Preconditioning strategies for stochastic elliptic partial differential equations

Nicolas Venkovic

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR
DE L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET

D'INFORMATIQUE

MATHÉMATIQUES APPLIQUÉES ET CALCUL SCIENTIFIQUE

Par **Nicolas VENKOVIC**

Stratégies de préconditionnement pour équations stochastiques elliptiques aux dérivées partielles

Sous la direction de : **Luc GIRAUD** et **Paul MYCEK**

Soutenue le 11 septembre 2023

Membres du jury:

M. Pietro CONGEDO	Directeur de Recherche	Inria, École Polytechnique	Examinateur
M. Olivier COULAUD	Directeur de Recherche	Inria, Université de Bordeaux	Président
M. Luc GIRAUD	Directeur de Recherche	Inria, Université de Bordeaux	Co-directeur de thèse
M. Julien LANGOU	Professeur	University of Colorado Denver	Rapporteur
M. Olivier LE MAÎTRE	Directeur de Recherche	CNRS, École Polytechnique	Examinateur
M. Paul MYCEK	Chercheur	Cerfacs	Co-directeur de thèse
M. Anthony NOUY	Professeur	École Centrale Nantes	Rapporteur
Mme. Nicole SPILLANE	Chargée de Recherche	CNRS, École Polytechnique	Examinatrice

Stratégies de préconditionnement pour équations stochastiques elliptiques aux dérivées partielles

Résumé: Nous nous intéressons à l'échantillonnage de Monte Carlo (MC) d'équations aux dérivées partielles (EDPs) elliptiques discrétisées à coefficients variables aléatoires. La charge de calcul dominante de ces applications consiste à résoudre un grand nombre de systèmes linéaires à matrice et second membre variables. Afin d'alléger cet effort, nous examinons, développons, implémentons et analysons des méthodologies efficaces et scalables pour les EDPs elliptiques stochastiques qui utilisent des combinaisons appropriées de solveurs itératifs et de préconditionneurs. Trois stratégies de préconditionnement sont développées et étudiées. Tout d'abord, des préconditionneurs parallèles usuels sont maintenus constants et utilisés pour résoudre tous les systèmes linéaires échantillonnés des simulations MC. Cette stratégie sert de point de comparaison pour les deux autres méthodes. Deuxièmement, des préconditionneurs basés sur la déflation de systèmes linéaires corrélés sont définis tout en échantillonnant le champ du coefficient aléatoire par méthode de Monte Carlo par chaînes de Markov. Différentes projections et méthodes de redémarrage de l'espace de recherche propre sont considérées pour l'approximation de l'information spectrale. Contrairement aux projections harmoniques de Rayleigh-Ritz, les projections de Rayleigh-Ritz évitent les applications de préconditionneur lors du recyclage des sous-espaces de Krylov, de sorte qu'elles doivent être privilégiées pour une meilleure performance. Le redémarrage épais et plus encore le redémarrage épais localement optimal de l'espace de recherche propre conduisent à des diminutions significatives du nombre d'itérations, en particulier pour les systèmes linéaires plus grands. La stratégie de préconditionnement basée sur la déflation, adaptée à l'inférence bayésienne, fonctionne particulièrement bien lors de l'utilisation de préconditionneurs dont l'action se traduit par des valeurs propres bien séparées aux extrémités du spectre. C'est le cas des préconditionneurs de Jacobi par blocs ainsi que des préconditionneurs basés sur la décomposition de domaine, mais pas du préconditionnement avec des multigrilles algébriques pour des équations isotropes. Troisièmement, nous partitionnons l'espace stochastique latent du champ du coefficient aléatoire en cellules de Voronoï, dont chacune est représentée par un champ du coefficient centroïdal sur la base duquel un préconditionneur est défini qui est utilisé pour résoudre les systèmes linéaires échantillonnés dont les champs du coefficient correspondants sont à l'intérieur de la cellule. Nous adoptons donc une représentation compacte du champ du coefficient aléatoire appelée quantifieur de Voronoï. Nous considérons différentes distributions de champs du coefficient centroïdaux et nous étudions les propriétés des stratégies de préconditionnement sous-jacentes en termes de nombre d'itérations moyen pour les simulations séquentielles et d'équilibrage de charge pour les simulations parallèles. Une distribution en particulier, qui minimise la distance moyenne entre le champ du coefficient et sa représentation compacte, minimise le nombre moyen d'itérations. Cette distribution est particulièrement adaptée aux simulations séquentielles. Une autre distribution est considérée qui conduit à des fréquences d'attribution égales pour les cellules, si bien qu'un même nombre de systèmes linéaires est résolu avec chaque préconditionneur du quantifieur. Cette stratégie réduit la répartition du nombre moyen d'itérations parmi les préconditionneurs, de sorte qu'elle est plus adaptée aux simulations parallèles. Enfin, une distribution basée sur des grilles déterministes avec une dimension stochastique qui augmente avec le nombre de préconditionneurs est proposée. Cette dernière distribution permet de s'affranchir des calculs préliminaires nécessaires pour déterminer la dimension optimale de l'espace stochastique approximatif pour un nombre donné de préconditionneurs.

Mots-clés : EDPs stochastiques, méthodes des sous-espaces de Krylov, déflation, quantifieurs de Voronoi

---

HiePACS, SATANAS, LaBRI
Université de Bordeaux, 351 cours de la Libération, 33405 Talence Cedex, France.

Preconditioning strategies for stochastic elliptic partial differential equations

Abstract: We are interested in the Monte Carlo (MC) sampling of discretized elliptic partial differential equations (PDEs) with random variable coefficients. The dominant computational load of such applications consists of solving large numbers of linear systems with variable matrix and right-hand side. As a means to alleviate this effort, we review, develop, implement and analyze efficient and scalable methodologies for stochastic elliptic PDEs that make use of appropriate combinations of iterative solvers and preconditioners. Three preconditioning strategies are developed and investigated. First, state-of-the-art parallel preconditioners are kept constant and used to solve all the sampled linear systems of MC simulations. This straightforward strategy serves as a point of comparison for the two other methods to improve upon. Second, preconditioners based on the deflation of correlated linear systems are defined while sampling the random coefficient field by Markov chain Monte Carlo. Different projections and restarting methods of the eigen-search space are considered for the online approximation of spectral information. As opposed to harmonic Rayleigh-Ritz projections, Rayleigh-Ritz projections are shown to avoid preconditioner applications when recycling the Krylov subspaces so that they should be favored for a better performance. Both the thick-restart and even more so the locally optimal thick-restart restarting of the eigen-search space lead to significant decreases of the number of solver iterations, particularly for larger linear systems. The preconditioning strategy based on deflation, which is adapted for Bayesian inference, works particularly well when using preconditoners whose action results in well-separated eigenvalues at the extremities of the spectrum. This is the case of block Jacobi preconditioners as well as preconditioners based on domain decomposition, but not when preconditioning with algebraic multigrids in the case of isotropic equations. Third, we partition the latent stochastic space of the random coefficient field into Voronoi cells, each of which is represented by a centroidal coefficient field on the basis of which a distinct preconditioner is defined which is used to solve the sampled linear systems whose corresponding coefficient fields lie within the cell. As such, we adopt a compact representation of the random coefficient field referred to as a Voronoi quantizer. We consider different distributions of centroidal coefficient fields, and we investigate the properties of the underlying preconditioning strategies in terms of expected number of solver iterations for sequential simulations, and of load balancing for parallel simulations. One distribution in particular, which minimizes the average distance between the coefficient field and its compact representation, minimizes the expected number of solver iterations. This distribution yields the smallest number of expected number of solver iterations so that it is particularly adapted for sequential simulations. Another distribution is considered which leads to equal attribution frequencies for all the cells, i.e., approximately the same number of linear systems is solved with each preconditioner of the quantizer. Our experiments show that this strategy yields a smaller spread of the expected number of solver iterations among the preconditioners so that it is more adapted for parallel simulations. Finally, a distribution based on deterministic grids with a stochastic dimension which increases with the number of preconditioners is proposed. This last distribution allows to bypass preliminary computations necessary to determine the optimal dimension of the approximating stochastic space for a given number of preconditioners.

Keywords: Stochastic PDEs, Krylov subspace methods, deflation, Voronoi quantizers

---

HiePACS, SATANAS, LaBRI
Université de Bordeaux, 351 cours de la Libération, 33405 Talence Cedex, France.

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgments

I wish to thank my close family for their lasting love and support—particularly my parents, Anna Goupil and Michel Venkovic, but also my wife, Mengyang Wang, my sisters, Naéma and Ilona Venkovic, as well as Marie Feix and Maxime Soubrenie. Not only do you all grant me the freedom to explore the world surrounding us, but you also tolerate my long absences. Thank you for your patience.

My journey in graduate school started twelve years ago and lead me to multiple stays in the US, France and Canada. During that time, I was fortunate enough to meet a lot of talented and kind-hearted people, some of whom marked me in both positive and significant ways. In particular, I think of Helia Sohrabi, Alireza Sheikh, Deniz Ozturk, George Weber, Gary Lin, Hwanpyo Kim and Yan Azdoud. I feel lucky to consider you as my friends, and cannot help but think that this journey would have been more difficult without you. Thank you for everything. I also want to thank Prof. Lori Graham-Brady, who did not only advise me during most of my stay at Johns Hopkins University, but who also supported my candidature to start a new PhD after close to five years of work together. I apologize for not completing my first PhD, and I look forward to complete our long due papers together.

Most importantly, I would like to thank those who made this thesis possible and guided me throughout three years at Cerfacs. Thank you to Luc Giraud, from whom I learned a lot, even though I would have liked to see and do much more. Thank you to Paul Mycek, of whom I was officially the first PhD student, and who also helped me discover a lot, often with great poise and care. Thank you to Olivier Le Maître for always coming up with new ideas on how to solve different problems as well as for taking an active role in shaping this thesis and, in general, for giving me valuable advices. I also wish to thank Michele Campassens. For my stay among the Cerfacs ALGO team on parallel algorithms, I also wish to thank Brigitte Yzel, Philippe Leleux, Pierre Matalon, Martin Kuehn, Benoît Pauwels, Vincent Darrigrand, Yongseok Jang, Rabeb Selmi, Jean Baptiste Latre, Carola Kruse, Anthony Weaver, Selime Gurol and Mbarek Fares as well as Iain Duff and Ulrich Ruede. Finally, I would like to thank Pietro Congedo, Olivier Coulaud, Julien Langou, Anthony Nouy and Nicole Spillane for agreeing to join my thesis committee, to read and comment my thesis as well as to attend and participate to my defense.

Résumé étendu

Les méthodes de Monte Carlo (MC) sont omniprésentes dans la quantification des incertitudes des systèmes physiques aléatoires. Malgré leur robustesse, ces méthodes deviennent trop coûteuses lorsqu'elles cherchent à calculer des estimations très précises de quantités d'intérêt dans de grands modèles informatiques. En particulier, lorsqu'il s'agit d'équations différentielles partielles (EDPs) elliptiques discrétisées avec des coefficients variables aléatoires, le calcul d'estimations précises d'une quantité d'intérêt repose sur l'échantillonnage d'un grand nombre de champs de coefficients, résultant en autant de systèmes linéaires creux dont les résolutions itératives dominent le temps de calcul global de la quantification de l'incertitude. Afin d'alléger cette charge de calcul, les solveurs itératifs classiques, qui sont bien compris dans le domaine spatial, doivent être adaptés pour exploiter des structures particulières dans l'espace stochastique. Dans ce contexte, l'objectif principal de cette thèse est d'examiner, développer, mettre en œuvre et analyser des méthodologies efficaces et scalables pour les EDPs elliptiques stochastiques qui utilisent des combinaisons appropriées de solveurs itératifs et de préconditionneurs.

Nous considérons trois stratégies de préconditionnement. Tout d'abord, des préconditionneurs parallèles usuels sont maintenus constants et utilisés pour résoudre tous les systèmes linéaires échantillonnés par simulations de MC. Deuxièmement, des préconditionneurs basés sur la déflation de systèmes linéaires corrélés sont définis lors de l'échantillonnage du champ du coefficient aléatoire par méthode de Monte Carlo par chaînes de Markov (MCMC). Troisièmement, nous partitionnons l'espace stochastique latent du champ du coefficient aléatoire en cellules, chacune étant représentée par un préconditionneur constant qui est utilisé pour résoudre les systèmes linéaires échantillonnés dont les champs de coefficient correspondants se trouvent dans la cellule.

Préconditionnement basé sur un seul préconditionneur constant

La stratégie de préconditionnement la plus simple pour les résolutions linéaires issues de la discrétisation spatiale d'une EDP elliptique et de l'échantillonnage de son coefficient variable aléatoire consiste à définir un seul préconditionneur constant construit de manière à préconditionner au mieux la résolution itérative d'un champ de coefficient particulier, c'est-à-dire , la référence. Un moyen d'obtenir les meilleures performances moyennes consiste à laisser la référence être un représentant central du champ du coefficient aléatoire, par exemple la moyenne ou la médiane.

Différents préconditionneurs qui ont la particularité d'être bien adaptés à une application parallèle sont considérés. Ce sont

- le Jacobi par blocs (bJ),
- la décomposition de domaine (DD) à un niveau,
- la DD à deux niveaux,
- la multigrille algébrique (AMG),

dont certains sont implémentés à l'aide du langage de programmation Julia. Parmi ces préconditionneurs, certains sont connus pour être scalables lorsqu'ils sont conçus pour le système linéaire sur lequel ils sont utilisés. C'est le cas des préconditionneurs DD à deux niveaux ainsi que des AMG. Cependant, le comportement de ces préconditionneurs lorsqu'ils sont utilisés pour d'autres systèmes linéaires à coefficients variables différents de la référence restent inconnus, notamment dans le cadre de simulations stochastiques dans lesquelles les réalisations de champs du coefficient peuvent être très différentes les unes des autres. Ici, nous cherchons à caractériser et analyser la scalabilité de tels préconditionneurs. Ces résultats sont également destinés à fournir une base de comparaison pour les autres stratégies de préconditionnement.

Préconditionnement basé sur la déflation de systèmes linéaires avec des chaînes de Markov de champs de coefficient

La stratégie de préconditionnement basée sur la déflation est bien adaptée aux applications dans lesquelles l'échantillonnage par MCMC est une nécessité. C'est par exemple le cas dans le contexte de l'inférence bayésienne du champ de coefficient, c'est-à-dire la modélisation inverse. L'échantillonnage des champs de coefficient par MCMC induit des corrélations entre les matrices successives et les seconds membres, respectivement. Cette corrélation se manifeste dans les valeurs propres et les vecteurs propres des matrices successives échantillonnées. En particulier, les vecteurs propres les moins dominants, qui correspondent aux valeurs propres gênantes des résolutions itératives, peuvent être approximés avec un minimum d'effort lors du recyclage des informations du sous-espace de Krylov d'une résolution itérative à une autre. À leur tour, ces informations sont utilisées pour développer un préconditionneur (singulier) basé sur la déflation [140] du système linéaire suivant.

La stratégie basée sur la déflation a un effet plus fort lorsqu'elle est utilisée avec des préconditionneurs dont l'action laisse des valeurs propres suffisamment bien séparées aux extrémités du spectre. Cet effet s'explique parce que les valeurs propres extrémales sont alors faciles à approximer, mais aussi parce que la déflation des valeurs propres extrémales conduit alors à une diminution plus importante du conditionnement. Les préconditionneurs bJ et DD conduisent à des valeurs propres bien séparées dans la partie la moins dominante du spectre pour les EDPs elliptiques discrétisées. En tant que telle, cette stratégie peut être considérée comme un outil complémentaire à l'approche de préconditionnement constant lors de la résolution d'EDPs elliptiques stochastiques dont les systèmes linéaires sont échantillonnés avec des chaînes de Markov de champs de coefficient. En revanche, les préconditionneurs AMG condensent plus efficacement le spectre des EDPs elliptiques discrétisées. En tant que tels, les préconditionneurs AMG sont moins sujets à l'amélioration de la convergence par déflation, car les valeurs pro-

pres moins bien séparées aux extrémités du spectre sont non seulement plus difficiles à approximer, mais leur déflation entraîne également des diminutions moins significatives du conditionnement. Compte tenu du coût additionnel de l'application de la déflation, cette stratégie n'est pas recommandée en combinaison avec un préconditionneur AMG pour résoudre l'équation de Poisson stochastique avec un coefficient isotrope variable. La déflation avec un préconditionneur AMG est cependant plus adaptée pour résoudre l'équation de Poisson stochastique avec un coefficient aléatoirement anisotrope qui donne des valeurs propres extrêmes plus séparées. Toujours avec des coefficients isotropes, une autre application appropriée de la déflation avec les préconditionneurs AMG pourrait être l'utilisation de schémas de discrétisation non standard tels que les approches de Galerkin discontinues. En effet, il a été montré dans [100] que pour la méthode hybridizable discontinuous Galerkin (HDG), les préconditionneurs AMG standards ne sont pas bien adaptés dans la mesure où ils ne compactent pas efficacement le spectre.

La déflation des systèmes linéaires corrélés est effectuée avec des vecteurs propres approximatifs obtenus en recyclant l'information spectrale tout au long de la résolution itérative par méthode de Krylov. Différentes projections sont considérées pour l'approximation des vecteurs propres les moins dominants, à savoir les méthodes de Rayleigh-Ritz (RR) et de Rayleigh-Ritz harmonique (HR). Nous détaillons comment construire efficacement les problèmes réduits de valeurs propres généralisées à la base de l'application des méthodes de projection RR et HR. Les méthodes de projection RR ont l'avantage que, contrairement aux projections HR, elles ne nécessitent aucune application de préconditionneur pour assembler le problème réduit aux valeurs propres. Différentes stratégies sont présentées et utilisées pour redémarrer l'espace de recherche propre tout au long de la résolution linéaire, à savoir le redémarrage épais (TR) et le redémarrage épais localement optimal (LO-TR). Le redémarrage de l'espace de recherche propre permet de tirer parti d'une plus grande partie des informations spectrales générées lors de la résolution linéaire. Les deux stratégies de redémarrage donnent de meilleures approximations de vecteurs propres et ont finalement un impact sur la déflation pour produire moins d'itérations de solveur. Lorsqu'elle est utilisée conjointement avec des projections RR, la stratégie de redémarrage LO-TR donne les meilleurs résultats, en particulier pour les systèmes linéaires plus grands, auquel cas nous avons observé jusqu'à une différence de 2X en termes de nombre d'itérations de solveur par rapport au cas sans stratégie de redémarrage.

L'échantillonnage des variables aléatoires latentes du champ de coefficient par MCMC induit une certaine corrélation qui entraîne des erreurs plus élevées des estimateurs statistiques tels que la moyenne de l'échantillon. En d'autres termes, il existe un sur-coût d'échantillonnage associée au MCMC pour l'estimation des quantités d'intérêt par rapport à l'échantillonnage direct par MC. Par conséquent, la stratégie de préconditionnement basée sur la déflation qui s'appuie sur l'échantillonnage MCMC du champ du coefficient aléatoire n'est pas une alternative viable à l'échantillonnage MC direct avec un préconditionneur constant.

Préconditionnement basé sur les quantifieurs de Voronoi des champs de coefficient

Dans le cas où l'EDP stochastique est traitée par échantillonnage MC direct du champ du coefficient aléatoire, on considère avoir un ensemble fini de préconditionneurs constants. Chaque préconditionneur est défini sur la base d'un champ de coefficient centroïdal associé. Précisément, nous nous appuyons sur une partition de l'espace stochastique latent du champ du coefficient aléatoire en cellules de Voronoi. Chaque cellule est associée à un champ de coefficient centroïdal. Lors de l'échantillonnage du champ du coefficient aléatoire, chaque réalisation du système linéaire est préconditionnée avec le préconditionneur associé au champ du coefficients centroïdal qui est le plus proche, selon une certaine distance, de la réalisation du champ du coefficient. Ainsi, nous adoptons une représentation compacte du champ du coefficient aléatoire appelée quantifieur. La définition d'un quantifieur de Voronoi du champ du coefficient est spécifiée à travers un ensemble fini de champs de coefficient centroïdaux. Nous considérons différentes manières de définir ces champs de coefficient centroïdaux et nous étudions les propriétés de leurs stratégies de préconditionnement sous-jacentes en termes de nombre moyen d'itérations du solveur pour les simulations séquentielles et de répartition de charge pour les simulations parallèles.

La première distribution de champs centroïdaux est définie de manière à minimiser la distance moyenne entre le champ de coefficient échantillonné et sa représentation compacte. Cette approche, qui définit un quantifieur stationnaire, produit le plus petit nombre moyen d'itérations de solveur. En tant que tel, elle est particulièrement adaptée aux simulations séquentielles dans lesquelles les variables aléatoires latentes du champ de coefficient sont échantillonnées au préalable, attribuées à leurs cellules de Voronoi correspondantes et enregistrées sur disque pour plus tard. La simulation séquentielle consiste alors à charger un à un les préconditionneurs en mémoire, à lire toutes les réalisations associées de variables aléatoires latentes, à assembler et à résoudre les systèmes linéaires associés avant de charger le préconditionneur suivant. La deuxième distribution de champs centroïdaux est également définie comme un quantifieur stationnaire, mais une transformation spécifique est considérée de manière à produire des fréquences d'attribution approximativement constantes. Autrement dit, chaque préconditionneur est utilisé pour résoudre approximativement le même nombre de systèmes linéaires. Nos expériences montrent que cette stratégie donne une plus petite répartition du nombre moyen d'itérations du solveur parmi les préconditionneurs. Ainsi, une simulation parallèle qui chargerait simultanément tous les préconditionneurs sur des mémoires distribuées se terminerait plus rapidement avec cette distribution qu'avec la distribution de la première approche. Ces deux premières distributions de champs centroïdaux nécessitent de spécifier a priori la dimension de l'espace stochastique, c'est-à-dire de spécifier la dimension de la variable aléatoire vectorielle latente de l'approximation du champ de coefficient. Cependant, nos expériences montrent que la dimension optimale de l'espace stochastique approximatif dépend du nombre de préconditionneurs. Pour peu de préconditionneurs, une plus petite dimension d'approximation de l'espace stochastique donne moins d'itérations du solveur, tandis que de plus grandes dimensions de l'approximation de l'espace stochastique deviennent nécessaires pour réduire le nombre d'itérations du solveur lors de l'utilisation de plus de préconditionneurs. En pratique, trouver la dimension optimale de l'espace stochas-

tique approximatif pour un nombre donné de préconditionneurs peut nécessiter des calculs préliminaires. Une alternative naturelle à ces deux premières distributions de champs centroïdaux est de définir une grille déterministe sur un espace stochastique dont la dimension augmente avec le nombre de préconditionneurs—c'est notre troisième distribution de champs centroïdaux. Nos expériences montrent que cette approche maintient toujours une dimension quasi-optimale de l'espace stochastique d'approximation en ce sens qu'elle produit toujours un petit nombre satisfaisant d'itérations de solveur et ne semble jamais stagner alors que nous continuons à augmenter le nombre de préconditionneurs. Ainsi, cette troisième distribution de champs centroïdaux donne une stratégie de préconditionnement pour laquelle il n'est pas nécessaire de faire une étude préalable pour connaître la dimension optimale de l'espace stochastique approché.

Enfin, nous tirons parti du fait que, même sur un système de mémoire distribuée, chaque nœud est susceptible d'avoir suffisamment de mémoire pour stocker plusieurs des préconditionneurs simultanément. Par conséquent, en utilisant algorithme k-means ou d'apprentissage compétitif, des groupes de petits nombres de préconditionneurs sont formés dont les champs de coefficient centroïdaux sont proches dans un certain sens. Chacun de ces clusters de champs centroïdaux est utilisé pour former une interpolation locale de préconditionneurs basée sur l'approche décrite dans [168]. C'est-à-dire que le préconditionnement proposé correspond alors à une projection optimale dans l'étendue linéaire de tous les préconditionneurs voisins d'un cluster donné. Cette méthode est appliquée avec des tailles de cluster relativement petites mais n'apporte aucune amélioration en termes d'itérations de solveur par rapport aux stratégies de préconditionnement originales basées sur des quantifieurs de Voronoi.

Extended summary

Monte Carlo (MC) methods are ubiquitous to the quantification of uncertainties in random physical systems. Despite their robustness, these methods become overly expensive when seeking to compute highly accurate estimates of quantities of interest (QoI) in large computational models. In particular, when dealing with discretized elliptic partial differential equations (PDEs) with random variable coefficients, the computation of accurate estimates of a QoI relies on sampling large numbers of coefficient fields, resulting in as many sparse linear systems whose iterative solves dominate the overall computing time of the uncertainty quantification (UQ). As a means to alleviate this computational load, classical iterative solvers, which are well understood in the spatial domain, need to be adapted to exploit particular structures in the stochastic space. In this context, the main goal of this thesis is to review, develop, implement and analyze efficient and scalable methodologies for stochastic elliptic PDEs that make use of appropriate combinations of iterative solvers and preconditioners.

We consider three preconditioning strategies. First, state-of-the-art parallel preconditioners are kept constant and used to solve all the sampled linear systems of MC simulations. Second, preconditioners based on the deflation of correlated linear systems are defined while sampling the random coefficient field by Markov chain Monte Carlo (MCMC). Third, we partition the latent stochastic space of the random coefficient field into cells, each of which is represented by a constant preconditioner which is used to solve the sampled linear systems whose corresponding coefficient fields lie within the cell.

Preconditioning based on a single constant preconditioner

The most straightforward preconditioning strategy for the linear solves which arise from the spatial discretization of an elliptic PDE and the sampling of its random variable coefficient consists of defining a single constant preconditioner built so as to best precondition the iterative solve of a particular coefficient field, i.e., the reference. A means towards best average performances is to let the reference be a central representative of the random coefficient field, e.g., the mean or the median.

Different preconditioners that have the particularity of being well-suited for parallel application are considered. Those are

- block Jacobi (bJ),
- one-level domain decomposition (DD),
- two-level DD,
- algebraic multigrid (AMG),

some of which are implemented using the Julia programming language. Among these preconditioners, some are known to be scalable when designed for the very linear system they are used on. This is the case of the two-level DD preconditioners as well as AMG. However, the behavior of these preconditioners when used for other linear systems with variable coefficients different from the reference remain unknown, particularly in the context of stochastic simulations in which the coefficient field realizations can be very different from one to another. Here, we aim to characterize and analyze the scalability of such preconditioners. These results are also intended to provide a basis of comparison for the other preconditioning strategies.

Preconditioning based on the deflation of linear systems with Markov chains of coefficient fields

The preconditioning strategy based on deflation is well-suited for applications in which sampling by MCMC is a necessity. This is for instance the case in the context of Bayesian inference of the coefficient field, i.e., inverse modeling. Sampling coefficient fields by MCMC induces correlations between successive matrices and right-hand sides, respectively. This correlation manifests itself in the eigenvalues and eigenvectors of successively sampled matrices. In particular, the least dominant eigenvectors, which correspond to hindering eigenvalues of the iterative solves, can be approximated with minimal effort upon recycling Krylov subspace information from one iterative solve to the next. In turn, this information is used to develop a (singular) preconditioner based on the deflation [140] of the subsequent linear system.

The strategy based on deflation has a stronger effect when used with preconditioners whose action leaves sufficiently well-separated eigenvalues at the extremities of the spectrum. This effect is explained because the extremal eigenvalues are then easy to approximate, but also because the deflation of extremal eigenvalues then leads to a more significant decrease of the condition number. Both bJ and DD preconditioners lead to well-seperated eigenvalues in the least-dominant part of the spectrum for discretized elliptic PDEs. As such, this strategy can be seen as a complementary tool to the constant preconditioning approach when solving stochastic elliptic PDEs whose linear systems are sampled with Markov chains of coefficient fields. On the other hand, AMG preconditioners more efficiently condense the spectrum of discretized elliptic PDEs. As such, AMG preconditioners are less prone to improvement of convergence by deflation, because the less well-separated eigenvalues at the extremeties of the spectrum are not only more difficult to approximate, but also their deflation yields less significant decreases of the condition number. Then, considering the additional cost of applying deflation, this strategy is not recommended in combination with an AMG preconditioner to solve the stochastic Poisson equation with a variable isotropic coefficient. Deflation along with an AMG preconditioner is however more suited to solve the stochastic Poisson equation with a randomly anisotropic coefficient which yields more well-seperated extremal eignevalues. Still with isotropic coefficients, another suitable application of deflation with AMG preconditioners might be when using involved discretization schemes such as discontinuous Galerkin approaches. Indeed, it was shown in [100] that for hybridizable discontinuous Galerkin (HDG) methods, standard AMG preconditioners are not well-adapted as they

do not efficiently compact the spectrum.

The deflation of the correlated linear systems is carried out with approximate eigenvectors obtained by recycling spectral information troughout the iterative solve by a Krylov method. Different projections are considered for the approximation of least dominant eigenvectors, namely Rayleigh-Ritz (RR) and harmonic Rayleigh-Ritz (HR) methods. We detail how to efficiently build reduced generalized eigenvalue problems at the root of the application of the RR and HR projection methods. RR projection methods have the advantage that, contrarily to HR projections, they do not require any preconditioner application to assemble the reduced eigenvalue problem. Different strategies are presented and used to restart the eigen-search space throughout the linear solve, namely thick-restart (TR) and locally optimal thick-restart (LO-TR). Restarting the eigen-search space allows to leverage more of the spectral information generated during the linear solve. Both restarting strategies yield better eigenvector approximations and enventually impact the deflation to yield fewer solver iterations. When used in conjunction with RR projections, the LO-TR restarting strategy yields the best results, especially for larger linear systems, in which case we observed up to a 2X difference in terms of numbers of solver iterations compared to not using any restarting strategy.

Sampling the latent random variables (RVs) of the coefficient field by MCMC induces some correlation which results in higher errors of statistical estimators such as the sample mean. In other words, there is a sampling overhead associated with MCMC for the estimation of QoIs in comparison to directly sampling by MC. Consequently, the preconditioning strategy based on deflation which relies on MCMC sampling of the random coefficient field is not a viable alternative to direct MC sampling with a constant preconditioner.

Preconditioning based on Voronoi quantizers of coefficient fields

In the case where the stochastic PDE is treated by direct MC sampling of the random coefficient field, we consider having a pool of constant preconditioners. Each preconditioner is defined on the basis of an associated centroidal coefficient field. Precisely, we rely on a partition of the latent stochastic space of the random coefficient field into Voronoi cells. Each cell is associated to a centroidal coefficient field. Upon sampling the random coefficient field, each realization of the linear system is preconditioned with the preconditioner associated with the centroidal coefficient field which is the closest, according to some distance, to the coefficient field realization. As such, we adopt a compact representation of the random coefficient field referred to as a quantizer. The definition of a Voronoi quantizer of the coefficient field is specified through a finite set of centroidal coefficient fields. We consider different ways of defining these centroidal coefficient fields, and we investigate the properties of their underlying preconditioning strategies in terms of expected number of solver iterations for sequential simulations, and of load balancing for parallel simulations.

The first distribution of centroidal fields is defined so as to minimize the average distance between the sampled coefficient field and its compact representation. This approach, which defines a stationary quantizer, yields the smallest number of expected number of solver iterations. As such, it is particularly adapted for sequential simulations in which the

latent RVs of the coefficient field are sampled beforehand, attributed to their corresponding Voronoi cells, and saved on disk for later. The sequential simulation then consists of loading preconditioners one-by-one on memory, reading all the associated realizations of latent RVs, assembling and solving the associated linear systems before loading the next precondtioner. The second distribution of centroidal fields is also defined as a stationary quantizer, but a specific transformation is considered so as to yield approximately constant attribution frequencies. That is, each preconditioner is used to solve approximately the same number of linear systems. Our experiments show that this strategy yields a smaller spread of the expected number of solver iterations among the preconditioners. As such, a parallel simulation which would simultaneously load all the preconditioners on distributed memories would finish quicker with this distribution than with the distribution of the first approach. These first two distributions of centroidal fields require to specify the dimension of the stochastic space a priori, i.e., to specify the dimension of the latent vectorial RV of the coefficient field approximation. However, our experiments show that the optimal dimension of the approximating stochastic space depends on the number of preconditioners. For few preconditioners, a smaller dimension of approximating stochastic space yields fewer solver iterations, whereas larger dimensions of approximating stochastic space become necessary to reduce the number of solver iterations when using more preconditioners. In practice, finding the optimal dimension of the approximating stochastic space for a given number of preconditioners may require preliminary computations. A natural alternative to these two first distributions of centroidal fields is to define a deterministic grid on a stochastic space whose dimension increases with the number of preconditioners—this is our third distribution of centroidal fields. Our experiments show that this approach always maintain a near-optimal dimension of the approximating stochastic space in that it always yields satisfactorily small numbers of solver iterations and never seems to stagnate as we keep increasing the number of preconditioners. As such, this third distribution of centroidal fields yields a preconditioning strategy for which one does not need to do a preliminary study to find out the optimal dimension of the approximating stochastic space.

Lastly, we leverage the fact that, even on a distributed memory system, each node is likely to have enough memory to store several of the preconditioners simultaneously. Therefore, using either a k-means or a competitive learning algortihm, clusters of small numbers of preconditioners are formed whose centroidal coefficient fields are close in some sense. Each such cluster of centroidal fields is used to form a local interpolation of preconditioners based on the approach in [168]. That is, the preconditioning offered then corresponds to an optimal projection in the linear span of all the neighboring preconditioners of a given cluster. This method is applied with relatively small cluster sizes but not does yield any improvement in terms of solver iterations compared to the original preconditioning strategies based on Voronoi quantizers.

Chapter 1

Introduction

1.1 Scientific context

1.1.1 Motivation

A large number of phenomena in physics, engineering, biology and finance can be modeled using partial differential equations (PDEs), that is, using equations which compute functions between various partial derivatives of multivariate functions. In particular, elliptic PDEs exhibit a certain degree of regularity and smoothness. They are generally used to model phenomena in which a system reaches a stable equilibrium. Examples of elliptic PDEs are the Laplace equation, the Poisson equation and, depending on the formulation, possibly even the Schrödinger equation. Standard Elliptic PDEs are deterministic, however, there exist situations in which either the forcing term, the boundary conditions or even the coefficient field of a an elliptic PDE are random. We refer to such equations as stochastic elliptic PDEs. Generally, we are interested in PDEs with a probabilistic description of uncertainty in the input data such that the model problem has the form

$$\mathcal{L}(\kappa)u = f \ \text{ in } \ \Omega \tag{1.1}$$

where $\mathcal{L}$ is an elliptic operator in a domain $\Omega \subset \mathbb{R}^d$, which depends on some coefficient $\kappa(x, \theta)$, with $x \in \Omega$, $\theta \in \Theta$, and Θ indicates the set of possible outcomes. Similarly, the forcing term $f = f(x, \theta)$ can be assumed random as well.

For the purpose of this work, we focus on the Poisson equation with a variable random coefficient. That is, we specifically consider the problem of finding $u : \overline{\Omega} \times \Theta \to \mathbb{R}$ such that

$$\nabla \cdot [\kappa(x, \theta)\nabla u(x, \theta)] = -f(x) \ \forall \ x \in \Omega \tag{1.2}$$

$$u(x, \theta) = g(x) \ \forall \ x \in \partial\Omega \tag{1.3}$$

is almost surely satisfied. A number of problems in physics and mechanics are modeled by Eq. (1.2); u may represent for instance a temperature, an electro-magnetic potential or the displacement of an elastic membrane fixed at the boundary under a transversal load of intensity f. A solution to Eq. (1.2) almost surely exists and is unique if the random coefficient field κ is almost surely bounded from below and above almost everywhere over

Ω. A detailed analysis of the existence and uniqueness of solutions to Eq. (1.2) can be found in [153].

1.1.2 State-of-the-art

The computation of statistics of the random solution to Eq. (1.2) has previously been done following two types of approaches, namely the approach based on functional representations of the random solution [12, 7], and the approach based on the Monte Carlo sampling of the solution [22, 94]. Although the present work focuses on the latter, we provide a bit of perspective by briefly introducing the two types of approaches.

1.1.2.1 Approaches based on functional representations

The approaches based on functional representations aim first at finding a finite representation of the random solution. In a second phase, the statistics of the solution can be evaluated by post-processing the available functional representation. Even prior to achieving the first step of these approaches, one needs to obtain a representation of the random coefficient field using a finite-dimensional probabilistic model. This representation is usually achieved through the use of a finite-dimensional latent random vector $\boldsymbol{\xi}(\theta)$ in order to parameterize some functional approximation of the random coefficient field. Perhaps the most common approximation is the truncated Karhunen-Loève expansion [73, 96], which we later present in Section 1.2. Then, the parametrization of the coefficient field enables the use of a functional representation of the random solution in the form of a spectral expansion

$$u(x,\theta) \approx u_M(x,\boldsymbol{\xi}(\theta)) := \sum_{\alpha=0}^{M} u_\alpha(x)\Psi_\alpha(\boldsymbol{\xi}(\theta)). \tag{1.4}$$

Once equipped with such an expansion, one can sample this representation as a means to compute statistics of the solution, or, alternatively, post-process the information inherently contained in the expansion [15, 79, 152].

Polynomial chaos (PC) expansions [52, 85] are a class of methods based on functional representations similar to Eq. (1.4) which have been used extensively to characterize the uncertainty of stochastic elliptic PDEs [162, 30, 29, 5, 109]. The basis $\{\Psi_\alpha\}_{\alpha=0}^{M}$ of random functions is set a priori as a finite polynomial basis, see generalized PC [167], so that in order to obtain a PC expansion, one is left to compute the deterministic coefficients $\{u_\alpha\}_{\alpha=0}^{M}$. These coefficients can be computed using either intrusive, i.e., Galerkin methods [36, 6, 4], regression [16, 15, 152, 1] or collocation [5, 109, 10].

The number $M+1$ of terms in the PC expansion depends on the dimension of the random latent vector used to represent the coefficient field. For a given level of representation error of the coefficient field, the dimension of the latent random vector increases as the correlation length of the coefficient field decreases. Meanwhile, the number of terms in the PC expansion increases very rapidly with the number of random variables, i.e., the dimension of $\boldsymbol{\xi}$. This effect is commonly known as the curse of dimensionality. Consequently, the cost of building a functional representation of the random solution of the form of Eq. (1.4) can be prohibitively expensive while the memory requirement can also be very high, especially when the coefficient field is highly variable. An alternative is then

to use low-rank representations which can be achieved by proper generalized decomposition [110, 112, 111]. Low-rank representations allow to achieve a linear dependence of M on the dimension of the latent random vector of the coefficient field making the approach more suitable for problems whose coefficient field require a parametrization based on large numbers of random variables.

In this work, we will not be using approaches based on functional representations. Besides being more involved than the approaches based on Monte Carlo sampling, another drawback of the approaches based on functional representations is that the representation error induced by the finite spectral expansion of the form given by Eq. (1.4) results into errors of the computed statistics which may govern the error of the computed estimators.

1.1.2.2 Approaches based on Monte Carlo sampling

The approaches based on Monte Carlo (MC) sampling consist of computing statistics using solutions to Eq. (1.2) for given realizations of the coefficient field. Each solution realization $u(\cdot,\theta)$ is the solution of a deterministic elliptic PDE associated with the corresponding realization $\kappa(\cdot,\theta)$ of the coefficient field. A common example of statistic computed by MC sampling is the average estimator of expectation of some quantity of interest $h(u(x,\cdot))$ given by

$$\widehat{\ell}(x) := \frac{1}{N}\sum_{\theta\in\hat{\Theta}} h(u(x,\theta))$$

where $\hat{\Theta}\subset\Theta$ is such that $N = |\hat{\Theta}|$. In order to get the solution $u(\cdot,\theta)$, we need to solve a deterministic elliptic PDE which, upon spatial discretization with n degrees of freedom as later described in Section 1.4, yields an n-by-n SPD linear system of the form

$$\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta). \tag{1.5}$$

We sometimes express $\mathbf{A}$ as a function of the coefficient field, i.e., $\mathbf{A}(\kappa)$. Hence, solving for the solution $u(\cdot,\theta)$ for all $\theta\in\hat{\Theta}$, which is necessary so as to compute the statistic $\widehat{\ell}(\cdot)$, boils down to solving multiple linear systems of the form given by Eq. (1.5).

The approaches based on MC sampling are easier to deploy than the approaches based on functional representations in that they do not require to build a spectral expansion prior to computing statistics. However, although the approaches based on MC sampling are reliable in that they do not induce any representation error, their main drawback is their low rate of convergence. Indeed, the error $|\mathbb{E}[h(u(x,\cdot))]-\widehat{\ell}(x)|$ is $\mathcal{O}(\sqrt{\mathbb{V}[h(u(x,\cdot))]/N})$, so that improving the accuracy of the estimator $\widehat{\ell}$ requires a significant increase of N. Other sampling methods exist, namely the multi-level Monte Carlo (MLMC) method [53] and quasi Monte Carlo (QMC) methods [38], which attempt to improve on the convergence rate of traditional MC methods. In paritucular, approaches based on MLMC and QMC sampling have been applied to stochastic elliptic PDEs [28, 9, 64, 57, 82, 128].

In order for statistics such as $\widehat{\ell}$ to be accurate estimators, large numbers of realizations need to be considered. Consequently, large numbers of linear systems of the form of Eq. (1.5) need to be solved. However, many applications require fine spatial discretizations so as to achieve accurate solutions for every single realization. As a result, in practice, the size n of the linear systems solved can be very large, so that the linear solves become

the dominant computational effort of the MC sampling approach. Hence, this is here that lie our work and contribution, in trying to find ways to accelerate the linear solves in the context of approaches based on MC sampling.

1.1.3 Two basic preconditioning strategies

The core of the work done in this thesis consists of solving large numbers of linear systems of the form of Eq. (1.5). Since every such linear system is assembled through spatial discretization of an elliptic PDE, the matrix $\mathbf{A}(\theta)$ tends to be sparse. As the number of degrees of freedom increases, we are left with solving large numbers of large sparse linear systems. For a better performance, it is common to favor iterative methods over direct methods to solve such problems. Hence, here, we focus on the iterative solve of Eq. (1.5) for large numbers of realizations. In particular, we rely on PCG using an SPD preconditioner which somehow approximates $\mathbf{A}$. In this chapter, we consider the two most straightforward preconditioning strategies.

First, there is the case in which the preconditioner $\mathbf{M}_\bullet(\theta)$ denominated by $\bullet$ is re-defined for every matrix realization $\mathbf{A}(\theta)$. We refer to this strategy as *realization-dependent ideal preconditioning*. The preconditioners considered in this work are such that their application is easily parallelizable, in particular, we use block-Jacobi (bJ) preconditioners, preconditioners based on non-overlapping domain decompostion as well as algebraic multigrids.

Albeit being ideal in terms of effect of the conditioning number, realization-dependent ideal preconditioning is not feasible for a Monte Carlo simulation as it would entail a potentially significant set-up cost of the preconditioner for every realization. Thus, re-defining the preconditioner for every matrix realization is only used as a point of comparison to benchmark the performance of other more practical strategies. The most straightforward strategy consists of considering a matrix $\mathbf{A}_0$ based on which the inverse of the preconditioner $\mathbf{M}_{0,\bullet}$ denominated by $\bullet$, or simply the data structures necessary for the application of its inverse $\mathbf{M}_{0,\bullet}^{-1}$ is assembled. Then, this preconditioner is applied, in turn, to all the realizations $\mathbf{A}(\theta)$. That is, the properties of the investigated preconditioners should focus on the spectrum of $\mathbf{M}_{0,\bullet}^{-1}\mathbf{A}(\theta)$ in which the preconditioner is constant but $\mathbf{A}(\theta)$ can be any matrix in the manifold of operators.

1.1.4 Outline

This chapter provides all the necessary background for further developments and applications in the next chapters. First, we present how random coefficient fields can be represented by means of Karhunen-Loève expansions, we also show how to compute these expansions, specifically in the context of large data problems using distributed computers. Second, we highlight the different sampling strategies of the underlying latent random vectors of random coefficient fields. Third, we present the finite element method which we use for the spatial discretization of PDEs after having sampled realizations of the coefficient field. Fourth, we present Krylov subspace methods for symmetric positive definite (SPD) matrices. Fifth, we introduce different state-of-the-art preconditioners which are well-suited for parallel applications. Lastly, we provide numerical examples in which we illustrate the computation of Karhunen-Loève expansions on distributed parallel computers and the linear solves which arise from sampling discretized stochastic elliptic PDEs.

In particular, we illustrate two basic preconditioning strategies for the linear solves of large numbers of linear systems which arise during the Monte Carlo sampling of stochastic elliptic PDEs. Although the numerical methods presented in chapter are not new, we provide new Julia implementations of some state-of-the-art preconditioners as well as a distributed parallel implementation of the computation of Karhunen Loève representations based on domain decomposition. The numerical results provided in the end of the chapter serve as basis of comparison against different preconditioning strategies presented throughout this work. The source code of all the methods implemented, the corresponding numerical examples and the post-processing scripts can all be found at `https://github.com/venkovic/julia-phd-krylov-spdes`.

1.2 Representation of random coefficient fields

In a direct Monte Carlo (MC) approach, we intend to draw instances of Eq. (1.5). To do so, we need to be able to represent and sample the coefficient field $\kappa(x,\theta)$ so as to draw realizations and then proceed by spatial discretization of Eq. (1.2) to obtain an equation of the form of Eq. (1.5) for each realization. Different methods exist to represent and simulate random fields like $\kappa(x,\theta)$. Here, we focus on the Karhunen-Loève (KL) representation [73, 96] as it allows to simulate non-stationary anisotropic random processes on unstructured grids. Another method which is not used in this work is the circulant embedding [39, 164, 80, 143, 113], which embeds the covariance matrix of a structured grid in a matrix of a particular structure to solve an eigenvalue problem using fast Fourier transforms (FFT). In addition to focusing on discrete Karhunen-Loève (KL) representations, we resort to domain decomposition. Domain decomposition was used by Contreras et al. [31] to compute KL expansions over domains of large dimensions in comparison to the characteristic length of the represented stochastic process. The method, referred to as DD-KL, follows a divide-and-conquer paradigm and relies on an atlas of local truncated KL expansions with non-overlapping supports, i.e., sub-domains. The computation of these local expansions is decoupled and thus naturally parallelizable on distributed machines so as to allow for the construction of accurate representations while circumventing the limits of memory availability on a single computer. In the case of Gaussian processes, the resulting DD-KL representation consists of local KL expansions with iid Gaussian RVs. These locally independent RVs are however correlated from one sub-domain to another.

1.2.1 Karhunen-Loève representation of random coefficient fields

Consider a probability space $(\Theta, \Sigma_\Theta, \mu_\Theta)$ where Θ is the set of events, Σ_Θ is a sigma-algebra over Θ and μ_Θ is a probability measure. We denote the expectation of a random variable X by $\mathbb{E}[X]$, defined by

$$\mathbb{E}[X] = \int_\Theta X(\theta) \mathrm{d}\mu_\theta(\theta). \tag{1.6}$$

Let us denote the space of second order random variables by $L^2(\Theta)$. That is for each $X \in L^2(\Theta)$ we have $\mathbb{E}[X^2] < \infty$. We consider that the domain Ω is a bounded subset of

$\mathbb{R}^d$ and define $L^2(\Omega)$ as the space of square integrable functionals $f : x \in \Omega \mapsto f(x) \in \mathbb{R}$. We denote by $\|\cdot\|_\Omega$ the norm of $L^2(\Omega)$ induced by the scalar product $\langle\cdot,\cdot,\rangle_\Omega$. That is

$$\|f\|_\Omega^2 = \langle f, f\rangle_\Omega = \int_\Omega |f(x)|^2 \mathrm{d}x \;\; \forall f \in L^2(\Omega). \tag{1.7}$$

Then, we denote by $L^2(\Omega,\Theta)$ the space of real valued second order processes $\kappa : \Omega\times\Theta \to \mathbb{R}$ such that $\kappa(\cdot,\theta) \in L^2(\Omega)$, $\kappa(x,\cdot) \in L^2(\Theta)$ and

$$\mathbb{E}\left[\|\kappa(\cdot,\theta)\|_\Omega^2\right] < \infty \quad \Longleftrightarrow \quad \kappa \in L^2(\Omega,\Theta). \tag{1.8}$$

Let $\kappa \in L^2(\Omega,\Theta)$ be a random process with zero mean, i.e., such that $\mathbb{E}[\kappa(x,\cdot)] = 0 \;\; \forall x \in \Omega$, with known covariance $C : \Omega \times \Omega \to \mathbb{R}$. That is

$$C(x,x') = \mathbb{E}[\kappa(x,\cdot)\kappa(x',\cdot)]. \tag{1.9}$$

The truncated Karhunen-Loève (KL) approximation κ_N of a second order stochastic process κ consists of a N-term expansion in which each term is the product of a deterministic function of $L^2(\Omega)$ with a random variable of $L^2(\Theta)$. The truncated KL expansion κ_N is defined so as to minimize the representation error $\kappa - \kappa_N$ in the $L^2(\Omega,\Theta)$ sense. Since covariance functions are symmetric and non-negative, it can be shown that the truncated KL expansion is given by

$$\kappa_N(x,\theta) := \sum_{\alpha=1}^{N} \sqrt{\lambda_\alpha}\xi_\alpha(\theta)\Phi_\alpha(x), \tag{1.10}$$

where $(\lambda_\alpha, \Phi_\alpha)$ is the α-th dominant eigen-pair of the covariance function where Φ_α is a normalized eigen-function. That is, $(\lambda_\alpha, \Phi_\alpha)$ is solution of the integral equation

$$\int_\Omega C(x,x')\Phi(x')\mathrm{d}x' = \lambda\Phi(x), \;\; \langle\Phi,\Phi\rangle_\Omega = 1. \tag{1.11}$$

The random variables ξ_α, also referred to as the stochastic coordinates of κ_N, are orthonormal, i.e., $\mathbb{E}[\xi_\alpha\xi_\beta] = \delta_{\alpha\beta}$. That is, ξ_α and ξ_β are uncorelated random variables for $\alpha \neq \beta$. Owing to the structure of the covariance function, the eigenvalues are non-negative so that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \geq 0$ while an energy criterion arises naturally for the truncation of the expansion. Then, N can be picked so as to satisfy some error tolerance in the $L^2(\Omega,\Theta)$ norm. Specifically, one sets N so as to satisfy the following inequality for some prescribed error tolerance $0 < \delta < 1$

$$\mathbb{E}[\|\kappa - \kappa_N\|_\Omega^2] = \mathbb{E}[\|\kappa\|_\Omega^2] - \sum_{\alpha=1}^{N}\lambda_\alpha \leq \mathbb{E}[\|\kappa\|_\Omega^2]\delta^2. \tag{1.12}$$

In most practical cases, an exact solution $(\lambda_\alpha, \Phi_\alpha) \in \mathbb{R}\times L^2(\Omega)$ of the integral Eq. (1.11) cannot be found, and we have to rely on numerical methods to build approximations of the eigen-pairs [14]. Here, we proceed with a Galerkin projection method. That is, a finite dimensional space $\mathcal{V}$ is introduced such that $\mathcal{V} = \mathrm{span}\{v_1(x), \ldots, v_Q(x)\}$ in which $v_k \in L^2(\Omega)$ is a basis function. Then, letting Φ be approximated by $\Phi^h(x) := \sum_{k=1}^{Q} c_k v_k(x) \in \mathcal{V}$

and substituting Φ by its approximation in Eq. (1.11) leads to the following residual:

$$r(x) := \lambda\Phi^h(x) - \int_\Omega C(x,x')\Phi^h(x')\mathrm{d}x' = \sum_{k=1}^{Q} c_k \left(\lambda v_k - \int_\Omega C(x,x')v_k(x')\mathrm{d}x'\right). \quad (1.13)$$

The vector $\mathbf{c} = [c_1, \ldots, c_Q]^T$ is chosen by forcing $r(x)$ to be orthogonal to all functions in $\mathcal{V}$. That is

$$\langle r, u\rangle_\Omega = 0 \ \ \forall u \in \mathcal{V}. \quad (1.14)$$

Substituting Eq. (1.13) into Eq. (1.14) leads to a discrete linear generalized eigenvalue problem of the form

$$\mathbf{K}\mathbf{c} = \lambda\mathbf{M}\mathbf{c} \quad (1.15)$$

where $\mathbf{K}$ and $\mathbf{M}$ are non-negative symmetric matrices of $\mathbb{R}^{Q\times Q}$ with components

$$K_{ij} = \int_\Omega\int_\Omega C(x,x')v_i(x')v_j(x)\mathrm{d}x'\mathrm{d}x \ \text{ and } \ M_{ij} = \langle v_i, v_j\rangle_\Omega. \quad (1.16)$$

1.2.2 Parallel KL decomposition

Domain decomposition was used by Contreras et al. [31] to compute KL expansions over domains of large dimensions in comparison to the characteristic length of the represented stochastic process. This approach starts by partitioning the domain Ω into n_d non-overlapping subdomains

$$\overline{\Omega} = \overline{\cup_{d=1}^{n_d}\Omega_d} \ \ , \ \ \Omega_i \cap \Omega_{j\neq i} = \emptyset. \quad (1.17)$$

For each subdomain Ω_d, some local eigenmodes $\tilde{\phi}_\beta^{(d)} : \Omega_d \to \mathbb{R}$ are then introduced and defined as the solutions of

$$\int_{\Omega_d} C(x,x')\tilde{\phi}_\beta^{(d)}(x')\mathrm{d}x' = \lambda_\beta^{(d)}\tilde{\phi}_\beta^{(d)}(x) \ \ , \ \ \|\tilde{\phi}_\beta^{(d)}\|_{\Omega_d} = 1 \quad (1.18)$$

where the restriction of the norm in $L^2(\Omega)$ to the subdomain Ω_d is denoted by $\|\cdot\|_{\Omega_d}$. Proceeding as before with a Galerkin projection method leads to a set of local discrete eigenvalue problems of the form

$$\mathbf{K}^{(d)}\mathbf{c}^{(d)} = \lambda^{(d)}\mathbf{M}^{(d)}\mathbf{c}^{(d)} \ \ , \ \ d = 1, \ldots, n_d. \quad (1.19)$$

It is clear that $\tilde{\phi}_\beta^{(d)}$ is an eigenfunction of the covariance $C(x,x')$ restricted to the d-th subdomain. These local eigenfunctions are extended to the global domain Ω by defining

$$\phi_\beta^{(d)}(x) := \begin{cases} \tilde{\phi}_\beta^{(d)}(x), & x \in \Omega_d \\ 0, & x \notin \Omega_d \end{cases} \ \ \forall x \in \overline{\Omega}. \quad (1.20)$$

Because $\phi_\beta^{(d)}$ and $\phi_{\beta'}^{(d)}$ are orthonormal in Ω_d, in light of Eq. (1.20), we have

$$\left\langle \phi_\beta^{(d)}, \phi_{\beta'}^{(d')} \right\rangle = \begin{cases} 1, & \text{if } d = d' \text{ and } \beta = \beta' \\ 0, & \text{otherwise.} \end{cases} \tag{1.21}$$

For each subdomain Ω_d the $m_d > 0$ dominant eigenpairs are retained according to a criterion discussed later. The n_d sets of dominant eigenfunctions are collected to form an orthonormal reduced basis $\mathcal{B}$ of $L^2(\Omega)$:

$$\mathcal{B} := \cup_{d=1}^{n_d} \mathcal{B}_d \ , \ \ \mathcal{B}_d := \left\{ \phi_\beta^{(d)}, \beta = 1, \ldots, m_d \right\}. \tag{1.22}$$

The linear span of $\mathcal{B}$ is denoted by $\mathcal{V}_\mathcal{B}$. Then, an approximation $\hat{\Phi} \in \mathcal{V}_\beta$ of the global modes of Eq. (1.11) is sought such that

$$\Phi(x) \approx \hat{\Phi}(x) = \sum_{d=1}^{n_d} \sum_{\beta=1}^{m_d} a_\beta^{(d)} \phi_\beta^{(d)}(x). \tag{1.23}$$

We set $\mathbf{a}^{(d)} := [a_1^{(d)}, \ldots, a_{m_d}^{(d)}]$, the vector of the local coordinates of $\hat{\Phi}$ for $x \in \Omega_d$. Applying the Galerkin projection method, we have

$$\left\langle \int_\Omega C(x, x') \hat{\Phi}(x') \mathrm{d}x', \phi_\beta^{(d)} \right\rangle_\Omega = \Lambda \left\langle \hat{\Phi}, \phi_\beta^{(d)} \right\rangle_\Omega . \tag{1.24}$$

The approximate eigenfunctions solve the following discrete eigenvalue problem

$$\begin{bmatrix} \hat{\mathbf{K}}_{11} & \hat{\mathbf{K}}_{12} & \ldots & \hat{\mathbf{K}}_{1n_d} \\ \hat{\mathbf{K}}_{21} & \hat{\mathbf{K}}_{22} & \ldots & \hat{\mathbf{K}}_{2n_d} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{K}}_{n_d1} & \hat{\mathbf{K}}_{n_d2} & \ldots & \hat{\mathbf{K}}_{n_dn_d} \end{bmatrix} \begin{bmatrix} \mathbf{a}^{(1)} \\ \mathbf{a}^{(2)} \\ \vdots \\ \mathbf{a}^{(n_d)} \end{bmatrix} = \Lambda \begin{bmatrix} \mathbf{a}^{(1)} \\ \mathbf{a}^{(2)} \\ \vdots \\ \mathbf{a}^{(n_d)} \end{bmatrix}, \tag{1.25}$$

where the block matrices $\hat{\mathbf{K}}_{ij} \in \mathbb{R}^{m_i \times m_j}$ have for respective components

$$(\hat{\mathbf{K}}_{ij})_{\alpha\beta} = \int_{\Omega_i} \int_{\Omega_j} C(x, x') \phi_\alpha^{(i)}(x) \phi_\beta^{(j)}(x') \mathrm{d}x \mathrm{d}x' \ , \ \ 1 \leq \alpha \leq m_i \ , \ \ 1 \leq \beta \leq m_j. \tag{1.26}$$

Eq. (1.25) is referred to as the condensed eigenvalue problem. The dimension of this problem is

$$n_t = \sum_{d=1}^{n_d} m_d = \mathrm{card}(\mathcal{B}). \tag{1.27}$$

It is easily shown that the matrix $\hat{\mathbf{K}} \in \mathbb{R}^{n_t \times n_t}$ is symmetric and positive definite if the covariance function is such that

$$\|u\|_\Omega > 0 \quad \Longrightarrow \quad \left\langle u, \int_\Omega C(\cdot, x) u(x) \mathrm{d}x \right\rangle_\Omega > 0 \tag{1.28}$$

for all $u \in L^2(\Omega)$. This assumption is satisfied for most covariance functions, in particular for the whole class of Matérn covariance functions [31]. Consequently, the n_t eigenvalues Λ_α of $\hat{\mathbf{K}}$ can be ordered with decreasing magnitude such that

$$\Lambda_1 \geq \Lambda_2 \geq \cdots \geq \Lambda_{n_t} \geq 0. \tag{1.29}$$

Then, we can select the smallest $\hat{N}$, $1 \leq \hat{N} \leq n_t$, such that for a prescribed relative error tolerance $0 \leq \delta \leq 1$ we have

$$\sum_{\alpha=\hat{N}+1}^{n_t} \Lambda_\alpha \leq \frac{\delta^2}{2} \sum_{\alpha=1}^{n_t} \Lambda_\alpha. \tag{1.30}$$

The truncated approximation of κ is then given by

$$\kappa(x,\theta) \approx \hat{\kappa}_N(x,\theta) := \sum_{\alpha=1}^{\hat{N}} \sqrt{\Lambda_\alpha}\hat{\xi}_\alpha(\theta)\hat{\Phi}_\alpha(x), \tag{1.31}$$

where

$$\hat{\Phi}_\alpha(x) = \sum_{d=1}^{n_d} \sum_{\beta=1}^{m_d} a_{\alpha,\beta}^{(d)} \phi_\beta^{(d)}(x) \tag{1.32}$$

is the eigenfunction corresponding to Λ_α.

As shown in [31], n_t is fixed by the requested accuracy and not by the size of the discretization space. Eq. (1.31) is referred to as the DD-KL expansion. The algorithm used to compute a DD-KL expansion is provided in Algo. 1.

Algorithm 1 Computation of the DD-KL representation

Input: domain partition $\Omega_1, \ldots \Omega_{n_d}$, covariance function $C : \Omega \times \Omega \to \mathbb{R}$
Output: DD-KL expansion

for $d = 1, \ldots, n_d$ **do**
  Discretize the local integral Eq. (1.18) to get $\mathbf{K}^{(d)}$ and $\mathbf{M}^{(d)}$
  Solve the local generalized eigenvalue problem $\mathbf{K}^{(d)}\tilde{\phi}^{(d)} = \lambda^{(d)}\mathbf{M}^{(d)}\tilde{\phi}^{(d)}$
end for
for $d = 1, \ldots, n_d$ **do**
  for $d' = 1, \ldots, n_d$ **do**
    for $\alpha = 1, \ldots, m_d$ **do**
      for $\beta = 1, \ldots, m_{d'}$ **do**
        Compute $(\mathbf{K}_{dd'})_{\alpha\beta} = \int_{\Omega_d}\int_{\Omega^{d'}} C(x,x')\phi_\alpha^{(d)}(x)\phi_\beta^{(d')}(x')\mathrm{d}x\mathrm{d}x'$
      end for
    end for
  end for
end for
Assemble and solve the reduced eigenvalue problem ▷ See Eq. (1.25)
Get approximated global eigenfunctions ▷ See Eq. (1.23)

The method of Contreras et al. [31] introduces two different sources of error in the approximation of κ by $\hat{\kappa}_{\hat{N}}$. First, an error is introduced when representing the eigenfunction of κ in the finite dimensional space $\mathcal{V}_{\mathcal{B}}$ built on the local bases, see Eq. (1.23). Let us denote by $\kappa_{\mathcal{B}}$ the projection of κ on $\mathcal{V}_{\mathcal{B}}$:

$$\kappa_{\mathcal{B}}(x,\theta) = \sum_{d=1}^{n_d} \sum_{\alpha=1}^{m_d} \sqrt{\lambda_\alpha^{(d)}} \xi_\alpha^{(d)} \phi_\alpha^{(d)}(x). \tag{1.33}$$

Second, the projected process $\kappa_{\mathcal{B}}$ is further reduced, through the resolution of the reduced problem, to yield the final representation $\hat{\kappa}_{\hat{N}}$.

Because $\kappa - \kappa_{\mathcal{B}}$ is orthogonal to $\kappa_{\mathcal{B}} - \hat{\kappa}_{\hat{N}}$, the squared norm of the error $\kappa - \hat{\kappa}_{\hat{N}}$ can be decomposed into two independent parts as follows [31]:

$$\mathbb{E}\left[\|\kappa - \hat{\kappa}_{\hat{N}}\|_\Omega^2\right] = \mathbb{E}\left[\|\kappa - \kappa_{\mathcal{B}}\|_\Omega^2\right] + \mathbb{E}\left[\|\kappa_{\mathcal{B}} - \hat{\kappa}_{\hat{N}}\|_\Omega^2\right]. \tag{1.34}$$

The first term is obtained by adding up the local contributions over the subdomains, which, by construction of the local modes, are given by

$$\epsilon_d^2 := \mathbb{E}\left[\|\kappa - \kappa_{\mathcal{B}}\|_{\Omega_d}^2\right] = \mathbb{E}\left[\|\kappa\|_{\Omega_d}^2\right] - \sum_{\alpha=1}^{m_d} \lambda_\alpha^{(d)} \quad , \quad d = 1, \ldots, n_d. \tag{1.35}$$

Then, gathering the local contributions leads up to

$$\epsilon_{\mathcal{B}}^2 := \mathbb{E}\left[\|\kappa - \kappa_{\mathcal{B}}\|_\Omega^2\right] = \sum_{d=1}^{n_d} \epsilon_d^2 = \mathbb{E}\left[\|\kappa\|_\Omega^2\right] - \sum_{d=1}^{n_d} \sum_{\alpha=1}^{m_d} \lambda_\alpha^{(d)}. \tag{1.36}$$

Similarly to the classical KL truncation error, the second error contribution is

$$\epsilon_{\mathcal{B}\hat{N}}^2 := \mathbb{E}\left[\|\kappa_{\mathcal{B}} - \hat{\kappa}_{\hat{N}}\|_\Omega^2\right] = \mathbb{E}\left[\|\kappa_{\mathcal{B}}\|_\Omega^2\right] - \sum_{\alpha=1}^{\hat{N}} \Lambda_\alpha. \tag{1.37}$$

Since $\mathbb{E}\left[\|\kappa_{\mathcal{B}}\|_\Omega^2\right] \leq \mathbb{E}\left[\|\kappa\|_\Omega^2\right]$, the overall error can be estimated from

$$\mathbb{E}\left[\|\kappa - \hat{\kappa}_{\hat{N}}\|_\Omega^2\right] = \epsilon_{\mathcal{B}}^2 + \epsilon_{\mathcal{B}\hat{N}}^2 \leq 2\mathbb{E}\left[\|\kappa\|_\Omega^2\right] - \sum_{d=1}^{n_d} \sum_{\alpha=1}^{m_d} \lambda_\alpha^{(d)} - \sum_{\alpha=1}^{\hat{N}} \Lambda_\alpha \tag{1.38}$$

This expression shows that to reduce the error, one needs to jointly increase the size of the local basis over all the subdomains and increase $\hat{N}$. This suggests the existence of an optimal set of values for m_d and $\hat{N}$. In order to achieve an overall relative error $0 \leq \delta \leq 1$, the following levels or error can be enforced. First, regarding the local errors, the m_d's can be selected so as to ensure the following for each d:

$$\epsilon_d^2 = \mathbb{E}\left[\|\kappa\|_{\Omega_d}^2\right] - \sum_{\alpha=1}^{m_d} \lambda_\alpha^{(d)} \leq \mathbb{E}\left[\|\kappa\|_{\Omega_d}^2\right] \frac{\delta^2}{2}, \tag{1.39}$$

so that $\epsilon_{\mathcal{B}}^2 \leq \mathbb{E}\left[\|\kappa\|_\Omega^2\right]\delta^2/2$. Then, $\hat{N}$ is selected so that Eq. (1.30) holds, which ensures that

$$\mathbb{E}\left[\|\kappa - \hat{\kappa}_{\hat{N}}\|_\Omega^2\right] \leq \delta^2 \mathbb{E}\left[\|\kappa\|_\Omega^2\right]. \tag{1.40}$$

1.2.3 Generalized eigen-solves

The computation of a KL or a DD-KL representation boils down to solving generalized eigenvalue problem(s), see Eq. (1.15) for the KL expansion and Eq. (1.19) for the DD-KL expansion. For high-dimensional approximation subspaces, only the dominant eigen-pairs of the generalized eigenvalue problem(s) are needed to capture the wanted energy. Note that both the left and right-had side of the generalized eigenvalue problem(s) are SPD, irrespective of whether the covariance model is stationary or not, even if the stochastic process is non-Gaussian—although, for the later, the distribution of the latent random variables is generally not known. For the KL expansion, we can then invoke the Cholesky factorization $\mathbf{L}\mathbf{L}^T$ of $\mathbf{M}$ and write

$$\mathbf{L}^{-1}\mathbf{K}\mathbf{L}^{-T}\mathbf{c} = \lambda\mathbf{c}. \tag{1.41}$$

Then, finding solutions to Eq. (1.41) can be done using a symmetric Lanzcos procedure, see Algo. 1.5.1, in which case we only need to be able to compute matrix-vector products with $\mathbf{L}^{-1}\mathbf{K}\mathbf{L}^{-T}$, which can be done by first solving an upper triangular system, then applying $\mathbf{K}$ and solving a lower triangular system. If on the other hand, the factorization of $\mathbf{M}$ is not known, one can still devise a Lanczos procedure to solve the standard eigenvalue problem

$$\mathbf{M}^{-1}\mathbf{K}\mathbf{c} = \lambda\mathbf{c} \tag{1.42}$$

such that one does not need to assemble $\mathbf{M}^{-1}\mathbf{K}$ at any point of the algorithm. The resulting Lanczos procedure is presented in Section 9.2.6 of [139]. In either case, note that the Lanczos procedure may not be used as is in order to find all the solution eigen-pairs. Indeed, due to the effect of floating point arithmetic, the computed vectors of the Lanczos procedure lose their assumed orthogonality, which results in spurious eigenvector approximations [121]. To circumvent this difficulty, one can resort to partial or selective orthogonalization [124]. Other possible alternatives are to implicitly [23] or explicitly restart [165] the Lanczos procedure and thereby achieve a lower memory footprint for the eigen-solve. Note that the explicitly restarted Lanczos procedure is presented in Algo. 8. An alternative to Lanczos procedures is LOBPCG [76, 77]. Note however that our numerical experiments show that using LOBPCG yields less efficient eigen-solves than implicitly restarted Lanczos procedures (see Arpack [90] for a readily deployable implementation of the implicitly restarted Lanczos procedure).

1.3 Sampling of random coefficient fields

In order to sample realizations of the random coefficient field in Eq. (1.2), we wish to sample the stochastic coordinates of the truncated KL representation of the coefficient. For Gaussian random fields, these coordinates are independent and identically distributed

(iid) Gaussian random variables. Here, we simulate Gaussian random variables using methods which are invariably based on uniform random number generators. Given a uniformly distributed random variable $U \sim \mathrm{U}(0,1)$, one can obtain a Gaussian random variable by inverse transformation, that is, by letting $X = F^{-1}(U)$ where F denotes the cumulative distribution function of the normal distribution. Note however that there is no closed form of F^{-1}, so that performing such a transformation requires some level of approximation. In the remaining of this Section, we provide a presentation of sampling strategies, namely, Monte Carlo methods, Quasi Monte Carlo methods and Markov chains Monte Carlo (MCMC). This presentation follows the lines of [81].

1.3.1 Monte Carlo (MC)

1.3.1.1 Random numbers

At the heart of any MC method is a random number generator: a procedure that produces an infinite stream

$$U_1, U_2, U_3, \ldots \overset{\text{iid}}{\sim} \text{Dist} \tag{1.43}$$

of random variables that are iid according to some probability distribution Dist. When this distribution is the uniform distribution on the interval (0,1) (that is, Dist = U(0,1)), the generator is said to be a uniform random number generator. Most computer languages already contain a built-in uniform random number generator. The user is typically requested only to input an initial number, called the seed, and upon invocation the random number generator produces a sequence of independent uniform random variables on the interval (0,1). In Julia, for example, this is provided by the `rand` function.

The concept of an infinite iid sequence of random variables is a mathematical abstraction that may be impossible to implement on a computer. The best one can hope to achieve in practice is to produce a sequence of "random" numbers with statistical properties that are indistinguishable from those of a true sequence of iid random variables. Although physical generation methods based on universal background radiation or quantum mechanics seem to offer a stable source of such true randomness, the vast majority of current random number generators are based on simple algorithms that can be easily implemented on a computer. Following, L'Ecuyer [87], such algorithms can be represented as a tuple $(\mathcal{S}, f, \mu, \mathcal{U}, g)$, where

- $\mathcal{S}$ is a finite set of states,
- f is a function from $\mathcal{S}$ to $\mathcal{S}$,
- μ is a probability distribution on $\mathcal{S}$,
- $\mathcal{U}$ is the output space; for a uniform random number generator $\mathcal{U}$ is the interval (0,1), and we will assume so from now on, unless otherwise specified,
- g is a function from $\mathcal{S}$ to $\mathcal{U}$.

A random number generator then has the structure given in Algo. 2. The algorihm produces a sequence $U_1, U_2, U_3, \ldots$ of pseudorandom numbers — we will refer to them

Algorithm 2 Generic random number generator

1: Initialize: Draw the seed S_0 from the distribution μ on $\mathcal{S}$. Set $t = 1$.
2: Transition: Set $S_t = f(S_{t-1})$.
3: Output: Set $U_t = g(S_t)$.
4: Repeat: Set $t = t + 1$ and return to Step 2.

simply as random numbers. Starting from a certain seed, the sequence of states (and hence of random numbers) must repeat itself, because the state space is finite. The smallest number of steps taken before entering a previously visited state is called the period length of the random number generator.

Properties of a Good Random Number Generator What constitutes a good random number generator depends on many factors. It is always advisable to have a variety of random number generators available, as different applications may require different properties of the random generator. Below are some desirable, or indeed essential, properties of a good uniform random number generator; see also [154].

1. *Pass statistical tests*: The ultimate goal is that the generator should produce a stream of uniform random numbers that is indistinguishable from a genuine uniform iid sequence. Although from a theoretical point of view this criterion is too imprecise and even unfeasible, from a practical point of view this means that the generator should pass a battery of simple statistical tests designed to detect deviations from uniformity and independence.

2. *Theoretical support*: A good generator should be based on sound mathematical principles, allowing for a rigorous analysis of essential properties of the generator.

3. *Reproducible*: An important property is that the stream of random numbers is reproducible without having to store the complete stream in memory. This is essential for testing and variance reduction techniques. Physical generation methods cannot be repeated unless the entire stream is recorded.

4. *Fast and efficient*: The generator should produce random numbers in a fast and efficient manner, and require little storage in computer memory. Many Monte Carlo techniques for optimization and estimation require billions or more random numbers. Current physical genertion methods are no match for simple algorithmic generators in terms of speed.

5. *Large period*: The period of a random number generator should be extremely large — on the order of 10^{50} — in order to avoid problems with duplication and dependence. Evidence exists [127] that in order to produce N random numbers, the period length needs to be at least $10N^2$. Most early algorithmic random number generators were fundamentally inadequate in this respect.

6. *Multiple streams*: In many applications it is necessary to run multiple independent random streams in parallel. A good random number generator should have easy provisions for multiple independent streams.

7. *Cheap and easy*: A good random number generator should be cheap and not require expensive external equipment. In addition, it should be easy to install, implement, and run. In general such a random number generator is also more easily portable over different computer platforms and architectures.

8. *Not produce 0 or 1*: A desirable property of a random number generator is that both 0 and 1 are exclude from the sequence of random numbers. This is to avoid division by 0 or other numerical complications.

From a theoretical point of view, a finite-state random number generator can always be distinguished from a true iid sequence, after observing the sequence longer than its period. However, from a practical point of view this may not be feasible within a "reasonable" amount of time. This idea can be formalized through the notion of computational complexity, see, for example, [108].

Choosing a Good Random Number Generator As Pierre L'Ecuyer puts it [88], choosing a good random generator is like choosing a new car: for some people or applications speed is preferred, while for others robustness and reliability are more important. For Monte Carlo simulation the distributional properties of random generators are paramount, whereas in coding and cryptography unpredictability is crucial.

Nevertheless, as with cars, there are many poorly designed and outdated models available that should be avoided. Indeed several of the standard generators that come with popular programming languages and computing packages can be appallingly poor [89].

Two classes of generators that have overall good performance are:

1. *Combined multiple recursive generators*, some of which have excellent statistical properties, are simple, have large period, support multiple streams, and are relatively fast. A popular choice is L'Ecuyer's `MRG32k3a`, which has been implemented as one of the core generators in Matlab, VSL, SAS, and the simulation packages SSJ, Arena, and Automod.

2. *Twisted general feedback shift register generators*, some of which have very good equidistributional properties, are among the fastest generators available (due to their essentially binary implementation), and can have extremely long periods. A popular choice is Matsumoto and Nishimura's Mersenne twister `MT19937ar`, which is currently the default generator in Matlab.

In general, a good uniform number generator has overall good performance, in terms of the criteria mentioned above, but is not usually the top performer over all these criteria. In choosing an appropriate generator, it pays to remember the following.

- Faster generators are not necessarily better (indeed, often the contrary is true).
- A small period is in general bad, but a larger period is not necessarily better.
- Good equidistribution is a necessary requirement for a good generator but not a sufficient requirement.

1.3.2 Quasi Monte Carlo (QMC)

Quasirandom numbers are akin to random numbers but exhibit much more regularity. This makes them well-suited for numerical evaluation of multidimensional integrals. This

section discusses the main types of quasirandom sequences, including Halton, Faure, Sobol', and Korobov sequences.

1.3.2.1 Multidimensional integration

Recall that the purpose of a uniform random number generator is to produce an unlimited stream of numbers $U_1, U_2, \ldots$ that behave statistically as independent and uniformly distributed random variables on $(0,1)$. From such a stream it is easy to construct an infinite sequence of independent and uniformly distributed random vectors (points) in $(0,1)^d$, by defining $\mathbf{U}_1 = (U_1, \ldots, U_d)$, $\mathbf{U}_2 = (U_{d+1}, \ldots, U_{2d})$, ... For any real-valued function h on $(0,1)^d$ these random vectors can then be used to approximate the d-dimensional integral

$$\ell = \int_{(0,1)^d} h(\mathbf{u}) \mathrm{d}\mathbf{u} \tag{1.44}$$

via the sample average

$$\widehat{\ell} = \frac{1}{N} \sum_{i=1}^{N} h(\mathbf{U}_i). \tag{1.45}$$

Precise error bounds on the approximation eeror can be found through simple statistical procedures. In particular, the standard error $[\mathbb{E}(\widehat{\ell} - \ell)^2]^{1/2}$ decreases at a rate $\mathcal{O}(N^{-1/2})$. Hence, asymptotically, to decrease the error by a factor 2, one needs 4 time as many samples. This convergence rate can often be improved by constructing quasirandom points $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N$ that fill the unit cube in a much more regular way than is achieved via iid random points. In general, the components of such points can be zero, so we assume from now on that quasirandom points lie in the unit cube $[0,1)^d$ rather than $(0,1)^d$. Quasi Monte Carlo methods are Monte Carlo methods in which the ordinary uniform random points are replaced by quasirandom points. Quasirandom points are no longer independent, but do have a high degree of uniformity, which is often expressed in terms of their discrepancy (first introduced by Roth [135]). Specifically, let $\mathcal{C}$ be a collection of subsets of $[0,1)^d$ and $\mathcal{P}_N = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ a set of points in $[0,1)^d$. The discrepancy of $\mathcal{P}_N$ relative to $\mathcal{C}$ is defined as

$$D_{\mathcal{C}}(\mathcal{P}_N) = \sup_{C \in \mathcal{C}} \left| \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{\{\mathbf{u}_i \in C\}} - \int \mathbf{1}_{\{\mathbf{u} \in C\}} \mathrm{d}\mathbf{u} \right|. \tag{1.46}$$

Special cases are the ordinary discrepancy, where $\mathcal{C}$ is the collection of rectangles $[a_1, b_1) \times \cdots \times [a_d, b_d)$, and the star discrepancy, where $\mathcal{C}$ is the collection of rectangles $[0, b_1) \times \cdots \times [0, b_d)$.

The sum in Eq. (1.46) is simply the number of points in C, whereas the integral is the d-dimensional volume of C. The integration error for all indicator functions $\mathbf{1}_{\{\mathbf{u} \in C\}}$, $C \in \mathcal{C}$ is thus bounded by the discrepancy of the point set. Similarly, the Koksma-Hlawka inequality provides, for suitable class of functions h; see page 19 of [108]. Discrepancy measures are therefore useful tools for studying convergence rates for multidimensional integration. Note that the star discrepancy may be viewed as the d-dimensional generalization of the Kolmogorov-Smirnov test statistic.

There are two main classes of low-discrepancy sequences: those based on van der Corput sequences, such as the Halton, Faure, and Sobol' point sets; and those based on lattice methods, such as the Koborov lattice rule.

1.3.2.2 Randomization and scrambling

One of the appealing features of ordinary Monte Carlo integration is that an assessment of the error in the sample average approximation (1.45) of the integral (1.44) is readily available in the form of standard errors and confidence intervals. For quasi Monte Carlo integration this is no longer the case, as the points, $\mathbf{u}_i$'s say, are deterministic and not $\cup[0,1)^d$ distributed.

However, the situation can be remedied by simply adding a fixed random vector $\mathbf{Z} \sim \cup[0,1)^d$ to each point and then taking the fractional part of the resulting point. It is easy to see that each point $\widetilde{\mathbf{U}}_i = (\mathbf{u}_i + \mathbf{Z}) \mod 1$ is $\cup[0,1)^d$ distributed. This procedure is called random shifting and was first proposed by Cranley and Patterson [33]. Using a random shift renders the quasi Monte Carlo approximation

$$\widetilde{\ell} = \frac{1}{N} \sum_{i=1}^{N} h(\widetilde{\mathbf{U}}_i) \tag{1.47}$$

a random variable with expectation ℓ in Eq. (1.44). By repeating the quasi Monte Carlo procedure independently with K different shift vectors one obtains K independent copies of $\widetilde{\ell}$, to which one can apply the standard statistical techniques for evaluating confidence intervals and standard error.

For digital sequences a random shift can also be applied directly to the digits. Specifically, suppose $\mathbf{a}_k = (a_{k1}, a_{k2}, \dots)^T$ is the infinite-dimensional vector that corresponds to the b-ary expansion of the k-th coordinate of a point $\mathbf{u}$; thus, the k-th coordinate of $\mathbf{u}$ is given by

$$u_k = \sum_{i=1}^{\infty} a_{ki} b^{-i}. \tag{1.48}$$

Let $\mathbf{W} = (W_1, W_2, \dots)^T$ be an infinite-dimensional random vector in which the W_i's are independent and discrete uniformly distributed on $\{0, 1, \dots, b-1\}$. In other words, $\mathbf{W}$ is the vector representing the b-ary expansion of a $\cup[0,1)$-distributed random number. Next, let $\mathbf{W}_1, \dots, \mathbf{W}_d$ be independent copies of $\mathbf{W}$. By adding $\mathbf{W}_k$ to $\mathbf{a}_k$ modulo b, for $k = 1, \dots, d$, one obtains vectors $\widetilde{\mathbf{a}}_1, \dots, \widetilde{\mathbf{a}}_d$ representing the b-ary expansion of $(\mathbf{u} + \mathbf{Z}) \mod 1$, where $\mathbf{Z} \sim \cup[0,1)^d$. By adding the same $\mathbf{W}_k$'s to all the points in the quasi Monte Carlo point set, this digital shift procedure yields a point set that has exactly the same distribution as one obtained using the original random shift method.

