hand side. We have tried to transfer this operation to the digital computer."

when the results are available, optimal ω known for the overspecialized linear problem is not that attractive from the perspective of applications. The generalization of the second approach, that is, adaptation based on the empirical output from the solver, seems more universally applicable. The goal of the present part is to build a general framework that allows for the online adaptation of parametric families of the linear solver.

Of course, SOR and other classical relaxation methods became less relevant with the advent of efficient preconditioners for projection methods [Saa20], [Saa19], and multigrid method [TOS00]. Fortunately, the methods that we develop in this part can be used to optimize smoothers and projection operators for multigrid, sparse approximate inverses, filtering preconditioners, modified BPX preconditioners, and any other parametric family of linear solvers with a modest set of free parameters.

To achieve this goal, we render parametric linear iteration in terms of reinforcement learning or online optimization. That is, we introduce a loss function that can be computed with a little additional cost as a byproduct of iteration and use this loss function to guide the optimization process. We start with $\epsilon-$greedy algorithm for multi-armed bandit [LS20] with fixed discretization, showing the inconsistency of the standard version of the algorithm, and introduce a few improved versions. In particular, we develop 1) restarted multi-armed bandit which produces a consistent estimation of spectral radius; 2) fidelity-based algorithm for arm-exclusion using Bauer-Fike upper bound [Saa11, Theorem 3.6]; 3) rediscretization strategy that allows "off the grid" convergence to the best arm. The resulting algorithms are compared on random linear systems as well as on different versions of second-order elliptic equations.

9.2 Reinforcement learning

Reinforcement learning (RL) is a process when an agent changes her course of actions guided by feedback from the environment [SB98, Section 1.1]. The initial development of the field was inspired by observations of how people and other animals learn. In particular, the notion of reinforcers and reward signals was developed in the school of behavioral psychology (see [Maz17] for a general introduction, and [Sut84] for the connection with reinforcement learning) and later lifted to the level of individual neurons and neural circuits [NA19]. To exemplify, Rescorla–Wagner model [Maz17, Chapter 4], [NA19, p. 135] describes learning process in the same fashion as temporal-difference learning algorithms [Sut88].

Nowadays, reinforcement learning is a mature field with well-defined mathematical formalism. A comprehensive introduction to the subject can be found in [SB98].

In this part, we give only a brief description of relevant mathematics and solution techniques used in the context of reinforcement learning.

9.2.1 Markov Decision Processes

Mathematical description of reinforcement learning can be effectively summarised by Markov Decision Processes (MDP). In the discrete setting MDP is fully specified by a transition function

$$p\left(s^{'}, r \middle| s, a\right), s^{'}, s \in \mathcal{S}, r \in \mathcal{R}, a \in \mathcal{A}(s), \tag{9.1}$$

that is the conditional probability to receive reward r and transfer from state s to state $s^{'}$ if the action a is chosen, where the sets of states $\mathcal{S}$ and actions $\mathcal{A}(s)$ for given state are finite.

When transition function Eq. (9.1) is given, the agent is described by a policy function $0 \leq \pi(a|s) \leq 1 : \sum_a \pi(a|s) = 1 \, \forall s$, which prescribes what action a to undertake in state s. The policy with a transition function generates a trajectory (or history)

$$(S_0, A_0), (S_1, A_1, R_1), \ldots, (S_{N-1}, A_{N-1}, R_{N-1}), (S_N, R_N), \tag{9.2}$$

that is, a record of states agent encountered S_i, actions she performed A_i, and rewards R_i she received during interaction with the environment. Trajectory Eq. (9.2) can be infinite (see [SB98, Section 3.3]), but for convenience we consider finite trajectories that ends after N actions. Different policies generate different trajectories, and the preferable (optimal) policy $\pi^\star$ is the one, that generates the largest mean cumulative reward (also known as return)

$$G_0(\pi, p) = \mathbb{E}\left[\sum_{i=1}^{N} R_i\right]_{\pi,p}, \; \pi^\star = \arg\max_\pi G_0(\pi, p). \tag{9.3}$$

MDP is a rather abstract framework that can be applied to a variety of situation, including playing complex games [Mni+13], [Sil+17], [Ber+19], learning coarsening strategies for multigrid [Tag+21], turbulence modelling [NLK21], self-driving cars [Dua+20], etc. More examples can be found in [Fra+18, Section 1].

9.2.2 Multi-armed bandits

Multi-armed bandit (or k-armed bandit) provides the simplest nontrivial case of MDP. In this particular situation, MDP consists of a single state s and $k \geq 2$ actions.

This means the agent only needs to decide which action to take, so the trajectory Eq. (9.2) reduces to the sequence of actions and rewards. The goal of the agent is to maximize the cumulative reward received after taking N actions. The challenge, as usual, is to balance exploration and exploitation. Exploration consists of estimation of the mean reward of each action, which is not known a priori. Exploitation consists of choosing optimal action as often as possible. Exploration and exploitation are in conflict. When the agent explores a lot, she is at risk of choosing non-optimal actions too often, when the agent only exploits she is at risk of never discovering true optimal action.

There is a large set of algorithms that can balance exploration and exploitation for k-armed bandits (see [LS20]). Here we are interested in the simplest one, known as $\epsilon-$greedy. The agent, following $\epsilon-$greedy strategy, chooses the best action (according to her current estimation) with probability $1-\epsilon$, and random action with probability ϵ for some small $\epsilon < 1$. The reward of action i is estimated as an empirical average of all rewards received when choosing i. The procedure is summarized in Algorithm 11.

Algorithm 11 $\epsilon-$greedy $k-$armed bandit

Input: $\epsilon < 1$ – probability to perform exploration, $k-$armed bandit, N – number of rounds.

Reward estimation and number of visits $R_i = 0, n_i = 0$, $i = 1, \dots, k$.
for $j = \overline{1, N}$ **do**
  $e \sim \text{Uniform}\,[0, 1]$
  if $e > 1 - \epsilon$ **then**
    $i = \text{random}\,\{1, \dots, k\}$
  else
    $i = \arg\max_j R_j$
  end if
  Take action A_i receive reward r
  $R_i \leftarrow R_i + (r - R_i) \big/ \max(n_i, 1)$, $n_i \leftarrow n_i + 1$
end for

In the later sections, we will show how to adapt this algorithm to the online optimization of the relaxation method. Our main focus is on the definition, construction, and estimation of the reward signal. The way how exploration and exploitation are balanced is less important. As a replacement to $\epsilon-$greedy strategy, it is definitely possible to use, for example, upper-confidence bound [ACF02], or strategies from Gaussian Optimization [Sha+15].

9.3 Linear iterative methods and reinforcement learning

Reinforcement learning is not something typically used in numerical linear algebra, so we discuss the motivation and prototypical linear problem we are going to solve.

There are plenty of situations in scientific computing when precisely the same linear system (or a family of related linear systems) is solved repeatedly for different right-hand sides. The following list contains a few relevant examples:

1. Chorin's projection method [Cho67].

   In this splitting scheme, the Poisson equation needs to be solved during each time step to recover pressure from auxiliary velocity.

2. Crank-Nicholson scheme for heat equation Section 1.3.10.

   As in the case of any implicit scheme, one needs to solve a linear system to obtain an approximation for the next time step.

3. Quasi-geostrophic omega equation [BS10].

   This anisotropic elliptic equation is solved during time marching when weather forecasts are performed.

4. PDE constrained optimization [Bie+03].

   In this scenario, one requires to solve the same PDE for different input data (boundary conditions, initial conditions, right-hand-side, values of parameters, etc), which typically requires the solution of linear systems. See [RDW10] for a concrete example where the constraints are formed by elliptic PDE.

   There is also a simple example of this situation in the present thesis. Namely, the optimal heating problem discussed in Section 4.4.3 leads to the situation when the heat equation needs to be solved multiple times.

5. Inverse problems [Tar05].

   The usual strategy to solve inverse problems includes Monte Carlo techniques [Tar05, Chapter 2], which require repeated solutions to forward problems. In the case of certain discretization of PDE, this, again, leads to large sparse linear systems.

6. More examples can be found in literature on Krylov subspace recycling techniques (see [Par+06] and references therein).

We hope the reader is convinced that in many practical applications the same linear system is often needed to be solved more than once. The general strategy in this scenario is to use some additional information to tune the linear solver. More precisely, when a solver is applied to a linear system, it is usually possible to measure its efficiency. For example, one can use a number of iterations needed to drop the norm of the residual by an order of magnitude. This measure of efficiency in its turn can be used to tune the free parameters of the solver (for example, relaxation parameters in SOR and projection weights in multigrid) to improve the speed of convergence. This online optimization problem is formulated below in a more abstract setting.

Problem 9.3.1 (**Online optimization of iterative solver**). *Let $\boldsymbol{M}(\theta)$ be a parametric family of linear iterative methods (see Section 1.2.1), $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}(\omega)$ (possibly $\boldsymbol{A}(\omega)\boldsymbol{x} = \boldsymbol{b}(\omega)$) is a parametric family of linear equations, and $N(\theta)$ is a number of iteration (in a worst case scenario) of iterative method with parameter θ needed to obtain relative error ϵ. Linear problems are presented according to some unknown schedule $\omega_1, \omega_2, \ldots, \omega_K$. The goal is to find a schedule $\theta_1, \theta_2, \ldots, \theta_K$ such that $\sum_{i=1}^{K} N(\theta_i)$ is as small as possible.*

As we mentioned in Section 9.1, the specialized adaptive algorithms of this kind are already presented in the literature. The main feature of algorithms we will describe in the next section is that they can be applied more broadly and do not depend explicitly on the nature of the underlying iterative method.

9.4 Linear iterative methods and bandits

In this section, we build a few specialized bandit algorithms of increasing sophistication, but first, we explain what is the set of actions, episodes, and the reward signal for the task described in Problem 9.3.1.

As we explained in Section 7.1.1 the classical measure of convergence speed for linear iterative method is a spectral radius of error propagation matrix $\rho(\boldsymbol{M}(\theta))$. Because of that it is naturally to use approximation to $1/\rho(\boldsymbol{M}(\theta))$ as a reward signal. How this approximation is computed is explained in the next sections.

Having a reward signal, we switch to the definition of actions and episodes. From now on we suppose that the space of parameters θ is discretized, so we have $\theta_i, i = 1, \ldots, K$ iterative solvers to choose from. These constitute actions. In Problem 9.3.1 we are repeatedly presented with a right-hand sides, so we are, in principle, at liberty to pick new θ_i for each new iteration. For simplicity we consider a set-up when we

Term	Definition
Action	The choice of parameter θ that specifies linear solver $\boldsymbol{M}(\theta)$.
Episode	K iterations with chosen solver until either relative residual is smaller then ϵ or the maximal number of iterations $K_{\max}$ is reached.
Reward signal	Estimation of spectral radius of error propagation matrix $\rho(\boldsymbol{M})$.

Table 9.1: Definition of action, episode and reward signal for Problem 9.3.1

pick one θ_i for each new right-hand side. These parameters is used for K iterations until the relative residual is small enough, that is $\left\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^K\right\| / \left\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^0\right\| \leq \epsilon < 1$. These K iterations constitute a single episode.

Definitions are summarized in Table 9.1.

9.4.1 Naive ϵ-greedy algorithm

To adopt Algorithm 11 we need to define how the reward should be computed. One simple solution is to look at the quantity

$$\hat{\rho}_{m+k,m} = \left(\frac{\left\|\boldsymbol{x}^{m+k} - \boldsymbol{x}^m\right\|}{\left\|\boldsymbol{x}^m - \boldsymbol{x}^{m-1}\right\|} \right)^{1/k}, \tag{9.4}$$

which is known to converge to $\rho(\boldsymbol{M})$ (for almost all starting error vectors) if 1 does not belongs to the spectrum of $\boldsymbol{M}$ (see [Hac16, Exercise 2.33]).

Eq. (9.4) can be used to construct iterative method that computes reward signal as explained in Algorithm 12. Now, when the reward signal is available, we can use it to balance exploration and exploitation according to Algorithm 11. Empirical evaluation and discussion of the resulting algorithm can be found in the section below.

Experiment

To illustrate typical behavior of Algorithm 12 combined with Algorithm 11, we perform a series of experiments on small randomly generated linear systems. The precise form of each matrix $\boldsymbol{A}$ reads

$$\boldsymbol{A} = \boldsymbol{I} + \boldsymbol{B}\boldsymbol{B}^T \in \mathbb{R}^{100\times 100}, \tag{9.5}$$

where each element B_{ij} is nonzero with probability $p = 0.2$, and the value of each nonzero element B_{ij} is drawn from uniform distribution on the interval $[0, 1]$.

Algorithm 12 Iterative solver with convergence speed estimation.

Input: $\boldsymbol{M}(\theta)$ – error propagation matrices that define the family of iterative methods, $\boldsymbol{x}$ – initial guess, $\boldsymbol{b}$ – right-hand side, $\boldsymbol{A}$ – matrix, δ – tolerance, $K_{\max}$ – maximal number of iterations.
Output: $\boldsymbol{x}$ – approximate solution to $\boldsymbol{Ax} = \boldsymbol{b}$, $\hat{\rho}(\theta)$ – spectral radius estimation of error propagation matrix $\boldsymbol{M}(\theta)$.

$\boldsymbol{y} = \boldsymbol{M}(\theta)\boldsymbol{x} + \boldsymbol{N}(\theta)\boldsymbol{b}$.
$\delta_0 = \|\boldsymbol{x} - \boldsymbol{y}\|$, $\boldsymbol{x} = \boldsymbol{y}$
$\mathrm{r}_0 = \|\boldsymbol{b} - \boldsymbol{Ax}\|$, $N_{\text{it}} = 1$
while $r > \delta$ or $N_{\text{it}} \leq N_{\max}$ **do**
    $\boldsymbol{x} \leftarrow \boldsymbol{M}(\theta)\boldsymbol{x} + \boldsymbol{N}(\theta)\boldsymbol{b}$
    $r \leftarrow \|\boldsymbol{b} - \boldsymbol{Ax}\| \,/ r_0$, $N_{\text{it}} \leftarrow N_{\text{it}} + 1$
end while
$\boldsymbol{y} = \boldsymbol{M}(\theta)\boldsymbol{x} + \boldsymbol{N}(\theta)\boldsymbol{b}$.
$\delta_1 = \|\boldsymbol{x} - \boldsymbol{y}\|$, $\boldsymbol{x} = \boldsymbol{y}$
$\hat{\rho}(\theta) = \left(\delta_1 / \delta_0\right)^{1/N_{it}}$

To test $\epsilon-$greedy bandit we drew 50 matrices Eq. (9.5) and run Algorithm 12 combined with Algorithm 11 for 2000 episodes. Results are given in Fig. 9.1. The details of the experiment is as follows. Tolerance δ was chosen to be 10^{-5}, probability of exploration $\epsilon = 0.1$, and the number of actions is 35. For Jacobi method relaxation parameter θ was taken from uniform discretization of the interval $[1/\lambda_{\min}, 1/\lambda_{\max})$, where $\lambda_{\min}, \lambda_{\max}$ are extreme eigenvalues of the matrix $\boldsymbol{D}(\boldsymbol{A})^{-1}\boldsymbol{A}$, $\boldsymbol{D}(\boldsymbol{A})$ is the diagonal of $\boldsymbol{A}$. For SOR relaxation parameter θ is chosen from the uniform grid on the interval $(0, 2)$. Since all matrices are positive-definite, both methods converge for all values of θ.

We can conclude from the result Fig. 9.1, that the additional cost incurred by computation of Eq. (9.4) is negligibly small in comparison with other operations, performed by the iterative method. This is clear from the wall-clock time measured for the adaptive algorithm and the non-adaptive one. We can see that the adaptive Jacobi method is 23% faster than the non-adaptive, whereas adaptive SOR is 66% faster. So in this case adaptation is well-justified.

However, we can see from the last figure in Fig. 9.1, that the accuracy of spectral radius estimation hardly improves over time. This is the case because our estimation of the spectral radius is biased. Namely, when we apply Algorithm 12, we always perform a fixed number of iterations, so the distance between the exact spectral radius

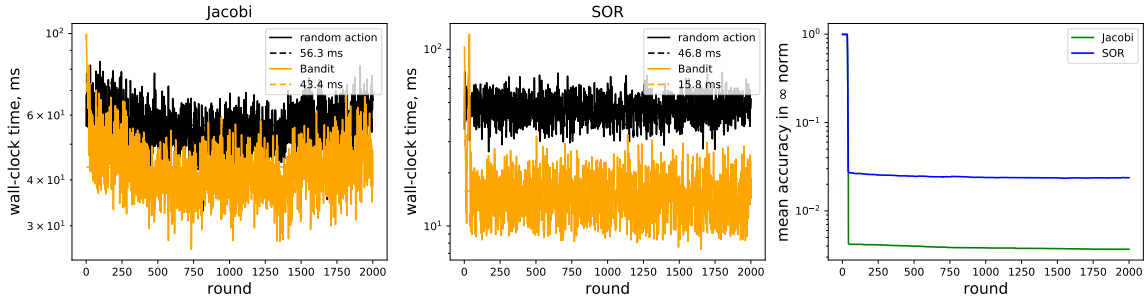


Figure 9.1: Comparison of adaptive (see Algorithm 12 in combination with Algorithm 11) and classical versions (random action is taken for each episode) of Jacobi and SOR iterations. First two plots contain average time needed to drop L_2 norm of initial residual by 10^{-5}. For each method, the mean execution time is specified with the dashed line in the legend. The last graph shows the maximal (among 35 arms) absolute discrepancy between estimated spectral radius and the exact one. All results are averaged over 50 random linear systems Eq. (9.5).

and our estimate never significantly decreases. Obviously, the statistical averaging in Algorithm 11 (see line 11) is not going to improve accuracy. That is why we put "naive" in the title of this section. To improve on this simple approach we need to reconsider how spectral radius is estimated. This is done in the next section.

9.4.2 Restarted ϵ-greedy algorithm

The main reason why Algorithm 12 fails to provide a convergent estimation of the spectral radius is that it always starts afresh. We solve this problem by making exploration and exploitation asymmetric. Now, the exploration step only estimates spectral radius and never improves the current estimation to the exact solution, whereas the exploitation step is used to actually solve the linear system and never provides an estimate of the spectral radius. Our approach is summarized in Algorithm 13

For simplicity, we use power iteration [Saa11, Algorithm 4.1] to estimate leading eigenvalue and eigenvector (line 9 in Algorithm 13). Albeit the convergence of power iterations is in general not impressive [Saa11, Theore, 4.1], we find it work sufficiently well to illustrate all features of Algorithm 13. In any case, almost arbitrary method for eigenvalue estimation can be used in Algorithm 13. The only requirement is that it should be possible to stop and later restart this method.

Algorithm 13 Restarted $\epsilon-$greedy algorithm.

Input: $\boldsymbol{M}(\theta)$ – error propagation matrices that define the family of iterative methods, $\theta_j,\ j = 1, \ldots, k$ – discretization of parameter space, N_{rounds} – number of rounds, $\boldsymbol{b}_i,\ i = 1, \ldots, N_{\text{rounds}}$ – right-hand sides for each round, $\boldsymbol{A}$ – matrix, δ – tolerance, $K_{\max}$ – maximal number of iterations, ϵ – probability of exploration, K – number of iteration allowed for eigenpair estimation, $E(\boldsymbol{s}, \lambda, K)$ – algorithm for eigenpair estimation that takes approximate leading eigenpair $\boldsymbol{s}, \lambda$ and provide improved estimation of eigenpair using K iterations.
Output: $\boldsymbol{x}_i$ – approximate solutions to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}_i$, $\hat{\rho}(\theta_j)$ – approximate spectral radii of error propagation matrices $\boldsymbol{M}(\theta_j)$.

Randomly initialize leading eigenvectors for each arm $\boldsymbol{v}_j$
Initialize approximate spectral radii for each arm $\hat{\rho}(\theta_j) = 0$.
for $i = 1, \ldots, N_{\text{rounds}}$ **do**
  $e \sim \text{Uniform}\,[0, 1]$
  if $e > 1 - \epsilon$ **then**
    $i = \text{random}\,\{1, \ldots, k\}$
    Improve eigenpair estimation $\boldsymbol{v}_j, \hat{\rho}(\theta_j) \leftarrow E(\boldsymbol{v}_j, \hat{\rho}(\theta_j), K)$.
  end if
  $k = \arg\min_j \hat{\rho}(\theta_j)$
  Solve linear system for current round with relaxation method $\boldsymbol{M}(\theta_k)$.
end for

Experiment

The experiments with Algorithm 13 were performed using the same setting that we described after Eq. (9.5). Additional parameter K (the number of iteration of power series) is chosen to be 10.

The results are presented in Fig. 9.2. Two differences are evident. First, we can see that for SOR the improvement on the average is much smaller than in Fig. 9.1. This behavior is due to the poor convergence of the power iteration, which fails to estimate spectral radius sufficiently fast. However, from the second figure of Algorithm 13 we can see that the wall-clock time needed for the adaptive method steadily declines. After a few more thousand rounds we expect to see the same improvement as in the case of the naive Bandit algorithm. Second, the accuracy of spectral radius estimation is improving with time (see the last figure in Fig. 9.2). Power iteration converges for the chosen family of random linear systems, so we can expect to obtain spectral radius with arbitrary accuracy after a sufficient number of

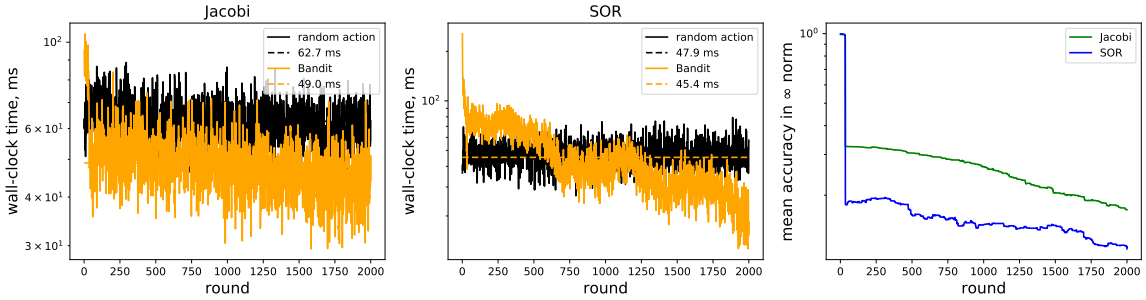


Figure 9.2: Comparison of adaptive (see Algorithm 13) and classical versions (random action is taken for each episode) of Jacobi and SOR iterations. For each method, the mean execution time is specified with the dashed line in the legend. The setup is the same as in Fig. 9.1.

rounds.