Digital sequences such as the Halton, Faure, and Sobol' sequences are sometimes "shuffled" with the aim of improving their uniformity and convergence properties. A general procedure, called nested permutation scrambling was introduced by Owen (see, for example, [115] and [116]) who permute the digits in the b_k-ary expansion for each component $k = 1, \dots, d$. This can be done in a deterministic or random way. A convenient subset of such procedures is obtained by premultiplying the digital vectors with a random lower-triangular matrix $\mathbf{L}_k$ with elements in $\{0, 1, \dots, b_k\}$, for each dimension $k = 1, \dots, d$.

There exist several variants of this procedure (see [101] and page 207 of [91]), but the most common approach is to choose the lower off-diagonal elements of $\mathbf{L}_k$ independently and uniformly from $\{0, 1, \ldots, b_k\}$ and the diagonal elements independently and uniformly from $\{1, \ldots, b_k\}$.

1.3.3 Markov chain Monte Carlo (MCMC)

Markov chain Monte Carlo (MCMC) is a generic method for approximate sampling from an arbitrary distribution. The main idea is to generate a Markov chain whose limiting distribution is equal to the desired distribution. In this Section, we describe some of the most prominent MCMC algorithms:

- The Metropolis-Hastings algorithm and in particular the independence sampler and random walk sampler;
- The Gibbs sampler, which is particularly uuseful in Bayesian analysis.

1.3.3.1 Metropolis-Hastings algorithm

The MCMC method originates from Metropolis et al. [102] and applies to the following setting. Suppose that we wish to generate samples from an arbitrary multidimensional probability density function (pdf)

$$f(\mathbf{x}) = \frac{p(\mathbf{x})}{\mathcal{Z}} \ , \quad \mathbf{x} \in \mathcal{X} \tag{1.49}$$

where $p(\mathbf{x})$ is a known positive function and $\mathcal{Z}$ is a known or unknown normalizing constant. Let $q(\mathbf{y}|\mathbf{x})$ be a proposal or instrumental density: a Markov transition density describing how to go from state $\mathbf{x}$ to $\mathbf{y}$. Similar to the acceptance-rejection method, the Metropolis-Hastings algorithm is based on the "trial-and-error" strategy presented in Algo. 3.

Algorithm 3 Metropolis-Hastings algorithm

1: Initialize with some $\mathbf{X}_0$ for which $f(\mathbf{X}_0) > 0$.
2: **for** $s = 0, \ldots, S-1$ **do**
3: Given the current state $\mathbf{X}_s$, generate $\mathbf{Y} \sim q(\mathbf{y}|\mathbf{X}_s)$.
4: Generate $U \sim U(0,1)$ and deliver

$$\mathbf{X}_{s+1} = \begin{cases} \mathbf{Y} & \text{if } U \leq \alpha(\mathbf{X}_s, \mathbf{Y}) \\ \mathbf{X}_s & \text{otherwise} \end{cases} \tag{1.50}$$

where

$$\alpha(\mathbf{x}, \mathbf{y}) = \min \left\{ \frac{f(\mathbf{y}) q(\mathbf{x}|\mathbf{y})}{f(\mathbf{x}) q(\mathbf{y}|\mathbf{x})}, 1 \right\} \tag{1.51}$$

5: **end for**

The probability $\alpha(\mathbf{x}, \mathbf{y})$ is called the acceptance probability. Note that in Eq. (1.51) we my replace f by p.

We thus obtain the so-called Metropolis-Hastings Markov chain, $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_S$ with $\mathbf{X}_S$ approximately distributed according to $f(\mathbf{x})$ for large S. A single Metropolis-Hastings iteration is equivalent to generating a point from the transition density $\kappa(\mathbf{x}_{s+1}|\mathbf{x}_s)$, where

$$\kappa(\mathbf{y}|\mathbf{x}) = \alpha(\mathbf{x}, \mathbf{y})q(\mathbf{y}|\mathbf{x}) + (1 - \alpha^*(\mathbf{x}))\delta_{\mathbf{x}}(\mathbf{y}), \tag{1.52}$$

with $\alpha^* = \int \alpha(\mathbf{x}, \mathbf{y})q(\mathbf{y}|\mathbf{x})\mathrm{d}\mathbf{y}$ and $\delta_{\mathbf{x}}(\mathbf{y})$ denoting the Dirac delta function. Since

$$f(\mathbf{x})\alpha(\mathbf{x}, \mathbf{y})q(\mathbf{y}|\mathbf{x}) = f(\mathbf{y})\alpha(\mathbf{y}, \mathbf{x})q(\mathbf{x}|\mathbf{y}) \tag{1.53}$$

and

$$(1 - \alpha^*(\mathbf{x}))\delta_{\mathbf{x}}(\mathbf{y})f(\mathbf{x}) = (1 - \alpha^*(\mathbf{y}))\delta_{\mathbf{y}}(\mathbf{x})f(\mathbf{y}) \tag{1.54}$$

the transition density satisfies the detailed balance equation:

$$f(\mathbf{x})\kappa(\mathbf{y}|\mathbf{x}) = f(\mathbf{y})\kappa(\mathbf{x}|\mathbf{y}), \tag{1.55}$$

from which it follows that f is the stationary pdf of the chain. In addition, if the transition density q satisfies the conditions

$$\mathrm{Prob}[\alpha(\mathbf{X}_s, \mathbf{Y}) < 1|\mathbf{X}_s] > 0, \tag{1.56}$$

that is, the event $\{\mathbf{X}_{s+1} = \mathbf{X}_s\}$ has positive probability, and

$$q(\mathbf{y}|\mathbf{x}) > 0 \; \forall \; \mathbf{x}, \mathbf{y} \in \mathcal{X}, \tag{1.57}$$

then f is the limiting pdf of the chain. As a consequence, to estimate an expectation $\mathbb{E}[H(\mathbf{X})]$ with $\mathbf{X} \sim f$, one can use the following ergodic estimator

$$\frac{1}{S+1}\sum_{s=0}^{T} H(\mathbf{X}_s). \tag{1.58}$$

The original Metropolis algorithm [102] is suggested for symmetric proposal functions; that is, for $q(\mathbf{y}|\mathbf{x}) = q(\mathbf{x}|\mathbf{y})$. Hastings [66] modified the original MCMC algorithm to allow non-symmetric proposal functions, hence the name Metropolis-Hastings algorithm.

Independence Sampler If the proposal function $q(\mathbf{y}|\mathbf{x})$ does not depend on $\mathbf{x}$, that is, $q(\mathbf{y}|\mathbf{x}) = g(\mathbf{y})$ for some pdf $g(\mathbf{y})$, then the acceptance probability is

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{\frac{f(\mathbf{y})g(\mathbf{x})}{f(\mathbf{x})g(\mathbf{y})}, 1\right\} \tag{1.59}$$

and Algo. 3 is referred to as the independence sampler. The independence sampler is very similar to the acceptance-rejection method. Just as in that method, it is important that the proposal density g is close to the target f. Note, however, that in contrast to the acceptance-rejection method the independence sampler produces dependent samples. In

addition, if there is a constant C such that

$$f(\mathbf{x}) = \frac{p(\mathbf{x})}{\int p(\mathbf{x})\mathrm{d}\mathbf{x}} \leq Cg(\mathbf{x}) \tag{1.60}$$

for all $\mathbf{x}$, then the acceptance rate in Eq. (1.50) is at least $1/C$ whenever the chain is in stationarity; namely,

$$\begin{aligned}
\mathrm{Prob}[U \leq \alpha(\mathbf{X}, \mathbf{Y})] &= \int\int \min\left\{\frac{f(\mathbf{y})g(\mathbf{x})}{f(\mathbf{x})g(\mathbf{y})}, 1\right\} f(\mathbf{x})g(\mathbf{y})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \\
&= 2\int\int \mathbf{1}\left\{\frac{f(\mathbf{y})g(\mathbf{x})}{f(\mathbf{x})g(\mathbf{y})} \geq 1\right\} f(\mathbf{x})g(\mathbf{y})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \\
&\geq \frac{2}{C}\int\int \mathbf{1}\left\{\frac{f(\mathbf{y})g(\mathbf{x})}{f(\mathbf{x})g(\mathbf{y})} \geq 1\right\} f(\mathbf{x})g(\mathbf{y})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \\
&\geq \frac{2}{C}\mathrm{Prob}\left[\frac{f(\mathbf{Y})}{g(\mathbf{Y})} \geq \frac{f(\mathbf{X})}{g(\mathbf{X})}\right] = \frac{1}{C}.
\end{aligned}$$

Random Walk Sampler If the proposal is symmetric, that is, $q(\mathbf{y}|\mathbf{x}) = q(\mathbf{x}|\mathbf{y})$, then the acceptance probability (1.51) is

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{\frac{f(\mathbf{y})}{f(\mathbf{x})}, 1\right\}, \tag{1.61}$$

and Algo. 3 is referred to as the random walk sampler. An example of a random walk sampler is when $\mathbf{Y} = \mathbf{X}_s + \vartheta\mathbf{Z}$ in step 3 of Algo. 3 where $\mathbf{Z}$ is typically generated from some spherically symmetrical distribution (in the continuous case), such as $\mathcal{N}(\mathbf{0}, \mathbf{I})$. In particular, letting $\vartheta = 2.38/\sqrt{n}$ where n denotes the size of the random vector was shown to maximize efficiency properties, see [130].

Consider the Langevin diffusion defined by the SDE

$$\mathrm{d}\mathbf{X}_s = \frac{1}{2}\nabla \ln f(\mathbf{X}_s)\mathrm{d}s + \mathrm{d}\mathbf{W}_s \tag{1.62}$$

where $\nabla \ln f(\mathbf{X}_s)$ denotes the gradient of $\ln f(\mathbf{x})$ evaluated at $\mathbf{X}_s$. The Langevin diffusion has stationary pdf f, and is nonexplosive and reversible. Suppose the proposal state $\mathbf{Y}$ in step X of Algo. 3 corresponds to the Euler discretization of the Langevin SDE for some step size h:

$$\mathbf{Y} = \mathbf{X}_s + \frac{h}{2}\nabla \ln f(\mathbf{X}_s) + \sqrt{h}\mathbf{Z}\ , \quad \mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{1.63}$$

This gives a more sophisticated random walk sampler with a "drift" term $\nabla \ln f(\mathbf{x}_s)$. Such random walk samplers are collectively known as Langevin Metropolis-Hastings algorithms [133]. Note that the gradient can be approximated numerically via finite differences and does not require knowledge of the normalizing constant of $f(\mathbf{x})$. In some cases the Langevin Metropolis-Hastings algorithms are more efficient than the simple random walk algorithms [131, 133]. For a discussion of the optimal tuning of Langevin Metropolis-Hastings algorithms see [106].

1.3.3.2 Gibbs sampler

The Gibbs sampler can be viewed as a particular instance of the Metropolis-Hastings algorithm for generating n-dimensional random vectors [49]. Due to its importance it is presented separately. The distinguishing feature of the Gibbs sampler is that the underlying Markov chain is constructed from a sequence of conditional distributions, in either a deterministic or random fashion.

Suppose that we wish to sample a random vector $\mathbf{X} = (X_1 \ldots, X_n)$ according to a target pdf $f(\mathbf{x})$. Let $f(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ represent the conditional pdf of the i-th component, X_i, given the other components $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. The Gibbs sampler is then given by Algo. 4.

Algorithm 4 Gibbs sampler

1: Initialize with some $\mathbf{X}_0$ for which $f(\mathbf{X}_0) > 0$.
2: **for** $t = 0, 1, \ldots$ **do**
3: Given the current state $\mathbf{X}_t$, generate $\mathbf{Y} = (Y_1, \ldots, Y_n)$ as follows:
4: (a) Draw Y_1 from the conditional pdf $f(x_1|X_{s,2}, \ldots, X_{s,n})$.
5: (b) Draw Y_i from $f(x_i|Y_1, \ldots, Y_{i-1}, X_{s,i+1}, \ldots, X_{s,n})$ for $i = 2, \ldots, n-1$.
6: (c) Draw Y_n from $f(x_n|Y_1, \ldots, Y_{n-1})$.
7: Let $\mathbf{X}_{s+1} = \mathbf{Y}$.
8: **end for**

The transition pdf is given by

$$\kappa_{1\to n}(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} f(y_i|y_1, \ldots, y_{i-1}, x_{i+1}, \ldots, x_n), \tag{1.64}$$

where the subscript $1 \to n$ indicates that the components of vector $\mathbf{x}$ are updated in the order $1 \to 2 \to 3 \to \cdots \to n$. Note that in the Gibbs sampler every "proposal" $\mathbf{y}$, is accepted. The transition density of the reverse move $\mathbf{y} \to \mathbf{x}$, in which the vector $\mathbf{y}$ is updated in the order $n \to n-1 \to n-2 \to \cdots \to 1$ is

$$\kappa_{n\to 1}(\mathbf{x}|\mathbf{y}) = \prod_{i=1}^{n} f(y_i|y_1, \ldots, y_{i-1}, x_{i+1}, \ldots, x_n). \tag{1.65}$$

Hammersley and Clifford [65] proved the following result.

Theorem 1. *Let $f(x_i)$ be the i-th marginal density of the pdf $f(\mathbf{x})$. Suppose that density $f(\mathbf{x})$ satisfies the positivity condition, that is, for every $\mathbf{y} \in \{\mathbf{x}|f(x_i) > 0,\ i = 1, \ldots, n\}$, we have $f(\mathbf{y}) > 0$. Then*

$$f(\mathbf{y})\kappa_{n\to 1}(\mathbf{x}|\mathbf{y}) = f(\mathbf{x})\kappa_{1\to n}(\mathbf{y}|\mathbf{x}). \tag{1.66}$$

Proof. Observe that

$$\begin{aligned}\frac{\kappa_{1\to n}(\mathbf{y}|\mathbf{x})}{\kappa_{n\to 1}(\mathbf{x}|\mathbf{y})} &= \prod_{i=1}^{n} \frac{f(y_i|y_1,\ldots,y_{i-1},x_{i+1},\ldots,x_n)}{f(x_i|y_1,\ldots,y_{i-1},x_{i+1},\ldots,x_n)} \\ &= \prod_{i=1}^{n} \frac{f(y_1,\ldots,y_i,x_{i+1},\ldots,x_n)}{f(y_1,\ldots,y_{i-1},x_i,\ldots,x_n)} \\ &= \frac{f(\mathbf{y})}{f(\mathbf{x})} \frac{\prod_{i=1}^{n-1} f(y_1,\ldots,y_i,x_{i+1},\ldots,x_n)}{\prod_{j=2}^{n} f(y_1,\ldots,y_{j-1},x_j,\ldots,x_n)} \\ &= \frac{f(\mathbf{y})}{f(\mathbf{x})} \frac{\prod_{i=1}^{n-1} f(y_1,\ldots,y_i,x_{i+1},\ldots,x_n)}{\prod_{j=1}^{n-1} f(y_1,\ldots,y_j,x_{j+1},\ldots,x_n)} = \frac{f(\mathbf{y})}{f(\mathbf{x})}.\end{aligned}$$

The result follows by rearranging the last identity. □

The Hammersley-Clifford condition is similar to the detailed balance condition for the Metropolis-Hastings sampler, because integrating both sides with respect to $\mathbf{x}$ yields the global balance equation:

$$\int f(\mathbf{x})\kappa_{1\to n}(\mathbf{y}|\mathbf{x})\mathrm{d}\mathbf{y} = f(\mathbf{y}), \tag{1.67}$$

from which we can conclude that f is the stationary pdf of the Markov chain with transition density $\kappa_{1\to n}(\mathbf{y}|\mathbf{x})$. In addition, it can be shown [129] that the positivity assumption on f implies that the Gibbs Markov chain is irreducible and that f is its limiting pdf. In practice the positivity condition is difficult to verify. However, there are a number of weaker and more technical conditions (see [94, 129]) which ensure that the limiting pdf of the process $\{\mathbf{X}_t, t = 1, 2, \ldots\}$ generated via the Gibbs sampler is f, and that the convergence to f is geometrically fast.

Algo. 4 presents a systematic (coordinatewise) Gibbs sampler. That is the components of vector $\mathbf{X}$ are updated in the coordinatewise order $1 \to 2 \to \cdots \to n$. The completion of all the conditional sampling steps in the specified order is called a cycle. Alternative updating of the components of vector $\mathbf{X}$ are possible. In the reversible Gibbs sampler a single cycle consists of the coordinatewise updating

$$1 \to 2 \to \cdots \to n-1 \to n \to n-1 \to \cdots \to 2 \to 1. \tag{1.68}$$

In the random sweep/scan Gibbs sampler a single cycle can either consist of one or several coordinates selected uniformly from the integers $1, \ldots, n$ or a random permutation $\pi_1 \to \pi_2 \to \cdots \to \pi_n$ of all coordinates. In all cases, except for the systematic Gibbs sampler, the resulting Markov chain $\{\mathbf{X}_t, t = 1, 2, \ldots\}$ is reversible. In the case where a cycle consists of a single randomly selected coordinate, the random Gibbs sampler can be formally viewed as a Metropolis-Hastings sampler with transition function

$$q(\mathbf{y}|\mathbf{x}) = \frac{1}{n} f(y_i|x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = \frac{1}{n}\frac{f(\mathbf{y})}{\sum_{y_i} f(\mathbf{y})}, \tag{1.69}$$

where $\mathbf{y} = (x_1,\ldots,x_{i-1},y_i,x_{i+1},\ldots,x_n)$. Since $\sum_{y_i} f(\mathbf{y})$ can also be written $\sum_{y_i} f(\mathbf{x})$,

we have

$$\frac{f(\mathbf{y})q(\mathbf{x}|\mathbf{y})}{f(\mathbf{x})q(\mathbf{y}|\mathbf{x})} = \frac{f(\mathbf{y})f(\mathbf{x})}{f(\mathbf{x})f(\mathbf{y})} = 1, \tag{1.70}$$

so that the acceptance probability $\alpha(\mathbf{x}, \mathbf{y})$ is 1 in this case.

1.4 Finite element method (FEM)

Upon specification of a realization of the random field $\kappa(x, \theta)$ for a given θ in Eq. (1.2), we are left to solve a deterministic problem of the form

$$\nabla \cdot [\kappa(x)\nabla u(x)] = -f(x) \; \forall \; x \in \Omega \tag{1.71}$$

$$u(x) = g(x) \; \forall \; x \in \partial\Omega \tag{1.72}$$

where the dependence on θ is disregarded and Ω is a bounded domain in the plane $\mathbb{R}^2 = \{x = (x, y) \mid x_i \in \mathbb{R}\}$ with boundary $\partial\Omega$. In what follows, we provide an overview of the finite element method as done in [68, 42].

1.4.1 Weak formulation

We shall now give a variational formulation of the problem given by Eqs. (1.71)–(1.72). First, we show that if u satisfies Eqs. (1.71)–(1.72), then u is the solution of the following variational problem: Find $u \in V$ such that

$$a(u, v) = b(v) \;\; \forall \; v \in V \tag{1.73}$$

where

$$a(u, v) = \int_\Omega \kappa(x)\nabla u(x) \cdot \nabla v(x) \mathrm{d}x, \tag{1.74}$$

$$b(v) = \int_\Omega f(x)v(x)\mathrm{d}x, \tag{1.75}$$

$$V = \left\{ v \;\middle|\; v \text{ is continuous on } \Omega, \; \frac{\partial v}{\partial x} \text{ and } \frac{\partial v}{\partial y} \text{ are piecewise continuous on } \Omega \text{ and } v = 0 \text{ on } \partial\Omega \right\}. \tag{1.76}$$

We see that $u \in V$ satisfies Eq. (1.73) if and only if u is the solution of the following minimization problem: Find $u \in V$ such that $F(u) \leq F(v) \; \forall \; v \in V$ where $F(v)$ is the total potential energy

$$F(v) = \frac{1}{2}a(v, v) - b(v). \tag{1.77}$$

To see that Eq. (1.73) follows from Eqs. (1.71)–(1.72), we multiply Eq. (1.71) with an arbitrary test function $v \in V$ and integrate over Ω. Using integration by parts followed

by the divergence theorem in $\mathbb{R}^2$, we have:

$$\begin{aligned}
\int_\Omega v(x)\nabla\cdot[\kappa(x)\nabla u(x)]\mathrm{d}x &= -\int_\Omega f(x)v(x)\mathrm{d}x \\
\int_\Omega \nabla\cdot[v(x)\kappa(x)\nabla u(x)]\mathrm{d}x - \int_\Omega \kappa(x)\nabla u(x)\cdot\nabla v(x)\mathrm{d}x &= -\int_\Omega f(x)v(x)\mathrm{d}x \\
\int_{\partial\Omega} v(x)\kappa(x)[\nabla u(x)]\cdot n(x)\mathrm{d}s - \int_\Omega \kappa(x)\nabla u(x)\cdot\nabla v(x)\mathrm{d}x &= -\int_\Omega f(x)v(x)\mathrm{d}x \\
\int_\Omega \kappa(x)\nabla u(x)\cdot\nabla v(x)\mathrm{d}x &= \int_\Omega f(x)v(x)\mathrm{d}x
\end{aligned}$$

where the boundary integral vanishes since $v = 0$ on $\partial\Omega$. On the other hand, if $u \in V$ satisfies Eq. (1.73) and u is sufficiently regular, then we see that u also satisfies Eqs. (1.71)–(1.72).

When giving variational formulations of boundary value problems for PDEs, it is from the mathematical point of view natural and useful to work with function spaces V that are slightly larger (i.e., that contain somewhat more functions) than the spaces of continuous functions with piecewise continuous derivatives used in the preceding formulation. It is also useful to endow the spaces V with various scalar products with the scalar product related to the boundary value problem. More precisely, V will be a Hilbert space.

Before introducing these Hilbert spaces, we recall a few simple concepts from linear algebra, namely, if V is a linear space, then we say that L is a linear form on V if $L : V \to \mathbb{R}$, i.e., $L(v) \in \mathbb{R}$ for $v \in V$, and L is linear, i.e., for all $v, w \in V$ and $\alpha, \beta \in \mathbb{R}$, we have:

$$L(\alpha v + \beta w) = \alpha L(v) + \beta L(w). \tag{1.78}$$

Furthermore, we say that $a(\cdot,\cdot)$ is a bilinear form on $V \times V$ if $a : V \times V \to \mathbb{R}$, i.e. $a(v, w) \in \mathbb{R}$ for $v, w \in V$, and a is linear in each argument, i.e., for all $u, v, w \in V$ and $\alpha, \beta \in \mathbb{R}$, we have

$$a(u, \alpha v + \beta w) = \alpha a(u, v) + \beta a(u, w), \tag{1.79}$$

$$a(\alpha u + \beta v, \beta w) = \alpha a(u, w) + \beta a(v, w). \tag{1.80}$$

The bilinear form $a(\cdot,\cdot)$ on $V \times V$ is said to be symmetric if

$$a(v, w) = a(w, v) \ \ \forall\, v, w \in V. \tag{1.81}$$

A symmetric bilinear form $a(\cdot,\cdot)$ on $V \times V$ is said to be a scalar product on V if

$$a(v, v) > 0 \ \ \forall\, v \in V,\ v \neq 0. \tag{1.82}$$

The norm $\|\cdot\|_a$ associated with a scalar product $a(\cdot,\cdot)$ is defined by

$$\|v\|_a = \sqrt{a(v, v)} \ \ \forall\, v \in V. \tag{1.83}$$

Further, if $\langle\cdot,\cdot\rangle$ is a scalar product with corresponding norm $\|\cdot\|$, then we have the

Cauchy's inequality

$$|\langle v, w\rangle| \leq \|v\|\|w\|. \tag{1.84}$$

We further recall that if V is a linear space with a scalar product with corresponding norm $\|\cdot\|$, then V is said to be a Hilbert space if V is complete, i.e., if every Cauchy sequence with respect to $\|\cdot\|$ is convergent. We recall that a sequence $v_1, v_2, v_3, \ldots$ of elements v_i in the space V with norm $\|\cdot\|$ is said to be a Cauchy sequence if for all $\varepsilon > 0$ there is a natural number N such thqt $\|v_i - v_j\| < \varepsilon$ if $i, j > N$. Moreover, the sequence converges to v if $\|v - v_i\| \to 0$ as $i \to \infty$.

Let us then define the space of square-integrable functions for the bounded domain Ω:

$$L^2(\Omega) = \left\{ v : \Omega \to \mathbb{R} \;\middle|\; \int_\Omega v(x)^2 \mathrm{d}x < \infty \right\} \tag{1.85}$$

with the following scalar product and norm:

$$(v, w) = \int_\Omega v(x)w(x)\mathrm{d}x, \quad \|v\|_{L^2(\Omega)} = \sqrt{\int_\Omega v(x)^2 \mathrm{d}x}. \tag{1.86}$$

Then, the Sobolev space $H^1(\Omega)$ given by

$$H^1(\Omega) = \left\{ v \in L^2(\Omega) \;\middle|\; \frac{\partial v}{\partial x_1}, \frac{\partial v}{\partial x_2} \in L^2(\Omega) \right\} \tag{1.87}$$

is the space where the weak solution of Eq. (1.73) naturally exists, and this space is also the natural home for the test functions v. We also introduce the corresponding scalar product and norm as follows:

$$(v, w)_{H^1(\Omega)} = \int_\Omega [v(x)w(x) + \nabla v(x) \cdot \nabla w(x)]\mathrm{d}x, \tag{1.88}$$

$$\|v\|_{H^1(\Omega)} = \sqrt{\int_\Omega [v(x) + |\nabla v(x)|^2]\mathrm{d}x}. \tag{1.89}$$

Then we define the solution and test spaces by

$$H^1_E(\Omega) = \{u \in H^1(\Omega) \mid u = g \text{ on } \partial\Omega\}, \tag{1.90}$$

$$H^1_{E_0}(\Omega) = \{v \in H^1(\Omega) \mid v = 0 \text{ on } \partial\Omega\} \tag{1.91}$$

and we equip these spaces with the same scalar product and norm as $H^1(\Omega)$.

The boundary value problem stated by Eqs. (1.71)–(1.72) can now be given the following variational formulation

$$(V) \qquad \text{Find } u \in H^1_E(\Omega) \text{ such that } a(u, v) = b(v) \;\; \forall\, v \in H^1_{E_0}(\Omega) \tag{1.92}$$

or equivalently

$$(M) \qquad \text{Find } u \in H^1_E(\Omega) \text{ such that } F(u) \leq F(v) \;\; \forall\, v \in H^1_{E_0}(\Omega) \tag{1.93}$$

where $F(v) = (1/2)a(v,v) - b(v)$. The formulation (V) is said to be a weak formulation of the boudary value problem and the solution of (V) is said to be a weak solution of the boundary value problem defined by Eqs. (1.71)–(1.72). If u is a weak solution the it is not immediately clear that u is also a classical solution of the boundary value problem, since this requires u to be sufficiently regular so that $\nabla \cdot [\kappa(x)\nabla u(x)]$ is defined in a classical sense. The advantage mathematically of the weak formulation (V) is that it is easy to prove the existence of a solution to (V), whereas it is relatively difficult to prove the existence of a classical solution of the boundary value problem defined by Eqs. (1.71)–(1.72). To prove the existence of a classical solution of the boundary value problem one usually starts with the weak solution and shows, often with considerable effort, that in fact this solution is sufficiently regular to be also a classical solution.

1.4.2 Galerkin FEM

We now develop the idea of approximating u by taking a finite-dimensional subspace of the solution space $H^1_E(\Omega)$. The starting point is the weak formulation (V) of the boundary value problem given by Eqs. (1.71)–(1.72). To construct an approximation method, we assume that $S^h_0 \subset H^1_{E_0}(\Omega)$ is a finite n-dimensional vector space of test functions of which $\{\phi_1, \phi_2, \ldots, \phi_n\}$ is a convenient basis. Then, in order to ensure that the Dirichlet boundary condition in Eq. (1.90) is satisfied, we extend this basis set by defining additional functions $\phi_{n+1}, \ldots, \phi_{n+n_\partial}$ and select fixed coefficients u_j, $j = n+1, \ldots, n+n_\partial$, so that the function $\sum_{j=n+1}^{n+n_\partial} u_j\phi_j(x)$ interpolates the boundary data g on $\partial\Omega$. The finite element approximation $u^h \in S^h_E$ is then uniquely associated with the vector $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$ or real coefficients in the expansion

$$u^h(x) = \sum_{j=1}^{n} u_j\phi_j(x) + \sum_{j=n+1}^{n+n_\partial} u_j\phi_j(x). \tag{1.94}$$

The functions ϕ_i, $i = 1, \ldots, n$ in the first sum in Eq. (1.94) define a set of trial functions also called shape functions in the context of finite elements.

The construction of the space S^h_E is achieved above by ensuring that the specific choice of trial functions in Eq. (1.94) coincides with the choice of test functions that form the basis for S^h_0, and is generally referred to as the Galerkin approximation method. A more general approach is to construct approximation spaces for Eqs. (1.90) and (1.91) using different trial and test functions. This alternative is called a Petrov-Galerkin approximation method.

The result of the Galerkin approximation is a finite-dimensional version of the weak formulation:

$$\text{Find } u^h \in S^h_E \text{ such that } a(u^h, v^h) = b(v^h) \;\; \forall\, v^h \in S^h_0. \tag{1.95}$$

For computations, it is convenient to enforce Eq. (1.95) for each basis function; then it follows from Eq. (1.94) that Eq. (1.95) is equivalent to finding u_j, $j = 1, \ldots, n$ such that

$$\sum_{j=1}^{n} u_j a(\phi_j, \phi_i) = b(\phi_i) - \sum_{j=n+1}^{n+n_\partial} u_j a(\phi_j, \phi_i) \tag{1.96}$$

for $i = 1, \ldots, n$. This can be written in matrix form as the linear system of equations

$$\mathbf{Au} = \mathbf{b} \tag{1.97}$$

where $\mathbf{A}$ has components

$$A_{ij} = a(\phi_j, \phi_i) \tag{1.98}$$

and $\mathbf{b}$ has components

$$b_i = b(\phi_i) - \sum_{j=n+1}^{n+n_\partial} u_j a(\phi_j, \phi_i) \tag{1.99}$$

The system of linear equations given by Eq. (1.97) is called the Galerkin system, and the function u_h computed by substituting the solution of Eq. (1.99) into Eq. (1.94) is the Galerkin solution.

The Galerkin coefficient matrix is clearly symmetric. In contrast, using different test and trial functions necessarily leads to a nonsymmetric system matrix. The matrix is also positive-definite. To see this, consider a general coefficient vector $\mathbf{v}$ corresponding to a specific function $v_h(x) = \sum_{j=1}^n v_j \phi_j(x) \in S_0^h$, so that

$$\begin{aligned}
\mathbf{v}^T\mathbf{Av} &= \sum_{j=1}^n \sum_{i=1}^n v_j A_{ji} v_i \\
&= \sum_{j=1}^n \sum_{i=1}^n v_j \left(\int_\Omega \nabla\phi_j(x) \cdot \nabla\phi_i(x) \mathrm{d}x \right) v_i \\
&= \int_\Omega \left(\sum_{j=1}^n v_j \nabla\phi_j(x) \right) \cdot \left(\sum_{i=1}^n v_i \nabla\phi_i(x) \right) \mathrm{d}x \\
&= \int_\Omega \nabla v^h(x) \cdot \nabla v^h(x) \mathrm{d}x \\
&\geq 0.
\end{aligned}$$

Thus we see that $\mathbf{A}$ is at least semi-definite. Definiteness follows from the fact that $\mathbf{vAv} = 0$ if and only if $\nabla v^h = 0$, that is, if and only if v^h is constant in Ω. Since $v^h \in S_0^h$, it is continuous up to the boundary and is zero on $\partial\Omega$, thus $\nabla v^h = 0$ implies $v^h = 0$. Finally, since the test functions are a basis for S_0^h we have that $v^h = 0$ implies $\mathbf{v} = \mathbf{0}$.

It is clear that the choices of S_E^h and S_0^h are central in that they determine whether or not u^h has any relation to the weak solution u. The inclusions $S_E^h \subset H_E^1(\Omega)$ and $S_0^h \subset H_{E_0}^1(\Omega)$ lead to conforming approximations; more general nonconforming approximation spaces containing specific discontinuous functions are also possible, but these are not considered here. The general desire is to choose S_E^h and S_0^h so that approximation to any required accuracy can be achieved if the dimension n is large enough. That is, it is required that the error $\|u - u^h\|$ reduces rapidly as n is increased, and moreover that the computational effort associated with solving Eq. (1.97) is acceptable — the choice of basis is critical in this respect.

The mathematical motivation for finite element approximation is the observation that

a smooth function can often be approximated to arbitrary accuracy using piecewise polynomials. Starting from the Galerkin system, the idea is to choose basis functions $\{\phi_j\}_{j=1}^{n+n_\partial}$ in Eq. (1.94) that are locally nonzero on a mesh of triangles or a grid of rectangles.

1.4.2.1 Triangular finite elements

For simplicity, we assume that $\Omega \subset \mathbb{R}^2$ is polygonal, so that we are able to tile (or tessellate) the domain with a set of triangles $\triangle_k$, $k = 1, \ldots, K$, defining a triangulation $\mathcal{T}_h$. This means that vertices of neighboring triangles coincide and that

- $\cup_k \overline{\triangle_k} = \overline{\Omega}$,
- $\triangle_k \cap \triangle_\ell = \emptyset$ for $k \neq \ell$.

The points where triangle vertices meet are called nodes. Surrounding any node is a patch of triangles that each have that node as a vertex. If we label the nodes $j = 1, \ldots, n$, then for each j, we define a basis function ϕ_j that is nonzero only on that patch. The simplest choice here (leading to a conforming approximation) is the $\mathbb{P}_1$ or piecewise linear basis function: ϕ_j is a linear function on triangle, which takes the value one at the node point j and zero at all other node points on the mesh. Notice that ϕ_j is clearly continuous on Ω. Moreover, although ϕ_j has discontinuities in slope across element boundaries, it is smooth enough that $\phi_j \in H^1(\Omega)$, and so it leads to conforming approximation space $S_0^h = \text{span}(\phi_1, \phi_2, \ldots, \phi_n)$ for use with Eq. (1.95).

In terms of approximation, the precise choice of basis for the space is not important; for practical application however, the availability of a locally defined basis such as this one is crucial. Having only three basis functions that are not identically zero on a given triangle means that the construction of the Galerkin matrix $\mathbf{A}$ in Eq. (1.97) is easily automated. Another point is that the Galerkin matrix has a well-defined sparse structure: $A_{ij} \neq 0$ only if the node points labeled i and j lie on the same edge of a triangular element. This is important for the development of efficient methods for solving the linear systems Eq. (1.97).

Summarizing, $\mathbb{P}_1$ approximation can be characterized by saying that the overall approximation is continuous, and that on any element with vertices i, j and k there are only the three basis functions ϕ_i, ϕ_j and ϕ_k that are not identically zero. Within an element, ϕ_i is a linear function that takes the value one at node i and zero at nodes j and k. This local characterization is convenient for implementation of the finite element method and it is also useful for the description of piecewise polynomial approximation spaces of higher degree.

Some practical considerations and implementation aspects of the Galerkin FEM method are presented in Appendix A.

1.5 Krylov subspace methods for SPD matrices

The main tasks of linear algebra in this work consist of solving standard and generalized eigenvalue problems as well as linear systems. All the matrices involved in these tasks are SPD and high-dimensional and we can efficiently compute matrix-vector products. The most suited algorithms to solve these problems are iterative projection based methods.

Consequently, for the eigenvalue problems, we consider the symmetric Lanczos procedure and its explicitly restarted variant. For the linear solves, we consider the preconditioned conjugate gradient algorithm.

1.5.1 Symmetric Lanczos procedures

The symmetric Lanczos algorithm is a particular case of Arnoldi's method that was introduced in 1951 [3] as a means to reduce dense matrices into Hessenberg form. In the case of the symmetric Lanczos algorithm, a symmetric dense matrix is reduced into tridiagonal form. The symmetric Lanczos algorithm consists of building an orthonormal basis of so called Lanczos vectors $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}$ for the Krylov subspace $\mathcal{K}^{(m)}(\mathbf{A}, \mathbf{v}^{(1)}) = \text{Span}\{\mathbf{v}^{(1)}, \mathbf{A}\mathbf{v}^{(1)}, \ldots, \mathbf{A}^{m-1}\mathbf{v}^{(1)}\}$. An essential algorithm for the understanding of the derivation of the Lanczos algorithm is the Gram-Schmidt orthogonalization algorithm. The Gram-Schmidt algorithm produces an orthonormal basis of vectors $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}$ for a subspace spanned by m given linearly independent vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$. The algorithm

Algorithm 5 MODIFIED GRAM-SCHMIDT($\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$)

Input: Linearly independent vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$
Output: Orthonormal basis $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}$ of Span$\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$
1: $r_{11} := \|\mathbf{x}^{(1)}\|_2$. If $r_{11} = 0$ Stop, else $\mathbf{v}^{(1)} := \mathbf{x}^{(1)}/r_{11}$
2: **for** $j = 2, \ldots, m$ **do**
3: $\quad \hat{\mathbf{v}} := \mathbf{x}^{(j)}$
4: $\quad$ **for** $i = 1, \ldots, j-1$ **do**
5: $\quad\quad r_{ij} := \hat{\mathbf{v}}^T \mathbf{v}^{(i)}$
6: $\quad\quad \hat{\mathbf{v}} := \hat{\mathbf{v}} - r_{ij}\mathbf{v}^{(i)}$
7: $\quad$ **end for**
8: $\quad r_{jj} := \|\hat{\mathbf{v}}\|_2$
9: $\quad$ If $r_{jj} = 0$ Stop, else $\mathbf{v}^{(j)} := \hat{\mathbf{v}}/r_{jj}$
10: **end for**

works as follows. First, normalize the vector $\mathbf{x}^{(1)}$, i.e., divide it by its 2-norm, to obtain the scaled vector $\mathbf{v}^{(1)}$ of unit norm. Then, $\mathbf{x}^{(2)}$ is orthogonalized against the vector $\mathbf{v}^{(1)}$ by substracting from $\mathbf{x}^{(2)}$ the projection of $\mathbf{x}^{(2)}$ onto $\mathbf{v}^{(1)}$, i.e. $\mathbf{x}^{(2)} \leftarrow \mathbf{x}^{(2)} - \mathbf{x}^{(2)T}\mathbf{v}^{(1)}\mathbf{v}^{(1)}$. The resulting vector is again normalized to yield the second vector $\mathbf{v}^{(2)}$. The i-th step of the Gram-Schmidt process consists of orthogonalizing the vector $\mathbf{x}^{(i)}$ against all previous vectors $\mathbf{v}^{(j)}$. It is possible to show that the Gram-Schmidt algorithm will break down if and only if the given vectors are not linearly independent. Algo. 5 is the modified Gram-Schmidt (MGS) algorithm which, when using exact arithmetic, is mathematically equivalent to the classical Gram-Schmidt (CGS) algorithm which we just described in words. While CGS is notorious for suffering from stability issues, MGS is known to offer better stability.

Starting with a given initial vector $\mathbf{v}^{(1)}$ with unit norm, the Lanczos procedure forms the Lanczos vectors computed by Gram-Schmidt orthogonalization where a new vector $\mathbf{A}\mathbf{v}^{(j)}$ is formed and orthogonalized against all the previously computed vectors $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(j)}$. That is, first, the Gram-Schmidt process is called to generate $\mathbf{v}^{(2)}$ from $\{\mathbf{v}^{(1)}, \mathbf{A}\mathbf{v}^{(1)}\}$, then the Gram-Schmidt process is called again to generate $\mathbf{v}^{(3)}$ from $\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{A}\mathbf{v}^{(2)}\}$

and so forth. Note that, by construction, we have $\mathbf{A}\mathbf{v}^{(j)} \in \text{Span}\{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(j+1)}\}$ and, for $k \geq j+2$, we have $\mathbf{v}^{(k)} \perp \text{Span}\{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(j+1)}\}$. As a result, we have $\mathbf{v}^{(k)T}\mathbf{A}\mathbf{v}^{(j)} = 0$ as long as $k \geq j+2$. Consequently, most of the terms of the series in line 6 of the Gram-Schmidt algorithm cancel. If we gather all the Lanczos vectors into a matrix $\mathbf{V}^{(m)} := [\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}]$, it hereby follows that $\mathbf{T}^{(m)} := \mathbf{V}^{(m)T}\mathbf{A}\mathbf{V}^{(m)}$ is tridiagonal. The resulting algorithm, provided in Algo. 6, is defined by LANCZOS($\mathbf{A}$, $\mathbf{v}^{(1)}$, m) and returns both the vectors $\{\mathbf{v}^{(j)}\}_{j=1}^{m}$, stored by columns in $\mathbf{V}^{(m)}$, and the tridiagonal matrix $\mathbf{T}^{(m)}$. The algorithm may break down in case the norm of $\mathbf{w}^{(j+1)}$ vanishes at a certain step j. In this case, the vector $\mathbf{v}^{(j+1)}$ cannot be computed and the algorithm stops. The computed $\alpha^{(j)}$'s and $\beta^{(j)}$'s in Algo. 6 are the components of the tridiagonal form of $\mathbf{A}$:

$$\mathbf{T}^{(m)} = \begin{bmatrix} \alpha^{(1)} & \beta^{(1)} & & & & \\ \beta^{(1)} & \alpha^{(2)} & \beta^{(2)} & & & \\ & \beta^{(2)} & \alpha^{(3)} & \ddots & & \\ & & \ddots & \ddots & \beta^{(m-2)} & \\ & & & \beta^{(m-2)} & \alpha^{(m-1)} & \beta^{(m-1)} \\ & & & & \beta^{(m-1)} & \alpha^{(m)} \end{bmatrix}. \tag{1.100}$$

After m iterations, the vectors $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}$ of a Lanczos procedure satisfy the Lanczos recurrence relation

$$\mathbf{A}\mathbf{V}^{(m)} = \mathbf{V}^{(m)}\mathbf{T}^{(m)} + \beta^{(m)}\mathbf{v}^{(m+1)}\mathbf{e}_m^T \tag{1.101}$$

where $\beta^{(m)} = \mathbf{v}^{(m)T}\mathbf{A}\mathbf{v}^{(m+1)}$. Since we intend to approximate the most dominant eigen-

Algorithm 6 LANCZOS($\mathbf{A}$, $\mathbf{v}^{(1)}$, m)

Input: SPD $\mathbf{A}$, $\mathbf{v}^{(1)} \in \mathcal{R}(\mathbf{A}), \|\mathbf{v}^{(1)}\|_2 = 1, m > 1$
Output: Orthonormal basis $\mathbf{V}^{(m)} := [\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}]$ of $\mathcal{K}^{(m)}(\mathbf{A}, \mathbf{v}^{(1)})$, and tridiagonal form $\mathbf{T}^{(m)}$

1: $\beta^{(1)} := 0, \mathbf{v}^{(0)} := \mathbf{0}, \mathbf{p} := \mathbf{A}\mathbf{v}^{(1)}, \alpha^{(1)} := \mathbf{v}^{(1)T}\mathbf{p}, \mathbf{V}^{(1)} := [\mathbf{v}^{(1)}], \mathbf{T}^{(1)} := [\alpha^{(1)}]$
2: **for** $j = 1, 2, \ldots, m-1$ **do**
3: $\quad \mathbf{w}^{(j+1)} := \mathbf{p} - \alpha^{(j)}\mathbf{v}^{(j)} - \beta^{(j-1)}\mathbf{v}^{(j-1)}$
4: $\quad \beta^{(j)} := \|\mathbf{w}^{(j+1)}\|_2$. If $\beta^{(j)} = 0$ then Stop
5: $\quad \mathbf{v}^{(j+1)} := \mathbf{w}^{(j+1)}/\beta^{(j)}$
6: $\quad \mathbf{p} := \mathbf{A}\mathbf{v}^{(j+1)}$
7: $\quad \alpha^{(j+1)} := \mathbf{v}^{(j+1)T}\mathbf{p}$
8: $\quad \mathbf{V}^{(j+1)} := [\mathbf{V}^{(j)}, \mathbf{v}^{(j+1)}]$
9: $\quad \mathbf{T}^{(j+1)} := \begin{bmatrix} \mathbf{T}^{(j)} & \beta^{(j)}\mathbf{e}_j \\ \beta^{(j)}\mathbf{e}_j^T & \alpha^{(j+1)} \end{bmatrix}$
10: **end for**

vectors of $\mathbf{A}$, we then use a Rayleigh-Ritz projection onto the Krylov subspace spanned by some Lanczos vectors.

Once, equipped with m Lanczos vectors $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}$ which span $\mathcal{K}^{(m)}(\mathbf{A}, \mathbf{v}^{(1)})$, we can formulate approximations $(\lambda, \mathbf{y})$ of the eigen-pairs of $\mathbf{A}$ using a Rayleigh-Ritz procedure

of the form

$$\mathbf{y} \in \mathcal{K}^{(m)}(\mathbf{A}, \mathbf{v}^{(1)}) \tag{1.102}$$

$$\mathbf{A}\mathbf{y} - \lambda\mathbf{y} \perp \mathcal{K}^{(m)}(\mathbf{A}, \mathbf{v}^{(1)}) \tag{1.103}$$

where $\text{Range}(\mathbf{V}^{(m)}) = \mathcal{K}^{(m)}(\mathbf{A}, \mathbf{v}^{(1)})$ with $\mathbf{V}^{(m)} := [\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}]$ implies that an approximate eigenvector of the form $\mathbf{y} := \mathbf{V}^{(m)}\hat{\mathbf{y}}$ can be obtained upon solving the tridiagonal eigenvalue problem

$$\mathbf{T}^{(m)}\hat{\mathbf{y}} = \lambda\hat{\mathbf{y}} \tag{1.104}$$

with $\mathbf{T}^{(m)} = \mathbf{V}^{(m)T}\mathbf{A}\mathbf{V}^{(m)}$. Solving a tridiagonal eigenvalue problem like Eq. (1.104) can be done using specialized algorithms with lower complexity than general-purpose algorithms. Note that since we wish to approximate the *nev* most dominant eigen-pairs of $\mathbf{A}$, the retained pairs $\{(\lambda^{(i)}, \hat{\mathbf{y}}^{(i)})\}_{i=1}^{nev}$ are the most dominant eigen-pairs of $\mathbf{T}^{(j)}$. The resulting algorithm, given in Algo. 7, is defined by RR-Lan($\mathbf{A}$, $\mathbf{v}^{(1)}$, m, nev). For all

Algorithm 7 RR-Lan($\mathbf{A}$, $\mathbf{v}^{(1)}$, m, nev)

Input: SPD $\mathbf{A}$,$\mathbf{v}^{(1)} \in \text{Range}(\mathbf{A})$, $\|\mathbf{v}^{(1)}\|_2 = 1, m > 1, 1 \leq nev \leq m$
Output: Approximations of the *nev* most dominant eigen-pairs $\{\lambda^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{nev}$ of $\mathbf{A}$

1: $\mathbf{V}^{(m)}, \mathbf{T}^{(m)} \leftarrow$Lanczos($\mathbf{A}$, $\mathbf{v}^{(1)}$, m)
2: Solve for the *nev* most dominant eigen-pairs $\{(\lambda^{(i)}, \hat{\mathbf{y}}^{(i)})\}_{i=1}^{nev}$ of $\mathbf{T}^{(m)}\hat{\mathbf{y}} = \lambda\hat{\mathbf{y}}$
3: **for** $i = 1, 2, \ldots, nev$ **do**
4: $\quad \mathbf{y}^{(i)} := \mathbf{V}^{(m)}\hat{\mathbf{y}}^{(i)}$
5: **end for**

the Rayleigh-Ritz pairs $(\lambda, \mathbf{y})$ of $\mathbf{A}$ in $\mathcal{K}^{(m)}(\mathbf{A}, \mathbf{v}^{(1)})$, the following expression lies for the corresponding eigen-residual:

$$\tilde{\mathbf{r}}(\lambda, \mathbf{y}) := \mathbf{A}\mathbf{y} - \lambda\mathbf{y} = \mathbf{A}\mathbf{V}^{(m)}\hat{\mathbf{y}} - \lambda\mathbf{V}^{(m)}\hat{\mathbf{y}} = \beta^{(m)}(\mathbf{e}_m^T\hat{\mathbf{y}})\mathbf{v}^{(m+1)} \tag{1.105}$$

where $\mathbf{y} = \mathbf{V}^{(m)}\hat{\mathbf{y}}$ and $(\lambda, \hat{\mathbf{y}})$ is an eigen-pair of $\mathbf{T}^{(m)}$. That is, it is conveniently possible to evaluate the eigen-residual $\tilde{\mathbf{r}}(\lambda, \mathbf{y}) := \mathbf{A}\mathbf{y} - \lambda\mathbf{y}$ of an approximate eigen-pair $(\lambda, \mathbf{y})$ of $\mathbf{A}$ without having to compute the matrix-vector product $\mathbf{A}\mathbf{y}$.

For large matrices, high-dimensional search spaces (i.e., large values of m) are often needed in order for the Rayleigh-Ritz vectors to precisely approximate several eigenvectors of $\mathbf{A}$. When the search space is generated by a Lanczos procedure, this requires storing a large basis so as to compute the Rayleigh-Ritz vectors using the expression $\mathbf{y} = \mathbf{V}^{(m)}\hat{\mathbf{y}}$, after solving Eq. (1.104). Until then, the computed vectors can be stored on disk. However, due to the effect of floating-point arithmetic, the computed vectors tend to lose their orthogonality as the dimension of the spanned Krylov subspace increases. This phenomenon, explained in [121, 120], requires some re-orthogonalization to prevent collateral effects on the Rayleigh-Ritz vectors. Different strategies exist to re-orthogonalize: full re-orthogonalization, which is the most computationally demanding; to which, partial [146], and selective [125] schemes were proposed as alternatives. Irrespective of the strategy selected, the cost of re-orthogonalization remains an issue for high-dimensional problems, often requiring Lanczos vectors to be stored on core for a faster execution,

hence limiting the range of suitable applications for un-restarted algorithms. Restarting strategies allow for a better use of resources than un-restarted algorithms [139]. In what follows, we present the explicitly restarted strategy.

1.5.1.1 Thick-Restart algorithm

Here, we consider a strategy in which the search space is explicitly restarted with several eigenvector approximations. While this approach was deemed the term *thick-restart* (TR) by Wu and Simon [165], it was not always the case when similar ideas were developed [104, 150]. We define TR methods as follows. First, the Lanczos vectors of a Krylov subspace are progressively generated to form search spaces of increasing dimension, each of which can be used to compute approximate eigenvectors. Once the dimension of the search space reaches a certain size $k+\ell$, k eigenvector approximations $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}$ are used as a basis of a new, *restarted* subspace. From hereon, a new vector $\hat{\mathbf{v}}^{(1)} \perp \text{Span}\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}\}$ is used to generate new Lanczos vectors $\hat{\mathbf{v}}^{(2)}, \ldots, \hat{\mathbf{v}}^{(j)}$ by letting each $\hat{\mathbf{v}}^{(j)}$ be $\mathbf{A}\hat{\mathbf{v}}^{(j-1)}$ orthogonalized against $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}, \hat{\mathbf{v}}^{(1)}, \ldots, \hat{\mathbf{v}}^{(j-1)}$. Once ℓ new Lanczos vectors have been generated, k new eigenvector approximations are computed in the current search space $\mathcal{S}_{\text{TR}-\text{Lan}}(\{\mathbf{y}^{(i)}\}_{i=1}^{k}, \hat{\mathbf{v}}^{(1)}, j)$ defined by

$$\mathcal{S}_{\text{TR}-\text{Lan}}(\{\mathbf{y}^{(i)}\}_{i=1}^{k}, \hat{\mathbf{v}}^{(1)}, j) := \text{Span}\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}, \hat{\mathbf{v}}^{(1)}, \mathbf{A}\hat{\mathbf{v}}^{(1)} \ldots, \mathbf{A}^{j-1}\hat{\mathbf{v}}^{(1)}\} \tag{1.106}$$

for $1 \leq j \leq \ell$. The search space is restarted again with these approximations and some $\hat{\mathbf{v}}^{(1)}$, yet to be defined. This process is repeated over and over again, and the overall number of Lanczos vectors generated is denoted by m. For all $m > k + \ell$, the current search space can be put in the form of Eq. (1.106). Note that, as the orthogonalization of $\hat{\mathbf{v}}^{(2)}, \ldots, \hat{\mathbf{v}}^{(j)}$ is done by a Gram-Schmidt procedure, it can reduce to short recurrent relations depending on how $\hat{\mathbf{v}}^{(1)}$ is defined for the given projection technique. Moreover, the choice of $\hat{\mathbf{v}}^{(1)}$ also strongly influences some properties of the current search space which are responsible for the convergence of the restarted procedure. Since we intend to approximate the most dominant eigenvectors of $\mathbf{A}$, its convenes to use a Rayleigh-Ritz projection. Note that, alternatively, we could use a harmonic Ritz projection if we intended to approximate the least dominant eigen-pairs of $\mathbf{A}$, see [159] for a harmonic Ritz TR algorithm.

The Rayleigh-Ritz TR Lanczos procedure, presented by Simon and Wu [165], generates Rayleigh-Ritz approximations $(\lambda^{(1)}, \mathbf{y}^{(1)}), \ldots, (\lambda^{(k)}, \mathbf{y}^{(k)})$ of $\mathbf{A}$ in the current search space $\mathcal{S}_{\text{TR}-\text{Lan}}(\{\mathbf{y}^{(i)}\}_{i=1}^{k}, \hat{\mathbf{v}}^{(1)}, m)$ of a TR Lanczos procedure. Just before the search space is restarted for the first time, by property of Rayleigh-Ritz pairs of $\mathbf{A}$ in Krylov subspaces, we have

$$\mathbf{A}\mathbf{y}^{(i)} = \lambda^{(i)}\mathbf{y}^{(i)} + \beta^{(k+\ell)}(\mathbf{e}_{k+\ell}^{T}\hat{\mathbf{y}}^{(i)})\mathbf{v}^{(k+\ell+1)} \tag{1.107}$$

where $\mathbf{y}^{(i)} := [\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(k+\ell)}]\hat{\mathbf{y}}^{(i)}$ for $1 \leq i \leq k$. Then, letting $\hat{\mathbf{v}}^{(1)} := \hat{\mathbf{v}}^{(k+\ell+1)}$ has important consequences on the procedure. First, the computation of $\hat{\mathbf{v}}^{(2)}, \ldots, \hat{\mathbf{v}}^{(j)}$ reduces

to the evaluation of short recurrence relations, see [165]. We then have

$$\mathbf{A}\hat{\mathbf{v}}^{(1)} = \hat{\alpha}^{(1)}\hat{\mathbf{v}}^{(1)} + \hat{\beta}^{(1)}\hat{\mathbf{v}}^{(2)} + \sum_{i=1}^{k} \beta^{(k+\ell)}(\mathbf{e}_{k+\ell}^T\hat{\mathbf{y}}^{(i)})\mathbf{y}^{(i)} \tag{1.108}$$

$$\mathbf{A}\hat{\mathbf{v}}^{(j)} = \hat{\alpha}^{(j)}\hat{\mathbf{v}}^{(j)} + \hat{\beta}^{(j)}\hat{\mathbf{v}}^{(j+1)} + \hat{\beta}^{(j-1)}\hat{\mathbf{v}}^{(j-1)} \text{ for } j > 1, \tag{1.109}$$

where $\hat{\alpha}^{(j)} := \hat{\mathbf{v}}^{(j)T}\mathbf{A}\hat{\mathbf{v}}^{(j)}$ and $\hat{\beta}^{(j)} := \hat{\mathbf{v}}^{(j+1)T}\mathbf{A}\hat{\mathbf{v}}^{(j)}$, so that $\mathbf{A}[\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}, \hat{\mathbf{v}}^{(1)}, \ldots, \hat{\mathbf{v}}^{(j)}]$ can be put in the form of a rank-1 update. In particular, for $j > 1$, we can write $\mathbf{V} := [\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}, \hat{\mathbf{v}}^{(1)}, \ldots, \hat{\mathbf{v}}^{(j)}]$ and

$$\mathbf{AV} - \mathbf{VT} = \mathbf{R} \tag{1.110}$$

where

$$\mathbf{T} = \begin{bmatrix} \lambda^{(1)} & & & \beta^{(k+\ell)}\mathbf{e}_{k+\ell}^T\hat{\mathbf{y}}^{(1)} & & & \\ & \ddots & & \vdots & & & \\ & & \lambda^{(k)} & \beta^{(k+\ell)}\mathbf{e}_{k+\ell}^T\hat{\mathbf{y}}^{(k)} & & & \\ \beta^{(k+\ell)}\mathbf{e}_{k+\ell}^T\hat{\mathbf{y}}^{(1)} & \cdots & \beta^{(k+\ell)}\mathbf{e}_{k+\ell}^T\hat{\mathbf{y}}^{(k)} & \hat{\alpha}^{(1)} & \hat{\beta}^{(1)} & & \\ & & & \hat{\beta}^{(1)} & \ddots & \ddots & \\ & & & & \ddots & \ddots & \hat{\beta}^{(j-1)} \\ & & & & & \hat{\beta}^{(j-1)} & \hat{\alpha}^{(j)} \end{bmatrix} \tag{1.111}$$

and

$$\mathbf{R} = [\beta^{(k+\ell)}(\mathbf{e}_{k+\ell}^T\hat{\mathbf{y}}^{(1)})\mathbf{v}^{(k+\ell+1)}, \ldots, \beta^{(k+\ell)}(\mathbf{e}_{k+\ell}^T\hat{\mathbf{y}}^{(k)})\mathbf{v}^{(k+\ell+1)}, \mathbf{0}, \ldots, \mathbf{0}] \tag{1.112}$$

has rank-1 so that, in light of Theorem 2, the current search space $\mathcal{S}_{\text{TR-Lan}}$ is, and remains a Krylov subspace as long as $\hat{\mathbf{v}}^{(1)} := \hat{\mathbf{v}}^{(\ell+1)}$ for all the next restarts. The resulting algorithm is given in Algo. 8 and defined by RR-TR-LAN($\mathbf{A}$, $\mathbf{v}^{(1)}$, m, k, ℓ, nev) where, for the sake of simplicity, the created vectors are all denoted by $\mathbf{v}^{(j)}$ for all j from 1 to m.

Theorem 2. *$\mathcal{K}^{(m)}(\mathbf{A}, \mathbf{v}^{(1)}) = \text{Span}\{\mathbf{v}^{(1)} \ldots, \mathbf{v}^{(m)}\}$ is a Krylov subspace if and only if there exists a m-by-m matrix $\mathbf{N}$ such that*

$$\mathbf{R} := \mathbf{AV}^{(m)} - \mathbf{V}^{(m)}\mathbf{N} \ , \ \mathbf{V}^{(m)} = [\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}] \tag{1.113}$$

has rank one and $\text{Span}\{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}, \text{Range}(\mathbf{R})\}$ has dimension $m + 1$.

Proof. See Section 11.7 in [2]. □

Then, once passed the first restart, all the eigenvector approximations are such that

$$\mathbf{A}\mathbf{y}^{(i)} = \lambda^{(i)}\mathbf{y}^{(i)} + \hat{\beta}^{(\ell)}(\mathbf{e}_{k+\ell}^T\hat{\mathbf{y}}^{(i)})\hat{\mathbf{v}}^{(\ell+1)}. \tag{1.114}$$

Algorithm 8 RR-TR-LAN($\mathbf{A}$, $\mathbf{v}_1$, m, k, ℓ, nev)

Input: SPD $\mathbf{A}$, $\mathbf{v}_1 \in \text{Range}(\mathbf{A})$, $\|\mathbf{v}_1\|_2 = 1$, $m > k + \ell$, $nev \leq k$
Output: Approximations of the nev most dominant eigen-pairs $\{\lambda_i, \mathbf{y}_i\}_{i=1}^{nev}$ of $\mathbf{A}$

$\beta_0 := 0, \mathbf{v}_0 := \mathbf{0}, \mathbf{p} := \mathbf{A}\mathbf{v}_1, \alpha_1 := \mathbf{v}_1^T\mathbf{p}, \mathbf{V} := [\mathbf{v}_1], \mathbf{T} := [\alpha_1]$
for $j = 1, 2, \ldots, m-1$ **do**
  $i := j\%(k+\ell)$
  if $i = 0$ **then**
    Solve for the k most dominant eigen-pairs $\{(\lambda, \hat{\mathbf{y}})\}_{i=1}^{k}$ such that $\mathbf{T}\hat{\mathbf{y}} = \lambda\hat{\mathbf{y}}$
    $\mathbf{V} := \mathbf{V}[\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_k]$
    $\mathbf{T} := \text{diag}(\lambda_1, \ldots, \lambda_k)$
    $\mathbf{s} := [\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_k]^T\mathbf{e}_{k+\ell}$
    $\mathbf{w}_{j+1} := \mathbf{p} - \alpha_j\mathbf{v}_j - \beta_{j-1}\mathbf{v}_{j-1}$
    $\beta_j := \|\mathbf{w}_{j+1}\|_2$. If $\beta_j = 0$ then Stop
    $\mathbf{v}_{j+1} := \mathbf{w}_{j+1}/\beta_j$
    $\mathbf{p} := \mathbf{A}\mathbf{v}_{j+1}$
    $\alpha_{j+1} := \mathbf{v}_{j+1}^T\mathbf{p}$
  else if $i = 1$ **and** $j > k + \ell$ **then**
    $\mathbf{V} := [\mathbf{V}, \mathbf{v}_j]$
    $\mathbf{T} := \begin{bmatrix} \mathbf{T} & \beta_{j-1}\mathbf{s} \\ \beta_{j-1}\mathbf{s}^T & \alpha_j \end{bmatrix}$
    $\mathbf{w}_{j+1} := \mathbf{p} - \alpha_j\mathbf{v}_j - \sum_{i=1}^{k} \beta_{j-1}(\mathbf{e}_{j-1}^T\hat{\mathbf{y}}_i)\mathbf{y}_i$
    $\beta_j := \|\mathbf{w}_{j+1}\|_2$. If $\beta_j = 0$ then Stop
    $\mathbf{v}_{j+1} := \mathbf{w}_{j+1}/\beta_j$
    $\mathbf{p} := \mathbf{A}\mathbf{v}_{j+1}$
    $\alpha_{j+1} := \mathbf{v}_{j+1}^T\mathbf{p}$
    $\mathbf{V} := [\mathbf{V}, \mathbf{v}_{j+1}]$
    $\mathbf{T} := \begin{bmatrix} \mathbf{T} & \beta_j\mathbf{e}_i \\ \beta_j\mathbf{e}_i^T & \alpha_{j+1} \end{bmatrix}$
  else
    $\mathbf{w}_{j+1} := \mathbf{p} - \alpha_j\mathbf{v}_j - \beta_{j-1}\mathbf{v}_{j-1}$
    $\beta_j := \|\mathbf{w}_{j+1}\|_2$. If $\beta_j = 0$ then Stop
    $\hat{\mathbf{v}}_{j+1} := \mathbf{w}_{j+1}/\beta_i$
    $\mathbf{p} := \mathbf{A}\mathbf{v}_{j+1}$
    $\alpha_{j+1} := \mathbf{v}_{j+1}^T\mathbf{p}$
    $\mathbf{V} := [\mathbf{V}, \mathbf{v}_{j+1}]$
    $\mathbf{T} := \begin{bmatrix} \mathbf{T} & \beta_j\mathbf{e}_i \\ \beta_j\mathbf{e}_i^T & \alpha_{j+1} \end{bmatrix}$
  end if
end for
Solve for the nev most dominant eigen-pairs $\{(\lambda_i, \hat{\mathbf{y}}_i)\}_{i=1}^{nev}$ of $\mathbf{T}\hat{\mathbf{y}} = \lambda\hat{\mathbf{y}}$
for $i = 1, 2, \ldots, nev$ **do**
  $\mathbf{y}_i := \mathbf{V}\hat{\mathbf{y}}_i$
end for

An important observation is that Eqs. (1.106), (1.107) and (1.114) imply

$$\mathcal{S}_{\text{TR}-\text{Lan}}(\{\mathbf{y}^{(r)}\}_{r=1}^{k}, \hat{\mathbf{v}^{(1)}}, j) = \text{Span}\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}, \mathbf{A}\mathbf{y}^{(i)}, \ldots, \mathbf{A}^{j}\mathbf{y}^{(i)}\} \text{ for } 1 \leq i \leq k. \tag{1.115}$$

This, in turn, implies that the current $(k+j)$-dimensional search space $\mathcal{S}_{\text{TR}-\text{Lan}}$ contains all the $(j+1)$-dimensional Krylov subspaces of $\mathbf{A}$ generated by each of the eigenvector approximations $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)}$:

$$\mathcal{K}^{(j+1)}(\mathbf{A}, \mathbf{y}^{(i)}) \subset \mathcal{S}_{\text{TR}-\text{Lan}}(\{\mathbf{y}^{(r)}\}_{r=1}^{k}, \hat{\mathbf{v}}^{(1)}, j) \text{ for } 1 \leq i \leq k. \tag{1.116}$$

This property plays an important role in explaining the effectiveness, and lack thereof, of restarted methods [105].

1.5.2 Conjugate gradient algorithms

The conjugate gradient (CG) is one of the best known iterative techniques for solving SPD linear systems. The CG algorithm is mathematically equivalent to the full orthogonalization method (FOM), see Section 6.4 of [138]. As such, this method seeks an approximate solution $\mathbf{u}^{(m)}$ from the affine subspace $\mathbf{u}^{(0)} + \mathcal{K}^{(m)}(\mathbf{A}, \mathbf{r}^{(0)})$ with $\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)}$, by imposing the Galerkin condition

$$\mathbf{r}^{(m)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(m)} \perp \mathcal{K}^{(m)}(\mathbf{A}, \mathbf{r}^{(0)}). \tag{1.117}$$

However, because $\mathbf{A}$ is symmetric, some simplifications resulting from the three-term Lanczos recurrence lead to a more elegant algorithm. The resulting algorithm, of which the derivation can be found in Section 6.7.1 of [138], is presented in Algo. 9. Note that if we set $\mathbf{v}^{(1)} := \mathbf{r}^{(0)}/\|\mathbf{r}^{(0)}\|_2$ in the Lanczos procedure and we set $\beta = \|\mathbf{r}^{(0)}\|_2$, then $\mathbf{V}^{(m)T}\mathbf{A}\mathbf{V}^{(m)} = \mathbf{T}^{(m)}$ and

$$\mathbf{V}^{(m)T}\mathbf{r}^{(0)} = \mathbf{V}^{(m)}(\beta\mathbf{v}^{(1)}) = \beta\mathbf{e}_1. \tag{1.118}$$

As a result, the approximate solution using the above m-dimensional subspaces is given by

$$\mathbf{u}^{(m)} = \mathbf{u}^{(0)} + \mathbf{V}^{(m)}\mathbf{y}^{(m)} \tag{1.119}$$

$$\mathbf{y}^{(m)} = \mathbf{T}^{(m)-1}(\beta\mathbf{e}_1). \tag{1.120}$$

Many of the results from the Lanczos procedure for linear systems are still valid. For example, the residual vector of the approximate solution $\mathbf{u}^{(m)}$ is such that

$$\mathbf{b} - \mathbf{A}\mathbf{u}^{(m)} = -\beta^{(m+1)}\mathbf{e}_m^T\mathbf{y}^{(m)}\mathbf{v}^{(m+1)}. \tag{1.121}$$

Since the CG algorithm is used only with positive definite matrices, the coefficients $\alpha^{(j)}$ and $\beta^{(j)}$ are always defined, and it can be shown that the $\mathbf{A}$-norm of the error $\mathbf{e}^{(j)} := \mathbf{u} - \mathbf{u}^{(j)}$ is actually minimized over the affine space $\mathbf{e}^{(0)} + \text{span}\{\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(j)}\}$.

Theorem 3. *Assume that* $\mathbf{A}$ *is SPD with dimension* n*. The CG algorithm generates the exact solution to the linear system* $\mathbf{A}\mathbf{u} = \mathbf{b}$ *in at most* n *steps. The error, residual,*

Algorithm 9 CG($\mathbf{A}$, $\mathbf{b}$, $\mathbf{u}^{(0)}$)

$\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)}$
$\mathbf{p}^{(0)} := \mathbf{r}^{(0)}$
for $j = 0, 1, \ldots, m-1$ **do**
  $\alpha^{(j)} := \mathbf{r}^{(j)T}\mathbf{r}^{(j)} / \mathbf{p}^{(j)T}\mathbf{A}\mathbf{p}^{(j)}$
  $\mathbf{u}^{(j+1)} := \mathbf{u}^{(j)} + \alpha^{(j)}\mathbf{p}^{(j)}$
  $\mathbf{r}^{(j+1)} := \mathbf{r}^{(j)} - \alpha^{(j)}\mathbf{A}\mathbf{p}^{(j)}$
  $\beta^{(j)} := \mathbf{r}^{(j+1)T}\mathbf{r}^{(j+1)} / \mathbf{r}^{(j)T}\mathbf{r}^{(j)}$
  $\mathbf{p}^{(j+1)} := \mathbf{r}^{(j+1)} + \beta^{(j)}\mathbf{p}^{(j)}$
end for

and direction vectors generated before the exact solution is obtained are well defined and satisfy

$$\mathbf{e}^{(j+1)^T}\mathbf{A}\mathbf{p}^{(k)} = \mathbf{p}^{(j+1)^T}\mathbf{A}\mathbf{p}^{(k)} = \mathbf{r}^{(j+1)^T}\mathbf{r}^{(k)} = 0 \ \ \forall\, k \leq j. \tag{1.122}$$

It follows that of all vectors in the affine space

$$\mathbf{e}^{(0)} + \operatorname{span}\{\mathbf{A}\mathbf{e}^{(0)}, \mathbf{A}^2\mathbf{e}^{(0)}, \ldots, \mathbf{A}^{j+1}\mathbf{e}^{(0)}\}, \tag{1.123}$$

$\mathbf{e}^{(j+1)}$ *has the smallest* $\mathbf{A}$*-norm.*

Proof. Since $\mathbf{A}$ is SPD, it is clear that the coefficients in the CG algorithm are well defined unless a residual vector is zero, in which case the exact solution has been found. Assume that $\mathbf{r}^{(0)}, \ldots, \mathbf{r}^{(j)}$ are non-zero. By the choice of $\alpha^{(0)}$, it is clear that $\mathbf{r}^{(1)^T}\mathbf{r}^{(0)} = \mathbf{e}^{(1)^T}\mathbf{A}\mathbf{p}^{(0)} = 0$, and from the choice of $\beta^{(0)}$ it follows that

$$\begin{aligned}\mathbf{p}^{(1)^T}\mathbf{A}\mathbf{p}^{(0)} &= \mathbf{r}^{(1)^T}\mathbf{A}\mathbf{p}^{(0)} + \frac{\mathbf{r}^{(1)^T}\mathbf{r}^{(1)}}{\mathbf{r}^{(0)^T}\mathbf{r}^{(0)}}\mathbf{p}^{(0)^T}\mathbf{A}\mathbf{p}^{(0)} \\ &= \mathbf{r}^{(1)^T}\frac{\left(\mathbf{r}^{(0)} - \mathbf{r}^{(1)}\right)}{\alpha^{(0)}} + \frac{\mathbf{r}^{(1)^T}\mathbf{r}^{(1)}}{\alpha^{(0)}} = 0,\end{aligned}$$

where the last equality holds because $\mathbf{r}^{(1)^T}\mathbf{r}^{(0)} = 0$ and $1/\alpha^{(0)}$ is real. Assume that

$$\mathbf{e}^{(j)^T}\mathbf{A}\mathbf{p}^{(k)} = {}^T\mathbf{A}\mathbf{p}^{(j)} = {}^T\mathbf{r}^{(k)} = 0 \ \ \forall\, k \leq j-1.$$

Then we also have

$$\begin{aligned}\mathbf{p}^{(j)^T}\mathbf{A}\mathbf{p}^{(j)} &= (\mathbf{r}^{(j)} + \beta^{(j-1)}\mathbf{p}^{(j-1)})^T\mathbf{A}\mathbf{p}^{(j)} = \mathbf{r}^{(j)^T}\mathbf{A}\mathbf{p}^{(j)}, \\ \mathbf{r}^{(j)^T}\mathbf{p}^{(j)} &= \mathbf{r}^{(j)^T}(\mathbf{r}^{(j)} + \beta^{(j-1)}\mathbf{p}^{(j-1)}) = \mathbf{r}^{(j)^T}\mathbf{r}^{(j)},\end{aligned}$$

so, by the choice of $\alpha^{(j)}$, it follows that

$$\begin{aligned}\mathbf{r}^{(j+1)^T}\mathbf{r}^{(j)} &= \mathbf{r}^{(j)^T}\mathbf{r}^{(j)} - \alpha^{(j)}\mathbf{r}^{(j)^T}\mathbf{A}\mathbf{p}^{(j)} = \mathbf{r}^{(j)^T}\mathbf{r}^{(j)} - \mathbf{r}^{(j)^T}\mathbf{r}^{(j)} = 0, \\ \mathbf{e}^{(j+1)^T}\mathbf{A}\mathbf{p}^{(j)} &= \mathbf{r}^{(j+1)^T}\mathbf{p}^{(j)} = \mathbf{r}^{(j)^T}\mathbf{p}^{(j)} - \alpha^{(j)}\mathbf{p}^{(j)^T}\mathbf{A}\mathbf{p}^{(j)} = \mathbf{r}^{(j)^T}\mathbf{r}^{(j)} - \mathbf{r}^{(j)^T}\mathbf{r}^{(j)} = 0.\end{aligned}$$

From the choice of $\beta^{(j)}$, we have

$$\mathbf{p}^{(j+1)^T}\mathbf{A}\mathbf{p}^{(j)} = \mathbf{r}^{(j+1)}\mathbf{A}\mathbf{p}^{(j)} + \frac{\mathbf{r}^{(j+1)^T}\mathbf{r}^{(j+1)}}{\mathbf{r}^{(j)^T}\mathbf{r}^{(j)}}\mathbf{p}^{(j)^T}\mathbf{A}\mathbf{p}^{(j)} = \mathbf{r}^{(j+1)^T}\frac{(\mathbf{r}^{(j)} - \mathbf{r}^{(j+1)})}{\alpha^{(j)}} + \frac{\mathbf{r}^{(j+1)^T}\mathbf{r}^{(j+1)}}{\alpha^{(j)}} = 0.$$

For $k \leq j-1$, we have

$$\begin{aligned}
\mathbf{e}^{(j+1)^T}\mathbf{A}\mathbf{p}^{(k)} &= (\mathbf{e}^{(j)} - \alpha^{(j)}\mathbf{p}^{(j)})^T\mathbf{A}\mathbf{p}^{(k)} = 0, \\
\mathbf{r}^{(j+1)^T}\mathbf{r}^{(k)} &= (\mathbf{r}^{(j)} - \alpha^{(j)}\mathbf{A}\mathbf{p}^{(j)})^T\mathbf{r}^{(k)} = -\alpha^{(j)}\mathbf{p}^{(j)^T}\mathbf{A}(\mathbf{p}^{(k)} - \beta^{(k-1)}\mathbf{p}^{(k-1)}) = 0, \\
\mathbf{p}^{(j+1)^T}\mathbf{A}\mathbf{p}^{(k)} &= (\mathbf{r}^{(j+1)} + \beta^{(j)}\mathbf{p}^{(j)})^T\mathbf{A}\mathbf{p}^{(k)} = \mathbf{r}^{(j+1)^T}\frac{(\mathbf{r}^{(k)} - \mathbf{r}^{(k+1)})}{\alpha^{(k)}} = 0,
\end{aligned}$$

so, by induction, the desired equalities are established. It is easily checked by induction that $\mathbf{e}^{(j+1)}$ lies in the space (1.123) and that $\text{span}\{\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(j)}\} = \text{span}\{\mathbf{A}\mathbf{e}^{(0)}, \ldots, \mathbf{A}^{j+1}\mathbf{e}^{(0)}\}$. Since $\mathbf{e}^{(j+1)}$ is $\mathbf{A}$-orthogonal to $\text{span}\{\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(j)}\}$, it follows that $\mathbf{e}^{(j+1)}$ is the vector in the space (1.123) with minimal $\mathbf{A}$-norm, and it also follows that if the exact solution is not obtained before step n, then $\mathbf{e}^{(n)} = 0$. □

Let us now assume that an SPD preconditioner $\mathbf{M}$ is available. The effect of the preconditioner on the algorithm is the following. Then one can precondition the system in one of several ways. When $\mathbf{M}$ is available in the form of an incomplete Cholesky factorization, i.e., when $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, then a simple way to preserve symmetry is to use split preconditioning, which yields the SPD matrix

$$\mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}\mathbf{x} = \mathbf{L}^{-1}\mathbf{b} \quad , \quad \mathbf{u} = \mathbf{L}^{-T}\mathbf{x}. \tag{1.124}$$

However it is not necessary to split the preconditioner in this manner in order to preserve symmetry. Observe that $\mathbf{M}^{-1}\mathbf{A}$ is self-adjoint for the $\mathbf{M}$-inner product $\mathbf{u}^T\mathbf{M}\mathbf{v}$, since

$$(\mathbf{M}^{-1}\mathbf{A}\mathbf{u})^T\mathbf{M}\mathbf{v} = (\mathbf{A}\mathbf{u})^T\mathbf{y} = \mathbf{u}^T\mathbf{A}\mathbf{v} = \mathbf{u}^T\mathbf{M}\mathbf{M}^{-1}\mathbf{A}\mathbf{v} = (\mathbf{M}^{-1}\mathbf{A}\mathbf{u})^T\mathbf{M}\mathbf{v}. \tag{1.125}$$

Therefore, an alternative is to replace the usual Euclidean inner product in the CG algorithm with the $\mathbf{M}$-inner product.

If the CG algorithm is rewritten for this new inner product, $\mathbf{r}^{(j)} = \mathbf{b} - \mathbf{A}\mathbf{u}^{(j)}$ the original residual and by $\mathbf{z}^{(j)} = \mathbf{M}^{-1}\mathbf{r}^{(j)}$the residual for the preconditioned system, the following sequence of operations is obtained, ignoring the initial step

$$\begin{aligned}
&1.\ \alpha^{(j)} := \mathbf{z}^{(j)T}\mathbf{M}\mathbf{z}^{(j)}/(\mathbf{M}^{-1}\mathbf{A}\mathbf{p}^{(j)})^T\mathbf{M}\mathbf{p}^{(j)}, && (1.126)\\
&2.;\ \mathbf{u}^{(j+1)} := \mathbf{u}^{(j)} + \alpha^{(j)}\mathbf{p}^{(j)}, && (1.127)\\
&3.\ \mathbf{r}^{(j+1)} := \mathbf{r}^{(j)} - \alpha^{(j)}\mathbf{A}\mathbf{p}^{(j)} \ \text{ and } \ \mathbf{z}^{(j+1)} := \mathbf{M}^{-1}\mathbf{r}^{(j+1)}, , && (1.128)\\
&4.\ \beta^{(j)} := \mathbf{z}^{(j+1)T}\mathbf{M}\mathbf{z}^{(j+1)}/\mathbf{z}^{(j)T}\mathbf{M}\mathbf{z}^{(j)}, && (1.129)\\
&5.\ \mathbf{p}^{(j+1)} := \mathbf{z}^{(j+1)} + \beta^{(j)}\mathbf{p}^{(j)}. && (1.130)
\end{aligned}$$

Since $\mathbf{z}^{(j)T}\mathbf{M}\mathbf{z}^{(j)} = \mathbf{r}^{(j)T}\mathbf{z}^{(j)}$ and $(\mathbf{M}^{-1}\mathbf{A}\mathbf{p}^{(j)})^T\mathbf{M}\mathbf{p}^{(j)} = (\mathbf{A}\mathbf{p}^{(j)})^T\mathbf{p}^{(j)}$, the $\mathbf{M}$-inner products do not have to be computed explicitly. With this observation, Algo. 10 is obtained.