9.4.3 Arm exclusion with Bauer-Fike upper bound

In the previous section, we constructed a consistent estimation of the spectral radius. However, we have seen that this estimator converges slower than the naive bandit algorithm. There are two natural strategies to improve the situation.

The first one is to simply combine the naive approach with the restarted one. Namely, we can start with Algorithm 12 when no information about eigenvector of a given arm is available and later switch to Algorithm 13. The resulting combined algorithm has straightforward properties so we do not provide benchmarks and further details.

The second strategy is to use confidence intervals for eigenvalues to completely exclude suboptimal arms as soon as possible. This can be done with Bauer-Fike upper bound [Saa11, Theorem 3.6] which states that for diagonalisable matrix $\boldsymbol{A}$ holds

$$\left|\lambda - \widetilde{\lambda}\right| \leq \frac{\|\boldsymbol{X}\|_2}{\left\|\boldsymbol{X}^{-1}\right\|_2} \frac{\left\|\boldsymbol{A}\widetilde{\boldsymbol{u}} - \widetilde{\lambda}\widetilde{\boldsymbol{u}}\right\|_2}{\|\widetilde{\boldsymbol{u}}\|_2}, \tag{9.6}$$

where $\boldsymbol{X}$ diagonalizes $\boldsymbol{A}$, $\widetilde{\lambda}$ and $\widetilde{\boldsymbol{u}}$ is approximate eigenpair and λ is the exact eigenvalue.

Evidently, Eq. (9.6) is a posteriori error estimate that is easy to compute if the condition number of $\boldsymbol{X}$ is known. For Hermitian matrices $\boldsymbol{A}$ situation is especially simple since $\boldsymbol{X}$ is unitary, so the condition number is 1. In this case situation

Algorithm 14 Arm exclusion with Bauer-Fike upper bound.

Input: approximate eigenpairs $\widetilde{\lambda}_i$, $\widetilde{\boldsymbol{u}}_i$, $i = 1, \ldots, N$, Hermitian matrix $\boldsymbol{A}$.
Output: set $S \subset \{1, \ldots, N\}$ of arms that contains the optimal one.

$i_{\text{opt}} = \arg\min_j \widetilde{\lambda}_j$
$\lambda_{\min}^{\text{u.b.}} = \widetilde{\lambda}_{i_{\text{opt}}} + \left\| \boldsymbol{A}\widetilde{\boldsymbol{u}}_{i_{\text{opt}}} - \widetilde{\lambda}_{i_{\text{opt}}} \widetilde{\boldsymbol{u}}_{i_{\text{opt}}} \right\| / \left\| \widetilde{\boldsymbol{u}}_{i_{\text{opt}}} \right\|$
$S = \{i_{\text{opt}}\}$
for $i = 1, \ldots, N$, $i \neq i_{\text{opt}}$ **do**
    $\lambda_i^{\text{l.b.}} = \widetilde{\lambda}_i + \left\| \boldsymbol{A}\widetilde{\boldsymbol{u}}_i - \widetilde{\lambda}_i \widetilde{\boldsymbol{u}}_i \right\| / \left\| \widetilde{\boldsymbol{u}}_i \right\|$
    if $\lambda_i^{\text{l.b.}} \leq \lambda_{\min}^{\text{u.b.}}$ **then**
        $S \leftarrow S \cup \{i\}$
    end if
end for

simplifies [Saa11, Corollary 3.3]:

$$\left| \lambda - \widetilde{\lambda} \right| \leq \frac{\left\| \boldsymbol{A}\widetilde{\boldsymbol{u}} - \widetilde{\lambda}\widetilde{\boldsymbol{u}} \right\|_2}{\left\| \widetilde{\boldsymbol{u}} \right\|_2}. \tag{9.7}$$

Result Eq. (9.7) can be improved (see [Saa11, Section 3.2.2]) but as we will see it is already sufficient for efficient arm exclusion algorithm.

The algorithm itself is given in Algorithm 14 and consists of comprising of upper bounds for the currently optimal arm with lower bounds for suboptimal arms.

Experiment

We benchmark Eq. (9.7) on random matrices Eq. (9.5) in the same setup described after Eq. (9.5). The results are in Fig. 9.3.

As we can see, Bayer-Fike bound successfully excludes almost all arms after 2000 rounds. This mean (on average) at the end adaptive algorithm Algorithm 13 chooses only between two arms that are indistinguishable close to each other. The accuracy of estimate (Fig. 9.3 last figure) reaches much lower values in comparison with algorithm without arm exclusion (Fig. 9.2) because, when forlorn arms are excluded, algorithm can allot more resources to the remaining ones.

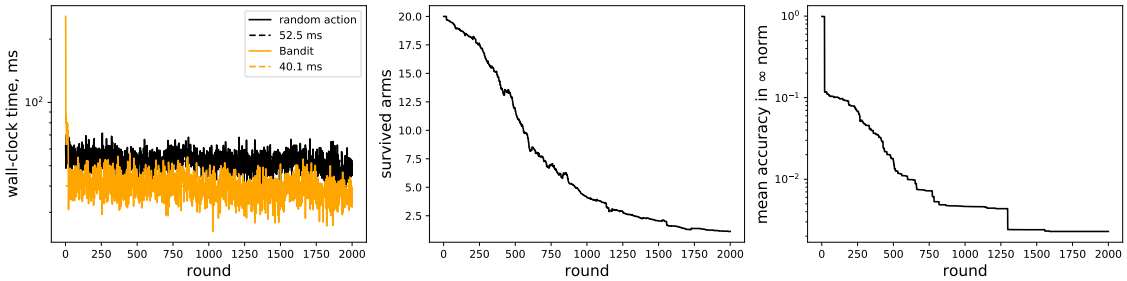


Figure 9.3: Benchmark of restarted $\epsilon-$greedy bandit Algorithm 13 combined with Bauer-Fike arm exclusion Algorithm 14. The first figure shows wall clock time for adaptive algorithm and for the solver that chooses random arm for each round. For each method, the mean execution time is specified with the dashed line in the legend. Second figure contains the average number of arms survived after exclusion step. The third figure contains accuracy of eigenvalue estimation.

9.4.4 Rediscretization

When Algorithm 14 excludes all but one arm, Algorithm 13 has nothing to explore and improve anymore, so the estimation of spectral radius becomes unnecessary. At this stage, we can recall we performed discretization of the continuous space of actions. This implies our estimation of the best action is (in the worst case) the discretization step away from the genuinely optimal action.

So when resources for explorations are exhausted, we can perform rediscretization. Namely, if θ_i is the last remaining arm after Algorithm 13 with Algorithm 14, we can substitute the initial set of arms with uniform discretization of interval $[\theta_{i-1}, \theta_{i+1}]$ that contains optimal arm.

Experiment

We use random matrices Eq. (9.5) to exemplify the performance of rediscretization strategy. The results are in Fig. 9.4.

We can see that the distance to the optimal arm is steadily decreasing. When the arms are closer to each other, algorithm Algorithm 13 needs more time to distinguish between them, that is why the distance on the right graph in Fig. 9.4 slowly saturates. Also, it is interesting to look at the walk-clock time (left panel in Fig. 9.4). The algorithm with rediscretization demonstrates a profound slowdown around 750 episode. The reason for that is, on average 750 is the episode when rediscretiazation occurs. At first, it seems rediscretization should not have this effect, because after it all arms are sufficiently close to the optimal one, so the solution should be

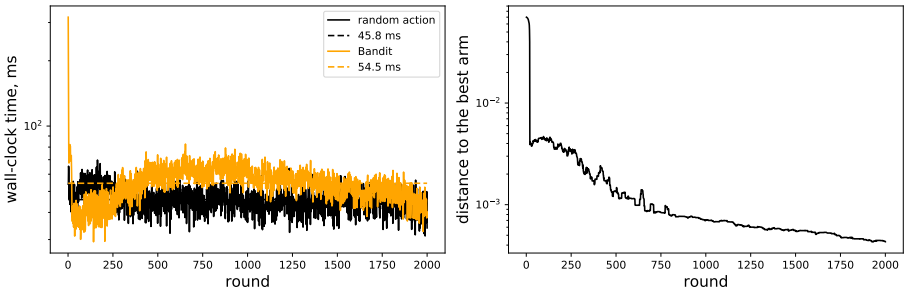


Figure 9.4: Benchmark for composition of Algorithm 13 with Algorithm 14 and rediscretization (see Section 9.4.4). For each method, the mean execution time is specified with the dashed line in the legend. Number of arms is 20 for each discretization. Original spacing of the grid in the space of actions is $\simeq 10^{-2}$.

obtained with fewer iterations. The effect can be explained by two factors. First, we used optimistic initialisation (line 4 in Algorithm 13). Second, we used priority exploration (not shown in the algorithms above) of arms with zero spectral radii. So, when rediscretization is done, the adaptive algorithm performs extensive exploration which leads to low iteration count but high collateral costs associated with spectral radii estimation. That is why wall-clock time increases after rediscretization. It is easy to come by with the algorithm without this undesired behavior, but in this case, convergence to the correct arm is delayed and more rounds are needed to obtain the result evident from the right figure in Fig. 9.4.

9.5 On convergence of proposed algorithms

We will not attempt to provide rigorous proof of convergence and convergence speed in this section. This task is beyond the scope of the current research. However, we will outline what can be done to establish convergence of proposed algorithms.

First, one needs to establish whether the estimate of the spectral radius converges, which is related to the reward in our case. All the novel algorithms (e.g., Algorithm 13) proposed in this section rely on the power iteration. The convergence of the power iteration is well understood (see [Saa11, Theorem 4.1]), so the sufficient condition for convergence is easy to obtain for an individual arm. However, it can be hard to obtain convergence for a set of arms. To do that, one needs to have results like the theorem of Young for SOR (see [Hac16, Theorem 4.27]), i.e., the result on how the spectrum of error propagation matrix depends on the relaxation parameter

(see also [Hac16, Section 4.6.4] for the discussion on how this information can be used to construct adaptive iterative method).

The second ingredient is the convergence of exploration-exploitation scheme (see [ACF02] for the variant of $\epsilon-$greedy bandit). Unfortunately, this is not possible to apply those results directly to relaxation techniques, because the rewards of the arms are not random variables. One way around could be to use Bauer-Fike upper bound Eq. (9.6) and model approximate eigenvalue $\widetilde{\lambda}$ as a uniformly distributed random variable inside the interval defined by the right-hand side of Eq. (9.6).

Lastly, to prove the convergence of the rediscretization approach, one needs to know the local and global properties of the function $\rho(\omega)$ (e.g., Lipshitz constant, smoothness). Similar regularity criteria plays important role in the convergence results for bandits with continuous action spaces [MCP14].

With all the mentioned results combined one should be able to prove the convergence of Algorithm 13 and other algorithms in this part.

9.6 Numerical examples

To show typical behaviour of proposed algorithms we on more serious examples, we perform a series of tests with geometric multigrid method. In all tests we optimize pre- and post-smoothing relaxation parameter in Jacobi iterations. As a stopping criteria we used the drop of relative residual below 10^{-10}. Equations used include 5-point Eq. (1.14) and Mehrstellen Eq. (1.17) discretisations of Poisson equation, Poisson equation with mixed derivative Eq. (1.19) with $2\tau = 2 - \eta$, anisotropic Poisson equation Eq. (1.21) with anisotropy parameter η. To measure the performance of algorithms we use wall-clock time and average spectral radius defined as $\sum_\theta p(\theta)\rho(\boldsymbol{M}(\theta))$.

The first portion of results is shown in Fig. 9.5. We can see (first two columns Fig. 9.5) that both adaptive algorithm consistently produces a better average spectral radius for different problem sizes J. For all problems, the average spectral radius weakly depends on J and slowly saturates.

In the last columns of Fig. 9.5 we can see that the same is true for anisotropy parameters η. However, the efficiency of both adaptive and non-adaptive iterations decreases profoundly for small η. The reason for that is not the inefficiency of the adaptive algorithm but the absence of good solvers among the presented ones, i.e., it is not enough to tune the relaxation parameter to improve the convergence speed for these equations.

Second portion of results is presented in Fig. 9.6. From Fig. 9.6b we can conclude that typically average wall-clock time is going to increase when the number of arms

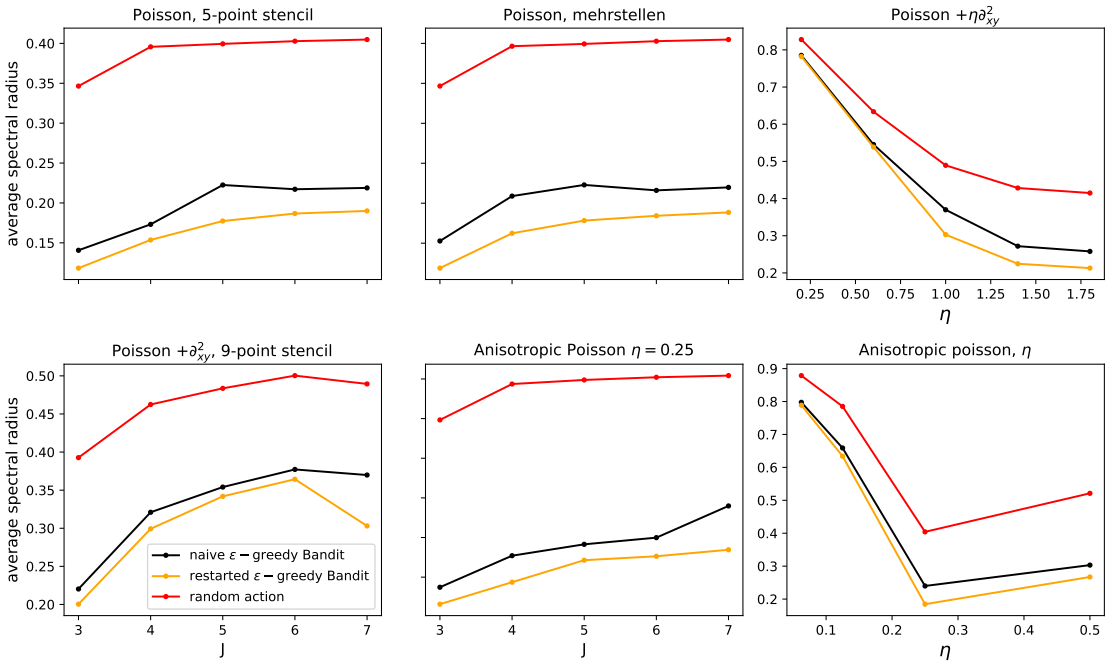
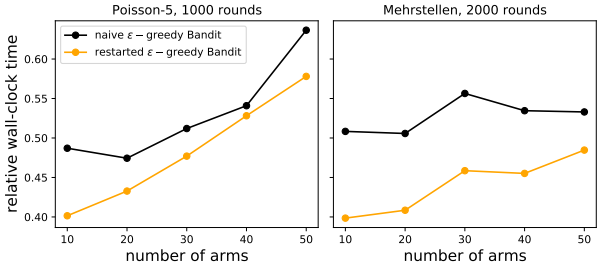


Figure 9.5: Results for Algorithm 13 with arm exclusion Algorithm 14 for four test problems. First two columns show the dependence of average spectral radius on the number of unknowns $N = 2^J$. Last column shows the dependence on the degree of anisotropy (for anisotropic equation) and the strength of mixed terms (for the equation with ∂^2_{xy}). Smaller values of η correspond to a more wayward equation. Number of arms is 50 for all experiments.

	naive	restarted
Poisson-5	0.51	0.44
Mehrstellen	0.49	0.41
∂^2_{xy}	0.64	0.82
Anisotropic	0.48	0.49

(a)



(b)

Figure 9.6: Relative wall-clock time (normalized on the average time needed for algorithm that chooses random action) for maximal number of rounds (left) and as a function of the number of arms.

is increased. This is because for the fixed number of episodes adaptive algorithm needs to spend more time finding the optimal arm. The situation can be improved with rediscretization approach described above.

Fig. 9.6a demonstrates that Algorithm 13 is not uniformly better than Algorithm 12 when wall-clock time is considered. Again, the reason is the fixed number of rounds N_{r}. When $N_{\mathsf{r}} \to \infty$ we can expect the performance of restarted Bandit to excel the performance of Algorithm 12, because the former is consistent whereas the later is not.

Conclusion

To summarize, we discuss the advantages and limitations of approaches (statistical, and machine learning) advocated in the thesis.

Statistical approaches

As we have seen, probabilistic numerical methods provide a unique new way to think about numerical algorithms. They are backed by powerful approaches used in Bayesian statistics and Bayesian decision theory. This allows for the development of new algorithms that, in some cases, are superior to classical ones [Owh15].

The main feature of these algorithms is the uncertainty about the exact solution encoded as a posterior distribution. This distribution can be used to solve inverse problems and forward problems with uncertain parameters (see [Coc+19b], [Coc+19a], [HOG15]). In the latter case, uncertainties can be meaningfully combined with standard rules of probability.

On the downside, the interpretation of the uncertainty is still vague. Probability theory used to quantify epistemic uncertainty focuses on the set of rules that ensures consistency. This means the manipulations with probabilities are uniquely fixed. On the other hand, the probability distribution is not uniquely prescribed. There is no agreement on how one should specify it so that it can be seen as subjective. This poses difficulties for uncertainty quantification.

Another problem is the high numerical cost of probabilistic numerical methods ([Owh15] provide notable exception). It is not always clear that these additional costs are justified. One way to justify them is to argue that well-calibrated probabilities can be combined. However, to the best of our knowledge, current literature lacks definitive examples that show how uncertainty can be combined in computational pipelines.

To recapitulate, probabilistic numerical approaches are often theoretical in spirit. They are seldom numerically attractive in comparison with classical methods. The

probability distribution provided by those methods is an interesting object to study because it provides an alternative to classical error estimation techniques which quite often result in pessimistic estimates.

Machine learning approaches

The machine learning approaches are more practical and traditionally oriented toward improvement in a given benchmark. This way it is possible to construct highly optimized solvers that beat classical algorithms. This was demonstrated in many research manuscripts (e.g., [Hsi+19], [Li+20], [Gre+19a]) including the present one. In our opinion, highly customized solvers are valuable for applications where the same linear system should be solved again and again. One of the examples is the weather forecast. Even a slight optimization of linear solvers in this computationally intensive field can save a lot of energy.

The tendency toward improvement in a given benchmark quite often masks that machine learning approaches tend to be brittle. This brittleness manifests itself in two forms.

First, when parameters of the problem shows "extreme" behaviour (large wavenumbers in Helmholtz problem, large anisotropy coefficient, large jump in the diffusion coefficient) the learning/optimization becomes unreliable and inefficient (see continuation method in [KDO20], discussion in Section 8.6.4). These kind of problems are usually solved by "hybrid" approaches, when classical technique aid machine learning to avoid difficulties with recalcitrant problem (e.g., rescaling in Section 7.3.6, AMG interpolation in [Gre+19a]). Hybrid approaches are legitimate, but being ad hoc only masks the problem.

Second, machine learning approaches often lead to solvers that generalize poorly on the out-of-distribution data. Examples for multigrid are given in Chapter 8. The same was recently demonstrated for neural operators [FO22]. The solution here is, again, to apply hybrid approaches with the same downsides as before.

Despite all the mentioned difficulties, we remain optimistic. Both probabilistic numerical methods and machine learning for numerical linear algebra (and scientific computing in general) approaches are in their infancy. We are definite we will see a lot of new fruitful ideas in years to come.

Part III

Proofs

Chapter 10

Gaussian belief propagation

10.1 Theorem 3.3.1

Here, following [WF00] we prove Theorem 3.3.1. That is, if there is a steady state under mapping (3.23), the solution given by the GaBP rules is exact.

We note that the proof in [WF00] also holds for the nonsymmetric case. We present a slightly different version of their reasoning, without referencing graphical models for normal distribution.

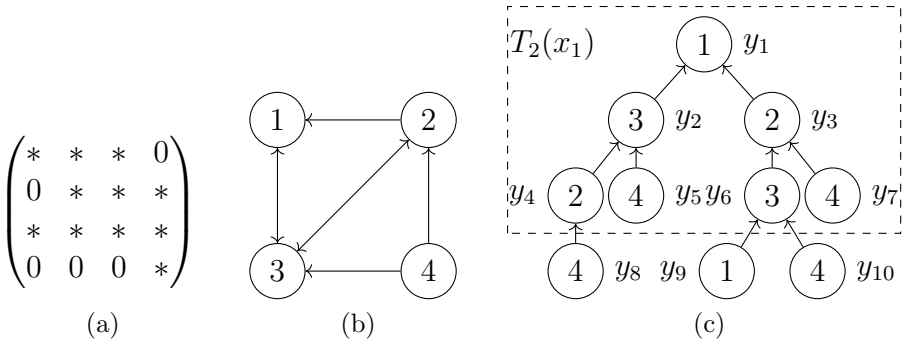
The first concept that we need is a computation tree, which captures the order of operations under the GaBP iteration scheme. The computation tree contains copies of vertices and edges of the graph corresponding to $\mathbf{A}$. The matrix is supposed to be fixed so the computation tree depends on the root node i and the number of steps n. We denote it by $T_n(x_i)$. To obtain $T_n(x_i)$ from $T_{n-1}(x_i)$, we consider each vertex $m \in \mathcal{V}_{T_{n-1}(x_i)}$ that has no incidence edges, find the corresponding variable on the graph of A, add to $T_{n-1}(x_i)$ copies of each neighbour k of m such that $e_{km} \in \mathcal{E}_A$ except for l for which $e_{ml} \in \mathcal{E}_{T_{n-1}(x_i)}$. The example of the tree $T_3(x_1)$ is in Fig. 10.1c, the $T_2(x_1)$ in the dashed box exemplifies the recursion process.

By the construction of the computation tree, the following proposition is true.

Proposition 10.1.1. *If $x_i^{(n)}$ is the solution on the n-th step of the Algorithm 1, then it coincides with the one obtained after the elimination of all variables but x_i (the root) from the computation tree $T_n(x_i)$.*

To relate the matrix $\mathbf{B}$ of the computation tree $T_n(x_i)$ to the matrix $\mathbf{A}$, we define the matrix $\mathbf{O}$ [WF00, eq. 15] that connects original variables with copies

$$\mathbf{y} = \mathbf{O}\mathbf{x}, \mathbf{d} = \mathbf{O}\mathbf{b}, \tag{10.1}$$



(a) (b) (c)

Figure 10.1: (a) – matrix with nonzero elements denoted by $*$; (b) – directed graph corresponding to the matrix. Note that by our convention e_{ij} agrees with A_{ji} not A_{ij}; (c) – computation tree of depth 3 for the first node $T_3(x_1)$ generated by a flood schedule. The subtree inside the box is $T_2(x_1)$.

or, more precisely, y_j is a copy of $x_i \Rightarrow O_{ji} = 1$ and $\sum_i O_{ji} = 1$. For example, matrix $\mathbf{O}$ for the tree in Fig. 10.1c is

$$\mathbf{O}^T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{10.2}$$

Now it is not hard to establish the connection between $\mathbf{B}$ and $\mathbf{A}$ [WF00, eq. 17]

$$\mathbf{BO} + \mathbf{E} = \mathbf{OA}, \tag{10.3}$$

where $\mathbf{E}$ is nonzero only for the subset of variables that n steps away from the root node on the computation tree $T_n(x_i)$. The final part of the proof depends on the following statement [WF00, Periodic beliefs lemma].

Proposition 10.1.2. *If there is $N \in \mathbb{N}$ such that $m_e^{(N+k)} = m_e^{(N)}$, $\widetilde{\Lambda}_e^{(N+k)} = \widetilde{\Lambda}_e^{(N)}$ for all $e \in \mathcal{E}$ and for any $k \in \mathbb{N}$, then it is possible to construct an arbitrary large computation tree $T_M(x_i)$ for any root node x_i such that $\mathbf{O}\boldsymbol{\mu}^{(N)} = \widetilde{\mathbf{B}}^{-1}\widetilde{\mathbf{d}}$. Where $\widetilde{B}_{ij} \neq B_{ij}$ and $\widetilde{d}_i \neq d_i$ only for $i = j$ that are M steps away from the root node.*

The crucial part here is that not only the solution for the root node coincides with the steady state solution of GaBP, but also the same is true for all the variables on the modified computation tree.

The proof is as follows. First, following the recursion procedure, we construct a computation tree of desired depth M. Then we continue to grow the tree till the subtrees of nodes M steps away from the root reach the depth N which corresponds to the steady state of GaBP. Now, elimination of subtrees results in the desired modified tree with the matrix $\widetilde{\mathbf{B}}$ and the right-hand side $\widetilde{\mathbf{d}}$.

Since we can construct an arbitrary modified computation tree, we can always get for arbitrary large M

$$\widetilde{\mathbf{B}}\mathbf{O} = \mathbf{O}\mathbf{A} \text{ for the first } M \text{ rows.} \tag{10.4}$$

And we know that by construction of the modified computation tree

$$\widetilde{\mathbf{B}}\mathbf{O}\boldsymbol{\mu}^{(N)} = \widetilde{\mathbf{d}}. \tag{10.5}$$

So we conclude that

$$\mathbf{O}\mathbf{A}\boldsymbol{\mu}^{(N)} = \mathbf{O}\mathbf{b} \text{ for the first } M \text{ rows.} \tag{10.6}$$

Note, that $\mathbf{O}^T\mathbf{O}$ is a diagonal matrix that counts the number of copies of each variable, therefore we can always choose M large enough to make $\det\left(\mathbf{O}^T\mathbf{O}\right) \neq 0$ and $\mathbf{A}\boldsymbol{\mu}^{(N)} = \mathbf{b}$ which means that the iterative scheme defined by the Algorithm 1 is consistent.

10.2 Theorem 3.3.2

Here we present the version of the proof from [MJW06] that extends to nonsymmetric matrices. Our modifications are relatively minor, but for the sake of logical coherence, we reproduce here the minimal set of arguments from [MJW06] tuning definitions and proposition when needed.

The whole idea of the proof [MJW06] is to relate GaBP operations with the recursive update of the weights of walks on the graph, corresponding to the matrix $\mathbf{A}$. For the start, we define a walk w as a an ordered set of vertices $w = \left(i_1, i_2, \ldots, i_{l(w)}\right)$ where $l(w)$ is a length of the walk w and $\forall k < l(w) \Rightarrow e_{i_k i_{k+1}} \in \mathcal{E}$. Each walk possesses a weight

$$\phi(w) = A_{i_{l(w)} i_{l(w)-1}} \cdots A_{i_3 i_2} A_{i_2 i_1}. \tag{10.7}$$

Note that the order is backward, which is a consequence of our definition of the directed graph. For the symmetric matrix, when the order is not essential, the equation (10.7) coincides with the weight defined in [MJW06, p. 3.1]. Now, if we

have a system $\mathbf{Ax} = \mathbf{b}$, we can rescale it using $\widetilde{b}_j = b_j / A_{jj}$. This procedure is valid for any $\mathbf{A}$ with nonzero diagonal and results in the equivalent system

$$\widetilde{\mathbf{A}}\mathbf{x} = \widetilde{\mathbf{b}},\ \widetilde{A}_{ij} = \delta_{ij} + (1 - \delta_{ij})\frac{A_{ij}}{A_{ii}} \equiv \delta_{ij} - \widetilde{R}_{ij} \tag{10.8}$$

It is possible to represent the solution of (10.8) in the form of Neumann series (see [HJ13, ch. 5]) because $\rho\left(\widetilde{\mathbf{R}}\right) < 1$ and therefore

$$\widetilde{\mathbf{A}}^{-1} = \left(\mathbf{I} - \widetilde{\mathbf{R}}\right)^{-1} = \sum_{n=0}^{\infty} \widetilde{\mathbf{R}}^n. \tag{10.9}$$

However, for being able to rearrange terms in the sum as necessary, which is sufficient to rewrite the inverse matrix using walks, one needs to require absolute convergence which is $\rho(|\widetilde{\mathbf{R}}|) < 1$. Having this condition it is not hard to prove [MJW06, Proposition 1, Proposition 5]

Proposition 10.2.1. *If $\rho(|\widetilde{\mathbf{R}}|) < 1$, then*

$$\widetilde{A}^{-1}_{ij} = \sum_{w:j\to i} \phi(w),\ x^{\star}_i \equiv \left(\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{b}}\right)_i = \sum_{k\in\mathcal{V}} \sum_{w:k\to i} \phi(w)\widetilde{b}_k.$$

Here, by $w : j \to i$ we mean the set of walks which start from the vertex j and end at the vertex i. If one defines [MJW06, p. 3.2] sets of single-visit $k \overset{\setminus i}{\to} i$ and single-revisit $i \overset{\setminus i}{\to} i$ walks by all walks which are not visiting the node i in between given start and end points, the sum over walks can be decomposed [MJW06, eq. 12, 13, Proposition 9]

$$x^{\star}_i = \left(\widetilde{b}_i + \sum_{k\in\mathcal{V}} \left[\widetilde{b}_k \sum_{w:k\overset{\setminus i}{\to} i} \phi(w)\right]\right) \Bigg/ \left(1 - \sum_{w:i\overset{\setminus i}{\to} i} \phi(w)\right). \tag{10.10}$$

The decomposition follows from "topological" considerations alone which depend only on the structure of walks and not on the particular definition of the weight. The last result that we need is [MJW06, Lemma 18]

Proposition 10.2.2. *For each finite length walk $k \to j$ on directed graph of the matrix A there is n and unique walk on the computation tree $T_n(x_i)$.*

Now, if we can relate update rules (3.23) with the recursive structure of walks on a tree, the proof of the Theorem 3.3.2 is done.

On the tree, for each vertex i, the sum over single-revisit walks splits into sums over subtrees $T_{k\cup i}$, which are maximal connected parts that contain i and among $N(i)$, only k. Then

$$\sum_{w:i\overset{\setminus i}{\to}i} \phi(w) = \sum_{k\in N(i)} \sum_{\substack{w:i\overset{\setminus i}{\to}i \\ w\in T_{k\cup i}}} \phi(w), \tag{10.11}$$

but the sums over subtrees $T_{k\cup i}$ can be written as a sum over $T_{k\setminus i} \equiv T_{k\cup i}\setminus\{i\}$,

$$\sum_{\substack{w:i\overset{\setminus i}{\to}i \\ w\in T_{k\cup i}}} \phi(w) = \frac{\widetilde{R}_{ki}\widetilde{R}_{ik}}{1-\sum\limits_{\substack{w:k\overset{\setminus k}{\to}k \\ w\in T_{k\setminus i}}} \phi(w)} = \frac{\widetilde{R}_{ki}\widetilde{R}_{ik}}{1-\sum\limits_{m\in N(k)\setminus i}\ \sum\limits_{\substack{w:k\overset{\setminus k}{\to}k \\ w\in T_{m\cup k}}} \phi(w)}, \tag{10.12}$$

where we used [MJW06, eq. 12, Proposition 9]

$$\sum_{w:k\to k} \phi(w) = \frac{1}{1-\sum\limits_{w:k\overset{\setminus k}{\to}k} \phi(w)}. \tag{10.13}$$

Using the definition of $\widetilde{\mathbf{R}}$, it is easy to see that if one denotes

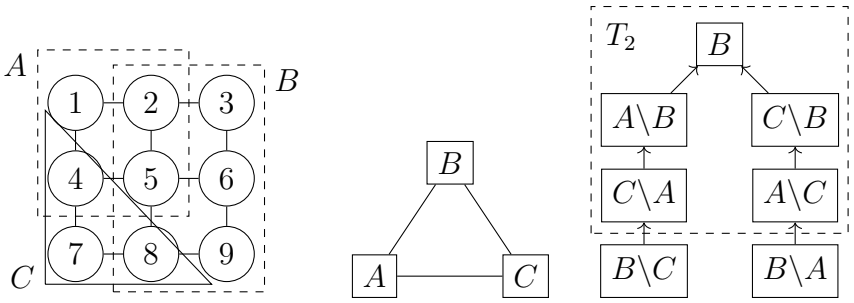
$$-\frac{A_{ii}}{A_{ki}} \sum_{\substack{w:i\overset{\setminus i}{\to}i \\ w\in T_{k\cup i}}} \phi(w) = \widetilde{\Lambda}_{ki}, \tag{10.14}$$

then the update rule (10.12) coincides with the one for $\widetilde{\mathbf{\Lambda}}$ in (3.23). Note that (10.14) is well defined because if $A_{ki}=0$, there is no contribution from this particular subtree, and we do not need to use the walk from there. In the same vein, the sum in the numerator of (10.10) can be decomposed

$$\sum_{k\in\mathcal{V}} \left[\widetilde{b}_k \sum_{w:k\overset{\setminus i}{\to}i} \phi(w)\right] = \sum_{m\in N(i)} \sum_{k\in T_{m\cup i}} \left[\widetilde{b}_k \sum_{\substack{w:k\overset{\setminus i}{\to}i \\ w\in T_{m\cup i}}} \phi(w)\right]. \tag{10.15}$$

Again, using the sum over subtrees $T_{k\setminus i}$

$$\sum_{k\in T_{m\cup i}} \left[\widetilde{b}_k \sum_{\substack{w:k\overset{\setminus i}{\to}i \\ w\in T_{m\cup i}}} \phi(w)\right] = \widetilde{R}_{im} \sum_{k\in T_{m\setminus i}} \left[\widetilde{b}_k \sum_{\substack{w:k\to m \\ w\in T_{m\setminus i}}} \phi(w)\right], \tag{10.16}$$



(a) (b) (c)

Figure 10.2: (a) – partition of the original graph on large regions; (b) – flat representation of the two-layer region graph; (c) – computation tree for the generalized GaBP.

decomposition on single-visit walks [MJW06, eq. 13] and equations (10.14), (10.12), we obtain

$$\begin{aligned}\frac{m_{mi}}{A_{ii}} &\equiv \sum_{k\in T_{m\cup i}} \left[\widetilde{b}_k \sum_{\substack{w: k\xrightarrow{\setminus i} i \\ w\in T_{m\cup i}}} \phi(w) \right] = \\ &= \frac{\widetilde{\Lambda}_{mi} A_{mm}}{A_{ii}} \left(\widetilde{b}_m + \sum_{l\in N(m)\setminus i} \sum_{k\in T_{l\cup m}} \left[\widetilde{b}_k \sum_{\substack{w: k\xrightarrow{\setminus m} m \\ w\in T_{l\cup m}}} \phi(w) \right] \right).\end{aligned} \tag{10.17}$$

The parameterization introduced in (10.17) leads to the same update rule for $\boldsymbol{m}$ as in (3.23). With that, the sufficient condition, given in Theorem 3.3.2, is established.

10.3 Theorem 3.4.1

Here we prove that the generalized GaBP is consistent. The idea of the proof is the same as for the regular GaBP. One needs to relate, considering the operations of generalized GaBP, equations that the generalized GaBP solves during the N-th step, with the original system of linear equations, and then to show that those systems coincide for a sufficiently large N if steady state exists.

To do so, we introduce a flat region graph. The flat region graph is an undirected graph $\{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of large regions (collections of variables in set-decomposition $\mathcal{P}$) and $\left(L, L^{'}\right) \in \mathcal{E}$ if L and $L^{'}$ has at least one common child. The example is in Fig. 10.2b.

Now one can introduce the computation tree exactly in the same way as for GaBP. The only difference is that, because of an overlap between large regions, when we add a leaf node, we include overlapping variables to the root node. An example of the computation tree $T_3(B)$ as well as the $T_2(B)$ is in Fig. 10.2c. By construction of the computation tree, we know that the following is true.

Proposition 10.3.1. *Elimination of all the variables on the computation tree $T_N(B)$ leads to the solution $\mathbf{x}_B$ that coincides with the one on the N-th step of generalized GaBP.*

The relation between the matrix $\mathbf{B}$, corresponding to the computation tree, and the original matrix $\mathbf{A}$ is the same as in the equation (10.3) if one introduces the matrix $\mathbf{O}$

$$O_{ij} = \begin{cases} 1 \text{ if } y_i \text{ is the copy of } x_j, \\ 0 \text{ otherwise.} \end{cases} \tag{10.18}$$

Here, $\mathbf{x}$ are variables on the graph of matrix $\mathbf{A}$, and $\mathbf{y}$ are the ones on the computation tree.

Having the same relation between $\mathbf{A}$ and $\mathbf{B}$, we can repeat the rest of the proof, using the same arguments as in Section 10.1. So it follows that generalized GaBP is consistent and Theorem 3.4.1 is true.

10.4 Theorem 3.4.2

In this section, we present a sufficient condition for the convergence of the two-layer generalized GaBP.

The proof consists of two parts. In the first one, we show that single-visit and single-revisit walks on a tree possess the same update rules as generalized GaBP messages. In the second part, we show that it is always possible to reorganize walks on the graph coming from the partition F (see equations (3.36) and (3.37)) to restore each walk on a computation tree.

10.4.1 Walk structure on a tree

To complete the first part, we define for a given partition F (equation (3.36)) of a matrix $\mathbf{A}$ the weight of a walk $w = (i_1 i_2 \dots i_L)$ by the product of matrices

$$\phi(w) = \widetilde{\mathbf{R}}_{i_L i_{L-1}} \cdots \widetilde{\mathbf{R}}_{i_3 i_2} \widetilde{\mathbf{R}}_{i_2 i_1}. \tag{10.19}$$

In the view of the standard result [Ama+05, ch. 8, Theorem 8.9] on absolute convergence in complete finite metric spaces it is possible to rearrange terms of the sum, such that we can formulate the following statement.

Proposition 10.4.1. *If* $\rho\left(\left\|\widetilde{\mathbf{R}}\right\|\right) < 1$, *then*

$$\left(\widetilde{\mathbf{A}}^{-1}\right)_{ii} = \sum_{n=0}^{\infty} \left(\widetilde{\mathbf{R}}^n\right)_{ii} = \sum_{w: i \to i} \phi(w), \;\; \mathbf{x}_i \equiv \sum_{j \in \mathcal{V}} \left(\widetilde{\mathbf{A}}^{-1}\right)_{ij} \widetilde{\mathbf{b}}_j = \sum_{j \in \mathcal{V}} \sum_{w: j \to i} \phi(w) \widetilde{\mathbf{b}}_j.$$

Here we used the same definition for the set of walks as in the Section 10.2. Again, [MJW06, eq. 12, 13] allows us to rewrite the diagonal blocks of the inverse matrix and the solution vector using single-visit and single-revisit walks

$$\begin{aligned} \left(\widetilde{\mathbf{A}}^{-1}\right)_{ii} &= \left(\mathbf{I}_{ii} - \sum_{w: i \overset{\setminus i}{\to} i} \phi(w)\right)^{-1}, \\ \mathbf{x}_i &= \left(\widetilde{\mathbf{A}}^{-1}\right)_{ii} \left(\widetilde{\mathbf{b}}_i + \sum_{j \in \mathcal{V}} \sum_{w: j \overset{\setminus i}{\to} i} \phi(w) \widetilde{\mathbf{b}}_j\right). \end{aligned} \tag{10.20}$$

On the tree we can split the sums over contributions from subtrees $T_{k \cup i}$ for each $k \in N(i)$. Therefore, from comparison with Algorithm 2, we can deduce that

$$\sum_{j \in T_{k \cup i}} \sum_{\substack{w: j \overset{\setminus i}{\to} i \\ w \in T_{k \cup i}}} \phi(w) \widetilde{\mathbf{b}}_j = \left(\mathbf{A}_{ii}\right)^{-1} \mathbf{m}_{ki}, \quad \sum_{\substack{w: i \overset{\setminus i}{\to} i \\ w \in T_{k \cup i}}} \phi(w) = -\left(\mathbf{A}_{ii}\right)^{-1} \mathbf{\Lambda}_{ki}. \tag{10.21}$$

Messages in Algorithm 2 propagate along edges of the region graph, whereas messages that we have just defined flow along edges of a graph of the matrix $\widetilde{\mathbf{R}}$. To have a more straightforward connection between them, we consider $\widetilde{\mathbf{R}}$ as a matrix originates from the computation tree itself. Under this set of circumstances, there is a one-to-one correspondence between messages (10.21) and the ones in Algorithm 2.

For single-revisit walks, one has

$$\sum_{\substack{w:i\overset{\setminus i}{\to}i\\ w\in T_{k\cup i}}}\phi(w)=(\mathbf{A}_{ii})^{-1}\mathbf{A}_{ik}\sum_{\substack{w:k\to k\\ w\in T_{k\setminus i}}}\phi(w)\,(\mathbf{A}_{kk})^{-1}\mathbf{A}_{ki}=$$

$$=(\mathbf{A}_{ii})^{-1}\mathbf{A}_{ik}\left(\mathbf{I}_{kk}-\sum_{\substack{w:k\overset{\setminus k}{\to}k\\ w\in T_{k\setminus i}}}\phi(w)\right)^{-1}(\mathbf{A}_{kk})^{-1}\mathbf{A}_{ki}\Rightarrow \tag{10.22}$$

$$\Rightarrow\mathbf{\Lambda}_{ki}=-\mathbf{A}_{ik}\left(\mathbf{A}_{kk}+\sum_{m\in N(k)\setminus i}\mathbf{\Lambda}_{mk}\right)^{-1}\mathbf{A}_{ki}.$$

Consider the following matrix

$$\left(\left(\left(\begin{array}{cc}\boldsymbol{X} & \mathbf{A}_{ik}\\ \mathbf{A}_{ki} & \left(\mathbf{A}_{kk}+\sum\limits_{m\in N(k)\setminus i}\mathbf{\Lambda}_{mk}\right)\end{array}\right)^{-1}\right)_{ii}\right)^{-1}-\boldsymbol{X}=\mathbf{\Lambda}_{ki}, \tag{10.23}$$

where $\boldsymbol{X}$ is arbitrary conformable invertible matrix. Equation (10.23) has precisely the same form as a $\mathbf{\Lambda}_{vu}$ in line 16 of Algorithm 2. Using standard identities for block matrix inversion one can easy convince herself that (10.23) provides the same update rule as (10.22).