Algorithm 10 PCG($\mathbf{A}$, $\mathbf{M}$, $\mathbf{b}$, $\mathbf{u}^{(0)}$)

1: $\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)}$
2: $\mathbf{z}^{(0)} := \mathbf{M}^{-1}\mathbf{r}^{(0)}$
3: $\mathbf{p}^{(0)} := \mathbf{z}^{(0)}$
4: **for** $j = 0, 1, \ldots, m-1$ **do**
5: $\quad \alpha^{(j)} := \mathbf{r}^{(j)T}\mathbf{r}^{(j)} / \mathbf{p}^{(j)T}\mathbf{A}\mathbf{p}^{(j)}$
6: $\quad \mathbf{u}^{(j+1)} := \mathbf{u}^{(j)} + \alpha^{(j)}\mathbf{p}^{(j)}$
7: $\quad \mathbf{r}^{(j+1)} := \mathbf{r}^{(j)} - \alpha^{(j)}\mathbf{A}\mathbf{p}^{(j)}$
8: $\quad \mathbf{z}^{(j+1)} := \mathbf{M}^{-1}\mathbf{r}^{(j+1)}$
9: $\quad \beta^{(j)} := \mathbf{r}^{(j+1)T}\mathbf{z}^{(j+1)} / \mathbf{r}^{(j)T}\mathbf{z}^{(j)}$
10: $\quad \mathbf{p}^{(j+1)} := \mathbf{z}^{(j+1)} + \beta^{(j)}\mathbf{p}^{(j)}$
11: **end for**

1.5.2.1 Induced approximations of eigenvectors

Sometimes, it is useful to be able to obtain the tridiagonal matrix $\mathbf{T}^{(m)}$ related to the underlying Lanczos iteration from the coefficients of the CG Algo. 9. This tridiagonal matrix can provide valuable eigenvalue information on the matrix $\mathbf{A}$. For example, the largest and smallest eigenvalues of the tridiagonal matrix can approximate the smallest and largest eigenvalues of $\mathbf{A}$. This could be used to compute an estimate of the condition number of $\mathbf{A}$, which in turn can help provide estimates of the error norm from the residual norm. We seek expressions for the coefficients $(\mathbf{T}^{(m)})_{j+1,j+1}$ and $(\mathbf{T}^{(m)})_{j,j+1}$ in terms of the coefficients $\alpha^{(j)}$ and $\beta^{(j)}$ obtained from the CG algorithm. The key information regarding the correspondence between the two pairs of coefficients resides in the corrrespondences between the vectors generated by the two algorithms. From Eq. (1.121) it is known that $\mathbf{r}^{(j)} \propto \mathbf{v}^{(j+1)}$. As a result, the diagonal components of the tridiagonal matrix $\mathbf{T}^{(m)}$ are given as follows,

$$(\mathbf{T}^{(m)})_{j+1,j+1} = \frac{\mathbf{v}^{(j+1)T}\mathbf{A}\mathbf{v}^{(j+1)}}{\mathbf{v}^{(j+1)T}\mathbf{v}^{(j+1)}} = \frac{\mathbf{r}^{(j)T}\mathbf{A}\mathbf{r}^{(j)}}{\mathbf{r}^{(j)T}\mathbf{r}^{(j)}}. \tag{1.131}$$

The denominator $\mathbf{r}^{(j)T}\mathbf{r}^{(j)}$ is readily available from the coefficients of the CG algorithm, but the numerator $\mathbf{r}^{(j)T}\mathbf{A}\mathbf{r}^{(j)}$ is not. The line 8 of Algo. 9 can be used to obtain

$$\mathbf{r}^{(j)} = \mathbf{p}^{(j)} - \beta^{(j-1)}\mathbf{p}^{(j-1)} \tag{1.132}$$

which is then substituted in $\mathbf{r}^{(j)T}\mathbf{A}\mathbf{r}^{(j)}$ to get

$$\mathbf{r}^{(j)T}\mathbf{A}\mathbf{r}^{(j)} = (\mathbf{p}^{(j)} - \beta^{(j-1)}\mathbf{p}^{(j-1)})^T\mathbf{A}(\mathbf{p}^{(j)} - \beta^{(j-1)}\mathbf{p}^{(j-1)}). \tag{1.133}$$

Note that the term $\beta^{(j-1)}\mathbf{p}^{(j-1)}$ is defined to be zero when $j = 0$. Because the $\mathbf{p}$-vectors are $\mathbf{A}$-orthogonal, we have

$$\mathbf{r}^{(j)T}\mathbf{A}\mathbf{r}^{(j)} = \mathbf{p}^{(j)T}\mathbf{A}\mathbf{p}^{(j)} + \beta^{(j-1)2}\mathbf{p}^{(j-1)T}\mathbf{A}\mathbf{p}^{(j-1)}, \tag{1.134}$$

from which we obtain the following for $j > 0$,

$$(\mathbf{T}^{(m)})_{j+1,j+1} = \frac{\mathbf{p}^{(j)T}\mathbf{A}\mathbf{p}^{(j)}}{\mathbf{r}^{(j)T}\mathbf{r}^{(j)}} + \beta^{(j-1)2}\frac{\mathbf{p}^{(j-1)T}\mathbf{A}\mathbf{p}^{(j-1)}}{\mathbf{r}^{(j)T}\mathbf{r}^{(j)}} = \frac{1}{\alpha^{(j)}} + \frac{\beta^{(j-1)}}{\alpha^{(j-1)}}. \tag{1.135}$$

The above expression is only valid for $j > 0$. For $j = 0$, the second term on the right-hand side should be omitted. Therefore, the diagonal components of the tridiagonal matrix $\mathbf{T}_m$ are given by

$$(\mathbf{T}^{(m)})_{j+1,j+1} = \begin{cases} \frac{1}{\alpha^{(j)}} & \text{for } j = 0, \\ \frac{1}{\alpha^{(j)}} + \frac{\beta^{(j-1)}}{\alpha^{(j-1)}} & \text{for } j > 0. \end{cases} \tag{1.136}$$

Now, an expression is needed for the codiagonal elements $(\mathbf{T}_m)_{j,j+1}$. From the definitions in the Lanczos algorithm, we have

$$(\mathbf{T}^{(m)})_{j,j+1} = \mathbf{v}^{(j)T}\mathbf{A}\mathbf{v}^{(j+1)} = \frac{|\mathbf{r}^{(j-1)T}\mathbf{A}\mathbf{r}^{(j)}|}{\|\mathbf{r}^{(j-1)}\|_2\|\mathbf{r}^{(j)}\|_2}. \tag{1.137}$$

From line 6 of Algo. 9, we have

$$\mathbf{A}\mathbf{p}^{(j)} = -\frac{1}{\alpha^{(j)}}(\mathbf{r}^{(j+1)} - \mathbf{r}^{(j)}). \tag{1.138}$$

Using Eqs. (1.132) and (1.138) as well as orthogonality properties of the CG algorithm, we obtain the following sequence of equalities:

$$\begin{aligned} \mathbf{r}^{(j-1)T}\mathbf{A}\mathbf{r}^{(j)} &= (\mathbf{p}^{(j-1)} - \beta^{(j-2)}\mathbf{p}^{(j-2)})\mathbf{A}\mathbf{r}^{(j)} && (1.139)\\ &= \mathbf{p}^{(j-1)T}\mathbf{A}\mathbf{r}^{(j)} - \beta^{(j-2)}\mathbf{p}^{(j-2)T}\mathbf{A}\mathbf{r}^{(j)} && (1.140)\\ &= \frac{-1}{\alpha^{(j-1)}}(\mathbf{r}^{(j)} - \mathbf{r}^{(j-1)})^T\mathbf{r}^{(j)} + \frac{\beta^{(j-2)}}{\alpha^{(j-2)}}(\mathbf{r}^{(j-1)} - \mathbf{r}^{(j-2)})^T\mathbf{r}^{(j)} && (1.141)\\ &= \frac{-1}{\alpha^{(j-1)}}\mathbf{r}^{(j)T}\mathbf{r}^{(j)}. && (1.142) \end{aligned}$$

Therefore,

$$(\mathbf{T}^{(m)})_{j,j+1} = \frac{1}{\alpha^{(j-1)}}\frac{\mathbf{r}^{(j)T}\mathbf{r}^{(j)}}{\|\mathbf{r}^{(j-1)}\|_2\|\mathbf{r}^{(j)}\|_2} = \frac{1}{\alpha^{(j-1)}}\frac{\|\mathbf{r}^{(j)}\|_2}{\|\mathbf{r}^{(j-1)}\|_2} = \frac{\sqrt{\beta^{(j-1)}}}{\alpha^{(j-1)}}. \tag{1.143}$$

This finally gives the general form of the m-dimensional Lanczos tridiagonal matrix in terms of the CG coefficients:

$$\mathbf{T}^{(m)} = \begin{bmatrix} \frac{1}{\alpha^{(0)}} & \frac{\sqrt{\beta^{(0)}}}{\alpha^{(0)}} & & & \\ \frac{\sqrt{\beta^{(0)}}}{\alpha^{(0)}} & \frac{1}{\alpha^{(1)}} + \frac{\beta^{(0)}}{\alpha^{(0)}} & \frac{\sqrt{\beta^{(1)}}}{\alpha^{(1)}} & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \frac{\sqrt{\beta^{(m-2)}}}{\alpha^{(m-2)}} \\ & & & \frac{\sqrt{\beta^{(m-2)}}}{\alpha^{(m-2)}} & \frac{1}{\alpha^{(m-1)}} + \frac{\beta^{(m-2)}}{\alpha^{(m-2)}} \end{bmatrix}. \tag{1.144}$$

1.5.2.2 Error bounds

It was shown that CG generates the optimal approximate solution from a Krylov subspace, where "optimal" is taken to mean having an error with minimal **A**-norm. Here, we derive bounds on the appropriate error norm for the optimal approximation from a Krylov subspace, following the work of [59].

A goal is to derive a sharp upper bound on the reduction in the **A**-norm of the error–that is, an upper bound that is independent of the initial vector but that is actually attained for certain initial vectors. This describes the worst-case behavior of the algorithm (for a given matrix **A**). It can sometimes be shown that the "typical" behavior of the algorithm is not much different from the worst-case behavior. That is, if the initial vector is random, then convergence may be only moderately faster than for the worst initial vector. For certain special initial vectors, however, convergence may be much faster than the worst-case analysis would suggest. Still, it is usually the same analysis that enables one to identify these "special" initial vectors, and it is often clear how the bound must be mofified to account for special properties of the initial vector.

It was shown that the **A**-norm of the error in the CG algorithm for SPD problems is minimized over the space

$$\mathbf{e}^{(0)} + \text{span}\{\mathbf{A}\mathbf{e}^{(0)}, \mathbf{A}^2\mathbf{e}^{(0)}, \ldots, \mathbf{A}^k\mathbf{e}^{(0)}\}. \tag{1.145}$$

It follows that the CG error vector at step j can be written in the form

$$\mathbf{e}^{(j)} = P_j(\mathbf{A})\mathbf{e}^{(0)} \tag{1.146}$$

where P_j is the jth-degree polynomial with value 1 at the origin and, of all such polynomials that could be substituted in Eq. (1.146), P_j gives the error of minimal **A**-norm in the CG algorithm. In other words, the error $\mathbf{e}^{(j)}$ in the CG approximation satisfies

$$\|\mathbf{e}^{(j)}\|_{\mathbf{A}} = \min_{p_j} \|p_j(\mathbf{A})\mathbf{e}^{(0)}\|_{\mathbf{A}} \tag{1.147}$$

where the minimum is taken over all polynomials p_j of degree j or less with $p_j(0) = 1$.

Here, we derive bounds on the expression in the right-hand side of Eq. (1.147) that are independent of the direction of the initial error $\mathbf{e}^{(0)}$, although they do depend on the size of this quantity. A sharp upper bound is derived involving all of the eigenvalues of **A**, and then a simpler (but nonsharp) bound is given based on the knowledge of just a few of the eigenvalues of **A**.

Let an eigen-decomposition of **A** be written as $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where **U** is a unitary matrix and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix of eigenvalues. If **A** is SPD, define $\mathbf{A}^{1/2}$ to be $\mathbf{U}\mathbf{\Lambda}^{1/2}\mathbf{U}^T$. Then the **A**-norm of a vector **v** is just the 2-norm of the vector $\mathbf{A}^{1/2}\mathbf{v}$. Eq. (1.147) implies that

$$\|\mathbf{e}^{(j)}\|_{\mathbf{A}} = \min_{p_j} \|\mathbf{A}^{1/2}p_j(\mathbf{A})\mathbf{e}^{(0)}\| = \min_{p_j} \|\mathbf{U}p_j(\mathbf{\Lambda})\mathbf{U}^T\mathbf{A}^{1/2}\mathbf{e}^{(0)}\| \leq \min_{p_j} \|p_j(\Lambda)\| \cdot \|\mathbf{e}^{(0)}\|_{\mathbf{A}} \tag{1.148}$$

with the inequalities following, because if $\hat{p}_j$ is the polynomial that minimizes $\|p_j(\Lambda)\|$,

then

$$\min_{p_j} \|\mathbf{U}p_j(\mathbf{\Lambda})\mathbf{U}^T\mathbf{w}\| \leq \|\mathbf{U}\hat{p}_j(\mathbf{\Lambda})\mathbf{U}^T\mathbf{w}\| \leq \|\mathbf{U}\hat{p}_j(\mathbf{\Lambda})\mathbf{U}^T\|\|\mathbf{w}\| \leq \|\hat{p}_j(\mathbf{\Lambda})\|\|\mathbf{w}\| \tag{1.149}$$

for any vector $\mathbf{w}$. Of course, the polynomial that minimizes the expression in Eq. (1.148) is not necessarily the same one that minimizes $\|p_j(\mathbf{\Lambda})\|$ in the inequalities. The CG polynomial depends on the initial vector, while this polynomial does not. Hence it is not immediately obvious that the bound in Eq. (1.148) is sharp, that is, that it can actually be attained for certain initial vectors. It turns out that this is the case, however. See, for example, [58, 60]. For each j there is an initial vector $\mathbf{e}^{(j)}$ for which the CG polynomial at step j is the polynomial that minimizes $\|p_j(\mathbf{\Lambda})\|$ and for which equality holds in Eq. (1.148).

The sharp upper bound (1.148) can be written in the form

$$\frac{\|\mathbf{e}^{(j)}\|_{\mathbf{A}}}{\|\mathbf{e}^{(0)}\|_{\mathbf{A}}} \leq \min_{p_j} \max_{i=1,\ldots,n} |p_j(\lambda_i)|. \tag{1.150}$$

The problem of describing the convergence of the CG algorithm therefore reduces to one in approximation theory–how well can one approximate zero on the set of eigenvalues of $\mathbf{A}$ using a jth-degree polynomial with value 1 at the origin. While there is no simple expression for the maximum value of the minimax polynomial on a discrete set of points, this minimax polynomial can be calculated if the eigenvalues of $\mathbf{A}$ are known; more importantly, this sharp upper bound provides intuition as to what constitutes "good" and "bad" eigenvalue distributions. Eigenvalues tightly clustered around a single point (away from the origin) are good, for instance, because the polynomial $(1 - z/c)^j$ is small in absolute value at all points near c. Widely spread eigenvalues, especially if they lie on both sides of the origin, are bad, because a low-degree polynomial with value 1 at the origin cannot be small at a large number of such points.

Since one usually has only limited information about the eigenvalues of $\mathbf{A}$, it is useful to have error bounds that involve only a few properties of the eigenvalues. For example, knowing only the largest and smallest eigenvalues of $\mathbf{A}$, one can obtain an error bound by considering the minimax polynomial on the interval from λ_{min} to λ_{max}, i.e., the Chebyshev polynomial shifted to the interval and scaled to have value 1 at the origin.

Theorem 4. *Let $\mathbf{e}^{(j)}$ be the error at step j of the CG algorithm applied to the SPD linear system $\mathbf{Au} = \mathbf{b}$. Then*

$$\frac{\|\mathbf{e}^{(j)}\|_{\mathbf{A}}}{\|\mathbf{e}^{(0)}\|_{\mathbf{A}}} \leq 2\left[\left(\frac{\sqrt{\text{cond}}-1}{\sqrt{\text{cond}}+1}\right)^j + \left(\frac{\sqrt{\text{cond}}+1}{\sqrt{\text{cond}}-1}\right)^j\right]^{-1} \leq 2\left(\frac{\sqrt{\text{cond}}-1}{\sqrt{\text{cond}}+1}\right)^j \tag{1.151}$$

where $\text{cond} = \lambda_{max}/\lambda_{min}$ *is the ratio of the largest to smallest eigenvalue of* $\mathbf{A}$.

Proof. Consider the jth scaled and shifted Chebyshev polynomial on the interval $[\lambda_{min}, \lambda_{max}]$

$$p_j(z) = \frac{T_j\left(\frac{2z-\lambda_{max}-\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)}{T_j\left(\frac{-\lambda_{max}-\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)} \tag{1.152}$$

where $T_j(z)$ is the Chebyshev polynomial of the first kind on the interval $[-1,1]$ satisfying

$$T_0(z) = 1, \;\; T_1(z) = z, \tag{1.153}$$

$$T_{k+1}(z) = 2zT_k(z) - T_{k-1}(z) \;, \;\; k = 1, 2, \ldots. \tag{1.154}$$

In the interval $[-1,1]$, we have

$$T_j(z) = \cosh(j \cosh^{-1} z), \tag{1.155}$$

so if z is of the form $z = \cosh(\ln y) = \frac{1}{2}(y + y^{-1})$, then $T_j(z) = \frac{1}{2}(y^j + y^{-j})$. The argument in the denominator of Eq. (1.152) can be expressed in the form $\frac{1}{2}(y + y^{-1})$ if y satisfies

$$-\frac{\lambda_{max} + \lambda_{min}}{\lambda_{max} - \lambda_{min}} = -\frac{\text{cond} + 1}{\text{cond} - 1} = \frac{1}{2}(y + y^{-1}), \tag{1.156}$$

which is equivalent to the quadratic equation

$$\frac{1}{2}y^2 + \frac{\text{cond} + 1}{\text{cond} - 1}y + \frac{1}{2} = 0. \tag{1.157}$$

Solving this equation, we find

$$y = -\frac{\sqrt{\text{cond}} + 1}{\sqrt{\text{cond}} - 1} \;\text{ or }\; y = -\frac{\sqrt{\text{cond}} - 1}{\sqrt{\text{cond}} + 1}. \tag{1.158}$$

In either case, the denominator in Eq. (1.152) has absolute value equal to

$$\frac{1}{2}\left[\left(\frac{\sqrt{\text{cond}} - 1}{\sqrt{\text{cond}} + 1}\right)^j + \left(\frac{\sqrt{\text{cond}} + 1}{\sqrt{\text{cond}} - 1}\right)^j\right] \tag{1.159}$$

and from this result Eq. (1.151) follows. □

Knowing only the largest and smallest eigenvalues of a SPD matrix $\mathbf{A}$, bound (1.151) is the best possible. If the interior eigenvalues of $\mathbf{A}$ lie at the points where the Chebyshev polynomial p_j in Eq. (1.152) attains its maximum absolute value on $[\lambda_{min}, \lambda_{max}]$, then for a certain initial error $\mathbf{e}^{(0)}$, the CG polynomial will be equal to the Chebyshev polynomial and the bound in Eq. (1.151) will actually be attained at step j.

If additional information is available about the interior eigenvalues of $\mathbf{A}$, one can often improve on the estimate of Eq. (1.151) while maintaining a simpler expression than the sharp bound of Eq. (1.150). Suppose, for example, that $\mathbf{A}$ has one eigenvalue much larger than the others, say, $\lambda_1 \leq \cdots \leq \lambda_{n-1} \ll \lambda_n$, that is, $\lambda_n/\lambda_{n-1} \gg 1$. Consider a polynomial p_j that is the product of a linear factor that is zero at λ_n and the $(j-1)$st-degree scaled and shifted Chebyshev polynomial on the interval $[\lambda_1, \lambda_{n-1}]$:

$$p_j(z) = \frac{T_{j-1}\left(\frac{2z - \lambda_{n-1} - \lambda_1}{\lambda_{n-1} - \lambda_1}\right)}{T_{j-1}\left(\frac{-\lambda_{n-1} - \lambda_1}{\lambda_{n-1} - \lambda_1}\right)} \cdot \left(\frac{\lambda_n - z}{\lambda_n}\right). \tag{1.160}$$

Since the second factor is zero at λ_n and less than one in absolute value at each of the

other eigenvalues, the maximum absolute value of this polynomial on $\{\lambda_1, \ldots, \lambda_n\}$ is less than the maximum absolute value of the first factor on $\{\lambda_1, \ldots, \lambda_{n-1}\}$. Using arguments like those in Theorem 4, it follows that

$$\frac{\|\mathbf{e}^{(j)}\|_{\mathbf{A}}}{\|\mathbf{e}^{(0)}\|_{\mathbf{A}}} \leq 2\left(\frac{\sqrt{\text{cond}}_{n-1}-1}{\sqrt{\text{cond}_{n-1}}+1}\right)^{j-1}, \quad \text{cond}_{n-1} = \frac{\lambda_{n-1}}{\lambda_1}. \tag{1.161}$$

Similarly, if the matrix $\mathbf{A}$ has just a few large outlying eigenvalues, say, $\lambda_1 \leq \cdots \leq \lambda_{n-\ell} \ll \lambda_{n-\ell+1} \leq \cdots \leq \lambda_n$, i.e., $\lambda_{n-\ell+1}/\lambda_{n-\ell} \gg 1$, one can consider a polynomial p_j that is the product of an ℓth-degree factor that is zero at each of the outliers (and less than one in magnitude at each of the other eigenvalues) and a scaled and shifted Chebyshev polynomial of degree $k-\ell$ on the interval $[\lambda_1, \lambda_{n-\ell}]$. Bounding the size of this polynomial gives

$$\frac{\|\mathbf{e}^{(j)}\|_{\mathbf{A}}}{\|\mathbf{e}^{(0)}\|_{\mathbf{A}}} \leq 2\left(\frac{\sqrt{\text{cond}}_{n-\ell}-1}{\sqrt{\text{cond}_{n-\ell}}+1}\right)^{j-\ell}, \quad \text{cond}_{n-\ell} = \frac{\lambda_{n-\ell}}{\lambda_1}. \tag{1.162}$$

1.6 Parallel preconditioners

We present here a handful of preconditioners for the iterative solves of Eq. (1.5) which are well-adapted for parallel applications. This presentation is by no means exhaustive. Nevertheless, it provides a decent range of techniques to benchmark the preconditioning strategies investigated throughout this thesis. First, we present block Jacobi (bJ) preconditioners, then non-overlapping domain decomposition methods followed by algebraic multigrids. Some of the preconditioners presented were implemented using the Julia programming language. Other well-known parallel preconditioning techniques, not used in this work, are mentioned in the end of the Section.

Roughly speaking, a preconditioner is any form of implicit or explicit modification of an original linear system that makes it easier to solve by a given iterative method. For example, scaling all rows of a linear system to make the diagonal elements equal to one is an explicit form of preconditioning. The resulting system can be solved by a Krylov subspace method and may require fewer steps to converge than the original system (although this is not guaranteed). As another example, solving the linear system

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{u} = \mathbf{M}^{-1}\mathbf{b} \tag{1.163}$$

where $\mathbf{M}^{-1}$ is some complicated mapping that may involve fast Fourier transforms (FFT), integral calculations, and subsidiary linear system solutions, may be another form of preconditioning. Here, it is unlikely that the matrices $\mathbf{M}$ and $\mathbf{M}^{-1}\mathbf{A}$ can be computed explicitly. Instead, the iterative processes operate with $\mathbf{A}$ and with $\mathbf{M}^{-1}$ whenever needed. In practice, the preconditioning operation $\mathbf{M}^{-1}$ should be inexpensive to apply to an arbitrary vector. Meanwhile, a parallel preconditioning operation is one that is well adapted for parallel application.

1.6.1 Block Jacobi preconditioners

We first consider non-overlapping diagonal bJ preconditioners of the form

$$\mathbf{M}_{\mathrm{bJ}}^{-1} := \sum_{j=1}^{n_b} \boldsymbol{\mathcal{R}}^{(j)^T} \mathbf{M}_j^{-1} \boldsymbol{\mathcal{R}}^{(j)} \tag{1.164}$$

with canonical restrictions $\boldsymbol{\mathcal{R}}^{(j)}$ from a global vector to each block $j \in [1, n_b]$. For each j, $\mathbf{M}_j^{-1}$ is applied with a Cholesky factorization $\mathbf{M}_j = \mathbf{L}_j \mathbf{L}_j^T = \boldsymbol{\mathcal{R}}^{(j)^T} \mathbf{A} \boldsymbol{\mathcal{R}}^{(j)}$.

1.6.2 Non-overlapping domain decomposition methods

Domain decomposition refers to an ensemble of numerical methods which rely on a divide-and-conquer paradigm to solve computational problems. Those are commonly referred to as either algebraic or domain-specific, depending on whether their formulation explicitly relies on the underlying problem they solve. While a substantial part of the scientific literature on domain decomposition emphasizes applications to linear solves of discretized PDEs, there exist communities, such as structural engineering, that resort to domain decomposition to solve large systems of algebraic equations which are not induced by the discretization of PDEs. In both cases, domain decomposition is a means to circumvent the limits of computer memory to solve large systems of equations. These methods have proven to be useful to solve PDEs on "non-standard" geometries decomposed into sub-domains that are more amenable to efficient solves such as when using fast Poisson solvers on structured rectangular sub-grids. Another use of domain decomposition is to allow a partition of the domain into sub-domains within which distinct physics are considered from one part of the domain to the others. But perhaps the most sought out feature of these methods within the scientific computing community is the natural way they offer to exploit parallel computing architectures. The zoology of these methods is vast enough that providing an exhaustive review goes beyond the scope of this work. For more general and complementary presentations of domain decomposition, we direct the reader to [26, 147, 155, 126].

Just like any other method aimed at solving a linear system, domain decomposition methods can be used to derive preconditioners. Here, we focus on preconditioners designed for and induced by Schur complement operators which arise in non-overlapping domain decomposition methods.

We briefly introduce non-overlapping domain decomposition as follows. For simplicity, we assume that $\mathbf{A}$ is obtained by spatial discretization with nodal finite elements of a scalar PDE subjected to Dirichlet boundary conditions on a domain Ω with boundary $\partial\Omega$. Then, a non-overlapping domain decomposition consists of sub-domains $\Omega_1, \ldots, \Omega_{n_d}$ of Ω such that

$$\overline{\Omega} = \bigcup_{d=1}^{n_d} \overline{\Omega_d} \quad \text{and} \quad \Omega_d \cap \Omega_{d'} = \emptyset \text{ if } d \neq d', \tag{1.165}$$

where $\overline{\Omega_d}$ denotes the closure of Ω_d. Each sub-domain Ω_d has a boundary $\partial\Omega_d$ and a local interface $\Gamma_d := \partial\Omega_d \backslash \partial\Omega$. Since we rely on finite elements, it is natural to let Ω_d be a union of elements, so that no element is split by the global interface $\Gamma = \cup_{d=1}^{n_d} \Gamma_d$ across different

sub-domains. Such decompositions are referred to as element-based partitions [138] and are characterized by sets of indices $\mathcal{N}_{\mathrm{I}}^{(1)}, \ldots, \mathcal{N}_{\mathrm{I}}^{(n_d)}, \mathcal{N}_\Gamma$ where the indices in $\mathcal{N}_{\mathrm{I}}^{(d)}$ refer to the finite element nodes which are in the interior of Ω_d, and $\mathcal{N}_\Gamma$ contains the indices of all the nodes in Γ. A convenient means of generating such domain decompositions is to operate on a graph representation of the finite element mesh. That is, each element of the mesh is represented by a vertex of a graph, and each edge of this graph indicates that the two elements it points to have at least one node in common. Then, a non-overlapping domain decomposition, i.e., a sub-structuring, can be obtained by vertex-based partitioning of the graph using either multilevel k-way, multilevel recursive bisection, multi-constraint partitioning or spectral clustering algorithms—all of which are readily deployable through softwares such as METIS [74] or Scotch [27].

Once equipped with a partition of the nodes, a permutation π can be defined onto $\mathcal{N}_{\mathrm{I}}^{(1)} \cup \cdots \cup \mathcal{N}_{\mathrm{I}}^{(n_d)} \cup \mathcal{N}_\Gamma$ with a matrix $\mathbf{P}_\pi$ such that, if Eq. (1.5) is already assembled, it can be re-ordered into a block arrow structure as follows:

$$\begin{gathered}\mathbf{P}_\pi \mathbf{A} \mathbf{P}_\pi^T (\mathbf{P}_\pi \mathbf{u}) = \mathbf{P}_\pi \mathbf{b} \\ \begin{bmatrix} \mathbf{A}_{\mathrm{II}}^{(1)} & & & \mathbf{A}_{\mathrm{I}\Gamma}^{(1)} \\ & \ddots & & \vdots \\ & & \mathbf{A}_{\mathrm{II}}^{(n_d)} & \mathbf{A}_{\mathrm{I}\Gamma}^{(n_d)} \\ \mathbf{A}_{\Gamma\mathrm{I}}^{(1)} & \cdots & \mathbf{A}_{\Gamma\mathrm{I}}^{(n_d)} & \mathbf{A}_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \mathbf{u}_{\mathrm{I}}^{(1)} \\ \vdots \\ \mathbf{u}_{\mathrm{I}}^{(n_d)} \\ \mathbf{u}_\Gamma \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\mathrm{I}}^{(1)} \\ \vdots \\ \mathbf{b}_{\mathrm{I}}^{(n_d)} \\ \mathbf{b}_\Gamma \end{bmatrix}\end{gathered} \tag{1.166}$$

where $\mathbf{u}_{\mathrm{I}}^{(d)}$ and $\mathbf{u}_\Gamma$ contain the solution of the discretized equation at the nodes within Ω_d and in Γ, respectively.

Conversely, if Eq. (1.5) is not assembled prior to partitioning the domain, then Eq. (1.166) can be obtained by reduction of extended local assemblies from all the sub-domains. That is, for a given sub-domain, a local assembly is performed by considering the discrete matrix and right-hand side contributions which arise from the elements in Ω_d. Numbering the interior nodes first, we obtain

$$\mathbf{A}^{(d)} = \begin{bmatrix} \mathbf{A}_{\mathrm{II}}^{(d)} & \mathbf{A}_{\mathrm{I}\Gamma}^{(d)} \\ \mathbf{A}_{\Gamma\mathrm{I}}^{(d)} & \mathbf{A}_{\Gamma\Gamma}^{(d)} \end{bmatrix} \quad , \quad \mathbf{b}^{(d)} = \begin{bmatrix} \mathbf{b}_{\mathrm{I}}^{(d)} \\ \mathbf{b}_\Gamma^{(d)} \end{bmatrix} \quad , \quad d = 1, \ldots, n_d. \tag{1.167}$$

Note that $\mathbf{A}^{(d)}$ corresponds to a discretization of the PDE on Ω_d with Neumann boundary conditions on Γ_d, whereas $\mathbf{A}_{\mathrm{II}}^{(d)}$ corresponds to a discretization with homogeneous Dirichlet boundary conditions on Γ_d. Therefore, if Ω_d is an internal sub-domain, i.e., if $\partial\Omega_d \cap \partial\Omega = \emptyset$, then $\mathbf{A}^{(d)}$ is singular. Nevertheless, irrespectively of the well-posedness of the local problems, the global blocks $\mathbf{A}_{\Gamma\Gamma}$ and $\mathbf{b}_\Gamma$ of Eq.(1.166) are obtained by summing extensions of the local contributions from all the sub-domains. That is

$$\mathbf{A}_{\Gamma\Gamma} = \sum_{d=1}^{n_d} \boldsymbol{\mathcal{R}}_\Gamma^{(d)^T} \mathbf{A}_{\Gamma\Gamma}^{(d)} \boldsymbol{\mathcal{R}}_\Gamma^{(d)} \quad , \quad \mathbf{b}_\Gamma = \sum_{d=1}^{n_d} \boldsymbol{\mathcal{R}}_\Gamma^{(d)^T} \mathbf{b}_\Gamma^{(d)} \tag{1.168}$$

where $\boldsymbol{\mathcal{R}}_\Gamma^{(d)}$ is the canonical restriction matrix that maps global vectors of nodal values in Γ to local vectors of nodal values in Γ_d.

From the point of view of linear algebra, the main feature of non-overlapping do-

main decomposition arises from a proper manipulation of the block arrow structure of Eq. (1.166) to yield a *condensed* linear system whose sole unknowns lie on the global interface Γ. To do so, it suffices to eliminate $\mathbf{u}_{\mathrm{I}}^{(1)}, \ldots, \mathbf{u}_{\mathrm{I}}^{(n_d)}$ from Eq. (1.166) so as to obtain a system of the form

$$\mathbf{S}\,\mathbf{u}_\Gamma = \mathbf{b}_{\mathrm{S}}\ , \quad \mathbf{b}_{\mathrm{S}} = \mathbf{b}_\Gamma - \sum_{d=1}^{n_d} \mathbf{A}_{\Gamma\mathrm{I}}^{(d)}\, \mathbf{A}_{\mathrm{II}}^{(d)^{-1}}\, \mathbf{b}_{\mathrm{I}}^{(d)}, \tag{1.169}$$

where $\mathbf{S}$ denotes the global Schur complement matrix given by

$$\mathbf{S} = \sum_{d=1}^{n_d} \boldsymbol{\mathcal{R}}_\Gamma^{(d)^T}\, \mathbf{S}^{(d)}\, \boldsymbol{\mathcal{R}}_\Gamma^{(d)} \tag{1.170}$$

in terms of local Schur complements

$$\mathbf{S}^{(d)} = \mathbf{A}_{\Gamma\Gamma}^{(d)} - \mathbf{A}_{\Gamma\mathrm{I}}^{(d)}\, \mathbf{A}_{\mathrm{II}}^{(d)^{-1}}\, \mathbf{A}_{\mathrm{I}\Gamma}^{(d)}\ , \quad d = 1, \ldots, n_d. \tag{1.171}$$

This additive decomposition of $\mathbf{S}$ into local contributions sheds light on another important feature of sub-structuring, i.e., its inherent propensity for parallel implementations. Let us remember that our goal is not to solve Eq. (1.169) with perfect accuracy, but rather to have a parallel preconditioner based on domain decomposition for the iterative solve of Eq. (1.5), in which case an approximate solution of Eq. (1.169) is good enough. Once a solution (resp. approximate solution) is obtained for the condensed linear system, the corresponding solution (resp. approximate solution) at the interior nodes is obtained by solving an independent linear system for each sub-domain. If the exact solution is known for the nodes in Γ, these systems are

$$\mathbf{A}_{\mathrm{II}}^{(d)}\, \mathbf{u}_{\mathrm{I}}^{(d)} = \mathbf{b}_{\mathrm{I}}^{(d)} - \mathbf{A}_{\mathrm{I}\Gamma}^{(d)}\, \boldsymbol{\mathcal{R}}_\Gamma^{(d)}\, \mathbf{u}_\Gamma\ , \quad d = 1, \ldots, n_d. \tag{1.172}$$

Clearly, these local solves can be carried on in a distributed fashion.

We note that, as long as $\mathbf{A}$ is SPD, so is the global Schur complement. Then, for a given sub-domain Ω_d, the local Schur complement $\mathbf{S}^{(d)}$ is only guaranteed to be symmetric. Moreover, if Ω_d is an internal sub-domain, then $\mathbf{S}^{(d)}$ is singular. These singularities need to be taken into account when implementing a domain decomposition preconditioner, and doing so can significantly impact the scalability of the preconditioner with respect to the number of sub-domains.

1.6.2.1 Non-overlapping domain decomposition preconditioners

It is common to distinguish between two types of domain decomposition preconditioners. That is, one-level and two-level preconditioners. One-level preconditioners generally admit additive decompositions with each summand requiring to solve local linear systems, that is, involving quantities that relate to single sub-domains. As a result, these methods tend to only bound one side of the preconditioned spectra, offering poorer convergence rates of the preconditioned iterative method as the number of sub-domains increases. Two-level preconditioners attempt to compensate for this shortcoming and can generally be recast into additive or multiplicative corrections of one-level methods. More detailed

reviews on one-, two- and multi-level domain decomposition preconditioners can be found in Giraud and Tuminaro [54], Tang et al. [151] as well as Poirel [126]. Here, we only present a couple of one- and two-level variants: preconditioning the Schur complement with $\mathbf{A}_{\Gamma\Gamma}$ and the Neumann-Neumann method [35] as one-level variants, and the LORASC preconditioner [61] as a two-level variant. Other two-level variants not seen in this work include the balancing Neumann-Neumann method [98] as well as the works of Li and Saad [92], Al Daas et al. [34], and several others.

Preconditioning the Schur complement with $\mathbf{A}_{\Gamma\Gamma}$. The simplest form of domain decomposition preconditioner consists of approximating the Schur complement with $\mathbf{A}_{\Gamma\Gamma}$. This is equivalent to neglecting the term $\mathbf{A}_{\Gamma\mathrm{I}}^{(d)}\,\mathbf{A}_{\mathrm{II}}^{(d)^{-1}}\,\mathbf{A}_{\mathrm{I}\Gamma}^{(d)}$ in Eq. (1.171) for each local Schur complement. It can be shown that, while the spectrum of ${\mathbf{A}_{\Gamma\Gamma}}^{-1}\,\mathbf{S}$ is bounded above by one, the lowest eigenvalues of the spectrum get increasingly close to zero as the number of sub-domains increases. The implication is that the corresponding rate of convergence of the preconditioned iterative method degrades as the number of sub-domains increases.

The Neumann-Neumann (NN) preconditioner. A well-known preconditioner for Eq. (1.169) was proposed by De Roeck and Le Tallec [35]. This method is referred to as Neumann-Neumann (NN) preconditioning due to the nature of the boundary conditions on the interface of the subdomains when considering the local problems solved by the approximation. To define a NN preconditioner, one has to specify diagonal weight matrices $\mathbf{D}_1, \ldots, \mathbf{D}_{n_d}$ so as to form a partition of unity given by

$$\sum_{d=1}^{n_d} {\boldsymbol{\mathcal{R}}_\Gamma^{(d)}}^T\,\mathbf{D}_d\,\boldsymbol{\mathcal{R}}_\Gamma^{(d)} = \mathbf{I}. \tag{1.173}$$

A simple choice consists of letting $\mathbf{D}_d$ be diagonal with each non-zero component set to one divided by the number of subdomains to which the coerrsponding DoF is associated. Different choices of weight matrices are discussed in [86]. The application of the preconditioner is then done as described in Algo. 11.

Algorithm 11 Application of the Neumann-Neumann preconditioner

Input: $\mathbf{r} \in \mathrm{range}(\mathbf{S})$
Output: $\mathbf{z} = \mathbf{M}_{\mathrm{NN}}^{-1}\,\mathbf{r}$
1: **for** $d = 1, \ldots, n_d$ **do**
2: $\quad\mathbf{r}^{(d)} := \mathbf{D}_d\,\boldsymbol{\mathcal{R}}_\Gamma^{(d)}\,\mathbf{r}$ ▷ get local residual
3: $\quad$Find $\mathbf{z}^{(d)}$ s.t. $\mathbf{S}^{(d)}\mathbf{z}^{(d)} = \mathbf{r}^{(d)}$ ▷ solve local problem
4: **end for**
5: $\mathbf{z} := \sum_{d=1}^{n_d} \boldsymbol{\mathcal{R}}_\Gamma^{(d)}\,\mathbf{D}_d\mathbf{z}^{(d)}$ ▷ average contributions of all sub-domains

When a local Schur complement $\mathbf{S}^{(d)}$ is singular, a solution $\mathbf{z}^{(d)}$ of $\mathbf{S}^{(d)}\mathbf{z}^{(d)} = \mathbf{r}^{(d)}$ exists if and only if $\mathbf{r}^{(d)} \perp \ker \mathbf{S}^{(d)}$. Moreover, if such a solution exists, it is unique only up to some $\mathbf{v} \in \ker(\mathbf{S}^{(d)})$. Hence, to prevent a potential breakdown of the iterative method, step 3 in Algo. 11 can be replaced by $\mathbf{z}^{(d)} := {\mathbf{S}^{(d)}}^\dagger\mathbf{r}^{(d)}$ where ${\mathbf{S}^{(d)}}^\dagger$ is a pseudo-inverse of

$\mathbf{S}^{(d)}$. Doing so for all subdomains yields the following matrix representation:

$$\mathbf{M}_{\mathrm{NN}}^{-1} = \sum_{d=1}^{n_d} \boldsymbol{\mathcal{R}}_{\Gamma}^{(d)^T} \mathbf{D}_d \mathbf{S}^{(d)^\dagger} \mathbf{D}_d \boldsymbol{\mathcal{R}}_{\Gamma}^{(d)} . \tag{1.174}$$

An alternative to using pseudo-inverses is to apply the inverse of a SPD shift of the local Schur complement. Being one-leveled, the preconditioning offered by the Neumann-Neumann method generally does not scale well with the number of sub-domains.

The low-rank robust algebraic Schur (LORASC) preconditioner. LORASC was introduced by Grigori et al. [61] as a full-matrix preconditioner induced by a low-rank robust approximation of the Schur complement operator. This preconditioner is deemed robust in that the condition number $\mathrm{cond}(\mathbf{M}^{-1}\mathbf{P}_\pi\mathbf{A}\mathbf{P}_\pi^T)$ is bounded above by design, independently of the number of sub-domains. Consequently, LORASC is expected to exhibit a good scaling in terms of the number of sub-domains.

An essential observation for the design and understanding of the LORASC preconditioner is that the block-arrow structure of the re-ordered matrix can be recast in a form

$$\mathbf{P}_\pi\mathbf{A}\mathbf{P}_\pi^T = (\mathbf{L}+\mathbf{D})\mathbf{D}^{-1}(\mathbf{L}+\mathbf{D})^T. \tag{1.175}$$

where $\mathbf{L}$ is a block lower-triangular matrix given by

$$\mathbf{L} = \begin{bmatrix} 0 & & & \\ \vdots & \ddots & & \\ 0 & \cdots & 0 & \\ \mathbf{A}_{\mathrm{II}}^{(1)} & \cdots & \mathbf{A}_{\mathrm{II}}^{(n_d)} & 0 \end{bmatrix} \tag{1.176}$$

and $\mathbf{D}$ is the block-diagonal matrix

$$\mathbf{D} = \text{block-diag}(\mathbf{A}_{\mathrm{II}}, \mathbf{S}) \;\text{ with }\; \mathbf{A}_{\mathrm{II}} = \text{block-diag}(\mathbf{A}_{\mathrm{II}}^{(1)}, \cdots, \mathbf{A}_{\mathrm{II}}^{(n_d)}). \tag{1.177}$$

Substituting the exact Schur complement by an approximation $\widetilde{\mathbf{S}}$ in place of $\mathbf{S}$ within the block-diagonal matrix yields a preconditioner

$$\mathbf{M}_{\mathrm{LORASC}} = (\mathbf{L}+\widetilde{\mathbf{D}})\widetilde{\mathbf{D}}^{-1}(\mathbf{L}+\widetilde{\mathbf{D}})^T. \tag{1.178}$$

where

$$\widetilde{\mathbf{D}} := \text{block-diag}(\mathbf{A}_{\mathrm{II}}, \widetilde{\mathbf{S}}). \tag{1.179}$$

Doing so allows to define an approximation $\widetilde{\mathbf{S}}$ which is both less dense and easier to store and apply than the exact Schur complement. Once given such an approximation $\widetilde{\mathbf{S}}$, the LORASC preconditioner is applied as described in Algo. 12, in which

$$\mathbf{A}_{\Gamma\mathrm{I}} = \begin{bmatrix} \boldsymbol{\mathcal{R}}_{\Gamma}^{(1)^T} \mathbf{A}_{\Gamma\mathrm{I}}^{(1)} & \ldots & \boldsymbol{\mathcal{R}}_{\Gamma}^{(n_d)^T} \mathbf{A}_{\Gamma\mathrm{I}}^{(\mathrm{n_d})} \end{bmatrix} \;\text{ and }\; \mathbf{A}_{\mathrm{I}\Gamma} = \mathbf{A}_{\Gamma\mathrm{I}}{}^T . \tag{1.180}$$

Algorithm 12 Application of the LORASC preconditioner for a given approximation $\widetilde{\mathbf{S}}$ of $\mathbf{S}$

Input: $\mathbf{r} = \begin{bmatrix} \mathbf{r}_\mathrm{I} \\ \mathbf{r}_\Gamma \end{bmatrix} \in \mathrm{range}(\mathbf{P}_\pi \mathbf{A} \mathbf{P}_\pi^T)$
Output: $\mathbf{z} = \mathbf{M}_{\mathrm{LORASC}}^{-1} \mathbf{r}$
1: $\mathbf{y}_\mathrm{I} := {\mathbf{A}_{\mathrm{II}}}^{-1} \mathbf{r}_\mathrm{I}$
2: Find $\mathbf{y}_\Gamma$ s.t. $\widetilde{\mathbf{S}} \mathbf{y}_\Gamma = \mathbf{r}_\Gamma - \mathbf{A}_{\Gamma\mathrm{I}} \mathbf{r}_\mathrm{I}$
3: $\mathbf{z} := \begin{bmatrix} \mathbf{y}_\mathrm{I} - {\mathbf{A}_{\mathrm{II}}}^{-1} \mathbf{A}_{\mathrm{I}\Gamma} \mathbf{y}_\Gamma \\ \mathbf{y}_\Gamma \end{bmatrix}$

The main feature of LORASC comes from the design of $\widetilde{\mathbf{S}}$. Indeed, one could build a one-level preconditioner by letting $\widetilde{\mathbf{S}} := \mathbf{A}_{\Gamma\Gamma}$. However, this would simply yield a preconditioner which is a full-matrix equivalent of preconditioning the Schur complement with $\mathbf{A}_{\Gamma\Gamma}$. Instead, Grigori et al. [61] propose a form

$$\widetilde{\mathbf{S}}^{-1} := {\mathbf{A}_{\Gamma\Gamma}}^{-1} + \mathbf{E}_{n_r} \mathbf{\Sigma}_{n_r} \mathbf{E}_{n_r}^T \tag{1.181}$$

in which $\mathbf{E}_{n_r} \mathbf{\Sigma}_{n_r} \mathbf{E}_{n_r}^T$ is of a priori unknown (low) rank n_r, and such that it shifts the lower eigenvalues of the spectrum of $\widetilde{\mathbf{S}}^{-1}\mathbf{S}$ to some prescribed value ε such that $0 < \varepsilon < 1$. In particular, this is achieved by letting

$$\mathbf{E}_{n_r} := \begin{bmatrix} \mathbf{e}_1 & \dots & \mathbf{e}_{n_r} \end{bmatrix} \quad \text{and} \quad \mathbf{\Sigma}_{n_r} := \mathrm{diag}\left(\frac{\varepsilon - \lambda_1}{\lambda_1}, \dots, \frac{\varepsilon - \lambda_{\mathrm{n_r}}}{\lambda_{\mathrm{n_r}}} \right) \tag{1.182}$$

where $(\lambda_1, \mathbf{e}_1), \dots, (\lambda_{n_r}, \mathbf{e}_{n_r})$ are the n_r least dominant generalized eigen-pairs of

$$\mathbf{S}\mathbf{e} = \lambda \, \mathbf{A}_{\Gamma\Gamma} \, \mathbf{e} \tag{1.183}$$

such that $\lambda_1 \leq \dots \leq \lambda_{n_r} \leq \varepsilon \leq 1$ with $\mathbf{A}_{\Gamma\Gamma}$-orthonormal eigenvectors. Then, it can be shown that $\kappa(\mathbf{M}_{\mathrm{LORASC}}^{-1} \mathbf{P}_\pi \mathbf{A} \mathbf{P}_\pi^T) \leq 1/\varepsilon$, irrespective of the number of sub-domains. In practice, for a given ε, it can be shown that n_r does not increase significantly as the number of sub-domains increases. Also, the authors provide a randomized procedure to approximate the generalized eigen-pairs at lower computational cost based on the methods found in [163]. The application of the LORASC preconditioner for the form just described of the approximation $\widetilde{\mathbf{S}}$ is detailed in Algo. 13. Clearly, the steps 1 and 7, which both require applications of ${\mathbf{A}_{\mathrm{II}}}^{-1}$, can be executed in parallel due to the block-diagonal structure of $\mathbf{A}_{\mathrm{II}}$.

A distinction between this preconditioner and the two last domain decomposition preconditioners presented lies in the fact that $\mathbf{M}_{\mathrm{LORASC}}^{-1}$ is designed for the iterative solve of the full-matrix equation, i.e., Eq. (1.5). Alternatively, one could equivalently precondition Eq. (1.169) with $\widetilde{\mathbf{S}}$. However, there are advantages in resorting to full-matrix iterations over Schur iterations. The main benefit of not operating on Eq. (1.169) is to not have to compute exact applications of $\mathbf{S}$ at every solver iteration. Hence, resorting to a full matrix iteration can become advantageous for cases in which the application of the domain decomposition preconditioner is much simplified compared to the application of the exact Schur complement. For instance, one can precondition the full-matrix equation with the

Algorithm 13 Application of the LORASC preconditioner

Input: $\mathbf{r} = \begin{bmatrix} \mathbf{r}_\mathrm{I} \\ \mathbf{r}_\Gamma \end{bmatrix} \in \mathrm{range}(\mathbf{P}_\pi \mathbf{A} \mathbf{P}_\pi^T)$
Output: $\mathbf{z} = \mathbf{M}_{\mathrm{LORASC}}^{-1} \mathbf{r}$
1: $\mathbf{y}_\mathrm{I} := {\mathbf{A}_{\mathrm{II}}}^{-1} \mathbf{r}_\mathrm{I}$
2: $\boldsymbol{\eta}_\Gamma := \mathbf{r}_\Gamma - \mathbf{A}_{\Gamma\mathrm{I}} \mathbf{r}_\mathrm{I}$
3: $\mathbf{y}_\Gamma := {\mathbf{A}_{\Gamma\Gamma}}^{-1} \boldsymbol{\eta}_\Gamma$
4: **for** $r = 1, \ldots, n_r$ **do**
5: $\quad \mathbf{y}_\Gamma := \mathbf{y}_\Gamma + \frac{\varepsilon - \lambda_r}{\lambda_r} (\mathbf{e}_r^T \boldsymbol{\eta}_\Gamma) \mathbf{e}_r$
6: **end for**
7: $\mathbf{z} := \begin{bmatrix} \mathbf{y}_\mathrm{I} - {\mathbf{A}_{\mathrm{II}}}^{-1} \mathbf{A}_{\mathrm{I}\Gamma} \mathbf{y}_\Gamma \\ \mathbf{y}_\Gamma \end{bmatrix}$

approximation $\mathbf{A}_{\Gamma\Gamma}$ of $\mathbf{S}$ simply by removing the steps 4 to 6 from Algo. 13.

1.6.3 Algebraic multigrid methods

Algebraic multigrid (AMG) methods are a highly scalable set of procedures to solve and precondition iterative solves of linear systems. The term algebraic refers to the fact that the method is agnostic to the problem from which the linear system is derived. As such, AMG is an evolution achieved by Brandt, McCormick and Ruge [18, 19] of multigrid methods which were already used to efficiently solve linear systems which particularly arise from the discretization of PDEs. See Section 6.9 in [37] for a detailed presentation of multigrid solvers for the Poisson's equation. The idea behind multigrid methods is motivated as follows. Though most relaxation-type iterative methods, such as Richardson, weighted Jacobi and Gauss-Seidel, may converge slowly for typical problems, the components of the residuals in the directions of the eigenvectors of the iteration matrix corresponding to the large eigenvalues are dampped very rapidly. These eigenvectors are known as high frequency modes. The other components, associated with low frequency modes, are difficult to damp with standard relaxation. This causes a slowdown of all basic iterative methods. However, many of these modes are mapped naturally into high frequency modes on a coarser mesh, or grid. The idea is thus to transfer to a coarser grid to eliminate the corresponding error components. This process can be repeated with the help of recursion using a hierarchy of multiple grids referred to as multigrid. Each of the so-called grids (sometimes referred to as levels) is used as a uniform coarsening of the next finer one. The solution process, which involves relaxation sweeps on each grid, fine-grid-to-coarse-grid transfers of residuals and coarse-to-fine interpolation of corrections, constitutes a fast solver for the finest grid equations. Although this process first seems to rely on the geometry and continuous nature of the problem, the principles involved in solving the fine grid matrix system can be abstracted and applied to various classes of matrix problems. This abstraction is done by AMG methods. Well-known implementations of algebraic multigrids are Hypre [44] in C++ and PyAMG [11] in Python. Here, we focus on the use of AMG for preconditioning iterative solves with PCG. For this purpose, it is sufficient to introduce V-cycles and ignore the other variants of sequences of operations otherwise used for linear solves. A more detailed review of these sequences of operations can be found in Chapter 13 of [138].

1.6.3.1 V-cycle

Let us consider a multigrid with $L+1$ levels indexed by ℓ such that $\ell = 0$ denotes the coarsest level whereas the finest level is referred to by the index $\ell = L$. We denote the prolongation operator going from level ℓ to level $\ell+1$ by $\mathbf{P}_\ell$. Let us remember that the matrix $\mathbf{A}$ is SPD and the restriction operator going from level $\ell+1$ to level ℓ is simply given by $\mathbf{P}_\ell^T$. The Galerkin operator is denoted by $\mathbf{A}_\ell := \mathbf{P}_\ell^T \mathbf{A}_{\ell+1} \mathbf{P}_\ell$ for $\ell = 0, \ldots, L-1$ with $\mathbf{A}_L := \mathbf{A}$ while $\mathbf{b}_\ell := \mathbf{P}_\ell^T \mathbf{b}_{\ell+1}$ for $\ell = 1, \ldots, L-1$ with $\mathbf{b}_L := \mathbf{b}$. The construction of the prolongation operators $\mathbf{P}_0, \ldots, \mathbf{P}_{L-1}$ is done in a set-up phase. This phase can be completed using coarse-fine splittings, sometimes referred to as classical AMG [19, 136], as well as by smoothed aggregation [158]. Here, we use a Julia wrapper of PyAMG [11] and proceed by smoothed aggregation. When resorting to smoothed aggregation, we use a strength measure on the connectivity of nodes to define a strength of connection matrix which is used to identify so-called aggregates. The goal of the strength measure is to ensure that algebraically smooth error at each DoF in an aggregate strongly correlates with algebraically smooth error at other DoFs in that aggregate.

A V-cycle corresponds to a coarse grid cycle and a relaxation sweep. The corresponding recursive algorithm is given by Algo. 14 where the numbers of presmoothing and postsmoothing steps are respectively given by ν_1 and ν_2. That is, the notation

$$\mathbf{u}_\nu^{(\ell)} = \text{smooth}^\nu(\mathbf{A}_\ell, \mathbf{u}_0^{(\ell)}, \mathbf{b}_\ell) \tag{1.184}$$

means that $\mathbf{u}_\nu^{(\ell)}$ is the result of ν smoothing steps for solving the system starting with the initial guess $\mathbf{u}_0^{(\ell)}$. Smoothing iterations are of the form

$$\mathbf{u}_{j+1}^{(\ell)} = \mathbf{S}_\ell \mathbf{u}_j^{(\ell)} + \mathbf{g}_\ell \tag{1.185}$$

where $\mathbf{S}_\ell$ is the iteration matrix associated with one smoothing step, that is, one iteration of a relaxation-type iterative method. By default, we consider that both presmoothing and postsmoothing consist of a single weighted Jacobi iteration. That is $\nu_1 = \nu_2 = 1$ and if we consider the decomposition

$$\mathbf{A}_\ell = \mathbf{L}_\ell + \mathbf{U}_\ell + \mathbf{D}_\ell \tag{1.186}$$

where $\mathbf{L}_\ell$ is the strictly lower triangular part of $\mathbf{A}_\ell$, $\mathbf{U}_\ell$ is the strictly upper triangular part of $\mathbf{A}_\ell$ and $\mathbf{D}_\ell := \text{diag}(\mathbf{A}_\ell)$, then we have

$$\mathbf{S}_\ell := (1-\omega)\mathbf{I} - \omega \mathbf{D}_\ell^{-1}(\mathbf{L}_\ell + \mathbf{U}_\ell) = \mathbf{I} - \omega \mathbf{D}_\ell^{-1} \mathbf{A}_\ell \quad \text{and} \quad \mathbf{g}_\ell := \omega \mathbf{D}_\ell^{-1} \mathbf{b}_\ell. \tag{1.187}$$

The convergence of the weighted Jacobi iteration method is guaranteed for $0 < \omega < 2/\rho(\mathbf{D}_\ell^{-1}\mathbf{A}_\ell)$ where $\rho(\bullet)$ denotes the spectral radius. By default, we let $\omega := 4\rho(\mathbf{A}_\ell)/3$. These parameters correspond to the default implementation of the AMG preconditioner in PyAMG [11]. We say that the AMG preconditioner is applied when the multigrid V-cycle is applied once at level L.

Algorithm 14 Application of a multigrid V-cycle at level ℓ

Input: $\mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{P}_0, \ldots, \mathbf{P}_{\ell-1}, \mathbf{b}_\ell, \mathbf{u}_0^{(\ell)}$
Output: $\mathbf{u}^{(\ell)} = \text{V-cycle}(\mathbf{A}_\ell, \mathbf{u}_0^{(\ell)}, \mathbf{b}_\ell)$

1: $\mathbf{u}^{(\ell)} := \text{smooth}^{\nu_1}(\mathbf{A}_\ell, \mathbf{u}_0^{(\ell)}, \mathbf{b}_\ell)$ ▷ presmooth
2: $\mathbf{r}_\ell := \mathbf{b}_\ell - \mathbf{A}_\ell \mathbf{u}^{(\ell)}$ ▷ compute residual
3: $\mathbf{r}_{\ell-1} := \mathbf{P}_{\ell-1}^T \mathbf{r}_\ell$ ▷ restrict residual
4: **if** $\ell = 1$ **then**
5: $\quad \mathbf{e}_0 := \mathbf{A}_0^{-1} \mathbf{r}_0$ ▷ solve the coarsest problem
6: **else**
7: $\quad \mathbf{e}_{\ell-1} := \text{V-cycle}(\mathbf{A}_{\ell-1}, \mathbf{0}, \mathbf{r}_{\ell-1})$ ▷ recurse
8: **end if**
9: $\mathbf{u}^{(\ell)} := \mathbf{u}^{(\ell)} + \mathbf{P}_{\ell-1} \mathbf{e}_{\ell-1}$ ▷ prolongate coarse grid correction
10: $\mathbf{u}^{(\ell)} := \text{smooth}^{\nu_2}(\mathbf{A}_\ell, \mathbf{u}^{(\ell)}, \mathbf{b}_\ell)$ ▷ postsmooth

1.6.4 Other methods

1.6.4.1 Incomplete LU factorizations

Iterative methods converge very fast if the matrix $\mathbf{A}$ is close to the identity matrix in some sense, and the main goal of preconditioning is to obtain a matrix $\mathbf{M}^{-1}\mathbf{A}$ which is close to $\mathbf{I}$. For Krylov subspace methods it is desirable that the condition number of $\mathbf{M}^{-1}\mathbf{A}$ is (much) smaller than that of $\mathbf{A}$, or that the eigenvalues of $\mathbf{M}^{-1}\mathbf{A}$ are strongly clustered around some point (usually 1). It is quite natural to start looking at a direct solution method for $\mathbf{Au} = \mathbf{b}$ and to see what variations we can make if the direct approach becomes too expensive. The most common direct technique is to factorize $\mathbf{A}$ as $\mathbf{A} = \mathbf{LU}$, if necessary with permutations for pivoting. One of the main problems with the *LU* factorization of a sparse matrix is that often the number of entries in the factors is substantially greater than in the original matrix so that, even if the original matrix can be stored, the factors cannot.

In incomplete *LU* factorization, we keep the factors artificially sparse in order to save computer time and storage for the decomposition. The incomplete factors are used for preconditioning in the following way. First note that we never need the matrices $\mathbf{A}$ or $\mathbf{M}$ explicitly, but we only need to be able to compute the result of $\mathbf{Ay}$ for any given vector $\mathbf{y}$. The same holds for the preconditioner $\mathbf{M}$, and typically we see in codes that these operations are performed by calls to appropriate subroutines. We need to be able to compute efficiently the result of $\mathbf{M}^{-1}\mathbf{y}$ for any vector $\mathbf{y}$. In the case of an incomplete *LU* factorization, $\mathbf{M}$ is given in the form $\mathbf{M} = \tilde{\mathbf{L}}\tilde{\mathbf{U}}$, where $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{U}}$ denote incomplete factors whereas $\mathbf{L}$ and $\mathbf{U}$ are the actual *LU* factors. $\mathbf{z} = \mathbf{M}^{-1}\mathbf{y}$ is computed by solving $\mathbf{z}$ from $\tilde{\mathbf{L}}\tilde{\mathbf{U}}\mathbf{z} = \mathbf{y}$. This is done in two steps: first solve $\mathbf{w}$ from $\tilde{\mathbf{L}}\mathbf{w} = \mathbf{y}$ and then compute $\mathbf{z}$ from $\tilde{\mathbf{U}}\mathbf{z} = \mathbf{w}$. Note that these solution steps are simple back substitutions and, if the right-hnd sides are not required further in the iterative process, the solution of either back substitution may overwrite the corresponding right-hand side, in order to save memory.

There is much recent effort to develop incomplete *LU* preconditioners that can be computed and used on parallel computers. Most of this work has been confined to highly structured problems from discretizations of elliptic PDEs in two and three dimensions, see for example [157]. Experiments with unstructured matricese have been reported in [67,

70], with reasonable speedups being achieved in [70].

1.6.4.2 Sparse Approximate Inverse (SPAI)

Of course, the *LU* factorization is one way of representing the inverse of a sparse matrix in a way that can be economically used to solve linear systems. The main reason why explicit inverses are not used is that, for irreducible matrices, the inverse will always be structurally dense. That is to say, sparse techniques will produce a dense matrix even if some of its entries are zero [40]. However, this need not be a problem if we follow the flavor of ILU factorizations and compute and use a sparse approximation to the inverse. Perhaps the most obvious technique for this is to solve the problem

$$\min_{\mathbf{M}^{-1}} \|\mathbf{I} - \mathbf{A}\mathbf{M}^{-1}\|_F \tag{1.188}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. The matrix $\mathbf{M}^{-1}$ in Eq. (1.188) has some fully or partially prescribed sparsity structure. One advantage of this is that this problem can be split into n independent least-squares problems for each of the n columns of $\mathbf{M}^{-1}$. Each of these least-squares problems only involves a few variables (corresponding to the number of entries in the column of $\mathbf{M}^{-1}$) and, because they are independent, they can be solved in parallel.

1.6.5 Preconditioning strategies

We recall that two basic preconditioning strategies are investigated in this chapter. First, there is the case in which the preconditioner $\mathbf{M}_\bullet(\theta)$ denominated by $\bullet$ is re-defined for every matrix realization $\mathbf{A}(\theta)$. This strategy was referred to as realization-dependent ideal preconditioning. In other words, this strategy consists of considering the unique approximation $\mathbf{u}^{(j)}(\theta)$ of $\mathbf{u}(\theta)$ which satisfies

$$\mathbf{u}^{(j)}(\theta) - \mathbf{u}^{(0)} \in \mathcal{K}^{(j)}(\mathbf{M}_\bullet^{-1}(\theta)\mathbf{A}(\theta), \mathbf{M}_\bullet^{-1}(\theta)\mathbf{r}^{(0)}(\theta)) \tag{1.189}$$

$$\mathbf{r}^{(j)}(\theta) \perp \mathcal{K}^{(j)}(\mathbf{M}_\bullet^{-1}(\theta)\mathbf{A}(\theta), \mathbf{M}_\bullet^{-1}(\theta)\mathbf{r}^{(0)}(\theta)) \tag{1.190}$$

where $\mathcal{K}^{(j)}(\mathbf{A}(\theta), \mathbf{r}^{(0)}(\theta)) := \text{Span}\{\mathbf{r}^{(0)}(\theta), \mathbf{A}(\theta)\mathbf{r}^{(0)}(\theta), \ldots, \mathbf{A}^{j-1}(\theta)\mathbf{r}^{(0)}(\theta)\}$ is the Krylov subspace of $\mathbf{A}(\theta)$ generated by the residual $\mathbf{r}^{(0)}(\theta) := \mathbf{A}(\theta)\mathbf{u}^{(0)} - \mathbf{b}(\theta)$ of an initial guess $\mathbf{u}^{(0)}$. This approximation is optimal in the sense that it minimizes $\|\mathbf{v} - \mathbf{u}(\theta)\|_{\mathbf{A}(\theta)}$ over the Krylov subspace, see [138]. We denote by $J(\theta)$ the smallest number j of iterations such that $\|\mathbf{A}(\theta)\mathbf{u}^{(j)}(\theta) - \mathbf{b}(\theta)\|_2 < \epsilon\|\mathbf{b}(\theta)\|_2$ for some $\epsilon > 0$ using $\mathbf{M}_\bullet(\theta)$. In other words, $J(\theta)$ is the number of necessary PCG iterations to reach a backward error of ϵ. Since $\mathbf{A}(\theta)$ is SPD, $J(\theta) \leq n$ as long as $\mathbf{u}^{(j)}(\theta)$ is computed with exact arithmetic. Meanwhile, a good preconditioner is such that $J(\theta) \ll n$ even when relying on finite arithmetic.

The second strategy consists of considering a single constant matrix $\mathbf{A}_0$ based on which the inverse of the preconditioner $\mathbf{M}_{0,\bullet}$ denominated by $\bullet$, or simply the data structures necessary for the application of its inverse $\mathbf{M}_{0,\bullet}^{-1}$ is assembled. Then, this preconditioner is applied, in turn, to all the realizations $\mathbf{A}(\theta)$. That is, the properties of the investigated preconditioners should focus on the spectrum of $\mathbf{M}_{0,\bullet}^{-1}\mathbf{A}(\theta)$ in which the preconditioner is constant but $\mathbf{A}(\theta)$ can be any matrix in the manifold of operators. Then, we consider the

unique approximation $\mathbf{u}^{(j)}(\theta)$ of $\mathbf{u}(\theta)$ which satisfies

$$\mathbf{u}^{(j)}(\theta) - \mathbf{u}^{(0)} \in \mathcal{K}^{(j)}(\mathbf{M}_{0,\bullet}^{-1}\mathbf{A}(\theta), \mathbf{M}_{0,\bullet}^{-1}\mathbf{r}^{(0)}(\theta)) \tag{1.191}$$

$$\mathbf{r}^{(j)}(\theta) \perp \mathcal{K}^{(j)}(\mathbf{M}_{0,\bullet}^{-1}\mathbf{A}(\theta), \mathbf{M}_{0,\bullet}^{-1}\mathbf{r}^{(0)}(\theta)). \tag{1.192}$$

We again denote by $J(\theta)$ the number of necessary PCG iterations to reach a backward error of ϵ.

1.7 Numerical illustration

First, we present numerical results for computation of a DD-KL expansion of a Gaussian process. Then, we show results about the iterative linear solves of the linear systems resulting from the FEM discretization of a 2D isotropic Poisson's equation. Finally, we present similar results for linear systems resulting from the FEM discretization of a 2D anisotropic Poisson's equation.

1.7.1 Simulation of Gaussian processes

The parallel subroutines implemented for the DD-KL expansion are tested in the Julia script `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/Example05_KarhunenLoevePllDomainDecomposition.jl`. The code is run on a cluster of 4 Linux desktop computers which were specifically assembled for this thesis. Each node of the cluster has an alias with the following specifications:

- `"hector0"`: Intel i7-4790S CPU running at 3.20 GHz with 16 Go of RAM
- `"hector1"`: Intel i7-6700 CPU running at 3.40 GHz with 16 Go of RAM
- `"hector2"`: Intel i7-4770 CPU running at 3.40 GHz with 16 Go of RAM
- `"hector4"`: Intel i7-6700 CPU running at 3.40 GHz with 16 Go of RAM

Following the naming convention of the Julia language, 8 workers are launched on each node. The number of subdomains, referred to as `ndom`, is set to 200 for a triangular mesh generated with approximately 100,000 mesh vertices. The resulting mesh contains 200,332 elements. Only the 20 most dominant eigen-pairs are computed for each local generalized eigenvalue problem. The stochastic process represented is Gaussian with unit variance and a squared exponential covariance defined over the domain $\Omega = [0, 1]^2$. That is, the field $\kappa(x)$ simulated is such that $\rho(x, y) := \mathbb{V}[\kappa(x)\kappa(y)] = \exp(-\|x - y\|^2/L^2)$ with a characteristic length L set to 0.1.

The 2D triangular mesh is obtained with a Julia wrapper of the library Triangle [145], and the mesh partition obtained using METIS [74] is displayed with color codes in Fig. 1.1 for 5, 10, 20, 30, 80 and 200 subdomains. An example of realization of the underlying Gaussian process is shown in Fig. 1.2 for the same different numbers of subdomains. The DD-KL approach exploits the fact that the convergence behavior of the KL expansion, for fixed covariance structure, is governed by the magnitude of L relative to the characteristic length of the domain. Indeed, decreasing (resp. increasing) the characteristic extent of the domain has a similar effect as increasing (resp. decreasing) the correlation length. In

the DD-KL approach, increasing the number of subdomains allows to reduce the extent of the subdomains, leading to an apparent larger L and a faster spectral decay for the local expansions. This is illustrated in Fig. 1.3, where the spectra of local expansions are shown for different values of the number of subdomains. Note that the Figure reports the spectra for all the n_d subdomains, so there are n_d spectra plotted when Ω is partitioned into n_d subdomains. Here, we observe that for a given number n_d of subdomains, the local subdomains all have roughly the same extent so the local expansions have a similar decay. Moreover, it is seen that, as expected, the local expansions have spectra which decay faster as n_d increases. The variability of the spectra among the n_d subdomains is principally due to the partitioning procedure that generates non-identical subdomains, with slightly variable apparent L as a result.

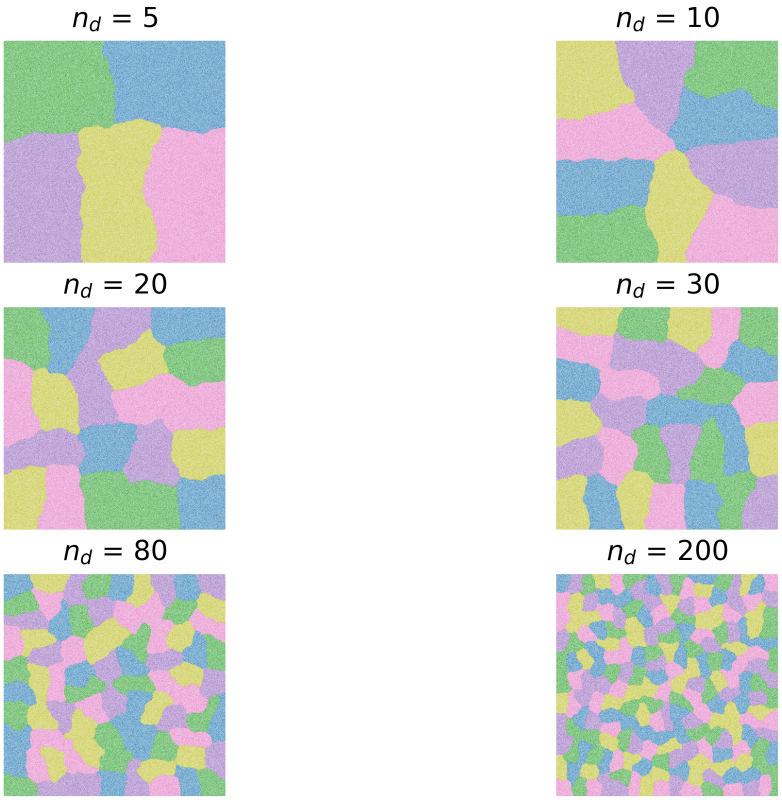

Figure 1.1: Partition of a triangular 2D mesh with 200,332 elements into 5, 10, 20, 30, 80 and 200 subdomains.

1.7.2 Solving the isotropic Poisson's equation

The coefficient field $\kappa(x)$ of Eq. (1.2), which is such that $\log \kappa(x)$ is a Gaussian process with zero mean, has a covariance $\rho(x, y) := \mathbb{V}[\kappa(x)\kappa(y)] = \sigma^2 \exp(-\|x - y\|^2/L^2)$. Eq. (1.2) is discretized with triangular finite elements for each realization θ. We consider the domain

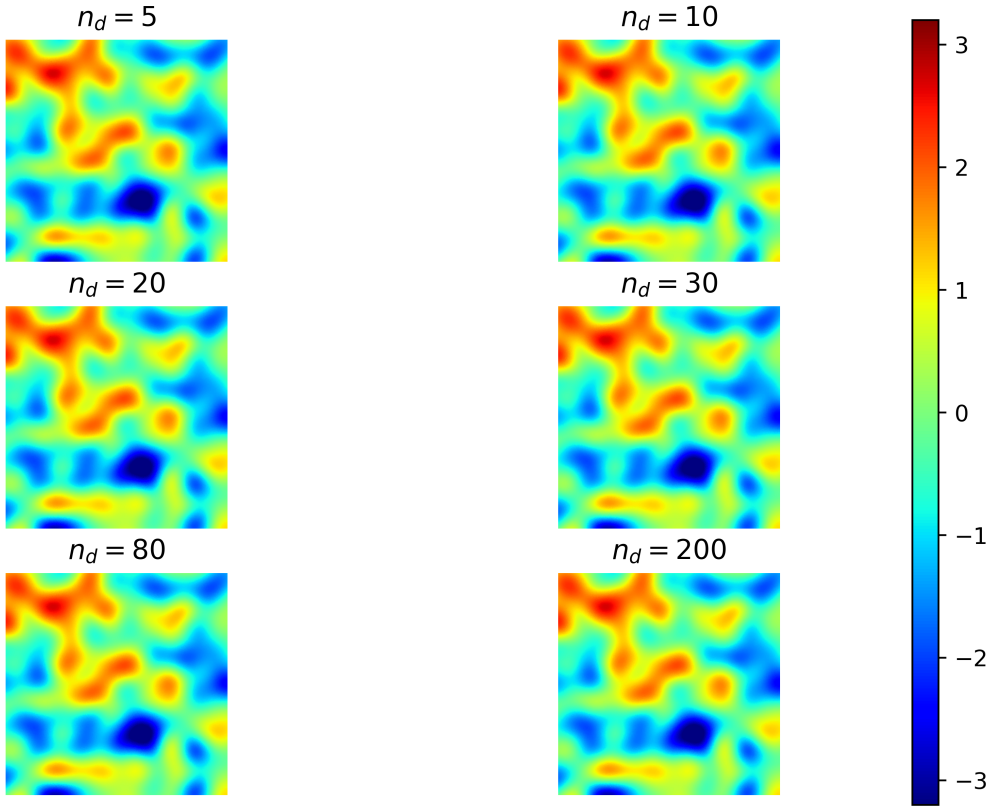

Figure 1.2: Realization of a Gaussian process $\kappa(x)$ with zero mean and covariance $\rho(x, y) := \mathbb{V}[\kappa(x)\kappa(y)] = \exp(-\|x - y\|^2/0.1^2)$ obtained by KL-DD with 5, 10, 20, 30, 80 and 200 subdomains.

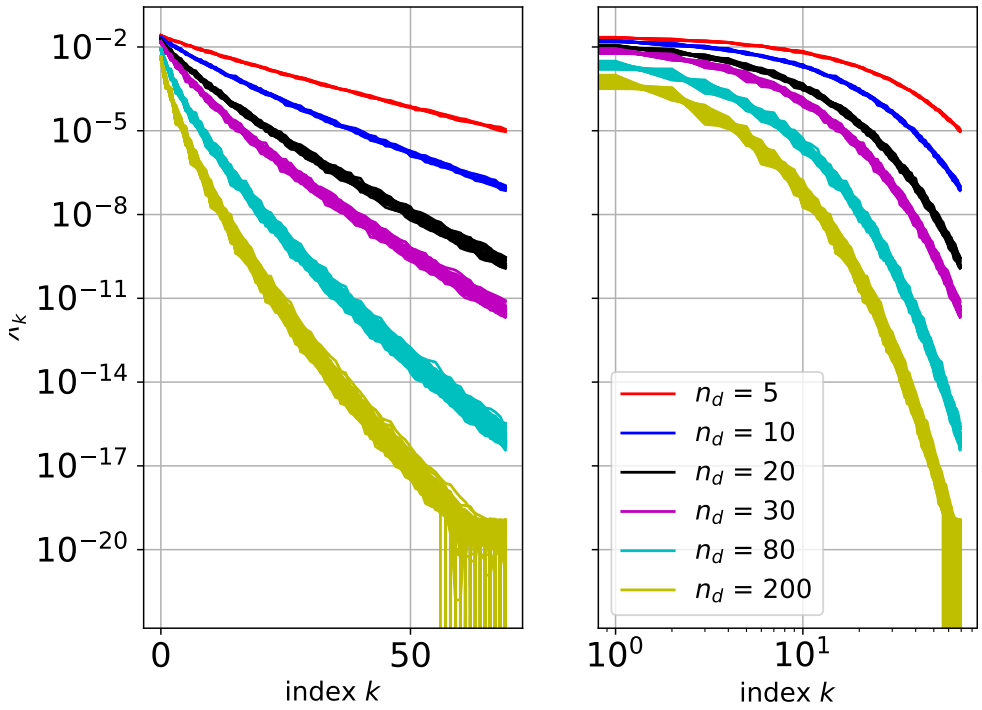

Figure 1.3: Spectra of local decompositions for a Gaussian process $\kappa(x)$ with zero mean and covariance $\rho(x, y) := \mathbb{V}[\kappa(x)\kappa(y)] = \exp(-\|x - y\|^2/0.1^2)$ obtained with 5, 10, 20, 30, 80 and 200 subdomains.

$\Omega = [0,1]^2$ along with a unit forcing, i.e., $f(x) := 1$ and a zero displacement at the boundary $\partial\Omega$, i.e., $g(x) := 0$. An equation of the form of Eq. (1.5) is then obtained for each θ. First, some of the subroutines implemented for the preconditioner based on non-overlapping domain decomposition are used in the Julia script `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/Example06_PcgStochasticEllipticPde.jl`. The number of subdomains, referred to as `ndom`, is set to 200 for a triangular mesh generated with approximately 100,000 mesh vertices. One can decide to assemble, or not, the local Schur complement operators through the specification of the boolean variable `do_assembly_of_local_schurs`. The preconditioner defined in this script is a constant LORASC preconditioner with $\varepsilon = 0.01$, which is used to solve the linear systems of several realizations. Note that AMG preconditioners are defined to solve for the local interior problems.

In light of the representation obtained by DD-KL for the coefficient field, the realization-dependent matrix $\mathbf{A}(\theta)$ is re-parameterized in terms of the stochastic coordinates of the DD-KL expansion. That is, we write $\mathbf{A}(\theta) = \mathbf{A}(\boldsymbol{\xi})$ where $\boldsymbol{\xi} := [\xi_1(\theta), \ldots, \xi_{n_{\mathrm{KL}}}(\theta)]$. As a means to compare the two preconditioning strategies presented in Section 1.1, we compute the spectra of $\mathbf{M}_\bullet^{-1}(\boldsymbol{\xi})\mathbf{A}(\boldsymbol{\xi})$ and $\mathbf{M}_{0,\bullet}^{-1}\mathbf{A}(\boldsymbol{\xi})$ for different preconditioners, as well as $\mathbf{M}_\bullet^{-1}(\boldsymbol{\xi})\mathbf{S}(\boldsymbol{\xi})$ and $\mathbf{M}_{0,\bullet}^{-1}\mathbf{S}(\boldsymbol{\xi})$ where $\mathbf{S}$ denotes the Schur complement of $\mathbf{A}$ for a given domain decomposition. For the strategy with a constant preconditioner, we assume that $\mathbf{M}_{0,\bullet}$ is constructed for the coefficient field $\kappa(x, \mathbf{0})$. That is $\mathbf{M}_{0,\bullet} = \mathbf{M}_\bullet(\mathbf{0})$. For each preconditioner, we draw three spectra for three distinct realizations of $\kappa(x, \boldsymbol{\xi})$ The resulting spectra are presented in Figs. 1.4 through 1.9 for a characteristic length $L = 0.1$ and a variance $\sigma^2 = 1$. For the preconditioners of $\mathbf{A}(\boldsymbol{\xi})$, we present results for bJ preconditioners with 5, 10, 20, 30, 80 and 200 blocks, for the LORASC preconditioner with 5, 10, 20, 30, 80 and 200 subdomains, as well as for the AMG preconditioner. For the preconditioners of $\mathbf{S}(\boldsymbol{\xi})$, we present results for the $\mathbf{A}_{\Gamma\Gamma}$ preconditioner with 5, 10, 20, 30, 80 and 200 subdomains as well as for the Neumann-Neumann preconditioner with the same numbers of subdomains. Irrespectively of the type $\bullet$ of preconditioner used, the realization-dependent ideal preconditioning approach, i.e., $\mathbf{M}_\bullet(\boldsymbol{\xi})$ leads to significantly denser concentrations of eigenvalues around one compared to the spectra of the constant preconditioning approach, i.e., $\mathbf{M}_\bullet(\mathbf{0})$. Irrespective of the number n_b of blocks, the bJ preconditioner leads to spectra with a trail of well-separated small eigenvalues whereas the remaining of the preconditioned spectra are more densely packed. These trails of well-separated small eigenvalues are also observed in the spectra of the LORASC preconditioners. The lower bounds of the LORASC spectra decrease as the number of subdomains is increased. We can see that using $\varepsilon = 0.01$ leads to slightly denser spectra than using $\varepsilon = 0$. Preconditioning the Schur complement with $\mathbf{A}_{\Gamma\Gamma}$ also leads to trails of well-separated small eigenvalues. Note that the constant preconditioning approach also leads to some separation of the largest eigenvalues. When preconditioning the Schur complement with a Neumann-Neumann preconditioner, we see some separation of the largest eigenvalues for both the constant and the realization-dependent preconditioning strategies. On the other hand, the number of eigenvalues in the trail of well-separated smallest eigenvalues increases with the number of subdomains. Overall, the bJ preconditioner leads to the widest spectra, this is followed by the LORASC preconditioner while AMG leads to relatively dense spectra. The investigation of the preconditioned spectra provides a hint of the expected behavior of the PCG solves of the simulated linear systems. The expected number of PCG itera-

tions are reported in Table 1.1 for preconditioners with different numbers of blocks and subdomains. These results, obtained for linear systems with 4,000 DoFs, are also plotted in Fig. 1.10. The expected numbers of PCG iterations are drawn with solid lines for the constant preconditioning strategy and with dashed lines for the realization-dependent ideal preconditioning strategy. As such, these lines provide bounds to the preconditioning strategies developed in the following chapters. The lower bound constitutes an ideal limit only achievable by the unfeasible re-definition of the preconditioner at every realization, and the upper bound constitutes a limit obtained when implementing the most straightforward preconditioning strategy. The AMG preconditioner leads to the smallest expected number of PCG iterations followed by LORASC preconditioners and the bJ preconditioner. The number of expected PCG iterations obtained with the bJ preconditioner increases with the number of blocks and, similarly, this number increases when increasing the number of subdomains for the LORASC preconditioners. The rates of these scalings do not seem to depend on the preconditioning strategy. Similar results are provided in Table 1.2 for the PCG solves of the Schur complement linear systems. These results, also obtained for linear systems with 4,000 DoFs, are plotted in Fig. 1.11. While the scaling of the number of expected PCG iterations with respect to the number of subdomains follows a constant rate when preconditioning with $\mathbf{A}_{\Gamma\Gamma}$, it is not the case when preconditioning with the Neumann-Neumann preconditioner. Hence, the number of expected PCG iterations is smaller when using the Neumann-Neumann preconditioner with small numbers of subdomains, and larger than using $\mathbf{A}_{\Gamma\Gamma}$ for larger numbers of subdomains. The rate of this scaling does not depend on the preconditioning strategy. The expected number of PCG iterations are also reported in Table 1.3 for linear systems with different numbers of DoFs. These results, obtained for preconditioners with 200 blocks/subdomains, are also plotted in Fig. 1.12. Once again, the expected numbers of PCG iterations are drawn with solid lines for the constant preconditioning strategy and with dashed lines for the realization-dependent ideal preconditioning strategy. We can see that the AMG preconditioner scales the least with respect to the size of the linear system. For sufficiently large systems, using a constant AMG preconditioner leads to smaller expected PCG iteration numbers than using a realization-dependent ideal preconditioner based on domain decomposition. The bJ preconditioner scales more with the number of DoFs than the LORASC preconditioner. In Table 1.4, we report the expected number of PCG iterations for the Schur complement with respect to different numbers of DoFs. These results, obtained for preconditioners with 200 subdomains, are also plotted in Fig. 1.13. We can see that $\mathbf{A}_{\Gamma\Gamma}$ scales more than the Neumann-Neumann preconditioner with respect to the number of DoFs. For small linear systems, $\mathbf{A}_{\Gamma\Gamma}$ performs better whereas, for larger systems, the Neumann-Neumann preconditioner will eventually lead to smaller numbers of expected PCG iterations.