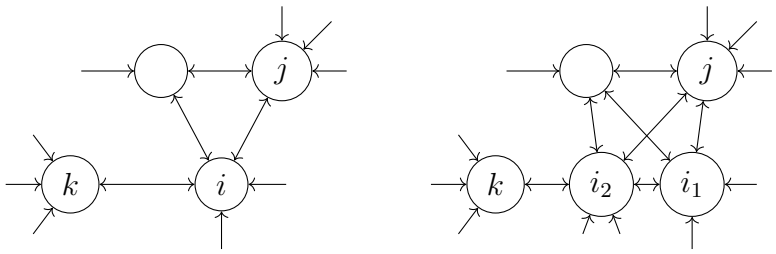
For single-visit walks, we have

$$\sum_{j\in T_{k\cup i}}\sum_{\substack{w:j\overset{\setminus i}{\to}i\\ w\in T_{k\cup i}}}\phi(w)\widetilde{\mathbf{b}}_j=-(\mathbf{A}_{ii})^{-1}\mathbf{A}_{ik}\sum_{j\in T_{k\setminus i}}\sum_{\substack{w:j\to k\\ w\in T_{k\setminus i}}}\phi(w)\widetilde{\mathbf{b}}_j=$$

$$=-(\mathbf{A}_{ii})^{-1}\mathbf{A}_{ik}\left(\mathbf{I}_{kk}-\sum_{\substack{w:k\overset{\setminus k}{\to}k\\ w\in T_{k\setminus i}}}\phi(w)\right)^{-1}\left(\widetilde{\mathbf{b}}_k+\sum_{j\in T_{k\setminus i}}\sum_{\substack{w:j\overset{\setminus k}{\to}k\\ w\in T_{k\setminus i}}}\phi(w)\widetilde{\mathbf{b}}_j\right), \tag{10.24}$$

or using (10.21), we get

$$\mathbf{m}_{ki}=-\mathbf{A}_{ik}\left(\mathbf{A}_{kk}+\sum_{m\in N(k)\setminus i}\mathbf{\Lambda}_{mk}\right)^{-1}\left(\mathbf{b}_k+\sum_{p\in N(k)\setminus i}\mathbf{m}_{pk}\right). \tag{10.25}$$



(a) (b)

Figure 10.3: (a) – graph of the matrix $\mathbf{A}$, each node corresponds to the diagonal block; (b) – refined version of (a), submatrix $\mathbf{A}_{ii}$ is split by four blocks $\mathbf{A}_{i_1i_1}, \mathbf{A}_{i_1i_2}, \mathbf{A}_{i_2i_1}, \mathbf{A}_{i_2i_2}$.

Now, consider the expression

$$\boldsymbol{\Delta}_i \equiv \left(\begin{pmatrix} \boldsymbol{X} & \mathbf{A}_{ik} \\ \mathbf{A}_{ki} & \left(\mathbf{A}_{kk} + \sum\limits_{m \in N(k) \setminus i} \boldsymbol{\Lambda}_{mk} \right) \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{y} \\ \mathbf{b}_k + \sum\limits_{p \in N(k) \setminus i} \mathbf{m}_{pk} \end{pmatrix} \right)_i, \tag{10.26}$$

where $\boldsymbol{X}$ is arbitrary conformable invertible matrix and $\boldsymbol{y}$ is arbitrary conformable vector. Equation (10.26) corresponds to $(\boldsymbol{x}_v)_u$ in line 16 of Algorithm 2. Note that the first matrix in equation (10.23), i.e., $\boldsymbol{\Lambda}_{ki} + \boldsymbol{X}$, is analogous to $((\boldsymbol{\Lambda}_v)_u)^{-1}$ in line 16 of Algorithm 2. Using standard identities for block matrix inversion we find

$$(\boldsymbol{\Lambda}_{ki} + \boldsymbol{X})\, \boldsymbol{\Delta}_i - \boldsymbol{y} = \boldsymbol{m}_{ki}, \tag{10.27}$$

which finishes the proof of updated rules (3.32).

10.4.2 Walk-sums and the graph refinement

The second part of the proof establishes the connection between sets of walks on the graph of the matrix $\widetilde{\mathbf{R}}$ and walks on the computation tree. First, for the matrix (3.37) we split a single region i into two parts i_1 and i_2

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{i_1i_1} & \mathbf{A}_{i_1i_2} & \mathbf{A}_{i_1j} & \dots \\ \mathbf{A}_{i_2i_1} & \mathbf{A}_{i_2i_2} & \mathbf{A}_{i_2j} & \dots \\ \mathbf{A}_{ji_1} & \mathbf{A}_{ji_2} & \mathbf{A}_{jj} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \mathbf{b}_{i_1} \\ \mathbf{b}_{i_2} \\ \mathbf{b}_j \\ \vdots \end{pmatrix}. \tag{10.28}$$

The transformation of the graph is in Fig. 10.3. We refer to this procedure as to the elementary refinement of the region i. From the construction of the refined matrix $\mathbf{A}$, the following proposition holds.

Proposition 10.4.2. *There is a one-to-one correspondence between walks on the graph of $\widetilde{\mathbf{R}}$ and the one obtained by the elementary refinement of the region i excluding three situations: 1) walk crosses i, 2) walk ends at i, 3) walk starts at i.*

We discuss each of these situations separately. First, we need to introduce a new notation. Let $k \xrightarrow{M} l$ be the set of walks, where each walk starts from k, ends at l and newer leaves the subset M. It is easy to see that on the refined graph

$$\phi\left(k \xrightarrow{\{i_1,i_2\}} l\right) = \left((\mathbf{A}_{ii})^{-1}\right)_{lk} \mathbf{A}_{kk}, \text{ where } l,k = \{i_1,i_2\}\,. \tag{10.29}$$

- Walk on $\widetilde{\mathbf{R}}$ that crosses i has a form $w_{\text{cross}} = (\dots jik \dots)$ (see Fig. 10.3a). The weight of this walk is

$$\phi(w_{\text{cross}}) = \cdots (\mathbf{A}_{kk})^{-1} \mathbf{A}_{ki} (\mathbf{A}_{ii})^{-1} \mathbf{A}_{ij} \cdots\,. \tag{10.30}$$

  On the refined graph we can consider the set of all walks that coincides with w outside i. The sum of weight of all these walks is

$$\phi(w)_{\text{refined}} = \sum_{l,k\in\{i_1,i_2\}} \cdots (\mathbf{A}_{kk})^{-1} \mathbf{A}_{kl} \phi\left(l \xrightarrow{\{i_1,i_2\}} k\right) (\mathbf{A}_{ll})^{-1} \mathbf{A}_{lj} \cdots\,. \tag{10.31}$$

  We see that due to equation (10.29), weights are the same.

- Walk on $\widetilde{\mathbf{R}}$ that ends at i has a form $w_{\text{end}} = (\dots ji)$ and a weight

$$\phi(w) = (\mathbf{A}_{ii})^{-1} \mathbf{A}_{ij} \cdots\,. \tag{10.32}$$

  On the refined graph we have two set of walks

$$\phi(w)^p_{\text{refined}} = \sum_{l\in\{i_1,i_2\}} \phi\left(l \xrightarrow{\{i_1,i_2\}} p\right) (\mathbf{A}_{ll})^{-1} \mathbf{A}_{lj} \cdots,\ p \in \{i_1,i_2\} \tag{10.33}$$

  that can be combined to have the same weight. Namely, using (10.29) we find that

$$\left[(\mathbf{A}_{ii})^{-1} \mathbf{A}_{ij} \cdots\right]_{l\star} = \left(\phi(w)^l_{\text{refined}}\right)_\star,\ l = \{i_1,i_2\}\,. \tag{10.34}$$

- Walk on $\widetilde{\mathbf{R}}$ that starts at i has a form $w_{\text{start}} = (ij\dots)$ and a weight

$$\phi(w_{\text{start}}) = \cdots (\mathbf{A}_{jj})^{-1} \mathbf{A}_{ji}. \tag{10.35}$$

  It is possible to relate this walk to two sets of walks $w_1 = (i_1 j\dots)$, $w_2 = (i_2 j\dots)$ on the refined graph multiplying by the corresponding inverse matrices

$$\left(\phi(w_{\text{start}}) (\mathbf{A}_{ii})^{-1}\right)_{\star l} = \sum_{k=\{i_1,i_2\}} \left(\phi(w_k)\phi\left(l \overset{\{i_1,i_2\}}{\longrightarrow} k\right) (\mathbf{A}_{ll})^{-1}\right)_{\star}, \tag{10.36}$$

  where $l = \{i_1, i_2\}$. The re-weight is needed because the original linear system and the refined one are multiplied by different block diagonal matrices and have different inverses.

We know the following two propositions to be true.

Proposition 10.4.3. *Any computation tree can be, by the set of elementary refinements, turned to a computation tree of GaBP under a proper schedule (see discision before [MJW06, Lemma 18]) operating on the graph of the matrix* (3.37) *partitioned according to* F.

Proposition 10.4.4. *For each walk on the graph of the matrix* (3.37)*, there is a unique walk on a sufficiently large computation tree formed by a proper schedule.*

Hence for each walk on the computation tree, it is always possible to find a unique set of walks on the graph of the matrix (3.37) that has the same weight after the multiplication by an appropriate inverse matrix (see (10.36). It allows us to conclude that if it is possible to define a walk-sum for matrix (3.37) (see proposition Proposition 10.4.1), walk-sum on the computation tree converges too, so Theorem 3.4.2 is proven.

Chapter 11

Probabilistic projection methods

11.1 Lemma 4.2.1

General result [Bar+19] for mean and covariance are

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m \left(\boldsymbol{S}_m^T \boldsymbol{A} \boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m\right)^{-1} \boldsymbol{S}_m^T \left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0\right),$$
$$\boldsymbol{\Sigma}_m = \boldsymbol{\Sigma}_0 - \boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m \left(\boldsymbol{S}_m^T \boldsymbol{A} \Sigma_0 \boldsymbol{A}^T \boldsymbol{S}_m\right)^{-1} \boldsymbol{S}_m^T \boldsymbol{A} \boldsymbol{\Sigma}_0.$$

Matrix $\boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m$ and its transpose appear frequently in $\boldsymbol{x}_m$ and $\boldsymbol{\Sigma}_m$. For a chosen covariance matrix $\boldsymbol{\Sigma}_0$ this combination has a simple form

$$\boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m = \left(\boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{\Psi}\right) \boldsymbol{A}^T \boldsymbol{W} = \boldsymbol{V} \left(\boldsymbol{V}^T \boldsymbol{A}^T \boldsymbol{W}\right), \tag{11.1}$$

where the second equality follows from the condition $\boldsymbol{W}^T \boldsymbol{A} \boldsymbol{\Psi} = 0$. Using this form of $\boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m$ we find

$$\left(\boldsymbol{S}_m^T \boldsymbol{A} \boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m\right)^{-1} = \left(\boldsymbol{V}^T \boldsymbol{A}^T \boldsymbol{W}\right)^{-1} \left(\boldsymbol{W}^T \boldsymbol{A} \boldsymbol{V}\right)^{-1}. \tag{11.2}$$

This implies that the second part of the covariance matrix simplifies as follows

$$\boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m \left(\boldsymbol{S}_m^T \boldsymbol{A} \boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m\right)^{-1} \boldsymbol{S}_m^T \boldsymbol{A} \boldsymbol{\Sigma}_0 = \boldsymbol{V}\boldsymbol{V}^T, \tag{11.3}$$

from which we conclude that

$$\boldsymbol{\Sigma}_m = \boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{\Psi} - \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{\Psi}. \tag{11.4}$$

In the same vein, using

$$\boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m \left(\boldsymbol{S}_m^T \boldsymbol{A} \boldsymbol{\Sigma}_0 \boldsymbol{A}^T \boldsymbol{S}_m\right)^{-1} \boldsymbol{S}_m^T = \boldsymbol{V} \left(\boldsymbol{W}^T \boldsymbol{A} \boldsymbol{V}\right)^{-1} \boldsymbol{W}^T \tag{11.5}$$

we can obtain $\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V} \left(\boldsymbol{W}^T \boldsymbol{A} \boldsymbol{V}\right)^{-1} \boldsymbol{W}^T (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0)$ for the mean vector.

11.2 Lemma 4.2.2

Note, that $\left|\mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)\right| = n - \left|\mathsf{Range}\left(\boldsymbol{A}^T\boldsymbol{W}\right)\right| = n - m$. The last equality follows from the fact that $\boldsymbol{A}^T$ is invertible, so $\boldsymbol{W}$ and $\boldsymbol{A}^T\boldsymbol{W}$ has the same rank. From this we conclude that there are exactly $n - m$ linearly independent vectors that span $\mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$. Stacking $k \leq n - m$ of them together we can construct $\boldsymbol{Y}$.

11.3 Lemma 4.2.3

It is easy to see that $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}$ is invertible iff no vector from $\boldsymbol{A}\mathcal{K} = \mathsf{Range}\left(\boldsymbol{A}\boldsymbol{V}\right)$ is orthogonal to $\mathcal{L} = \mathsf{Range}\left(\boldsymbol{W}\right)$. Vectors $\boldsymbol{Y}_{\star i}$, where $i \leq n - m$, form basis for $\mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$, whereas m vectors $\boldsymbol{V}_{\star i} \notin \mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$, hence $\boldsymbol{V}_{\star i} \in \mathsf{Range}\left(\boldsymbol{A}^T\boldsymbol{W}\right)$. By definition $\boldsymbol{V}_{\star i}$ are linearly independent, so they form a basis for $\mathsf{Range}\left(\boldsymbol{A}^T\boldsymbol{W}\right)$. According to a fundamental result of linear algebra $\mathbb{R}^n = \mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right) \cup \mathsf{Range}\left(\boldsymbol{A}^T\boldsymbol{W}\right)$, which means columns of $\boldsymbol{V}$ and $\boldsymbol{Y}$ form a basis for $\mathbb{R}^n$.

11.4 Theorem 4.2.2

1. Both random variables are normal, so it is sufficient to demonstrate that first two moments are equal. Substitution of $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{V}\boldsymbol{v}_0 + \boldsymbol{A}\boldsymbol{x}_0$ into the definition of general projection method (1.5) gives us $\boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{v}_0$ which is a mean of random variable $\boldsymbol{x}^\star$ given $\boldsymbol{v} = \boldsymbol{v}_0$. Covariance matrices coincide as a consequence of Theorem 4.2.1 and definition of $\boldsymbol{x}^\star$.

2. After the projection step, arbitrary sample of random variable $\boldsymbol{v}$ is completely specified, because $\boldsymbol{V}\boldsymbol{v} \in \mathsf{Range}(\boldsymbol{V})$. Namely, $\widetilde{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{v}$, which implies $p(\boldsymbol{x}^\star - \widetilde{\boldsymbol{x}}) = \mathcal{N}(\cdot|0, \boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T)$. Now, since $\boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T$ is positive semidefinite, it is always possible to find a full-rank matrix $\boldsymbol{X} \in \mathbb{R}^{n\times k}$, where $k = \mathsf{rank}(\boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T)$ such that $\boldsymbol{X}\boldsymbol{X}^T$ coincides with $\boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T$. It is easy to check that

$$\left(\boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T\right)^\dagger = \left(\boldsymbol{X}\boldsymbol{X}^T\right)^\dagger = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-2}\boldsymbol{X}^T. \tag{11.6}$$

   Since $\boldsymbol{x}^\star - \widetilde{\boldsymbol{x}} = \boldsymbol{X}\boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is a standard multivariate normal random variable, we can find that test statistic

$$\left\|\boldsymbol{x}^\star - \widetilde{\boldsymbol{x}}\right\|^2_{\left(\boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T\right)^\dagger} = \boldsymbol{\delta}^T\boldsymbol{X}^T\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-2}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\delta} = \boldsymbol{\delta}^T\boldsymbol{\delta} \tag{11.7}$$

   follows χ^2_{n-m} distribution.

11.5 Lemma 4.2.4

Since $\boldsymbol{x}^{\star} = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{\delta}_1 + \boldsymbol{Y}\boldsymbol{G}^{1/2}\boldsymbol{\delta}$, where $\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}$ are independent standard multivariate normal variables, an error $\widetilde{\boldsymbol{e}}$ after the projection step (1.5) is $\boldsymbol{Y}\boldsymbol{G}^{1/2}\boldsymbol{\delta}$. Using the definition of the acute angle θ (see Fig. 4.1), and orthogonal projector $\boldsymbol{P}_{\perp} = \sum_{i=1}^{p} \boldsymbol{Y}_{\star i}\boldsymbol{Y}_{\star i}^T$ on subspace spanned by vectors $\boldsymbol{Y}_{\star i},\ i = 1, \ldots, p$ we can show that

$$\begin{aligned} \cos(\theta) &= \frac{\widetilde{e}^T \left(\sum_{i=1}^{p} \boldsymbol{Y}_{\star i}\boldsymbol{Y}_{\star i}^T\right) \widetilde{\boldsymbol{e}}}{\widetilde{\boldsymbol{e}}^T\widetilde{\boldsymbol{e}}} \\ &= \frac{s^2 \sum_{i=1}^{p} \delta_i^2}{\sum_{i=p+1}^{n-m-p} \delta_i^2 + s^2 \sum_{i=1}^{p} \delta_i^2} = \frac{1}{1 + \frac{\chi^2_{n-m-p}}{s^2\chi^2_p}} \end{aligned} \tag{11.8}$$

Since $z = \left(p\chi^2_{n-m-p}\right) / \left((n-m-p)\chi^2_p\right)$ is F-distributed (see Chapter 13 in [Kri16]) the proof is complete.

11.6 Theorem 4.2.3

1. It is enough to demonstrate that $\boldsymbol{I} - \boldsymbol{P}_1$ is a projection operator. Indeed, if this is the case, $\boldsymbol{P}_1$ is a projection operator too since $(\boldsymbol{I} - \boldsymbol{P}_1)^2 = \boldsymbol{I} - \boldsymbol{P}_1$ implies that $\boldsymbol{P}_1^2 = \boldsymbol{P}_1$. Using $\boldsymbol{I} - \boldsymbol{P}_1 = \left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T\boldsymbol{A}$ for $(\boldsymbol{I} - \boldsymbol{P}_1)^2$ we find

$$\begin{aligned} &\boldsymbol{V}\left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T\boldsymbol{A} \\ &\qquad = \boldsymbol{V}\left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{P}_1, \end{aligned} \tag{11.9}$$

   so $\boldsymbol{I} - \boldsymbol{P}_1$ is a projection operator.

2. It is easy to see that $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{P}_1 = 0$. Indeed,

$$\begin{aligned} &\boldsymbol{W}^T\boldsymbol{A}\left(\boldsymbol{I} - \boldsymbol{V}\left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T\boldsymbol{A}\right) \\ &\qquad = \boldsymbol{W}^T\boldsymbol{A} - \boldsymbol{W}^T\boldsymbol{A} = 0 \end{aligned} \tag{11.10}$$

   From $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{P}_1 = 0$ we have $\mathsf{Range}(\boldsymbol{P}) \subseteq \mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A})$. On the other hand $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = 0 \Rightarrow \boldsymbol{P}_1\boldsymbol{x} = \boldsymbol{x}$, so $\mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A}) \subseteq \mathsf{Range}(\boldsymbol{P})$. From two inclusions we conclude that $\mathsf{Range}(\boldsymbol{P}) = \mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A})$.

3. Any $\boldsymbol{\Psi} \geq 0$ from Lemma 4.2.2 has a form $\boldsymbol{Y}\boldsymbol{Y}^T$ where columns of $\boldsymbol{Y}$ belong to $\mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A})$. This fact follows from spectral decomposition of $\boldsymbol{\Psi}$, $\boldsymbol{\Psi} \geq 0$ and

$\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{\Psi} = 0$. Since $\mathsf{Range}(\boldsymbol{P}_1) = \mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$ we know that $\boldsymbol{P}_1\boldsymbol{Y} = \boldsymbol{Y}$. This allows us to take $\boldsymbol{G} = \boldsymbol{Y}\boldsymbol{Y}^T$ for which the covariance matrix reads

$$\begin{aligned}\boldsymbol{\Sigma}_0 &= \boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{P}_1\boldsymbol{Y}\left(\boldsymbol{P}_1\boldsymbol{Y}\right)^T \\ &= \boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{Y}\boldsymbol{Y}^T = \boldsymbol{V}\boldsymbol{V}^T + \boldsymbol{\Psi}.\end{aligned} \tag{11.11}$$

So with the appropriate choice of $\boldsymbol{G}$ we can reproduce arbitrary covariance matrix from Lemma 4.2.2.

11.7 Theorem 4.2.4

1. $\boldsymbol{P}_2^2 = \boldsymbol{Y}\left(\boldsymbol{Y}^T\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^T\boldsymbol{Y}\left(\boldsymbol{Y}^T\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^T = \boldsymbol{P}_2$, so $\boldsymbol{P}_2$ is a projection operator. Next, $\boldsymbol{P}_2^T = \boldsymbol{P}_2$ so $\boldsymbol{P}_2$ is an orthogonal projector. Finally, $\mathsf{Range}(\boldsymbol{P}_2) = \mathsf{Null}\left(\boldsymbol{W}^T\boldsymbol{A}\right)$ by definition of $\boldsymbol{Y}$.

2. From $\mathsf{Range}(\boldsymbol{P}) = \mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A})$ it follows that $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{P}_2 = 0$, and $\boldsymbol{\Sigma}_0\boldsymbol{A}^T\boldsymbol{W} = \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{A}^T\boldsymbol{W}$. Since the proof of Lemma 4.2.1 relies only on the fact that $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{\Psi} = 0$, we can substitute $\boldsymbol{\Psi}$ by $\boldsymbol{P}_2$ and obtain the same result. With that we conclude that the posterior distribution has a probability density $\mathcal{N}(\cdot|\widetilde{\boldsymbol{x}}, \boldsymbol{P}_2\boldsymbol{G}\boldsymbol{P}_2^T)$.