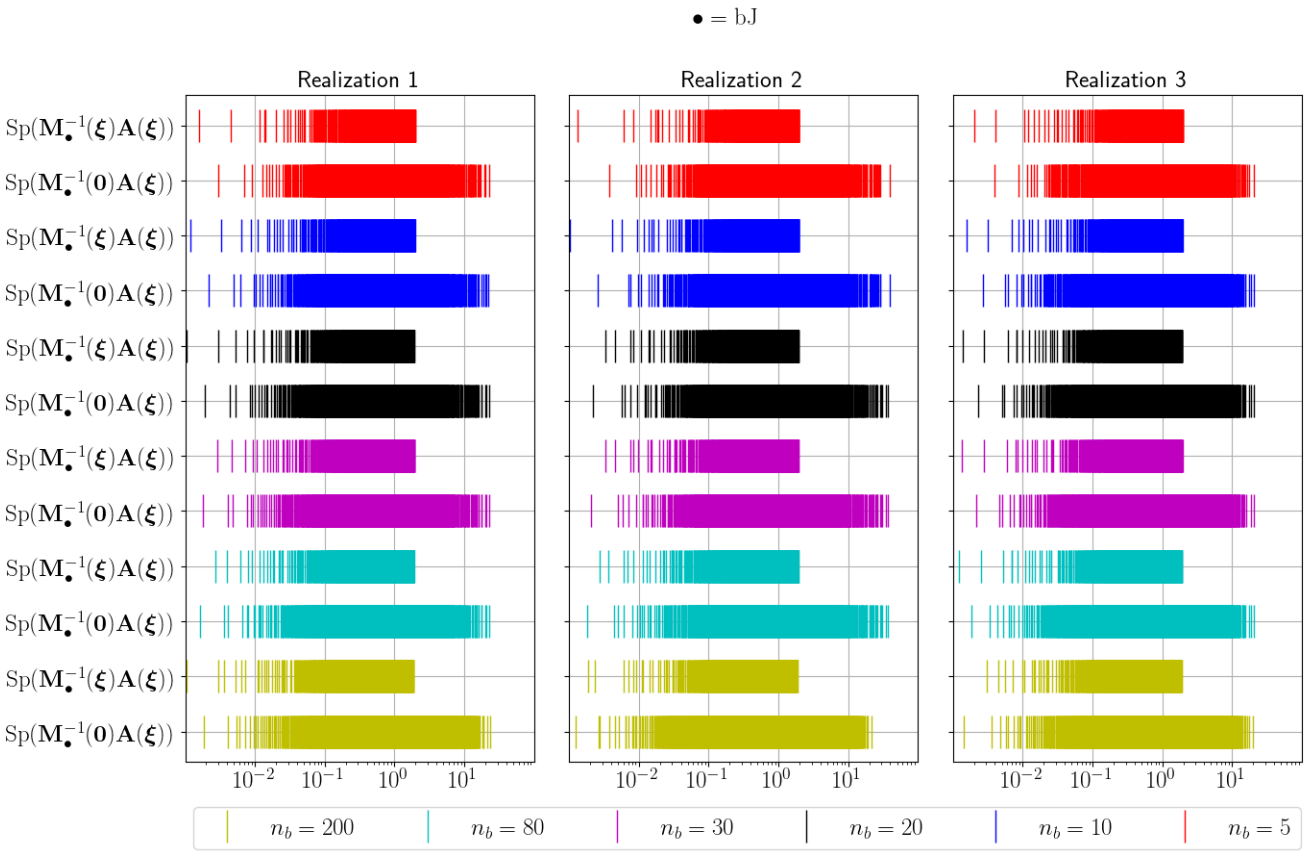

Figure 1.4: Spectra preconditioned by bJ preconditioners, $\sigma^2 = 1$ and $L = 0.1$.

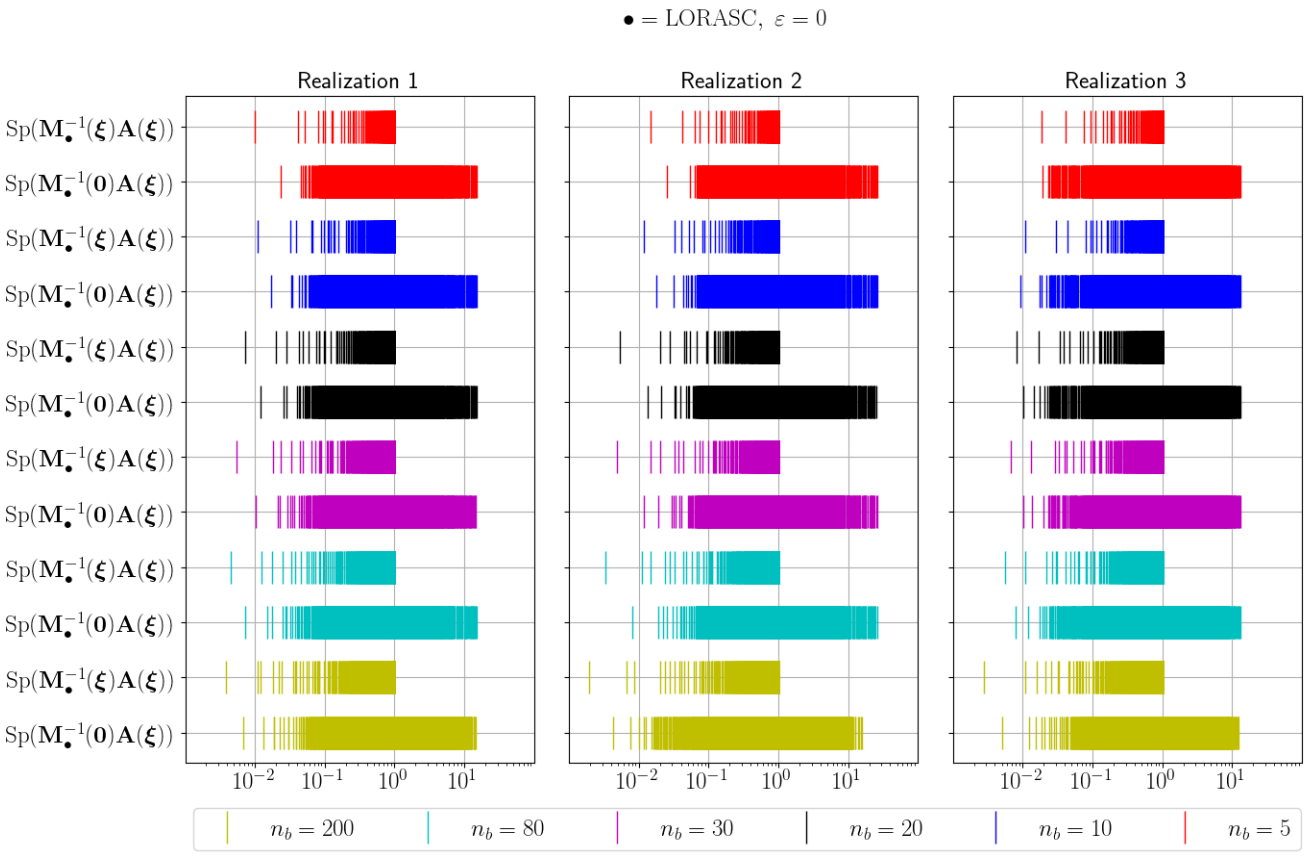

Figure 1.5: Spectra preconditioned with LORASC preconditioners for $\varepsilon = 0$, $\sigma^2 = 1$ and $L = 0.1$.

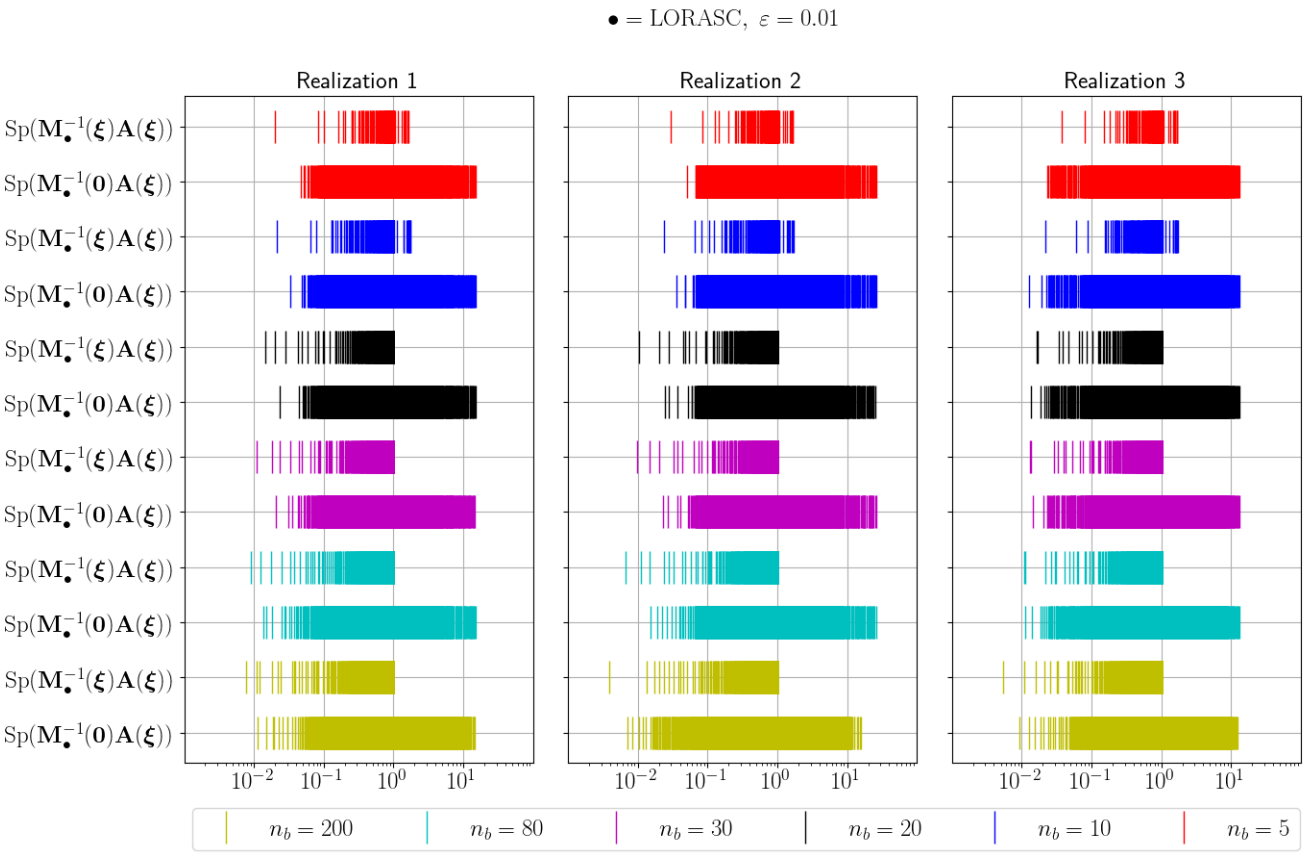

Figure 1.6: Spectra preconditioned with LORASC preconditioners for $\varepsilon = 0.1$, $\sigma^2 = 1$ and $L = 0.1$.

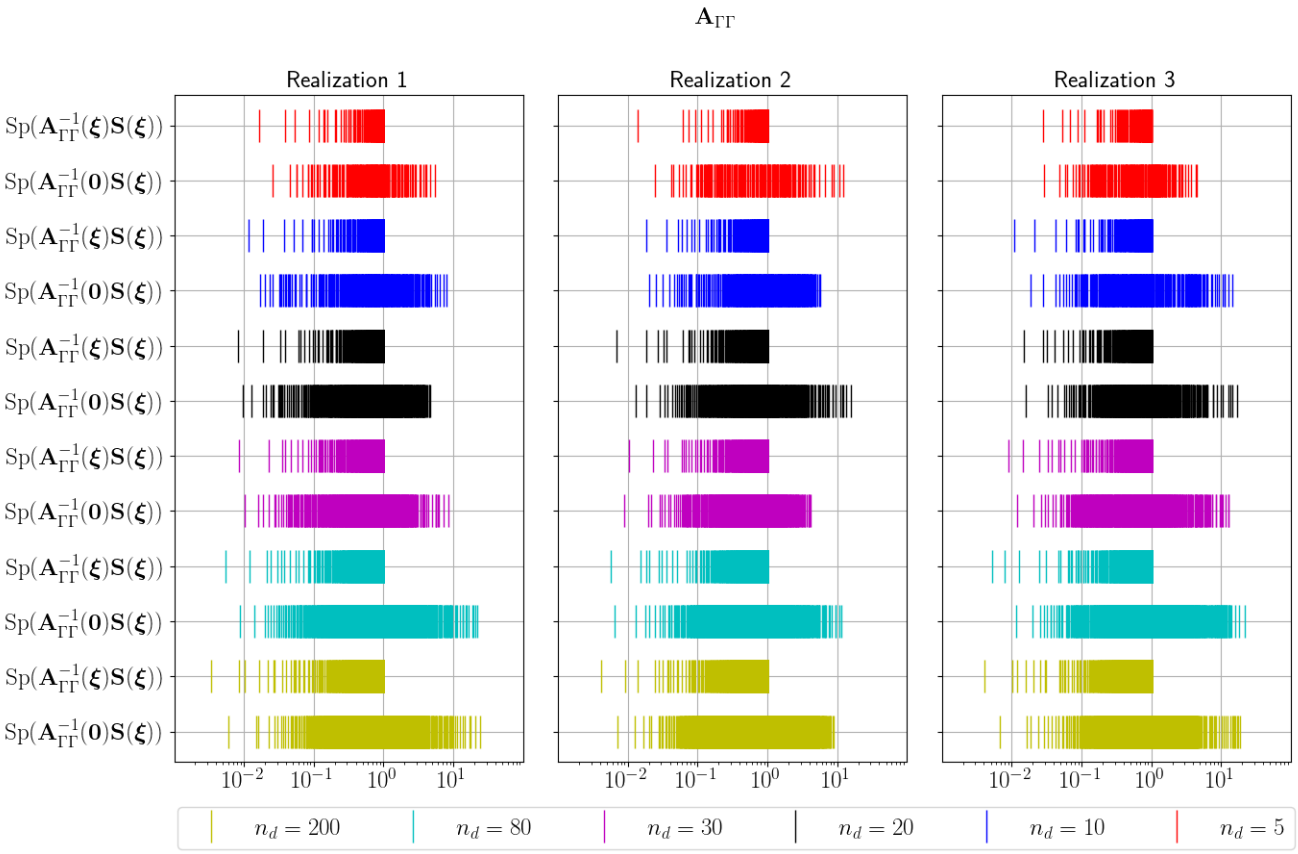

Figure 1.7: Schur spectra preconditioned with $\mathbf{A}_{\Gamma\Gamma}$, $\sigma^2 = 1$ and $L = 0.1$.

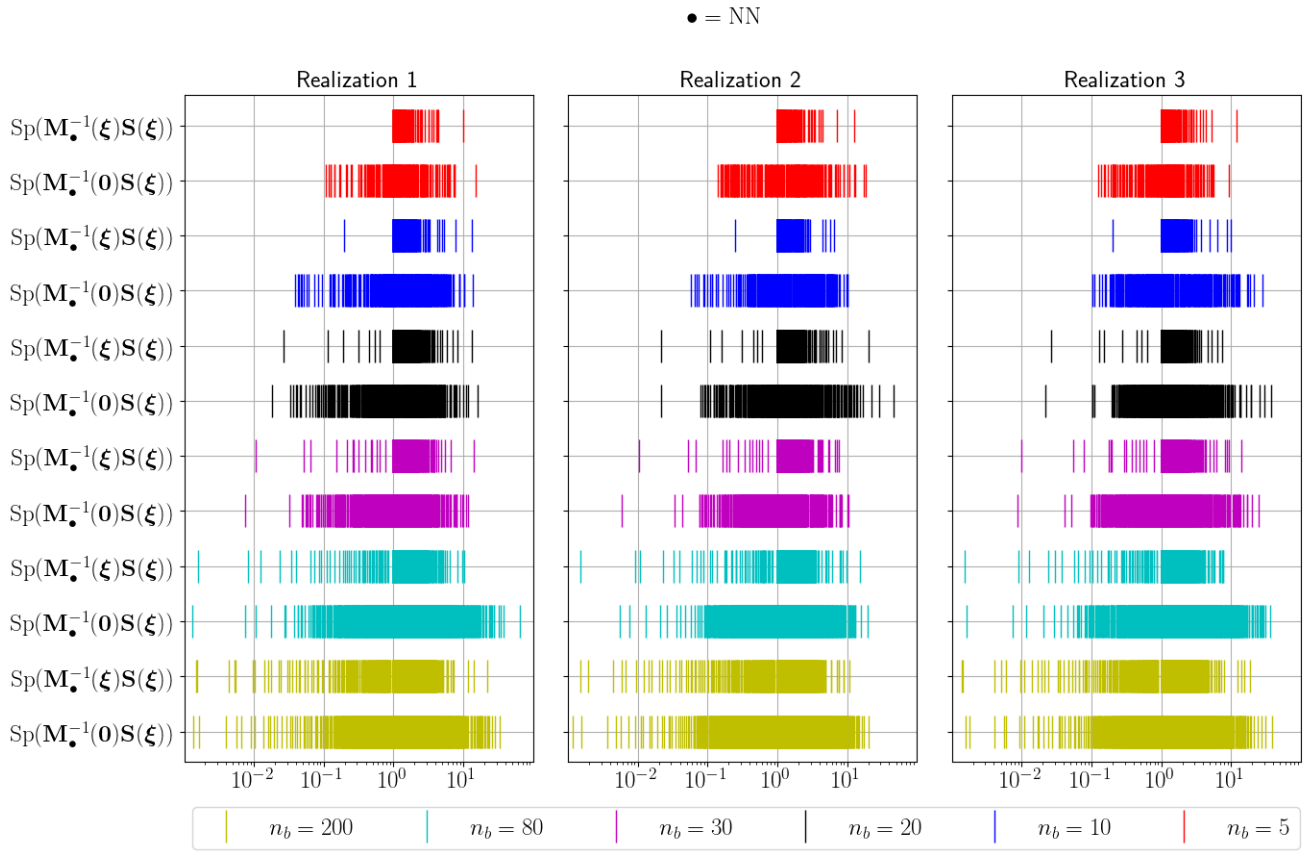

Figure 1.8: Schur spectra preconditioned with Neumann-Neumann preconditioner, $\sigma^2 = 1$ and $L = 0.1$.

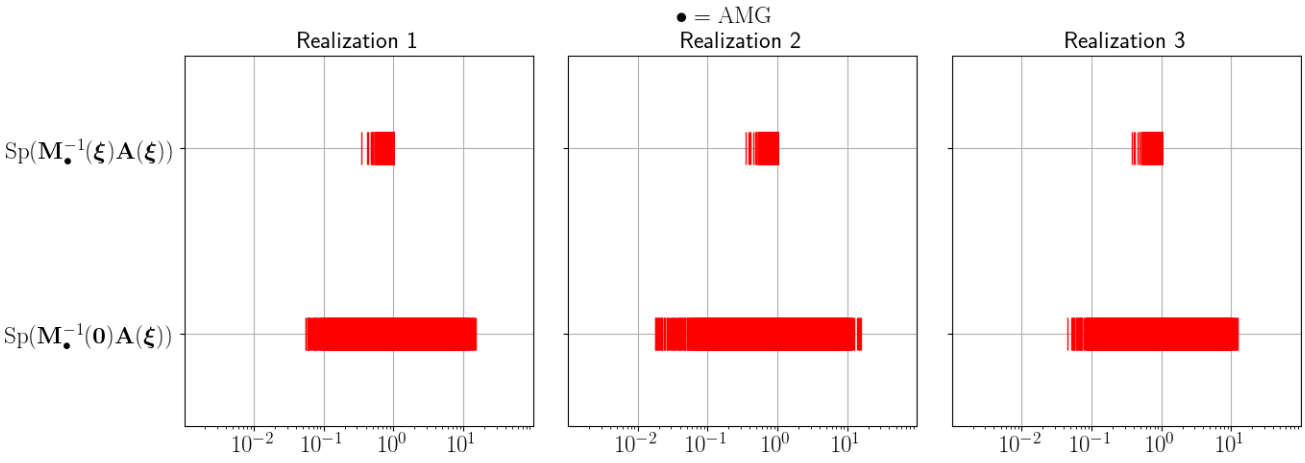

Figure 1.9: Preconditioned spectra with AMG preconditioner, $\sigma^2 = 1$ and $L = 0.1$.

Preconditioner	n					
	5	10	20	30	80	200
$\mathbf{M}_{\text{bJ}}(\mathbf{0}),\ n_b = n$	337.44	404.22	436.03	451.01	484.06	516.66
$\mathbf{M}_{\text{bJ}}(\boldsymbol{\xi}),\ n_b = n$	120.96	145.37	155.67	161.46	172.11	182.07
$\mathbf{M}_{\text{LORASC}}(\mathbf{0}),\ n_d = n,\ \varepsilon = 0$	144.56	162.83	183.44	192.70	221.46	253.74
$\mathbf{M}_{\text{LORASC}}(\boldsymbol{\xi}),\ n_d = n,\ \varepsilon = 0$	32.91	40.76	48.54	51.99	62.30	73.11
$\mathbf{M}_{\text{LORASC}}(\mathbf{0}),\ n_d = n,\ \varepsilon = 0.01$	120.32	133.37	148.85	155.45	211.23	242.07
$\mathbf{M}_{\text{LORASC}}(\boldsymbol{\xi}),\ n_d = n,\ \varepsilon = 0.01$	29.54	37.24	45.67	49.41	59.21	66.52
$\mathbf{M}_{\text{AMG}}(\mathbf{0})$	106.47	106.47	106.47	106.47	106.47	106.47
$\mathbf{M}_{\text{AMG}}(\boldsymbol{\xi})$	11.89	11.89	11.89	11.89	11.89	11.89

Table 1.1: Expected numbers of solver iterations for different preconditioners with respect to the number of blocks/subdomains.

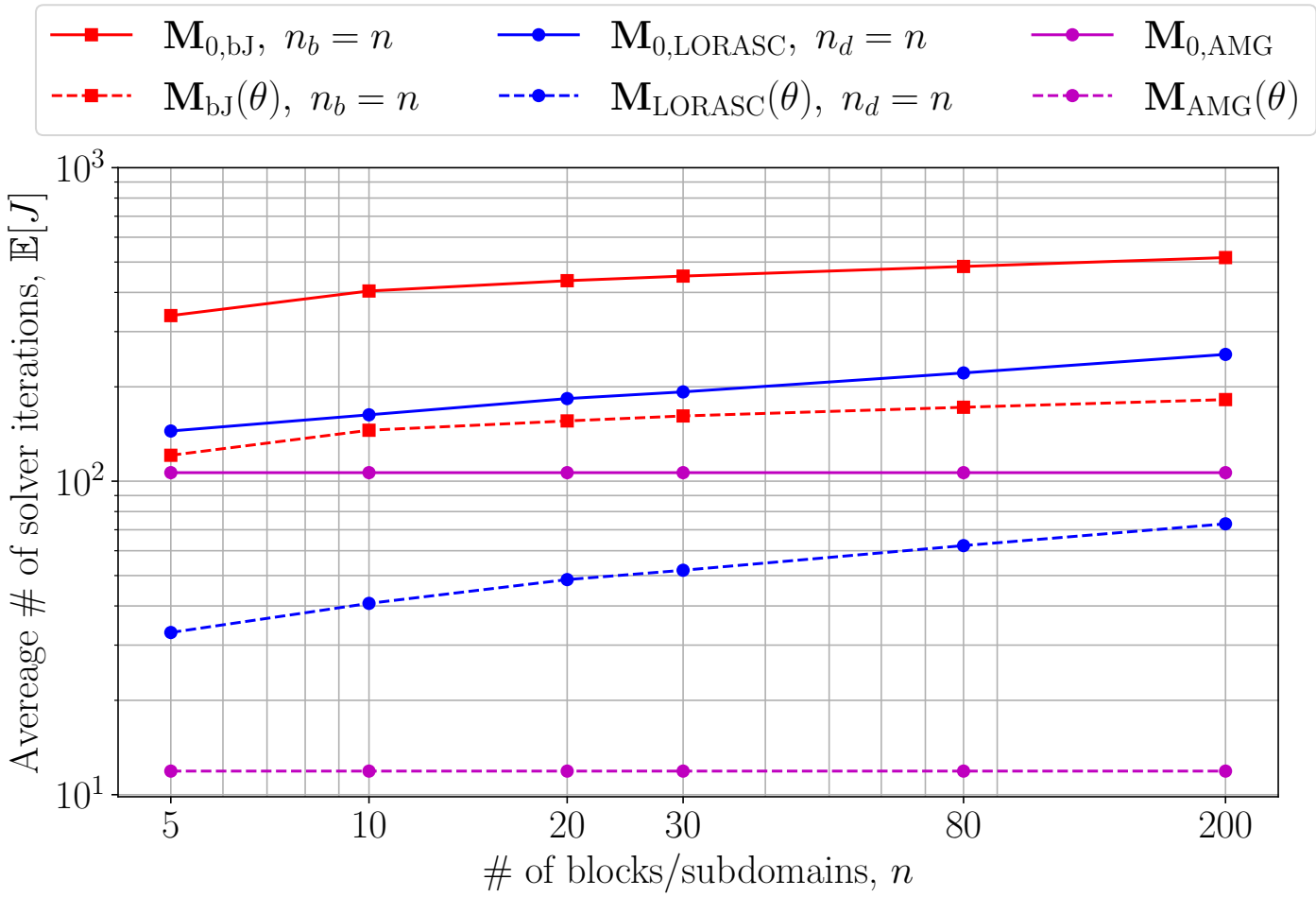

Figure 1.10: Scaling with respect to n_b (or n_d) of expected numbers of solver iterations to solve $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with different preconditioners and preconditioning strategies.

Preconditioner	n					
	5	10	20	30	80	200
$\mathbf{A}_{\Gamma\Gamma}(\mathbf{0}),\ n_b = n$	82.13	104.13	129.16	143.22	178.50	222.58
$\mathbf{A}_{\Gamma\Gamma}(\boldsymbol{\xi}),\ n_b = n$	35.12	42.75	50.63	53.93	63.99	74.42
$\mathbf{M}_{\mathrm{NN}}(\mathbf{0}),\ n_d = n$	55.84	70.76	100.84	129.83	273.14	684.26
$\mathbf{M}_{\mathrm{NN}}(\boldsymbol{\xi}),\ n_d = n$	17.30	21.29	42.00	63.75	154.66	352.99

Table 1.2: Expected numbers of solver iterations for different preconditioners with respect to the number of blocks/subdomains.

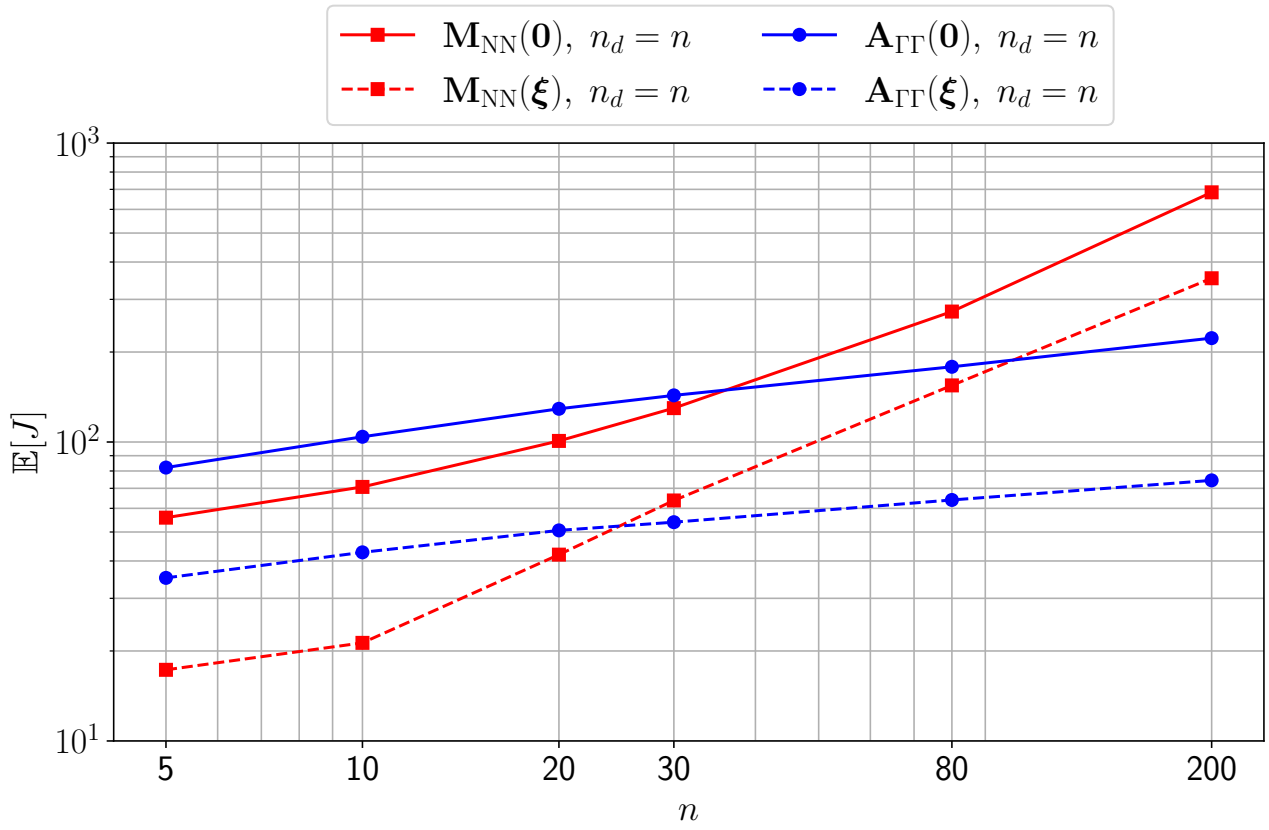

Figure 1.11: Scaling with respect to n_b (or n_d) of expected numbers of solver iterations to solve $\mathbf{S}(\theta)\mathbf{u}_\Gamma(\theta) = \mathbf{b}_{\mathrm{S}}(\theta)$ with different preconditioners and preconditioning strategies.

Preconditioner	DoFs					
	4,000	8,000	16,000	32,000	64,000	128,000
$\mathbf{M}_{\text{bJ}}(\mathbf{0}),\ n_b = 200$	516.66	724.15	1,067.48	1,461.31	2,212.16	2,953.65
$\mathbf{M}_{\text{bJ}}(\boldsymbol{\xi}),\ n_b = 200$	182.07	239.87	332.58	441.46	641.68	844.41
$\mathbf{M}_{\text{LORASC}}(\mathbf{0}),\ n_d = 200,\ \varepsilon = 0$	253.74	331.75	422.36	517.85	633.49	741.09
$\mathbf{M}_{\text{LORASC}}(\boldsymbol{\xi}),\ n_d = 200,\ \varepsilon = 0$	73.11	89.80	110.32	132.91	159.75	188.41
$\mathbf{M}_{\text{LORASC}}(\mathbf{0}),\ n_d = 200,\ \varepsilon = 0.01$	242.07	288.32	357.71	419.64	509.63	596.70
$\mathbf{M}_{\text{LORASC}}(\boldsymbol{\xi}),\ n_d = 200,\ \varepsilon = 0.01$	66.52	75.34	89.50	108.51	132.13	158.38
$\mathbf{M}_{\text{AMG}}(\mathbf{0})$	106.47	118.25	127.80	134.85	143.40	146.46
$\mathbf{M}_{\text{AMG}}(\boldsymbol{\xi})$	11.89	12.67	14.09	15.50	17.11	17.69

Table 1.3: Expected numbers of solver iterations for different preconditioners with respect to the number of DoFs.

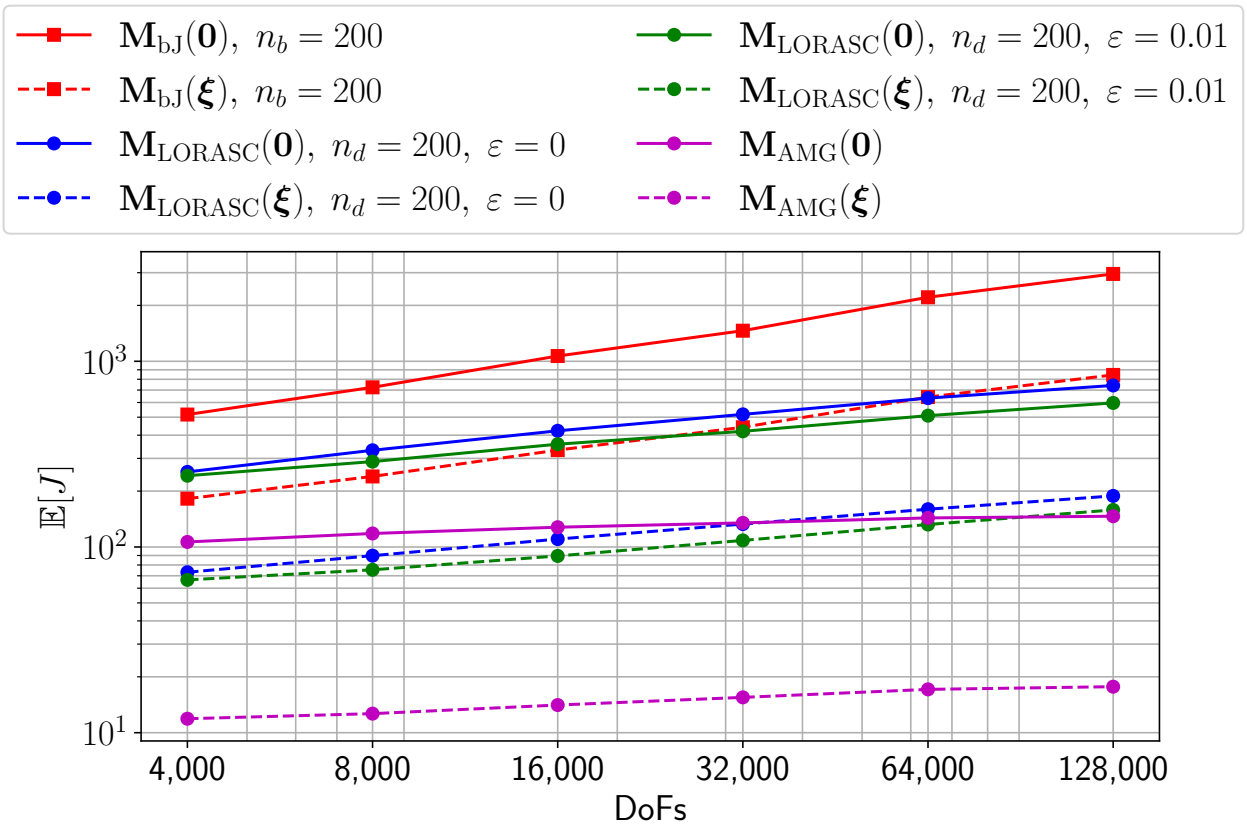

Figure 1.12: Scaling with respect to DoFs of expected numbers of solver iterations to solve $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with different preconditioners and preconditioning strategies.

Preconditioner	DoFs					
	4,000	8,000	16,000	32,000	64,000	128,000
$\mathbf{A}_{\Gamma\Gamma}(\mathbf{0}),\ n_b = n$	222.58	286.44	372.80	459.79	584.11	696.36
$\mathbf{A}_{\Gamma\Gamma}(\boldsymbol{\xi}),\ n_b = n$	74.42	92.94	113.82	137.44	166.70	198.86
$\mathbf{M}_{\mathrm{NN}}(\mathbf{0}),\ n_d = n$	684.26	764.64	886.60	948.77	1,058.06	1,110.16
$\mathbf{M}_{\mathrm{NN}}(\boldsymbol{\xi}),\ n_d = n$	352.99	379.36	415.75	432.53	504.23	528.39

Table 1.4: Expected numbers of solver iterations for different preconditioners with respect to the number of DoFs.

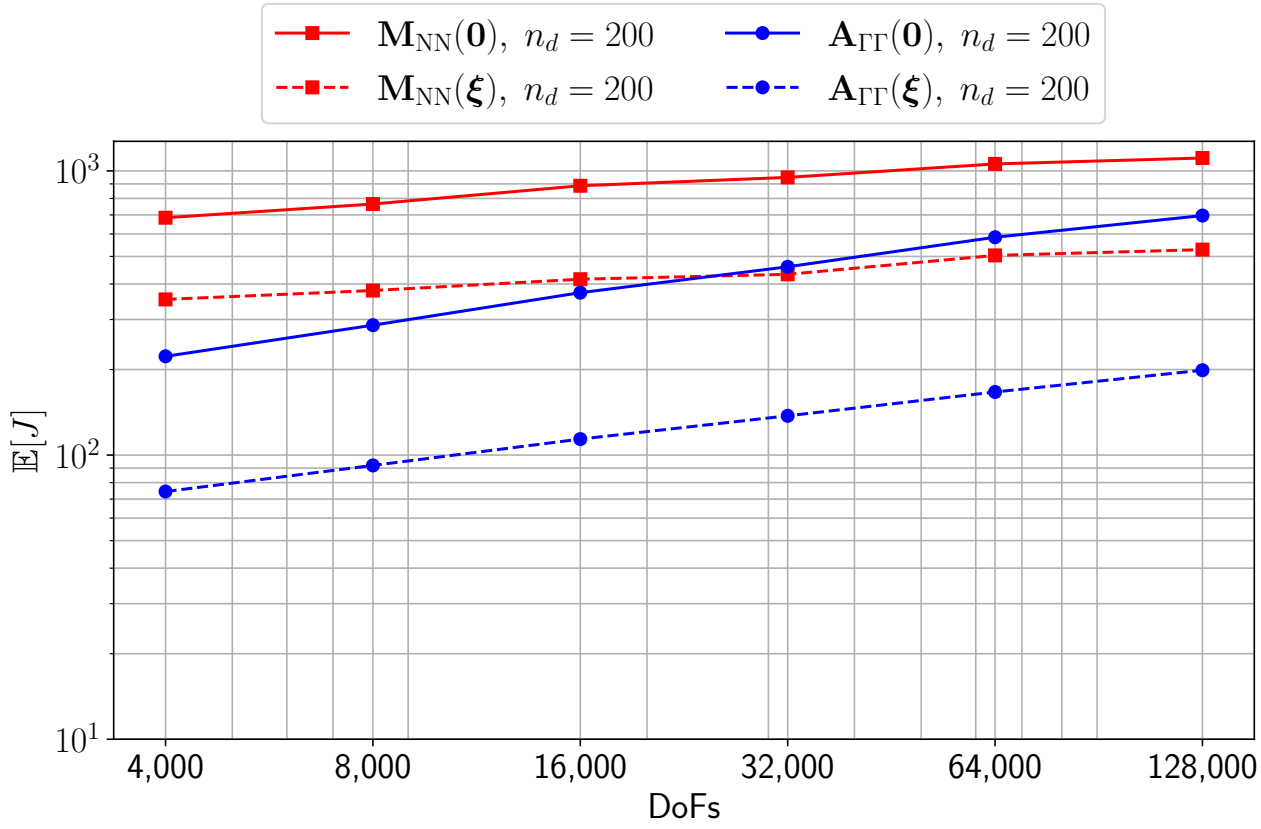

Figure 1.13: Scaling of expected numbers of solver iterations to solve $\mathbf{S}(\theta)\mathbf{u}_\Gamma(\theta) = \mathbf{b}_{\mathrm{S}}(\theta)$ with different preconditioners and preconditioning strategies.

Chapter 2

Deflation of linear systems

2.1 Introduction

We illustrate the application of deflation to potentially accelerate the iterative resolution of SPD linear systems. This description is inspired by the works of [55], [63] and [32]. In particular, the deflated conjugate gradient (CG) proposed by [107] and analyzed by [140] is presented as we investigate its application to preconditioned systems.

2.2 Deflated Krylov subspace methods

In case of a spatial discretization using finite elements, $\mathbf{A}$ is sparse, and its fast application is leveraged when searching for an approximation $\mathbf{u}^{(j)}$ of $\mathbf{u}$ in the affine subspace $\mathbf{u}^{(0)} + \mathcal{K}^{(j)}(\mathbf{A}, \mathbf{r}^{(0)})$, where

$$\mathcal{K}^{(j)}(\mathbf{A}, \mathbf{r}^{(0)}) := \text{Span}\{\mathbf{r}^{(0)}, \mathbf{A}\mathbf{r}^{(0)}, \cdots, \mathbf{A}^{j-1}\mathbf{r}^{(0)}\} \tag{2.1}$$

is the j-th Krylov subspace of $\mathbf{A}$ generated by $\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)}$ for a given initial guess $\mathbf{u}^{(0)}$. Then, the orthogonal projection obtained by letting $\mathbf{r}^{(j)} \perp \mathcal{K}^{(j)}(\mathbf{A}, \mathbf{r}^{(0)})$, leads to an optimal iterate in the sense that $\|\mathbf{u}^{(j)} - \mathbf{u}\|_{\mathbf{A}}^2 := (\mathbf{u}^{(j)} - \mathbf{u})^T\mathbf{A}(\mathbf{u}^{(j)} - \mathbf{u})$ is minimized by the approximation over the search space. In this work, $\mathbf{A}$ is SPD, and the sequence $\{\mathbf{u}^{(j)}\}_{j=1}^m$ of these optimal iterates is obtained by the CG algorithm, i.e., CG($\mathbf{A}$, $\mathbf{b}$, $\mathbf{u}^{(0)}$).

While the rate of convergence of CG($\mathbf{A}$, $\mathbf{b}$, $\mathbf{u}^{(0)}$) is governed by the distribution of

the eigenvalues of $\mathbf{A}$, it admits the following bound:

$$\|\mathbf{u}^{(m)} - \mathbf{u}\|_{\mathbf{A}} \leq 2\|\mathbf{u}^{(0)} - \mathbf{u}\|_{\mathbf{A}} \left(\frac{\sqrt{\text{cond}(\mathbf{A})} - 1}{\sqrt{\text{cond}(\mathbf{A})} + 1} \right)^m \tag{2.2}$$

where $\text{cond}(\mathbf{A}) = \lambda^{(1)}(\mathbf{A})/\lambda^{(n)}(\mathbf{A})$ is the condition number of $\mathbf{A}$ with eigenvalues $\lambda^{(1)}(\mathbf{A}) \geq \cdots \geq \lambda^{(n)}(\mathbf{A}) \geq 0$. While the tightness of this bound also depends on the distribution of the eigenvalues, Eq. (2.2) provides a way to understand the relation between the convergence behavior of CG and the eigenvalues at both ends of the spectrum $\text{Sp}(\mathbf{A})$. In particular, it is understood that an increase of $\lambda^{(n)}(\mathbf{A})$ (or a decrease of $\lambda^{(1)}(\mathbf{A})$) results in a decrease of the bound on the rate of convergence. Also, if the eigenvalue $\lambda^{(n)}(\mathbf{A})$ (resp. $\lambda^{(1)}(\mathbf{A})$) is moved towards the center of the spectrum past its closest neighbor, then the upper bound of Eq. (2.2) is scaled by $\lambda^{(n)}(\mathbf{A})/\lambda^{(n-1)}(\mathbf{A})$ (resp. $\lambda^{(2)}(\mathbf{A})/\lambda^{(1)}(\mathbf{A})$). Hence, well separated eigenvalues located at either end of the spectrum of $\mathbf{A}$ can significantly hinder convergence and, canceling, or at least, attenuating their effect may result in a substantially faster convergence [59].

If k eigenvectors of $\mathbf{A}$ are known, they can be used to force the CG procedure to work with subspaces which are convenient enough to enable a convergence to $\mathbf{u}$ at a rate bounded as in Eq. (2.2), but with a potentially smaller condition number. Let these k eigenvectors be stored by columns in $\mathbf{Y} \in \mathbb{R}^{n \times k}$, and paired with the eigenvalues in $\Lambda \subset \text{Sp}(\mathbf{A})$. We want $\{\mathbf{u}^{(j)}\}_{j=1}^m$ to be such that $\mathbf{r}^{(j)} \perp \mathcal{R}(\mathbf{Y})$, where $\mathcal{R}(\mathbf{Y})$ is the range of $\mathbf{Y}$, and $\mathbf{r}^{(j)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(j)}$ for all $0 \leq j \leq m$. Since $\mathcal{R}(\mathbf{Y})^\perp$ is invariant under the action of $\mathbf{A}$, this is achieved by setting $\mathbf{u}^{(0)}$ such that $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{Y})$ in CG($\mathbf{A}$, $\mathbf{b}$, $\mathbf{u}^{(0)}$), i.e., by letting $\mathbf{u}^{(0)} := \mathbf{Y}(\mathbf{Y}^T\mathbf{A}\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{b}$. Then, the sequence $\{\mathbf{u}^{(j)}\}_{j=1}^m$ converges to $\mathbf{u}$ at a rate bounded by Eq. (2.2), but where the condition number is now given by $\text{cond} = \max\{\text{Sp}(\mathbf{A}) \setminus \Lambda\} / \min\{\text{Sp}(\mathbf{A}) \setminus \Lambda\}$. Essentially, the effect of the eigenvalues in Λ is canceled at the expense of an initial $\mathbf{A}$-orthogonal projection of the solution $\mathbf{u}$ onto $\mathcal{R}(\mathbf{Y})$, which does not require much computation as long as $k \ll n$. The resulting procedure, referred to as Init-CG, was introduced in [43] motivated by [123, 137, 156], and can lead to significantly improved convergence behaviors if a small number of well separated eigenpairs are properly selected at the end(s) of the spectrum.

In practice, $\mathbf{Y}$ is not known. Instead, it is possible to construct $\mathbf{W} \in \mathbb{R}^{n \times k}$, where $\mathcal{R}(\mathbf{W})$ somehow approximates the subspace associated with the eigenvalues of interest. If this approximation is good enough, using $\mathbf{W}$ in place of $\mathbf{Y}$ in Init-CG may yield improved convergence behaviors similar to when using $\mathbf{Y}$, depending on the target accuracy [55]. As the quality of this approximation deteriorates, the $\mathbf{A}$-invariance of $\mathcal{R}(\mathbf{W})$ is no longer guaranteed, causing the residuals $\mathbf{r}^{(j)}$ generated by Init-CG to lose their orthogonality with respect to $\mathcal{R}(\mathbf{W})$, and the original behavior of CG is recovered. Deflation can be used as a means to circumvent the effects of this loss of invariance, by forcing the residuals to remain orthogonal to the subspace.

Deflation consists of splitting the approximation space into two complementary subspaces with a projector such that the projected linear system, referred to as the deflated system, is more amenable to iterative solving than the original system. In what follows, we illustrate the application of deflation to potentially accelerate the iterative resolution of sequences of SPD linear systems with multiple operators. This description is inspired by the works of [55], [63] and [32]. In particular, the deflated conjugate gradient (CG)

proposed by [107] and analyzed by [140] is presented as we investigate its application to preconditioned systems.

Let $\mathbf{A}$ be an SPD matrix in $\mathbb{R}^{n\times n}$. For some $\mathbf{b}\in\mathbb{R}^n$, we let $\mathbf{u}^* := \mathbf{A}^{-1}\mathbf{b}$ of which we intend to accelerate the iterative resolution governed by the distribution of the eigenvalues of $\mathbf{A}$. One way to do so, deflation, consists of constructing a new, semi-definite operator $\mathbf{A}'$ whose strictly positive spectrum is *deflated* of some of the eigenvalues of $\mathbf{A}$ which would hinder the convergence of an iterative resolution of $\mathbf{A}\mathbf{u}^* = \mathbf{b}$. To do so, it is customary to introduce a projection operator, a projector, whose application decomposes the solution of the linear system into two contributions lying in different subspaces. In particular, we consider $\mathbf{\Pi}\in\mathbb{R}^{n\times n}$ defined by

$$\mathbf{\Pi} := \mathbf{I}_n - \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T \tag{2.3}$$

for some $\mathbf{W}\in\mathbb{R}^{n\times k}$ whose columns form the basis of a subspace of dimension $k<n$. Since $\mathbf{W}$ is full column rank and $\mathbf{A}$ is SPD, so is $\mathbf{W}^T\mathbf{A}\mathbf{W}$ which is hence invertible. The following holds:

Proposition 1.

1.1 $\mathbf{\Pi}$ *is idempotent, i.e.,* $\mathbf{\Pi}^2=\mathbf{\Pi}$.
Proof:
$$\begin{aligned}\mathbf{\Pi}^2 &= \mathbf{I}_n + \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T\\ &\quad -2\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T\\ \mathbf{\Pi}^2 &= \mathbf{I}_n + \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T\\ &\quad -2\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T\\ \mathbf{\Pi}^2 &= \mathbf{I}_n + \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T - 2\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T\\ \mathbf{\Pi}^2 &= \mathbf{I}_n - \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T = \mathbf{\Pi}.\end{aligned}$$

1.2 $\mathbf{A}\mathbf{\Pi}$ *is symmetric.*
Proof:
$$\begin{aligned}(\mathbf{A}\mathbf{\Pi})^T &= (\mathbf{A} - \mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T)^T\\ &= \mathbf{A} - \mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T = \mathbf{A}\mathbf{\Pi}.\end{aligned}$$

1.3 $\mathbf{W}^T\mathbf{A}\mathbf{\Pi}=0$.
Proof:
$$\begin{aligned}\mathbf{W}^T\mathbf{A}\mathbf{\Pi} &= \mathbf{W}^T\mathbf{A} - \mathbf{W}^T\mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T\\ &= \mathbf{W}^T\mathbf{A} - \mathbf{W}^T\mathbf{A} = 0.\end{aligned}$$

1.4 $\mathbf{\Pi}\mathbf{W}=0$.
Proof:
$$\begin{aligned}\mathbf{\Pi}\mathbf{W} &= \mathbf{W} - \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}(\mathbf{A}\mathbf{W})^T\mathbf{W} = \mathbf{W} - \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A}\mathbf{W}\\ &= \mathbf{W} - \mathbf{W} = 0.\end{aligned}$$

1.5 $\mathcal{K}(\mathbf{\Pi})$ *is* $\mathbf{A}$*-invariant, i.e.,* $\mathcal{K}(\mathbf{A}\mathbf{\Pi})\subseteq\mathcal{K}(\mathbf{\Pi})$.
Proof: For all $\mathbf{u}\in\mathcal{K}(\mathbf{A}\mathbf{\Pi})$, we have $\mathbf{A}\mathbf{\Pi}\mathbf{u}=0$ which, because $\mathbf{A}$ is SPD, implies $\mathbf{\Pi}\mathbf{u}=0$ and $\mathbf{u}\in\mathcal{K}(\mathbf{\Pi})$.

The following is deduced from Prop. 1:

Corollary 1.

1.1 $\mathbf{\Pi}^T\mathbf{A}\mathbf{\Pi} = \mathbf{\Pi}^T\mathbf{A}$.
Proof: Props. 1.1 and 1.2 imply $\mathbf{\Pi}^T\mathbf{A}\mathbf{\Pi} = \mathbf{A}\mathbf{\Pi}\mathbf{\Pi} = \mathbf{A}\mathbf{\Pi} = \mathbf{\Pi}^T\mathbf{A}$.

1.2 $\mathcal{K}(\mathbf{\Pi}^T\mathbf{A}) = \mathcal{K}(\mathbf{A\Pi}) \subseteq \mathcal{K}(\mathbf{\Pi})$.
Proof: Follows directly from Props. 1.2 and 1.5.

Prop. 1.1 implies that $\mathbf{\Pi}$ is a projector. From basic properties of projections, see Theorem 7.3 in [84], $\mathbf{I}_n - \mathbf{\Pi}$ is also a projector. Finally, Prop. 1.1 implies $(\mathbf{\Pi}^T)^2 = \mathbf{\Pi}^T$ so that $\mathbf{\Pi}^T$ is also a projector. More precisely, we have:

Proposition 2.

2.1 $\mathbf{\Pi}$ *is an* $\mathbf{A}$*-orthogonal projector onto* $\mathcal{R}(\mathbf{AW})^\perp$ *along* $\mathcal{R}(\mathbf{W})$.
Proof:

- Prop. 1.3 implies $(\mathbf{AWv})^T\mathbf{\Pi u} = \mathbf{v}^T\mathbf{W}^T\mathbf{A\Pi u} = 0$ for all $(\mathbf{u},\mathbf{v}) \in \mathbb{R}^n \times \mathbb{R}^k$ so that $\mathcal{R}(\mathbf{\Pi}) \perp \mathcal{R}(\mathbf{AW})$;
  The fundamental theorem of subspaces implies $\mathcal{R}(\mathbf{AW})^\perp = \mathcal{K}(\mathbf{W}^T\mathbf{A})$ so that
  $\mathbf{u} \in \mathcal{R}(\mathbf{AW})^\perp \implies \mathbf{W}^T\mathbf{Au} = 0 \implies \mathbf{\Pi u} = \mathbf{u} \implies \mathbf{u} \in \mathcal{R}(\mathbf{\Pi})$
  $\implies \mathcal{R}(\mathbf{AW})^\perp \subseteq \mathcal{R}(\mathbf{\Pi})$;
  $\mathcal{R}(\mathbf{\Pi}) \perp \mathcal{R}(\mathbf{AW})$ and $\mathcal{R}(\mathbf{AW})^\perp \subseteq \mathcal{R}(\mathbf{\Pi})$ imply $\mathcal{R}(\mathbf{\Pi}) = \mathcal{R}(\mathbf{AW})^\perp$.
- Prop. 1.4 implies $\mathbf{\Pi W u} = 0$ for all $\mathbf{u} \in \mathbb{R}^k$ so that $\mathcal{R}(\mathbf{W}) \subseteq \mathcal{K}(\mathbf{\Pi})$;
  $\mathbf{u} \in \mathcal{K}(\mathbf{\Pi}) \implies \mathbf{u} - \mathbf{W}(\mathbf{W}^T\mathbf{AW})^{-1}\mathbf{W}^T\mathbf{Au} = 0 \implies \mathbf{u} = \mathbf{W}(\mathbf{W}^T\mathbf{AW})^{-1}\mathbf{W}^T\mathbf{Au}$
  $\implies \mathbf{u} \in \mathcal{R}(\mathbf{W})$;
  $\mathcal{R}(\mathbf{W}) \subseteq \mathcal{K}(\mathbf{\Pi})$ and $\mathcal{K}(\mathbf{\Pi}) \subseteq \mathcal{R}(\mathbf{W})$ imply $\mathcal{K}(\mathbf{\Pi}) = \mathcal{R}(\mathbf{W})$.

2.2 $\mathbf{I}_n - \mathbf{\Pi}$ *is an* $\mathbf{A}$*-orthogonal projector onto* $\mathcal{R}(\mathbf{W})$ *along* $\mathcal{R}(\mathbf{AW})^\perp$,
Proof: see Theorem 7.3 in [84].

2.3 $\mathbf{\Pi}^T$ *is a projector onto* $\mathcal{R}(\mathbf{W})^\perp$,
Proof:
Prop. 1.4 implies $(\mathbf{Wv})^T\mathbf{\Pi}^T\mathbf{u} = \mathbf{u}^T\mathbf{\Pi W v} = 0$ for all $(\mathbf{u},\mathbf{v}) \in \mathbb{R}^n \times \mathbb{R}^k$ so that $\mathcal{R}(\mathbf{\Pi}^T) \perp \mathcal{R}(\mathbf{W})$;
By the fundamental theorem of subspaces, we have $\mathcal{R}(\mathbf{W})^\perp = \mathcal{K}(\mathbf{W}^T)$ so that
$\mathbf{u} \in \mathcal{R}(\mathbf{W})^\perp \implies \mathbf{W}^T\mathbf{u} = 0 \implies \mathbf{\Pi}^T\mathbf{u} = \mathbf{u} \implies \mathbf{u} \in \mathcal{R}(\mathbf{\Pi}^T)$
$\implies \mathcal{R}(\mathbf{W})^\perp \subseteq \mathcal{R}(\mathbf{\Pi}^T)$;
$\mathcal{R}(\mathbf{\Pi}^T) \perp \mathcal{R}(\mathbf{W})$ and $\mathcal{R}(\mathbf{W})^\perp \subseteq \mathcal{R}(\mathbf{\Pi}^T)$ imply $\mathcal{R}(\mathbf{\Pi}^T) = \mathcal{R}(\mathbf{W})^\perp$.

Moreover, every $\mathbf{u} \in \mathbb{R}^n$ is uniquely decomposed into

$$\mathbf{u} \;=\; \underbrace{(\mathbf{I}_n - \mathbf{\Pi})\mathbf{u}}_{\mathbf{u}_1 \in \mathcal{R}(\mathbf{W})} \;+\; \underbrace{\mathbf{\Pi u}}_{\mathbf{u}_2 \in \mathcal{R}(\mathbf{AW})^\perp}, \tag{2.4}$$

i.e., $\mathbb{R}^n = \mathcal{R}(\mathbf{W}) \oplus \mathcal{R}(\mathbf{AW})^\perp$.

Corollary 2. *Every* $\mathbf{u} \in \mathbb{R}^n$ *is such that* $\mathbf{u} = \mathbf{W}\hat{\boldsymbol{\mu}} + \mathbf{\Pi u}$ *where* $\hat{\boldsymbol{\mu}} \in \mathbb{R}^k$ *is the unique solution of the so-called reduced system* $\mathbf{W}^T\mathbf{AW}\hat{\boldsymbol{\mu}} = \mathbf{W}^T\mathbf{Au}$.
Proof: Eq. (2.4) implies $\mathbf{W}\hat{\boldsymbol{\mu}} = (\mathbf{I}_n - \mathbf{\Pi})\mathbf{u} = \mathbf{W}(\mathbf{W}^T\mathbf{AW})^{-1}\mathbf{W}^T\mathbf{Au}$. $\mathbf{W}$ being full column rank by definition, we have $\hat{\boldsymbol{\mu}} = (\mathbf{W}^T\mathbf{AW})^{-1}\mathbf{W}^T\mathbf{Au}$ in which $\mathbf{W}^T\mathbf{AW}$ is invertible.

Proposition 3. $\mathbf{\Pi}^T\mathbf{A}$ *is symmetric positive semi-definite with null space* $\mathcal{K}(\mathbf{\Pi}^T\mathbf{A}) = \mathcal{R}(\mathbf{W})$.
Proof: Since $\mathbf{A}$ is SPD, we have $\mathbf{u}^T\mathbf{Au} > 0$ for all $\mathbf{u} \neq 0$. In particular, using Coro. 1.1 and Prop. 2.1, we have $\mathbf{v}^T\mathbf{\Pi}^T\mathbf{Av} = \mathbf{v}^T\mathbf{\Pi}^T\mathbf{A\Pi}_s\mathbf{v} = (\mathbf{\Pi v})^T\mathbf{A}(\mathbf{\Pi v}) > 0$ for all $\mathbf{v} \in \mathcal{K}(\mathbf{\Pi})^\perp = \mathcal{R}(\mathbf{W})^\perp$, and $\mathbf{v}^T\mathbf{\Pi}^T\mathbf{Av} = 0$ for all $\mathbf{v} \in \mathcal{K}(\mathbf{\Pi}) = \mathcal{R}(\mathbf{W})$.

Proposition 4. *Each solution* $\mathbf{u}_2 \in \mathbb{R}^n$ *of* $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}_2 = \mathbf{\Pi}^T\mathbf{b}$ *is such that* $\mathbf{\Pi}\mathbf{u}_2 = \mathbf{u}_2^* = \mathbf{\Pi}\mathbf{u}^*$, *where* $\mathbf{u}_2^*$ *is the only solution of the system lying in* $\mathcal{R}(\mathbf{\Pi})$.

Proof:

- Coro. 1.1 implies $\mathbf{\Pi}^T\mathbf{b} = \mathbf{\Pi}^T\mathbf{A}\mathbf{u}^* = \mathbf{\Pi}^T\mathbf{A}\mathbf{\Pi}\mathbf{u}^* \in \mathcal{R}(\mathbf{\Pi}^T\mathbf{A})$ so that $\mathbf{u}_2^* := \mathbf{\Pi}\mathbf{u}^* \in \mathcal{R}(\mathbf{\Pi})$ is solution of $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}_2^* = \mathbf{\Pi}^T\mathbf{b}$;

- Let $\mathbf{u}_3 \in \mathcal{R}(\mathbf{\Pi})$ be such that $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}_3 = \mathbf{\Pi}^T\mathbf{b}$ and define $\mathbf{e} := \mathbf{u}_3 - \mathbf{u}_2 \in \mathcal{R}(\mathbf{\Pi})$;
  Then, $\mathbf{\Pi}^T\mathbf{A}\mathbf{e} = \mathbf{\Pi}^T\mathbf{b} - \mathbf{\Pi}^T\mathbf{b} = 0$, so that Coro. 1.2 implies $\mathbf{e} \in \mathcal{K}(\mathbf{\Pi}^T\mathbf{A}) \subseteq \mathcal{K}(\mathbf{\Pi})$;
  Since $\mathbf{e} \in \mathcal{R}(\mathbf{\Pi}) \cap \mathcal{K}(\mathbf{\Pi})$, we have $\mathbf{e} = \mathbf{\Pi}\mathbf{e} = 0$ which implies $\mathbf{u}_3 = \mathbf{u}_2$;
  Thus $\mathbf{u}_2^* := \mathbf{\Pi}\mathbf{u}^*$ is the only solution of $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}_2 = \mathbf{\Pi}^T\mathbf{b}_s$ lying in $\mathcal{R}(\mathbf{\Pi})$.

- Coro. 2 implies that each solution $\mathbf{u}_2 \in \mathbb{R}^n$ of $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}_2 = \mathbf{\Pi}^T\mathbf{b}$ admits a unique decomposition of the form $\mathbf{u}_2 = \mathbf{\Pi}\mathbf{u}_2 + \mathbf{W}\hat{\boldsymbol{\mu}}_2$;
  Prop. 1.3 implies $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}_2 = \mathbf{\Pi}^T\mathbf{A}\mathbf{\Pi}\mathbf{u}_2 + \mathbf{\Pi}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}_2 = \mathbf{\Pi}^T\mathbf{A}\mathbf{\Pi}\mathbf{u}_2$ so that $\mathbf{\Pi}\mathbf{u}_2 \in \mathcal{R}(\mathbf{W})$ is also solution;
  By unicity of solutions in $\mathcal{R}(\mathbf{\Pi})$, we have $\mathbf{\Pi}\mathbf{u}_2 = \mathbf{u}_2^*$.

Corollary 3. $\mathbf{u}^* = \mathbf{u}_1^* + \mathbf{\Pi}\mathbf{u}_2$ *in which (i)* $\mathbf{u}_1^* = \mathbf{W}\boldsymbol{\mu}_1^* \in \mathcal{R}(\mathbf{W})$ *is such that* $\boldsymbol{\mu}_1^*$ *is the unique solution of the reduced system* $\mathbf{W}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}_1^* = \mathbf{W}^T\mathbf{b}$, *and (ii)* $\mathbf{u}_2 \in \mathbb{R}^n$ *is any solution of the deflated system* $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}_2 = \mathbf{\Pi}^T\mathbf{b}$.

Proof:

- $\mathbf{A}\mathbf{u}^* = \mathbf{b}$ and Coro. 2 imply a decomposition of the form $\mathbf{u}^* = \mathbf{W}\hat{\boldsymbol{\mu}}_1^* + \mathbf{\Pi}\mathbf{u}^*$ where $\hat{\boldsymbol{\mu}}_1^*$ is the unique solution of $\mathbf{W}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}_1^* = \mathbf{W}^T\mathbf{A}\mathbf{u}^* = \mathbf{W}^T\mathbf{b}$;

- By Prop. 4, any solution $\mathbf{u}_2 \in \mathbb{R}^n$ of $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}_2 = \mathbf{\Pi}^T\mathbf{b}$ is such that $\mathbf{\Pi}\mathbf{u}_2 = \mathbf{\Pi}\mathbf{u}^*$, therefore $\mathbf{u}^* = \mathbf{W}\boldsymbol{\mu}_1^* + \mathbf{\Pi}\mathbf{u}_2$.

Definition 1. *Consider a symmetric* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *along with* $\mathbf{b} \in \mathbb{R}^n$ *and* $\mathbf{u}^{(0)} \in \mathbb{R}^n$. *We define the conjugate gradient (CG) as a method to generate a sequence of approximations* $\{\mathbf{u}^{(j)}\}_{j=0}^m$ *of* $\mathbf{u}^* := \mathbf{A}^{-1}\mathbf{b}$ *such that*

$$\mathbf{u}^{(j)} \in \mathbf{u}^{(0)} + \mathcal{K}^{(j)}(\mathbf{A}, \mathbf{r}^{(0)})$$
$$\mathbf{r}^{(j)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(j)} \perp \mathcal{K}^{(j)}(\mathbf{A}, \mathbf{r}^{(0)})$$

where $\mathcal{K}^{(j)}(\mathbf{A}, \mathbf{r}^{(0)}) := \mathrm{Span}\{\mathbf{r}^{(0)}, \mathbf{A}\mathbf{r}^{(0)}, \ldots, \mathbf{A}^{j-1}\mathbf{r}^{(0)}\}$ *is the* j*-th Krylov subspace generated by* $\mathbf{A}$ *and* $\mathbf{r}^{(0)}$. *Although several interpretations of the underlying procedure exist which sometimes lead to different implementations, we consider Algo. 6.18 of [138] in particular and refer to it as* $\mathrm{CG}(\mathbf{A}, \mathbf{b}, \mathbf{u}^{(0)})$.

Definition 2. *Consider a symmetric positive semi-definite matrix* $\mathbf{A}' \in \mathbb{R}^{n\times n}$, *which admits a decomposition of the form* $\mathbf{A}' = \mathbf{U}'\mathbf{\Lambda}'\mathbf{U}'^T$ *with* $\mathbf{U}' \in \mathbb{R}^{n\times n'}$ *such that* $\mathbf{U}'^T\mathbf{U}' = \mathbf{I}_{n'}$ *and* $\mathbf{\Lambda}' \in \mathbb{R}^{n'\times n'}$ *is a diagonal matrix whose entries are the* n' *strictly positive eigenvalues of* $\mathbf{A}'$. *Then, we refer to* $\mathrm{cond}'(\mathbf{A}') := \max\{\sigma(\mathbf{\Lambda}')\}/\min\{\sigma(\mathbf{\Lambda}')\}$ *as the effective conditioning number of* $\mathbf{A}'$.

Threorem 1. *Let* $\mathbf{A}' \in \mathbb{R}^{n\times n}$ *be a symmetric positive semi-definite matrix with null space* $\mathcal{K}(\mathbf{A}')$, *and consider* $\mathbf{b}' \perp \mathcal{K}(\mathbf{A}')$. *Then, for each* $\mathbf{u}^{(0)'} \in \mathbb{R}^n$ *there is a unique* $\mathbf{u}' \in \mathbb{R}^n$ *which satisfies both* $\mathbf{A}\mathbf{u}' = \mathbf{b}$ *and* $\mathbf{u}' - \mathbf{u}^{(0)'} \perp \mathcal{K}(\mathbf{A}')$. *Moreover, assuming exact arithmetic, the iterates generated by* $\mathrm{CG}(\mathbf{A}', \mathbf{b}', \mathbf{u}^{(0)'})$ *are such that:*

1.1 If the residuals $\mathbf{r}^{(0)'}, \ldots, \mathbf{r}^{(j-1)'}$ *are non-zero, then* $\alpha^{(0)'}, \ldots, \alpha^{(j-1)'}$ *and* $\mathbf{p}^{(0)'}, \ldots, \mathbf{p}^{(j-1)'}$ *are defined and non-zero. Consequently,* $\mathbf{r}^{(j)'} \perp \mathcal{K}(\mathbf{A}')$*. Then, if* $\mathbf{r}^{(j)'} = 0$*, we have* $\mathbf{u}^{(j)'} = \mathbf{u}'$*.*
Proof:

- Since $\mathbf{A}'$ is symmetric positive semi-definite, it admits a decomposition $\mathbf{A}' = \mathbf{U}'\mathbf{\Lambda}'\mathbf{U}'^T$ where the diagonal matrix $\mathbf{\Lambda}' \in \mathbb{R}^{n' \times n'}$ has the n' strictly positive eigenvalues $\lambda^{(1)}, \ldots, \lambda^{(n')}$ of $\mathbf{A}'$ as entries and $\mathbf{U}'^T\mathbf{U}' = \mathbf{I}_{n'}$. Moreover, $\mathbf{I}_n - \mathbf{U}'\mathbf{U}'^T$ is a projector onto $\mathcal{K}(\mathbf{A}')$, while $\mathbf{U}'\mathbf{U}'^T$ projects onto $\mathcal{K}(\mathbf{A}')^\perp$. Thus, since $\mathbf{b}' \perp \mathcal{K}(\mathbf{A}')$, there exists $\mathbf{v}_b \in \mathbb{R}^n$ such that $\mathbf{b}' = \mathbf{U}'\mathbf{U}'^T\mathbf{v}_b$. Let $\mathbf{u}' := \mathbf{U}'\mathbf{\Lambda}'^{-1}\mathbf{U}'^T\mathbf{b}' + \tilde{\mathbf{u}}^{(0)'}$ with $\tilde{\mathbf{u}}^{(0)'} := (\mathbf{I}_n - \mathbf{U}'\mathbf{U}'^T)\mathbf{u}^{(0)'} \in \mathcal{K}(\mathbf{A}')$ so that $\mathbf{A}'\mathbf{u}' = \mathbf{b}'$ and $\mathbf{u}' - \mathbf{u}^{(0)'} = \mathbf{U}'\mathbf{\Lambda}'^{-1}\mathbf{U}'^T\mathbf{b}' - \mathbf{U}'\mathbf{U}'^T\mathbf{u}^{(0)'} \perp \mathcal{K}(\mathbf{A}')$. Now, if we let $\mathbf{u}'' := \mathbf{u}' + \Delta\tilde{\mathbf{u}}'$ where $\Delta\tilde{\mathbf{u}}' \in \mathbb{R}^n$, clearly, $\mathbf{A}'\mathbf{u}'' = \mathbf{b}'$ if and only if $\Delta\tilde{\mathbf{u}}' \in \mathcal{K}(\mathbf{A}')$, in which case $\mathbf{u}'' - \mathbf{u}^{(0)'} \perp \mathcal{K}(\mathbf{A}')$ implies $\Delta\tilde{\mathbf{u}}' = 0$ and $\mathbf{u}'' = \mathbf{u}'$. Thus, $\mathbf{u}' = \mathbf{U}'\mathbf{\Lambda}'^{-1}\mathbf{U}'^T\mathbf{b}' + \tilde{\mathbf{u}}^{(0)'}$ is the only solution of $\mathbf{A}'\mathbf{u}' = \mathbf{b}'$ satisfying $\mathbf{u}' - \mathbf{u}^{(0)'} \perp \mathcal{K}(\mathbf{A}')$.

- If $\mathbf{u}^{(0)'} \in \mathcal{K}(\mathbf{A}')$, we have $\mathbf{r}^{(0)'} := \mathbf{b}' - \mathbf{A}'\mathbf{u}^{(0)'} = \mathbf{b}' \perp \mathcal{K}(\mathbf{A}')$. If, however, $\mathbf{u}^{(0)'} \perp \mathcal{K}(\mathbf{A}')$ then, for all $\mathbf{v} \in \mathcal{K}(\mathbf{A}')$ we have $\mathbf{v}^T\mathbf{A}'\mathbf{u}^{(0)'} = (\mathbf{A}'\mathbf{v})^T\mathbf{u}^{(0)'} = 0$ so that $\mathbf{A}'\mathbf{u}^{(0)'} \perp \mathcal{K}(\mathbf{A}')$ and $\mathbf{b}' - \mathbf{A}'\mathbf{u}^{(0)'} \perp \mathcal{K}(\mathbf{A}')$—in either case, $\mathbf{r}^{(0)'} \perp \mathcal{K}(\mathbf{A}')$. Then, if $\mathbf{r}^{(0)'} = 0$, we have $\mathbf{u}' - \mathbf{u}^{(0)'} \perp \mathcal{K}(\mathbf{A}')$ and $\mathbf{A}'\mathbf{u}^{(0)'} = \mathbf{b}'$, so that $\mathbf{u}^{(0)'} = \mathbf{u}'$.

- $\mathrm{CG}(\mathbf{A}', \mathbf{b}', \mathbf{u}^{(0)'})$ defines $\mathbf{p}^{(0)'} := \mathbf{r}^{(0)'}$ along with the recurrence formula

$$\begin{aligned}
\alpha^{(i-1)'} &:= \mathbf{r}^{(i-1)'^T}\mathbf{r}^{(i-1)'} / \mathbf{p}^{(i-1)'^T}\mathbf{A}'\mathbf{p}^{(i-1)'}, \\
\mathbf{u}^{(i)'} &:= \mathbf{u}^{(i-1)'} + \alpha^{(i-1)'}\mathbf{p}^{(i-1)'}, \\
\mathbf{r}^{(i)'} &:= \mathbf{r}^{(i-1)'} - \alpha^{(i-1)'}\mathbf{A}'\mathbf{p}^{(i-1)'}, \\
\beta^{(i-1)'} &:= \mathbf{r}^{(i)'^T}\mathbf{r}^{(i)'} / \mathbf{r}^{(i-1)'^T}\mathbf{r}^{(i-1)'}, \\
\mathbf{p}^{(i)'} &:= \mathbf{r}^{(i)'} + \beta^{(i-1)'}\mathbf{p}^{(i-1)'},
\end{aligned}$$

  for $1 \leq i \leq j$. First, note that $\mathbf{r}^{(i-1)'} \in \mathcal{K}(\mathbf{A}')^\perp \setminus \{0\}$ and $\mathbf{p}^{(i-1)'} \in \mathcal{K}(\mathbf{A}')^\perp \setminus \{0\}$ imply that $\alpha^{(i-1)'}$ is defined and non-zero. Second, since $\mathbf{p}^{(i-1)'} \perp \mathcal{K}(\mathbf{A}')$ implies $\mathbf{A}'\mathbf{p}^{(i-1)'} \perp \mathcal{K}(\mathbf{A}')$, we have that $\mathbf{r}^{(i-1)'} \perp \mathcal{K}(\mathbf{A}')$ and $\mathbf{p}^{(i-1)'} \perp \mathcal{K}(\mathbf{A})$ altogether imply $\mathbf{r}^{(i)'} \perp \mathcal{K}(\mathbf{A}')$. Then, we have $\mathbf{p}^{(i)'} \in \mathcal{K}(\mathbf{A}')^\perp$, which is zero if and only if $\mathbf{r}^{(i)'} = 0$. Therefore, proceeding by induction with $\mathbf{r}^{(0)'} = \mathbf{p}^{(0)'} \perp \mathcal{K}(\mathbf{A}')$, we have that $\{\mathbf{p}^{(i)'}\}_{i=0}^{j} \perp \mathcal{K}(\mathbf{A}')$, $\{\mathbf{r}^{(i)'}\}_{i=0}^{j} \perp \mathcal{K}(\mathbf{A}')$ and $\{\alpha^{(i)'}\}_{i=0}^{j}$ are all defined unless $\mathbf{r}^{(i)'} = 0$ for some $0 \leq i < j$.

- Now, if $\mathbf{r}^{(j)'} = 0$, we have $\mathbf{p}^{(j)'} = 0$ which leads to an undefined $\alpha^{(j)'}$. However, from the recurrence formula, we have $\mathbf{u}^{(j)'} = \mathbf{u}^{(0)'} + \sum_{i=0}^{j-1} \alpha^{(i)'}\mathbf{p}^{(i)'}$ in which every $\mathbf{p}^{(i)'} \perp \mathcal{K}(\mathbf{A}')$. Therefore, $\mathbf{u}' - \mathbf{u}^{(j)'} = \mathbf{U}'\mathbf{\Lambda}'^{-1}\mathbf{U}'^T\mathbf{b}' + (\mathbf{I}_n - \mathbf{U}'\mathbf{U}'^T)\mathbf{u}^{(0)'} - \mathbf{u}^{(j)'} = \mathbf{U}'\mathbf{\Lambda}'^{-1}\mathbf{U}'^T\mathbf{b}' - \mathbf{U}'\mathbf{U}'^T\mathbf{u}^{(0)'} - \sum_{i=0}^{j-1} \alpha^{(i)'}\mathbf{p}^{(i)'}$ so that $\mathbf{u}' - \mathbf{u}^{(j)'} \perp \mathcal{K}(\mathbf{A}')$. Consequently, $\mathbf{r}^{(j)'} = \mathbf{b}' - \mathbf{A}'\mathbf{u}^{(j)'} = \mathbf{A}'(\mathbf{u}' - \mathbf{u}^{(j)'}) = 0$ implies $\mathbf{u}^{(j)'} = \mathbf{u}'$.

1.2 Assuming $\mathbf{r}^{(0)'}, \ldots, \mathbf{r}^{(j-1)'}$ *are non-zero,* $\mathbf{u}^{(j)'}$ *is such that*

$$\mathbf{u}^{(j)'} \in \mathbf{u}^{(0)'} + \mathcal{K}^{(j)}(\mathbf{A}', \mathbf{r}^{(0)'}), \tag{2.5}$$

$$\mathbf{r}^{(j)'} \perp \mathcal{K}^{(j)}(\mathbf{A}', \mathbf{r}^{(0)'}). \tag{2.6}$$

Moreover, $\mathbf{u}^{(j)'}$ *is optimal in the sense that it minimizes the* $\mathbf{A}'$*-norm of the error* $\mathbf{e}^{(j)'} := \mathbf{u}' - \mathbf{u}^{(j)'}$ *over* $\mathbf{u}^{(0)'} + \mathcal{K}^{(j)}(\mathbf{A}', \mathbf{r}^{(0)'})$*, i.e.* $\mathbf{u}^{(j)'} = \min_{\mathbf{u} \in \mathbf{u}^{(0)'} + \mathcal{K}^{(j)}(\mathbf{A}', \mathbf{r}^{(0)'})} \|\mathbf{u}' - \mathbf{u}\|_{\mathbf{A}'}$.

Proof:

- Since $\mathbf{u}^{(j)'} = \mathbf{u}^{(0)'} + \sum_{i=0}^{j-1} \alpha^{(i)'} \mathbf{p}^{(i)'}$ for $j > 0$, we have

$$\begin{aligned}
\mathbf{p}^{(0)'} = \mathbf{r}^{(0)'} &\implies \mathbf{p}^{(0)'} \in \mathrm{Span}\{\mathbf{r}^{(0)'}\}, \\
\mathbf{p}^{(1)'} = \beta^{(0)} \mathbf{p}^{(0)'} + \mathbf{r}^{(0)'} - \alpha^{(0)} \mathbf{A}' \mathbf{p}^{(0)'} &\implies \mathbf{p}^{(1)'} \in \mathrm{Span}\{\mathbf{r}^{(0)'}, \mathbf{A}'\mathbf{r}^{(0)'}\} = \mathcal{K}^{(1)}(\mathbf{A}', \mathbf{r}^{(0)'}), \\
\mathbf{p}^{(2)\prime} = \beta^{(1)'} \mathbf{p}^{(1)'} + \mathbf{r}^{(0)'} - \alpha^{(0)'} \mathbf{A}' \mathbf{p}^{(0)'} - \alpha^{(1)'} \mathbf{A}' \mathbf{p}^{(1)\prime} &\implies \mathbf{p}^{(2)'} \in \mathrm{Span}\{\mathbf{r}^{(0)'}, \mathbf{A}'\mathbf{r}^{(0)'}, \mathbf{A}'^2\mathbf{r}^{(0)'}\} = \mathcal{K}^{(2)}(\mathbf{A}', \mathbf{r}^{(0)'})
\end{aligned}$$

  and so on so that $\mathbf{p}^{(j)'} \in \mathcal{K}^{(j)}(\mathbf{A}', \mathbf{r}^{(0)'})$ and $\mathbf{u}^{(j)'} \in \mathbf{u}^{(0)'} + \mathcal{K}^{(j)}(\mathbf{A}', \mathbf{r}^{(0)'})$.

- The recurrence relations are such that, for $j > 0$, $\mathbf{r}^{(j)'} = \mathbf{r}^{(0)'} - \sum_{i=0}^{j-1} \alpha^{(i)'} \mathbf{A}' \mathbf{p}^{(i)'}$ so that $\mathbf{r}^{(j)'} \in \mathbf{r}^{(0)'} + \mathbf{A}'\mathcal{K}^{(j)}(\mathbf{A}', \mathbf{r}^{(0)'})$. Therefore, $\mathbf{r}^{(j)'^T} \mathbf{r}^{(j-1)'} = 0$ implies $\mathbf{r}^{(j)'} \perp \mathbf{r}^{(0)'} + \mathbf{A}'\mathcal{K}^{(j-1)}(\mathbf{A}', \mathbf{r}^{(0)'})$, which is equivalently stated by $\mathbf{r}^{(j)'} \perp \mathcal{K}^{(j)}(\mathbf{A}', \mathbf{r}^{(0)'})$.

- The optimality of the iterate follows from the property of orthogonal projections, see Prop. 5.2 in [138].

1.3 If $\mathbf{r}^{(0)'}, \ldots, \mathbf{r}^{(n'-1)'}$ *are non-zero, then* $\mathbf{r}^{(n')'} = 0$ *and* $\mathbf{u}^{(n')'} = \mathbf{u}'$*, where* n' *is the number of strictly positive eigenvalues of* $\mathbf{A}'$*.*

Proof:

- For each $\mathbf{u} \in \mathbf{u}^{(0)'} + \mathcal{K}^{(j)}(\mathbf{A}', \mathbf{r}^{(0)'})$, there exists $\{\gamma^{(i)}\}_{i=0}^{j-1}$ such that $\mathbf{u} = \mathbf{u}^{(0)'} + \sum_{i=0}^{j-1} \gamma^{(i)} \mathbf{A}'^i \mathbf{r}^{(0)'}$. Then, since $\mathbf{r}^{(0)'} = \mathbf{b}' - \mathbf{A}'\mathbf{u}^{(0)'} = \mathbf{A}'\mathbf{u}' - \mathbf{A}'\mathbf{u}^{(0)'}$, we can write

$$\mathbf{u}' - \mathbf{u} = \mathbf{u}' - \mathbf{u}^{(0)'} - \sum_{i=0}^{j-1} \gamma^{(i)} \mathbf{A}'^i \mathbf{A}'(\mathbf{u}' - \mathbf{u}^{(0)'}) = \left(\mathbf{I}_n - \sum_{i=0}^{j-1} \gamma^{(i)} \mathbf{A}'^{i+1} \right) (\mathbf{u}' - \mathbf{u}^{(0)'}).$$

- Let's recast $\mathbf{u}' - \mathbf{u}^{(0)'} = \mathbf{U}'\mathbf{\Lambda}'^{-1}\mathbf{U}'\mathbf{b}' - \mathbf{U}'\mathbf{U}'^T\mathbf{u}^{(0)'}$ into $\mathbf{u}' - \mathbf{u}^{(0)'} = \mathbf{U}'\Delta\mathbf{v}$ with $\Delta\mathbf{v} := \mathbf{\Lambda}'^{-1}\mathbf{U}'\mathbf{b}' - \mathbf{U}'^T\mathbf{u}^{(0)'} \in \mathbb{R}^{n'}$. We then have

$$\|\mathbf{u}' - \mathbf{u}\|_{\mathbf{A}'} = \left\| \mathbf{A}'^{1/2}(\mathbf{u}' - \mathbf{u}) \right\|_2 = \left\| \mathbf{U}'\mathbf{\Lambda}'^{1/2}\mathbf{U}'^T \left(\mathbf{I}_n - \sum_{i=0}^{j-1} \gamma^{(i)} \mathbf{U}'\mathbf{\Lambda}'^{i+1}\mathbf{U}'^T \right) \mathbf{U}'\Delta\mathbf{v} \right\|_2$$

$$\|\mathbf{u}' - \mathbf{u}\|_{\mathbf{A}'} = \left\| \left(\mathbf{I}_{n'} - \sum_{i=0}^{j-1} \gamma^{(i)} \mathbf{\Lambda}'^{i+1} \right) \mathbf{\Lambda}'^{1/2} \Delta\mathbf{v} \right\|_2$$

  so that

$$\|\mathbf{u}' - \mathbf{u}\|_{\mathbf{A}'} \leq \left\| \mathbf{I}_{n'} - \sum_{i=0}^{j-1} \gamma^{(i)} \mathbf{\Lambda}'^{i+1} \right\|_2 \left\| \mathbf{\Lambda}'^{1/2} \Delta\mathbf{v} \right\|_2 \leq \left\| \mathbf{I}_{n'} - \sum_{i=0}^{j-1} \gamma^{(i)} \mathbf{\Lambda}'^{i+1} \right\|_2 \|\mathbf{u}' - \mathbf{u}^{(0)'}\|_{\mathbf{A}'}$$

in which we have

$$\left\| \mathbf{I}_{n'} - \sum_{i=0}^{j-1} \gamma^{(i)} {\mathbf{\Lambda'}}^{i+1} \right\|_2 = \max_{\mathbf{v}\in\mathbb{R}^{n'} \text{ s.t. } \|\mathbf{v}\|_2=1} \left\| \left(\mathbf{I}_{n'} - \sum_{i=0}^{j} \gamma^{(i)} {\mathbf{\Lambda'}}^{i+1} \right) \mathbf{v} \right\|_2 \leq \max_{z\in\sigma(\mathbf{\Lambda'})} \left| 1 - \sum_{i=0}^{j-1} \gamma^{(i)} z^{i+1} \right|.$$

Thus, for every $\mathbf{u} \in {\mathbf{u}^{(0)}}' + \mathcal{K}^{(j)}(\mathbf{A}', {\mathbf{r}^{(0)}}')$, there is a polynomial $p : \mathbb{R} \to \mathbb{R}$ of degree j such that $p(0) = 1$ and $\|\mathbf{u}' - \mathbf{u}\|_{\mathbf{A}'} \leq \max_{z\in\sigma(\mathbf{\Lambda'})} |p(z)| \|\mathbf{u}' - {\mathbf{u}^{(0)}}'\|_{\mathbf{A}'}$.

- Owing to the optimal property of the iterate ${\mathbf{u}^{(j)}}'$, see Theo. 1.2, we have

$$\|\mathbf{u}' - {\mathbf{u}^{(j)}}'\|_{\mathbf{A}'} \leq \min_{p\in\mathbb{P}_j \text{ s.t. } p(0)=1} \max_{z\in\sigma(\mathbf{\Lambda'})} |p(z)| \|\mathbf{u}' - {\mathbf{u}^{(0)}}'\|_{\mathbf{A}'}$$

where $\mathbb{P}_j$ is the set of all the polynomials of degree j. In particular, we consider

$$\tilde{p}(z) = \prod_{i=1}^{j} \frac{\lambda^{(i)} - z}{\lambda^{(i)}}$$

which does satisfy $\tilde{p}(0) = 1$. As we let $j := n'$, we have $\tilde{p}(z) = 0$ for all $z \in \sigma(\mathbf{\Lambda'})$. Therefore,

$$\|\mathbf{u}' - {\mathbf{u}^{(n')}}'\|_{\mathbf{A}'} \leq \max_{z\in\sigma(\mathbf{\Lambda'})} |\tilde{p}(z)| \|\mathbf{u}' - {\mathbf{u}^{(0)}}'\|_{\mathbf{A}'} = 0.$$

From Theo. 1.1, we have that ${\mathbf{r}^{(0)}}', \ldots, {\mathbf{r}^{(j-1)}}'$ being non-zero implies $\mathbf{u}' - {\mathbf{u}^{(j)}}' \perp \mathcal{K}(\mathbf{A}')$. Consequently, we have $\mathbf{u}' - {\mathbf{u}^{(n')}}' \perp \mathcal{K}(\mathbf{A}')$ so that $\|\mathbf{u}' - {\mathbf{u}^{(n')}}'\|_{\mathbf{A}'} = 0$ implies ${\mathbf{u}^{(n')}}' = \mathbf{u}'$.

1.4 The $\mathbf{A}'$-norm of the error of each iterate ${\mathbf{u}^{(j)}}'$ is bounded by

$$\frac{\|\mathbf{e}^{(j)}\|_{\mathbf{A}'}}{\|\mathbf{e}^{(0)}\|_{\mathbf{A}'}} \leq 2 \left[\left(\frac{\sqrt{\text{cond}'(\mathbf{A}')} - 1}{\sqrt{\text{cond}'(\mathbf{A}')} + 1} \right)^j + \left(\frac{\sqrt{\text{cond}'(\mathbf{A}')} + 1}{\sqrt{\text{cond}'(\mathbf{A}')} - 1} \right)^j \right]^{-1} \leq 2 \left(\frac{\sqrt{\text{cond}'(\mathbf{A}')} - 1}{\sqrt{\text{cond}'(\mathbf{A}')} + 1} \right)^j \tag{2.7}$$

where $\text{cond}'(\mathbf{A}')$ *is the effective conditioning number of A' defined as in Def. 2.*
Proof: Similar to proof of Theorem 4.

If $\mathbf{A}'$ is SPD, Theos. 1.1–1.4 apply without conditions on $\mathbf{b}'$.

Proposition 5. *An approximation of $\mathbf{u}^* := \mathbf{A}^{-1}\mathbf{b}_s$ can be obtained after the following procedure:*

1. *Solve for $\hat{\boldsymbol{\mu}}_1^* \in \mathbb{R}^k$ in $\mathbf{W}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}_1^* = \mathbf{W}^T\mathbf{b}$.*
2. *Pick any ${\mathbf{u}^{(0)}}' \in \mathbb{R}^n$ and solve for $\mathbf{u}'$ in*

$$\mathbf{\Pi}^T\mathbf{A}\mathbf{u}' = \mathbf{\Pi}^T\mathbf{b}$$
$$\mathbf{u}' - {\mathbf{u}^{(0)}}' \perp \mathcal{R}(\mathbf{W})$$

which can be approximated with the sequence generated by $\mathrm{CG}(\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime})$ *of which the behavior conforms to Theos. 1.1–1.4 with* $\mathbf{A}' := \mathbf{\Pi}^T\mathbf{A}$ *and* $\mathbf{b}' := \mathbf{\Pi}^T\mathbf{b}$.

3. *Then,* $\mathbf{u}^* = \mathbf{W}\hat{\boldsymbol{\mu}}_1^* + \mathbf{\Pi}\mathbf{u}'$.

Proof:

- Prop. 3 states that $\mathbf{\Pi}^T\mathbf{A}$ is symmetric positive semi-definite with null space $\mathcal{R}(\mathbf{W})$; Prop. 2.3 states that $\mathbf{\Pi}^T$ projects onto $\mathcal{R}(\mathbf{W})^\perp$ so that $\mathbf{\Pi}^T\mathbf{b} \perp \mathcal{R}(\mathbf{W})$;
  Let $\mathbf{A}' := \mathbf{\Pi}^T\mathbf{A}$ and $\mathbf{b}' := \mathbf{\Pi}^T\mathbf{b}$. Then, the necessary conditions for the application of Theos. 1.1–1.4 are met and $\mathrm{CG}(\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime})$ generates a sequence of approximations of $\mathbf{u}'$.

- By Coro. 3, since $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}' = \mathbf{\Pi}^T\mathbf{b}$, we have $\mathbf{u}^* = \mathbf{W}\hat{\boldsymbol{\mu}}_1^* + \mathbf{\Pi}\mathbf{u}'$.

The procedure presented in Prop. 5 is referred to as Proj-CG in [55]. Note that, although the solution $\mathbf{u}'$ of $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}' = \mathbf{\Pi}^T\mathbf{b}$ such that $\mathbf{u}' - \mathbf{u}^{(0)\prime} \perp \mathcal{R}(\mathbf{W})$ depends on the choice of $\mathbf{u}^{(0)\prime}$, this is not the case of $\mathbf{\Pi}\mathbf{u}'$ which remains unchanged irrespective of $\mathbf{u}^{(0)\prime}$. The potential of deflation becomes clear when looking at the right hand side of the bound on the $\mathbf{\Pi}^T\mathbf{A}$-norm of the error of the approximations generated by $\mathrm{CG}(\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime})$, see Theo. 1.4. Clearly, the convergence of those iterates is governed by the distribution of the $n' = n - k$ strictly positive eigenvalues of $\mathbf{\Pi}^T\mathbf{A}$ which, if $\mathbf{W}$ is properly chosen, can consist of the central part of the spectrum of $\mathbf{A}$, possibly resulting in $\mathrm{cond}'(\mathbf{\Pi}^T\mathbf{A}) \ll \mathrm{cond}(\mathbf{A})$ and a faster convergence of $\mathrm{CG}(\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime})$ than $\mathrm{CG}(\mathbf{A}, \mathbf{b}, \mathbf{u}^{(0)})$. The resulting speed-up, however, should be compounded with the resolution of the reduced system of dimension k inherent to the application of the operator $\mathbf{\Pi}^T\mathbf{A}$, hence motivating a choice of $k \ll n$.

Proposition 6. *Given the sequence* $\{\mathbf{u}^{(i)\prime}\}_{i=0}^j$ *of estimates generated by* $\mathrm{CG}(\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime})$ *for some* $\mathbf{u}^{(0)\prime}$*, let* $\{\mathbf{u}^{(i)}\}_{i=0}^j := \{\mathbf{W}\hat{\boldsymbol{\mu}}_1^* + \mathbf{\Pi}\mathbf{u}^{(i)\prime}\}_{i=0}^j$ *and* $\{\mathbf{r}^{(i)}\}_{i=0}^j := \{\mathbf{b} - \mathbf{A}\mathbf{u}^{(i)}\}_{i=0}^j$*. Then, for* $0 \leq i \leq j$*, we have*

6.1 $\mathbf{\Pi}^T\mathbf{r}^{(i)} = \mathbf{r}^{(i)\prime}$.
Proof: $\mathbf{\Pi}^T\mathbf{r}^{(i)} = \mathbf{\Pi}^T(\mathbf{b} - \mathbf{A}\mathbf{u}^{(i)}) = \mathbf{\Pi}^T\mathbf{b} - \mathbf{\Pi}^T\mathbf{A}(\mathbf{W}\hat{\boldsymbol{\mu}}_1^* + \mathbf{\Pi}\mathbf{u}^{(i)\prime})$ where Prop. 1.3 $\implies$ $\mathbf{\Pi}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}_1^* = 0$, and Coro. 1.1 $\implies$ $\mathbf{\Pi}^T\mathbf{A}\mathbf{\Pi}\mathbf{u}^{(i)\prime} = \mathbf{\Pi}^T\mathbf{A}\mathbf{u}^{(i)\prime}$ so that $\mathbf{\Pi}^T\mathbf{r}^{(i)} = \mathbf{\Pi}^T\mathbf{b} - \mathbf{\Pi}^T\mathbf{A}\mathbf{u}^{(i)\prime} =: \mathbf{r}^{(i)\prime}$.

6.2 If $\mathbf{u}^{(0)\prime}$ *is such that* $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$*, then* $\mathbf{r}^{(i)} = \mathbf{r}^{(i)\prime}$.
Proof: $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$ and Prop. 2.3 $\implies$ $\mathbf{r}^{(0)} = \mathbf{\Pi}^T\mathbf{r}^{(0)}$;
$\mathbf{r}^{(0)} = \mathbf{\Pi}^T\mathbf{r}^{(0)}$ and $\mathbf{\Pi}^T\mathbf{r}^{(i)} = \mathbf{r}^{(i)\prime}$ $\implies$ $\mathbf{r}^{(0)} = \mathbf{r}^{(0)\prime}$;
Note that $\mathbf{r}^{(i)} - \mathbf{r}^{(i-1)} = \mathbf{A}(\mathbf{u}^{(i-1)} - \mathbf{u}^{(i)}) = \mathbf{A}\mathbf{\Pi}(\mathbf{u}^{(i-1)\prime} - \mathbf{u}^{(i)\prime})$;
Coro. 1.1 $\implies$ $\mathbf{A}\mathbf{\Pi}(\mathbf{u}^{(i-1)\prime} - \mathbf{u}^{(i)\prime}) = \mathbf{\Pi}^T\mathbf{A}(\mathbf{u}^{(i-1)\prime} - \mathbf{u}^{(i)\prime}) = \mathbf{r}^{(i)\prime} - \mathbf{r}^{(i-1)\prime}$ so that $\mathbf{r}^{(i)} - \mathbf{r}^{(i-1)} = \mathbf{r}^{(i)\prime} - \mathbf{r}^{(i-1)\prime}$;
$\mathbf{r}^{(0)} = \mathbf{r}^{(0)\prime}$ and $\mathbf{r}^{(i)} - \mathbf{r}^{(i-1)} = \mathbf{r}^{(i)\prime} - \mathbf{r}^{(i-1)\prime}$ $\implies$ $\mathbf{r}^{(i)} = \mathbf{r}^{(i)\prime}$.

An alternative to Proj-CG is to exploit the relation between $\mathbf{u}^*$ and $\mathbf{u}'$ for a given $\mathbf{u}^{(0)\prime}$, and construct a sequence $\{\mathbf{u}^{(i)}\}_{i=0}^j$ of approximations of $\mathbf{u}^*$ from the sequence $\{\mathbf{u}^{(i)\prime}\}_{i=0}^j$ generated by $\mathrm{CG}(\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime})$. This procedure, presented in Prop. 6, is equivalent to the algorithm introduced by [107] and further analyzed by [140] in the sense that it generates the same iterates.

Proposition 7. *Given the sequence* $\{\mathbf{u}^{(i)\prime}\}_{i=0}^{j}$ *of estimates generated by* $\mathrm{CG}(\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime})$ *for some* $\mathbf{u}^{(0)\prime}$ *such that* $\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)} \perp \mathcal{R}(\mathbf{W})$*, there exists* $\hat{\boldsymbol{\mu}}_1^* \in \mathbb{R}^k$ *such that the sequence* $\{\mathbf{u}^{(i)}\}_{i=0}^{j} := \{\mathbf{W}\hat{\boldsymbol{\mu}}_1^* + \mathbf{\Pi}\mathbf{u}^{(i)\prime}\}_{i=0}^{j}$ *consists of approximations of* $\mathbf{u}^* := \mathbf{A}^{-1}\mathbf{b}$ *such that*

$$\begin{aligned}\mathbf{u}^{(i)} &\in \mathbf{u}^{(0)} + \mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)})\\ \mathbf{r}^{(i)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(i)} &\perp \mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)})\end{aligned}$$

for $i \leq j$*, where* $\mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}) := \mathcal{K}^{(i)}(\mathbf{A}, \mathbf{r}^{(0)}) \oplus \mathcal{R}(\mathbf{W})$ *is the i-th Krylov subspace* $\mathcal{K}^{(i)}(\mathbf{A}, \mathbf{r}^{(0)})$ *augmented by the null space of* $\mathbf{\Pi}^T\mathbf{A}$*. Moreover,* $\mathbf{u}^{(i)}$ *is optimal in the sense that it minimizes the* $\mathbf{A}$*-norm of the error* $\mathbf{e}^{(i)} := \mathbf{u}^* - \mathbf{u}^{(i)}$ *over* $\mathbf{u}^{(0)} + \mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)})$*, i.e.* $\mathbf{u}^{(i)} = \min_{\mathbf{u} \in \mathbf{u}^{(0)} + \mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)})} \|\mathbf{u}^* - \mathbf{u}\|_{\mathbf{A}}$.

Proof:

- Coro. 2 implies $\mathbf{\Pi}\mathbf{u}^{(i)\prime} = \mathbf{u}^{(i)\prime} - \mathbf{W}\hat{\boldsymbol{\mu}}^{(i)\prime}$ where $\hat{\boldsymbol{\mu}}^{(i)\prime}$ is the solution of $\mathbf{W}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}^{(i)\prime} = \mathbf{W}^T\mathbf{A}\mathbf{u}^{(i)\prime}$ for $0 \leq i \leq j$.
  Then, $\mathbf{u}^{(i)} := \mathbf{W}\hat{\boldsymbol{\mu}}_1^* + \mathbf{\Pi}\mathbf{u}^{(i)\prime} = \mathbf{W}\hat{\boldsymbol{\mu}}_1^* + \mathbf{u}^{(i)\prime} - \mathbf{W}\hat{\boldsymbol{\mu}}^{(i)\prime}$ so that, using $\mathbf{u}^{(0)} := \mathbf{W}\hat{\boldsymbol{\mu}}_1^* + \mathbf{\Pi}\mathbf{u}^{(0)\prime}$, we get $\mathbf{u}^{(i)} = \mathbf{u}^{(0)} - \mathbf{W}(\hat{\boldsymbol{\mu}}^{(i)*} - \hat{\boldsymbol{\mu}}^{(0)\prime}) + (\mathbf{u}^{(j)\prime} - \mathbf{u}^{(0)\prime})$ where we know from Theo. 1.2 that $\mathbf{u}^{(j)\prime} \in \mathbf{u}^{(0)\prime} + \mathcal{K}^{(i)}(\mathbf{\Pi}^T\mathbf{A}, \mathbf{r}^{(0)\prime})$. Therefore, we have

  $$\mathbf{u}^{(i)} \in \mathbf{u}^{(0)} + \mathcal{R}(\mathbf{W}) + \mathcal{K}^{(i)}(\mathbf{\Pi}^T\mathbf{A}, \mathbf{r}^{(0)\prime}).$$

  Note for all $\mathbf{v} \in \mathcal{R}(\mathbf{\Pi}^T\mathbf{A})$, there exists $\mathbf{w}$ such that $\mathbf{v} = \mathbf{\Pi}^T\mathbf{A}\mathbf{w}$. Then, since $\mathbf{A}$ is invertible, we can define $\mathbf{w}' = \mathbf{A}^{-1}\mathbf{\Pi}^T\mathbf{A}\mathbf{w}$ so that $\mathbf{v} = \mathbf{A}\mathbf{w}' \in \mathcal{R}(\mathbf{A})$. Consequently, we have $\mathcal{R}(\mathbf{\Pi}^T\mathbf{A}) \subseteq \mathcal{R}(\mathbf{A})$, which can be used to show $\mathcal{K}^{(i)}(\mathbf{\Pi}^T\mathbf{A}, \mathbf{r}^{(0)\prime}) \subseteq \mathcal{K}^{(i)}(\mathbf{A}, \mathbf{r}^{(0)\prime})$.
  Then, note that $\mathbf{r}^{(0)\prime} = \mathbf{\Pi}^T\mathbf{b} - \mathbf{\Pi}^T\mathbf{A}\mathbf{u}^{(0)\prime} \in \mathcal{R}(\mathbf{\Pi}^T) = \mathcal{R}(\mathbf{W})^\perp$ so that $\mathcal{R}(\mathbf{W}) \cap \mathcal{K}^{(i)}(\mathbf{A}, \mathbf{r}^{(0)\prime}) = \{0\}$ and

  $$\mathbf{u}^{(i)} \in \mathbf{u}^{(0)} + \mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)\prime})$$

  where $\mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{r}^{(0)\prime}) := \mathcal{K}^{(i)}(\mathbf{A}, \mathbf{r}^{(0)\prime}) \oplus \mathcal{R}(\mathbf{W})$. For the case in which $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$, see Prop. 6.2, we have $\mathbf{r}^{(0)} = \mathbf{r}^{(0)\prime}$ so that $\mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)\prime}) = \mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)})$.

- Assume $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$ so that $\mathbf{r}^{(i)} = \mathbf{r}^{(i)\prime}$, see Prop. 6.2. Then, since $\mathbf{r}^{(i)} = \mathbf{r}^{(0)} - \mathbf{A}(\mathbf{u}^{(i)} - \mathbf{u}^{(0)})$, we have $\mathbf{r}^{(i)} \in \mathbf{r}^{(0)} + \mathbf{A}\mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)})$. Hence, $\mathbf{r}^{(i)^T}\mathbf{r}^{(i-1)} = 0$ implies $\mathbf{r}^{(i)} \perp \mathbf{r}^{(0)} + \mathbf{A}\mathcal{K}^{(k,i-1)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)})$ so that

  $$\mathbf{r}^{(i)} \perp \mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}).$$

- Assuming $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$, the two precedent points imply that every iterate $\mathbf{u}^{(i)}$ is the result of an orthogonal projection onto $\mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)})$ with initial iterate $\mathbf{u}^{(0)}$. From the property of orthogonal projections, see Prop. 5.2 in [138], we have that $\mathbf{u}^{(i)}$ is optimal in the sense that it minimizes the $\mathbf{A}$-norm of the error $\mathbf{e}^{(i)} := \mathbf{u}^* - \mathbf{u}^{(i)}$ over $\mathbf{u}^{(0)\prime} + \mathcal{K}^{(k,i)}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)})$.

Although the relevance of Prop. 7 may seem questionable when simply presented

as a post-processing stage of the sequence generated by CG($\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime}$), we will see that a proper change of variables can be used to construct the sequence $\{\mathbf{u}^{(i)}\}_{i=0}^{j}$ of approximations of $\mathbf{u}^*$, without having to explicitly form the iterates $\{\mathbf{u}^{(i)\prime}\}_{i=0}^{j}$. The resulting algorithm, presented as deflated CG in [140], requires a single resolution of reduced system per iterate, similarly as for the application of the projector in an iteration of CG($\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime}$) as called by Proj-CG.

2.2.1 Deflated conjugate gradient

The sequence $\{\mathbf{u}^{(i)}\}_{i=0}^{j}$ of approximations of $\mathbf{u}^*$ generated by the procedure described in Prop. 7 is equivalently obtained by the deflated CG algorithm—herein referred to as Def-CG($\mathbf{A}, \mathbf{W}, \mathbf{b}, \mathbf{u}^{(0)}$) in Algo. 17—which is presented and analyzed by [140]. Def-CG($\mathbf{A}, \mathbf{W}, \mathbf{b}, \mathbf{u}^{(0)}$) is introduced by [140] as a deflated Lanczos procedure to build a sequence $\{\mathbf{v}^{(i)}\}_{i=1}^{j+1}$ of vectors such that

$$\mathbf{v}^{(i+1)} \perp \mathcal{R}(\mathbf{W}) + \mathrm{Span}\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(i)}\}$$

and $\|\mathbf{v}^{(i+1)}\|_2 = 1$ with $\mathbf{v}^{(1)} := \mathbf{r}^{(0)}/\|\mathbf{r}^{(0)}\|_2 \perp \mathcal{R}(\mathbf{W})$. Each approximation $\mathbf{u}^{(i)}$ of $\mathbf{u}^*$ is then constructed as an orthogonal projection onto the affine subspace $\mathbf{u}^{(0)} + \mathcal{R}(\mathbf{W}) + \mathrm{Span}\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(i)}\}$.

The equivalence between Def-CG($\mathbf{A}, \mathbf{W}, \mathbf{b}, \mathbf{u}^{(0)}$) and CG($\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime}$) was stated in Theo. 4.6 of [140]. Assuming $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$, Def-CG($\mathbf{A}, \mathbf{W}, \mathbf{b}, \mathbf{u}^{(0)}$) can be obtained from CG($\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime}$) using the relations

$$\mathbf{u}^{(j)} := \mathbf{W}\hat{\boldsymbol{\mu}}_1^* + \mathbf{\Pi}\mathbf{u}^{(j)\prime}, \tag{2.8}$$

$$\mathbf{r}^{(j)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(j)}, \tag{2.9}$$

$$\mathbf{p}^{(j)} := \mathbf{\Pi}\mathbf{p}^{(j)\prime}. \tag{2.10}$$

To illustrate this equivalence, we first present CG($\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime}$) in Algo. 15.

Algorithm 15 CG applied to $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}' = \mathbf{\Pi}^T\mathbf{b}$, CG($\mathbf{\Pi}^T\mathbf{A}$, $\mathbf{\Pi}^T\mathbf{b}$).

1: $\mathbf{r}^{(0)\prime} := \mathbf{\Pi}^T\mathbf{b} - \mathbf{\Pi}^T\mathbf{A}\mathbf{u}^{(0)\prime}$ ▷ $\mathbf{u}^{(0)\prime}$ s.t. $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$
2: $\mathbf{p}^{(0)\prime} := \mathbf{r}^{(0)\prime}$
3: **for** $j = 1, \ldots, m$ **do**
4: $\quad\alpha^{(j-1)} := \mathbf{r}^{(j-1)\prime T}\mathbf{r}^{(j-1)\prime}/\mathbf{p}^{(j-1)\prime T}\mathbf{\Pi}^T\mathbf{A}\mathbf{p}^{(j-1)\prime}$
5: $\quad\mathbf{u}^{(j)\prime} := \mathbf{u}^{(j-1)\prime} + \alpha^{(j-1)}\mathbf{p}^{(j-1)\prime}$
6: $\quad\mathbf{r}^{(j)\prime} := \mathbf{r}^{(j-1)\prime} - \alpha^{(j-1)}\mathbf{\Pi}^T\mathbf{A}\mathbf{p}^{(j-1)\prime}$
7: $\quad\beta^{(j-1)} := \mathbf{r}^{(j)\prime T}\mathbf{r}^{(j)\prime}/\mathbf{r}^{(j-1)\prime T}\mathbf{r}^{(j-1)\prime}$
8: $\quad\mathbf{p}^{(j)\prime} := \mathbf{r}^{(j)\prime} + \beta^{(j-1)}\mathbf{p}^{(j-1)\prime}$
9: **end for**

Proposition 8. *Assuming* $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$*, the following recurrence relations hold:*

8.1 $\mathbf{u}^{(j)} = \mathbf{u}^{(j-1)} + \alpha^{(j-1)}\mathbf{p}^{(j-1)}$.

Proof: Eq. (2.8), line 5 of Algo. 15 and Eq. (2.10) $\implies$ $\mathbf{u}^{(j)} - \mathbf{u}^{(j-1)} = \mathbf{\Pi}(\mathbf{u}^{(j)\prime} -$

$\mathbf{u}^{(j-1)\prime}) = \alpha^{(j-1)}\mathbf{\Pi}\mathbf{p}^{(j-1)\prime} = \alpha^{(j-1)}\mathbf{p}^{(j-1)}$.

8.2 $\mathbf{r}^{(j)} = \mathbf{r}^{(j-1)} - \alpha^{(j-1)}\mathbf{A}\mathbf{p}^{(j-1)}$.
Proof: Coro. 1.1 and Eq. (2.10) $\implies$ $\mathbf{\Pi}^T\mathbf{A}\mathbf{p}^{(j-1)\prime} = \mathbf{A}\mathbf{\Pi}\mathbf{p}^{(j-1)\prime} = \mathbf{A}\mathbf{p}^{(j-1)}$;
Line 6 of Algo. 15 and $\mathbf{\Pi}^T\mathbf{A}\mathbf{p}^{(j-1)\prime} = \mathbf{A}\mathbf{p}^{(j-1)} \implies \mathbf{r}^{(j)\prime} = \mathbf{r}^{(j-1)\prime} - \alpha^{(j-1)}\mathbf{A}\mathbf{p}^{(j-1)}$;
Prop.6.2 and $\mathbf{r}^{(j\prime} = \mathbf{r}^{(j-1)\prime} - \alpha^{(j-1)}\mathbf{A}\mathbf{p}^{(j-1)} \implies \mathbf{r}^{(j)} = \mathbf{r}^{(j-1)} - \alpha^{(j-1)}\mathbf{A}\mathbf{p}^{(j-1)}$.

Algo. 16 is obtained after applying the substitutions given by Eqs. (2.9), (2.10), Prop. 6.2 and Props. 8.1–8.2 to Algo. 15. Note also that the condition $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$ need not be enforced through $\mathbf{u}^{(0)\prime}$ as, instead, $\mathbf{u}_s^{(0)}$ can be picked accordingly.

Algorithm 16 CG applied to $\mathbf{\Pi}^T\mathbf{A}\mathbf{u}' = \mathbf{\Pi}^T\mathbf{b}$, CG($\mathbf{\Pi}^T\mathbf{A}$, $\mathbf{\Pi}^T\mathbf{b}$, $\mathbf{u}^{(0)}$) with substitutions.

1: $\mathbf{r}^{(0} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)}$ ▷ $\mathbf{u}^{(0)}$ s.t. $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$
2: $\mathbf{p}^{(0)} := \mathbf{\Pi}\mathbf{r}^{(0)}$
3: **for** $j = 1, \ldots, m$ **do**
4: $\quad \alpha^{(j-1)} := \mathbf{r}^{(j-1)^T}\mathbf{r}^{(j-1)} / \mathbf{p}^{(j-1)^T}\mathbf{A}\mathbf{p}^{(j-1)}$
5: $\quad \mathbf{u}^{(j)} := \mathbf{u}^{(j-1)} + \alpha^{(j-1)}\mathbf{p}^{(j-1)}$
6: $\quad \mathbf{r}^{(j)} := \mathbf{r}^{(j-1)} - \alpha^{(j-1)}\mathbf{A}\mathbf{p}^{(j-1)}$
7: $\quad \beta^{(j-1)} := \mathbf{r}^{(j)^T}\mathbf{r}^{(j)} / \mathbf{r}^{(j-1)^T}\mathbf{r}^{(j-1)}$
8: $\quad \mathbf{p}^{(j)} := \mathbf{\Pi}\mathbf{r}^{(j)} + \beta^{(j-1)}\mathbf{p}^{(j-1)}$
9: **end for**

Decomposing $\mathbf{r}^{(j)}$ similarly as in Coro. 2 yields a unique $\hat{\boldsymbol{\mu}}^{(j)} \in \mathbb{R}^k$ such that $\mathbf{r}^{(j)} = \mathbf{W}\hat{\boldsymbol{\mu}}^{(j)} + \mathbf{\Pi}\mathbf{r}^{(j)}$, which leads to $\mathbf{\Pi}\mathbf{r}^{(j)} = \mathbf{r}^{(j)} - \mathbf{W}\hat{\boldsymbol{\mu}}^{(j)}$ where $\hat{\boldsymbol{\mu}}^{(j)}$ is the solution of $\mathbf{W}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}^{(j)} = \mathbf{W}^T\mathbf{A}\mathbf{r}^{(j)}$. Incorporating this application of the projector to $\mathbf{r}^{(j)}$ into Algo. 16 leads to Algo. 17, referred to as deflated CG in [140].

Algorithm 17 Deflated CG, Def-CG($\mathbf{A}$, $\mathbf{W}$, $\mathbf{b}$, $\mathbf{u}^{(0)}$).

1: $\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)}$ ▷ $\mathbf{u}^{(0)}$ s.t. $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$
2: Solve for $\hat{\boldsymbol{\mu}}^{(0)}$ in $\mathbf{W}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}^{(0)} = \mathbf{W}^T\mathbf{A}\mathbf{r}^{(0)}$
3: $\mathbf{p}^{(0)} := \mathbf{r}^{(0)} - \mathbf{W}\hat{\boldsymbol{\mu}}^{(0)}$
4: **for** $j = 1, \ldots, m$ **do**
5: $\quad \alpha^{(j-1)} := \mathbf{r}^{(j-1)^T}\mathbf{r}^{(j-1)} / \mathbf{p}^{(j-1)^T}\mathbf{A}\mathbf{p}^{(j-1)}$
6: $\quad \mathbf{u}^{(j)} := \mathbf{u}^{(j-1)} + \alpha^{(j-1)}\mathbf{p}^{(j-1)}$
7: $\quad \mathbf{r}^{(j)} := \mathbf{r}^{(j-1)} - \alpha^{(j-1)}\mathbf{A}\mathbf{p}^{(j-1)}$
8: $\quad \beta^{(j-1)} := \mathbf{r}^{(j)^T}\mathbf{r}^{(j)} / \mathbf{r}^{(j-1)^T}\mathbf{r}^{(j-1)}$
9: $\quad$ Solve for $\hat{\boldsymbol{\mu}}^{(j)}$ in $\mathbf{W}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}^{(j)} = \mathbf{W}^T\mathbf{A}\mathbf{r}^{(j)}$
10: $\quad \mathbf{p}^{(j)} := \beta^{(j-1)}\mathbf{p}^{(j-1)} + \mathbf{r}^{(j)} - \mathbf{W}\hat{\boldsymbol{\mu}}^{(j)}$
11: **end for**

The resulting algorithm, Algo. 17, requires to resolve one reduced system per iterate (see line 9), similarly as for the application of the deflated operator in an iteration of CG($\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime}$) as called by Proj-CG, see $\mathbf{\Pi}^T\mathbf{A}\mathbf{p}^{(j-1)\prime}$ in line 4 of Algo. 15. The main difference between the two methods is that every iterate $\mathbf{u}^{(j)}$ of Def-CG($\mathbf{A}, \mathbf{W}, \mathbf{b}, \mathbf{u}^{(0)}$) is an optimal approximation of $\mathbf{u}^*$, while each iterate $\mathbf{u}^{(j)\prime}$ of CG($\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime}$) is an

approximation of $\mathbf{u}'$ which, in itself, is insignificant as its relation with $\mathbf{u}^*$ depends on the arbitrary initial iterate $\mathbf{u}^{(0)\prime}$.

Given that Algo. 15 is obtained under the assumption that $\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)} \perp \mathcal{R}(\mathbf{W})$, it is of practical interest to be able to construct an initial iterate $\mathbf{u}^{(0)}$ such that $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$ from some arbitrary $\mathbf{u}^{(-1)} \in \mathbb{R}^n$. This can be achieved using the following relation given by [140]:

$$\mathbf{u}^{(0)} = \mathbf{u}^{(-1)} + \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{r}^{(-1)} \tag{2.11}$$

with $\mathbf{r}^{(-1)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(-1)}$ and for which we have

$$\begin{aligned}\mathbf{W}^T\mathbf{r}^{(0)} &= \mathbf{W}^T(\mathbf{b} - \mathbf{A}\mathbf{u}^{(0)}) = \mathbf{W}^T\mathbf{b} - \mathbf{W}^T\mathbf{A}\mathbf{u}^{(-1)} - \mathbf{W}^T\mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{r}^{(-1)} \\ &= \mathbf{W}^T\mathbf{b} - \mathbf{W}^T\mathbf{A}\mathbf{u}^{(-1)} - \mathbf{W}^T\mathbf{r}^{(-1)} = 0\end{aligned}$$

so that, indeed, $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$.

Now, given the equivalence between Def-CG$(\mathbf{A}, \mathbf{W}, \mathbf{b}, \mathbf{u}^{(0)})$ and CG$(\mathbf{\Pi}^T\mathbf{A}, \mathbf{\Pi}^T\mathbf{b}, \mathbf{u}^{(0)\prime})$, the behavior of the deflated CG can be analyzed making use of Theo. 1 with $\mathbf{A}' := \mathbf{\Pi}^T\mathbf{A}$ and $\mathbf{b}' := \mathbf{\Pi}^T\mathbf{b}$.

2.2.1.1 Preconditioned case

Deflation can be presented in the context of split preconditioned systems with inverse preconditioners admitting a Cholesky decomposition of the form $\mathbf{M}^{-1} = \mathbf{L}^{-T}\mathbf{L}^{-1}$. Similarly as for the non-preconditioned case, a matrix $\dot{\mathbf{\Pi}} \in \mathbb{R}^{n\times n}$ is introduced and defined by

$$\dot{\mathbf{\Pi}} := \mathbf{I}_n - \dot{\mathbf{W}}(\dot{\mathbf{W}}^T\dot{\mathbf{A}}\dot{\mathbf{W}})^{-1}\dot{\mathbf{W}}^T\dot{\mathbf{A}} \tag{2.12}$$

where $\dot{\mathbf{A}} := \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}$ and $\dot{\mathbf{W}}$ has columns that form the basis of a subspace of dimension $k < n$. Since $\dot{\mathbf{W}}$ is full column rank and $\dot{\mathbf{A}}$ is SPD, so is $\dot{\mathbf{W}}^T\dot{\mathbf{A}}\dot{\mathbf{W}}$ which is hence invertible. Note that Props. 1–4 and Coros. 1–2 remain applicable when substituting $\mathbf{A}$, $\mathbf{W}$ and $\mathbf{\Pi}$ by $\dot{\mathbf{A}}$, $\dot{\mathbf{W}}$ and $\dot{\mathbf{\Pi}}$, respectively.

Proposition 9. *Given the sequence* $\{\dot{\mathbf{u}}^{(i)\prime}\}_{i=0}^j$ *of estimates generated by* CG$(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}, \dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}, \dot{\mathbf{u}}^{(0)\prime})$ *for some* $\dot{\mathbf{u}}^{(0)\prime}$*, there exists* $\hat{\boldsymbol{\mu}}_1^* \in \mathbb{R}^k$ *such that we can define the sequence* $\{\dot{\mathbf{u}}^{(i)}\}_{i=0}^j := \{\dot{\mathbf{W}}\hat{\boldsymbol{\mu}}_1^* + \dot{\mathbf{\Pi}}\dot{\mathbf{u}}^{(i)\prime}\}_{i=0}^j$ *along with* $\{\dot{\mathbf{r}}^{(i)}\}_{i=0}^j := \{\mathbf{L}^{-1}\mathbf{b} - \dot{\mathbf{A}}\dot{\mathbf{u}}^{(i)}\}_{i=0}^j$*. Let also* $\{\mathbf{u}^{(i)}\}_{i=0}^j := \{\mathbf{L}^{-T}\dot{\mathbf{u}}^{(i)}\}_{i=0}^j$ *and* $\{\mathbf{r}^{(i)}\}_{i=0}^j := \{\mathbf{b} - \mathbf{A}\mathbf{u}^{(i)}\}_{i=0}^j$*. Then, for* $0 \le i \le j$*, we have:*

9.1 $\dot{\mathbf{\Pi}}^T\dot{\mathbf{r}}^{(i)} = \dot{\mathbf{r}}^{(i)\prime}$.
Proof: Same as for Prop. 6.1.

9.2 If $\dot{\mathbf{u}}^{(0)\prime}$ *is such that* $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$*, then* $\dot{\mathbf{r}}^{(i)} = \dot{\mathbf{r}}^{(i)\prime}$.
Proof: Same as for Prop. 6.2.

9.3 $\mathbf{L}\dot{\mathbf{r}}^{(i)} = \mathbf{r}^{(i)}$.
Proof: $\mathbf{L}\dot{\mathbf{r}}^{(i)} = \mathbf{L}(\mathbf{L}^{-1}\mathbf{b} - \dot{\mathbf{A}}\dot{\mathbf{u}}^{(i)}) = \mathbf{b} - \mathbf{A}\mathbf{L}^{-T}\dot{\mathbf{u}}^{(i)} = \mathbf{b} - \mathbf{A}\mathbf{u}^{(i)} =: \mathbf{r}^{(i)}$.

Proposition 10. *Given the sequence* $\{\dot{\mathbf{u}}^{(i\prime)}\}_{i=0}^j$ *of estimates generated by* CG$(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}, \dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}, \dot{\mathbf{u}}^{(0)\prime})$ *for some* $\dot{\mathbf{u}}^{(0)\prime}$ *such that* $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$ *where* $\dot{\mathbf{r}}^{(i)} := \mathbf{L}^{-1}\mathbf{b} - \dot{\mathbf{A}}\dot{\mathbf{u}}^{(i)}$ *and* $\dot{\mathbf{u}}^{(i)} := \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(1)*} +$

$\dot{\mathbf{\Pi}}\dot{\mathbf{u}}^{(i)\prime}$, *the sequence* $\{\mathbf{u}^{(i)}\}_{i=0}^{j} := \{\mathbf{L}^{-T}\dot{\mathbf{u}}^{(i)}\}_{i=0}^{j}$ *consists of approximations of* $\mathbf{u}^* := \mathbf{A}^{-1}\mathbf{b}$ *such that*

$$\mathbf{u}^{(i)} \in \mathbf{u}^{(0)} + \mathcal{K}^{(k,i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{W}, \mathbf{M}^{-1}\mathbf{r}^{(0)})$$
$$\mathbf{r}^{(i)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(i)} \perp \mathcal{K}^{(k,i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{W}, \mathbf{M}^{-1}\mathbf{r}^{(0)})$$

for $i \leq j$, *where* $\mathcal{K}^{(k,i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{W}, \mathbf{M}^{-1}\mathbf{r}^{(0)}) = \mathcal{K}^{(i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{M}^{-1}\mathbf{r}^{(0)}) \oplus \mathcal{R}(\mathbf{W})$ *is the i-th Krylov subspace generated by* $\mathbf{M}^{-1}\mathbf{A}$ *and* $\mathbf{M}^{-1}\mathbf{r}^{(0)}$, *augmented by the null space of* $\mathbf{\Pi}^T\mathbf{A}$ *spanned by the columns of* $\mathbf{W} := \mathbf{L}^{-T}\dot{\mathbf{W}}$. *Moreover,* $\mathbf{u}^{(i)}$ *is optimal in the sense that it minimizes the* $\mathbf{A}$*-norm of the error* $\mathbf{e}^{(i)} := \mathbf{u}^* - \mathbf{u}^{(i)}$ *over* $\mathbf{u}^{(0)} + \mathcal{K}^{(k,i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{W}, \mathbf{M}^{-1}\mathbf{r}^{(0)})$.

Proof:

- Coro. 2 implies $\dot{\mathbf{\Pi}}\dot{\mathbf{u}}^{(i)\prime} = \dot{\mathbf{u}}^{(i)\prime} - \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(i)\prime}$ where $\hat{\boldsymbol{\mu}}^{(i)\prime}$ is the solution of $\dot{\mathbf{W}}^T\dot{\mathbf{A}}\dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(i)\prime} = \dot{\mathbf{W}}^T\dot{\mathbf{A}}\dot{\mathbf{u}}^{(i)\prime}$.
  Then, $\dot{\mathbf{u}}^{(i)} := \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(1)*} + \dot{\mathbf{\Pi}}\dot{\mathbf{u}}^{(i)\prime} = \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(1)*} + \dot{\mathbf{u}}^{(i)\prime} - \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(i)\prime}$ so that, using $\dot{\mathbf{u}}^{(0)} := \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(1)*} + \dot{\mathbf{\Pi}}\dot{\mathbf{u}}^{(0)\prime}$, we get $\dot{\mathbf{u}}^{(i)} = \dot{\mathbf{u}}^{(0)} - \dot{\mathbf{W}}(\hat{\boldsymbol{\mu}}^{(i)\prime} - \hat{\boldsymbol{\mu}}^{(0)\prime}) + (\dot{\mathbf{u}}^{(i)\prime} - \dot{\mathbf{u}}^{(0)\prime})$ where we know from Theo. 2.2 that $\dot{\mathbf{u}}^{(0)\prime} \in \dot{\mathbf{u}}^{(0)\prime} + \mathcal{K}^{(i)}(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}, \dot{\mathbf{r}}^{(0)\prime})$.
  Therefore, we have $\dot{\mathbf{u}}^{(i)} \in \dot{\mathbf{u}}^{(0)} + \mathcal{R}(\dot{\mathbf{W}}) + \mathcal{K}^{(i)}(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}, \dot{\mathbf{r}}^{(0)\prime})$.
  Since $\dot{\mathbf{A}}$ is SPD, we have $\mathcal{R}(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}) \subseteq \mathcal{R}(\dot{\mathbf{A}})$, which can be used to show $\mathcal{K}^{(i)}(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}, \dot{\mathbf{r}}^{(0)\prime}) \subseteq \mathcal{K}^{(i)}(\dot{\mathbf{A}}, \dot{\mathbf{r}}^{(0)\prime})$.
  For the case in which $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$, see Prop. 9.2, we have $\dot{\mathbf{r}}^{(0)} = \dot{\mathbf{r}}^{(0)\prime}$ so that

  $$\mathbf{L}^{-T}\dot{\mathbf{u}}^{(i)} \in \mathbf{L}^{-T}\dot{\mathbf{u}}^{(0)} + \mathbf{L}^{-T}\mathcal{R}(\dot{\mathbf{W}}) + \mathbf{L}^{-T}\mathcal{K}^{(i)}(\dot{\mathbf{A}}, \dot{\mathbf{r}}^{(0)}).$$

  Using the definitions of $\mathbf{u}^{(i)}$ and $\mathbf{W}$ as well as Prop. 9.3, we obtain

  $$\mathbf{u}^{(i)} \in \mathbf{u}^{(0)} + \mathcal{R}(\mathbf{W}) + \mathcal{K}^{(i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{M}^{-1}\mathbf{r}^{(0)}).$$

  Then $\mathbf{r}^{(0)} = \mathbf{L}\dot{\mathbf{r}}^{(0)} = \mathbf{L}\dot{\mathbf{r}}^{(0)\prime} = \mathbf{L}\dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b} - \mathbf{L}\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}\dot{\mathbf{u}}^{(0)\prime} \in \mathcal{R}(\mathbf{L}\dot{\mathbf{\Pi}}^T)$ so that $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{L}^{-T}\dot{\mathbf{W}}) = \mathcal{R}(\mathbf{W})$.
  Hence, since $\mathcal{R}(\mathbf{W})^\perp = \mathcal{R}(\mathbf{\Pi}^T)$ is invariant under SPD operators, $\mathcal{R}(\mathbf{W}) \cap \mathcal{K}^{(i)}(\mathbf{M}^{-1}\dot{\mathbf{A}}, \mathbf{M}^{-1}\mathbf{r}^{(0)}) = \{0\}$ implies $\mathbf{u}^{(i)} \in \mathbf{u}^{(0)} + \mathcal{R}(\mathbf{W}) \oplus \mathcal{K}^{(i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{M}^{-1}\mathbf{r}^{(0)})$.

- Assume $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$ so that $\dot{\mathbf{r}}^{(i)} = \dot{\mathbf{r}}^{(i)\prime}$, see Prop. 9.2.
  Using Prop. 9.3, we get $\mathbf{r}^{(i)} - \mathbf{r}^{(0)} = \mathbf{L}(\dot{\mathbf{r}}^{(i)} - \dot{\mathbf{r}}^{(0)}) = -\mathbf{L}\dot{\mathbf{A}}(\dot{\mathbf{u}}^{(i)} - \dot{\mathbf{u}}^{(0)}) = -\mathbf{A}(\mathbf{u}^{(i)} - \mathbf{u}^{(0)})$ so that $\mathbf{r}^{(i)} \in \mathbf{r}^{(0)} + \mathbf{A}\mathcal{K}^{(k,i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{W}, \mathbf{M}^{-1}\mathbf{r}^{(0)})$.
  Then, $\mathbf{r}^{(i)T}\mathbf{M}^{-1}\mathbf{r}^{(i-1)} = \dot{\mathbf{r}}^{(i)T}\dot{\mathbf{r}}^{(i-1)} = \dot{\mathbf{r}}^{(i)\prime T}\dot{\mathbf{r}}^{(i-1)\prime} = 0$ implies $\mathbf{r}^{(i)} \perp \mathbf{M}^{-1}\mathbf{r}^{(0)} + \mathbf{M}^{-1}\mathbf{A}\mathcal{K}^{(k,i-1)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{W}, \mathbf{M}^{-1}\mathbf{r}^{(0)})$ so that

  $$\mathbf{r}^{(i)} \perp \mathcal{K}^{(k,i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{W}, \mathbf{M}^{-1}\mathbf{r}^{(0)}).$$

- Assuming $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$, the two precedent points imply that every iterate $\mathbf{u}^{(i)}$ is the result of an orthogonal projection onto $\mathcal{K}^{(k,i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{W}, \mathbf{M}^{-1}\mathbf{r}^{(0)})$ with initial iterate $\mathbf{u}^{(0)}$. From the property of orthogonal projections, see Prop. 5.2 in [138], we have that $\mathbf{u}^{(i)}$ is optimal in the sense that it minimizes the $\mathbf{A}$-norm of the error $\mathbf{e}^{(i)} := \mathbf{u}^* - \mathbf{u}^{(i)}$ over $\mathbf{u}^{(0)} + \mathcal{K}^{(k,i)}(\mathbf{M}^{-1}\mathbf{A}, \mathbf{W}, \mathbf{M}^{-1}\mathbf{r}^{(0)})$.

The sequence $\{\mathbf{u}^{(i)}\}_{i=0}^{j}$ of approximations of $\mathbf{u}^*$ generated by the procedure described

in Prop. 10 is equivalently obtained by the deflated preconditioned CG algorithm—herein referred to as Def-PCG($\mathbf{A}, \mathbf{M}, \mathbf{W}, \mathbf{b}, \mathbf{u}^{(0)}$) in Algo. 21—which is presented in Algo. 3.6 of [140]. The equivalence between Def-PCG($\mathbf{A}, \mathbf{M}, \mathbf{W}, \mathbf{b}, \mathbf{u}^{(0)}$) and CG($\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}, \dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}, \dot{\mathbf{u}}^{(0)\prime}$) is highlighted here by deriving the later algorithm through the application of the required changes of variables in the former procedure. Assuming $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$, Def-PCG($\mathbf{A}, \mathbf{M}, \mathbf{W}, \mathbf{b}, \mathbf{u}^{(0)}$) can be obtained from CG($\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}, \dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}, \dot{\mathbf{u}}^{(0)\prime}$) using the relations

$$\dot{\mathbf{u}}^{(j)} := \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(1)*} + \dot{\mathbf{\Pi}}\dot{\mathbf{u}}^{(j)\prime}, \tag{2.13}$$

$$\dot{\mathbf{r}}^{(j)} := \mathbf{L}^{-1}\mathbf{b} - \dot{\mathbf{A}}\dot{\mathbf{u}}^{(j)}, \tag{2.14}$$

$$\dot{\mathbf{p}}^{(j)} := \dot{\mathbf{\Pi}}\dot{\mathbf{p}}^{(j)\prime} \tag{2.15}$$

as well as

$$\mathbf{u}^{(j)} := \mathbf{L}^{-T}\dot{\mathbf{u}}^{(j)}, \tag{2.16}$$

$$\mathbf{r}^{(j)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(j)}, \tag{2.17}$$

$$\mathbf{p}^{(j)} := \mathbf{L}^{-T}\dot{\mathbf{p}}^{(j)}, \tag{2.18}$$

$$\mathbf{W} := \mathbf{L}^{-T}\dot{\mathbf{W}}. \tag{2.19}$$

To illustrate this equivalence, we first present CG($\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}, \dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}, \dot{\mathbf{u}}^{(0)\prime}$) in Algo. 18.

Algorithm 18 CG applied to $\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}\dot{\mathbf{u}}' = \dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}$, CG($\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}$, $\dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}$, $\dot{\mathbf{u}}^{(0)\prime}$).

1: $\dot{\mathbf{r}}^{(0)\prime} := \dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b} - \dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}\dot{\mathbf{u}}^{(0)\prime}$
2: $\dot{\mathbf{p}}^{(0)\prime} := \dot{\mathbf{r}}^{(0)\prime}$
3: **for** $j = 1, \ldots, m$ **do**
4: $\quad\alpha^{(j-1)} := \dot{\mathbf{r}}^{(j-1)\prime T}\dot{\mathbf{r}}^{(j-1)\prime} / \dot{\mathbf{p}}^{(j-1)\prime T}\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}\dot{\mathbf{p}}^{(j-1)\prime}$
5: $\quad\dot{\mathbf{u}}^{(j)\prime} := \dot{\mathbf{u}}^{(j-1)\prime} + \alpha^{(j-1)}\dot{\mathbf{p}}^{(j-1)\prime}$
6: $\quad\dot{\mathbf{r}}^{(j)\prime} := \dot{\mathbf{r}}^{(j-1)\prime} - \alpha^{(j-1)}\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}\dot{\mathbf{p}}^{(j-1)\prime}$
7: $\quad\beta^{(j-1)} := \dot{\mathbf{r}}^{(j)\prime T}\dot{\mathbf{r}}^{(j)\prime} / \dot{\mathbf{r}}^{(j-1)\prime T}\dot{\mathbf{r}}^{(j-1)\prime}$
8: $\quad\dot{\mathbf{p}}^{(j)\prime} := \dot{\mathbf{r}}^{(j)\prime} + \beta^{(j-1)}\dot{\mathbf{p}}^{(j-1)\prime}$
9: **end for**

Proposition 11. *Assuming $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$, the following recurrence relations hold:*

11.1 $\dot{\mathbf{u}}^{(j)} = \dot{\mathbf{u}}^{(j-1)} + \alpha^{(j-1)}\dot{\mathbf{p}}^{(j-1)}$.
Proof: Same as for Prop. 8.1.

11.2 $\dot{\mathbf{r}}^{(j)} = \dot{\mathbf{r}}^{(j-1)} - \alpha^{(j-1)}\dot{\mathbf{A}}\dot{\mathbf{p}}^{(j-1)}$.
Proof: Same as for Prop. 8.2.

Decomposing $\dot{\mathbf{r}}^{(j)}$ similarly as in Coro. 2 yields a unique $\hat{\boldsymbol{\mu}}^{(j)} \in \mathbb{R}^k$ such that $\dot{\mathbf{r}}^{(j)} = \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(j)} + \dot{\mathbf{\Pi}}\dot{\mathbf{r}}^{(j)}$, which leads to $\dot{\mathbf{\Pi}}\dot{\mathbf{r}}^{(j)} = \dot{\mathbf{r}}^{(j)} - \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(j)}$ where $\hat{\boldsymbol{\mu}}^{(j)}$ is the solution of $\dot{\mathbf{W}}^T\dot{\mathbf{A}}\dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(j)} = \dot{\mathbf{W}}^T\dot{\mathbf{A}}\dot{\mathbf{r}}^{(j)}$. Incorporating this application of the projector to $\dot{\mathbf{r}}^{(j)}$ into Algo. (18) along with the substitutions given by Eqs. (2.14)–(2.15) and Props. 9.2, 11.1–11.2 yields Algo. (19).

Algorithm 19 CG applied to $\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}\dot{\mathbf{u}}' = \dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}$, CG($\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}$, $\dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}$, $\dot{\mathbf{u}}^{(0)\prime}$) with substitutions.

$\dot{\mathbf{r}}^{(0)} := \mathbf{L}^{-1}\mathbf{b} - \dot{\mathbf{A}}\dot{\mathbf{u}}^{(0)}$ ▷ $\dot{\mathbf{u}}^{(0)}$ s.t. $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$
Solve for $\hat{\boldsymbol{\mu}}^{(0)}$ in $\dot{\mathbf{W}}^T\dot{\mathbf{A}}\dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(0)} = \dot{\mathbf{W}}^T\dot{\mathbf{A}}\dot{\mathbf{r}}^{(0)}$
$\dot{\mathbf{p}}^{(0} := \dot{\mathbf{r}}^{(0)} - \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(0)}$
for $j = 1, \ldots, m$ **do**
  $\alpha^{(j-1)} := \dot{\mathbf{r}}^{(j-1)T}\dot{\mathbf{r}}^{(j-1)} / \dot{\mathbf{p}}^{(j-1)T}\dot{\mathbf{A}}\dot{\mathbf{p}}^{(j-1)}$
  $\dot{\mathbf{u}}^{(j)} := \dot{\mathbf{u}}^{(j-1)} + \alpha^{(j-1)}\dot{\mathbf{p}}^{(j-1)}$
  $\dot{\mathbf{r}}^{(j)} := \dot{\mathbf{r}}^{(j-1)} - \alpha^{(j-1)}\dot{\mathbf{A}}\dot{\mathbf{p}}^{(j-1)}$
  $\beta^{(j-1)} := \dot{\mathbf{r}}^{(j)T}\dot{\mathbf{r}}^{(j)} / \dot{\mathbf{r}}^{(j-1)T}\dot{\mathbf{r}}^{(j-1)}$
  Solve for $\hat{\boldsymbol{\mu}}^{(j)}$ in $\dot{\mathbf{W}}^T\dot{\mathbf{A}}\dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(j)} = \dot{\mathbf{W}}^T\dot{\mathbf{A}}\dot{\mathbf{r}}^{(j)}$
  $\dot{\mathbf{p}}^{(j} := \beta^{(j-1)}\dot{\mathbf{p}}^{(j-1)} + \dot{\mathbf{r}}^{(j)} - \dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(j)}$
end for

In order to make the application of coming changes of variables more intelligible, the occurrence of the right hand sides of Eqs. (2.16)–(2.19) within Algo. 19 are made more explicit in Algo. 20. Finally, applying the substitutions given by Eqs. (2.16)–(2.19) and Prop. 9.3 within Algo. 20 leads to Algo. 21, which we refer to as Def-PCG($\mathbf{A}, \mathbf{M}, \mathbf{W}, \mathbf{b}, \mathbf{u}^{(0)}$) and is given by Algo. 3.6 in [140].

Algorithm 20 CG applied to $\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}\dot{\mathbf{u}}' = \dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}$, CG($\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}$, $\dot{\mathbf{\Pi}}^T\mathbf{L}^{-1}\mathbf{b}$, $\dot{\mathbf{u}}^{(0)\prime}$) reformulated.

$\mathbf{L}\dot{\mathbf{r}}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{L}^{-T}\dot{\mathbf{u}}^{(0)}$ ▷ $\dot{\mathbf{u}}^{(0)}$ s.t. $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$
Solve for $\hat{\boldsymbol{\mu}}^{(0)}$ in $(\mathbf{L}^{-T}\dot{\mathbf{W}})^T\mathbf{A}\mathbf{L}^{-T}\dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(0)} = (\mathbf{L}^{-T}\dot{\mathbf{W}})^T\mathbf{A}\mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{L}\dot{\mathbf{r}}^{(0)}$
$\mathbf{L}^{-T}\dot{\mathbf{p}}^{(0)} := \mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{L}\dot{\mathbf{r}}^{(0)} - \mathbf{L}^{-T}\dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(0)}$
for $j = 1, \ldots, m$ **do**
  $\alpha^{(j-1)} := (\mathbf{L}\dot{\mathbf{r}}^{(j-1)})^T\mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{L}\dot{\mathbf{r}}^{(j-1)} / (\mathbf{L}^{-T}\dot{\mathbf{p}}^{(j-1)})^T\mathbf{A}\mathbf{L}^{-T}\dot{\mathbf{p}}^{(j-1)}$
  $\mathbf{L}^{-T}\dot{\mathbf{u}}^{(j)} := \mathbf{L}^{-T}\dot{\mathbf{u}}^{(j-1)} + \alpha^{(j-1)}\mathbf{L}^{-T}\dot{\mathbf{p}}^{(j-1)}$
  $\mathbf{L}\dot{\mathbf{r}}^{(j)} := \mathbf{L}\dot{\mathbf{r}}^{(j-1)} - \alpha^{(j-1)}\mathbf{A}\mathbf{L}^{-T}\dot{\mathbf{p}}^{(j-1)}$
  $\beta^{(j-1)} := (\mathbf{L}\dot{\mathbf{r}}^{(j)})^T\mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{L}\dot{\mathbf{r}}^{(j)} / (\mathbf{L}\dot{\mathbf{r}}^{(j-1)})^T\mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{L}\dot{\mathbf{r}}^{(j-1)}$
  Solve for $\hat{\boldsymbol{\mu}}^{(j)}$ in $(\mathbf{L}^{-T}\dot{\mathbf{W}})^T\mathbf{A}\mathbf{L}^{-T}\dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(j)} = (\mathbf{L}^{-T}\dot{\mathbf{W}})^T\mathbf{A}\mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{L}\dot{\mathbf{r}}^{(j)}$
  $\mathbf{L}^{-T}\dot{\mathbf{p}}^{(j)} := \beta^{(j-1)}\mathbf{L}^{-T}\dot{\mathbf{p}}^{(j-1)} + \mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{L}\dot{\mathbf{r}}^{(j)} - \mathbf{L}^{-T}\dot{\mathbf{W}}\hat{\boldsymbol{\mu}}^{(j)}$
end for

Given that Algo. 21 is obtained under the assumption that $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$, it is of practical interest to be able to construct an initial iterate $\mathbf{u}^{(0)}$ satisfying this condition for some arbitrary $\mathbf{u}^{(-1)} \in \mathbb{R}^n$. Note that the initial iterate given by Eq. (2.11) satisfies this condition. Indeed, assuming $\mathbf{u}^{(0)} = \mathbf{u}^{(-1)} + \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{r}^{(-1)}$ and using Eq. (2.19) along with Prop. 9.3 leads to

$$\begin{aligned}\dot{\mathbf{W}}^T\dot{\mathbf{r}}^{(0)} &= (\mathbf{L}^T\mathbf{W})^T(\mathbf{L}^{-1}\mathbf{r}^{(0)})\\ \dot{\mathbf{W}}^T\dot{\mathbf{r}}^{(0)} &= \mathbf{W}^T\mathbf{r}^{(0)}\\ \dot{\mathbf{W}}^T\dot{\mathbf{r}}^{(0)} &= 0\end{aligned}$$

which shows that $\mathbf{r}^{(0)} \perp \mathcal{R}(\mathbf{W})$ implies $\dot{\mathbf{r}}^{(0)} \perp \mathcal{R}(\dot{\mathbf{W}})$.

Algorithm 21 Deflated preconditioned CG, Def-PCG($\mathbf{A}$, $\mathbf{M}$, $\mathbf{W}$, $\mathbf{b}$, $\mathbf{u}^{(0)}$).

$\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)}$ ▷ $\mathbf{u}^{(0)}$ s.t. $\mathbf{r}^{(0)} := \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)} \perp \mathcal{R}(\mathbf{W})$
$\mathbf{z}^{(0)} := \mathbf{M}^{-1}\mathbf{r}^{(0)}$
Solve for $\hat{\boldsymbol{\mu}}^{(0)}$ in $\mathbf{W}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}^{(0)} = \mathbf{W}^T\mathbf{A}\mathbf{z}^{(0)}$
$\mathbf{p}^{(0)} := \mathbf{z}^{(0)} - \mathbf{W}\hat{\boldsymbol{\mu}}^{(0)}$
for $j = 1, \ldots, m$ **do**
  $\alpha^{(j-1)} := \mathbf{r}^{(j-1)T}\mathbf{z}^{(j-1)} / \mathbf{p}^{(j-1)T}\mathbf{A}\mathbf{p}^{(j-1)}$
  $\mathbf{u}^{(j)} := \mathbf{u}^{(j-1)} + \alpha^{(j-1)}\mathbf{p}^{(j-1)}$
  $\mathbf{r}^{(j)} := \mathbf{r}^{(j-1)} - \alpha^{(j-1)}\mathbf{A}\mathbf{p}^{(j-1)}$
  $\mathbf{z}^{(j)} := \mathbf{M}^{-1}\mathbf{r}^{(j)}$
  $\beta^{(j-1)} := \mathbf{r}^{(j)T}\mathbf{z}^{(j)} / \mathbf{r}^{(j-1)T}\mathbf{z}^{(j-1)}$
  Solve for $\hat{\boldsymbol{\mu}}^{(j)}$ in $\mathbf{W}^T\mathbf{A}\mathbf{W}\hat{\boldsymbol{\mu}}^{(j)} = \mathbf{W}^T\mathbf{A}\mathbf{z}^{(j)}$
  $\mathbf{p}^{(j)} := \beta^{(j-1)}\mathbf{p}^{(j-1)} + \mathbf{z}^{(j)} - \mathbf{W}\hat{\boldsymbol{\mu}}^{(j)}$
end for

From hereon, there remains to specify procedures for the computation of $\mathbf{W}$. In doing so, our intent is to decrease the effective condition number $\mathrm{cond}'(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}})$ as much as possible in comparison to $\mathrm{cond}(\mathbf{A})$. However, since $\mathbf{W} = \mathbf{L}^{-T}\dot{\mathbf{W}}$ implies $\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}} = \mathbf{L}^{-1}\mathbf{\Pi}^T\mathbf{A}\mathbf{L}^{-T}$, two types of approaches arise. In the first approach, we let the columns of $\mathbf{W}$ be approximate eigenvectors of $\mathbf{A}$, in which case it is convenient to think of $\mathrm{cond}'(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}})$ as the effective condition number of an operator which is deflated prior to be split preconditioned. The second approach consists of letting the columns of $\dot{\mathbf{W}}$ be approximate eigenvectors of $\dot{\mathbf{A}}$, while still only using $\mathbf{W}$ in Def-PCG. In this case, $\mathrm{cond}'(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}})$ is thought of as the effective condition number of an operator which is split preconditioned prior to be deflated. Note that both approaches are not equivalent. Indeed, if $\mathbf{w}$ is an eigenvector of $\mathbf{A}$, then $\dot{\mathbf{w}} := \mathbf{L}^T\mathbf{w}$ is a right eigenvector of $\mathbf{L}^T\mathbf{A}\mathbf{L}^{-T}$, but generally not of $\dot{\mathbf{A}}$. Conversely, if $\dot{\mathbf{w}}$ is an eigenvector of $\dot{\mathbf{A}}$, then $\mathbf{w} := \mathbf{L}^{-T}\dot{\mathbf{w}}$ is a right eigenvector of $\mathbf{M}^{-1}\mathbf{A}$, but generally not of $\mathbf{A}$. Both approaches are investigated in this work.

2.3 Perturbation of the deflation subspace

All the matrices sampled in this work show greater ratios $\lambda^{(\bullet)}(\mathbf{A})/\lambda^{(\bullet+1)}(\mathbf{A})$ in the lower end of the spectrum than in the upper part. Therefore, a more significant decrease of the active condition number $\mathrm{cond}'(\mathbf{\Pi}^T\mathbf{A})$ can be expected when letting the columns of $\mathbf{W}$ be least dominant (LD) eigenvectors of $\mathbf{A}$ rather than most dominant (MD) eigenvectors. When using a preconditioner $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, there are two options. First, similarly as without a preconditioner, a good choice is to let the columns of $\mathbf{W}$ be LD eigenvectors of $\mathbf{A}$. The other option is to let the columns of $\dot{\mathbf{W}}$ be LD eigenvectors of $\dot{\mathbf{A}}$ and then set $\mathbf{W} = \mathbf{L}^{-T}\dot{\mathbf{W}}$. This second option is equivalent to letting the columns of $\mathbf{W}$ be right eigenvectors of $\mathbf{M}^{-1}\mathbf{A}$. In practice, the eigenvectors of $\mathbf{A}$, $\dot{\mathbf{A}}$ and $\mathbf{M}^{-1}\mathbf{A}$ are not known. That is, instead of using $\mathbf{W}$ as previously defined, an approximation is used in the deflated linear solver. In their paper [72], Kahl and Rittich present some results on how the effective condition number of the deflated system depends on the extent of the perturbation of the deflation subspace. In particular, for the case in which we intend to let the columns of $\dot{\mathbf{W}}$ be LD

eigenvectors of $\dot{\mathbf{A}}$, a bound is given for $\text{cond}'(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}})$ in Prop. 12. For the preconditioned case where we let the columns of $\mathbf{W}$ be approximate LD eigenvectors of $\mathbf{A}$, there is unfortunately no similar bound on the effective condition number of the deflated system.

Proposition 12. *Let $\dot{\mathbf{y}}^{(1)}, \ldots, \dot{\mathbf{y}}^{(n)}$ be an orthonormal basis of eigenvectors corresponding to the eigenvalues $\dot{\lambda}^{(1)} \geq \cdots \geq \dot{\lambda}^{(n)} \geq 0$ of $\dot{\mathbf{A}}$. Then, the effective condition number of the deflated system with respect to the deflation subspace $\mathcal{R}(\dot{\mathbf{W}})$ fulfills*

$$\text{cond}'(\dot{\mathbf{\Pi}}^T\dot{\mathbf{A}}) \leq \left(\sqrt{\frac{\dot{\lambda}^{(1)}}{\dot{\lambda}^{(k)}}} + \sqrt{\frac{\dot{\lambda}^{(1)}}{\dot{\lambda}^{(n)}}}\sin(\dot{\theta})\right)^2 = \frac{\dot{\lambda}^{(1)}}{\dot{\lambda}^{(k)}} + \mathcal{O}(\dot{\theta}), \quad \text{for } \dot{\theta} \to 0, \tag{2.20}$$

where $\dot{\theta}$ is the largest principal angle between $\mathcal{R}(\dot{\mathbf{W}})$ and $\mathcal{R}([\dot{\mathbf{y}}^{(k+1)}, \ldots, \dot{\mathbf{y}}^{(n)}])$.
Proof: *See Kahl and Rittich [72].*

Chapter 3

Preconditioning based on deflation of linear systems sampled with Markov chains of coefficient fields

3.1 Introduction

Here, we are interested in the case where a sequence $\{\kappa(\cdot,\theta_s)\}_{s=0}^{S}$ of coefficient fields is constructed upon sampling the underlying latent random variables $\{\boldsymbol{\xi}(\theta_s)\}_{s=0}^{S}$ after a Markov chain. Then, we consider the sequence of matrices $\{\mathbf{A}_s\}_{s=0}^{S}$ obtained upon discretization of a PDE for the corresponding sequence of coefficient fields given by $\{\kappa(\cdot,\theta_s)\}_{s=0}^{S}$. We want to solve the linear systems $\mathbf{A}_s\mathbf{u}_s = \mathbf{b}_s$ with a constant preconditioner $\mathbf{M} = \mathbf{L}\mathbf{L}^T$ along with deflation. Let us then consider deflated PCG with a deflation subspace $\mathcal{R}(\mathbf{W}_s)$ and an initial iterate $\mathbf{u}^{(0)}$. In Chapter 2, we derived the equivalence between the iterates of Def-PCG$(\mathbf{A}_s, \mathbf{M}, \mathbf{W}_s, \mathbf{b}_s, \mathbf{u}_s^{(0)})$ and those of CG$(\dot{\mathbf{\Pi}}_s^T\dot{\mathbf{A}}_s, \dot{\mathbf{\Pi}}_s^T\mathbf{L}^{-1}\mathbf{b}_s, \dot{\mathbf{u}}_s^{(0)\prime})$ where

$$\dot{\mathbf{\Pi}}_s = \mathbf{I}_n - \dot{\mathbf{W}}_s(\dot{\mathbf{W}}_s^T\dot{\mathbf{A}}_s\dot{\mathbf{W}}_s)^{-1}\dot{\mathbf{W}}_s^T\dot{\mathbf{A}}_s \tag{3.1}$$

$$\dot{\mathbf{A}}_s = \mathbf{L}^{-1}\mathbf{A}_s\mathbf{L}^{-T} \tag{3.2}$$

$$\dot{\mathbf{W}}_s = \mathbf{L}^T\mathbf{W}_s \tag{3.3}$$

and $\dot{\mathbf{u}}_s^{(0)\prime}$ is some initial iterate. Because of this relation, the behavior of deflated PCG can be analyzed making use of Theo. 2 with $\mathbf{A}_s' := \dot{\mathbf{A}}_s$ and $\mathbf{b}_s' := \dot{\mathbf{\Pi}}_s^T\mathbf{L}^{-1}\mathbf{b}_s$. Then, solving a linear system with Def-PCG$(\mathbf{A}_s, \mathbf{M}, \mathbf{W}_s, \mathbf{b}_s, \mathbf{u}_s^{(0)})$ is equivalent to solving a linear system by CG with a realization-dependent positive semidefinite preconditioner $\mathbf{M}_s = \mathbf{L}_s\mathbf{L}_s^T$ where the inverse of the Cholesky factor of $\mathbf{M}_s$ is given by $\mathbf{L}_s^{-1} = \mathbf{L}^{-1}\mathbf{\Pi}_s$, in which $\mathbf{\Pi}_s = \mathbf{I}_n - \mathbf{W}_s(\mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s)^{-1}\mathbf{W}_s^T\mathbf{A}_s$.

In this Chapter, we look at different ways to construct the deflation basis $\mathbf{W}_{s+1}$ by recycling by-products of Def-PCG$(\mathbf{A}_s, \mathbf{M}, \mathbf{W}_s, \mathbf{b}_s, \mathbf{u}_s^{(0)})$. We consider different constant parallel preconditioners as we investigate the effect of different online approximation methods of eigenvectors on the number of solver iterations.

3.2 Sampling coefficient fields of stochastic PDEs using Monte Carlo Markov chains

Let (Θ, Σ, μ) be a probability triplet and $\Omega \subset \mathbb{R}^d$ be a bounded open domain for $d \in \{1, 2, 3\}$. We want to find $u : \overline{\Omega} \times \Theta \to \mathbb{R}$, such that

$$\nabla \cdot (\kappa(x,\theta)\nabla u(x,\theta)) = -f(x) \; \forall \; x \in \Omega, \tag{3.4}$$

and deterministic boundary conditions for all $x \in \partial\Omega$. For almost all $\theta \in \Theta$, realizations $\kappa(\cdot,\theta)$ of the random coefficient field are strictly positive and bounded above almost everywhere in the domain Ω. Then, the solution u of Eq. (3.4) has finite second order

moments so that

$$\mathbb{E}[u(x)^2] = \int_\Theta u(x,\theta)^2 \mathrm{d}\mu < \infty \tag{3.5}$$

for all $x \in \Omega$, i.e., $u(x,\theta) \in L^2(\Theta,\mu)$.

3.2.1 Spatial discretization

Let the domain Ω be partitioned into a conforming triangulation of N_e non-overlapping elements $\Omega_e \subseteq \Omega$. Each element is paired with nodes of a mesh according to some discretization scheme, and each node i has a position x_i. All the nodes of the mesh such that $x_i \notin \partial\Omega$ form a set $\mathcal{N}$ with $n := |\mathcal{N}|$. A set of nodal functions $\{\phi_i, i \in \mathcal{N}\}$ is defined such that $\phi_i(x_j) = \delta_{ij}$ for all $i,j \in \mathcal{N}$. The span of those nodal basis functions forms the deterministic finite element approximation subspace denoted by V^h. Assuming homogeneous Dirichlet boundary conditions, we consider approximate solutions of u given by

$$u^h(x,\theta) = \sum_{i\in\mathcal{N}} \phi_i(x) u_i(\theta) \in V^h \otimes L^2(\Theta,\mu) \tag{3.6}$$

where $u_{1\leq i\leq n} \in L^2(\Theta,\mu)$. Hence, the approximate solution u^h has finite second order moments.

3.2.2 Stochastic discretization

In the present work, we restrict ourselves to the case of random coefficients with log-normal distributions; denoting $h(x,\theta) = \log\kappa(x,\theta)$ the underlying Gaussian field, the stochastic discretization of κ relies on the Karhunen-Loève [71] expansion of h,

$$h(x,\theta) = \mathbb{E}[h(x,\theta)] + \sum_{l=0}^{\infty} \sqrt{\gamma_l} h_l(x) \xi_l(\theta), \tag{3.7}$$

where (γ_l, h_l) are the eigen-pairs of the covariance function of the Gaussian process. Ordering the normalized eigen-pairs, $\gamma_1 \geq \gamma_2 \geq \cdots \geq 0$, the KL expansion can be truncated to retain the first $n_s > 1$ dominant modes. In addition, the random variables ξ_l are independent and identically distributed standard Gaussian variables: $\boldsymbol{\xi} := (\xi_1 \cdots \xi_{n_s})^T \sim \mathcal{N}(0, \mathbf{I}_{n_s})$. Therefore, the field κ can be sampled by sampling the so-called stochastic coordinates $\boldsymbol{\xi}$.

3.2.3 Sampling of the stochastic coordinates by MCMC

Upon subsequent spatial discretization, the approximation of u reduces to an n-by-n SPD linear system $\mathbf{A}(\boldsymbol{\xi})\mathbf{u}(\boldsymbol{\xi}) = \mathbf{b}(\boldsymbol{\xi})$. In this work, we present and use iterative methods to solve linear systems $\mathbf{A}_s\mathbf{u}_s = \mathbf{b}_s$ in which $\mathbf{A}_s := \mathbf{A}(\boldsymbol{\xi}_s)$, and so on, where $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots$ are samples of $\boldsymbol{\xi} \sim \mathcal{N}(0, \mathbf{I}_{n_s})$. Because the recycling strategy described in Section 3.3 is best behaved when $\mathbf{A}_{s+1}$ and $\mathbf{A}_s$ are similar, we consider the case where $\boldsymbol{\xi}$ is sampled by MCMC using a Gaussian proposal distribution. As a result, $\mathbf{A}_1, \mathbf{A}_2, \ldots$ are correlated matrices,

and a reduction of the variance $\mathbb{V}[(\delta\mathbf{A}_{s+1})_{ij}]$ where $\delta\mathbf{A}_{s+1} := \mathbf{A}_{s+1} - \mathbf{A}_s$ is induced, which does not occur when sampling by standard Monte Carlo. While $\delta\mathbf{A}_{s+1}$ is generally not low rank, the (improved) similarity between $\mathbf{A}_{s+1}$ and $\mathbf{A}_s$ still provides better working conditions for the iterative methods we design to solve the linear systems $\mathbf{A}_s\mathbf{u}_s = \mathbf{b}_s$. The coefficient field being lognormal, we refer to $\widehat{\mathbf{A}} := \mathbf{A}(\mathbf{0})$ as the median operator.