11.8 Lemma 4.3.1

The result is a slight generalisation of a standard Bayesian hierarchical modelling for multivariate normal distribution [BS09]. Using definition of inverse-gamma distribution and probability density function of multivariate normal distribution we obtain

$$\begin{aligned}&p(\boldsymbol{x}|s, \boldsymbol{\Sigma}, \boldsymbol{\mu})p(s|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}s^{-(\alpha+1)}\exp(-\beta/s) \\ &\frac{\exp\left(-(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^\dagger(\boldsymbol{x}-\boldsymbol{\mu})/(2s)\right)}{(2\pi s)^{k/2}\sqrt{\det \boldsymbol{D}}}\mathsf{Ind}\left[\boldsymbol{U}^T\boldsymbol{x} \neq \boldsymbol{U}^T\boldsymbol{\mu}\right] \\ &= \frac{\beta^\alpha}{\Gamma(\alpha)}\frac{\Gamma(\overline{\alpha})}{\overline{\beta}^{\overline{\alpha}}}\frac{\mathsf{IG}\left(s|\overline{\alpha}, \overline{\beta}\right)\mathsf{Ind}\left[\boldsymbol{U}^T\boldsymbol{x} \neq \boldsymbol{U}^T\boldsymbol{\mu}\right]}{(2\pi)^{k/2}\sqrt{\det \boldsymbol{D}}},\end{aligned} \tag{11.12}$$

where $\overline{\alpha} = \alpha + k/2$, $\overline{\beta} = \beta + (\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^\dagger(\boldsymbol{x}-\boldsymbol{\mu})/2$. Probability density function $\mathsf{IG}\left(s|\overline{\alpha}, \overline{\beta}\right)$ disappears after integration, and it is easy to see that the remaining factors form $\mathsf{St}_{2\alpha}\left(\boldsymbol{x}\,\middle|\boldsymbol{\mu}, \frac{\beta}{\alpha}\boldsymbol{\Sigma}\right)$.

11.9 Lemma 4.3.2

1. We define random variable $\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x}$. Since $\boldsymbol{x} = \boldsymbol{x}_0 + s^{1/2}\boldsymbol{V}\boldsymbol{\delta}_1 + s^{1/2}\boldsymbol{\Psi}^{1/2}\boldsymbol{\delta}_2$, where $\boldsymbol{\delta}_i,\ i = 1, 2$ are independent standard multivariate normal random variables and $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{\Psi}^{1/2} = 0$, probability density function for $\boldsymbol{z}$ reads

$$p(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z}|\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x}_0, s\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^T\right). \tag{11.13}$$

Using definition of posterior distribution we find

$$\begin{aligned} &p\left(s|\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}\right) \propto p(\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{b})\mathsf{IG}(s|\alpha, \beta) \\ &\propto s^{-m/2}\exp\left(-\boldsymbol{\delta}^T\boldsymbol{\delta}/(2s)\right)s^{-(\alpha+1)}\exp\left(-\beta/s\right), \end{aligned} \tag{11.14}$$

where $\boldsymbol{\delta} = \left(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\right)^{-1}\boldsymbol{W}^T\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0\right)$. From the last line we can identify parameters of the posterior distribution $\widetilde{\alpha} = \alpha + m/2$, $\widetilde{\beta} = \beta + \boldsymbol{\delta}^T\boldsymbol{\delta}/2$.

2. Predictive distribution is

$$\begin{aligned} &p\left(\boldsymbol{x}\left|\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}\right.\right) \\ &= \int ds\ p\left(\boldsymbol{x}\left|\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}, \boldsymbol{x}_0, s\boldsymbol{\Sigma}_0\right.\right)p(s|\alpha, \beta), \end{aligned} \tag{11.15}$$

where the first factor under the integral is multivariate normal $\mathcal{N}\left(\boldsymbol{x}|\widetilde{\boldsymbol{x}}, s\boldsymbol{\Psi}\right)$ (see Theorem 4.2.3), and the second is $\mathsf{IG}(s|\widetilde{\alpha}, \widetilde{\beta})$. Using the result from Lemma 4.3.1 we obtain $\mathsf{St}_{2\widetilde{\alpha}}\left(x|\widetilde{x}, \frac{\widetilde{\beta}}{\widetilde{\alpha}}\boldsymbol{\Psi}\right)$ as a predictive distribution.

11.10 Lemma 4.3.3

1. We define random variable $\boldsymbol{z} = \boldsymbol{P}_1(\boldsymbol{x} - \boldsymbol{x}_0)$. To find probability density function of $\boldsymbol{z}$ we use three facts. First, $\boldsymbol{P}_1\boldsymbol{V} = 0$, which follows from definition of $\boldsymbol{P}_1$. Second, because $\mathsf{Range}(\boldsymbol{P}_2) = \mathsf{Range}(\boldsymbol{P}_1)$, we conclude that $\boldsymbol{P}_1\boldsymbol{P}_2 = \boldsymbol{P}_2$. Finally, $\boldsymbol{x} = \boldsymbol{x}_0 + s^{1/2}\boldsymbol{V}\boldsymbol{\delta}_1 + s^{1/2}\boldsymbol{P}_2\boldsymbol{\delta}_2$, where $\boldsymbol{\delta}_i,\ i = 1, 2$ are independent standard multivariate normal distributions. Using these three facts we find $p(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z}|0, s\boldsymbol{P}_2\right)$. Now, it is easy to find a posterior distribution

$$\begin{aligned} &p(s|\boldsymbol{X}) \propto \left(\prod_{i=1}^{k} p\left(\boldsymbol{z}_i = \boldsymbol{X}_{\star i}\right)\right)p(s|\alpha, \beta) \propto \exp(-\beta/s) \\ &s^{-(\alpha+1)}s^{-k(n-m)/2}\exp\left(-\sum_{i=1}^{k}\boldsymbol{X}_{\star i}^T\boldsymbol{P}_2^{\dagger}\boldsymbol{X}_{\star i}/(2s)\right). \end{aligned} \tag{11.16}$$

Because $\boldsymbol{P}_2$ is orthogonal projector $\boldsymbol{P}_2^{\dagger} = \boldsymbol{P}_2$. In addition to that, $\boldsymbol{X}_{\star i}$ belongs to $\mathsf{Range}(\boldsymbol{P}_1)$, so each term of the quadratic form simplifies $\boldsymbol{X}_{\star i}^T \boldsymbol{P}_2^{\dagger} \boldsymbol{X}_{\star i} = \boldsymbol{X}_{\star i}^T \boldsymbol{X}_{\star i}$. Using the definition of inverse-gamma distribution we can identify new parameters $\widetilde{\alpha} = \alpha + k(n-m)/2$, $\widetilde{\beta} = \beta + \mathsf{tr}\left(\boldsymbol{X}^T\boldsymbol{X}\right)/2$.

2. Predictive distribution is

$$\begin{aligned} & p\left(\boldsymbol{x} \,\middle|\, \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}, \boldsymbol{X}\right) \\ & = \int ds\; p\left(\boldsymbol{x} \,\middle|\, \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}, \boldsymbol{x}_0, s\boldsymbol{\Sigma}_0\right) p(s|\boldsymbol{X}), \end{aligned} \tag{11.17}$$

where the first factor under the integral is multivariate normal $\mathcal{N}\left(\boldsymbol{x}|\widetilde{\boldsymbol{x}}, s\boldsymbol{P}_2\right)$ (this follows from Theorem 4.2.4 with $\boldsymbol{G} = s\boldsymbol{I}$), and the second is $\mathsf{IG}\left(s|\widetilde{\alpha},\widetilde{\beta}\right)$. Using the result from Lemma 4.3.1 we confirm that the predictive distribution is $\mathsf{St}_{2\widetilde{\alpha}}\left(\boldsymbol{x}|\widetilde{\boldsymbol{x}}, \frac{\widetilde{\beta}}{\widetilde{\alpha}}\boldsymbol{P}_2\right)$.

11.11 Lemma 4.3.4

1. We define random variable $\boldsymbol{z} = \boldsymbol{A}^{1/2}\boldsymbol{P}_1(\boldsymbol{x} - \boldsymbol{x}_0)$. To find probability density function of $\boldsymbol{z}$ we use three facts. First, $\boldsymbol{P}_1\boldsymbol{V} = 0$, which follows from definition of $\boldsymbol{P}_1$. Second, because $\mathsf{Range}(\boldsymbol{P}_1) = \mathsf{Range}(\boldsymbol{Y})$, we conclude that $\boldsymbol{P}_1\boldsymbol{Y} = \boldsymbol{Y}$. Finally, $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{\delta}_1 + s^{1/2}\boldsymbol{Y}\boldsymbol{\delta}_2$, where $\boldsymbol{\delta}_i,\ i = 1,2$ are independent standard multivariate normal random variables. Using these three facts we find a probability density function $p(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z}|0, s\boldsymbol{A}^{1/2}\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{A}^{1/2}\right)$. It is easy to see that $\boldsymbol{P}_3 = \boldsymbol{A}^{1/2}\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{A}^{1/2}$ is an orthogonal projector. Indeed, from $\boldsymbol{A}-$orthogonality we conclude that

$$\boldsymbol{P}_3^2 = \boldsymbol{A}^{1/2}\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{A}\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{A}^{1/2} = \boldsymbol{P}_3. \tag{11.18}$$

The orthogonality $\boldsymbol{P}_3^T = \boldsymbol{P}_3$ follows from $\boldsymbol{A}^T = \boldsymbol{A}$. Now, it is easy to find a posterior distribution

$$\begin{aligned} & p(s|\boldsymbol{Z}) \propto \left(\prod_{i=1}^{k} p\left(\boldsymbol{z}_i = \boldsymbol{Z}_{\star i}\right)\right) p(s|\alpha,\beta) \propto \exp(-\beta/s) \\ & s^{-(\alpha+1)} s^{-k(n-m)/2} \exp\left(-\sum_{i=1}^{k} \boldsymbol{Z}_{\star i}^T \boldsymbol{P}_3^{\dagger} \boldsymbol{Z}_{\star i}/(2s)\right). \end{aligned} \tag{11.19}$$

Using that $\boldsymbol{P}_3^{\dagger} = \boldsymbol{P}_3$ ($\boldsymbol{P}_3$ is an orthogonal projector) and that $\boldsymbol{z} \in \mathsf{Range}(\boldsymbol{A}^{1/2}\boldsymbol{Y})$ (this follows from $\mathsf{Range}(\boldsymbol{P}_1) = \mathsf{Range}(\boldsymbol{Y})$) we simplify quadratic form $\boldsymbol{Z}_{\star i}^T \boldsymbol{P}_3^{\dagger} \boldsymbol{Z}_{\star i} = \boldsymbol{Z}_{\star i}^T \boldsymbol{Z}_{\star i}$. Using the definition of inverse-gamma distribution we can identify new parameters $\widetilde{\alpha} = \alpha + k(n-m)/2$, $\widetilde{\beta} = \beta + \mathsf{tr}\left(\boldsymbol{Z}^T\boldsymbol{Z}\right)/2$.

2. Predictive distribution is

$$\begin{aligned} & p\left(\boldsymbol{x} \,\middle|\, \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}, \boldsymbol{Z}\right) \\ & = \int ds\; p\left(x \,\middle|\, \boldsymbol{W}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{b}, \boldsymbol{x}_0, s\boldsymbol{\Sigma}_0\right) p(s|\boldsymbol{Z}), \end{aligned} \tag{11.20}$$

where the first factor under the integral is multivariate normal $\mathcal{N}\left(\boldsymbol{x}|\widetilde{\boldsymbol{x}}, s\boldsymbol{Y}\boldsymbol{Y}^T\right)$ (this follows from Theorem 4.2.1 with $\boldsymbol{G} = s\boldsymbol{I}$), and the second is $\mathsf{IG}\left(s|\widetilde{\alpha}, \widetilde{\beta}\right)$. Using the result from Lemma 4.3.1 we confirm that the predictive distribution is $\mathsf{St}_{2\widetilde{\alpha}}\left(\boldsymbol{x}|\widetilde{\boldsymbol{x}}, \frac{\widetilde{\beta}}{\widetilde{\alpha}}\boldsymbol{Y}\boldsymbol{Y}^T\right)$.

11.12 Lemma 4.3.5

Distribution of $\boldsymbol{x} - \widetilde{\boldsymbol{x}}$ is $\mathcal{N}(\cdot|0, s\boldsymbol{Y}\boldsymbol{Y}^T)$, so $\boldsymbol{x} - \widetilde{\boldsymbol{x}} = s^{1/2}\boldsymbol{Y}\boldsymbol{\delta}_1$, where $\boldsymbol{\delta}_1$ is standard multivariate normal variable. Using this we find $S(\boldsymbol{x}) = s\boldsymbol{\delta}_1^T\boldsymbol{Y}^T\boldsymbol{A}\boldsymbol{Y}\boldsymbol{\delta}_1 = s\boldsymbol{\delta}_1^T\boldsymbol{\delta}_1$, where the last equality follows from $\boldsymbol{A}-$orthogonality of columns of $\boldsymbol{Y}$.

11.13 Lemma 4.3.6

Let columns of $\widetilde{\boldsymbol{V}}$ and $\widetilde{\boldsymbol{W}}$ be new bases in subspaces $\mathsf{Range}\,(\boldsymbol{V})$ and $\mathsf{Range}\,(\boldsymbol{W})$. It is always possible to find invertible square matrices $\boldsymbol{G}_1$, $\boldsymbol{G}_2$ that perform a change of bases, i.e., $\boldsymbol{V} = \widetilde{\boldsymbol{V}}\boldsymbol{G}_1$ and $\boldsymbol{W} = \widetilde{\boldsymbol{W}}\boldsymbol{G}_1$. After the substitution of $\widetilde{\boldsymbol{V}}$ and $\widetilde{\boldsymbol{W}}$ in (1.5) yields

$$\begin{aligned} \widetilde{\boldsymbol{x}} &= \boldsymbol{x}_0 + \widetilde{\boldsymbol{V}}\boldsymbol{G}_1\left(\boldsymbol{G}_2^T\widetilde{\boldsymbol{W}}^T\boldsymbol{A}\widetilde{\boldsymbol{V}}\boldsymbol{G}_1\right)^{-1}\boldsymbol{G}_2^T\widetilde{\boldsymbol{W}}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0) \\ &= \boldsymbol{x}_0 + \widetilde{\boldsymbol{V}}\left(\widetilde{\boldsymbol{W}}^T\boldsymbol{A}\widetilde{\boldsymbol{V}}\right)^{-1}\widetilde{\boldsymbol{W}}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0). \end{aligned}$$

So the mean vector does not depend on the choice of basis.

11.14 Theorem 4.3.1

By construction $\widetilde{\boldsymbol{v}}_i,\ i = 1, \ldots, m$ form an $\boldsymbol{A}$-orthogonal basis for $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0)$. Using Lemma 4.3.6 we can transform matrix $\boldsymbol{V}$, such that columns of new matrix $\widetilde{\boldsymbol{V}}$ are $\widetilde{\boldsymbol{v}}_i,\ i = 1, \ldots, m$.

To apply Theorem 4.2.1 we need to check that $\mathsf{Range}(\boldsymbol{Y}) = \mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A})$. Indeed, if $\boldsymbol{y} \in \mathsf{Range}(\boldsymbol{Y})$ it has a form $\boldsymbol{y} = \sum_{i=m+1}^{n} y_i \widetilde{\boldsymbol{v}}_i$, we can see that $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{y} = \boldsymbol{V}^T\boldsymbol{A}\boldsymbol{y} = 0$, because $\widetilde{\boldsymbol{v}}_i^T\boldsymbol{A}\widetilde{\boldsymbol{v}}_j = 0$ for $i = 1, \ldots, m,\ j = m+1, \ldots, n$. This means $\mathsf{Range}(\boldsymbol{Y}) \subset \mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A})$. Now, if $y \in \mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A})$ it is $\boldsymbol{A}$-orthogonal to the first m vectors $\widetilde{\boldsymbol{v}}_i$, because $\widetilde{\boldsymbol{v}}_i,\ i = 1, \ldots, n$ form a complete set[1], we conclude that $\boldsymbol{y} \in \mathsf{Range}(\boldsymbol{Y})$ and $\mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A}) \subset \mathsf{Range}(\boldsymbol{Y})$.

Because, $\mathsf{Range}(\boldsymbol{Y}) = \mathsf{Null}(\boldsymbol{W}^T\boldsymbol{A})$ we can apply Theorem 4.2.1 which gives us (1.5) as mean and $\boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T$ as a covariance matrix.

11.15 Lemma 4.3.7

Let columns of $\boldsymbol{U}$ be $\boldsymbol{u}_j,\ j = 1, \ldots, K$ and D be a diagonal matrix with $D_{jj} = d_j$. Using the definition we get

$$\sup_{\boldsymbol{x}} \frac{\sqrt{\boldsymbol{x}^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T\boldsymbol{x}}}{\sqrt{\boldsymbol{x}^T\boldsymbol{A}^{-1}\boldsymbol{x}}} = \sup_{\boldsymbol{y}} \frac{\sqrt{\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{U}\boldsymbol{D}^2\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{y}}}{\sqrt{\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{y}}},$$

where we used $\boldsymbol{A}$-orthogonality and define $\boldsymbol{y} = \boldsymbol{A}^{-1}\boldsymbol{x}$. Now, without the loss of generality we take $\boldsymbol{y} = \boldsymbol{U}\boldsymbol{\alpha}$ to obtain

$$\|\boldsymbol{B}\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} = \sup_{\boldsymbol{\alpha}} \frac{\sqrt{\boldsymbol{\alpha}^T\boldsymbol{D}^2\boldsymbol{\alpha}}}{\sqrt{\boldsymbol{\alpha}^T\boldsymbol{\alpha}}} = d_1.$$

11.16 Lemma 4.3.8

From Lemma 4.3.7 we know that $\|\boldsymbol{B} - \boldsymbol{B}_m\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} = d_{m+1}$. For the second part we use a proof by contradiction from [TB97, Theorem 5.8].

Suppose that there is $\boldsymbol{C},\ \mathsf{rank}(\boldsymbol{C}) \leq m$ for which the norm of the difference is smaller, i.e., $\|\boldsymbol{B} - \boldsymbol{C}\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} < d_{m+1}$. Because $\boldsymbol{C}$ has rank m there is a $n - m$ dimensional subspace $R \subset \mathbb{R}^n : \forall \boldsymbol{r} \in R \Rightarrow \boldsymbol{C}\boldsymbol{r} = 0$. This implies

$$\begin{aligned}\|\boldsymbol{B}\boldsymbol{r}\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} &= \|(\boldsymbol{B} - \boldsymbol{C})\, r\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} \\ &\leq \|\boldsymbol{B} - \boldsymbol{C}\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} \|\boldsymbol{r}\|_{\boldsymbol{A}^{-1}} < \sigma_{m+1} \|\boldsymbol{r}\|_{\boldsymbol{A}^{-1}}.\end{aligned}$$

[1] We do not consider "lucky breakdowns" [Saa03, Section 6.3.1]

We know that in the subspace $\widetilde{R}$ spanned by $\boldsymbol{u}_j$, $j = 1, \ldots, m+1$ the norm of the matrix $\boldsymbol{B}$ fulfills $\|\boldsymbol{B}\widetilde{\boldsymbol{r}}\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} \geq d_{m+1} \|\widetilde{\boldsymbol{r}}\|_{\boldsymbol{A}^{-1}}$. Because for these subspaces $\left|\widetilde{R}\right| + |R| = n + 1$, there is a vector that belongs to both of them. Thus by contradiction $\|\boldsymbol{B} - \boldsymbol{C}\|_{\boldsymbol{A},\boldsymbol{A}^{-1}} \geq d_{m+1}$, and the bound is attained by $\boldsymbol{B}_m$.

Chapter 12

Hidden representation

12.1 Proposition 5.4.1

From Eq. (5.41) and condition $\boldsymbol{U}_n \equiv \boldsymbol{I}$ we conclude that iteration matrices are $N_n = (\boldsymbol{V}_n)^{-1}$ and $\boldsymbol{M}_n = \left(\boldsymbol{I} - (\boldsymbol{V}_n)^{-1}\boldsymbol{A}\right)$. In this proof be $\mathbb{E}$ we denote the average with respect to the distribution Eq. (5.47). For arbitrary $N \times N$ matrices $\boldsymbol{C}$, $\boldsymbol{D}$ the following averages are known

$$\begin{aligned}
&\text{[GN18, p. 2.3.2] :}\\
&\mathbb{E}\left[\boldsymbol{C}\left(\boldsymbol{V}_n\right)^{-1}\boldsymbol{D}\right] = \boldsymbol{C}\boldsymbol{R}_n\boldsymbol{D},\ \mathbb{E}\left[\boldsymbol{C}\left(\boldsymbol{V}_n\right)^{-T}\boldsymbol{D}\right] = \boldsymbol{C}\boldsymbol{R}_n^T\boldsymbol{D};\\
&\text{[GN18, Theorem 2.3.5] :}\\
&\mathbb{E}\left[(\boldsymbol{V}_n)^{-1}\boldsymbol{C}\left((\boldsymbol{V}_n)^{-1}\right)^T\right] = \operatorname{tr}\left(\boldsymbol{C}^T\widetilde{\boldsymbol{\Psi}}_n\right)\widetilde{\boldsymbol{\Sigma}}_n + \boldsymbol{R}_n\boldsymbol{C}\left(\boldsymbol{R}_n\right)^T.
\end{aligned} \tag{12.1}$$

With (12.1) one can easily obtain (5.48), (5.49) from (5.44), (5.45) respectively after straightforward algebraic manipulations.