Proposal In order to sample a Markov chain $\{\boldsymbol{\xi}_s\}_{s=0}^S$ we introduce a sequence of proposed states $\{\boldsymbol{\chi}_s\}_{s=1}^{S+1}$ such that $\boldsymbol{\chi}_{s+1} \sim \mathcal{N}(\boldsymbol{\xi}_s, \vartheta^2\mathbf{I}_{n_s})$. Using $\{\boldsymbol{\chi}_s\}_{s=1}^{S+1}$ as proposals for a Metropolis-Hastings algorithm, we obtain the following acceptance probability:

$$\alpha(\boldsymbol{\chi}_{s+1}, \boldsymbol{\xi}_s) = \min\left\{\exp\left[\frac{\|\boldsymbol{\xi}_s\|_2^2 - \|\boldsymbol{\chi}_{s+1}\|_2^2}{2}\right], 1\right\}. \tag{3.8}$$

We can then proceed as detailed in Algo. 22. As a result of the sampling strategy,

Algorithm 22 Metropolis-Hastings algorithm for random walk proposals.

Draw $\boldsymbol{\xi}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{n_s})$
for $s = 0, 1, 2, \ldots$ **do**
  Draw $\boldsymbol{\chi}_{s+1} \sim \mathcal{N}(\boldsymbol{\xi}_s, \vartheta^2\mathbf{I}_{n_s})$
  Compute $\alpha(\boldsymbol{\chi}_{s+1}, \boldsymbol{\xi}_s)$
  Draw $z \sim \mathcal{U}[0, 1]$
  if $z < \alpha(\boldsymbol{\chi}_{s+1}, \boldsymbol{\xi}_s)$ **then**
    $\boldsymbol{\xi}_{s+1} := \boldsymbol{\chi}_{s+1}$
  else
    $\boldsymbol{\xi}_{s+1} := \boldsymbol{\xi}_s$
  end if
end for

$\mathbf{A}_1, \mathbf{A}_2, \ldots$ are correlated matrices.

3.2.4 Properties and comparison of MCMC with MC sampler

3.2.4.1 Comparison of MCMC with (non-deflated) Monte Carlo simulations

Another reason for sampling the coefficient field by MCMC is to provide an alternative to (non-deflated) Monte Carlo simulations of the stochastic PDE. For instance, say we are interested in the estimator of $\mathbb{E}[u^h(x)]$ given by

$$\overline{u}^S(x) := \frac{1}{S+1}\sum_{s=0}^{S}\sum_{i=1}^{n} u_s^{(i)}\phi_i(x) \tag{3.9}$$

where $\mathbf{u}_s = [u_s^{(1)}, \ldots, u_s^{(n)}]^T$ is the solution of $\mathbf{A}_s\mathbf{u}_s = \mathbf{b}_s$. We sample sequences $\{\boldsymbol{\xi}_s\}_{s=0}^S$ using both Monte Carlo (MC) and MCMC strategies.

MC sampler Let the sequence $\{\boldsymbol{\xi}_s\}_{s=0}^{M_{\mathrm{MC}}}$ be sampled such that $\boldsymbol{\xi}_s \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{n_s})$ are iid. If so, since for all $1 \leq i \leq n$, $\{u_s^{(i)}\}_{s=0}^{M_{\mathrm{MC}}}$ are replicates of the random variable u_i which is

distributed as $u^h(x_i)$, the following limit theorem holds for all $x \in \Omega$:

$$\sqrt{S+1}(\overline{u}^S(x) - \mathbb{E}[u^h(x)]) \xrightarrow{d} \mathcal{N}(0, \mathbb{V}[u^h(x)]) \text{ as } S \to \infty. \tag{3.10}$$

Therefore, for sufficiently large S, we have $\mathbb{V}[\overline{u}^S(x) - \mathbb{E}[\overline{u}^S(x)]] = \mathbb{V}[u^h(x)]/(S+1)$, which can be used to suggest a MC sample size. Precisely, S_{MC} is selected as the smallest S such that $\mathbb{V}[\overline{u}^S(x) - \mathbb{E}[u^h(x)]] < r^2\mathbb{V}[u^h(x)]$ for all $x \in \Omega$, where $0 < r < 1$ is a fixed relative tolerance on the standard error. In other words, we take $S_{\text{MC}} = \lceil 1/r^2 \rceil$. In general, the sample set size is selected to ensue an error smaller than some $r^2\mathbb{V}[u^h(x)]$ with a prescribed probability.

MCMC sampler Since the elements of the Markov chain are correlated, standard limit theorems do not apply. However, $\{\boldsymbol{\xi}_s\}_{s=0}^S$ was proven to be geometrically ergodic, see [134], and the following limit theorem applies, see Corollary 4 in [69]:

$$\sqrt{S+1}(\overline{u}^S(x) - \mathbb{E}[u^h(x)]) \xrightarrow{d} \mathcal{N}(0, \sigma_f^2(x)) \text{ as } S \to \infty \tag{3.11}$$

where

$$\sigma_f^2(x) := \mathbb{V}[u_0(x)] + 2\sum_{s=1}^{\infty} \mathbf{Cov}[u_0(x), u_s(x)] < \infty, \tag{3.12}$$

so that, for sufficiently large S, we have $\mathbb{V}[\overline{u}^S(x) - \mathbb{E}[u^h(x)]] = \sigma_f^2(x)/(S+1)$, which can be used to suggest a MCMC sampled size. We wish to apply the same relative tolerance on the standard error as for MC, namely $\mathbb{V}[\overline{u}^S(x) - \mathbb{E}[u^h(x)]] < r^2\mathbb{V}[u^h(x)]$ for some $x \in \Omega$. Therefore, neglecting the burn-in period, this leads to $S_{\text{MCMC}} = \sigma_f^2(x)/(r^2\mathbb{V}[u^h(x)])$. For a good reference on the burn-in and convergence issues in MCMC algorithms, including how to diagnose convergence and estimate the required burn-in period, see Chapter 4 in [48].

MCMC sampling overhead In order to compare the relative performance of the sampling strategies, we need to quantify the number of distinct linear systems $\mathbf{A}_s\mathbf{u}_s = \mathbf{b}_s$ solved to evaluate an instance of the estimator (3.9) with the same precision. In case of MC sampling, we solve as many linear systems as there are realizations in a sequence, i.e., S_{MC}. However, when using the Markov chain $\{\boldsymbol{\xi}_s\}_{s=0}^{S_{\text{MCMC}}}$, the number of solves differs from S_{MCMC} for the following reason. Every time a proposed state $\boldsymbol{\chi}_s$ is rejected, a new realization is sampled at the expense of no computation since we then have $\mathbf{u}_{s+1} = \mathbf{u}_s$. Consequently, the number of distinct linear systems actually solved when sampling by MCMC is $p_a S_{\text{MCMC}}$ where p_a is usually called the acceptance ratio. We then define the relative sampling overhead of MCMC over MC as follows:

$$\gamma(x) := \frac{p_a S_{\text{MCMC}}}{S_{\text{MC}}} = \frac{p_a \sigma_f^2(x)}{\mathbb{V}[u^h(x)]}. \tag{3.13}$$

Note that there is no statistical advantage of sampling several independent chains whose total numbers of steps add up to the number of steps of a single chain.

3.2.4.2 Properties of the Markov chain

The MCMC sampling by random walk of the stochastic coordinates of the KL representation of the coefficient field presents two degrees of freedoms that can be adjusted. First, the variance ϑ^2 of the proposal affects the correlation between subsequent matrices. This variance can be increased, in which case the statistical cost of sampling with correlation is lowered, i.e., but the increase of ϑ^2 also leads to a decrease of the ratio p_a, and the similarity between the eigenvectors of the subsequent matrices is diminished. On the other hand, lowering this variance introduce more similarity between the eigenvectors of subsequent matrices at the cost of an increased statistical cost, i.e., through an increase of σ_f^2. Another parameter of interest is the number of random variables sampled by MCMC among the stochastic coordinates of the KL expansion. That is to say, not all the coordinates have to be sampled by MCMC, only the dominant ones, those remaining associated with smaller eigenvalues may be sampled by Monte Carlo. This hybrid sampling strategy presents the advantage of decreasing the statistical cost of sampling RVs by MCMC, but it maintains some similarity between the dominant eigenvectors of subsequent matrices. Here, although we did experiments towards tuning the variance of the proposal as well as the number of coordinates sampled by MCMC, we only present results for sampling with a variance $\vartheta^2 = 2.38^2/n_s$ where n_s denotes the total number of coordinates of the truncated KL expansion. This choice of variance comes from the fact that it yields an asymptotical optimal acceptance rate in the case of Metropolis-Hastings algorithms, see [130] and [132], for a multivariate Gaussian distribution with covariance $\boldsymbol{\Sigma} := \mathbf{I}_{n_s}$.

3.2.5 Posterior sampling

A situation that calls for sampling the coefficient field by MCMC is Bayesian inference. That is, let us consider a deterministic version of the problem of interest,

$$\nabla \cdot (\kappa(x)\nabla u(x)) = -f(x) \;\; \forall\, x \in \Omega, \tag{3.14}$$

with boundary conditions defined for all $x \in \partial\Omega$. We are interested in the case where m noisy measurements $d_1, \ldots, d_m$ of the solution $u : \Omega \to \mathbb{R}$ are made at $x_1^*, \ldots, x_m^* \in \Omega$. We assume $\varepsilon > 0$ is known such that $d_i \sim \mathcal{N}(u(x_i^*), \varepsilon^2)$, and further assume prior knowledge about the coefficient $\kappa : \Omega \to \mathbb{R}^+$ in the form of a mapping $(x, \boldsymbol{\xi}) \mapsto \kappa(x|\boldsymbol{\xi})$ with $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{n_s})$ as in Eq. (3.7). As a means to quantify the uncertainty of the coefficient, κ can be sampled through a posterior distribution given by the following Bayes' formula:

$$p(\boldsymbol{\xi}|\mathbf{d}) = \frac{p(\mathbf{d}|\boldsymbol{\xi})p(\boldsymbol{\xi})}{p(\mathbf{d})} \tag{3.15}$$

in which $\mathbf{d} := [d_1, \ldots, d_m]^T$ is the noisy data. The evaluation of the marginal likelihood $p(\mathbf{d})$ in Eq. (3.15) becomes increasingly complex as the dimension of $\boldsymbol{\xi}$ grows. Hence, for high-dimensional stochastic discretizations of the prior, the posterior distribution $p(\boldsymbol{\xi}|\mathbf{d})$ is generally sampled by MCMC, making use of the following relation:

$$p(\boldsymbol{\xi}|\mathbf{d}) \propto p(\mathbf{d}|\boldsymbol{\xi})p(\boldsymbol{\xi}) \propto \exp(-\|\mathbf{d} - \mathbf{u}^*(\boldsymbol{\xi})\|_2^2/(2\varepsilon^2) - \|\boldsymbol{\xi}\|_2^2/2) \tag{3.16}$$

where $\mathbf{u}^*(\boldsymbol{\xi}) := [u(x_1^*|\boldsymbol{\xi}), \ldots, u(x_m^*|\boldsymbol{\xi})]^T$ is the solution of Eq. (3.14) evaluated at $x_1^*, \ldots, x_m^*$ given $\kappa(\cdot|\boldsymbol{\xi})$. In this case, the acceptance probability is given by

$$\alpha(\boldsymbol{\chi}_{s+1}, \boldsymbol{\xi}_s) = \exp\left[\frac{\|\mathbf{d} - \mathbf{u}^*(\boldsymbol{\chi}_{s+1})\|_2^2 - \|\mathbf{d} - \mathbf{u}^*(\boldsymbol{\xi}_s)\|_2^2}{2\varepsilon^2} + \frac{\|\boldsymbol{\chi}_{s+1}\|_2^2 - \|\boldsymbol{\xi}_s\|_2^2}{2}\right]. \tag{3.17}$$

and Algo. 22 is modified accordingly. Note that the evaluation of the acceptance probability α requires a linear solve for each $\boldsymbol{\chi}_s$.

3.3 Recycling strategy

As we intend to solve the sampled linear systems either by Def-CG, or by Init-CG, a procedure needs to be defined in order to compute $\mathbf{W}_{s+1}$ prior to solving $\mathbf{A}_{s+1}\mathbf{u}_{s+1} = \mathbf{b}_{s+1}$. For this, a key observation is that a basis of $\mathcal{K}^{(j)}(\mathbf{A}_s, \mathbf{W}_s, \mathbf{r}_s^{(0)})$ can be recycled from the linear solve of $\mathbf{A}_s\mathbf{u}_s = \mathbf{b}_s$. Moreover, different quantities needed to construct projection-based approximations of the eigenvectors of $\mathbf{A}_{s+1}$ or $\mathbf{M}^{-1}\mathbf{A}_{s+1}$ in these subspaces, such as the tridiagonalization of $\mathbf{A}_{s+1}$, are readily accessible byproducts of the linear solver. Therefore, approximate eigenpairs of $\mathbf{A}_s$ or $\mathbf{M}^{-1}\mathbf{A}_s$ can be promptly generated while solving $\mathbf{A}_s\mathbf{u}_s = \mathbf{b}_s$. Now, because the operators sampled by MCMC are correlated, the eigenvectors of $\mathbf{A}_{s+1}$ (resp. $\mathbf{M}^{-1}\mathbf{A}_{s+1}$) are correlated with those of $\mathbf{A}_s$ (resp. $\mathbf{M}^{-1}\mathbf{A}_s$). For this reason, we let the columns of $\mathbf{W}_{s+1}$ be approximate eigenvectors of $\mathbf{A}_s$ or of $\mathbf{M}^{-1}\mathbf{A}_s$.

Another incentive for deflating with LD eigenvector approximates is given as follows. The LD eigenvectors correspond to low frequency modes of the sampled operators, as opposed to the MD eigenvectors, which capture more detailed features of the solution. It was observed that the LD eigenvectors of subsequent matrices sampled by MCMC are more correlated than the MD ones. For these reasons, we let the columns of $\mathbf{W}_{s+1}$ be projection-based approximations of LD eigenvectors of $\mathbf{A}_s$ or $\mathbf{M}^{-1}\mathbf{A}_s$. Note that this strategy is analogous to the approach used in [122].

3.4 Online approximation of eigenvectors

3.4.1 Projection techniques without preconditioner

We only intend to approximate a small number $k \ll n$ of the LD eigenvectors of every sampled operator $\mathbf{A}_s$. Iterative methods based on projection techniques are well suited for this task [139]. In particular, given a full column rank matrix $\mathbf{V}_s \in \mathbb{R}^{n\times m}$ with $k \leq m < n$, we consider both Rayleigh-Ritw (RR) and harmonic Rayleigh-Ritz (HR) projection techniques with respect to the eigen-search space $\mathcal{R}(\mathbf{V}_s)$, an augmented Krylov subspace of $\mathbf{A}_s$ spanned by residual (or search direction) vectors of the linear solver in addition to approximate eigenvectors of $\mathbf{A}_s$

RR procedures are perhaps the most commonly used projection-based techniques for eigenvector computation which are well known to provide optimal eigenvalue approximations at the extremities of the spectrum [124]. They are defined as follows.

RR projection A RR vector $\mathbf{w}$ of $\mathbf{A}_s$ with respect to $\mathcal{R}(\mathbf{V}_s)$ is such that $\mathbf{A}_s\mathbf{w} - \vartheta\mathbf{w} \perp \mathcal{R}(\mathbf{V}_s)$ for some ϑ. As the RR vector is recast as $\mathbf{w} := \mathbf{V}_s\hat{\mathbf{w}}$, the pair $(\vartheta, \hat{\mathbf{w}})$ becomes solution of the reduced (generalized) eigenvalue problem

$$\mathbf{V}_s^T\mathbf{A}_s\mathbf{V}_s\hat{\mathbf{w}} = \vartheta\mathbf{V}_s^T\mathbf{V}_s\hat{\mathbf{w}}. \tag{3.18}$$

HR procedures were introduced as an alternative to RR projections in order to better approximate the interior eigenvalues of Hermitian operators [103, 119]. They are now commonly defined as follows.

HR projection A HR vector $\mathbf{w}$ of $\mathbf{A}_s$ with respect to $\mathcal{R}(\mathbf{V}_s)$ is such that $\mathbf{A}_s\mathbf{w} - \vartheta\mathbf{w} \perp \mathcal{R}(\mathbf{A}_s\mathbf{V}_s)$ for some ϑ. As the HR vector is recast as $\mathbf{w} := \mathbf{V}_s\hat{\mathbf{w}}$, the pair $(\vartheta, \hat{\mathbf{w}})$ becomes solution of the following reduced generalized eigenvalue problem:

$$(\mathbf{A}_s\mathbf{V}_s)^T\mathbf{A}_s\mathbf{V}_s\hat{\mathbf{w}} = \vartheta(\mathbf{A}_s\mathbf{V}_s)^T\mathbf{V}_s\hat{\mathbf{w}}. \tag{3.19}$$

In this work, we let the columns of $\mathbf{V}_s$ be previous eigenvector approximations, if any, along with search directions or residuals generated by the linear solve. Approximations of the k LD eigenvectors of $\mathbf{A}_s$ are formed by, first, solving for the k LD eigenpairs $\{(\vartheta_i, \hat{\mathbf{w}}_i)\}_{i=1}^k$ of the reduced (RR or HR) eigenvalue problem, and then, letting $\mathbf{W}_{s+1} := \mathbf{V}_s[\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_k]$.

A procedure based on HR projections was favored in [140] to approximate the LD eigenvectors of SPD matrices. Notwithstanding, whether or not HR vectors better approximate the LD eigenvectors of an SPD matrix than their RR counterparts remains, to our knowledge, an open question. Irrespective of which projection technique is used, high-dimensional eigen-search spaces are generally needed in order to obtain accurate eigenvector approximations. However, since the computed approximate eigenvectors of $\mathbf{A}_s$ are only used to deflate $\mathbf{A}_{s+1}$, the need for highly accurate approximations is not justified in the context of this study.

3.4.2 Restarting the eigen-search space without preconditioner

Although the computed vectors need not be exact eigenvectors of $\mathbf{A}_s$, the better these vectors approximate the eigenvectors of $\mathbf{A}_{s+1}$, the better the convergence behaviors of Def-CG [72] and Init-CG [55]. Up to some level, which depends on $\delta\mathbf{A}_{s+1}$, this may be achieved by increasing the dimension of the eigen-search space, which might cause memory problems because $\mathbf{V}_s$, which is dense, needs to be stored in order to compute $\mathbf{W}_{s+1}$. Moreover, due to the effects of finite arithmetic, the linear independence of the recycled residuals (or search directions) tends not to hold anymore as the dimension of the eigen-search space increases. This, in turn, can cause a loss of rank in the matrices of the reduced eigenvalue problem, eventually leading to spurious eigenvector approximations [121]. As a means to circumvent these difficulties, the recycled vectors can be orthogonalized, at a computational cost which increases quadratically with the dimension of the eigen-search space. An alternative which allows for a better use of computational resources is to periodically restart the eigen-search space with updated eigenvector approximations of $\mathbf{A}_s$, every time the dimension of the eigen-search space reaches a certain limit. We refer to these approaches as restarted methods. To avoid any confusion, note that the linear

solve is *not* restarted, as opposed to what is done when restarting FOM or GMRES [138].

3.4.2.1 Thick-restart (TR)

The term *thick-restart* was first coined in [165] to refer to an explicitly restarted Lanczos procedure which we summarized in Section 1.5.1.1. However, the thick-restart Lanczos method of [165] is a dedicated eigensolver, it does not solve a linear system. Here, we merely intend to make use of all the residuals (or search directions) generated by the linear solver without applying an augmented Krylov subspace strategy to the eigenvector approximations, as this would entail an unwanted restart of the linear solver. Therefore, the eigen-search space is simply *restarted* by appending the residuals (or search directions) of the linear solver to the most recently computed eigenvector approximations. The resulting strategy, referred to as thick-restart (TR) and summarized in Algo. 23, is run concurrently with CG, DEF-CG or INIT-CG. As a result, the restarted eigen-search space does not remain a Krylov subspace, and the convenient property of thick-restart Lanczos is not satisfied. Nevertheless, this strategy does allow to compute eigenvector approximations of $\mathbf{A}_s$ by making use of all the residuals (or search directions) generated by the linear solver while avoiding the difficulties described in Section 3.4.2.

Algorithm 23 Thick-restart (TR) of the eigen-search space

if $s = 1$ **then**
  $\mathbf{V}_s := [\,]$, $d := 0$
else if $s > 1$ **then**
  $\mathbf{V}_s := \mathbf{W}_s$, $d := k$
end if
for $j = 0, 1, \ldots$ **do**
  $\mathbf{V}_s := [\mathbf{V}_s, \mathbf{r}_s^{(j)}]$, $d := d + 1$ ▷ or $\mathbf{V}_s := [\mathbf{V}_s, \mathbf{p}_s^{(j)}]$
  if $d = spdim$ **then**
    Compute the k LD RR (or HR) vectors $\{\mathbf{w}_i\}_{i=1}^{k}$ of $\mathbf{A}_s$ w.r.t. $\mathcal{R}(\mathbf{V}_s)$
    $\mathbf{V}_s := [\mathbf{w}_1, \ldots, \mathbf{w}_k]$, $d := k$
  end if
end for
$\mathbf{W}_{s+1} := [\mathbf{w}_1, \ldots, \mathbf{w}_k]$

3.4.2.2 Locally optimal thick-restart (LO-TR)

An alternative use of all the information generated by the linear solver is described in Algo. 24, which we analogously refer to as a locally optimal thick-restart (LO-TR) of the eigen-search space. LO-TR essentially works as follows. Every time the eigen-search space $\mathcal{R}(\mathbf{V}_s)$ reaches a dimension $spdim$, it is used to generate eigenvector approximations $\mathbf{y}_1, \ldots, \mathbf{y}_k$, whereas additional eigenvector approximations $\overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_k$ are computed with respect to $\mathcal{R}(\overline{\mathbf{V}}_s)$, where $\overline{\mathbf{V}}_s := \mathbf{V}_s[:, 1 : spdim - 1]$ consists of all but the last column of $\mathbf{V}_s$. All the eigenvector approximations of $\mathbf{A}_s$ with respect to $\mathcal{R}([\mathbf{y}_1, \overline{\mathbf{y}}_1, \ldots, \mathbf{y}_k, \overline{\mathbf{y}}_k])$ are then used to restart the eigen-search space. When applied concurrently with INIT-CG using RR projections, this strategy is equivalent to eigCG [148]. Not only does this method outperforms thick-restart Lanczos procedures for eigenvalue approximation, but

it was also shown to perform as well as orthogonalized un-restarted Lanczos procedures under favorable conditions [148].

Algorithm 24 Locally optimal thick-restart (LO-TR) of the eigen-search space

if $s = 1$ **then**
  $\mathbf{V}_s := [\,]$, $d := 0$
else if $s > 1$ **then**
  $\mathbf{V}_s := \mathbf{W}_s$, $d := k$
end if
for $j = 0, 1, \ldots$ **do**
  $\mathbf{V}_s := [\mathbf{V}_s, \mathbf{r}_s^{(j)}]$, $d := d + 1$ ▷ or $\mathbf{V}_s := [\mathbf{V}_s, \mathbf{p}_s^{(j)}]$
  if $d = spdim$ **then**
    $\overline{\mathbf{V}}_s := \mathbf{V}_s[:, 1 : spdim - 1]$ ▷ $\overline{\mathbf{V}}_s$ contains all but the last column of $\mathbf{V}_s$
    Compute the k LD RR (or HR) vectors $\{\mathbf{y}_i\}_{i=1}^{k}$ of $\mathbf{A}_s$ w.r.t. $\mathcal{R}(\mathbf{V}_s)$
    Compute the k LD RR (or HR) vectors $\{\overline{\mathbf{y}}_i\}_{i=1}^{k}$ of $\mathbf{A}_s$ w.r.t. $\mathcal{R}(\overline{\mathbf{V}}_s)$
    Get an orthonormal basis $\mathbf{Q} := [\mathbf{q}_1, \ldots, \mathbf{q}_{nvec}]$ of $\mathcal{R}([\mathbf{y}_1, \overline{\mathbf{y}}_1, \ldots, \mathbf{y}_k, \overline{\mathbf{y}}_k])$
    Compute the RR (or HR) vectors $\{\mathbf{w}_i\}_{i=1}^{nvec}$ of $\mathbf{A}_s$ w.r.t. $\mathcal{R}(\mathbf{Q})$
    $\mathbf{V}_s := [\mathbf{w}_1, \ldots, \mathbf{w}_{nvec}]$, $d := nvec$
  end if
end for
$\mathbf{W}_{s+1} := [\mathbf{w}_1, \ldots, \mathbf{w}_k]$

To the best of our knowledge, among all the restarting strategies described in this Section, RR-LO-TR-Init-CG (i.e. eigCG) is the only one documented in the literature.

3.4.3 Projection techniques with preconditioner

Let us extract sequences of preconditioned residuals $\mathbf{z}_s^{(j)}$ (or search directions $\mathbf{p}_s^{(j)}$) from the linear solver and store them by columns in $\mathbf{V}_s$, along with $\mathbf{W}_s$ which may be restarted or not. From hereon, as explained before, there are two options. First, we can let the deflation subspace be spanned by approximate eigenvectors of $\mathbf{A}_s$, and second, by eigenvectors of $\mathbf{M}^{-1}\mathbf{A}_s$. Here, we present the projection methods with respect to $\mathbf{V}_s$ used for both cases.

3.4.3.1 Approximate eigenvectors of $\mathbf{A}_s$

In this case, we consider the eigen-search space $\mathcal{R}(\mathbf{V}_s)$ spanned by preconditioned residuals (or search directions) and which consists of a Krylov subspace of $\mathbf{M}^{-1}\mathbf{A}_s$ augmented by $\mathcal{R}(\mathbf{W}_s)$. As for the non-preconditioned case, we construct both RR and HR projection-based eigenvector approximations of $\mathbf{A}_s$ with respect to $\mathcal{R}(\mathbf{V}_s)$. Similarly as before, these eigenvector approximations are recast as $\mathbf{V}_s\hat{\mathbf{w}}$ where $\hat{\mathbf{w}}$ is the eigenvector of a reduced eigenvalue problem given by Eqs. (3.18) and (3.19) for RR and HR projections, respectively. It is worth noting that the eigen-search space $\mathcal{R}(\mathbf{V}_s) = \mathcal{K}^{(j)}(\mathbf{M}^{-1}\mathbf{A}_s, \mathbf{M}^{-1}\mathbf{r}^{(0)}) \oplus \mathcal{R}(\mathbf{W}_s)$ is not an augmented Krylov subspace of the matrix $\mathbf{A}_s$ of which we are approximating the eigenvectors, but rather of $\mathbf{M}^{-1}\mathbf{A}_s$.

3.4.3.2 Approximate eigenvectors of $\mathbf{M}^{-1}\mathbf{A}_s$

Here, we set $\dot{\mathbf{V}}_s := \mathbf{L}^T\mathbf{V}_s$ which allows us to construct an eigen-search space $\mathcal{R}(\dot{\mathbf{V}}_s)$ consisting of a Krylov subspace of $\dot{\mathbf{A}}_s$ augmented by $\mathcal{R}(\dot{\mathbf{W}}_s)$. We then intend to compute approximations $\{\mathbf{w}_i\}_{i=1}^k$ of the k LD right eigenvectors of $\mathbf{M}^{-1}\mathbf{A}_s$, which we store by columns in $\mathbf{W}_{s+1}$. This is done by first computing LD approximate eigenvectors $\dot{\mathbf{w}}_i$ of $\dot{\mathbf{A}}_s$ with respect to $\mathcal{R}(\dot{\mathbf{V}}_s)$, before setting $\mathbf{w}_i := \mathbf{L}^{-T}\dot{\mathbf{w}}_i$. As we proceed to do so using RR and HR projections, we note that the Cholesky factor $\mathbf{L}$ is in fact not needed in either of the resulting procedures.

RR projection Let $\dot{\mathbf{w}}$ be a RR eigenvector approximation of $\dot{\mathbf{A}}_s$ with respect to $\mathcal{R}(\dot{\mathbf{V}}_s)$. We then search for $\dot{\mathbf{w}} \in \mathcal{R}(\dot{\mathbf{V}}_s)$ such that $\dot{\mathbf{A}}_s\dot{\mathbf{w}} - \vartheta\dot{\mathbf{w}} \perp \mathcal{R}(\dot{\mathbf{V}}_s)$, for some ϑ. Recasting $\dot{\mathbf{w}}$ as $\dot{\mathbf{V}}_s\hat{\mathbf{w}}$, we obtain $\dot{\mathbf{V}}_s^T\dot{\mathbf{A}}_s\dot{\mathbf{V}}_s\hat{\mathbf{w}} = \vartheta\dot{\mathbf{V}}_s^T\dot{\mathbf{V}}_s\hat{\mathbf{w}}$, which can be recast as

$$\mathbf{V}_s^T\mathbf{A}_s\mathbf{V}_s\hat{\mathbf{w}} = \vartheta\mathbf{V}_s^T\mathbf{M}\mathbf{V}_s\hat{\mathbf{w}}. \tag{3.20}$$

Then, we have $\mathbf{w} = \mathbf{L}^{-T}\dot{\mathbf{V}}_s\hat{\mathbf{w}} = \mathbf{V}_s\hat{\mathbf{w}}$, so that this procedure is equivalent to searching for $\mathbf{w} \in \mathcal{R}(\mathbf{V}_s)$ such that $\mathbf{A}_s\mathbf{w} - \vartheta\mathbf{M}\mathbf{w} \perp \mathcal{R}(\mathbf{V}_s)$. Note that, in practice, $\mathbf{M}$ does not need to be applied to assemble the matrix on the right hand side of Eq. (3.20), as explained in Section 3.5.3.

HR projection Alternatively, Saad et al. [140] consider $\dot{\mathbf{w}}$ to be an HR eigenvector approximation of $\dot{\mathbf{A}}_s$ with respect to $\mathcal{R}(\dot{\mathbf{V}}_s)$. Then, $\dot{\mathbf{w}} \in \mathcal{R}(\dot{\mathbf{V}}_s)$ is such that $\dot{\mathbf{A}}_s\dot{\mathbf{w}} - \vartheta\dot{\mathbf{w}} \perp \mathcal{R}(\dot{\mathbf{A}}_s\dot{\mathbf{V}}_s)$ for some ϑ. Recasting $\dot{\mathbf{w}}$ as $\dot{\mathbf{V}}_s\hat{\mathbf{w}}$ leads to a reduced eigenvalue problem $(\dot{\mathbf{A}}_s\dot{\mathbf{V}}_s)^T\dot{\mathbf{A}}_s\dot{\mathbf{V}}_s\hat{\mathbf{w}} = \vartheta(\dot{\mathbf{A}}_s\dot{\mathbf{V}}_s)^T\dot{\mathbf{V}}_s\hat{\mathbf{w}}$, which can be recast as

$$(\mathbf{A}_s\mathbf{V}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{V}_s\hat{\mathbf{w}} = \vartheta(\mathbf{A}_s\mathbf{V}_s)^T\mathbf{V}_s\hat{\mathbf{w}}. \tag{3.21}$$

Then, once again, we have $\mathbf{w} = \mathbf{V}_s\hat{\mathbf{w}}$ so that this procedure is equivalent to searching for $\mathbf{w} \in \mathcal{R}(\mathbf{V}_s)$ such that $\mathbf{M}^{-1}\mathbf{A}_s\mathbf{w} - \vartheta\mathbf{w} \perp \mathcal{R}(\mathbf{A}_s\mathbf{V}_s)$.

In summary, $\mathbf{W}_{s+1}$ is computed by first solving for the k LD eigenpairs $\{(\vartheta_i, \hat{\mathbf{w}}_i)\}_{i=1}^k$ of the reduced (RR or HR) eigenvalue problem, and then, letting $\mathbf{W}_{s+1} := \mathbf{V}_s[\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_k]$. Finally, the restarting strategies of Section 3.4.2 are applied similarly using the projection methods described this Section.

3.5 Practical considerations

We assume that $k \ll n$ so that the extra computational cost of Def-CG compared to CG is mainly due to the computation of $\mathbf{\Pi}_s\mathbf{r}_s^{(j)}$, which can be done at a cost of $\mathcal{O}(kn)$ per iteration when both $\mathbf{W}_s$ and $\mathbf{A}_s\mathbf{W}_s$ are stored [140], sometimes along with $\mathbf{A}_s\mathbf{Z}_s$ when using a preconditioner, where $\mathbf{Z}_s$ stores preconditioned residuals by column. However, Init-CG, which is CG with a particular initial guess, only requires one extra computation to set $\mathbf{u}_s^{(0)}$ such that $\mathbf{r}_s^{(0)} \perp \mathcal{R}(\mathbf{W}_s)$. Hence, there is an incentive to investigate whether the convergence behavior of Def-CG can be reproduced by Init-CG, when applied to sequences of correlated operators.

3.5.1 Assembly of the reduced eigenvalue problem without preconditioner

The assembly of the reduced eigenvalue problem depends on the type of projection (RR or HR) used, as well as on whether residuals or search directions are stored in $\mathbf{V}_s$. Indeed, although Algos. 23 and 24 both suggest to extract residuals from the solver run, these vectors can be replaced by search directions as they both span the same subspace Krylov subspace. Irrespective of which approach is taken, the matrices $\mathbf{A}_s\mathbf{W}_s$ and $\mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s$ need to be computed prior to assembly.

RR projection For RR projections, the cost of assembly is minimized when we extract sequences of $\ell := spdim - k$ normalized residuals $\mathbf{r}_s^{(j)}/\|\mathbf{r}_s^{(j)}\|_2$, which we store by columns in $\mathbf{R}_s$. Then, we have $\mathbf{V}_s = [\mathbf{W}_s, \mathbf{R}_s]$ so that Eq. (3.18) becomes:

$$\begin{bmatrix} \mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{W}_s^T\mathbf{A}_s\mathbf{R}_s \\ \mathbf{R}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{R}_s^T\mathbf{A}_s\mathbf{R}_s \end{bmatrix} \hat{\mathbf{w}} = \vartheta \begin{bmatrix} \mathbf{I}_k & \mathbf{W}_s^T\mathbf{R}_s \\ \mathbf{R}_s^T\mathbf{W}_s & \mathbf{I}_\ell \end{bmatrix} \hat{\mathbf{w}} \tag{3.22}$$

where $\mathbf{R}_s^T\mathbf{A}_s\mathbf{R}_s$ is tridiagonal with coefficients which can be recovered as a byproduct of the solver. Assuming no restart, if the linear solver reaches *spdim* iterations, we have

$$\mathbf{R}_s^T\mathbf{A}_s\mathbf{R}_s = \begin{bmatrix} \frac{1}{\alpha_s^{(0)}} & \frac{\sqrt{\beta_s^{(0)}}}{\alpha_s^{(0)}} & & & & \\ \frac{\sqrt{\beta_s^{(0)}}}{\alpha_s^{(0)}} & \frac{1}{\alpha_s^{(1)}} + \frac{\beta_s^{(0)}}{\alpha_s^{(0)}} & \frac{\sqrt{\beta_s^{(1)}}}{\alpha_s^{(1)}} & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \frac{\sqrt{\beta_s^{(spdim-2)}}}{\alpha_s^{(spdim-2)}} & \\ & & & \frac{\sqrt{\beta_s^{(spdim-2)}}}{\alpha_s^{(spdim-2)}} & \frac{1}{\alpha_s^{(spdim-1)}} + \frac{\beta_s^{(spdim-2)}}{\alpha_s^{(spdim-2)}} \end{bmatrix}, \tag{3.23}$$

see Section 1.5.2.1. For cases with restarts, analogous matrix components are straightforwardly defined. Computing the product $\mathbf{W}_s^T\mathbf{A}_s\mathbf{R}_s$ can be done column-by-column as the residuals are being generated by the solver, for a total cost of $\mathcal{O}(k\ell n)$. The off-diagonal block $\mathbf{W}_s^T\mathbf{R}_s$ vanishes as a property of DEF-CG. For INIT-CG, this term vanishes if $\mathcal{R}(\mathbf{W}_s)$ is invariant under the action of $\mathbf{A}_s$, which would be the case if the columns of $\mathbf{W}_s$ were exact eigenvectors of $\mathbf{A}_s$. Note that this term is ignored in [148].

HR projection When using HR projections, a simpler problem is obtained when extracting sequences of ℓ search directions $\mathbf{p}_s^{(j)}$, which we store by columns in $\mathbf{P}_s$. Then, we have $\mathbf{V}_s = [\mathbf{W}_s, \mathbf{P}_s]$ so that Eq. (3.19) is recast as:

$$\begin{bmatrix} (\mathbf{A}_s\mathbf{W}_s)^T\mathbf{A}_s\mathbf{W}_s & (\mathbf{A}_s\mathbf{W}_s)^T\mathbf{A}_s\mathbf{P}_s \\ (\mathbf{A}_s\mathbf{P}_s)^T\mathbf{A}_s\mathbf{W}_s & (\mathbf{A}_s\mathbf{P}_s)^T\mathbf{A}_s\mathbf{P}_s \end{bmatrix} \hat{\mathbf{w}} = \vartheta \begin{bmatrix} \mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{W}_s^T\mathbf{A}_s\mathbf{P}_s \\ \mathbf{P}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{P}_s^T\mathbf{A}_s\mathbf{P}_s \end{bmatrix} \hat{\mathbf{w}} \tag{3.24}$$

in which the computation of $(\mathbf{A}_s\mathbf{W}_s)^T\mathbf{A}_s\mathbf{W}_s$ costs $\mathcal{O}(k^2n)$. The matrices $(\mathbf{A}_s\mathbf{P}_s)^T\mathbf{A}_s\mathbf{P}_s$ and $\mathbf{P}_s^T\mathbf{A}_s\mathbf{P}_s$ are tridiagonal and diagonal, respectively, and may be recovered as byproducts of the solver. Assuming no restart, if the linear solver reaches *spdim* iterations, we

have (see [140])

$$(\mathbf{A}_s\mathbf{P}_s)^T\mathbf{A}_s\mathbf{P}_s = \begin{bmatrix} \frac{d_s^{(0)}}{\alpha_s^{(0)}}(1+\beta_s^{(0)}) & -\frac{d_s^{(1)}}{\alpha_s^{(0)}} & & & & \\ -\frac{d_s^{(1)}}{\alpha_s^{(0)}} & \frac{d_s^{(1)}}{\alpha_s^{(1)}}(1+\beta_s^{(1)}) & -\frac{d_s^{(2)}}{\alpha_s^{(1)}} & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & -\frac{d_s^{(spdim-1)}}{\alpha_s^{(spdim-2)}} & \\ & & & -\frac{d_s^{(spdim-1)}}{\alpha_s^{(spdim-2)}} & \frac{d_s^{(spdim-1)}}{\alpha_s^{(spdim-1)}}(1+\beta_s^{(spdim-1)}) \end{bmatrix}, \tag{3.25}$$

where $d_s^{(j)} := \mathbf{p}_s^{(j)T}\mathbf{A}_s\mathbf{p}_s^{(j)}$ is computed during the linear solve, and

$$\mathbf{P}_s^T\mathbf{A}_s\mathbf{P}_s = \mathrm{diag}(d_s^{(0)}, \ldots, d_s^{(spdim-1)}). \tag{3.26}$$

For cases with restarts, analogous matrix components are straightforwardly defined. Then, by property of DEF-CG, $\mathbf{W}_s^T\mathbf{A}_s\mathbf{P}_s$ vanishes, while $(\mathbf{A}_s\mathbf{W}_s)^T\mathbf{A}_s\mathbf{P}_s$ may be computed at negligible cost, in the way described by [140]. That is, we have

$$(\mathbf{A}_s\mathbf{W}_s)^T\mathbf{A}_s\mathbf{P}_s = \mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s\boldsymbol{\Delta}_s\mathbf{L}_s \tag{3.27}$$

where $\boldsymbol{\Delta}_s = [\widehat{\boldsymbol{\mu}}^{(0)}, \ldots, \widehat{\boldsymbol{\mu}}^{(\ell-1)}]$ and

$$\mathbf{L}_s = \begin{bmatrix} \frac{1}{\alpha_s^{(0)}} & & & & \\ -\frac{1}{\alpha_s^{(0)}} & \frac{1}{\alpha_s^{(1)}} & & & \\ & -\frac{1}{\alpha_s^{(1)}} & \ddots & & \\ & & \ddots & \ddots & \frac{1}{\alpha_s^{(\ell-1)}} \\ & & & & -\frac{1}{\alpha_s^{(\ell-1)}} \end{bmatrix}. \tag{3.28}$$

3.5.2 Restarting the eigen-search space without preconditioner

Every restart of the eigen-search space entails a new assembly at a cost comparable to what is described in Section 3.5.1. Note that the orthogonalization step (line 12) of Algo. 24 can conveniently be performed between the reduced eigenvectors, as described in [148]. That is, the orthogonal basis stored in the columns of $\mathbf{Q}$ that spans $\mathcal{R}([\mathbf{y}_1, \overline{\mathbf{y}}_1, \ldots, \mathbf{y}_k, \overline{\mathbf{y}}_k])$, can be recast in $\mathbf{Q} = \mathbf{V}_s\widehat{\mathbf{Q}}$ where $\widehat{\mathbf{Q}}$ is such that

$$\mathcal{R}(\widehat{\mathbf{Q}}) = \mathrm{Span}\left\{\widehat{\mathbf{y}}_1, \begin{bmatrix}\widehat{\overline{\mathbf{y}}}_1 \\ 0\end{bmatrix}, \ldots, \widehat{\mathbf{y}}_k, \begin{bmatrix}\widehat{\overline{\mathbf{y}}}_k \\ 0\end{bmatrix}\right\} \tag{3.29}$$

where $\mathbf{y}_i = \mathbf{V}_s\widehat{\mathbf{y}}_i$ and $\overline{\mathbf{y}}_i = \overline{\mathbf{V}}_s\widehat{\overline{\mathbf{y}}}_i$. As a result, $\mathbf{Q}$ is a computed at a cost of only $\mathcal{O}(k^3)$ instead of $\mathcal{O}(nk^2)$ if we were to factorize $[\mathbf{y}_1, \overline{\mathbf{y}}_1, \ldots, \mathbf{y}_k, \overline{\mathbf{y}}_k]$ directly .

3.5.3 Assembly of the reduced eigenvalue problem with preconditioner

Let us consider the implementation details of DEF-CG($\mathbf{A}_s$, $\mathbf{M}$, $\mathbf{W}_s$, $\mathbf{b}_s$, $\mathbf{u}_s^{(0)}$). First, both matrices $\mathbf{A}_s\mathbf{W}_s$ and $\mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s$ are computed prior to the first solver iteration and stored at memory costs of kn and k^2, respectively. Then, the computational cost of assembling the reduced (generalized) eigenvalue problems is minimized as follows for RR and HR projections.

3.5.3.1 Approximate eigenvectors of $\mathbf{A}_s$

RR projection For RR projections, we extract sequences of normalized preconditioned residuals $\mathbf{z}_s^{(j)}/\|\mathbf{z}_s^{(j)})\|$ which we store by columns in $\mathbf{Z}_s$. Then, we have $\mathbf{V}_s = [\mathbf{W}_s, \mathbf{Z}_s]$ so that Eq. (3.18) becomes:

$$\begin{bmatrix} \mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{W}_s^T\mathbf{A}_s\mathbf{Z}_s \\ \mathbf{Z}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{Z}_s^T\mathbf{A}_s\mathbf{Z}_s \end{bmatrix} \hat{\mathbf{w}} = \vartheta \begin{bmatrix} \mathbf{Z}_s^T\mathbf{Z}_s & \mathbf{W}_s^T\mathbf{Z}_s \\ \mathbf{Z}_s^T\mathbf{W}_s & \mathbf{I}_\ell \end{bmatrix} \hat{\mathbf{w}} \tag{3.30}$$

where $\mathbf{Z}_s^T\mathbf{A}_s\mathbf{Z}_s$ is dense and $\mathbf{A}_s\mathbf{Z}_s$ is a byproduct of the solver so that $\mathbf{Z}_s^T\mathbf{A}_s\mathbf{Z}_s$ is computed at a cost of $\mathcal{O}(\ell^2 n)$. Computing the product $\mathbf{W}_s^T\mathbf{A}_s\mathbf{Z}_s$ can be done column-by-column as the preconditioned residuals are being generated by the solver, for a total cost of $\mathcal{O}(k\ell n)$. $\mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s$ is already known and stored. The terms $\mathbf{Z}_s^T\mathbf{Z}_s$ and $\mathbf{Z}_s^T\mathbf{W}_s$ are computed at costs of $\mathcal{O}(\ell^2 n)$ and $\mathcal{O}(k\ell n)$, respectively. Lastly, note that we assumed that the approximate eigenvectors stored in the columns of $\mathbf{W}_s$ are orthonormal.

HR projection When using HR projections, a simpler problem is obtained when extracting sequences of ℓ search directions $\mathbf{p}_s^{(j)}$, which we store by columns in $\mathbf{P}_s$. Then, we have $\mathbf{V}_s = [\mathbf{W}_s, \mathbf{P}_s]$ so that Eq. (3.19) is recast as:

$$\begin{bmatrix} (\mathbf{A}_s\mathbf{W}_s)^T\mathbf{A}_s\mathbf{W}_s & (\mathbf{A}_s\mathbf{W}_s)^T\mathbf{A}_s\mathbf{P}_s \\ (\mathbf{A}_s\mathbf{P}_s)^T\mathbf{A}_s\mathbf{W}_s & (\mathbf{A}_s\mathbf{P}_s)^T\mathbf{A}_s\mathbf{P}_s \end{bmatrix} \hat{\mathbf{w}} = \vartheta \begin{bmatrix} \mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{W}_s^T\mathbf{A}_s\mathbf{P}_s \\ \mathbf{P}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{P}_s^T\mathbf{A}_s\mathbf{P}_s \end{bmatrix} \hat{\mathbf{w}} \tag{3.31}$$

in which the computation of $(\mathbf{A}_s\mathbf{W}_s)^T\mathbf{A}_s\mathbf{W}_s$ costs $\mathcal{O}(k^2 n)$. Computing $\mathbf{P}_s^T\mathbf{A}_s\mathbf{W}_s$ and $(\mathbf{A}_s\mathbf{P}_s)^T\mathbf{A}_s\mathbf{P}_s$ cost $\mathcal{O}(k\ell n)$ and $\mathcal{O}(\ell^2 n)$, respectively. Meanwhile, the tridiagonal $\mathbf{P}_s^T\mathbf{A}_s\mathbf{P}_s$ is given by Eq.(3.26). Finally, $\mathbf{W}_s^T\mathbf{A}_s\mathbf{P}_s$ cancels and $\mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s$ is already computed and stored.

3.5.3.2 Approximate eigenvectors of $\mathbf{M}^{-1}\mathbf{A}_s$

RR projection For RR projections, we extract sequences of normalized preconditioned residuals $\mathbf{z}_s^{(j)}/({\mathbf{r}_s^{(j)}}^T\mathbf{z}_s^{(j)})^{1/2}$ which we store by columns in $\mathbf{Z}_s$. Then, we have $\mathbf{V}_s = [\mathbf{W}_s, \mathbf{Z}_s]$ so that Eq. (3.20) is recast as:

$$\begin{bmatrix} \mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{W}_s^T\mathbf{A}_s\mathbf{Z}_s \\ \mathbf{Z}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{Z}_s^T\mathbf{A}_s\mathbf{Z}_s \end{bmatrix} \hat{\mathbf{w}} = \vartheta \begin{bmatrix} \mathbf{W}_s^T\mathbf{M}\mathbf{W}_s & \mathbf{W}_s^T\mathbf{M}\mathbf{Z}_s \\ \mathbf{Z}_s^T\mathbf{M}\mathbf{W}_s & \mathbf{Z}_s^T\mathbf{M}\mathbf{Z}_s \end{bmatrix} \hat{\mathbf{w}}. \tag{3.32}$$

Note that, by property of DEF-PCG, we have $\mathbf{W}_s^T\mathbf{M}\mathbf{z}_s^{(j)} = \mathbf{W}_s^T\mathbf{r}_s^{(j)} = 0$, and ${\mathbf{z}_s^{(i)}}^T\mathbf{M}\mathbf{z}_s^{(j)} = \delta_{ij}{\mathbf{r}_s^{(i)}}^T\mathbf{z}_s^{(j)}$. Moreover, we have $\mathbf{W}_2 := \mathbf{V}_1\hat{\mathbf{W}}_1 = \mathbf{Z}_1\hat{\mathbf{W}}_1$ so that $\mathbf{W}_2^T\mathbf{M}\mathbf{W}_2 = \hat{\mathbf{W}}_1^T\hat{\mathbf{W}}_1$. Therefore, as long as the reduced eigenvectors are orthonormal, the generalized eigenvalue problem simplifies to

$$\begin{bmatrix} \mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{W}_s^T\mathbf{A}_s\mathbf{Z}_s \\ \mathbf{Z}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{Z}_s^T\mathbf{A}_s\mathbf{Z}_s \end{bmatrix} \hat{\mathbf{w}} = \vartheta\hat{\mathbf{w}} \tag{3.33}$$

for all s, where $\mathbf{Z}_s^T\mathbf{A}_s\mathbf{Z}_s$ is dense. Considering that $\mathbf{A}_s\mathbf{Z}_s$ is a byproduct of the deflated solver, the computation of $\mathbf{Z}_s^T\mathbf{A}_s\mathbf{Z}_s$ carries an additional cost of $\mathcal{O}(\ell^2 n)$. The computation of $\mathbf{W}_s^T\mathbf{A}_s\mathbf{Z}_s$ does not simplify and entails an additional cost, which considering that $\mathbf{A}_s\mathbf{Z}_s$ is a byproduct of the deflated solver, amounts to $\mathcal{O}(k\ell n)$ every time a reduced eigenvalue problem is assembled. Note that the time complexity of assembling the reduced RR eigenvalue problem is *not* affected by the use of a preconditioner. A more detailed implementation can be found in [148].

HR projection For HR projections, the most efficient assembly of the reduced generalized eigenvalue problem is obtained when extracting sequences of search directions $\mathbf{p}_s^{(j)}$ stored by columns in $\mathbf{P}_s$. Then, we have $\mathbf{V}_s = [\mathbf{W}_s, \mathbf{P}_s]$ so that Eq. (3.21) is recast as:

$$\begin{bmatrix} (\mathbf{A}_s\mathbf{W}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{W}_s & (\mathbf{A}_s\mathbf{W}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{P}_s \\ (\mathbf{A}_s\mathbf{P}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{W}_s & (\mathbf{A}_s\mathbf{P}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{P}_s \end{bmatrix} \hat{\mathbf{w}} = \vartheta \begin{bmatrix} \mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s & \mathbf{0}_{k,\ell} \\ \mathbf{0}_{\ell,k} & \mathbf{P}_s^T\mathbf{A}_s\mathbf{P}_s \end{bmatrix} \hat{\mathbf{w}}, \tag{3.34}$$

where $\mathbf{P}_s^T\mathbf{A}_s\mathbf{P}_s$ and $(\mathbf{A}_s\mathbf{P}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{P}_s$ are diagonal and tridiagonal matrices, respectively, whose coefficients may be extracted as byproducts of the solver. In particular, $\mathbf{P}_s^T\mathbf{A}_s\mathbf{P}_s$ is given by Eq. (3.26), and

$$(\mathbf{A}_s\mathbf{P}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{P}_s = \begin{bmatrix} \frac{d_s^{(0)}}{\alpha_s^{(0)}}(1+\beta_s^{(0)}) & -\frac{d_s^{(1)}}{\alpha_s^{(0)}} & & & & \\ -\frac{d_s^{(1)}}{\alpha_s^{(0)}} & \frac{d_s^{(1)}}{\alpha_s^{(1)}}(1+\beta_s^{(1)}) & -\frac{d_s^{(2)}}{\alpha_s^{(1)}} & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & -\frac{d_s^{(spdim-1)}}{\alpha_s^{(spdim-2)}} & \\ & & & -\frac{d_s^{(spdim-1)}}{\alpha_s^{(spdim-2)}} & \frac{d_s^{(spdim-1)}}{\alpha_s^{(spdim-1)}}(1+\beta_s^{(spdim-1)}) \end{bmatrix}. \tag{3.35}$$

The off-diagonal block $(\mathbf{A}_s\mathbf{W}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{P}_s$ is assembled in the same fashion as the corresponding block in Eqs. (3.24) and (3.27), see [140]. That is, we have $(\mathbf{A}_s\mathbf{W}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{P}_s = \mathbf{W}_s^T\mathbf{A}_s\mathbf{W}_s\mathbf{\Delta}_s\mathbf{L}_s$. However, $(\mathbf{A}_s\mathbf{W}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{W}_s$ does not simplify. Therefore, the time complexity of assembling the reduced HR generalized eigenvalue value problem is affected by the necessity to perform k preconditioner applications for the computation of $(\mathbf{A}_s\mathbf{W}_s)^T\mathbf{M}^{-1}\mathbf{A}_s\mathbf{W}_s$. Even though these preconditioner applications can be avoided when using RR projections, this was not pointed out in [140].

3.6 Numerical experiments

3.6.1 Monte Carlo Markov chains of coefficient fields

In the present work, we consider a log-normal coefficient field κ whose underlying Gaussian process has zero mean, $\mathbb{E}[h(x;\theta)] = 0$. In particular, two cases are considered. Case 1 corresponds to a 1D stationary coefficient field with exponential covariance function for h, given by $\mathbb{E}[h(x)h(y)] = \sigma^2 \exp(-|x-y|/L)$, in which $\sigma^2 = 0.5$ and $L = 0.05$. Case 2 corresponds to a 2D stationary coefficient field with squared exponential covariance function for h given by $\mathbb{E}[h(x)h(y)] = \sigma^2 \exp(-\|x-y\|^2/L^2)$ in which $\sigma^2 = 1$ and $L = 0.1$. For Case 1 we used $n_s = 1{,}802$ while for Case 2 $n_s = 176$. Figures 3.1 and 3.2 report few realizations of κ for Case 1 and 2 respectively.

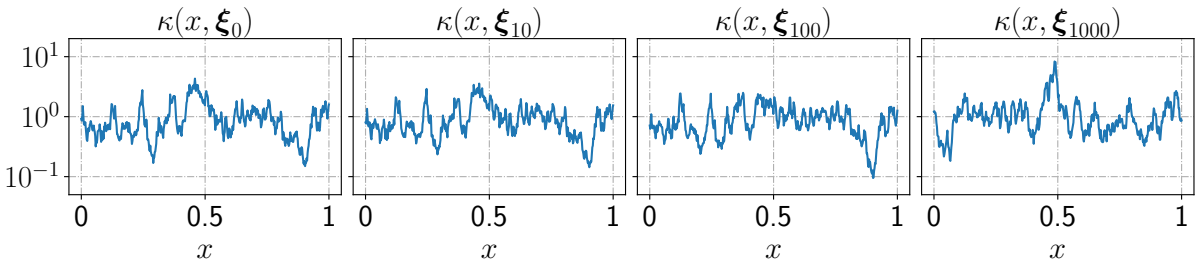

Figure 3.1: Realizations of $\kappa(x) := \exp(h(x))$ for Case 1.

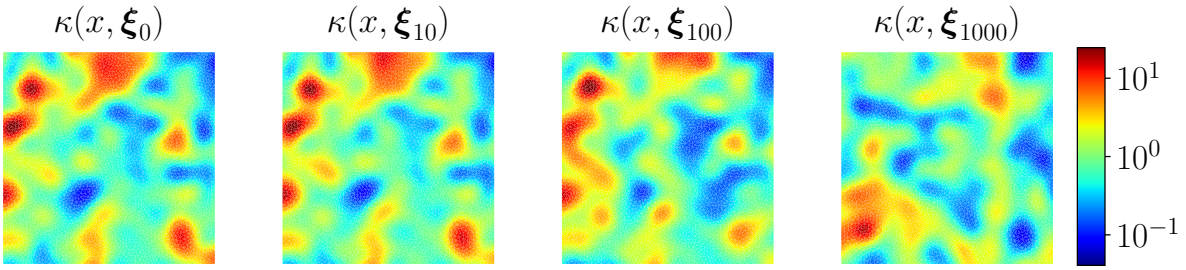

Figure 3.2: Realizations of $\kappa(x) := \exp(h(x))$ for Case 2.

3.6.2 Deflation subspaces constructed without preconditioner

We consider a 1,000-long sequence of 500-by-500 linear systems obtained by setting $u(0) := 0$ and $\partial_x u(1) := 0$ in Eq. (3.4) with the 1D coefficient fields presented in Fig. 3.1 over a domain $\Omega = [0, 1]$ and a forcing term $f(x) = 1$. These systems are solved by CG, and the convergence histories of the norm of the iterated residual are presented in Fig. 3.3. Because the sampled coefficients are highly heterogeneous, the corresponding operators are ill-conditioned, and CG systematically converges after more than n iterations. Note that, while the first linear systems of this particular sampled sequence require more iterations to solve, this is not generally true of all sequences.

The same sequence of linear systems is solved by Def-CG and Init-CG using $k = 10$ RR eigenvector approximations with and without restarting the eigen-search space. The

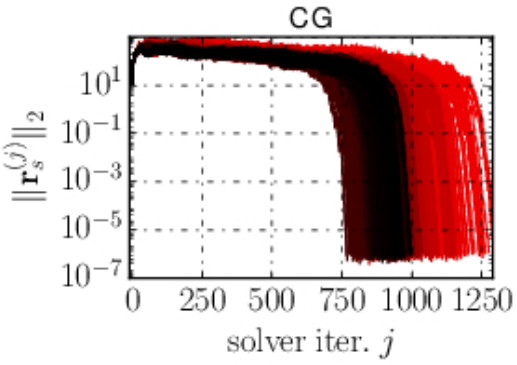

Figure 3.3: Norm of iterated residuals obtained by CG for a sequence of 1000 linear systems with 500 degrees of freedom (DoFs). Coefficient field of Case 1. System index s gradually color coded from first ($s = 0$, —) to last ($s = 1000$, —) in sequence.

resulting convergence histories are presented in Fig. 3.4. The results obtained by DEF-CG are on the first row of the graph, and those of INIT-CG on the second row. Analogous results obtained using as many HR eigenvector approximations are presented in Fig. 3.5. All the eigen-search spaces are kept at a dimension $spdim = 40$.

We observe the following. First, despite a high correlation between realizations of the coefficient field—see the similarity between $\kappa(x, \boldsymbol{\xi}_0)$, $\kappa(x, \boldsymbol{\xi}_{10})$, $\kappa(x, \boldsymbol{\xi}_{100})$ and $\kappa(x, \boldsymbol{\xi}_{1000})$ in Fig. 3.1—all the INIT-CG procedures recover the behavior of regular CG once the norm of the iterated residual reaches a value smaller than 10^{-1}. RR-LO-TR-INIT-CG, known as eigCG [148], provides the best convergence behavior before the norm of the iterated residual reaches this value. Second, all the DEF-CG procedures show significant improvements in their convergence behavior between the beginning and the end of the sampled sequence. Despite the fact that the operators are different in consecutive linear systems, it does seem that those remain sufficiently correlated so that the approximated eigenvectors are accurate enough for the deflation to be effective. However, the convergence behaviors observed toward the end of the sampled sequence, i.e., the black curves, have a different shape depending on whether RR or HR projections are used—RR projections showing slightly faster convergence. Third, only one restarting strategy seems to be effective, namely, RR-LO-TR-DEF-CG.

3.6.3 Scaling of deflated preconditioned conjugate gradient algorithms

We first consider a 200-long sequence of 2,000-by-2,000 linear systems obtained by setting $u(0) := 0$ and $\partial_x u(1) := 0$ in Eq. (3.4) with the 1D coefficient fields presented in Fig. 3.1 over a domain $\Omega = [0, 1]$ and a forcing term $f(x) = 1$. These systems are solved by PCG while preconditioning with a single V-cycle of an AMG solver [114] based on the operator $\widehat{\mathbf{A}}$ constructed with the median realization, as well as with HR-DEF-PCG using constant block-Jacobi (bJ) preconditioners with 10, 20 and 30 non-overlapping diagonal blocks of $\widehat{\mathbf{A}}$. For each instance of HR-DEF-PCG, the number k of approximated eigenvectors is set equal to the number of blocks, whereas the dimension of the eigen-search space is set

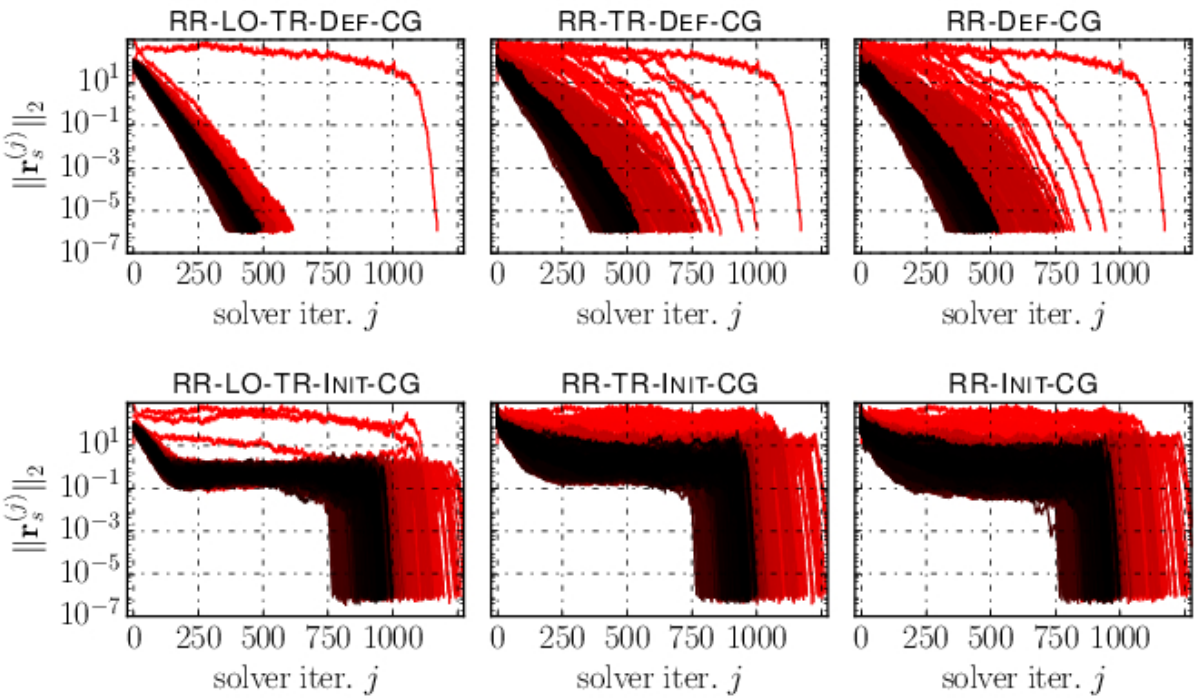

Figure 3.4: Norm of iterated residuals obtained by DEF-CG (top row) and INIT-CG (bottom row) procedures using RR projections for a sequence of 1000 linear systems with 500 DoFs. Coefficient field of Case 1. System index s gradually color coded from first ($s = 0$, —) to last ($s = 1000$, —) in sequence.

to $spdim := 2k$. The number of solver iterations to reach a backward error smaller than 10^{-7} is presented on the left side of Fig. 3.6, whereas the right side of Fig. 3.6 presents the spectra of $\mathbf{A}_s$, $\dot{\mathbf{A}}_s$ and, for the instances of HR-DEF-PCG, of $\dot{\mathbf{\Pi}}_s\dot{\mathbf{A}}_s$, at the end of the sequence, i.e., for $s = 200$.

Even though PCG (AMG) uses a constant preconditioner, the resulting spectrum $\mathrm{Sp}(\dot{\mathbf{A}}_s)$ is efficiently condensed between 10^{-1} and 10 for all the realizations, leading to a roughly constant iteration number for this sequence. On the other hand, the spectra $\mathrm{Sp}(\dot{\mathbf{A}}_s)$ associated with the bJ preconditioners mostly consist of two parts: (i) a dense set of eigenvalues ranging from 10^{-1} to 10, and (ii) a trail of as many eigenvalues as the number of blocks used for the preconditioner, spread throughout the lower end of the spectrum. The iteration numbers obtained by HR-DEF-PCG (bJ) at $s = 0$ are the same as what would be obtained using PCG (bJ). As expected, this number increases with the number of blocks. However, as the k approximate LD eigenvectors generated throughout the sequence are used for deflation, the iteration numbers of HR-DEF-PCG (bJ) decrease significantly, and most of the trailing eigenvalues of $\mathrm{Sp}(\dot{\mathbf{A}}_s)$ are "removed" in $\mathrm{Sp}(\dot{\mathbf{\Pi}}_s^T\dot{\mathbf{A}}_s)$. Note that the few remaining trailing eigenvalues can be removed by increasing the number k of approximated eigenvectors, in which case HR-DEF-PCG (bJ) slightly outperforms PCG (AMG).

We now consider a 1,000-long sequence of 4,000-by-4,000 linear systems obtained by setting $u(x) := 0$ on the border of a 2D square domain $\Omega := (0, 1)^2$, with the coefficient fields presented in Fig. 3.2 and a constant $f = 1$. These systems are solved by PCG (AMG) as well as by PCG (bJ), HR-DEF-PCG (bJ) and RR-LO-TR-DEF-

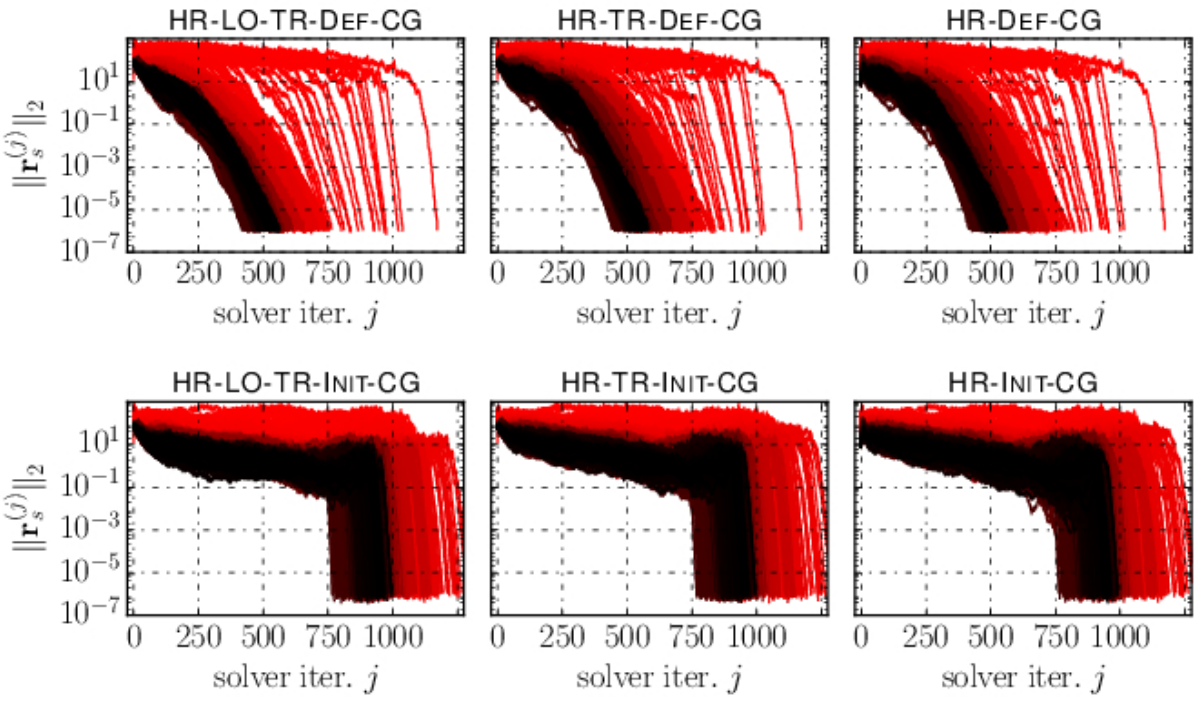

Figure 3.5: Norm of iterated residuals obtained by DEF-CG (top row) and INIT-CG (bottom row) procedures using HR projections for a sequence of 1000 linear systems with 500 DoF. Coefficient field of Case 1. System index s gradually color coded from first ($s = 0$, —) to last ($s = 1000$, —) in sequence.

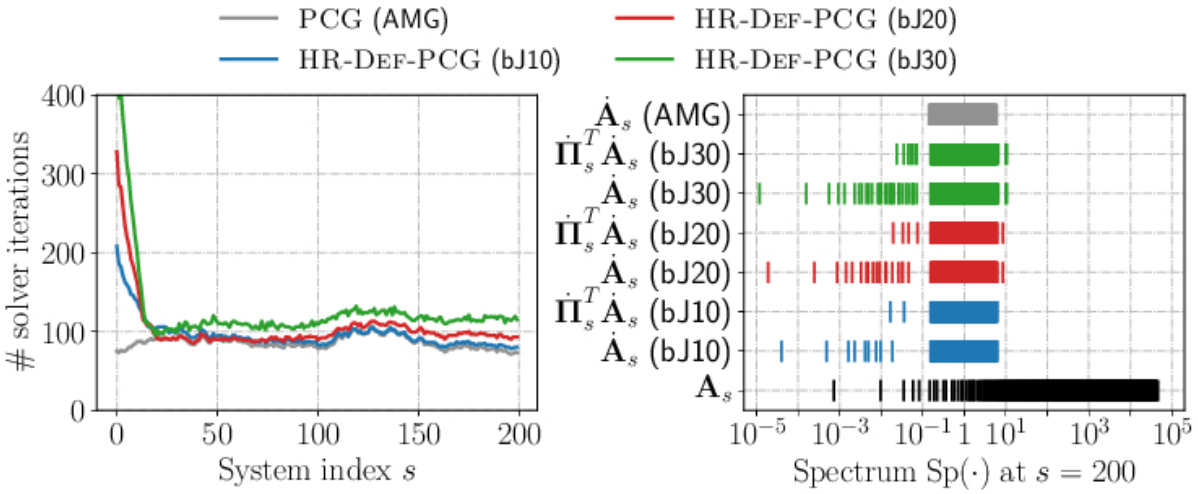

Figure 3.6: Left side: number of solver iterations needed to reach a backward error smaller than 10^{-7} using HR-DEF-PCG (bJ10, bJ20, bJ30) and PCG (AMG). Right side: original, preconditioned and deflated spectra of $\mathbf{A}_s$ at $s = 200$. Linear systems with 2000 DoFs and coefficient field of Case 1.

PCG (bJ) using constant preconditioners with 10, 20 and 30 non-overlapping blocks of $\widehat{\mathbf{A}}$. Note that HR-Def-PCG and RR-LO-TR-Def-PCG are considered in particular because thorough details of their implementation can be found in [140] and [148], respectively. For each instance of deflated solver, the number k of approximated eigenvectors is set equal to the number of blocks, and the dimension of the eigen-search space is set to $spdim := 2k + 10$. The convergence histories of PCG (bJ10), HR-Def-PCG (bJ10) and RR-LO-TR-Def-PCG (bJ10) are presented in Fig. 3.7. The number of necessary solver iterations to reach a backward error smaller than 10^{-7} using PCG (AMG) and RR-LO-TR-Def-PCG (bJ10, 20, 30) is presented on the left side of Fig. 3.8 for $s \leq 200$, whereas the right side of Fig. 3.8 presents the corresponding spectra of $\mathbf{A}_s$, $\dot{\mathbf{A}}_s$ and $\dot{\mathbf{\Pi}}_s\dot{\mathbf{A}}_s$ at $s = 100$.

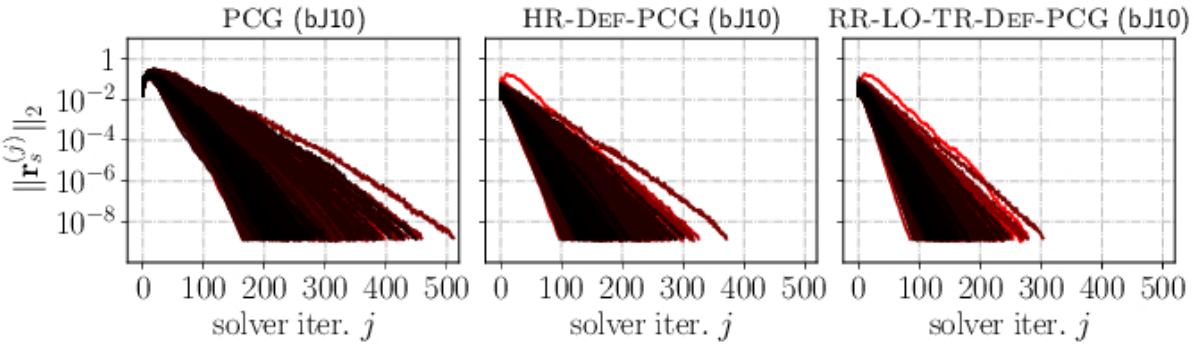

Figure 3.7: Norm of iterated residuals obtained by PCG (bJ10), HR-Def-PCG (bJ10) and RR-LO-TR-PCG (bJ10) for a sequence of 1000 linear systems with 4000 DoFs. Coefficient field of Case 2. System index s gradually color coded from first ($s = 0$, —) to last ($s = 1000$, —) in sequence.

Unlike the non-preconditioned results of Figs. 3.4–3.5, the results of Fig. 3.7 do not show a significant difference between the convergence behavior of the linear systems solved at the beginning and at the end of the sampled sequence. The preconditioned spectra of Fig. 3.8 are also denser than those of Fig. 3.6, due to the fact that the coefficient fields of this example are significantly smoother than those of Fig. 3.1. However, despite a smaller correlation between coefficient fields, and less separated eigenvalues after the application of the bJ preconditioners, the deflated spectra $\mathrm{Sp}(\dot{\mathbf{\Pi}}_s^T\dot{\mathbf{A}}_s)$ obtained by RR-LO-Def-PCG (bJ) are nearly as condensed as the spectrum of $\dot{\mathbf{A}}_s$ obtained with the constant AMG preconditioner. Consequently, the iteration numbers (left side of Fig. 3.8) are also similar to those obtained by PCG (AMG)—for linear systems with 4,000 DoFs.

The same sequence of coefficient fields is considered with different spatial discretizations, along with the same f and boundary conditions, leading to linear systems of size 4,000, 16,000 and 64,000. These linear systems are solved by PCG (AMG), PCG (bJ10), HR-Def-PCG (bJ10) and RR-LO-TR-Def-PCG (bJ10) using $k := 20$ approximate eigenvectors with $spdim := 50$. The resulting numbers of solver iterations are presented in Fig. 3.9. We observe that increasing the number of DoFs results in a more significant degradation of the convergence for PCG (bJ10) than for PCG (AMG). For small numbers of DoFs, this difference is successfully compensated by the use of deflation, irrespective of whether the eigen-search space is restarted or not. However, as the number of DoFs increases, the deflated bJ preconditioner does not work as well, and the restarting of the

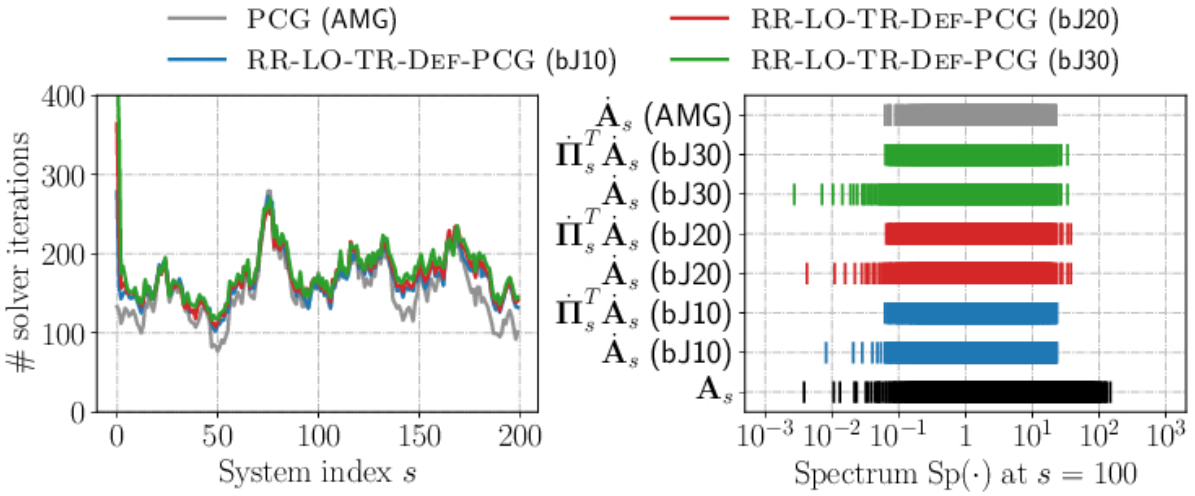

Figure 3.8: Left side: number of solver iterations needed to reach a backward error smaller than 10^{-7} using RR-LO-TR-DEF-PCG (bJ10, bJ20, bJ30) and PCG (AMG). Right side: original, preconditioned and deflated spectra of $\mathbf{A}_s$ at $s = 100$. Linear systems with 4000 DoFs and coefficient field of Case 2.

eigen-search space positively impacts more and more the convergence behavior of the deflated solver, as shown by the iteration number of RR-LO-TR-DEF-PCG compared to those of HR-DEF-PCG.

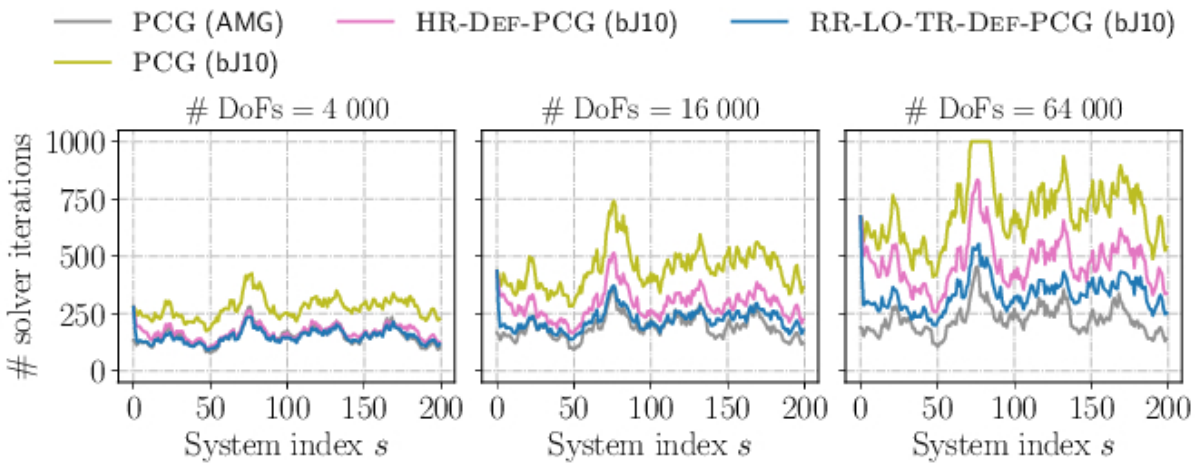

Figure 3.9: Number of solver iterations needed to reach a backward error smaller than 10^{-7} using HR-DEF-PCG (bJ10), RR-LO-TR-DEF-PCG (bJ10) and PCG (AMG, bJ10) for increasing numbers of DoFs. Coefficient field of Case 2.

The expected number of solver iterations are reported in Table 3.1 for RR projection-based deflation strategies with and without restarts, in comparison to PCG. These results, obtained for linear systems with 32,000 DoFs and a bJ preconditioner with different number of blocks, are also plotted in Fig. 3.10. Clearly, the average number of PCG iterations increases with the number of blocks. Interestingly, this is not strictly the case once deflation is introduced. Irrespective of whether the eigen-search space is restarted or

not, the numbers of solver iterations saved by deflation increase with the number of blocks of the preconditioner. The effect of restarting the eigen-search space is quite apparent as the average numbers of solver iterations is the smallest for RR-LO-TR-Def-PCG, followed by RR-TR-Def-PCG and RR-Def-PCG. Similar results as those of Table 3.1 and Fig. 3.10 are presented in Table 3.2 and Fig. 3.11 for HR projection-based deflation strategies, once again, with and without restarts, in comparison to PCG. The results are similar to those obtained by RR projection, with the exception that the difference between the performance of HR-LO-TR-Def-PCG and HR-TR-Def-PCG are less apparent.

In Table 3.3 and Fig. 3.12, we present average numbers of solver iterations for RR projection-based deflation strategies when using a LORASC preconditioner with different numbers of subdomains. Similarly as for the bJ preconditioner as a function of the number of blocks, the average number of PCG iterations increases with the number of subdomains. This dependence does not hold any more once deflation is introduced (with as many approximate eigenvectors as there are subdomains). Indeed, the average number of solver iterations for the deflated RR strategies is almost constant as a function of the number of subdomains. Consequently, the numbers of solver iterations saved by deflation in comparison to PCG increases with the number of subdomains. The difference between the two restarted strategies, i.e., HR-TR-Def-PCG and RR-LO-TR-Def-PCG, is not significant. Analogous results to those of Table 3.3 and Fig. 3.12 are presented in Table 3.4 and Fig. 3.13 for HR projection-based deflation strategies. The same observations as those of RR projections hold for the HR projections.

Method	n_b					
	5	7	10	15	20	30
PCG	866.19	986.45	1,077.60	1,180.85	1,200.84	1,237.42
RR-Def-PCG	831.79	928.38	973.89	1,019.28	958.73	823.91
RR-TR-Def-PCG	673.54	701.10	676.17	638.72	591.30	550.61
RR-LO-TR-Def-PCG	552.74	573.49	566.22	556.10	520.99	486.94

Table 3.1: Expected numbers of Def-PCG iterations with a bJ preconditioner for different numbers of blocks and RR projection method. Linear systems with 32 000 DoFs. Coefficient field of Case 2.