12.2 Proposition 5.4.2

It is evident that for arbitrary $\boldsymbol{A}$ and $\boldsymbol{R}_n$, if $\boldsymbol{\Sigma}^n > 0$ (or $\Sigma^n \geq 0$)

$$\left(\boldsymbol{I} - \boldsymbol{R}_n\boldsymbol{A}\right)\boldsymbol{\Sigma}^n\left(\boldsymbol{I} - \boldsymbol{R}_n\boldsymbol{A}\right)^T \geq 0. \tag{12.2}$$

Hence, it is enough to show that $\operatorname{tr}\left(\boldsymbol{\Sigma}^n\boldsymbol{A}^T\widetilde{\boldsymbol{\Psi}}_n\boldsymbol{A}\right) + \left(\boldsymbol{r}_n\right)^T\widetilde{\boldsymbol{\Psi}}_n\boldsymbol{r}_n > 0$ (or ≥ 0). Because in this case the left-hand side of (5.49) will be the sum of positive semi-definite

and positive-definite (or positive semi-definite) matrices. If $\widetilde{\boldsymbol{\Psi}} > 0$ (or $\widetilde{\boldsymbol{\Psi}} \geq 0$) the last term is positive (or non-negative) for non-zero residual, i.e. $(\boldsymbol{r}_n)^T \widetilde{\boldsymbol{\Psi}}_n \boldsymbol{r}_n > 0$ (or ≥ 0). For the term $\operatorname{tr}\left(\boldsymbol{\Sigma}^n \boldsymbol{A}^T \widetilde{\boldsymbol{\Psi}}_n \boldsymbol{A}\right)$ we will use the following representation [GN18, Theorem 1.2.22] $\operatorname{vec}(\boldsymbol{A})\left(\boldsymbol{\Sigma}^n \otimes \widetilde{\boldsymbol{\Psi}}_n\right)(\operatorname{vec}(\boldsymbol{A}))^T$. It is known [GN18, Theorem 1.2.21] that the Kronecker product of two positive definite (or positive definite and positive semi-definite) matrices are positive definite (or positive semi-definite) so the trace is positive (or non-negative). Now, since the after the first iteration $\boldsymbol{\Sigma}^1 = \widetilde{\boldsymbol{\Sigma}}_0 (\boldsymbol{r}_n)^T \widetilde{\boldsymbol{\Psi}}_0 \boldsymbol{r}_n$ we conclude that for all subsequent iterations with nonzero residuals $\boldsymbol{\Sigma}^n > 0$ if $\widetilde{\boldsymbol{\Psi}}_n > 0$, $\widetilde{\boldsymbol{\Sigma}}_n > 0$ (or $\boldsymbol{\Sigma}^n \geq 0$ if $\widetilde{\boldsymbol{\Psi}}_n \geq 0$, $\widetilde{\boldsymbol{\Sigma}}_n \geq 0$).

12.3 Proposition 5.4.3

After the single iteration $\boldsymbol{\Sigma}^1 = \widetilde{\boldsymbol{\Sigma}}_0 (\boldsymbol{r}_n)^T \widetilde{\boldsymbol{\Psi}}_0 \boldsymbol{r}_n$, so

$$\operatorname{tr}\left(\boldsymbol{\Sigma}^{n+1}\left(\boldsymbol{\Sigma}^1\right)^{-1}\right) = \frac{\operatorname{tr}\left(\left(\widetilde{\boldsymbol{\Sigma}}_0\right)^{-1}(\boldsymbol{I} - \boldsymbol{R}_n\boldsymbol{A})\,\boldsymbol{\Sigma}^n\,(\boldsymbol{I} - \boldsymbol{R}_n\boldsymbol{A})^T\right)}{(\boldsymbol{r}_0)^T \widetilde{\boldsymbol{\Psi}}_0 \boldsymbol{r}_0} + \frac{\operatorname{tr}\left(\left(\widetilde{\boldsymbol{\Sigma}}_0\right)^{-1}\widetilde{\boldsymbol{\Sigma}}_n\right)}{(\boldsymbol{r}_0)^T \widetilde{\boldsymbol{\Psi}}_0 \boldsymbol{r}_0}\left(\operatorname{tr}\left(\boldsymbol{\Sigma}^n \boldsymbol{A}^T \widetilde{\boldsymbol{\Psi}}_n \boldsymbol{A}\right) + (\boldsymbol{r}_n)^T \widetilde{\boldsymbol{\Psi}}_n \boldsymbol{r}_n\right). \quad (12.3)$$

Using the representation $\operatorname{tr}\left(\boldsymbol{C}^T \boldsymbol{D}_2 \boldsymbol{C} \boldsymbol{D}_1\right) = (\operatorname{vec}(\boldsymbol{C}))^T \boldsymbol{D}_1^T \otimes \boldsymbol{D}_2 \operatorname{vec}(\boldsymbol{C})$ (for references see the proof for Proposition 5.4.2), the fact that the following matrices $\widetilde{\boldsymbol{\Sigma}}_0$, $\widetilde{\boldsymbol{\Sigma}}_n$, $\boldsymbol{\Sigma}^n$, $\widetilde{\boldsymbol{\Psi}}_n$ are positive definite, and the statement that the Kronecker product preserve positive definiteness, we conclude that

$$\operatorname{tr}\left(\left(\widetilde{\boldsymbol{\Sigma}}_0\right)^{-1}(\boldsymbol{I} - \boldsymbol{R}_n\boldsymbol{A})\,\boldsymbol{\Sigma}^n\,(\boldsymbol{I} - \boldsymbol{R}_n\boldsymbol{A})^T\right) \geq 0, \quad \operatorname{tr}\left(\boldsymbol{\Sigma}^n \boldsymbol{A}^T \widetilde{\boldsymbol{\Psi}}_n \boldsymbol{A}\right) > 0. \quad (12.4)$$

So, we can drop two first terms and recover (5.50).

12.4 Proposition 5.4.4

Observe that iterations for mean

$$\boldsymbol{\mu}^{n+1} = \boldsymbol{\mu}^n + \boldsymbol{R}\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{\mu}^n\right), \quad (12.5)$$

is a preconditioned Richardson iterations scheme, that decouples from iterations for covariance matrix. Hence the standard necessary and sufficient condition [Hac16, Theorem 2.16] applies, which gives us the first condition $\rho(\boldsymbol{I} - \boldsymbol{RA}) < 1$. To obtain the second condition we apply vec operation to (5.49) (see [GN18, Definition 1.2.6 and Theorem 1.2.22])

$$\operatorname{vec}\left(\boldsymbol{\Sigma}^{n+1}\right) = \left((\boldsymbol{I} - \boldsymbol{RA}) \otimes (\boldsymbol{I} - \boldsymbol{RA}) + \mathsf{vec}\left(\widetilde{\boldsymbol{\Sigma}}\right) \mathsf{vec}\left(\boldsymbol{A}^T \widetilde{\boldsymbol{\Psi}} \boldsymbol{A}\right)^T\right) \operatorname{vec}\left(\boldsymbol{\Sigma}^n\right) + \\ + \operatorname{vec}\left(\widetilde{\boldsymbol{\Sigma}}\right) (\boldsymbol{r}_n)^T \widetilde{\boldsymbol{\Psi}} \boldsymbol{r}_n. \quad (12.6)$$

Since (12.6) has the same form as (12.5), the same necessary and sufficient condition applies, which finishes the proof.

12.5 Proposition 5.4.5

Let $\|\cdot\|_{\boldsymbol{R},\boldsymbol{A}}$ is the operator norm for which $\|\boldsymbol{I} - \boldsymbol{RA}\|_{\boldsymbol{R},\boldsymbol{A}} < 1$. Note, that such norm always exists if $\rho(\boldsymbol{I} - \boldsymbol{RA}) < 1$ (see [Hac16, Lemma B.26] for the proof). Let $\boldsymbol{e}^n = \boldsymbol{A}^{-1}\boldsymbol{b} - \boldsymbol{\mu}^n$, then

$$\boldsymbol{e}^{n+1} = (\boldsymbol{I} - \boldsymbol{R}_n \boldsymbol{A})\, \boldsymbol{e}^n. \quad (12.7)$$

For convergence it is sufficient to show that $\lim\limits_{n\to\infty} \|\boldsymbol{e}^n\| = 0$ for some norm. Since $\lim\limits_{n\to\infty} \boldsymbol{R}_n = \boldsymbol{R}$ exists the sequence of matrices $\boldsymbol{L}_n$:

$$\boldsymbol{I} - \boldsymbol{R}_n \boldsymbol{A} = \boldsymbol{I} - \boldsymbol{RA} + \boldsymbol{L}_n, \ \lim_{n\to\infty} \|\boldsymbol{L}_n\| = 0, \quad (12.8)$$

for arbitrary operator norm. Using the basic inequalities of operator norms, we obtain:

$$\left\|\boldsymbol{e}^{n+1}\right\|_{\boldsymbol{R},\boldsymbol{A}} \leq \left(\|\boldsymbol{I} - \boldsymbol{RA}\|_{\boldsymbol{R},\boldsymbol{A}} + \|\boldsymbol{L}_n\|_{\boldsymbol{R},\boldsymbol{A}}\right) \|\boldsymbol{e}^n\|_{\boldsymbol{R},\boldsymbol{A}}. \quad (12.9)$$

From the convergence of $\boldsymbol{L}_n$ we can conclude that $\forall\epsilon\ \exists N(\epsilon) : \forall n \geq N(\epsilon) \Rightarrow \|\boldsymbol{L}_n\|_{\boldsymbol{R},\boldsymbol{A}} \leq \epsilon$. By construction $\|\boldsymbol{I} - \boldsymbol{RA}\|_{\boldsymbol{R},\boldsymbol{A}} < 1$, so it is always possible to find ϵ that ensures $\alpha_n \equiv \left(\|\boldsymbol{I} - \boldsymbol{RA}\|_{\boldsymbol{R},\boldsymbol{A}} + \|\boldsymbol{L}_n\|_{\boldsymbol{R},\boldsymbol{A}}\right) < 1$ for all $n > N$. Taking $\alpha = \sup\{\alpha_n\}$ which is smaller than one by construction (note, that $\alpha_n \to \|\boldsymbol{I} - \boldsymbol{RA}\|_{\boldsymbol{R},\boldsymbol{A}} < 1$), we can conclude that

$$\forall K \Rightarrow 0 \leq \left\|\boldsymbol{e}^{N+K}\right\|_{\boldsymbol{R},\boldsymbol{A}} \leq \alpha^K \left\|\boldsymbol{e}^N\right\|_{\boldsymbol{R},\boldsymbol{A}}, \quad (12.10)$$

which establishes the convergence.

The proof for variance follows the same tactics. Let $\boldsymbol{e}^n \equiv \operatorname{vec}(\boldsymbol{\Sigma})^n$, $\boldsymbol{b}_n = \operatorname{vec}\left(\widetilde{\boldsymbol{\Sigma}}_n\right)(\boldsymbol{r}^n)^T \widetilde{\boldsymbol{\Psi}}_n \boldsymbol{r}^n$,

$$\begin{aligned}
\boldsymbol{M}_n &\equiv \left((\boldsymbol{I}-\boldsymbol{R}_n\boldsymbol{A})\otimes(\boldsymbol{I}-\boldsymbol{R}_n\boldsymbol{A}) + \mathsf{vec}\left(\widetilde{\boldsymbol{\Sigma}}_n\right)\mathsf{vec}\left(\boldsymbol{A}^T\widetilde{\boldsymbol{\Psi}}_n\boldsymbol{A}\right)^T\right), \\
\boldsymbol{M} &\equiv \left((\boldsymbol{I}-\boldsymbol{R}\boldsymbol{A})\otimes(\boldsymbol{I}-\boldsymbol{R}\boldsymbol{A}) + \mathsf{vec}\left(\widetilde{\boldsymbol{\Sigma}}\right)\mathsf{vec}\left(\boldsymbol{A}^T\widetilde{\boldsymbol{\Psi}}\boldsymbol{A}\right)^T\right),
\end{aligned} \tag{12.11}$$

and $\|\cdot\|_{\boldsymbol{M}}$ is the norm in which $\|\boldsymbol{M}\|_{\boldsymbol{M}} < 1$. Using the representation $\boldsymbol{M}_n = \boldsymbol{M} + \boldsymbol{Y}_n$ with $\|\boldsymbol{Y}_n\| \to 0$, we construct the analogous bound for covariance matrix

$$0 \le \left\|\boldsymbol{e}^{n+1}\right\|_{\boldsymbol{M}} \le \left(\|\boldsymbol{M}\|_{\boldsymbol{M}} + \|\boldsymbol{Y}_n\|_{\boldsymbol{M}}\right)\|\boldsymbol{e}^n\|_{\boldsymbol{M}} + \|\boldsymbol{b}_n\|_{\boldsymbol{M}}. \tag{12.12}$$

From convergence of $\boldsymbol{b}_n$, $\boldsymbol{Y}_n$ and $\|\boldsymbol{M}\|_{\boldsymbol{M}} < 1$ one can infer the convergence of $\|\boldsymbol{e}^{n+1}\|_{\boldsymbol{M}}$ the same way as it was done in the first part of the proof. Namely, it is always possible to choose sufficiently large N, that $\forall n \ge N\left(\|\boldsymbol{M}\|_{\boldsymbol{M}} + \|\boldsymbol{Y}_n\|_{\boldsymbol{M}}\right) = \gamma < 1$ and in the same time $\|\boldsymbol{b}_n\|_{\boldsymbol{M}} < \epsilon$ for any desired ϵ. From that we can find the upper bound

$$\left\|\boldsymbol{e}^{N+n}\right\|_M \le K^n\left\|\boldsymbol{e}^N\right\|_M + \epsilon\sum_{k=1}^{n} K^{k-1} = K^n\left\|\boldsymbol{e}^N\right\|_M + \epsilon\frac{1-K^n}{1-K}, \tag{12.13}$$

and take the limit:

$$0 \le \lim_{n\to\infty}\|\boldsymbol{e}^n\|_M \le \frac{\epsilon}{1-K}. \tag{12.14}$$

Since ϵ is arbitrary $\lim\limits_{n\to\infty}\boldsymbol{e}^n = 0$.

12.6 Proposition 5.4.6

Since $\lim_{k\to\infty}\widetilde{\boldsymbol{\Sigma}}_k = 0$ and $\exists C : \forall n \left\|\widetilde{\boldsymbol{\Psi}}_n\right\| < C$ we can conclude that

$$\begin{aligned}
\lim_{n\to\infty}\Big((\boldsymbol{I}-\boldsymbol{R}_n\boldsymbol{A})\otimes(\boldsymbol{I}-\boldsymbol{R}_n\boldsymbol{A}) + \operatorname{vec}\left(\widetilde{\boldsymbol{\Sigma}}_n\right)\operatorname{vec}\left(\boldsymbol{A}^T\widetilde{\boldsymbol{\Psi}}_n\boldsymbol{A}\right)^T\Big) = \\
= (\boldsymbol{I}-\boldsymbol{R}\boldsymbol{A})\otimes(\boldsymbol{I}-\boldsymbol{R}\boldsymbol{A}).
\end{aligned} \tag{12.15}$$

Since $\rho\left((\boldsymbol{I}-\boldsymbol{R}\boldsymbol{A})\otimes(\boldsymbol{I}-\boldsymbol{R}\boldsymbol{A})\right) = \rho(\boldsymbol{I}-\boldsymbol{R}\boldsymbol{A})^2 < 1$, all conditions of Proposition 5.4.5 are met, so the convergence follows.

12.7 Proposition 5.4.7

Updates for the mean vectors are identical. The only part we need to prove is the convergence of variance iterations. Let $\boldsymbol{\Sigma}^n$ represents the convergent sequence arises from the original iterations (5.49), and $\boldsymbol{E}^n$ the sequence produced by the low-rank iterations with the same matrices $\widetilde{\boldsymbol{\Sigma}}_n$, $\widetilde{\boldsymbol{\Psi}}_n$, $\boldsymbol{R}_n$. The strategy is to bound $\boldsymbol{E}^n$ by $\boldsymbol{\Sigma}^n$.

After a single step one has

$$\boldsymbol{\Sigma}^1 = \widetilde{\boldsymbol{\Sigma}}_0 \left(\boldsymbol{r}^0\right)^T \widetilde{\boldsymbol{\Psi}}_0 \boldsymbol{r}^0,\ \boldsymbol{E}^1 = \left[\widetilde{\boldsymbol{\Sigma}}_0\right]_L \left(\boldsymbol{r}^0\right)^T \widetilde{\boldsymbol{\Psi}}_0 \boldsymbol{r}^0 \Longrightarrow \boldsymbol{\Sigma}^1 \geq \boldsymbol{E}^1. \tag{12.16}$$

Let $\boldsymbol{\Sigma}^{n+1} = \boldsymbol{G}_n\left(\boldsymbol{\Sigma}^n\right)$ represents the update of covariance (5.49), then $\boldsymbol{E}^{n+1} = \left[\boldsymbol{G}_n\left(\boldsymbol{E}^n\right)\right]_L$. Suppose, on the step n the following condition holds $\boldsymbol{\Sigma}^n \geq \boldsymbol{E}^n$. We know that iterations preserve positive definiteness, so

$$\boldsymbol{G}_n\left(\boldsymbol{\Sigma}^n\right) - \boldsymbol{G}_n\left(\boldsymbol{E}^n\right) = \boldsymbol{G}_n\left(\boldsymbol{\Sigma}^n - \boldsymbol{E}^n\right)\big|_{\boldsymbol{r}^n=0} \geq 0. \tag{12.17}$$

In addition it is evident that $\boldsymbol{G}_n\left(\boldsymbol{E}^n\right) \geq \left[\boldsymbol{G}_n\left(\boldsymbol{E}^n\right)\right]_L = \boldsymbol{E}^{n+1}$, so we can write

$$\boldsymbol{\Sigma}^n \geq \boldsymbol{E}^n \Rightarrow \boldsymbol{\Sigma}^{n+1} \geq \boldsymbol{E}^{n+1}. \tag{12.18}$$

Equation (12.16) form the base case and (12.18) is the step, so we can conclude that by inductive argument $\boldsymbol{\Sigma}^n \geq \boldsymbol{E}^n\ \forall n$. That allows us to construct the following bounds

$$0 \leq \boldsymbol{E}^n \leq \boldsymbol{\Sigma}^n \Longrightarrow \left(\lim_{n\to\infty} \boldsymbol{\Sigma}^n = 0 \Rightarrow \lim_{n\to\infty} \boldsymbol{E}^n = 0\right). \tag{12.19}$$

12.8 Proposition 5.4.8

The result can be readily established from the chosen schedule. The form of iterations for $n > 1$ becomes

$$\boldsymbol{\Sigma}^{n+1} = \boldsymbol{M}\boldsymbol{\Sigma}^n\boldsymbol{M}^T = \boldsymbol{S}\boldsymbol{D}\boldsymbol{S}^T\boldsymbol{\Sigma}^n\boldsymbol{S}\boldsymbol{D}\boldsymbol{S}^T \Rightarrow \boldsymbol{S}^T\boldsymbol{\Sigma}^{n+1}\boldsymbol{S} = \boldsymbol{D}\boldsymbol{S}^T\boldsymbol{\Sigma}^{\boldsymbol{n}}\boldsymbol{S}\boldsymbol{D}. \tag{12.20}$$

Taking into account that $\boldsymbol{D}$ is a diagonal matrix, we find how σ_{ij} evolve

$$\sigma_{ij}^n = \lambda_i\lambda_j\sigma_{ij}^{n-1}. \tag{12.21}$$

The desired limit is an immediate consequence of (12.21).

Chapter 13

Black-Box optimization of BPX preconditioners

13.1 Proposition 7.2.1

For $D = 1$ we define a set of meshes by $x_j^l = j/2^l$, where $j = 0, 1, \ldots, 2^l - 1, 2^l$, and $l = L_m, L_m+1, \ldots, L_M-1, L_M$. For each level l a standard basis in FE space is formed by tent functions $\phi_i^l(x) = \phi^l(x-x_i)$, where $\phi^l(x) = \left(1 + x/2^l\right) \mathrm{Ind}\left[-1/2^l \le x \le 0\right] + \left(1 - x/2^l\right) \mathrm{Ind}\left[0 < x \le 1/2^l\right]$. The standard form of BPX preconditioner is

$$\mathcal{B}u = \sum_{l=L_m}^{L_M} \sum_{k=0}^{2^l} \left(\phi_k^l, u\right) \phi_k^l. \tag{13.1}$$

So we can rewrite Eq. (13.1) using downsampling, that is, computation of (ϕ_k^{l-1}, u) from $\left(\phi_k^l, u\right)$, and interpolation. Both operations are represented by matrix

$$P_l^{l-1} = \begin{pmatrix} 1 & 1/2 & & & & & \\ & 1/2 & 1 & 1/2 & & & \cdots \\ & & & 1/2 & 1 & 1/2 & \\ & & & \vdots & & & \ddots \end{pmatrix}, \; P_l^{l-1} \in \mathbb{R}^{\left(2^{l-1}+1\right)\times\left(2^l+1\right)}, \tag{13.2}$$

and its transpose in case of interpolation. It is clear that if we define $B_{L_M}^l = P_{l+1}^l P_{l+2}^{l+1} \cdots P_{L_M-1}^{L_M-2} P_{L_M}^{L_M-1}$ for $L_m \le l < L_M$, $B_{L_M}^{L_M} = I$, and $B_l^{L_M} = \left(B_{L_M}^l\right)^T$, BPX preconditioner will be represented by the following matrix

$$\mathcal{B} = \left(\sum_{k=L_m}^{L_M} B_k^{L_M} B_{L_M}^k \right) S, \tag{13.3}$$

where $S_{ij} = \frac{2^{-L_M}}{6}\left(4\delta_{ij} + \delta_{i+1j} + \delta_{i-1j}\right)$, $S \in \mathbb{R}^{\left(2^{L_M}+1\right)\times\left(2^{L_M}+1\right)}$ is used to obtain $\left(\phi_k^{L_M}, u\right)$ from vector u_j that appears in $u = \sum_{j=0}^{2^{L_M}} u_j \phi_j^{L_M}$. We need to show that matrix $B_{L_M}^k$ has a particular form for chosen boundary conditions. Below we demonstrate this by induction for all standard boundary conditions in $1D$.

13.2 Drichlet-Neumann boundary conditions

In this case value on the left boundary is fixed, so we have 2^l points on level l and $B_l^{L_M}$ reads

$$B_l^{L_M} = I_l \otimes \eta_{L_M-l} + S_l \otimes \left(\xi_{L_M-l} - \eta_{L_M-l}\right), \tag{13.4}$$

where $(\eta_k)_i = i/2^k, (\xi_k)_i = 1,\ i = 1,\ldots,2^k$ and $(S_l)_{ij} = \delta_{ij+1}, (I_l)_{ij} = \delta_{ij},\ i,j = 1,\ldots,2^l$.

Base: for $l = L_M - 1$ we have $B_{L_M-1}^{L_M} = I_{L_M-1} \otimes \begin{pmatrix} 1/2 \\ 1 \end{pmatrix} + S_{L_M-1} \otimes \begin{pmatrix} 1/2 \\ 0 \end{pmatrix}$, that is precisely $P_{L_M-1}^{L_M}$ that has a form

$$P_{L_M-1}^{L_M} = \begin{pmatrix} 1/2 & & \\ 1 & & \\ 1/2 & 1/2 & \\ 0 & 1 & \\ 0 & 1/2 & \ldots \\ 0 & 0 & \\ \vdots & & \ddots \end{pmatrix} \tag{13.5}$$

Induction step: suppose for $L_M - l + 1$ the result holds. For $L_M - l$ we will have

$$\begin{aligned} B_{L_M-l}^{L_M} &= B_{L_M-l+1}^{L_M} P_{L_M-l}^{L_M-l+1} = \left(I_{L_M-l+1} \otimes \eta_{l-1} + S_{L_M-l+1} \otimes \left(\xi_{l-1} - \eta_{l-1}\right)\right) P_{L_M-l}^{L_M-l+1} \\ &= \left(\left[I_{L_M-l+1} P_{L_M-l}^{L_M-l+1}\right] \otimes \eta_{l-1} + \left[S_{L_M-l+1} P_{L_M-l}^{L_M-l+1}\right] \otimes \left(\xi_{l-1} - \eta_{l-1}\right)\right) \end{aligned} \tag{13.6}$$

The first term can be simplified

$$P_{L_M-l}^{L_M-l+1} \otimes \eta_{l-1} = I_{L_M-l} \otimes \left(\begin{pmatrix} 1/2 \\ 1 \end{pmatrix} \otimes \eta_{l-1}\right) + S_{L_M-l} \otimes \left(\begin{pmatrix} 1/2 \\ 0 \end{pmatrix} \otimes \eta_{l-1}\right) \tag{13.7}$$

The second term can be transformed using

$$S_{L_M-l+1} P_{L_M-l}^{L_M-l+1} = I_{L_M-1} \otimes \begin{pmatrix} 0 \\ 1/2 \end{pmatrix} + S_{L_M-1} \otimes \begin{pmatrix} 1 \\ 1/2 \end{pmatrix} \tag{13.8}$$

With that we obtain

$$I_{L_M-l} \otimes \left(\begin{pmatrix} 0 \\ 1/2 \end{pmatrix} \otimes (\xi_{l-1} - \eta_{l-1}) \right) + S_{L_M-l} \otimes \left(\begin{pmatrix} 1 \\ 1/2 \end{pmatrix} \otimes (\xi_{l-1} - \eta_{l-1}) \right). \quad (13.9)$$

It is easy to see that $\begin{pmatrix} 1 \\ 1/2 \end{pmatrix} \otimes \xi_{l-1} - \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} \otimes \eta_{l-1} = \xi_l - \eta_l$ and $\begin{pmatrix} 0 \\ 1/2 \end{pmatrix} \otimes \xi_{l-1} + \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} \otimes \eta_{l-1} = \eta_l$ which completes the proof.

13.3 Neumann-Dirichlet boundary conditions

In this case value on the right boundary is fixed, so we have 2^l points on level l and $B_l^{L_M}$ reads

$$B_l^{L_M} = I_l \otimes \eta^r_{L_M-l} + (S_l)^T \otimes \left(\xi_{L_M-l} - \eta^r_{L_M-l} \right), \quad (13.10)$$

where $(\eta^r_k)_i = (\eta_k)_{2^k-i+1}$.

In this case the results follows from the one for Dirichlet-Neumann boundary conditions. Indeed, it is evident that for a chosen FE space

$$\begin{pmatrix} 1 & & \\ 1/2 & 1/2 & \\ 0 & 1 & \\ 0 & 1/2 & \dots \\ 0 & 0 & \\ \vdots & & \ddots \end{pmatrix}_{ij} = \begin{pmatrix} 1/2 & & \\ 1 & & \\ 1/2 & 1/2 & \\ 0 & 1 & \\ 0 & 1/2 & \dots \\ 0 & 0 & \\ \vdots & & \ddots \end{pmatrix}_{2^L-i+1, 2^{L-1}-j+1}, \quad (13.11)$$

so we need to flip both indices in all interpolation matrices. This is equivalent to the flip of both indices in the final result for Dirichlet-Neumann boundary conditions, which gives Eq. (13.10).

13.4 Neumann-Neumann boundary conditions

In this case both left and right boundaries contribute to the degrees of freedom, so there are $2^l + 1$ points on level l. It is not hard to show that

$$B_l^{L_M} = \begin{pmatrix} 1 & 0_{1\times 2^l} \\ e_l \otimes (\xi_{L_M-l} - \eta_{L_M-l}) & I_l \otimes \eta_{L_M-l} + S_l \otimes (\xi_{L_M-l} - \eta_{L_M-l}) \end{pmatrix}, \quad (13.12)$$

where $(e_k)_i = \delta_{i1},\ i = 1, \ldots, 2^k$.

An explicit form of $B^{L_M}_{L_M-1}$ can be obtained from the result of essential-natural BCs

$$B^{L_M}_{L_M-1} = \begin{pmatrix} 1 & & \\ 1/2 & 1/2 & \\ 0 & 1 & \\ 0 & 1/2 & \ldots \\ 0 & 0 & \\ \vdots & & \ddots \end{pmatrix} \equiv \widetilde{P}^{L_M}_{L_M-1} = \begin{pmatrix} 1 & 0_{L_{M-1}} \\ e_{L_M}/2 & P^{L_M}_{L_M-1} \end{pmatrix}. \tag{13.13}$$

It is easy to see that $\widetilde{P}^{L_M}_{L_M-1}\widetilde{P}^{L_M-1}_{L_M-2} = \begin{pmatrix} 1 & 0_{L_M-2} \\ e_{L_M}/2 + P^{L_M}_{L_M-1}e_{L_M-1}/2 & P^{L_M}_{L_M-1}P^{L_M-1}_{L_M-2} \end{pmatrix}$. From that we can conclude that block $2,2$ indeed the same as we claimed above. The only nontivial block is $2,1$. We prove its form by induction.

Base: For $P^{L_M}_{L_M-1}$ the result is established.

Induction step: suppose for $L_M - l + 1$ the result holds. For $L_M - l$ we will have

$$\begin{aligned} B^{L_M}_{L_M-l} &= B^{L_M}_{L_M-l+1}\widetilde{P}^{L_M-l+1}_{L_M-l} \\ &= \begin{pmatrix} 1 & 0_{1\times 2^{L_M-l+1}} \\ e_{L_M-l+1}\otimes(\xi_{l-1}-\eta_{l-1}) & I_{L_M-l+1}\otimes\eta_{l-1} + S_{L_M-l+1}\otimes(\xi_{l-1}-\eta_{l-1}) \end{pmatrix} \\ &\qquad \cdot \begin{pmatrix} 1 & 0_{L_{M-l}} \\ e_{L_M-l+1}/2 & P^{L_M-l+1}_{L_M-l} \end{pmatrix}. \end{aligned} \tag{13.14}$$

From that we can find that $2,1$ block is $e_{L_M-l+1}\otimes(\xi_{l-1}-\eta_{l-1}) + e_{L_M-l+1}\otimes\eta_{l-1}/2 + e^2_{L_M-l+1}\otimes(\xi_{l-1}-\eta_{l-1})/2$ first 2^{L_M} components of this expressions are the same as $\begin{pmatrix}1\\1/2\end{pmatrix}\otimes\xi_{l-1} - \begin{pmatrix}1/2\\1/2\end{pmatrix}\otimes\eta_{l-1}$ which is $\xi_l - \eta_l$. The whole block will be $e_{L_{M-l}}\otimes(\xi_l-\eta_l)$, which completes the proof.

13.5 Drichlet-Dirichlet boundary conditions

In this case both left and right boundaries are absent, so each level has $2^l - 1$ points we just take result for essential-natural BCs and delete the last row and the last column

$$\begin{aligned} B^{L_M}_l &= \left[I_l\otimes\eta_{L_M-l} + S_l\otimes(\xi_{L_M-l}-\eta_{L_M-l})\right]_{\text{remove last row and last column}} \\ &= \left[I_{2^l\times(2^l-1)}\otimes\eta_{L_M-l} + S_{2^l\times(2^l-1)}\otimes(\xi_{L_M-l}-\eta_{L_M-l})\right]_{\text{remove last row}}. \end{aligned} \tag{13.15}$$

An explicit form of $B^{L_M}_{L_M-1}$ can be obtained from the result of essential-natural BCs. Namely, we consider a submatrix as explained below

$$\underset{2^{L_M}\times 2^{L_M-1}}{B^{L_M}_{L_M-1}} = \begin{pmatrix} \ddots & \vdots & \\ \dots & 1/2 & \\ & 1 & \\ & 1/2 & 1/2 \\ & & 1 \end{pmatrix} = \begin{pmatrix} \widetilde{P}^{L_M}_{L_M-1} & \widetilde{e}^{r}_{L_M}/2 \\ \widetilde{0}_{L_{M-1}} & 1 \end{pmatrix}. \tag{13.16}$$

Here all variables with wave, i.e., $\widetilde{0}_k$, have dimension $1 \times (2^k - 1)$ not 1×2^k. We know that if we apply a chain of operators on the left-hand side, we obtain a result for essential-natural BCs. On the other hand, if we multiply two operators on the right, we get

$$\begin{pmatrix} \widetilde{P}^{L_M}_{L_M-1} & \widetilde{e}^{r}_{L_M}/2 \\ \widetilde{0}_{L_{M-1}} & 1 \end{pmatrix} \begin{pmatrix} \widetilde{P}^{L_M-1}_{L_M-2} & \widetilde{e}^{r}_{L_M-1}/2 \\ \widetilde{0}_{L_{M-2}} & 1 \end{pmatrix} = \begin{pmatrix} \widetilde{P}^{L_M}_{L_M-1}\widetilde{P}^{L_M-1}_{L_M-2} & \widetilde{P}^{L_M}_{L_M-1}\widetilde{e}^{r}_{L_M-1}/2 + \widetilde{e}^{r}_{L_M}/2 \\ \widetilde{0}_{L_{M-2}} & 1 \end{pmatrix}. \tag{13.17}$$

So we see that the structure of the matrix is preserved and in block $1,1$ we will accumulate the product of interpolation operator. As such, it is enough to delete the last row and the last column from the already known result.

Chapter 14

Neural multigrid architectures

14.1 Proposition 8.4.1

Without loss of generality we can assume that for grid H the radius of influence $r_H(\mathcal{N})$ is smaller than a grid size in a physical space, which is R for our problem. For $h = H/2^p, p > 1$ the radius of influence is $r_h(\mathcal{N}) = r_H(\mathcal{N})/2^p$. Let $\boldsymbol{b}_h$ be a discrete right hand side corresponding to a delta function in (8.4). If we start from zero initial guess $\boldsymbol{x}^{(0)} = 0$, an estimation to the initial error in L_2 norm reads

$$h^3 \left\|\boldsymbol{e}^{(0)}\right\|_2^2 \simeq 4\pi \int_0^R dr\ r^2 u(r)^2 = R/(12\pi), \tag{14.1}$$

and a lower bound on error for step $n = K$ $(Kr_h(\mathcal{N}) < R)$ reads

$$\begin{aligned} h^3 \left\|\boldsymbol{e}^{(K)}\right\|_2^2 \geq 4\pi \int_{Kr_h(\mathcal{N})}^R dr\ r^2 u(r)^2 = \\ = \big(R/(12\pi)\big) \left(1 - \frac{Kr_h(\mathcal{N})}{R}\right)^3. \end{aligned} \tag{14.2}$$

To derive (14.2), we assumed that our iterative method recovers the exact solution for all points that the network reached. Note that this argument is valid only because b_h is a sparse vector.

From $\left\|\boldsymbol{e}^{(n+1)}\right\| \leq \rho\left(\boldsymbol{I} - \boldsymbol{N}\boldsymbol{A}\right) \left\|\boldsymbol{e}^{(n)}\right\|$ we conclude

$$\rho(\boldsymbol{I} - \boldsymbol{\mathcal{N}}_h \boldsymbol{A})^K \geq \left\|\boldsymbol{e}^{(K)}\right\|_2 / \left\|\boldsymbol{e}^{(0)}\right\|_2 \geq \left(1 - \frac{Kr_h(\mathcal{N})}{R}\right)^{3/2}. \tag{14.3}$$

Because r_h can be arbitrary small for sufficiently small $h = H/2^p$, the expression in the brackets above can be arbitrary close to 1, which signifies arbitrary slow convergence.

Bibliography

[FL50] George E Forsythe and Richard A Leibler. "Matrix inversion by a Monte Carlo method". In: *Mathematics of Computation* 4.31 (1950), pp. 127–129.

[HS+52] Magnus Rudolph Hestenes, Eduard Stiefel, et al. *Methods of conjugate gradients for solving linear systems*. Vol. 49. 1. NBS Washington, DC, 1952.

[You54] David Young. "Iterative methods for solving partial difference equations of elliptic type". In: *Transactions of the American Mathematical Society* 76.1 (1954), pp. 92–111.

[Fed62] Radii Petrovich Fedorenko. "A relaxation method for solving elliptic difference equations". In: *USSR Computational Mathematics and Mathematical Physics* 1.4 (1962), pp. 1092–1096.

[Bra66] James H Bramble. "A second order finite difference analog of the first biharmonic boundary value problem". In: *Numerische Mathematik* 9.3 (1966), pp. 236–249.

[Rei66] John K Reid. "A method for finding the optimum successive over-relaxation parameter". In: *The Computer Journal* 9.2 (1966), pp. 200–204.

[Cho67] Alexandre Joel Chorin. "The numerical solution of the Navier-Stokes equations for an incompressible fluid". In: *Bulletin of the American Mathematical Society* 73.6 (1967), pp. 928–931.

[CFL67] R. Courant, K. Friedrichs, and H. Lewy. "On the partial difference equations of mathematical physics". In: *IBM J. Res. Develop.* 11 (1967), pp. 215–234. ISSN: 0018-8646. DOI: 10.1147/rd.112.0215. URL: https://doi.org/10.1147/rd.112.0215.

[Ros75] J Barkley Rosser. "Nine-point difference solutions for Poisson's equation". In: *Computers & Mathematics with Applications* 1.3-4 (1975), pp. 351–360.

[Man78] Thomas A Manteuffel. "Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration". In: *Numerische Mathematik* 31.2 (1978), pp. 183–208.

[GM79] Murli M Gupta and Ram P Manohar. "Direct solution of the biharmonic equation using noncoupled approach". In: *Journal of Computational Physics* 33.2 (1979), pp. 236–248.

[Alc+81] Raymond E. Alcouffe et al. "The multi-grid method for the diffusion equation with strongly discontinuous coefficients". In: *SIAM Journal on Scientific and Statistical Computing* 2.4 (1981), pp. 430–454.

[Sut84] Richard Stuart Sutton. "Temporal credit assignment in reinforcement learning". PhD thesis. University of Massachusetts Amherst, 1984.

[Ske86] Robert D Skeel. "Thirteen ways to estimate global error". In: *Numerische Mathematik* 48.1 (1986), pp. 1–20.

[Str86] Gilbert Strang. "A proposal for Toeplitz matrix calculations". In: *Studies in Applied Mathematics* 74.2 (1986), pp. 171–176.

[You87] David M Young. "An historical review of iterative methods". In: *Proceedings of the ACM conference on History of scientific and numeric computation*. 1987, pp. 117–123.

[Cha88] Tony F Chan. "An optimal circulant preconditioner for Toeplitz systems". In: *SIAM journal on scientific and statistical computing* 9.4 (1988), pp. 766–771.

[Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. The Morgan Kaufmann Series in Representation and Reasoning. Morgan Kaufmann, San Mateo, CA, 1988, pp. xx+552. ISBN: 0-934613-73-7.

[Sut88] Richard S Sutton. "Learning to predict by the methods of temporal differences". In: *Machine learning* 3.1 (1988), pp. 9–44.

[BPX90] James H Bramble, Joseph E Pasciak, and Jinchao Xu. "Parallel multilevel preconditioners". In: *Mathematics of Computation* 55.191 (1990), pp. 1–22.

[Rio92] Olivier Rioul. "Simple regularity criteria for subdivision schemes". In: *SIAM Journal on Mathematical Analysis* 23.6 (1992), pp. 1544–1576.

[TCK92] Charles H Tong, Tony F Chan, and CC Jay Kuo. "Multilevel filtering preconditioners: Extensions to more general elliptic problems". In: *SIAM Journal on Scientific and Statistical Computing* 13.1 (1992), pp. 227–242.

[Tyr92] Evgenij E Tyrtyshnikov. "Optimal and superoptimal circulant preconditioners". In: *SIAM Journal on Matrix Analysis and Applications* 13.2 (1992), pp. 459–473.

[Zha92] Xuejun Zhang. "Multilevel Schwarz methods". In: *Numerische Mathematik* 63.1 (1992), pp. 521–539.

[CS94] Edmond Chow and Yousef Saad. "Approximate inverse preconditioners for general sparse matrices". In: *Res. Rep. UMSI* 94.1.01 (1994).

[GE94] Ming Gu and Stanley C Eisenstat. "A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem". In: *SIAM journal on Matrix Analysis and Applications* 15.4 (1994), pp. 1266–1276.

[Hac94] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*. Vol. 95. Springer, 1994.

[She+94] Jonathan Richard Shewchuk et al. *An introduction to the conjugate gradient method without the agonizing pain*. 1994.

[Tai94] Mary M Tai. "A mathematical model for the determination of total area under glucose tolerance and other metabolic curves". In: *Diabetes care* 17.2 (1994), pp. 152–154.

[UNK94] Masataka Usui, Hiroshi Niki, and Toshiyuki Kohno. "Adaptive Gauss-Seidel method for linear systems". In: *International Journal of Computer Mathematics* 51.1-2 (1994), pp. 119–125.

[CC95] Tianping Chen and Hong Chen. "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems". In: *IEEE Transactions on Neural Networks* 6.4 (1995), pp. 911–917.

[GO95] Michael Griebel and Peter Oswald. "Tensor product type subspace splittings and multilevel iterative methods for anisotropic problems". In: *Advances in Computational Mathematics* 4.1 (1995), p. 171.

[Tib96] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[BS97] Ivo M Babuska and Stefan A Sauter. "Is the pollution effect of the FEM avoidable for the Helmholtz equation considering high wave numbers?" In: *SIAM Journal on numerical analysis* 34.6 (1997), pp. 2392–2423.

[Bre97] Marian Brezina. "Robust iterative methods on unstructured meshes". PhD thesis. University of Colorado at Denver, 1997.

[GG97] Walter Gander and Gene H. Golub. "Cyclic reduction—history and applications". In: *Scientific computing (Hong Kong, 1997)*. Springer, Singapore, 1997, pp. 73–85.

[GH97] Marcus J Grote and Thomas Huckle. "Parallel preconditioning with sparse approximate inverses". In: *SIAM Journal on Scientific Computing* 18.3 (1997), pp. 838–853.

[TB97] Lloyd N. Trefethen and David Bau III. *Numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997, pp. xii+361. ISBN: 0-89871-361-7. DOI: 10.1137/1.9780898719574. URL: https://doi.org/10.1137/1.9780898719574.

[CS98] Edmond Chow and Yousef Saad. "Approximate inverse preconditioners via sparse-sparse iterations". In: *SIAM Journal on Scientific Computing* 19.3 (1998), pp. 995–1023.

[FM98] Brendan J Frey and David JC MacKay. "A revolution: Belief propagation in graphs with cycles". In: *Advances in neural information processing systems*. 1998, pp. 479–485.

[Ste98] Gilbert W Stewart. *Matrix Algorithms: Volume 1: Basic Decompositions*. Vol. 1. Siam, 1998.

[SB98] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge, 1998.

[Jac99] John David Jackson. *Classical electrodynamics*. 1999.

[Had00] A Hadjidimos. "Successive overrelaxation (SOR) and related methods". In: *Journal of Computational and Applied Mathematics* 123.1-2 (2000), pp. 177–199.

[KO00] Marc C Kennedy and Anthony O'Hagan. "Predicting the output from a complex computer code when fast approximations are available". In: *Biometrika* 87.1 (2000), pp. 1–13.

[TOS00] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid*. Elsevier, 2000.

[WF00] Yair Weiss and William T Freeman. "Correctness of belief propagation in Gaussian graphical models of arbitrary topology". In: *Advances in neural information processing systems*. 2000, pp. 673–679.

[Min01] Thomas P Minka. "Expectation propagation for approximate Bayesian inference". In: *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 2001, pp. 362–369.

[SV01] Yousef Saad and Henk A Van Der Vorst. "Iterative solution of linear systems in the 20th century". In: *Numerical Analysis: Historical Developments in the 20th Century*. Elsevier, 2001, pp. 175–207.

[YFW01a] Jonathan S Yedidia, William T Freeman, and Yair Weiss. "Generalized belief propagation". In: *Advances in neural information processing systems*. 2001, pp. 689–695.

[YFW01b] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. *Bethe free energy, Kikuchi approximations, and belief propagation algorithms*. Tech. rep. TR2001-16. Cambridge, MA 02139: MERL - Mitsubishi Electric Research Laboratories, May 2001.

[ACF02] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* 47.2 (2002), pp. 235–256.

[Cia02] Philippe G Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[Ada+03] Mark Adams et al. "Parallel multigrid smoothing: polynomial versus Gauss-Seidel". In: *J. Comput. Phys.* 188.2 (2003), pp. 593–610. ISSN: 0021-9991. DOI: 10.1016/S0021-9991(03)00194-3.

[Bie+03] Lorenz T Biegler et al. "Large-scale PDE-constrained optimization: an introduction". In: *Large-Scale PDE-Constrained Optimization*. Springer, 2003, pp. 3–13.