In Table 3.5 and Fig. 3.14, we present average numbers of solver iterations for RR projection-based deflation strategies applied to linear systems with different numbers of DoFs. For each system solved, the preconditioner used is bJ with 10 blocks. It can be said that the number of iterations saved by deflation decreases with the numbers of DoFs when no restarting strategy is used, i.e., for RR-Def-PCG. The benefit of resorting to a locally optimal restart, i.e., RR-LO-TR-Def-PCG over a simple thick restart, i.e., RR-TR-Def-PCG becomes more significant as the number of DoFs is increased. Analogous results are presented in Table 3.6 and Fig. 3.15 for HR projection-based deflation strategies. The only difference with RR projections is that there is significant differences between the two restarted startegies when using HR projections.

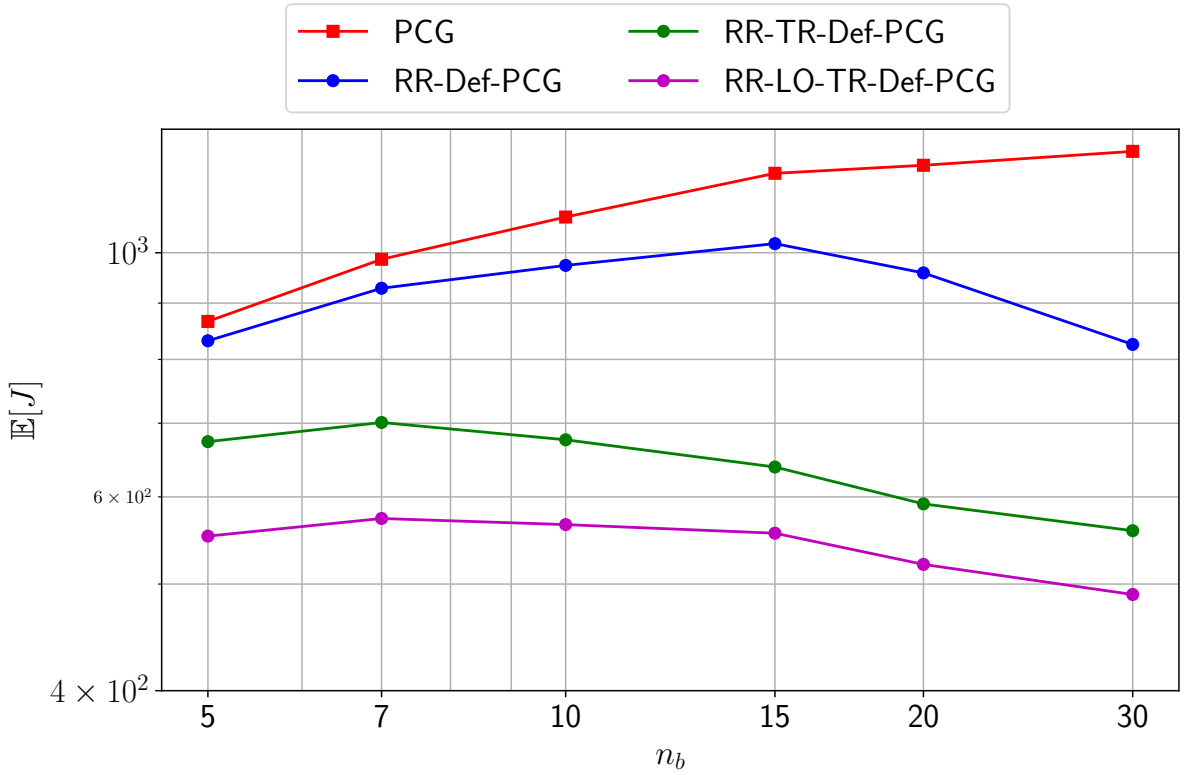

Figure 3.10: Scaling with respect to n_d of expected numbers of DEF-PCG iterations to solve $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with the bJ preconditioner and different RR projections. Linear systems with 32 000 DoFs. Coefficient field of Case 2.

Method	n_b					
	5	7	10	15	20	30
PCG	866.19	986.45	1,077.60	1,180.85	1,200.84	1,237.42
HR-Def-PCG	839.04	941.63	994.58	1,026.79	973.27	834.39
HR-TR-Def-PCG	616.67	649.47	642.74	613.86	577.27	521.38
HR-LO-TR-Def-PCG	605.46	636.83	627.76	610.51	569.92	502.95

Table 3.2: Expected numbers of Def-PCG iterations with a bJ preconditioner for different numbers of blocks and HR projection method. Linear systems with 32 000 DoFs. Coefficient field of Case 2.

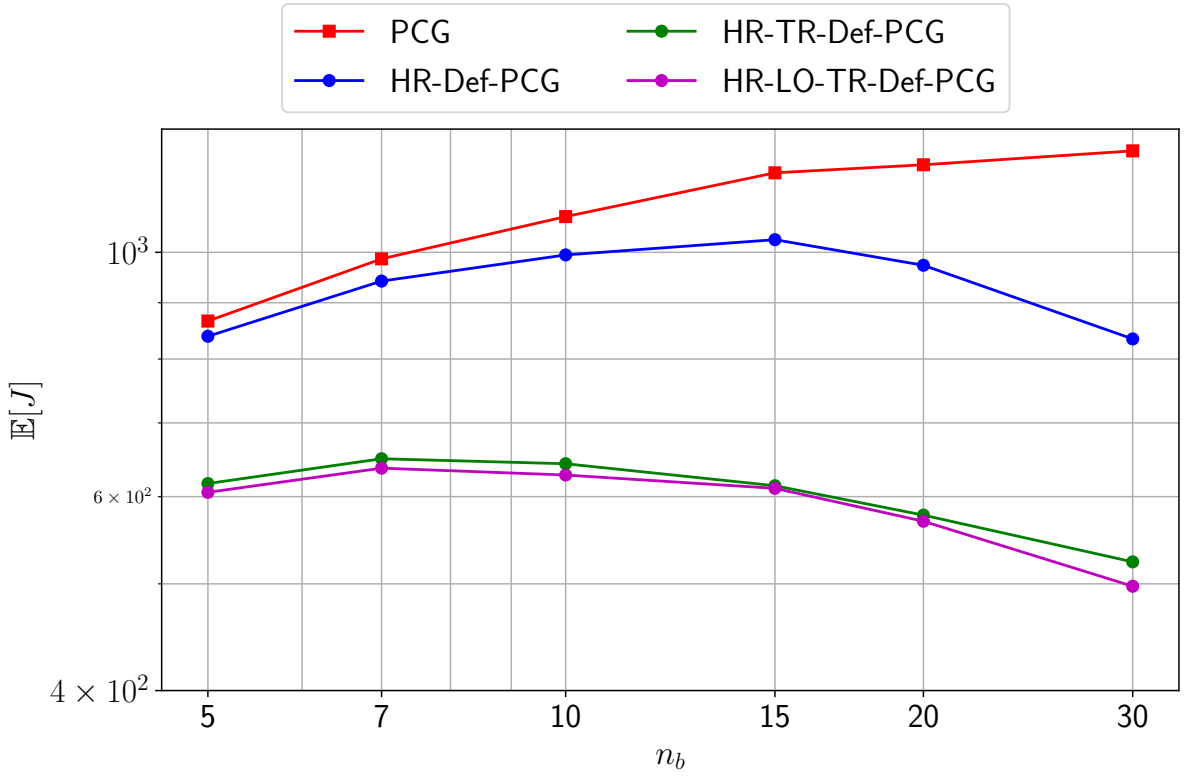

Figure 3.11: Scaling with respect to n_d of expected numbers of Def-PCG iterations to solve $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with the bJ preconditioner and different HR projections. Linear systems with 32 000 DoFs. Coefficient field of Case 2.

Similar results to those of Table 3.5 and Fig. 3.14 are presented in Table 3.7 and Fig. 3.16 where a LORASC preconditioner is used. Similarly as when using the bJ preconditioner, the benefit of restarting the eigen-search space because more apparent as the number of DoFs is increased. This is the case when using RR projections, but also for HR projections, see Table 3.8 and Fig. 3.17.

Method	n_d					
	5	7	10	15	20	30
PCG	254.37	274.80	293.47	311.31	341.08	358.36
RR-Def-PCG	226.16	224.46	217.65	214.90	228.04	214.27
RR-TR-Def-PCG	186.23	184.47	179.30	172.78	174.34	170.90
RR-LO-TR-Def-PCG	175.15	174.56	174.33	168.99	171.12	168.17

Table 3.3: Expected numbers of Def-PCG iterations with a LORASC preconditioner for different numbers of subdomains and RR projection method. Linear systems with 32 000 DoFs. Coefficient field of Case 2.

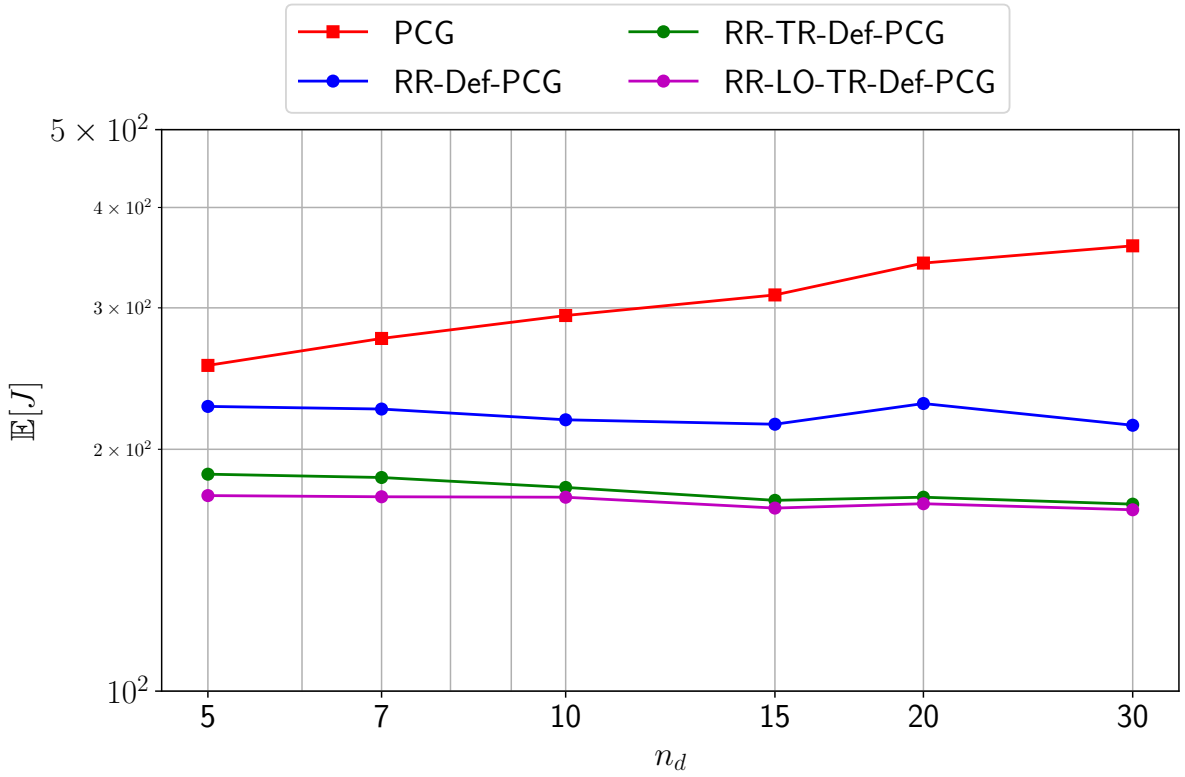

Figure 3.12: Scaling with respect to n_d of expected numbers of Def-PCG iterations to solve $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with the LORASC preconditioner and different RR projections. Linear systems with 32 000 DoFs. Coefficient field of Case 2.

Method	n_d					
	5	7	10	15	20	30
PCG	254.37	274.80	293.47	311.31	341.08	358.36
HR-Def-PCG	231.28	233.57	221.09	218.00	233.40	219.21
HR-TR-Def-PCG	184.00	182.08	184.46	182.93	187.98	177.65
HR-LO-TR-Def-PCG	186.04	187.41	189.41	183.94	186.17	178.07

Table 3.4: Expected numbers of Def-PCG iterations with a LORASC preconditioner for different numbers of subdomains and HR projection method. Linear systems with 32 000 DoFs. Coefficient field of Case 2.

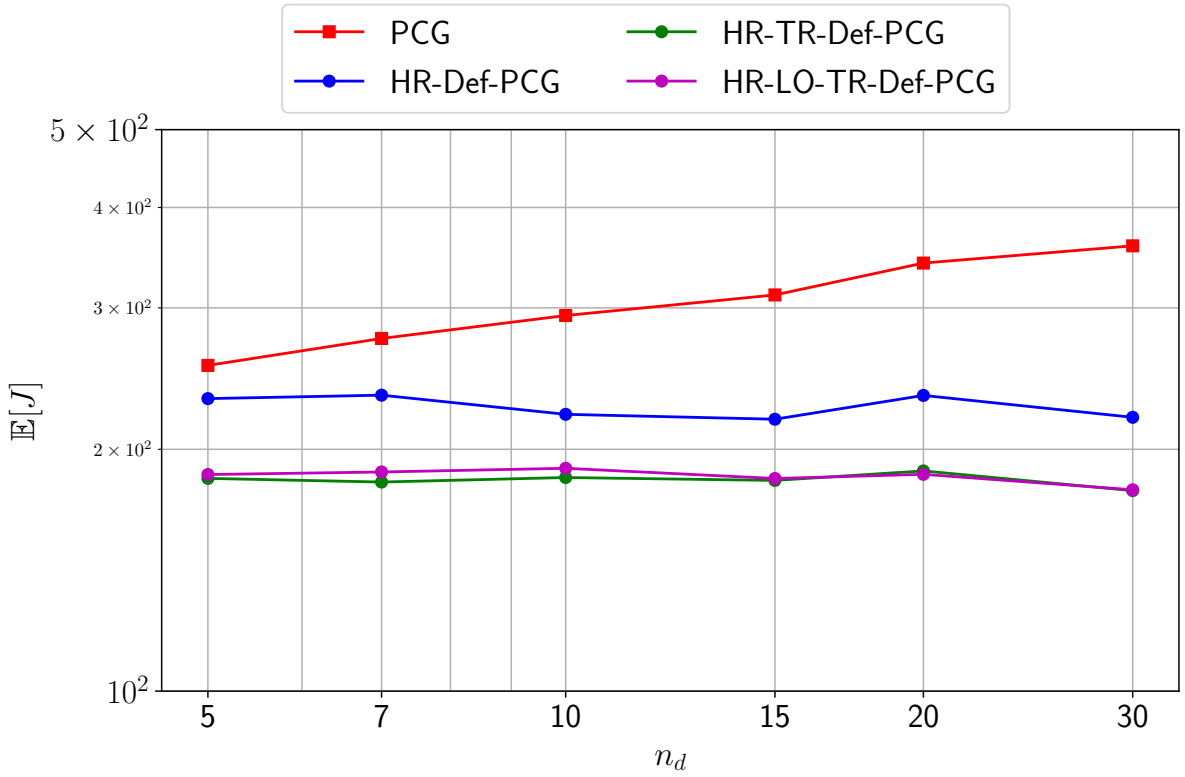

Figure 3.13: Scaling with respect to n_d of expected numbers of Def-PCG iterations to solve $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with the LORASC preconditioner and different HR projections. Linear systems with 32 000 DoFs. Coefficient field of Case 2.

Method	DoFs					
	4,000	8,000	16,000	32,000	64,000	128,000
PCG	402.26	518.06	835.01	1,077.60	1,742.58	2,060.58
RR-Def-PCG	307.46	417.62	734.32	973.89	1,620.53	1,958.50
RR-TR-Def-PCG	231.32	314.61	511.32	676.17	1,155.16	1,418.75
RR-LO-TR-Def-PCG	216.57	286.70	444.52	566.22	883.11	1,045.75

Table 3.5: Expected numbers of Def-PCG iterations with a bJ preconditioner for different numbers of DoFs and RR projection method. Preconditioner with 10 blocks. Coefficient field of Case 2.

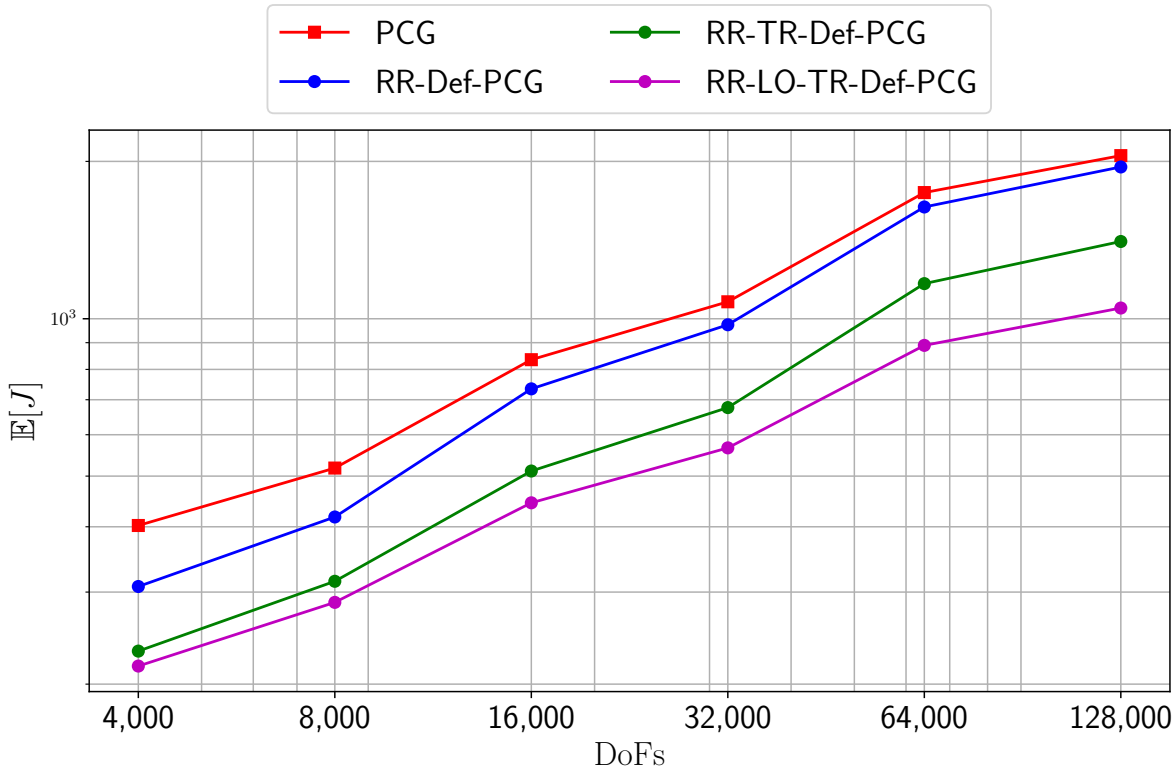

Figure 3.14: Scaling with respect to the number of DoFs of expected numbers of Def-PCG iterations to solve $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with the bJ preconditioner and different RR projections. Preconditioner with 10 blocks. Coefficient field of Case 2.

Method	DoFs					
	4,000	8,000	16,000	32,000	64,000	128,000
PCG	402.26	518.06	835.01	1,077.60	1,742.58	2,060.58
HR-Def-PCG	323.58	441.80	755.86	994.58	1,645.68	1,978.25
HR-TR-Def-PCG	225.41	305.65	484.78	642.74	1,027.21	1,215.67
HR-LO-TR-Def-PCG	221.27	301.30	479.78	627.76	1,016.68	1,239.50

Table 3.6: Expected numbers of Def-PCG iterations with a bJ preconditioner for different numbers of DoFs and HR projection method. Preconditioner with 10 blocks. Coefficient field of Case 2.

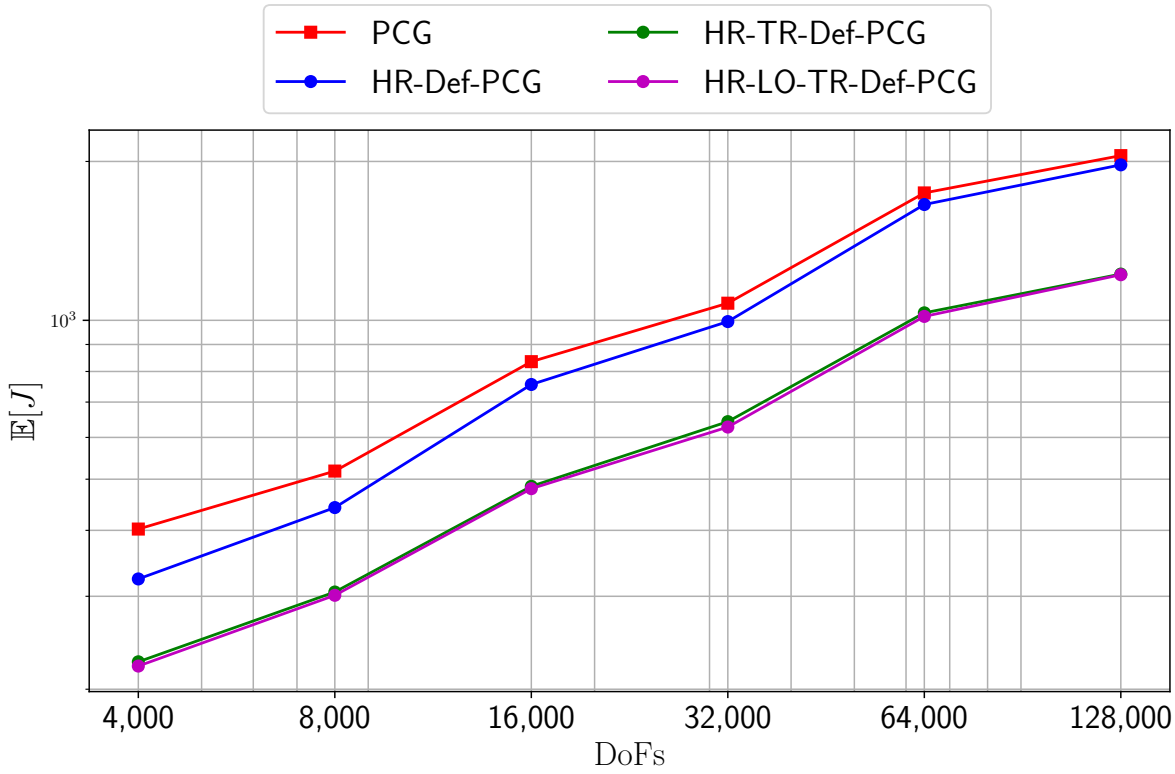

Figure 3.15: Scaling with respect to the number of DoFs of expected numbers of Def-PCG iterations to solve $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with the bJ preconditioner and different HR projections. Preconditioner with 10 blocks. Coefficient field of Case 2.

Method	DoFs					
	4,000	8,000	16,000	32,000	64,000	12,8000
PCG	160.53	202.41	253.95	293.47	387.13	375.33
RR-Def-PCG	115.35	148.22	186.52	217.65	296.93	274.67
RR-TR-Def-PCG	109.56	132.66	157.08	179.30	240.53	232.67
RR-LO-TR-Def-PCG	109.35	131.83	153.96	174.33	230.93	219.67

Table 3.7: Expected numbers of Def-PCG iterations with a LORASC preconditioner for different numbers of DoFs and RR projection method. Preconditioner with 10 subdomains. Coefficient field of Case 2.

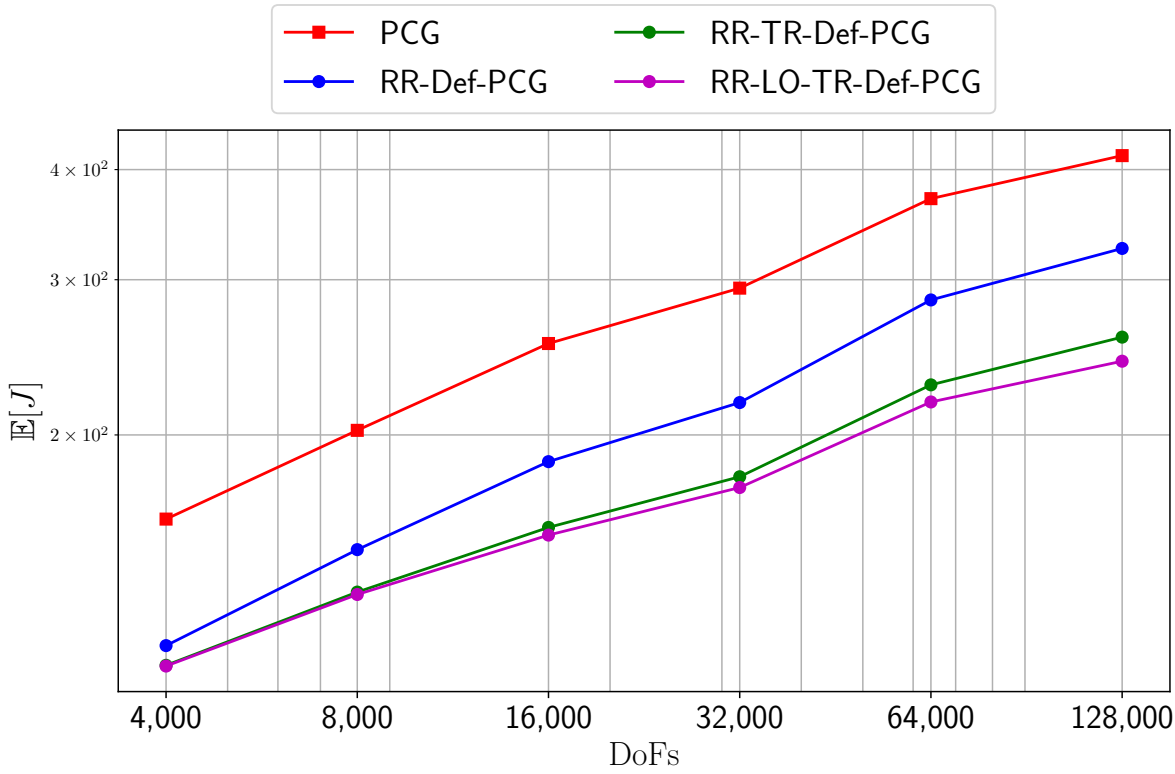

Figure 3.16: Scaling with respect to the number of DoFs of expected numbers of Def-PCG iterations to solve $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with the LORASC preconditioner and different RR projections. Preconditioner with 10 subdomains. Coefficient field of Case 2.

Method	DoFs					
	4,000	8,000	16,000	32,000	64,000	128,000
PCG	160.53	202.41	253.95	293.47	387.13	375.33
HR-Def-PCG	116.15	148.74	188.46	221.09	308.93	282.00
HR-TR-Def-PCG	110.20	133.51	159.11	184.46	249.60	246.33
HR-LO-TR-Def-PCG	111.55	136.86	163.61	189.41	255.47	262.67

Table 3.8: Expected numbers of Def-PCG iterations with a LORASC preconditioner for different numbers of DoFs and HR projection method. Preconditioner with 10 subdomains. Coefficient field of Case 2.

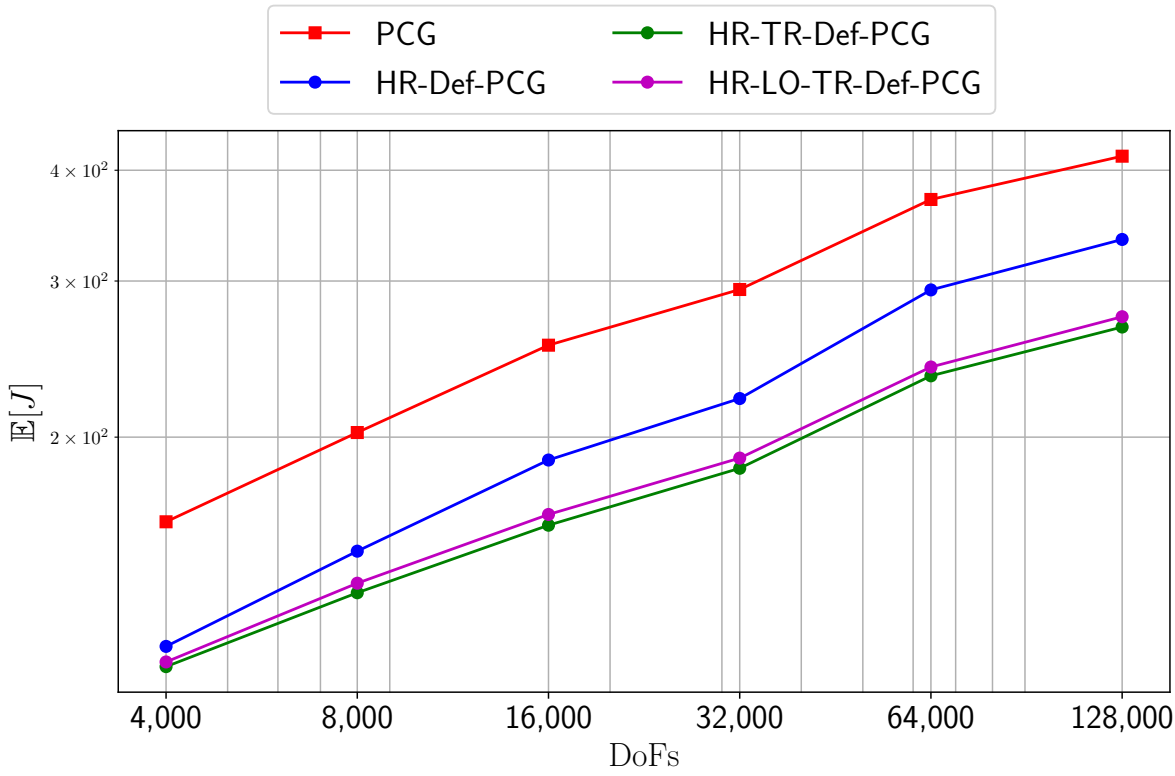

Figure 3.17: Scaling with respect to the number of DoFs of expected numbers of Def-PCG iterations to solve $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with the LORASC preconditioner and different HR projections. Preconditioner with 10 subdomains. Coefficient field of Case 2.

3.7 Conclusion

Recycling Krylov subspace strategies were investigated as a means to accelerate the iterative solution of SPD linear systems in sequences of discretized elliptic PDEs with random coefficient fields sampled by MCMC. Every sampled linear system was deflated with LD eigenvector approximations of the previously sampled operator These approximate eigenvectors were obtained using RR and HR projections with respect to eigen-search spaces

spanned by sequences of matrix-vector products of the previous linear solve along with previous approximate eigenvectors. Different strategies were investigated to make use of all the matrix-vector products generated by the solver while keeping the dimension of the eigen-search space to a certain *spdim* in order to avoid problems of loss of orthogonality and memory consumption. These strategies are referred to as restarting the eigen-search space, even though the linear solver is not restarted.

For the non-preconditioned cases, both DEF-CG and INIT-CG were used, and our experiments showed that, even if the coefficient fields are highly correlated, all the strategies based on INIT-CG eventually yield similar convergence behaviors to what is obtained with regular CG. For small numbers of DoFs, it was observed that the asymptotic convergence behavior of DEF-CG does not depend on the restarting strategy of the eigen-search space. However, using eigCG [148] while explicitly orthogonalizing the residual against $\mathcal{R}(\mathbf{W}_s)$ at each solver iteration (method referred to as RR-LO-TR-DEF-CG), albeit incurring an additional cost of $\mathcal{O}(kn)$ per iteration, allows for a significantly faster transition to better convergence behaviors than all the other restarting strategies.

Three types of constant preconditioners were considered based on the operator $\hat{\mathbf{A}}$ constructed with the median realization: (i) a single V-cycle of an AMG solver, (ii) bJ preconditioners with non-overlapping blocks, and (iii) LORASC preconditioners based on domain decomposition. While the constant AMG preconditioner efficiently condenses the spectra of the sampled operators, numerical experiments (not reported here) showed that the behavior of PCG (AMG) is not significantly improved by deflation. However, the constant bJ and LORASC preconditioners tend to segregate the spectrum into two parts, leaving a trail of as many eigenvalues as the number of blocks or domains spread throughout the lower end of the spectrum. These eigenvalues being well separated, the bJ and LORASC preconditioners are well suited for deflation. Both 1D and 2D examples show that deflating with as many approximate eigenvectors as the number of bJ blocks significantly improves the convergence behavior of DEF-PCG (bJ) compared to PCG (bJ). For small numbers of DoFs, the convergence behaviors of DEF-PCG (bJ) are comparable to those of PCG (AMG). However this is only the case for moderate numbers of blocks, say 30 or less. Indeed, by increasing the number of bJ blocks, one needs to increase the number k of approximated eigenvectors, and thus *spdim*, which can lead to undefined reduced (generalized) eigenvalue problems and larger memory requirements. As the number of DoFs increases, DEF-PCG (bJ) does not work as well as PCG (AMG), and the restarting strategy of the eigen-search space starts to matter. It is then recommended to use eigCG [148] with explicit orthogonalization of the iterated residual (i.e., RR-LO-TR-DEF-CG) over the recycling strategy of [140] adapted for multiple operators (i.e., HR-DEF-PCG). Similarly, the LORASC preconditioner is deflated with as many approximate eigenvectors as the number of subdomains. The strategy works well and even more so for larger numbers of subdomains. Restarting the eigen-search space has shown to be useful for the deflation, irrespective of the restarting scheme. Meanwhile, LO-TR is the most efficient restarting scheme for RR approximations with a bJ preconditioner; both restarting scheme show no difference for HR approximations. It is also worth mentioning that the positive effect of deflation on convergence increases with the number of blocks.

As long as constant preconditioners are used, deflation does not seem to allow for significantly better convergence behaviors than simply using PCG (AMG), particularly for large numbers of DoFs. However, it is clear that the convergence behavior of PCG

preconditioned with a bJ or LORASC preconditioner can be significantly improved by deflation, and increasingly so for larger numbers of blocks and subdomains. Also, it should be noted that other classes of elliptic PDEs such as, e.g., the Helmholtz equation, for which there is no obvious choice of a robust and scalable preconditioner comparable to AMG for the diffusion equation considered here, may benefit more from deflation techniques provided that they manage to compensate for a less efficient preconditioning strategy. Furthermore, deflation may also prove useful in legacy software and/or when problem-specific, highly tuned preconditioners are used. Besides deflation, a potentially more promising option to improve the iterative solve of SPD linear systems arising from the discretization of stochastic elliptic PDEs is to periodically re-define the preconditioner for small groups (clusters) or subsequences of realizations.

Chapter 4

Preconditioning based on Voronoi quantizers of coefficient fields

4.1 Introduction

Consider a measurable space (Θ, Σ) with σ-algebra Σ rich enough to support all the random variables we encounter, and let $\Omega \subset \mathbb{R}^d$ be an open bounded domain. Let κ be a second-order $L^2(\Omega)$-valued random field, as defined by Eq. (1.85), with probability

measure μ on (Θ, Σ). The problem at hand consists of finding $u : \overline{\Omega} \times \Theta \to \mathbb{R}$ such that

$$\nabla \cdot [\kappa(x,\theta)\nabla u(x,\theta)] = f(x) \; \forall \; x \in \Omega \tag{4.1}$$

$$u(x,\theta) = g(x) \; \forall \; x \in \partial\Omega \tag{4.2}$$

is almost surely satisfied in a weak sense. That is, given a realization $\kappa(\cdot,\theta) \in L^2(\Omega)$, we are interested in the realization $u(\cdot,\theta) \in V \subset H^1_E(\Omega)$ defined as in Eq. (1.90), such that

$$\int_\Omega \kappa(x,\theta)\nabla u(x,\theta) \cdot \nabla v(x) \mathrm{d}x = \int_\Omega f(x)v(x)\mathrm{d}x \; \forall \; v \in V \tag{4.3}$$

in some approximation space $V = \mathrm{Span}\{\phi_i\}_{i=1}^n \in L^2(\Omega)$. We assume that κ is strictly positive and bounded almost everywhere in $\Omega \times \Theta$, as well as $f \in L^2(\Omega)$ and $g \in H^{1/2}(\partial\Omega)$. Then, there almost surely exists a unique realization $u(\cdot,\theta) = \sum_{i=1}^n u_i(\theta)\phi_i(\cdot) \in V$ satisfying Eq. (4.3), whose coefficients $\{u_i(\theta)\}_{i=1}^n$ are found by solving for the unique solution of an n-by-n SPD linear system of the form

$$\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta) \tag{4.4}$$

where $\mathbf{u}(\theta) = [u_1(\theta), \ldots, u_n(\theta)]$. We sometimes express the Galerkin operator as a function of a deterministic coefficient field, i.e., $\mathbf{A}(\kappa)$, in which case it should be understood that κ lies in

$$\mathcal{A} = \left\{\kappa \in L^2(\Omega), \; \kappa(x) > 0, \; \forall \, x \in \Omega\right\}. \tag{4.5}$$

By definition, $\kappa(\cdot,\theta) \in \mathcal{A}$ for almost all $\theta \in \Theta$.

Here, we focus on the iterative solve of Eq. (4.4) for large numbers of realizations. In particular, we rely on PCG using an SPD preconditioner $\mathbf{M}(\theta)$ which somehow approximates $\mathbf{A}(\theta)$. In other words, we consider the unique approximation $\mathbf{u}^{(j)}(\theta)$ of $\mathbf{u}(\theta)$ which satisfies

$$\mathbf{u}^{(j)}(\theta) - \mathbf{u}^{(0)} \; \in \mathcal{K}^{(j)}(\mathbf{M}^{-1}(\theta)\mathbf{A}(\theta), \mathbf{M}^{-1}(\theta)\mathbf{r}^{(0)}(\theta)) \tag{4.6}$$

$$\mathbf{r}^{(j)}(\theta) \; \perp \mathcal{K}^{(j)}(\mathbf{M}^{-1}(\theta)\mathbf{A}(\theta), \mathbf{M}^{-1}(\theta)\mathbf{r}^{(0)}(\theta)) \tag{4.7}$$

where $\mathcal{K}^{(j)}(\mathbf{A}(\theta), \mathbf{r}^{(0)}(\theta)) := \mathrm{Span}\{\mathbf{r}^{(0)}(\theta), \mathbf{A}(\theta)\mathbf{r}^{(0)}(\theta), \ldots, \mathbf{A}^{j-1}(\theta)\mathbf{r}^{(0)}(\theta)\}$ is the Krylov subspace of $\mathbf{A}(\theta)$ generated by the residual $\mathbf{r}^{(0)}(\theta) := \mathbf{A}(\theta)\mathbf{u}^{(0)} - \mathbf{b}(\theta)$ of an initial guess $\mathbf{u}^{(0)}$. This approximation is optimal in the sense that it minimizes $\|\mathbf{v} - \mathbf{u}(\theta)\|_{\mathbf{A}(\theta)}$ over the Krylov subspace, see [138]. We denote by $J(\theta)$ the smallest number j of iterations such that $\|\mathbf{A}(\theta)\mathbf{u}^{(j)}(\theta) - \mathbf{b}(\theta)\|_2 < \epsilon\|\mathbf{b}(\theta)\|_2$ for some $\epsilon > 0$. In other words, $J(\theta)$ is the number of necessary PCG iterations to reach a normwise backward error η_b of ϵ. Since $\mathbf{A}(\theta)$ is SPD, $J(\theta) \leq n$ as long as $\mathbf{u}^{(j)}(\theta)$ is computed with exact arithmetic. Meanwhile, a good preconditioner is such that $J(\theta) \ll n$ even when relying on finite arithmetic.

4.2 Preconditioning strategies

Let us define preconditioning strategies as consisting of both

1. A *P-quantizer* $q : \kappa \in \mathcal{A} \mapsto \hat{\kappa} \in \hat{\mathcal{A}}$ with centroidal coefficient fields given by the

elements of a codebook $\hat{\mathcal{A}} := \{\hat{\kappa}_p \in \mathcal{A},\ p = 1, \ldots, P\}$. The aim of q is to serve as a compact representation of the random coefficient field—see [50, 56, 93] for references on vector quantization, and [83, 45] for extensions to Hilbert and Banach function spaces.

2. A *preconditioner* $\mathbf{M} : \kappa \in \mathcal{A} \mapsto \mathbf{M}(\kappa) \in \mathrm{Sym}^+_{n\times n}(\mathbb{R})$ where $\mathrm{Sym}^+_{n\times n}(\mathbb{R})$ denotes the space of SPD matrices. We are interested in the composition $\mathbf{M} \circ q : \kappa \in \mathcal{A} \mapsto \hat{\mathbf{M}} \in \{\hat{\mathbf{M}}_1, \ldots, \hat{\mathbf{M}}_P\}$. The preconditioners $\hat{\mathbf{M}}_1, \ldots, \hat{\mathbf{M}}_P$ may consist of factorizations of the Galerkin operators of the centroidal fields, in which case we have $\hat{\mathbf{M}}_p := \mathbf{A}(\hat{\kappa}_p)$ for $p = 1, \ldots, P$. Other possible choices are: solving cycles of algebraic multigrids of $\mathbf{A}(\hat{\kappa}_p)$, in which case $\hat{\mathbf{M}}_p$ needs not even be known as long as one can evaluate $\mathbf{v} \in \mathbb{R}^n \mapsto \hat{\mathbf{M}}_p^{-1}\mathbf{v}$; block Jacobi (bJ), i.e., factorizations of diagonal blocks of $\mathbf{A}(\hat{\kappa}_p)$; domain decomposition. See [138] for more suggestions.

Let us first consider an invertible transport map $T : L^2(\Omega) \to L^2(\Omega)$ such that $T^{-1}\kappa$ has zero mean and is a second order random field which admits a Karhunen-Loève (KL) expansion. In order to introduce the KL expansion of $T^{-1}\kappa$, we must know the covariance function $C : \Omega \times \Omega \to \mathbb{R}$ given by $(x, x') \mapsto \mathbb{E}[T^{-1}\kappa(x)T^{-1}\kappa(x')]$. Moreover, we assume that the kernel C is such that the integral operator $f \in L^2(\Omega) \mapsto \int_\Omega C(x, \cdot)f(x)\mathrm{d}x$ is symmetric and positive semi-definite of rank n_{KL}. Then, the n_{KL} nontrivial eigen-pairs of the operator are such that $(\lambda_k, \Phi_k) \in \mathbb{R} \times V$ for $k = 1, \ldots, n_{\mathrm{KL}}$, and we have

$$\int_\Omega \int_\Omega C(x', x)\Phi_k(x')\mathrm{d}x' v(x)\mathrm{d}x = \lambda_k \int_\Omega \Phi_k(x)v(x)\mathrm{d}x \ \forall\, v \in V \tag{4.8}$$

with $\langle \Phi_k, \Phi_\ell \rangle_\Omega = \delta_{k\ell}$ and $\lambda_1 \geq \cdots \geq \lambda_{n_{\mathrm{KL}}} > 0$, where $\langle f, g \rangle_\Omega$ is the $L^2(\Omega)$ inner product

$$\langle f, g \rangle_\Omega := \int_\Omega f(x)g(x)\mathrm{d}x \ \ \forall\ f, g \in L^2(\Omega) \tag{4.9}$$

with induced norm

$$\|f\|^2_\Omega = \langle f, f \rangle_\Omega = \int_\Omega f(x)^2\mathrm{d}x. \tag{4.10}$$

Note that for all $u(\cdot, \theta), v(\cdot, \theta) \in V$ such that $u(x, \theta) = \sum_{i=1}^n u_i(\theta)\phi_i(x)$ and $v(x, \theta) = \sum_{i=1}^n v_i(\theta)\phi_i(x)$, we have

$$\langle u(\cdot, \theta), v(\cdot, \theta) \rangle_\Omega = \mathbf{u}(\theta)^T \boldsymbol{\phi}_{\mathbf{M}} \mathbf{v}(\theta) \tag{4.11}$$

where $\mathbf{u}(\theta) = [u_1(\theta), \ldots, u_n(\theta)]^T$, $\mathbf{v}(\theta) = [v_1(\theta), \ldots, v_n(\theta)]^T$ and $\boldsymbol{\phi}_{\mathbf{M}}$ denotes the mass matrix with components $\phi_{Mij} = \langle \phi_i, \phi_j \rangle_\Omega$. The random field $T^{-1}\kappa$ can then be expressed as

$$T^{-1}\kappa(x, \theta) = \sum_{k=1}^{n_{\mathrm{KL}}} \lambda_k^{1/2} \Phi_k(x)\xi_k(\theta) \tag{4.12}$$

where $\xi_1, \ldots, \xi_{n_{\mathrm{KL}}}$ are uncorrelated random variables. In the special case where $T^{-1}\kappa$ is a Gaussian process, the random variables $\xi_1, \ldots, \xi_{n_{\mathrm{KL}}}$ are independent and identi-

cally distributed standard normal. By only using the $m < n_{\mathrm{KL}}$ dominant eigen-pairs $(\lambda_1, \Phi_1), \ldots, (\lambda_m, \Phi_m)$, we can approximate $T^{-1}\kappa$ with

$$\hat{T}_m^{-1}\kappa(x,\theta) := \sum_{k=1}^{m} \lambda_k^{1/2} \Phi_k(x) \xi_k(\theta) \tag{4.13}$$

where $\xi_1, \ldots, \xi_m$ are the components of the m-dimensional random vector $\boldsymbol{\xi}$ with probability measure $\mu_{\boldsymbol{\xi}}$. We denote the underlying representation error by

$$\epsilon(\hat{T}_m^{-1}\kappa) := \mathbb{E}[\|\hat{T}_m^{-1}\kappa - T^{-1}\kappa\|_\Omega^2]. \tag{4.14}$$

Let us assume that $T^{-1}\kappa(x)$ has a stationary variance denoted by σ^2 and that the pairs $(\lambda_k, \Phi_k) \in \mathbb{R} \times V$ with $k = 1, \ldots, m$ properly approximate the dominant eigen-pairs of C. Then we have

$$\epsilon(\hat{T}_m^{-1}\kappa) = \sigma^2 - \sum_{k=1}^{m} \lambda_k. \tag{4.15}$$

As a means to more efficiently operate on realizations of the coefficient field, we make use of the projection

$$\hat{P}_m^{\leftarrow} : f \in L^2(\Omega) \mapsto \begin{bmatrix} \lambda_1^{-1/2} \langle \Phi_1, f \rangle_\Omega \\ \vdots \\ \lambda_m^{-1/2} \langle \Phi_m, f \rangle_\Omega \end{bmatrix} \in \mathbb{R}^m \tag{4.16}$$

as well as of

$$\hat{P}_m^{\rightarrow} : \boldsymbol{\chi} \in \mathbb{R}^m \mapsto \sum_{k=1}^{m} \lambda_k^{1/2} \chi_k \Phi_k(\cdot) \in V. \tag{4.17}$$

Note that if $f \in V$ admits a representation $f(x) = \sum_{i=1}^n f_i \phi_i(x)$ and $\Phi_k(x) = \sum_{i=1}^n \Phi_{k,i} \phi_i(x)$ for $k = 1, \ldots, m$, we have $\hat{P}_m^{\leftarrow}(f) = \boldsymbol{\Lambda}^{-1/2} \boldsymbol{\Phi}^T \boldsymbol{\phi}_{\mathbf{M}} \mathbf{f}$ where $\boldsymbol{\Phi}$ has components $\Phi_{ik} = \Phi_{k,i}$, $\boldsymbol{\Lambda} := \mathrm{diag}(\lambda_1, \ldots, \lambda_m)$ and $\mathbf{f} := [f_1, \ldots, f_n]$. Then, $T(\hat{P}_m^{\rightarrow}(\hat{P}_m^{\leftarrow}(T^{-1}\kappa(\cdot))))$ serves as an approximation of $\kappa(\cdot)$. Moreover, the use of $\hat{P}_m^{\leftarrow}$ and $\hat{P}_m^{\rightarrow}$ allows to substitute the quantization of κ, or more precisely of $\hat{T}_m^{-1}\kappa$, by a finite-dimensional problem. That is, we will be searching for a quantizer $q_2 : \eta \in \mathbb{R}^m \mapsto \hat{\eta} \in \hat{\mathcal{H}} \subset \mathbb{R}^m$ with a codebook $\hat{\mathcal{H}} := \{\hat{\eta}_1, \ldots, \hat{\eta}_P\}$ for the random vector $T_2^{-1}(\boldsymbol{\xi})$, where the invertible map $T_2 : \mathbb{R}^m \to \mathbb{R}^m$ is introduced to allow some flexibility in the design of q_2. For a given q_2, we are interested in the quantizer

$$q : \kappa(\cdot) \in \mathcal{A} \mapsto \widetilde{T}_m^{\rightarrow}(q_2(\widetilde{T}_m^{\leftarrow}\kappa(\cdot))) \in \hat{\mathcal{A}} \subset \mathcal{A} \tag{4.18}$$

in which we made use of the composed mapping $\widetilde{T}_m^{\rightarrow} := T \circ \hat{P}_m^{\rightarrow} \circ T_2$ such that $\widetilde{T}_m^{\leftarrow} = T_2^{-1} \circ \hat{P}_m^{\leftarrow} \circ T^{-1}$. Then, the application of a strategy consists of preconditioning the iterative solve of $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ with $\mathbf{M}(\theta) := \mathbf{M}(q(\kappa(\cdot, \theta)))$. To do so, there remains to design q_2 and to select a type of preconditioner.

4.2.1 Optimal preconditioning strategies

An optimal preconditioning strategy is one that somehow minimizes the iteration number J. In theory, one can define a quantizer $\tilde{q}$ that maps every element of $\mathcal{A}$ to itself. Then, a preconditioning strategy based on $\tilde{q}$ with $\mathbf{M}(\theta) := \mathbf{A}(\kappa(\cdot, \theta))$ is such that $J(\theta) = 1$ almost everywhere in Θ. While being optimal, this solution has no practical use. The reasons are the following. First, computing a precise factorization of $\mathbf{A}(\theta)$ is at least as costly as solving $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$. Second, computing or setting-up a preconditioner carries some computational and memory costs. Thus, P will be limited by practical considerations, namely, the total memory available for a simulation. P may be referred to as the rate of quantization, and we are interested in fixed rate quantization. Another matter of concern lies in the scheduling of computations. That is, when allocating resources for a simulation, one may not only want to minimize $\mathbb{E}[J]$, but also some higher order moments, or even to impose some distribution on J. A first step in these directions lies in the formulation of measurable information about J which statistically relates to the random coefficient field and its quantizer.

Irrespective of the type of preconditioner used, the representation error of $\kappa(\cdot, \Theta)$ by $q(\kappa(\cdot, \Theta))$ is characterized by a distortion

$$w(q, d) := \mathbb{E}[d(\kappa, q(\kappa))] = \int_{\Theta} d(\kappa(\cdot, \theta), q(\kappa(\cdot, \theta)))\mathrm{d}\mu(\theta) \tag{4.19}$$

where the distortion functional (or divergence) $d : \mathcal{A} \times \mathcal{A} \to [0, \infty)$ measures proximity between realizations of the coefficient field. A loose definition of proximity allows, in principle, to recast a variety of functionals in the form of distortions. Given our objective to minimize the number of solver iterations J, we should aim at using divergences d such that minimizing $w(q, d)$ over all P-quantizers $q \in \mathcal{Q}_P$ allows to minimize some relevant information about J.

Let us denote by $\mathcal{V}_P \subset \mathcal{Q}_P$ the set of all Voronoi P-quantizers. That is, for all $q \in \mathcal{V}_P$, there exists $\hat{\mathcal{A}} := \{\hat{\kappa}_1, \ldots, \hat{\kappa}_P\} \subset \mathcal{A}$ such that

$$q : \kappa \in \mathcal{A} \mapsto \sum_{p=1}^{P} \hat{\kappa}_p \mathbf{1}[\kappa \in \mathcal{A}_p] \tag{4.20}$$

where

$$\mathcal{A}_p \subset \{\kappa \in \mathcal{A},\ d(\kappa, \hat{\kappa}_p) \leq d(\kappa, \hat{\kappa}_q),\ q = 1, \ldots, P\}\,, \tag{4.21}$$

and $\mathcal{A}_1, \ldots, \mathcal{A}_P$ form a Borel partition of $\mathcal{A}$. We say that q is the $\hat{\mathcal{A}}$-nearest projection. The advantage of Voronoi quantizers lies in that, for all pairs $(q, q') \in \mathcal{V}_P \times \mathcal{Q}_P$ of quantizers with the same codebook $\hat{\mathcal{A}}$, we have $w(q, d) \leq w(q', d)$. Therefore, in an attempt to minimize distortion, we rely on Voronoi quantizers.

Consider a Voronoi partition $\mathcal{A}_1, \ldots, \mathcal{A}_P$ induced by a quantizer q with an unknown codebook $\hat{\mathcal{A}} := \{\hat{\kappa}_1, \ldots, \hat{\kappa}_P\}$. We can then express the distortion as a functional of the codebook, i.e., $w(\hat{\mathcal{A}}, d)$. To do so, we define a conditional expectation operator $\mathbb{E}_p[\cdot] := \mathbb{E}[\cdot \,|\, \kappa \in \mathcal{A}_p]$ and an attribution frequency $f_p := \mu(\mathcal{A}_p)$ so that $\sum_{p=1}^{P} f_p = 1$. We also introduce the local distortion $w_p(q, d) := \mathbb{E}_p[d(\kappa, q(\kappa))]$ which we also express in terms of

the centroidal field, i.e., $w_p(\hat{\kappa}_p, d)$, so that

$$w(\hat{\mathcal{A}}, d) = \sum_{p=1}^{P} f_p w_p(\hat{\kappa}_p, d). \tag{4.22}$$

Then, the following can be shown [45]. If $f_p > 0$ and $\mathbb{E}_p[\kappa] \in ri(\mathcal{A})$ for $p = 1, \ldots, P$, where $ri(\mathcal{A})$ denotes the relative interior of $\mathcal{A}$, then, in order to minimize $w(\hat{\mathcal{A}}, d)$, the codebook $\hat{\mathcal{A}} = \{\hat{\kappa}_1, \ldots, \hat{\kappa}_P\}$ must be such that

$$\hat{\kappa}_p \in \arg\min_{\hat{\kappa} \in ri(\mathcal{A})} w_p(\hat{\kappa}, d) \tag{4.23}$$

for $p = 1 \ldots, P$. That is, given a Voronoi partition, the distortion is minimized by selecting centroidal fields which are minimizers of local distortions.

Among all distortion functionals, we wish to consider those that allow for the minimization of Eq. (4.19) with respect to q. In particular, Bregman divergences [20] admit properties which allow to devise algorithms for the minimization of distortion in both finite [8], and infinite-dimensional spaces [45]. They can be defined as follows.

Definition 1. *Let $\varphi : L^2(\Omega) \to \mathbb{R}$ be a strictly convex, twice-continuously Fréchet-differentiable functional. Then, a Bregman divergence is given by*

$$d_\varphi(\kappa, \kappa') = \varphi(\kappa) - \varphi(\kappa') - \delta\varphi(\kappa, \kappa' - \kappa) \; \forall \, \kappa, \kappa' \in \mathcal{A}, \tag{4.24}$$

where $\delta\varphi(\kappa, \kappa' - \kappa)$ denotes the Fréchet derivative of φ at κ in the direction of $\kappa' - \kappa$, see [47].

An important result of [8] was extended by [45] to the case of functional Bregman divergences. That is, for any Bregman divergence d_φ, if a (Borel) subset $\mathcal{A}_p$ of $\mathcal{A}$ is such that $f_p > 0$ and $\mathbb{E}_p[\kappa] \in ri(\mathcal{A})$, then, the local distortion $w_p(\hat{\kappa}, d_\varphi)$ reaches its infimum at a unique element of $ri(\mathcal{A})$, namely $\mathbb{E}_p[\kappa]$. A consequence of this result is that k-means algorithms can be designed to approximate optimal quantizers that converge to local minima of distortion.

4.3 Computation of stationary quantizers

There exist several numerical methods for the computation of stationary vector quantizers. Coincidentally, we compute q_2 which we transform following Eq. (4.18) to obtain a quantizer of the coefficient field. In particular we consider L^2 quantizers of $T_2^{-1}(\boldsymbol{\xi})$. That is, we consider the following distortion

$$w_2(q_2) := \mathbb{E}[\|T_2^{-1}(\boldsymbol{\xi}) - q_2(T_2^{-1}(\boldsymbol{\xi}))\|^2] = \int_\Theta \|T_2^{-1}(\boldsymbol{\xi}) - q_2(T_2^{-1}(\boldsymbol{\xi}))\|^2 \mathrm{d}\mu_{\boldsymbol{\xi}}(\theta). \tag{4.25}$$

We let q_2 be Voronoi, and denote by $\mathcal{H}_1, \ldots, \mathcal{H}_P$ the partition of $T_2^{-1}(\mathbb{R}^m)$ induced by q_2. Then, we have

$$q_2(T_2^{-1}(\boldsymbol{\xi})) := \sum_{p=1}^{P} \hat{\boldsymbol{\eta}}_p \mathbf{1}[T_2^{-1}(\boldsymbol{\xi}) \in \mathcal{H}_p] \tag{4.26}$$

where $\hat{\boldsymbol{\eta}}_p = T_2^{-1}(\hat{\boldsymbol{\xi}}_p)$, and the following decomposition of distortion into local contributions:

$$w_2(q_2) = \sum_{p=1}^{P} w_{2,p}(q_2)\mu_{\boldsymbol{\xi}}(T_2^{-1}(\mathcal{H}_p)) \tag{4.27}$$

where $w_{2,p}(q_2) := \mathbb{E}[\|T_2^{-1}(\boldsymbol{\xi}) - q_2(T_2^{-1}(\boldsymbol{\xi}))\|^2 \,|\, T_2^{-1}(\boldsymbol{\xi}) \in \mathcal{H}_p]$. A P-quantizer q_2 is P-stationary if it is a critical point of $w_2 : \mathcal{Q}_P \to \mathbb{R}^+$, or equivalently the quantizer q_2 is Voronoi and such that $q_2(T_2^{-1}(\boldsymbol{\xi})) = \mathbb{E}[T_2^{-1}(\boldsymbol{\xi}) \,|\, q_2(T_2^{-1}(\boldsymbol{\xi}))]$. Obviously, an optimal P-quantizer is always P-stationary.

In practice, it is not tractable to measure with $\mu_{\boldsymbol{\xi}}$, and empirical measures need be introduced. That is, we are given an n_s-sample $\kappa_1, \ldots, \kappa_{n_s}$ of i.i.d. observations of the random coefficient field, and we compute $\boldsymbol{\xi}_s := \hat{P}_m^{\leftarrow}(T^{-1}\kappa_s)$ for $s = 1, \ldots, n_s$. The distortion $w_2(q_2)$ is then approximated by

$$w_2^{(n_s)}(q_2) := \frac{1}{n_s}\sum_{s=1}^{n_s} \|T_2^{-1}(\boldsymbol{\xi}_s) - q_2(T_2^{-1}(\boldsymbol{\xi}_s))\|^2 \tag{4.28}$$

which is also given by

$$w_2^{(n_s)}(q_2) = \sum_{p=1}^{P} f_{2,p}^{(n_s)} w_{2,p}^{(n_s)}(q_2) \tag{4.29}$$

where

$$f_{2,p}^{(n_s)} = \frac{1}{n_s}\sum_{s=1}^{n_s} \mathbf{1}[T_2^{-1}(\boldsymbol{\xi}_s) \in \mathcal{H}_p] \tag{4.30}$$

is the empirical measure of $\mathcal{H}_p$ associated with $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{n_s}$, and

$$w_{2,p}^{(n_s)}(q_2) = \frac{1}{f_{2,p}^{(n_s)} n_s}\sum_{s=1}^{n_s} \|T_2^{-1}(\boldsymbol{\xi}_s) - q_2(T_2^{-1}(\boldsymbol{\xi}_s))\|^2 \mathbf{1}[T_2^{-1}(\boldsymbol{\xi}_s) \in \mathcal{H}_p]. \tag{4.31}$$

Different algorithms exist to compute finite dimensional stationary quantizers. The k-means algorithm, sometimes referred to as Lloyd's method [75, 141], and the competitive learning vector quantization (CLVQ) algorithm [117] are presented in details along with a brief mention of their variants and other existing methods.

4.3.1 k-means

The term k-means was first used by MacQueen in 1967 [97], though the idea goes back to Steinhaus in 1956 [149]. The standard algorithm was first proposed by Lloyd of Bell Labs in 1957 as a technique for pulse-code modulation, although it was not published as a journal article until 1982 [95]. The algorithm works as follows, see Algo. 25. Given an initial codebook $\hat{\mathcal{H}}^{(0)}$ for a P-quantizer, alternate between the two following steps. First, compute the Voronoi partition induced by the codebook for a given Bregman divergence. Second, re-calculate the centroidal fields of the codebook as the conditional expectations of the new Voronoi partition. The k-means algorithm converges, but slowly for large values of the dimension m. It attempts to solve an NP-hard problem, even for low values

of m, at a cost of $\mathcal{O}(n_s P)$ per iteration. Both online [17] and mini-batch [144] variants of the k-means algorithms exist. A source of variety of k-means algorithms comes from the different methods used to design the initial quantizer. For instance, some of the well-known initialization methods are presented as follows. First, the method of Forgy [46], where the initial centroids $\hat{\boldsymbol{\eta}}_1^{(0)}, \ldots, \hat{\boldsymbol{\eta}}_P^{(0)}$ are chosen randomly from the data $\mathcal{H}$. Second, the minmax method consists of choosing the first centroid $\hat{\boldsymbol{\eta}}_1^{(0)}$ randomly. Then the p-th centroid $\hat{\boldsymbol{\eta}}_p^{(0)}$ with $p \in \{2, \ldots, P\}$ is chosen to be the point that has the largest minimum distance to the previously selected centroids, i.e., $\hat{\boldsymbol{\eta}}_1^{(0)}, \hat{\boldsymbol{\eta}}_2^{(0)}, \ldots, \hat{\boldsymbol{\eta}}_{p-1}^{(0)}$. This method yields a complexity of $\mathcal{O}(n_s P)$ where n_s denotes the size of the data set $\mathcal{H}$. Third, the k-means++ method consists of selecting the first centroid $\hat{\boldsymbol{\eta}}_1^{(0)}$ randomly. Then the p-th centroid $\hat{\boldsymbol{\eta}}_p^{(0)}$ such that $p \in \{2, \ldots, P\}$ is chosen to be $\boldsymbol{\eta}_s \in \mathcal{H}$ with a probability of $D(\boldsymbol{\eta}_s) / \sum_{t=1}^{n_s} D(\boldsymbol{\eta}_t)$ where $D(\boldsymbol{\eta}_s)$ denotes the minimum distance from a point $\boldsymbol{\eta}_s$ to the previously selected centroids. A more extensive review of initialization methods is given by [25].

Algorithm 25 k-means($\mathcal{H}$, $\hat{\mathcal{H}}^{(0)}$)

Input: Data $\mathcal{H} := \{\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_{n_s}\}$,
Initial codebook $\hat{\mathcal{H}}^{(0)} := \{\hat{\boldsymbol{\eta}}_1^{(0)}, \ldots, \hat{\boldsymbol{\eta}}_P^{(0)}\} \subset \mathbb{R}^m$.
Output: Voronoi P-quantizers $q_2^{(t)} : \mathbb{R}^m \to \hat{\mathcal{H}}^{(t)}$ induced by $\{\hat{\mathcal{H}}^{(t)}\}_{t=0,1,\ldots}$ are s.t. $w_2^{(n_s)}(q_2^{(t)})$ converges to a local minimum of $w_2^{(n_s)} : \mathcal{Q}_P \to \mathbb{R}^+$ as $t \to \infty$.

$t := 0$
while not converged **do**
▷ Compute partition $\mathcal{H}_1^{(t)}, \ldots, \mathcal{H}_P^{(t)}$ of $\mathcal{H}$
  for $p = 1, \ldots, P$ **do**
    Pick $\mathcal{H}_p^{(t)} \subseteq \{\boldsymbol{\eta}_s \in \mathcal{H} \text{ s.t. } \|\boldsymbol{\eta}_s - \hat{\boldsymbol{\eta}}_p^{(t)}\|^2 \leq \|\boldsymbol{\eta}_s - \hat{\boldsymbol{\eta}}_q^{(t)}\|^2 \text{ for } q = 1, \ldots, P\}$
    s.t. $\mathcal{H}_p^{(t)} \cap \mathcal{H}_{q \neq p}^{(t)} = \emptyset$ and $\cup_{p=1}^P \mathcal{H}_p^{(t)} = \mathcal{H}$
  end for
▷ Compute codebook $\hat{\mathcal{H}}^{(t+1)} = \{\hat{\boldsymbol{\eta}}_1^{(t+1)}, \ldots, \hat{\boldsymbol{\eta}}_P^{(t+1)}\}$
  for $p = 1, \ldots, P$ **do**
    $\hat{\boldsymbol{\eta}}_p^{(t+1)} := |\mathcal{H}_p^{(t)}|^{-1} \sum_{\boldsymbol{\eta}_s \in \mathcal{H}_p^{(t)}} \boldsymbol{\eta}_s$
  end for
  $t := t + 1$
end while

4.3.2 Competitive learning

It is often argued (see [117]) that Lloyd's method, i.e., k-means, is untractable in multiple dimensions, i.e., when m becomes large. Approaches based on learning algorithms are then favored when m is large so as to yield better results, see [78] for an example in automatic classification. In particular, we consider the CLVQ algorithm (see [161] or [13]) which is used by Pagès and Printemps [118] for the quantization of Gaussian RVs, as well as by Pagès [117] for numerical integration. The algorithm works as follows, see Algo. 26. First, the competitive phase of the method is the most consuming because it uses a nearest neighbor search at each step to find the nearest centroid in the codebook

to the randomly generated vector. The second phase, called the learning phase, updates the nearest centroid of the quantizer by a homothety.

Algorithm 26 CLVQ($\mathcal{H}$, $\hat{\mathcal{H}}^{(0)}$, $\{\gamma_t\}_{t=0,1,2,\dots}$)

Input: Data $\mathcal{H} := \{\boldsymbol{\eta}_1, \dots, \boldsymbol{\eta}_{n_s}\}$,
Initial codebook $\hat{\mathcal{H}}^{(0)} := \{\hat{\boldsymbol{\eta}}_1^{(0)}, \dots, \hat{\boldsymbol{\eta}}_P^{(0)}\} \subset \mathbb{R}^m$,
Step sequence $\{\gamma_t\}_{t=0,1,2,\dots}$ s.t. $\gamma_t > 0 \, \forall \, t$, $\sum_{t\geq 1} \gamma_t = \infty$ and $\sum_{t\geq 1} \gamma_t^2 < \infty$.

Output: Voronoi P-quantizers $q_2^{(t)} : \mathbb{R}^m \to \hat{\mathcal{H}}^{(t)}$ induced by $\{\hat{\mathcal{H}}^{(t)}\}_{t=0,1,\dots}$ s.t. $w_2^{(n_s)}(q_2^{(t)})$ converges to a local minimum of $w_2^{(n_s)} : \mathcal{Q}_P \to \mathbb{R}^+$ as $t \to n_s$ and $n_s \to \infty$.

$t := 0$
while not converged **and** $t < n_s$ **do**
  ▷ Competitive phase: find nearest neighbor
  Pick $p \in [1, P]$ s.t. $\|\boldsymbol{\eta}_{t+1} - \hat{\boldsymbol{\eta}}_p^{(t)}\|^2 \leq \|\boldsymbol{\eta}_{t+1} - \hat{\boldsymbol{\eta}}_q^{(t)}\|^2$ for $q = 1, \dots, P$
  ▷ Learning phase: update codebook $\hat{\mathcal{H}}^{(t+1)} = \{\hat{\boldsymbol{\eta}}_1^{(t+1)}, \dots, \hat{\boldsymbol{\eta}}_P^{(t+1)}\}$
  for $q = 1, \dots, P$ **do**
    if $q \neq p$ **then**
      $\hat{\boldsymbol{\eta}}_q^{(t+1)} := \hat{\boldsymbol{\eta}}_q^{(t)}$
    else
      $\hat{\boldsymbol{\eta}}_q^{(t+1)} := \hat{\boldsymbol{\eta}}_q^{(t)} - \gamma_{t+1}(\hat{\boldsymbol{\eta}}_q^{(t)} - \boldsymbol{\eta}_{t+1})$
    end if
  end for
  $t := t + 1$
end while

CLVQ is an online algorithm that can be seen as a particular case of the stochastic gradient method with decreasing step. A good reference here is [41]. It is assumed that the positive gain parameter sequence $\gamma_0, \gamma_1, \gamma_2, \dots$ satisfies

$$\gamma_t > 0 \; \forall \; t \quad , \quad \sum_{t\geq 1} \gamma_t = +\infty \quad , \quad \sum_{t\geq 1} \gamma_t^2 < +\infty. \tag{4.32}$$

Then, the algorithm is shown to converge. However, in general, we cannot ensure that the algorithm converges to a global minimum. In higher dimensions, uniqueness of stationary quantizers clearly always fails (see [56]), so that the CLVQ reaches, a priori, only locally optimal quantizers. Note that, even when using k-means, there is no reason to suppose that the obtained quantizers are the optimal ones. An important aspect of the CLVQ algorithm is the choice of the gain sequence. Usually, some heuristic methods are developed based on the distribution of the quantized random vector, see [118]. For instance, a gain sequence usually used [142] is given by

$$\gamma_t = \gamma_0 \frac{\alpha}{t^c + \beta}, \quad t \geq 1 \tag{4.33}$$

for some constants γ_0, α, β and $c > 1/2$. This is the gain sequence used in the numerical experiments of this chapter.

4.3.3 Other methods

Several other methods exist to reach estimators of stationary vector quantizers, see [25] for a list of methods applied to color quantization (i.e., with $m = 3$). In particular, we mention the following approaches. First, the octree method [51] is a two-phase method that starts by building an octree (a tree data structure in which each internal node has up to eight children) that represents the vector distribution of the input data $\mathcal{H}$ and then, starting from the bottom of the tree, prunes the tree by merging its nodes until P vectors are obtained. Second, the modified minmax method [166] chooses the first centroid $\hat{\boldsymbol{\eta}}_1^{(0)}$ arbitrarily from the data set $\mathcal{H}$ and the p-th centroid $\hat{\boldsymbol{\eta}}_p^{(0)}$ with $p \in \{2, \ldots, P\}$ is chosen to be the point that has the largest minimum weighted L^2 distance to the previously selected centroids $\hat{\boldsymbol{\eta}}_1^{(0)}, \ldots, \hat{\boldsymbol{\eta}}_{p-1}^{(0)}$. Each of these initial centroids is then recalculated as the mean of the vectors assigned to it. Third, the split & merge method [21] has two phases. First, the data set $\mathcal{H}$ is partitioned uniformly into $B > P$ clusters. This initial set of partitions is represented as an adjacency graph. In the second phase, $B - P$ merge operations are performed to obtain the final P clusters. At each step of the second phase, the pair of clusters with the minimum joint quantization error are merged.

4.4 Quantizations based on deterministic grids

Until now, we have attempted to design P-quantizers of the m modes of a truncated KL representation with a representation error denoted by $\epsilon(\hat{T}_m^{-1}\kappa)$. As it turns out, depending on the number P of preconditioners wanted, it is not always the better option to consider all the modes of the approximating KL representation to perform the partitioning. To more efficiently impact the preconditioning strategy, it is better to limit the number of modes considered for the quantization based on the number of preconditioners wanted. That is, the number of modes considered should depend on with the number of preconditioners. One way to establish a dependence of the number of modes to the desired number of preconditioners is to resort to deterministic grids. The deterministic grid is dimensioned through a positive grid parameter denoted by s. To indicate the number of modes of the KL representation considered for the quantization, we resort to the notation $q_2^{(m)}$ where m is the number of modes. In particular, for $m = 1$, we use

$$q_2^{(1)}(\xi) = T_2^{-1}(0)\mathbf{1}[-s/2 \leq \xi < s/2] + T_2^{-1}(-s)\mathbf{1}[\xi < s/2] + T_2^{-1}(s)\mathbf{1}[s/2 \leq \xi] \tag{4.34}$$

so as to provide a symmetric solution. In order for $q_2^{(1)}$ to yield a partition with constant attribution frequencies, we let $s := 2F^{-1}(2/3) \approx 0.8614$. In higher dimensions, i.e., for $m > 1$, we have

$$q_2^{(m)}(\boldsymbol{\xi}) = \sum_{p=0}^{2^m} T_2^{-1}(\hat{\boldsymbol{\xi}}_p)\mathbf{1}[T_2^{-1}(\boldsymbol{\xi}) \in \mathcal{H}_p] \tag{4.35}$$

where $\mathcal{H}_0, \ldots, \mathcal{H}_{2^m}$ form a Voronoi partition of $T_2^{-1}(\mathbb{R}^m)$ and are given such that

$$\mathcal{H}_p \subset \left\{ T_2^{-1}(\boldsymbol{\xi}),\ \boldsymbol{\xi} \in \mathbb{R}^m,\ \|\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}_p\| \leq \|\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}_q\|,\ q = 0, \ldots, 2^m \right\} \tag{4.36}$$

in which the components of $\hat{\boldsymbol{\xi}}_0, \ldots, \hat{\boldsymbol{\xi}}_{2^m}$ are given by Algo. 27. These quantizations based

Algorithm 27 GetGridCoordinates(s, m)

Input: Grid parameter s,
Number of KL modes m
Output: Centroids $T_2^{-1}(\hat{\boldsymbol{\xi}}_0), \ldots, T_2^{-1}(\hat{\boldsymbol{\xi}}_{2^m})$ of quantizer $q_2^{(m)}$.

$p := 0$
$\hat{\boldsymbol{\xi}}_p := \mathbf{0}$
for $\hat{\xi}_{p,1} \in (-s, s)$ **do**
  for $\hat{\xi}_{p,2} \in (-s, s)$ **do**
    $\vdots$
    for $\hat{\xi}_{p,m} \in (-s, s)$ **do**
      $p := p + 1$
      $\hat{\boldsymbol{\xi}}_p := [\hat{\xi}_{p,1}, \ldots, \hat{\xi}_{p,m}]^T$
    end for
  end for
end for
return $T_2^{-1}(\hat{\boldsymbol{\xi}}_0), \ldots, T_2^{-1}(\hat{\boldsymbol{\xi}}_{2^m})$

on deterministic grids have shown good effects on the preconditioning strategy. They show a dependence between the size of the P-quantizer and the number m of KL modes, namely, we have $P = 1 + 2^m$.

4.5 Choice of map T_2

The choice of the map T_2 bears important consequences on the design of the quantizer q_2. Here, we present two choices of maps later used for our numerical experiments, namely, minimizing the $L^2(\Omega)$-distortion of $\hat{T}_m^{-1}\kappa$, and clustering for constant frequencies. Beyond the choice of T_2, it is particularly important to understand the distribution of the preconditioned iterative solves among the P preconditioners for balancing the computational load in distributed implementations.

4.5.1 Minimizing the $L^2(\Omega)$-distortion of $\hat{T}_m^{-1}\kappa$

The first choice of map, which aims at minimizing the $L^2(\Omega)$-distortion of $\hat{T}_m^{-1}\kappa$, is

$$T_2^{-1} : \boldsymbol{\chi} \mapsto \boldsymbol{\phi}_{\mathbf{M}}^{1/2} \boldsymbol{\Phi} \boldsymbol{\Lambda}^{1/2} \boldsymbol{\chi} \tag{4.37}$$

where we recall that $\phi_{Mij} = \langle \phi_i, \phi_j \rangle_\Omega$ denotes the components of the mass matrix $\boldsymbol{\phi}_{\mathbf{M}}$ and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_m)$. It is assumed that the eigenfunction Φ_k of Eq. (4.8) is written as $\Phi_k(x) = \sum_{i=1}^n \Phi_{k,i}\phi_i(x)$ so that the component Φ_{ik} of $\boldsymbol{\Phi} \in \mathbb{R}^{n \times m}$ stores $\Phi_{k,i}$. Then, using Eq. (4.13) along with the fact that $\|u(\cdot,\theta)\|_\Omega^2 = \langle u(\cdot,\theta), u(\cdot,\theta)\rangle_\Omega = \mathbf{u}(\theta)^T \boldsymbol{\phi}_{\mathbf{M}} \mathbf{u}(\theta)$ where

$\mathbf{u}(\theta) = [u_1(\theta), \ldots, u_2(\theta)]$ and $u(\theta) = \sum_{i=1}^{n} u_i(\theta)\phi_i(x)$, we have that

$$\|T_2^{-1}(\boldsymbol{\xi}(\theta))\|^2 = \boldsymbol{\xi}(\theta)^T \boldsymbol{\Lambda}^{1/2} \boldsymbol{\Phi}^T \boldsymbol{\phi}_{\mathbf{M}} \boldsymbol{\Phi} \boldsymbol{\Lambda}^{1/2} \boldsymbol{\xi}(\theta) = \langle \hat{T}_m^{-1}\kappa(\cdot,\theta), \hat{T}_m^{-1}\kappa(\cdot,\theta)\rangle_\Omega = \|\hat{T}_m^{-1}\kappa(\cdot,\theta)\|_\Omega^2. \tag{4.38}$$

Already, Eq. (4.38) states that the functional $L^2(\Omega)$-distortion of $\hat{T}_m^{-1}\kappa(\cdot,\theta)$ induced by $\hat{P}_m^{\rightarrow}(T_2(q_2(\widetilde{T}_m^{\leftarrow}(\kappa(\cdot,\theta)))))$ is equivalently given by the finite-dimensional L^2-distortion of $T_2^{-1}(\boldsymbol{\xi}(\theta))$ induced by $q_2(T_2^{-1}(\boldsymbol{\xi}(\theta)))$.

An even more useful observation is done by invoking the orthonormality of the eigenfunctions $\Phi_1, \ldots, \Phi_m$ along with Eq. (4.13) to show that

$$\|\hat{T}_m^{-1}\kappa(\cdot,\theta)\|_\Omega^2 = \sum_{k=1}^{m} \lambda_k \xi_k(\theta)^2 = \boldsymbol{\xi}(\theta)^T \boldsymbol{\Lambda} \boldsymbol{\xi}(\theta) = \|\boldsymbol{\Lambda}^{1/2}(\boldsymbol{\xi}(\theta))\|^2. \tag{4.39}$$

That is, the map

$$T_2^{-1} : \boldsymbol{\chi} \mapsto \boldsymbol{\Lambda}^{1/2}\boldsymbol{\chi} \tag{4.40}$$

is also such that $\|T_2^{-1}(\boldsymbol{\xi}(\theta))\|^2 = \|\hat{T}_m^{-1}\kappa(\cdot,\theta)\|_\Omega^2$. Note however that we have $n \gg m$, so that computing $\|T_2^{-1}(\boldsymbol{\xi}(\theta))\|^2$ with the map given by Eq. (4.40) is far less expensive than using Eq (4.37).

As a consequence, Eq. (4.40) will be favored, which involves a (dense) matrix-vector product in $\mathcal{O}(nm^2)$ operations. We will see, as revealed by our numerical experiments, that the minimization of the $L^2(\Omega)$-distortion of $\hat{T}_m^{-1}\kappa$ yields stationary quantizers such that f_p depends on the magnitude of $\|\hat{\boldsymbol{\xi}}_p\|^2$, that is, f_p is larger (smaller) for centroids with smaller (larger) $\|\hat{\boldsymbol{\xi}}_p\|^2$. Processes with centroid $\hat{\boldsymbol{\xi}}_p$ such that $\|\hat{\boldsymbol{\xi}}_p\|^2$ is small correspond to linear systems that tend to be better conditioned and thus result in fewer solver iterations J. These processes with smaller $\|\hat{\boldsymbol{\xi}}_p\|^2$ will be solving more, but easy to solve linear systems, while those with larger $\|\hat{\boldsymbol{\xi}}_p\|^2$ will be solving fewer, but more difficult to solve linear systems.

4.5.2 Clustering for constant frequencies

The second choice of map, which aims at clustering for constant frequencies, is

$$T_2^{-1} : \boldsymbol{\chi} \mapsto \boldsymbol{\Lambda}^{1/2} F_\xi \circ \boldsymbol{\chi} \;\text{ where }\; F_\xi \circ \boldsymbol{\chi} = \begin{bmatrix} F_\xi(\chi_1) \\ \vdots \\ F_\xi(\chi_m) \end{bmatrix} \tag{4.41}$$

in which $F_\xi(\chi) = \Pr[\xi \leq \chi]$, assuming $T^{-1}\kappa$ is a Gaussian process. Our experiments will show that this choice of T_2^{-1} yields stationary quantizers q_2 with $f_1 \approx \cdots \approx f_P$, i.e., with constant frequencies. This means that both types of processes with centroid $\hat{\boldsymbol{\xi}}_p$ such that $\|\hat{\boldsymbol{\xi}}_p\|^2$ is small, and such that $\|\hat{\boldsymbol{\xi}}_p\|^2$ is large, will solve as many linear systems.

4.5.3 Effect on load balancing

It is interesting to consider these two choices of map T_2 from the perspective of load balancing. Load balancing involves allocating tasks to processes so as to ensure the most efficient use of resources. Assuming a preconditioning strategy of P preconditioners with sampled linear solves distributed a priori on as many or fewer processes, the overall time of the distributed solves is given by the process which takes the longest time to process all its attributed linear solves. This elapsed time is well accounted for by solver iterations. Hence, although the distribution of $\mathbb{E}_p[J]$ as a function of p is an interesting measure of the performance of the preconditioning strategy, the values taken by the following quantity are more indicative of the idle time of the distributed implementation:

$$\Sigma_{J(\hat{\Theta}_p)} = \sum_{\theta \in \hat{\Theta}_p} J(\theta) \tag{4.42}$$

where $\hat{\Theta}_p$ is the set of all the realizations of the linear systems preconditioned by $\hat{\mathbf{M}}_p$. The sets $\hat{\Theta}_1, \ldots, \hat{\Theta}_P$ form a partition of the finite subset $\hat{\Theta} \subset \Theta$ of realizations for a given simulation. Then, the distribution of $\Sigma_{J(\hat{\Theta}_p)}$ as a function of p serves as a detailed representation of process activity. In particular, we are interested in $\max_{p \in [1,P]} \Sigma_{J(\hat{\Theta}_p)}$, as it directly relates to the total time of the simulation.