[Ras03] Carl Edward Rasmussen. "Gaussian processes in machine learning". In: *Summer school on machine learning*. Springer. 2003, pp. 63–71.

[Saa03] Yousef Saad. *Iterative methods for sparse linear systems*. Second. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003, pp. xviii+528. ISBN: 0-89871-534-2. DOI: 10.1137/1.9780898718003. URL: https://doi.org/10.1137/1.9780898718003.

[Ste+03] J Steppeler et al. "Review of numerical methods for nonhydrostatic weather prediction models". In: *Meteorology and Atmospheric Physics* 82.1 (2003), pp. 287–301.

[YFW03] Jonathan S Yedidia, William T Freeman, and Yair Weiss. "Understanding belief propagation and its generalizations". In: *Exploring artificial intelligence in the new millennium* 8 (2003), pp. 236–239.

[CLL04] Guo Chen, Zhilin Li, and Ping Lin. *A fast finite difference method for biharmonic equations on irregular domains*. Tech. rep. North Carolina State University. Center for Research in Scientific Computation, 2004.

[Hes04] Tom Heskes. "On the uniqueness of loopy belief propagation fixed points". In: *Neural Computation* 16.11 (2004), pp. 2379–2413.

[PK04] Kurt Hermann Plarre and PR Kumar. "Extended message passing algorithm for inference in loopy Gaussian graphical models". In: *Ad Hoc Networks* 2.2 (2004), pp. 153–169.

[SWW04] Erik B Sudderth, Martin J Wainwright, and Alan S Willsky. "Embedded trees: Estimation of Gaussian processes on graphs with cycles". In: *IEEE Transactions on Signal Processing* 52.11 (2004), pp. 3136–3150.

[Wel04] Max Welling. "On the choice of regions for generalized belief propagation". In: *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press. 2004, pp. 585–592.

[Ama+05] Herbert Amann et al. *Analysis*. Vol. 1. Springer, 2005.

[New05] Robert E Newnham. *Properties of materials: anisotropy, symmetry, structure*. Oxford University Press on Demand, 2005.

[Pel05] Alessandro Pelizzola. "Cluster variation method in statistical physics and probabilistic graphical models". In: *Journal of Physics A: Mathematical and General* 38.33 (2005), R309.

[Tar05] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*. SIAM, 2005.

[WJ05] Roman Wienands and Wolfgang Joppich. *Practical Fourier analysis for multigrid methods*. Vol. 4. Numerical Insights. With 1 CD-ROM (Windows and UNIX). Chapman & Hall/CRC, Boca Raton, FL, 2005, pp. xiv+217. ISBN: 1-58488-492-4.

[WBJ05] John Winn, Christopher M Bishop, and Tommi Jaakkola. "Variational message passing." In: *Journal of Machine Learning Research* 6.4 (2005).

[YFW05] Jonathan S Yedidia, William T Freeman, and Yair Weiss. "Constructing free-energy approximations and generalized belief propagation algorithms". In: *IEEE Transactions on information theory* 51.7 (2005), pp. 2282–2312.

[Bis06a] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[Bis06b] Christopher M. Bishop. *Pattern recognition and machine learning*. Information Science and Statistics. Springer, New York, 2006, pp. xx+738. ISBN: 978-0387-31073-2. DOI: 10.1007/978-0-387-45528-0.

[EMK06] Gal Elidan, Ian McGraw, and Daphne Koller. "Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing". In: *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*. UAI'06. Cambridge, MA, USA: AUAI Press, 2006, pp. 165–173. ISBN: 0-9749039-2-2.

[MJW06] Dmitry M Malioutov, Jason K Johnson, and Alan S Willsky. "Walksums and belief propagation in Gaussian graphical models". In: *Journal of Machine Learning Research* 7.Oct (2006), pp. 2031–2064.

[Par+06] Michael L Parks et al. "Recycling Krylov subspaces for sequences of linear systems". In: *SIAM Journal on Scientific Computing* 28.5 (2006), pp. 1651–1674.

[LeV07] Randall J. LeVeque. *Finite difference methods for ordinary and partial differential equations*. Steady-state and time-dependent problems. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2007, pp. xvi+341. ISBN: 978-0-898716-29-0. DOI: 10.1137/1.9780898717839.

[Bic08] Danny Bickson. "Gaussian belief propagation: Theory and aplication". In: *arXiv preprint arXiv:0811.2518* (2008).

[Erl08] Yogi A Erlangga. "Advances in iterative methods and preconditioners for the Helmholtz equation". In: *Archives of Computational Methods in Engineering* 15.1 (2008), pp. 37–66.

[She+08] Ori Shental et al. "Gaussian belief propagation solver for systems of linear equations". In: *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*. IEEE. 2008, pp. 1863–1867.

[WJ+08] Martin J Wainwright, Michael I Jordan, et al. "Graphical models, exponential families, and variational inference". In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305.

[BS09] José M Bernardo and Adrian FM Smith. *Bayesian theory*. Vol. 405. John Wiley & Sons, 2009.

[Ise09a] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Second. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 2009, pp. xx+459. ISBN: 978-0-521-73490-5.

[Ise09b] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Cambridge university press, 2009.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models*. Adaptive Computation and Machine Learning. Principles and techniques. MIT Press, Cambridge, MA, 2009, pp. xxxvi+1231. ISBN: 978-0-262-01319-2.

[Koz09] Victor Kozyakin. "On accuracy of approximation of the spectral radius by the Gelfand formula". In: *Linear Algebra and its Applications* 431.11 (2009), pp. 2134–2141.

[Li09] Stan Z. Li. *Markov random field modeling in image analysis*. Third. Advances in Pattern Recognition. With forewords by Anil K. Jain and Rama Chellappa. Springer-Verlag London, Ltd., London, 2009, pp. xxiv+357. ISBN: 978-1-84800-278-4.

[BS10] Sean Buckeridge and Robert Scheichl. "Parallel geometric multigrid for global weather prediction". In: *Numerical Linear Algebra with Applications* 17.2-3 (2010), pp. 325–342.

[RDW10] Tyrone Rees, H Sue Dollar, and Andrew J Wathen. "Optimal solvers for PDE-constrained optimization". In: *SIAM Journal on Scientific Computing* 32.1 (2010), pp. 271–298.

[AT11] Haim Avron and Sivan Toledo. "Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix". In: *Journal of the ACM (JACM)* 58.2 (2011), pp. 1–34.

[Bra+11] A. Brandt et al. "Bootstrap AMG". In: *SIAM J. Sci. Comput.* 33.2 (2011), pp. 612–632. ISSN: 1064-8275. DOI: 10.1137/090752973. URL: https://doi.org/10.1137/090752973.

[RSB11] Dorit Ron, Ilya Safro, and Achi Brandt. "Relaxation-based coarsening and multiscale graph organization". In: *Multiscale Modeling & Simulation* 9.1 (2011), pp. 407–423.

[Saa11] Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition*. SIAM, 2011.

[Sha+11] Shai Shalev-Shwartz et al. "Online learning and online convex optimization". In: *Foundations and trends in Machine Learning* 4.2 (2011), pp. 107–194.

[CMS12] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3642–3649.

[Col12] Lothar Collatz. *The numerical treatment of differential equations*. Vol. 60. Springer Science & Business Media, 2012.

[GK12] Martin J Gander and Felix Kwok. "Chladni figures and the Tacoma bridge: motivating PDE eigenvalue problems via vibrating plates". In: *SIAM Review* 54.3 (2012), pp. 573–596.

[GV12] Gene H Golub and Charles F Van Loan. *Matrix computations*. Vol. 3. JHU press, 2012.

[Gon+12] Joseph E Gonzalez et al. "Powergraph: Distributed graph-parallel computation on natural graphs". In: *10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12)*. 2012, pp. 17–30.

[HA12] Ken Habgood and Itamar Arel. "A condensation-based application of Cramer's rule for solving large-scale linear systems". In: *Journal of Discrete Algorithms* 10 (2012), pp. 98–109.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[EGG12] Yousef El-Kurdi, Dennis Giannacopoulos, and Warren J Gross. "Relaxed Gaussian belief propagation". In: *2012 IEEE International Symposium on Information Theory Proceedings*. IEEE. 2012, pp. 2002–2006.

[PTA12] Richard H Pletcher, John C Tannehill, and Dale Anderson. *Computational fluid mechanics and heat transfer*. CRC press, 2012.

[CF13] Robert M. Corless and Nicolas Fillion. *A graduate introduction to numerical methods*. From the viewpoint of backward error analysis, With a foreword by John Butcher. Springer, New York, 2013, pp. xl+868. ISBN: 978-1-4614-8453-0. DOI: 10.1007/978-1-4614-8453-0.

[Gel+13] Andrew Gelman et al. *Bayesian data analysis*. CRC press, 2013.

[HK13] Philipp Hennig and Martin Kiefel. "Quasi-Newton methods: a new direction". In: *J. Mach. Learn. Res.* 14 (2013), pp. 843–865. ISSN: 1532-4435.

[HJ13] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Second. Cambridge University Press, Cambridge, 2013, pp. xviii+643. ISBN: 978-0-521-54823-6.

[Min13] Thomas P Minka. "Expectation propagation for approximate Bayesian inference". In: *arXiv preprint arXiv:1301.2294* (2013).

[Mni+13] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[MWJ13] Kevin Murphy, Yair Weiss, and Michael I Jordan. "Loopy belief propagation for approximate inference: An empirical study". In: *arXiv preprint arXiv:1301.6725* (2013).

[Ste13] Sergey S Stepanov. *Stochastic world*. Springer, 2013.

[ZTZ13] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The finite element method: its basis and fundamentals*. Seventh. Elsevier/Butterworth Heinemann, Amsterdam, 2013, pp. xxxviii+714. ISBN: 978-1-85617-633-0. DOI: 10.1016/B978-1-85617-633-0.00001-0.

[Bru+14] Joan Bruna et al. "Spectral networks and locally connected networks on graphs". English (US). In: *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*. 2014.

[Gel+14] Andrew Gelman et al. *Bayesian data analysis*. Third. Texts in Statistical Science Series. CRC Press, Boca Raton, FL, 2014, pp. xiv+661. ISBN: 978-1-4398-4095-5.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[MCP14] Stefan Magureanu, Richard Combes, and Alexandre Proutiere. "Lipschitz bandits: Regret lower bound and optimal algorithms". In: *Conference on Learning Theory*. PMLR. 2014, pp. 975–999.

[SDH14] Michael Schober, David K Duvenaud, and Philipp Hennig. "Probabilistic ODE solvers with Runge-Kutta means". In: *Advances in neural information processing systems*. 2014, pp. 739–747.

[Shl14] Jonathon Shlens. "A tutorial on principal component analysis". In: *arXiv preprint arXiv:1404.1100* (2014).

[Bra+15] James Brannick et al. "Local Fourier analysis of multigrid methods with polynomial smoothers and aggressive coarsening". In: *Numer. Math. Theory Methods Appl.* 8.1 (2015), pp. 1–21. ISSN: 1004-8979. DOI: 10.4208/nmtma.2015.w01si.

[CP15] Edmond Chow and Aftab Patel. "Fine-grained parallel incomplete LU factorization". In: *SIAM J. Sci. Comput.* 37.2 (2015), pp. C169–C193. ISSN: 1064-8275. DOI: 10.1137/140968896.

[CE15] Samuel N Cohen and Robert James Elliott. *Stochastic calculus and applications*. Vol. 2. Springer, 2015.

[Hen15] Philipp Hennig. "Probabilistic interpretation of linear solvers". In: *SIAM J. Optim.* 25.1 (2015), pp. 234–260. ISSN: 1052-6234. DOI: 10.1137/140955501. URL: https://doi.org/10.1137/140955501.

[HOG15] Philipp Hennig, Michael A Osborne, and Mark Girolami. "Probabilistic numerics and uncertainty in computations". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2179 (2015), p. 20150142.

[Owh15] Houman Owhadi. "Bayesian numerical homogenization". In: *Multiscale Modeling & Simulation* 13.3 (2015), pp. 812–828.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, [Cham]. 2015, pp. 234–241.

[Rus+15] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[Sha+15] Bobak Shahriari et al. "Taking the human out of the loop: A review of Bayesian optimization". In: *Proceedings of the IEEE* 104.1 (2015), pp. 148–175.

[Wat15] Andrew J Wathen. "Preconditioning". In: *Acta Numerica* 24 (2015).

[Aba+16] Martin Abadi et al. "TensorFlow: A system for large-scale machine learning". In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 2016, pp. 265–283.

[DRS16] Timothy A Davis, Sivasankaran Rajamanickam, and Wissam M Sid-Lakhdar. "A survey of direct methods for sparse linear systems". In: *Acta Numerica* 25 (2016), pp. 383–566.

[Goo+16] Ian Goodfellow et al. *Deep learning*. MIT press Cambridge, 2016.

[Hac16] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*. Second. Vol. 95. Applied Mathematical Sciences. Springer, [Cham], 2016, pp. xxiii+509. ISBN: 978-3-319-28483-5. DOI: 10.1007/978-3-319-28483-5.

[He+16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Hol16] Mark H Holmes. *Introduction to scientific computing and data analysis*. Vol. 13. Springer, 2016.

[Kri16] K. Krishnamoorthy. *Handbook of statistical distributions with applications*. Second. CRC Press, Boca Raton, FL, 2016, pp. xxvi+398. ISBN: 978-1-4987-4149-1.

[RLP16] J. Revels, M. Lubin, and T. Papamarkou. "Forward-Mode Automatic Differentiation in Julia". In: *arXiv:1607.07892 [cs.MS]* (2016).

[Bez+17] Jeff Bezanson et al. "Julia: a fresh approach to numerical computing". In: *SIAM Rev.* 59.1 (2017), pp. 65–98. ISSN: 0036-1445. DOI: 10.1137/141000671. URL: https://doi.org/10.1137/141000671.

[GPS+17] Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, et al. "Program synthesis". In: *Foundations and Trends® in Programming Languages* 4.1-2 (2017), pp. 1–119.

[Li+17] Lisha Li et al. "Hyperband: A novel bandit-based approach to hyperparameter optimization". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6765–6816.

[Lis17] Vladimir D Liseikin. *Grid generation methods*. Springer, 2017.

[Maz17] JE Mazur. *Learning and behavior (Eight ed.)* 2017.

[Owh17] Houman Owhadi. "Multigrid with rough coefficients and multiresolution operator decomposition from hierarchical information games". In: *SIAM Review* 59.1 (2017), pp. 99–149.

[Sil+17] David Silver et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm". In: *arXiv preprint arXiv:1712.01815* (2017).

[XZ17] Jinchao Xu and Ludmil Zikatanov. "Algebraic multigrid methods". In: *Acta Numerica* 26 (2017), pp. 591–721.

[Bar+18] Simon Bartels et al. "Probabilistic linear solvers: A unifying view". In: *arXiv preprint arXiv:1810.03398* (2018).

[Fra+18] Vincent François-Lavet et al. "An introduction to deep reinforcement learning". In: *arXiv preprint arXiv:1811.12560* (2018).

[Fra18] Peter I Frazier. "A tutorial on Bayesian optimization". In: *arXiv preprint arXiv:1807.02811* (2018).

[GN18] Arjun K Gupta and Daya K Nagar. *Matrix variate distributions*. Vol. 104. CRC Press, 2018.

[Li+18] Lisha Li et al. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *Journal of Machine Learning Research* 18.185 (2018), pp. 1–52. URL: http://jmlr.org/papers/v18/16-558.html.

[MLH18] Man Kwong Mak, Chun Sing Leung, and Tiberiu Harko. "Solving the nonlinear biharmonic equation by the Laplace-Adomian and Adomian Decomposition Methods". In: *arXiv preprint arXiv:1810.09544* (2018).

[PWG18] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. "Survey of multifidelity methods in uncertainty propagation, inference, and optimization". In: *Siam Review* 60.3 (2018), pp. 550–591.

[Bar+19] Simon Bartels et al. "Probabilistic linear solvers: a unifying view". In: *Stat. Comput.* 29.6 (2019), pp. 1249–1263. ISSN: 0960-3174. DOI: 10.1007/s11222-019-09897-7. URL: https://doi.org/10.1007/s11222-019-09897-7.

[Ber+19] Christopher Berner et al. "Dota 2 with large scale deep reinforcement learning". In: *arXiv preprint arXiv:1912.06680* (2019).

[Bri+19] François-Xavier Briol et al. "Probabilistic integration: A role in statistical computation?" In: *Statistical Science* 34.1 (2019), pp. 1–22.

[Coc+19a] Jon Cockayne et al. "A Bayesian conjugate gradient method (with discussion)". In: *Bayesian Anal.* 14.3 (2019), pp. 937–1012. ISSN: 1936-0975. DOI: 10.1214/19-BA1145. URL: https://doi.org/10.1214/19-BA1145.

[Coc+19b] Jon Cockayne et al. "Bayesian probabilistic numerical methods". In: *SIAM Review* 61.4 (2019), pp. 756–789.

[Gre+19a] Daniel Greenfeld et al. "Learning to Optimize Multigrid PDE Solvers". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2415–2423. URL: http://proceedings.mlr.press/v97/greenfeld19a.html.

[Gre+19b] Daniel Greenfeld et al. "Learning to optimize multigrid PDE solvers". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2415–2423.

[Hsi+19] Jun-Ting Hsieh et al. "Learning Neural PDE Solvers with Convergence Guarantees". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rklaWn0qK7.

[NA19] Emre O Neftci and Bruno B Averbeck. "Reinforcement learning in artificial and biological systems". In: *Nature Machine Intelligence* 1.3 (2019), pp. 133–143.

[OS19] Chris J Oates and Timothy John Sullivan. "A modern retrospective on probabilistic numerics". In: *Statistics and Computing* 29.6 (2019), pp. 1335–1351.

[Saa19] Yousef Saad. "Iterative methods for linear systems of equations: A brief historical journey". In: *Brenner, SC, Shparlinski, I., Shu, C.-W., Szyld, DB (eds.)* 75 (2019), pp. 197–216.

[SKK19] Jonas Schmitt, Sebastian Kuckuk, and Harald Köstler. "Optimizing Geometric Multigrid Methods with Evolutionary Computation". In: *arXiv preprint arXiv:1910.02749* (2019).

[Tro19] Joel A Tropp. "Matrix Concentration & Computational Linear Algebra". In: (2019).

[BK20] Markus Bachmayr and Vladimir Kazeev. "Stability of low-rank tensor representations and structured multilevel preconditioning for elliptic PDEs". In: *Foundations of Computational Mathematics* (2020), pp. 1–62.

[Bro+20] Jed Brown et al. "Tuning Multigrid Methods with Robust Optimization". In: *arXiv preprint arXiv:2001.00887* (2020).

[Dua+20] Jingliang Duan et al. "Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data". In: *IET Intelligent Transport Systems* 14.5 (2020), pp. 297–305.

[KDO20] Alexandr Katrutsa, Talgat Daulbaev, and Ivan Oseledets. "Black-box learning of multigrid parameters". In: *Journal of Computational and Applied Mathematics* 368 (2020), p. 112524.

[LS20] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[Li+20] Zongyi Li et al. "Fourier neural operator for parametric partial differential equations". In: *arXiv preprint arXiv:2010.08895* (2020).

[Luz+20] Ilay Luz et al. "Learning algebraic multigrid using graph neural networks". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6489–6499.

[Rei+20] Tim W Reid et al. "A Probabilistic Numerical Extension of the Conjugate Gradient Method". In: *arXiv preprint arXiv:2008.03225* (2020).

[Saa20] Yousef Saad. "Iterative methods for linear systems of equations: A brief historical journey". In: *75 years of mathematics of computation*. Vol. 754. Contemp. Math. Amer. Math. Soc., Providence, RI, 2020, pp. 197–215. DOI: 10.1090/conm/754/15141. URL: https://doi.org/10.1090/conm/754/15141.

[WH20] Jonathan Wenger and Philipp Hennig. "Probabilistic Linear Solvers for Machine Learning". In: *arXiv preprint arXiv:2010.09691* (2020).

[Fan21a] Vladimir Fanaskov. "Neural Multigrid Architectures". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533736.

[Fan21b] Vladimir Fanaskov. "Uncertainty calibration for probabilistic projection methods". In: *Statistics and Computing* 31.5 (2021), pp. 1–17. DOI: https://doi.org/10.1007/s11222-021-10031-9. URL: https://link.springer.com/article/10.1007/s11222-021-10031-9.

[NLK21] Guido Novati, Hugues Lascombes de Laroussilhe, and Petros Koumoutsakos. "Automating turbulence modelling by multi-agent reinforcement learning". In: *Nature Machine Intelligence* 3.1 (2021), pp. 87–96.

[OF21] Ivan Oseledets and Vladimir Fanaskov. "Direct optimization of BPX preconditioners". In: *Journal of Computational and Applied Mathematics* (2021), p. 113811. DOI: https://doi.org/10.1016/j.cam.2021.113811. URL: https://www.sciencedirect.com/science/article/pii/S0377042721004337.

[Set21] James P. Sethna. *Statistical mechanics*. Second. Vol. 14. Oxford Master Series in Physics. Entropy, order parameters, and complexity. Oxford University Press, Oxford, 2021, p. 464. ISBN: 978-0-19-886525-4.

[Tag+21] Ali Taghibakhshi et al. "Optimization-Based Algebraic Multigrid Coarsening Using Reinforcement Learning". In: *Advances in Neural Information Processing Systems* 34 (2021).

[Fan22] Vladimir Fanaskov. "Gaussian belief propagation solvers for nonsymmetric systems of linear equations". SIAM Journal on Scientific Computing. 2022. URL: https://epubs.siam.org/doi/abs/10.1137/19M1275139.

[FO22] Vladimir Fanaskov and Ivan Oseledets. "Spectral Neural Operators". In: *arXiv preprint arXiv:2205.10573* (2022).