4.6 Local interpolation of preconditioner realizations

Monte Carlo simulations with a quantized preconditioner can be run in parallel on a network of computational nodes, each of which stores only some of the centroidal preconditioners. To sample without bias, we can draw realizations $\kappa(\theta)$ of the coefficient field from a master node and attribute each realization to the node with the closest centroidal field, i.e., best-suited preconditioner, in some sense. Assuming that this node stores more than one centroidal preconditioner, we want to leverage all (or some of) these local preconditioners so as to further improve the preconditioning of the system. This can be done using an interpolation of the preconditioner realization $\mathbf{M}^{-1}(\theta)$ via an optimal projection in the linear span of all (or some of) the local preconditioners. In what follows, we summarize how the works of Zahm and Nouy [168] is used to accelerate the computation of such realization-dependent projections in the context of Monte Carlo simulations with a P-quantization of preconditioners.

Let us consider a network of M nodes with a P-quantized preconditioner such that $M \leq P$. We assume that every single centroidal preconditioner is stored entirely on one or another node, and each node stores at least one preconditioner. The preconditioners stored by the m-th node are $\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_1^{(m)}), \ldots, \mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_{P_m}^{(m)})$ where P_m denotes the number of local preconditioners. Then, $\boldsymbol{\xi}$ is drawn randomly, close to a centroid of the m-th node. Following the work of Zahm and Nouy [168], we approximate the action of the realization-dependent preconditioner $\mathbf{M}^{-1}(\boldsymbol{\xi})$ with an interpolation of the form

$$\hat{\mathbf{M}}_m^{-1}(\boldsymbol{\xi}) = \sum_{p=1}^{P_m} \alpha_p^{(m)}(\boldsymbol{\xi}) \mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)}). \tag{4.43}$$

Although it is possible to find $\alpha_1^{(m)}(\boldsymbol{\xi}), \ldots, \alpha_{P_m}^{(m)}(\boldsymbol{\xi}) \in \mathbb{R}$ that minimize the condition number of $\hat{\mathbf{M}}_m^{-1}(\boldsymbol{\xi})\mathbf{A}(\boldsymbol{\xi})$, this problem [99] can not be solved efficiently for large numbers of realizations. Therefore, a more reasonable alternative is to minimize the Frobenius norm $\|\mathbf{I} - \hat{\mathbf{M}}_m^{-1}(\boldsymbol{\xi})\mathbf{A}(\boldsymbol{\xi})\|_F$ [62] over the subspace $\mathcal{Y}_m = \left\{\hat{\mathbf{M}}_m^{-1}(\hat{\boldsymbol{\xi}}_1^{(m)}), \ldots, \hat{\mathbf{M}}_m^{-1}(\hat{\boldsymbol{\xi}}_{P_m}^{(m)})\right\}$ of $\mathbb{R}^{n\times n}$. The solution of this problem is unique and satisfies

$$\mathbf{B}(\boldsymbol{\xi}) \begin{bmatrix} \alpha_1^{(m)}(\boldsymbol{\xi}) \\ \vdots \\ \alpha_{P_m}^{(m)}(\boldsymbol{\xi}) \end{bmatrix} = \begin{bmatrix} \mathrm{tr}\left(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_1^{(m)})\mathbf{A}(\boldsymbol{\xi})\right) \\ \vdots \\ \mathrm{tr}\left(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_{P_m}^{(m)})\mathbf{A}(\boldsymbol{\xi})\right) \end{bmatrix} \tag{4.44}$$

where $\mathbf{B}(\boldsymbol{\xi})$ has components

$$B_{pq}(\boldsymbol{\xi}) = \mathrm{tr}\left(\left(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)})\mathbf{A}(\boldsymbol{\xi})\right)^T \mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_q^{(m)})\mathbf{A}(\boldsymbol{\xi})\right) \quad , \quad (p,q) \in [1, P_m]^2. \tag{4.45}$$

Note however that assembling this P_m-dimensional linear system with matrix-free preconditioners requires to apply each local preconditioner to every column of $\mathbf{A}(\boldsymbol{\xi})$. This cost is considerably reduced when using sketching techniques to compute solutions of a random sub-optimal problem [168]. That is, a random sketching matrix $\boldsymbol{\Theta} \in \mathbb{R}^{n\times K}$ is introduced such that $P_m \leq K \ll n$ and $\mathrm{rank}(\mathbf{A}(\boldsymbol{\xi})\boldsymbol{\Theta}) \geq P_m$. Then, the semi-norm $\|(\mathbf{I} - \hat{\mathbf{M}}_m^{-1}(\boldsymbol{\xi})\mathbf{A}(\boldsymbol{\xi}))\boldsymbol{\Theta}\|_F$ is minimized over $\mathcal{Y}_m$ instead of the exact Frobenius norm. This semi-norm admits a unique minimizer: the solution of the P_m-dimensional linear system given by

$$\mathbf{C}(\boldsymbol{\xi}) \begin{bmatrix} \alpha_1^{(m)}(\boldsymbol{\xi}) \\ \vdots \\ \alpha_{P_m}^{(m)}(\boldsymbol{\xi}) \end{bmatrix} = \begin{bmatrix} \mathrm{tr}\left(\boldsymbol{\Theta}^T\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_1^{(m)})\mathbf{A}(\boldsymbol{\xi})\boldsymbol{\Theta}\right) \\ \vdots \\ \mathrm{tr}\left(\boldsymbol{\Theta}^T\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_{P_m}^{(m)})\mathbf{A}(\boldsymbol{\xi})\boldsymbol{\Theta}\right) \end{bmatrix} \tag{4.46}$$

where $\mathbf{C}(\boldsymbol{\xi})$ has components

$$C_{pq}(\boldsymbol{\xi}) = \mathrm{tr}\left(\left(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)})\mathbf{A}(\boldsymbol{\xi})\boldsymbol{\Theta}\right)^T \mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_q^{(m)})\mathbf{A}(\boldsymbol{\xi})\boldsymbol{\Theta}\right) \quad , \quad (p,q) \in [1, P_m]^2. \tag{4.47}$$

Unlike for Eq. (4.44), the assembly of Eq. (4.46) requires that each local preconditioner is applied to every column of $\mathbf{A}(\xi)\boldsymbol{\Theta} \in \mathbb{R}^{n\times K}$. The minimized semi-norm is a statistical estimator of the optimal Frobenius norm whose error can be controlled with high probability as shown in [168] for different sketching techniques, i.e., ways to generate $\boldsymbol{\Theta}$. However, the resulting interpolating preconditioner is not guaranteed to be SPD, even if all the local preconditioners are SPD. Therefore, Zahm and Nouy [168] also define linear constraints on $\alpha_1^{(m)}(\boldsymbol{\xi}), \ldots, \alpha_{P_m}^{(m)}(\boldsymbol{\xi})$ which guarantee that the constrained minimization of the semi-norm yields an SPD interpolating preconditioner. We introduce two convex

subsets of $\mathcal{Y}_m$ defined by

$$\mathcal{Y}_m^+ = \left\{ \sum_{p=1}^{P_m} \alpha_p^{(m)} \mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)}),\ \alpha_p^{(m)} \geq 0 \right\} \tag{4.48}$$

and

$$\mathcal{Y}_m^{\overline{\text{cond}}} = \left\{ \sum_{p=1}^{P_m} (\alpha_{p,+}^{(m)} - \alpha_{p,-}^{(m)}) \mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)}), \quad \begin{array}{c} \alpha_{p,+}^{(m)} \geq 0,\ \alpha_{p,-}^{(m)} \geq 0,\ p = 1, \ldots, P_m \\ \boldsymbol{\alpha}_+^{(m)T} \boldsymbol{\lambda}_n^{(m)} - \boldsymbol{\alpha}_-^{(m)T} \boldsymbol{\lambda}_1^{(m)} \geq 0 \\ \boldsymbol{\alpha}_+^{(m)T} (\overline{\text{cond}}\ \boldsymbol{\lambda}_n^{(m)} - \mathbf{c}) - \boldsymbol{\alpha}_-^{(m)T} (\overline{\text{cond}}\ \boldsymbol{\lambda}_1^{(m)} + \mathbf{c}) \geq 0 \end{array} \right\} \tag{4.49}$$

in which $\alpha_{p,+}^{(m)} := \max\{0, \alpha_p^{(m)}\}$ and $\alpha_{p,-}^{(m)} := \max\{0, -\alpha_p^{(m)}\}$ with $\boldsymbol{\alpha}_+^{(m)} = [\alpha_{1,+}^{(m)}, \ldots, \alpha_{P_m,+}^{(m)}]$ and $\boldsymbol{\alpha}_-^{(m)} = [\alpha_{1,-}^{(m)}, \ldots, \alpha_{P_m,-}^{(m)}]$. We also have $\boldsymbol{\lambda}_1^{(m)} = [\lambda_1(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_1^{(m)})), \ldots, \lambda_1(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_{P_m}^{(m)}))]$ and $\boldsymbol{\lambda}_n^{(m)} = [\lambda_n(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_1^{(m)})), \ldots, \lambda_n(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_{P_m}^{(m)}))]$ in which $\lambda_1(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)}))$ and $\lambda_n(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)}))$ denote the largest and the smallest eigenvalues of $\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)})$, respectively. The vector $\mathbf{c} = [c_1, \ldots, c_{P_m}]$ has components $c_p = \|\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)})\|$, where $\|\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)})\|$ denotes the operator of $\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)})$. Under the condition $\overline{\text{cond}} \geq \max_{p \in [1,P_m]} c_p / \lambda_n(\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}}_p^{(m)}))$, we have

$$\mathcal{Y}_m^+ \subset \mathcal{Y}_m^{\overline{\text{cond}}} \subset \mathcal{Y}_m. \tag{4.50}$$

Then the minimization problems for the design of $\hat{\mathbf{M}}_m^{-1}(\boldsymbol{\xi})$ are given by

$$\hat{\mathbf{M}}_m^{-1}(\boldsymbol{\xi}) = \underset{\mathbf{M}^{-1} \in \mathcal{Y}_m^+ \cup \mathcal{Y}_m^{\overline{\text{cond}}}}{\text{argmin}} \|\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}(\boldsymbol{\xi})\|_F \tag{4.51}$$

$$\hat{\mathbf{M}}_m^{-1}(\boldsymbol{\xi}) = \underset{\mathbf{M}^{-1} \in \mathcal{Y}_m^+ \cup \mathcal{Y}_m^{\overline{\text{cond}}}}{\text{argmin}} \|(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}(\boldsymbol{\xi}))\boldsymbol{\Theta}\|_F \tag{4.52}$$

which are quadratic optimizations problems with linear inequality constraints. Furthermore, since $\mathbf{M}^{-1}(\hat{\boldsymbol{\xi}_p^{(m)}}) \in \mathcal{Y}_m^+$ for all p, all the resulting projections $\hat{\mathbf{M}}_m^{-1}(\boldsymbol{\xi})$ interpolate $\mathbf{M}^{-1}(\boldsymbol{\xi})$ at the points $\hat{\boldsymbol{\xi}}_1^{(m)}, \ldots, \hat{\boldsymbol{\xi}}_{P_m}^{(m)}$.

Adding preconditioners to a strategy based on a realization-dependent sub-optimal interpolation should result in a decrease of solver iterations. However, the scaling of such a strategy is hindered by the computational cost of setting-up the interpolation, which depends quadratically on the number of preconditioners, and is done for each realization. Another limiting factor is that, for each linear system, there is a number of preconditioners beyond which the application cost of the interpolation outweighs the time saved in spared solver iterations. On the other hand, a preconditioning strategy based on quantization is a more trivial interpolation technique with no significant online set-up cost, requiring no more than a single preconditioner application per interpolation application. In principle, combining quantization with *local interpolation* allows to mobilize a limited number of centroidal preconditioners for each realization, thus decreasing online set-up and application costs of the strategy, while providing potentially better preconditioning than a strategy based only on quantization.

4.7 Numerical experiments

We investigate some of the methods presented in this chapter by means of numerical experiments. In particular, we focus on the two-dimensional isotropic Poisson equation with a random variable coefficient defined on a unit square subjected to homogeneous Dirichlet boundary conditions. The random coefficient is a stationary log-normal process whose underlying Gaussian process has a squared exponential covariance with unit variance and a correlation length of 0.1. That is, we have $T := \log$ and $T^{-1}\kappa(x,\theta) = G(x,\theta)$ where $G : \Omega \times \Theta \to \mathbb{R}$ is a zero-mean Gaussian process with covariance $C(x,x') = \mathbb{E}[G(x,\cdot)G(x',\cdot)] = \exp(-(x-x')^T(x-x')/(0.1)^2)$. We divide the domain $\Omega = [0,1]^2$ with an unstructured mesh of 200,332 triangular elements and 100,652 degrees of freedom. Then, the solution is discretized with P1 finite elements. Along with the application of boundary conditions, the discretization process associates every single realization $\kappa(\cdot,\theta) \in L^2(\Omega)$ of the coefficient field to a corresponding linear system $\mathbf{A}(\theta)\mathbf{u}(\theta) = \mathbf{b}(\theta)$ where $\mathbf{A}(\theta)$ is an n-by-n SPD matrix with n=99,681. The random coefficient field is represented by a KL expansion computed using the same finite element discretization, along with the domain decomposition method described in Chapter 2. Given that $T^{-1}\kappa$ is a Gaussian process, the random variables $\xi_1, \ldots, \xi_{n_{\mathrm{KL}}}$ are iid standard Gaussian random variables. The following numerical experiments are conducted. First, ideal preconditioners are considered, i.e., a preconditioner is built on the basis of every single realization of the coefficient. Different levels of truncation are considered in the approximation $\hat{T}_m^{-1}\kappa$ of the Gaussian process to construct the preconditioner. We illustrate the effect of this truncation on the convergence of PCG. Second, two-dimensional clustering experiments are conducted with different ratios λ_1/λ_2 and maps T_2. We investigate the effect of these parameters on the distribution and shape of the Voronoi cells of the quantizer q_2 as well as on the distribution of attribution frequencies and local distortions. Third, both k-means and CLVQ are compared to compute stationary quantizers of different dimensions. Then, the two preconditioning strategies based on stationary quantizers (see Sections 4.5.1 and and 4.5.2) are applied and their effect on the distribution of average solver iterations per preconditoner, as well as on the number of systems solved per preconditioner and on the cumulated numbers of solver iterations per preconditioner are investigated. Finally, we look at the overall average number of solver iterations for the three different preconditioning strategies presented in Sections 4.4, 4.5.1 and 4.5.2 using different number m of KL modes to approximate $T^{-1}\kappa$.

4.7.1 Effect of relative energy of the approximating coefficient field on theoretically ideal preconditioners

A theoretically ideal preconditioner is based on the trivial quantizer $\tilde{q} : \kappa(\cdot) \mapsto T(\hat{T}_m^{-1}\kappa(\cdot))$. In other words, a preconditioner $\mathbf{M}(T(\hat{T}_m^{-1}\kappa))$ is assembled on the basis of every single realization $\kappa(\cdot)$ of the coefficient field. For the case where $m = n_{\mathrm{KL}}$, no distortion is induced by the quantization. Therefore, if $\mathbf{M}(T(\hat{T}_m^{-1}\kappa))$ is a factorization of $\mathbf{A}(\kappa)$, a single PCG iteration is necessary for the iterative solve of $\mathbf{A}(\kappa)\mathbf{u}(\kappa) = \mathbf{b}(\kappa)$. In practice, this approach makes no sense because it takes approximately as much effort to compute a factorization of $\mathbf{A}(\kappa)$ as it takes to solve $\mathbf{A}(\kappa)\mathbf{u}(\kappa) = \mathbf{b}(\kappa)$ without preconditioner. If, however, we pick $m < n_{\mathrm{KL}}$, the representation error $\epsilon(\hat{T}_m^{-1}\kappa)$ induces a difference between

$\mathbf{A}(T(\hat{T}_m^{-1}\kappa))$ and $\mathbf{A}(\kappa)$. Consequently, the number of PCG iterations should increase with $\epsilon(\hat{T}_m^{-1}\kappa)$. Using such a preconditoning strategy with $m < n_{\mathrm{KL}}$ is, in practice, completely useless since factorizing $\mathbf{A}(T(\hat{T}_m^{-1}\kappa))$ is a priori as difficult as factorizing $\mathbf{A}(\kappa)$. However, for the case in which a non-trivial P-quantizer q of κ with a finite rate P is considered, the dimension m of the underlying quantizer q_2 is an important feature. Indeed, the dimension m of the stochastic space spanned by $\boldsymbol{\xi}$ plays an important role on the level of difficulty to compute compact representations $q_2(\boldsymbol{\xi})$ of $\boldsymbol{\xi}$ with low distortion, and this relative difficulty depends on the rate P of the quantizer. This dependence is indirectly shown by our investigation of the preconditioning strategies for different values of m and P, see Section 4.7.4. Furthermore, irrespective of the type of preconditioner used, the performance of a strategy based on $\tilde{q}$ can serve as a limit of comparison for strategies with less trivial quantizers. Therefore, as a mean to better understand the role of m on a preconditioning strategy, it is important to understand how the average number of solver iterations depends on m as well as on the related error $\epsilon(\hat{T}_m^{-1}\kappa)$ when using an ideal quantizer $\tilde{q}$.

Here, we do not only let $\mathbf{M}(T(\hat{T}_m^{-1}\kappa))$ be a (Cholesky) factorization of $\mathbf{A}(T(\hat{T}_m^{-1}\kappa))$, but also an AMG preconditioner based on $\mathbf{A}(T(\hat{T}_m^{-1}\kappa))$. AMG preconditioners have shown to be particularly efficient when applied to stochastic elliptic PDEs such as the Poisson equation, see [160]. In particular, the AMG preconditioner we use is based on a single V-cycle with smoothed aggregation. In Fig. 4.1, we present an estimate of the expected number of PCG iterations J as a function of the number m of KL modes of $\hat{T}_m^{-1}\kappa$, and of the corresponding relative energy given by $\sum_{k=1}^{m}\lambda_k = 1 - \epsilon(\hat{T}_m^{-1}\kappa) \leq 1$. The estimate of the expected number of solver iterations is based on 100,000 realizations.

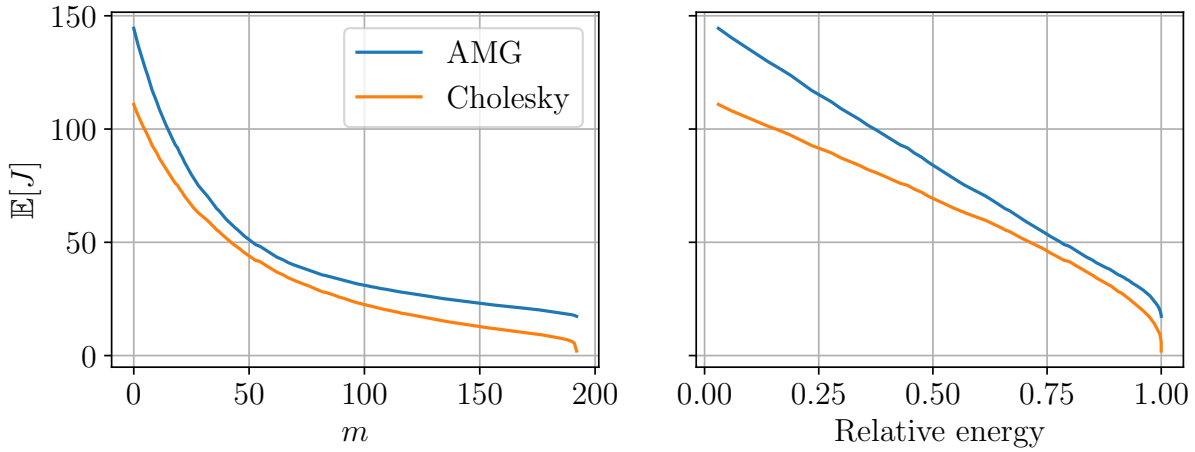

Figure 4.1: Average number of solver iterations J as a function of the number m of KL modes used for $\hat{T}_m^{-1}\kappa$ with a quantizer $\tilde{q} : \kappa(\cdot) \mapsto T(\hat{T}_m^{-1}\kappa(\cdot))$.

In Fig. 4.1, for both the Cholesky factorizations and the AMG preconditioners, we see that the expected number of solver iterations decreases monotonically as m and the relative energy increase. This behavior is not surprising since the difference between $\mathbf{A}(T(\hat{T}_m^{-1}\kappa))$ and $\mathbf{A}(\kappa)$ should increase with m. We can see that the maximum difference of average number of solver iterations between the Cholesky factorizations and the AMG preconditioners occurs at $m = 0$, where it is of the order of 20%. This difference is smaller for all non-null values of m. As such, we can tell that AMG is a very good preconditioner

in that it provides average numbers of solver iterations which are close to what is obtained with Cholesky factorizations. We can see a clear dependence of the difference of average numbers of solver iterations between Cholesky factorizations and AMG preconditioners on the relative energy. As the relative energy approaches 1, i.e., as $\epsilon(\hat{T}_m^{-1}\kappa)$ vanishes, the difference of effect on convergence of the AMG preconditioners compared to the Cholesky factorizations decreases significantly. For most of the range of relative energy, the average number of solver iterations depends almost linearly on the relative energy. Therefore, as a means to characterize the dependence of preconditioning strategies on m, we pick values of m that correspond to evenly spread values of relative energy.

4.7.2 Two-dimensional clustering experiments

Here, we try to illustrate how the properties of the stationary vector quantizers q_2 depend on the choice of map T_2 as well as on the distribution of the eigenvalues $\lambda_1, \ldots, \lambda_m$. Two choices of map T_2 were previously formulated. First, using $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ is such that minimizing the L^2-distortion of $T_2^{-1}(\boldsymbol{\xi})$ induced by $q_2(T_2^{-1}(\boldsymbol{\xi}))$ is equivalent to minimizing the $L^2(\Omega)$-distortion of $\hat{T}_m^{-1}\kappa$ induced by $\hat{P}_m^{\rightarrow}(T_2(q_2(\widetilde{T}_m^{\leftarrow}\kappa)))$. Second, we considered $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$ in an attempt to obtain stationary quantizers q_2 with approximately constant attribution frequencies, i.e., such that $f_1^{(n_s)} \approx \cdots \approx f_P^{(n_s)}$. Both cases are considered in this experiment. The distribution of $\lambda_1, \ldots, \lambda_m$ depends strongly on the covariance function of $G(x, \theta)$, particularly on the correlation length and the roughness in case of Matérn covariance functions. As said earlier, we limit ourselves to square exponential covariances with a single correlation length, but understanding the relation between the properties of the stationary quantizers q_2 and the distribution of $\lambda_1, \ldots, \lambda_m$ can provide useful insights for other covariance functions. For purposes of illustration, we carry experiments with $m = 2$ and different values of the ratio λ_2/λ_1 so as to emulate the effect of the distribution of $\lambda_1, \ldots, \lambda_m$ on the properties of the stationary quantizers q_2.

In Fig. 4.2, we show the normalized quantization error $\|T_2^{-1}(\boldsymbol{\xi}) - q_2(T_2^{-1}(\boldsymbol{\xi}))\|^2/\mathrm{tr}(\boldsymbol{\Lambda})^2$ along with an overlay of the boundaries of the Voronoi cells and centroids of the stationary quantizer q_2 for the first choice of map, i.e., $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$. These results are presented for the quantization rates $P = 10$, 100 and 1,000 with a ratio $\lambda_2/\lambda_1 = 1$, 0.1 and 0.01. For the same quantizers, Fig. 4.3 presents the normalized attribution frequencies $f_1^{(n_s)}, \ldots, f_P^{(n_s)}$. Similarly, Figs. 4.4 and 4.5 present the normalized quantization error and attribution frequencies, respectively, of the stationary quantizers q_2 obtained with the second choice of map, i.e., $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$. In Fig. 4.6, we present scatter plots of the attribution frequencies $f_1^{(n_s)}, \ldots, f_P^{(n_s)}$ with the corresponding norms $\|\hat{\boldsymbol{\xi}}_1\|, \ldots, \|\hat{\boldsymbol{\xi}}_P\|$. These scatter plots are drawn for the same quantization rates P and ratio λ_2/λ_1 as in the previous figures. The deviation of the attribution frequencies $f_1^{(n_s)}, \ldots, f_P^{(n_s)}$ with respect to $1/P$ is drawn as a function of the quantization rate P in Fig. 4.7 for both choices of maps T_2. The evolution of the total distortion $w_2(q_2)$ is also presented as a function of P in Fig. 4.8 for the two choices of map T_2. In all these experiments, the stationary vector quantizers are computed by k-means with a sample size $n_s = 100,000$.

In Figs. 4.2 and 4.3, we can see that the spatial distribution of centroids of the stationary quantizer q_2 strongly depends on the ratio λ_2/λ_1, particularly so for the smaller values of P. For $\lambda_2/\lambda_1 = 1$ and $P = 10$, the centroids of q_2 form a circular cloud with two concentric rings of cells. The Voronoi cells located near the origin are small polygons with

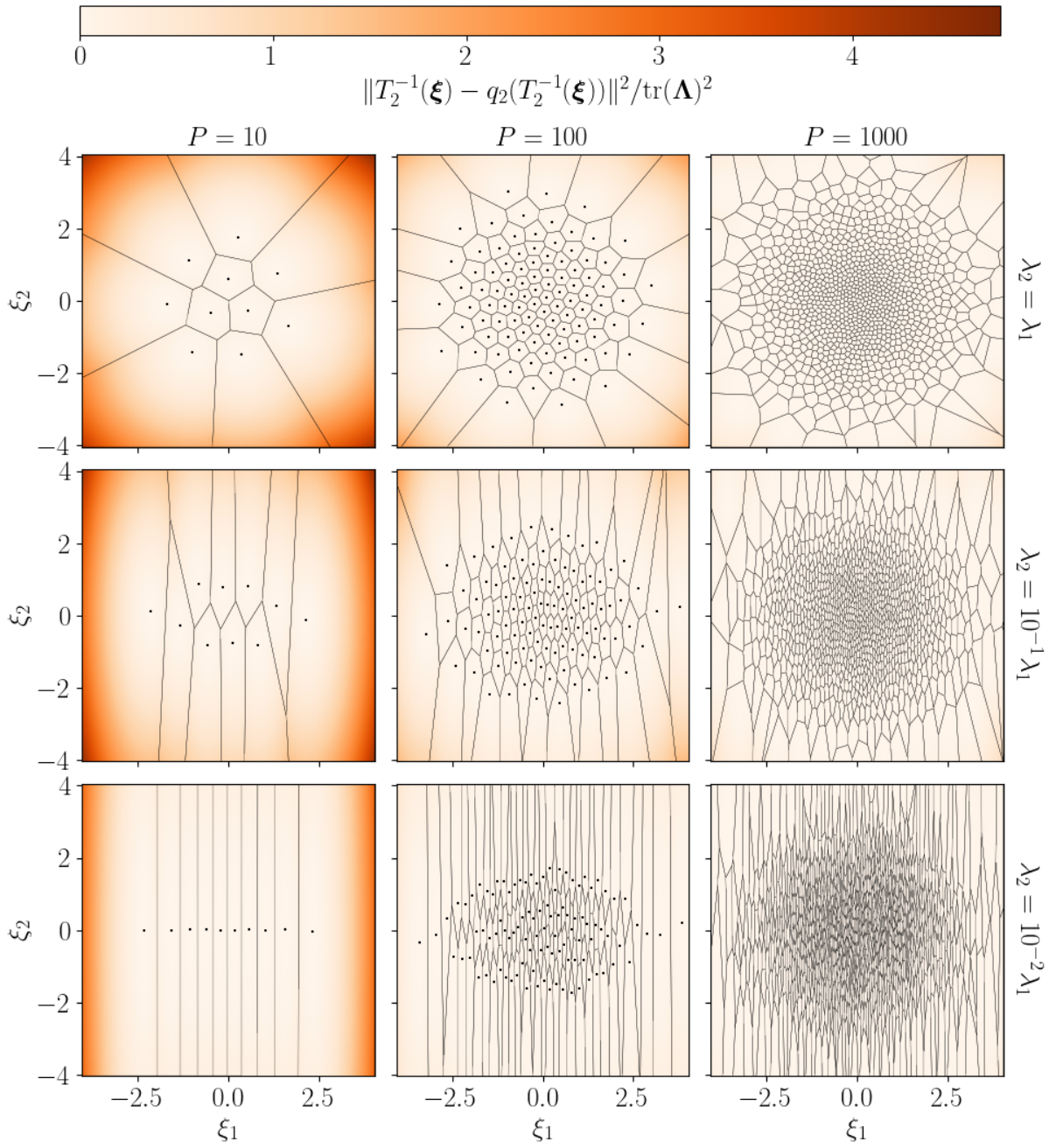

Figure 4.2: Distortion field $\|T_2^{-1}(\boldsymbol{\xi}) - q_2(T_2^{-1}(\boldsymbol{\xi}))\|^2$ of the quantizer q_2 obtained by k-means with the map $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2)$.

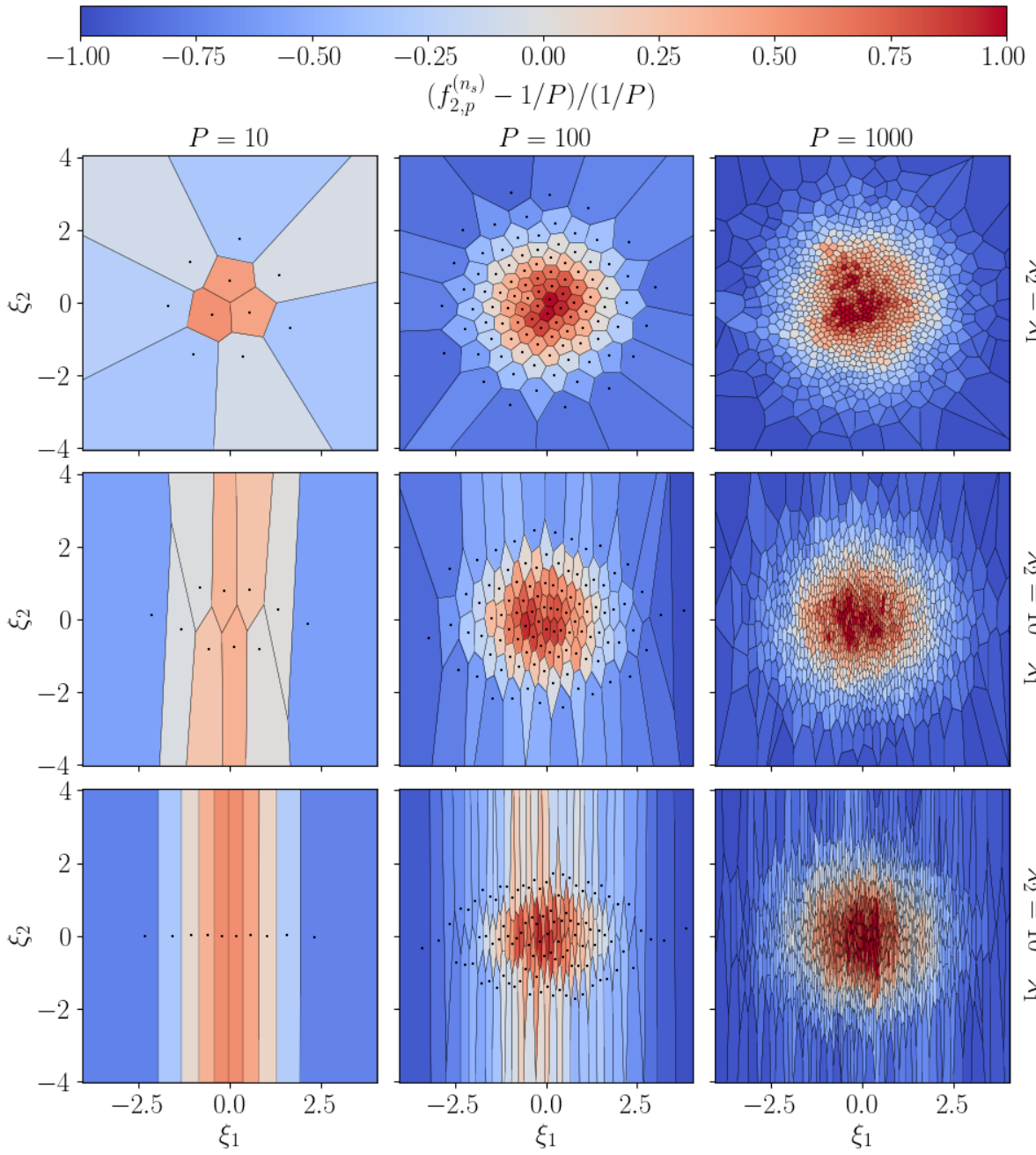

Figure 4.3: Attribution frequencies $f_{2,p}^{(n_s)}$ of the quantizer q_2 obtained by k-means with the map $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2)$.

aspect ratios close to 1. The Voronoi cells located further from the origin are much larger and elongated as they indefinitely extend away from the origin. For the larger values of P, the circular shape of the cloud of centroids is preserved. Similarly as for $P = 10$, there are concentric rings of Voronoi cells, each with somewhat constant cell sizes. The size of these cells progressively increases as the distance of the centroid from the origin increases. For $\lambda_2/\lambda_1 = 0.1$ and $P = 10$, the centroids of q_2 form a nearly ellipsoidal cloud with the major axis along ξ_1. The two Voronoi cells whose centroids are at the extremities of the major axis extend indefinitely along ξ_1 away from the origin and in both the positive and negative directions of ξ_2. All the other Voronoi cells whose centroids are closer to the origin are bounded on both sides along ξ_2 as well as near the center of ξ_1 while they extend indefinitely along ξ_1 away from the origin. For larger values of P, the centroids form clouds which are less and less elongated as P increases. All the Voronoi cells are elongated along ξ_2, and increasingly so as the centroid is further from the origin. For $\lambda_2/\lambda_1 = 0.01$ and $P = 10$, the centroids of q_2 form a line along ξ_1. The two Voronoi cells whose centroids are at the edges of the line extend indefinitely along ξ_1 away from the origin and in both the positive and negative directions along ξ_2. All the other Voronoi cells whose centroids are closer to the origin are bounded on both sides along ξ_2 while they extend indefinitely along both the positive and negative directions of ξ_1. Similarly as before, for larger values of P, the centroids form clouds which are less and less elongated as P increases. Again, all the Voronoi cells are increasingly elongated along ξ_2 as the centroids are further from the origin. Irrespective of the values of P and λ_2/λ_1, the Voronoi cells whose centroid is near the origin have attribution frequencies which are significantly greater than $1/P$. These attribution frequencies keep decreasing as the distance of the centroid from the origin increases. All the Voronoi cells whose centroids are the most distant from the origin have attribution frequencies which are significantly smaller than $1/P$.

In Figs. 4.4 and 4.5, the spatial distribution of centroids of the stationary quantizer q_2 also depends on the ratio λ_2/λ_1 for $P = 10$. For $\lambda_2/\lambda_1 = 1$ and $P = 10$, the centroids of q_2 form a square cloud made of three parallel horizontal lines. A Voronoi cell has its centroid located almost exactly at the origin. This cell is small with an aspect ratio close to 1. There are narrow Voronoi cells which start at each of the edges of the central cell and extend indefenitely away from those edges. The tessellation is completed by four large square cells at the corners of the square cloud of centroids. For larger values of P, the shape of the centered square cloud of centroids is preserved. Similarly as for $P = 10$, there are four large square cells at the corners of the square cloud of centroids. In the center of the cloud, there are multiple small Voronoi cells with aspect ratios close to 1. All along the edges of the cloud of centroids and between the four large square cells, there are multiple narrow cells which extend indefinitely away from the edges of the square cloud of centroids. For $\lambda_2/\lambda_1 = 0.1$ and $P = 10$, the centroids of q_2 form a rectangular cloud made of two parallel horizontal lines. Once again, there are four large Voronoi cells at the corners of the cloud of centroids. All the other cells are narrowly trapped between the large square cells, they span indefinitely away from the ξ_1 axis. For larger values of P, the centroids still form a rectangular cloud. The center of the cloud of centroids is also packed with very small cells with aspect ratios close to 1. All along the edges of the rectangular cloud of cells, there are narrow cells which indefinitely away from the edges. There are more of these cells on the top and bottom edges than there are on the left and

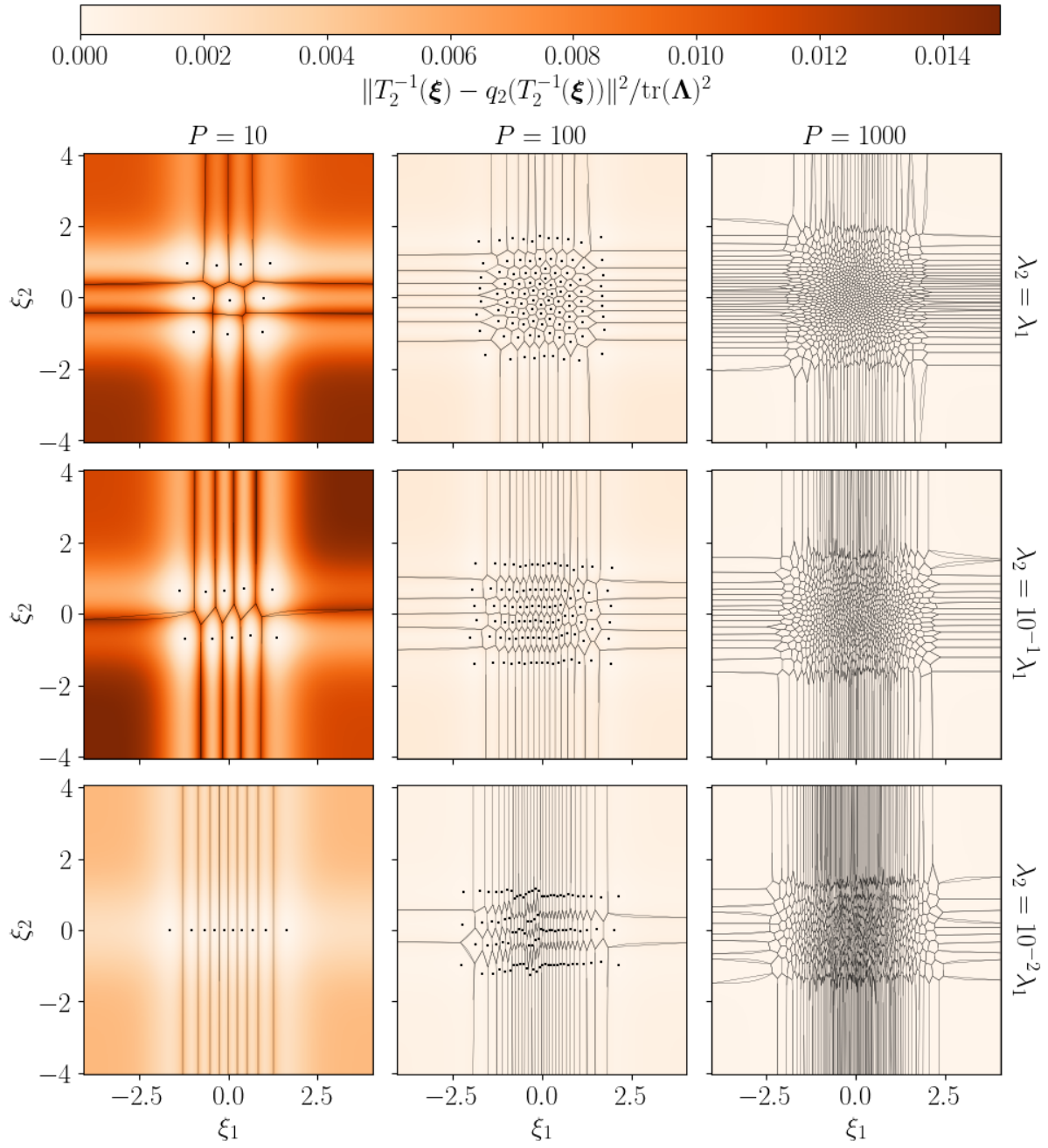

Figure 4.4: Distortion field $\|T_2^{-1}(\boldsymbol{\xi}) - q_2(T_2^{-1}(\boldsymbol{\xi}))\|^2$ of the quantizer q_2 obtained by k-means with the map $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2} F_\xi \circ \boldsymbol{\xi}$ where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2)$.

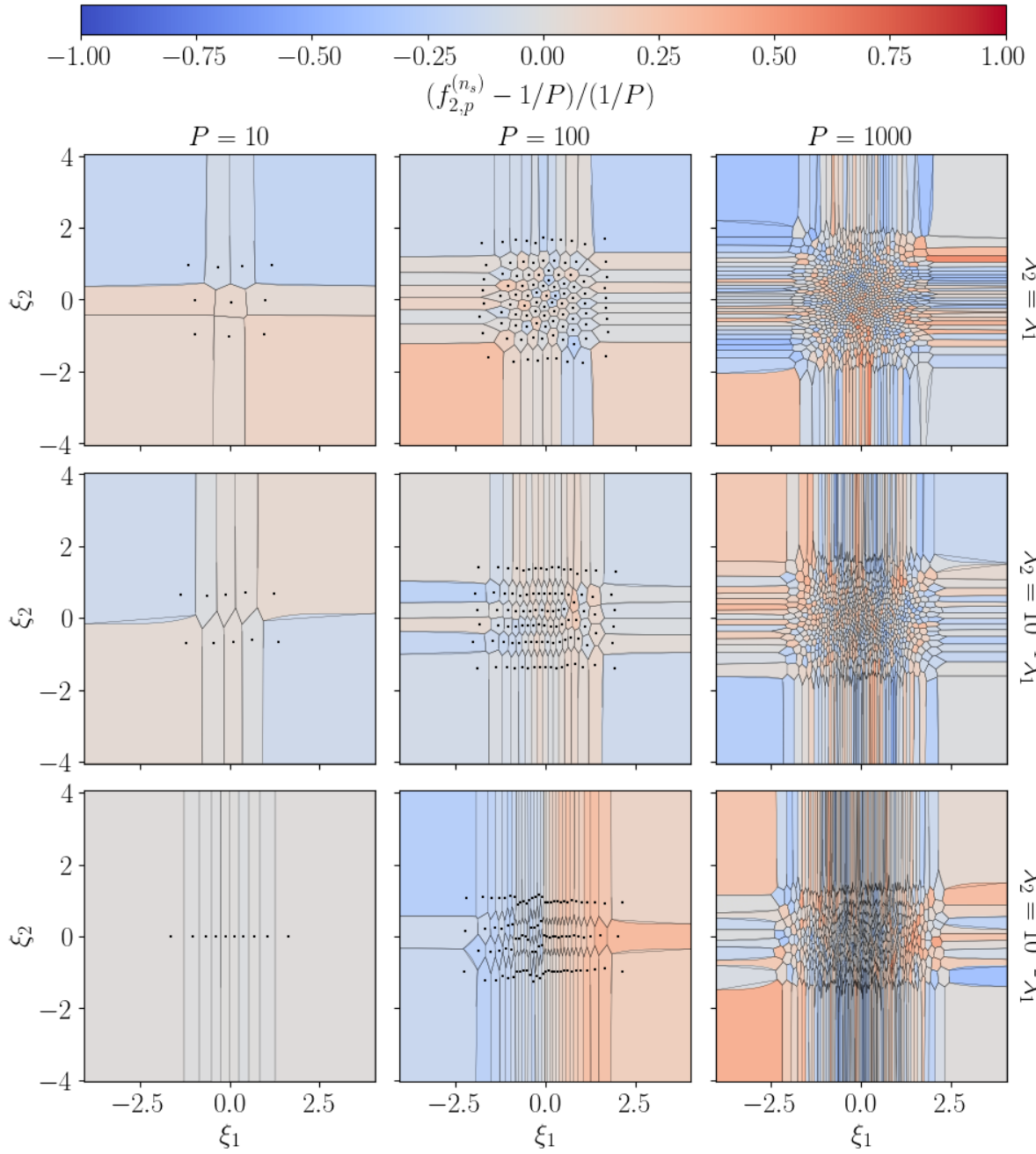

Figure 4.5: Attribution frequencies $f_{2,p}^{(n_s)}$ of the quantizer q_2 obtained by k-means with the map $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2} F_\xi \circ \boldsymbol{\xi}$ where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_2)$.

right edges. For $\lambda_2/\lambda_1 = 0.01$ and $P = 10$, the centroids of q_2 form a line along ξ_1. The two Voronoi cells whose centroids are at the edges of the line extend indefinitely along ξ_1 away from the origin and in both the positive and negative directions along ξ_2. All the other Voronoi cells whose centroids are closer to the origin are bounded on both sides along ξ_2 while they extend indefinitely along both the positive and negative directions of ξ_1. For larger values of P, the centroids form a rectangular cloud with the longer edges along ξ_1. The center of the cloud of centroids is packed with very small cells. Again, all along the edges of the rectangular cloud of centroids, there are narrow cells which span indefinitely away from the edges. There are significantly more of these cells on the top and bottom edges than there are on the left and right edges. Irrespective of the values of P and λ_2/λ_1, all the Voronoi cells have attribution frequencies close to $1/P$. There is no discernible spatial pattern of how the attribution frequencies deviate from $1/P$.

In Fig. 4.6, we can clearly see how the relation between the attribution frequency $f_p^{(n_s)}$ and the distance $\|\hat{\boldsymbol{\xi}}_p\|$ of the centroid from the origin depends on the choice of map T_2. For the first case, i.e., $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$, the attribution frequency $f_p^{(n_s)}$ seems to depend on the distance $\|\hat{\boldsymbol{\xi}}_p\|$ after a half bell curve. This is more evident for $P = 100$. The spread around this curve seems to increase with P. For the second case, i.e., $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$, we can see that the attribution frequency $f_p^{(n_s)}$ scatters around $1/P$ irrespective of the distance $\|\hat{\boldsymbol{\xi}}_p\|$.

In Fig. 4.7, it is clear that the deviation of the attribution frequencies $f_1^{(n_s)}, \ldots, f_P^{(n_s)}$ is more significant for $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ than for $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$. This deviation decreases as the quantization rate P is increased.

In Fig. 4.8, we can see that picking $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$ yields a smaller distortion $w_2(q_2)$ than using $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$. As expected, the distortion decreases as we increase the quantization rate P.

4.7.3 Computation of stationary vector quantizers with k-means and CLVQ

Here, we wish to compare the use of k-means and CLVQ for the computation of stationary quantizers q_2 for both choices of the map T_2. These quantizers are computed for different numbers m of KL modes and quantization rate P. For each pair (m, P), 100 clustering experiments are run using both k-means and CLVQ. In Fig. 4.9, we report the average distortion $w_2^{(n_s)}(q_2)$ and the ratio of computing time of CLVQ over k-means for $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$. Equivalent results are presented in Fig. 4.10 for $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$. Each of these experiments is performed with a sample size $n_s = 10,000$. We can say that the results do not depend on the choice of T_2. Almost the same level of average distortion is achieved by each stationary quantizer irrespectively of whether it is computed with k-means or CLVQ. The average distortion decreases as a function of P and increases as a function of m. The relative runtime of CLVQ compared to k-means strongly depends on P. For $P = 10$, CLVQ is faster than k-means for all values of m. For $P = 100$, CLVQ is faster for small values of m, and slower for larger values of m. For $P = 1,000$, k-means is faster for all values of m with a speedup which increases from 4x to more than 11x for increasing values of m.

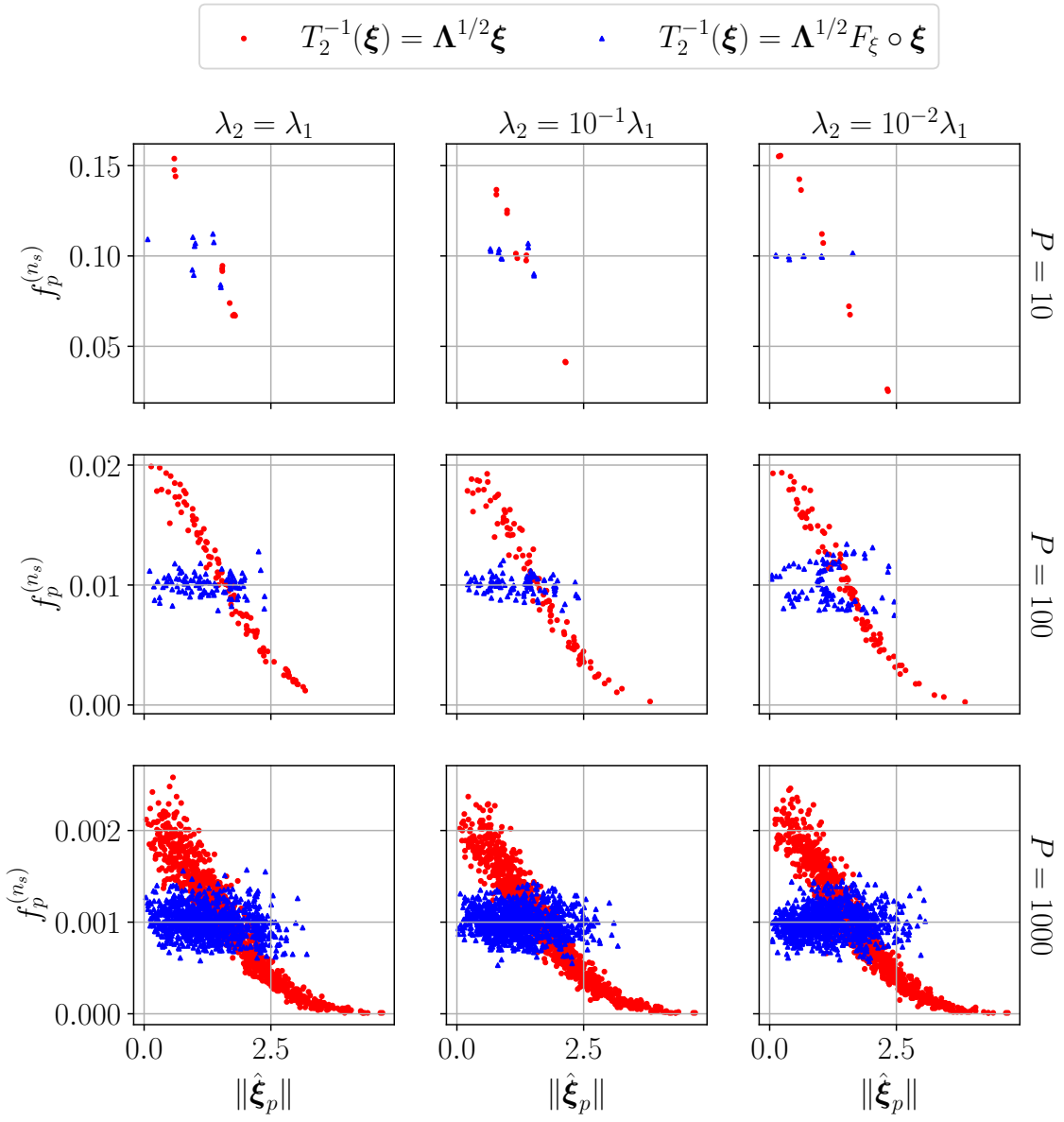

Figure 4.6: Scatter plots of attribution frequencies $\{f_p^{(n_s)}\}_{p=1}^P$ and Euclidean distances from the corresponding centers $\{\hat{\boldsymbol{\xi}}_p\}_{p=1}^P$ to the origin.

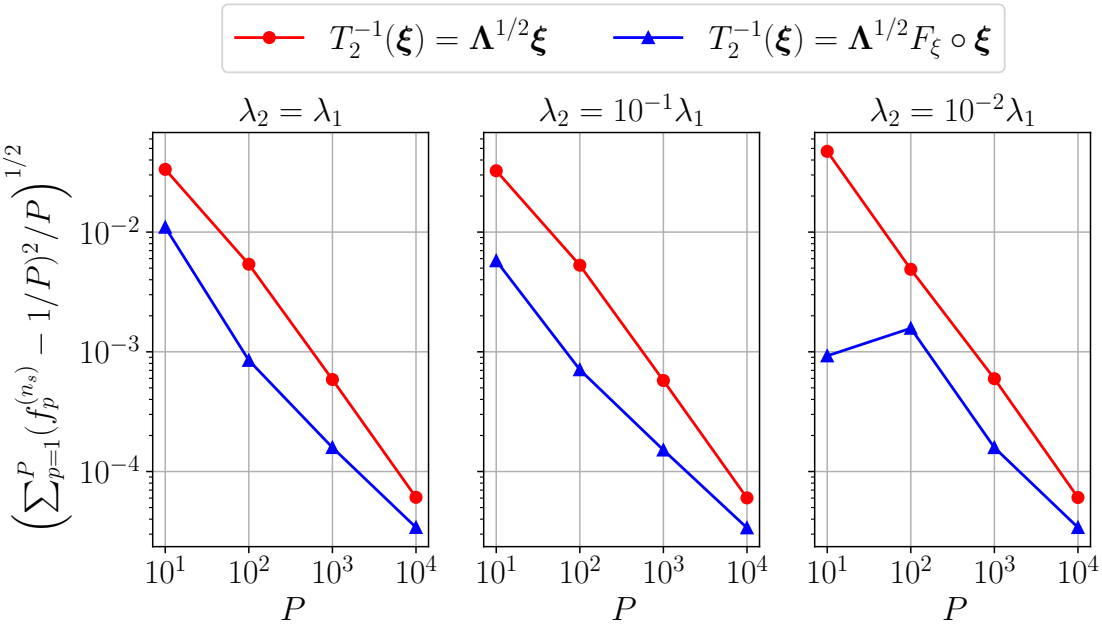

Figure 4.7: Standard deviation of attribution frequency $f_p^{(n_s)}$ for different numbers of preconditioners.

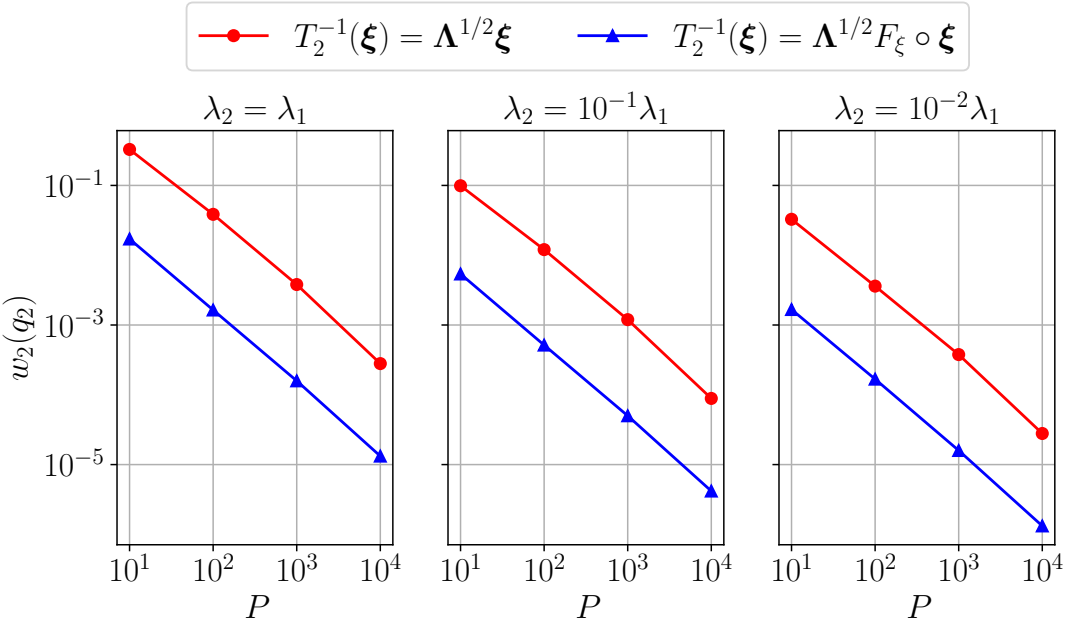

Figure 4.8: Distortion of the quantizer q_2 for different numbers of preconditioners.

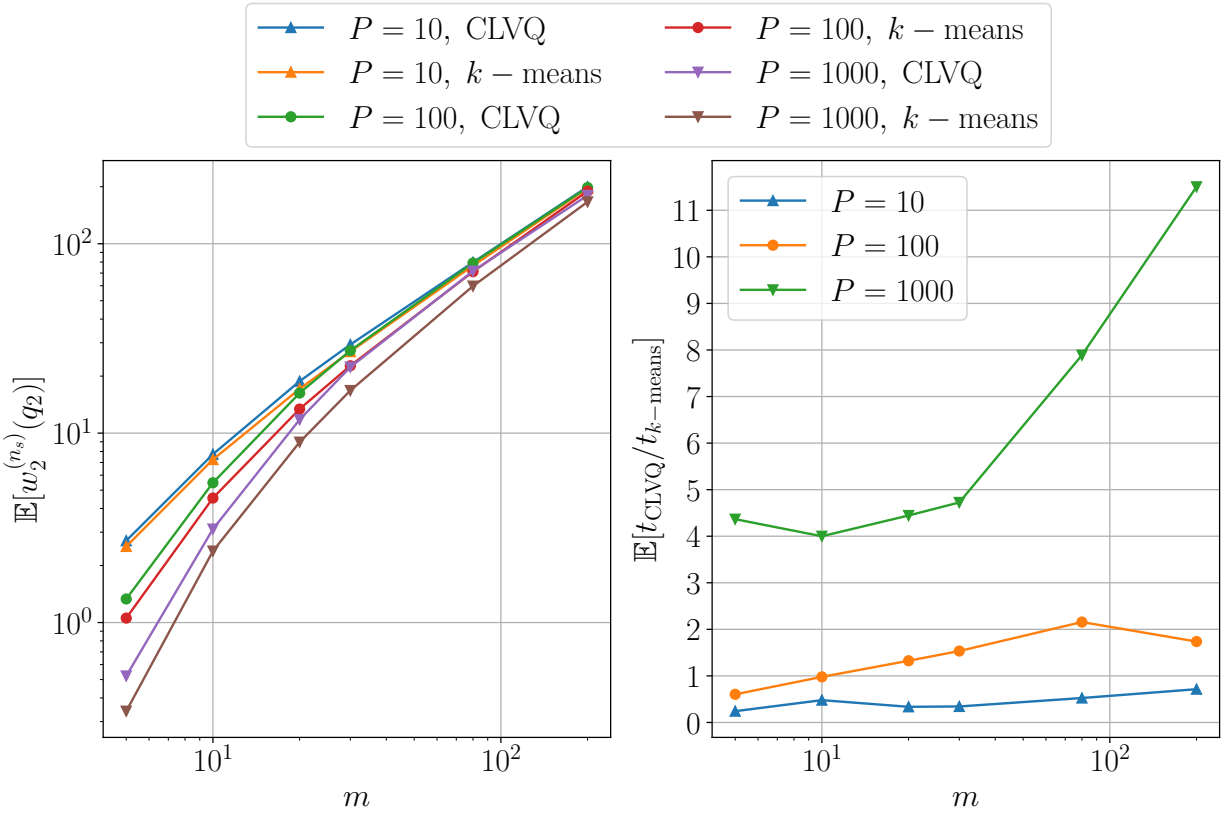

Figure 4.9: Clustering results for CLVQ and k-means with $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ and $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_m)$.

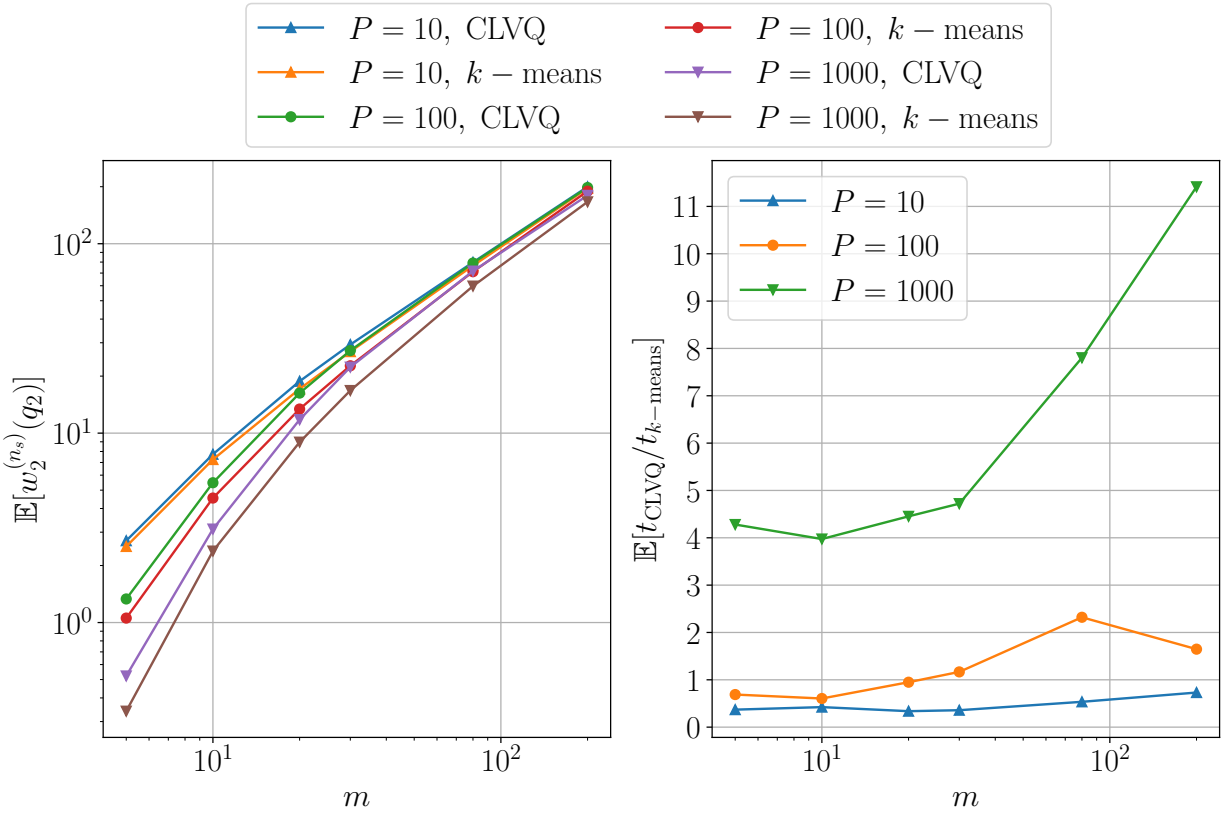

Figure 4.10: Clustering results for CLVQ and k-means with $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2} F_\xi \circ \boldsymbol{\xi}$ and $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_m)$.

4.7.4 Performance of the preconditioning strategies

Perhaps the most important experiment of this chapter is the investigation of the effect of the preconditioning strategies on the convergence of PCG. In Fig. 4.11, we consider the preconditioning strategies based on stationary quantizers q_2 for the two possible choices of T_2 and we present the effect on (i) the expected number of solver iterations per preconditioner as well as on (ii) the number of linear systems solved per precondtioner, and (iii) the cumulated number of solver iterations per precondtioner. These quantities are presented in scatter plots as functions of the distance $\|\hat{\boldsymbol{\xi}}_p\|$ of the centroid $\hat{\boldsymbol{\xi}}_p$ to the origin. The results presented are obtained with an approximating coefficient field composed of $m = 8$ KL modes for a quantization with $P = 1,000$ preconditioners. The preconditioners used are AMG, and the MC simulation consists of 100,000 realizations. First, we can say that, irrespective of T_2, there is some level of linear correlation between $\mathbb{E}_p[J]$ and $\|\hat{\boldsymbol{\xi}}_p\|$. It is actually surprising that these quantities are not more correlated. Indeed, our experiments have shown that the conditioning of a Galerkin operator $\mathbf{A}(\kappa(\hat{\boldsymbol{\xi}}_p))$ tends to increase with the norm $\|\hat{\boldsymbol{\xi}}_p\|$ of the latent variable $\hat{\boldsymbol{\xi}}_p$ of the underlying coefficient field $\kappa(\hat{\boldsymbol{\xi}}_p)$. Second, we can see that, when using $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$, the number n_p of linear systems solved with a preconditioner $\hat{\mathbf{M}}_p$ is rather strongly correlated with the size of the norm $\|\hat{\boldsymbol{\xi}}_p\|$. This result is not surprising when we consider the dependence of the attribution frequency $f_{2,p}^{(n_s)}$ on the distance $\|\hat{\boldsymbol{\xi}}_p\|$ presented in Section 4.7.2 for $m = 2$. In particular, the number n_p of linear systems solved decreases with $\|\hat{\boldsymbol{\xi}}_p\|$. We can see that choosing $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ leads to a larger spread of the possible values taken by n_p than when using $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$. In the case $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$, the number n_p of linear systems solved per preconditioner does not strongly correlate with $\|\hat{\boldsymbol{\xi}}_p\|$. Third, the scatter plot of the cumulated number of solver iterations as a function of the norm $\|\hat{\boldsymbol{\xi}}_p\|$ strongly resembles the scatter plot of n_p. Hence, the number of cumulated solver iterations strongly correlates with $\|\hat{\boldsymbol{\xi}}_p\|$ when using $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$. On the other hand, using $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$ leads to less spread of the number of cumulated solver iterations. In other words, the computational load is more balanced in the latter case. Although the figure is not presented here, similar results as those of Fig. 4.11 were obtained using Cholesky factorizations in place of AMG preconditioners. The results are almost identically the same, which is why we do not present them here, with the exception that all the average and cumulated numbers of solver iterations are slightly smaller.

In Fig. 4.12, we present the overall average number of solver iterations for each of the three preconditioning strategies presented in this chapter. That is, the two strategies based on stationary quantizers q_2 built with different choices of T_2, but also the strategy built with deterministic grids (see Section 4.4). In the case of the stationary quantizers, the results are presented for different numbers m of KL modes for the approximating coefficient field. As explained in Section 4.7.1, these values of m are selected so as to correspond to relative energies which properly cover the whole range from 0 to 1. In particular, we use $m = 8$ which corresponds to approximately 20% of relative energy as well as $m = 24$ (50%), $m = 48$ (75%) and $m = 170$ (99%). Different numbers P of preconditioners are considered, namely P =1, 10, 100, 1,000 and 10,000. When it comes to the preconditioning strategy based on deterministic grids, all the possible numbers of preconditioners from $P = 1$ to $P = 8,193$ are considered, i.e., we used $P = 1 + 2^m$

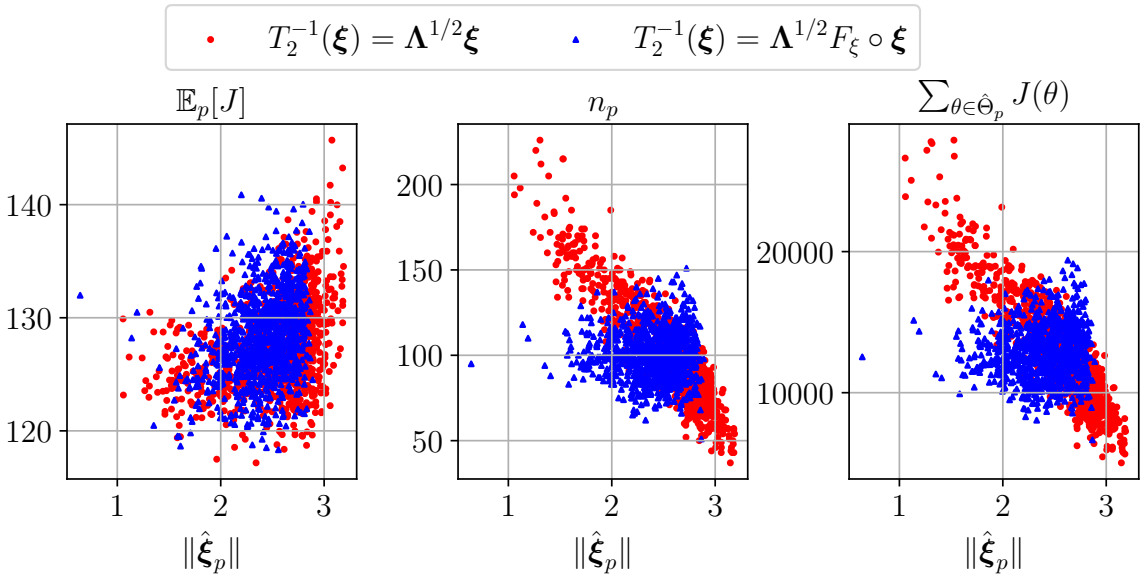

Figure 4.11: Average numbers of solver iterations, numbers of linear systems solved and total numbers of solver iterations per preconditioner. Number of KL modes $m = 8$. Number of preconditioners $P = 1,000$. Results obtained with AMG precondtioners.

for $m = 1, \ldots, 13$ and $P = 1$ for $m = 0$. For the preconditioning strategies based on stationary quantizers, the relation between $\mathbb{E}[J]$ and the number P of preconditioners is fairly similar irrespective of T_2, with the exception that for a given pair (m, P), using $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$ yields slightly larger values of $\mathbb{E}[J]$ than using $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$. Besides this difference, the same phenomenon can be observed, irrespective of T_2. That is, for $m = 8$, $\mathbb{E}[J]$ decreases as a function of P, at a decreasing rate as $\mathbb{E}[J]$ starts to stagnate for larger values of P. Using $m = 8$ seems optimal for the smallest number of preconditioners considered, that is $P = 10$, i.e. a constant zero coefficient field is used for $P = 1$. Similarly, for $m = 24$, $\mathbb{E}[J]$ decreases as a function of P, also at a decreasing rate, but stagnation starts occurring for larger values of P. Meanwhile, selecting $m = 24$ seems optimal when using $P = 100$ preconditioners. The same type of behavior is observed for $m = 48$, with the difference that $\mathbb{E}[J]$ is larger than when using smaller values of m for the smaller values P. Eventually, a stagnation also starts occuring, but for larger values of P. Using $m = 48$ is optimal when using $P = 100$ preconditioners. Finally, for $m = 170$, the stagnation is not captured for our investigated numbers P of preconditioners. Using $m = 170$ however does not as strongly impact $\mathbb{E}[J]$ as using smaller values of m in the case of $P = 10$, 100 and 1,000. The choice $m = 170$ is optimal for $P = 10,000$ preconditioners. A major difference of the preconditioning strategy based on deterministic grids is that the number m of approximating KL modes depends on P. A consequence of this scheme on the evolution of the average number of solver iterations is that the decrease of $\mathbb{E}[J]$ does not stagnate as P is increased, unlike the case of the preconditioning strategies based on stationary quantizers with a fixed value of m. Meanwhile, although the preconditioning strategy based on deterministic grids does not necessarily yield values of $\mathbb{E}[J]$ as low as using a stationary quantizers with $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ with an optimally selected value of m, it does reasonably well for the entire range of values for the number P of preconditioners

considered in this study.

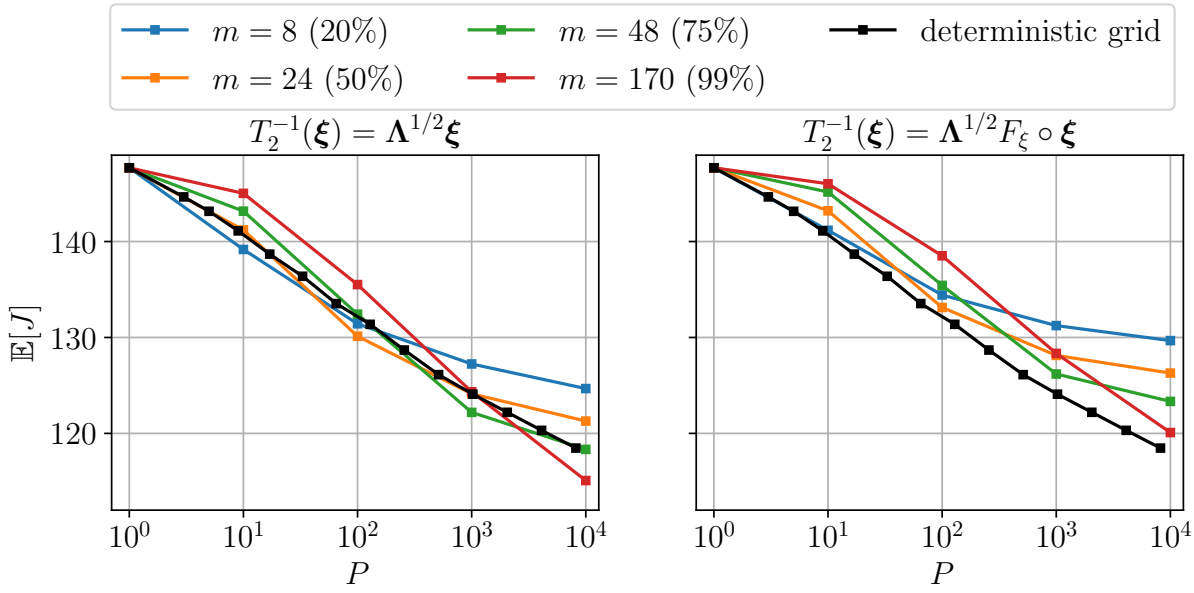

Figure 4.12: Average numbers of PCG iterations for different quantizations of preconditioners.

4.7.5 Effect of local interpolation

The last experiment conducted in this chapter consists of building local interpolations of multiple preconditioners instead of applying a single preconditioner $\hat{\mathbf{M}}_p$ to solve a sampled linear system. As a means to build such local interpolations, groups of centroids must be formed. We do so using the k-means algorithm to compute clusters of close centroids for every given quantization. For a given number P of preconditioners, we need to pick a number M of clusters. We do so so as to have an average cluster size of 4. That is, for $P = 100$, 1,000 and 10,000, we use $M = 25$, 250 and 2,500, respectively. We develop local interpolators for the preconditioning strategies based on stationary quantizers for both choices of T_2. As described in Section 4.6, we compute the coefficients of the local interpolation using the randomly sketched matrix $\mathbf{A\Theta}$. Similarly as in [168], we use matrix-free sub-sampled randomized Hadamard transforms, with $K = 1,000$. The experiment is conducted with AMG preconditioners as well as with Cholesky factorizations. For the totality of the linear systems solved, using a local interpolator shows absolutely no impact whatsoever on the number of solver iterations compared to using a single preconditioner $\hat{M}_p$. In addition to the interpolation presented in Section 4.6, another interpolation method described in the work of Zahm and Nouy [168] is used, namely the Shepard interpolation. Similarly, absolutely no impact is observed on the convergence of PCG.

4.8 Conclusion

In this chapter, we developed preconditioning strategies based on the quantization of coefficient fields for the iterative solve of linear systems which arise from the discretization

of sampled stochastic elliptic PDEs with random variable coefficients.

The problem of functional quantization of coefficient fields was transformed to a problem of vector quantization in the stochastic space induced by the KL expansion of a mapping $T^{-1}\kappa$ of the coefficient field. From thereon, an additional map T_2 was introduced so as to allow some flexibility in the design of the vector quantizer q_2 which can be related to a quantizer of the coefficient field through Eq. (4.18). In particular, we considered Voronoi stationary quantizers which minimize the distortion induced by the L^2 norm of the random vector $T_2^{-1}(\boldsymbol{\xi})$. Two methods were considered to compute these stationary quantizers, namely k-means and CLVQ. In [117], it is argued that k-means is untractable in multiple dimensions, i.e., when m becomes large. Hence, we compared the use of CLVQ to the use of k-means to compute stationary vector quantizers of $T_2^{-1}(\boldsymbol{\xi})$ where $\boldsymbol{\xi}$ is an m-dimensional standard Gaussian random vector with covariance $\mathbf{I}_m$. Experiments where conducted with both $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ and $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$ where $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_m)$ in which $\lambda_1, \ldots, \lambda_m$ are the dominant eigenvalues of the KL expansion of a stationary Gaussian process with unit variance and squared exponential covariance. Contrarily to what is claimed in [117], we had no difficulty computing stationary quantizers with comparable levels of distortion using both methods with values of m going up to 200 for numbers of clusters spanning from $P = 10$ to $P = 10,000$ with sample sizes of 100,000. When it comes to running times, CLVQ was faster for $P = 10$ while k-means was faster for $P = 1,000$ and $10,000$. Note that the clustering of the latent random vectors to compute stationary quantizers is only a pre-processing phase of the MC simulation which, for sufficiently large numbers of realizations, only represents a negligible fraction of the total computing effort. The great majority of the computational load is the linear solve of the multiple linear systems corresponding to all the sampled realizations of the coefficient field.

We investigated the effect of the choice of the map T_2 on the properties of the stationary quantizer q_2. It was shown that choosing $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ leads to ellipsoidal clouds of centroids with aspect ratios depending on the distribution of $\lambda_1, \ldots, \lambda_m$. The distribution of eigenvalues also impacts the shape of the Voronoi cells of the quantizer, whose sizes increase with the distance from the centroid of the cell to the origin. The attribution frequency $f_{2,p}^{(n_s)}$ also depends on the norm $\|\hat{\boldsymbol{\xi}}_p\|$ of the cell centroid in a way somewhat similar to a half bell curve. Meanwhile, using $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$ leads to rectangular clouds of centroids with aspect ratios which depend on the distribution of $\lambda_1, \ldots, \lambda_m$. The distribution of eigenvalues does not significantly impact the shape of the Voronoi cells within the rectangular cloud, which all are very small. Systematically, some square Voronoi cells occur at the corners of the cloud of centroids, between which are packed elongated cells. The attribution frequency $f_{2,p}^{(n_s)}$ only slightly varies around the value $1/P$ without following any specific pattern with respect to the location of the centroid. The distribution of the attribution frequencies can be interpreted in terms of its effect on the underlying preconditioning strategy. For, $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$, the somewhat uniform distribution of $f_{2,1}^{(n_s)}, \ldots, f_{2,P}^{(n_s)}$ with low spread around $1/P$ means that the underlying strategy is such that every preconditioner is used for approximately as many linear systems. On the other hand, the dependence of the attribution frequency on the norm $\|\hat{\boldsymbol{\xi}}_p\|$ which we observe when using $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ is such that the preconditioners which correspond to smaller values of $\|\hat{\boldsymbol{\xi}}_p\|$ are used to solve more linear systems. Note that, typically, the conditioning number of a Galerkin operator increases with the norm of

the latent vector $[\xi_1, \ldots, \xi_{n_{\mathrm{KL}}}]$ which defines the realization of the underlying field $T^{-1}\kappa$.

In this work, we considered linear systems of dimension $n = 99,681$. Consequently, we were not able to simultaneously store all the preconditioners in memory for some values of P, i.e., 1,000 and 10,000. In practice, even larger linear systems may have to be solved, in which case only one or a couple of preconditioners may be stored in memory at once. Then, two possible ways to carry an MC simulation arise. First, a sequential simulation relies on a preliminary step which consists of sampling all the latent random vectors of the simulation and attributing them to the corresponding centroids. These realizations are stored on disk. Then, we start by loading the first preconditioner in memory as well as all the corresponding realizations. While the preconditioner is stored in memory, we assemble the linear system of each realization and solve each linear system. Once all the corresponding linear systems have been solved, it is time to load the second preconditioner in memory along with its corresponding realization, and so on. The second approach is parallel. Typically, one has access to a distributed memory system. Then, one can identify a main node which is used to sample realizations of the coefficient field which are sent to the node with the corresponding preconditioner. In order to account for the total running time of a sequential simulation, one can simply look at the expected number of solver iterations of a given preconditioning strategy. Meanwhile, the total time of a parallel simulation is rather accounted by the maximum time taken by any of the nodes to solve all its attributed linear systems. Therefore, in the case of parallel simulations, we are more interested in the distribution of cumulated solver iterations among preconditioners for a given simulation.

For the case of parallel simulations, the preconditioning strategies based on stationary quantizers were considered for both choices of the map T_2. It was observed that choosing $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ leads to a larger spread of the number of cumulated solver iterations among the different preconditioners. Although the minimum number of cumulated solver iterations for a preconditioner is smaller with this choice of T_2 than when choosing $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$, so is the maximum number of cumulated solver iterations. Hence, we can say that $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$ leads to more balanced numbers of cumulated solver iterations and should thus be preferred for parallel simulations.

For the case of sequential simulations, the performance of a given preconditioning strategy can be evaluated in terms of the expected number of solver iterations. Once again, preconditioning strategies were considered which are based on stationary quantizers based on the two possible choices of the map T_2. It was observed that choosing $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ leads to smaller average numbers of solver iterations and should thus be preferred for sequential simulations. However, the optimal choice of the number m of KL modes of the approximating coefficient field was shown to depend on the number P of preconditioners. That is, for smaller values of P, it is preferable to chose a dimension m of the approximating stochastic space which is rather small. However, as we increase the value of P, the same choice m is such that the stochastic space starts to saturate with centroids and the effect on the convergence of PCG starts to be less significant, in which case it is advantageous to increase the dimension m. Hence, in order to chose the optimal number m for given number P of preconditioners, one may have to run a preliminary study in which the relative performance of different values of m is investigated. A third preconditioning strategy based on deterministic grids was shown to provide a reasonable alternative to the strategies based on stationary quantizers. Indeed, the increase of the

dimension m of the approximating stochastic space of the deterministic grid as a function of the number P of preconditioners is such that the strategy based on deterministic grids never seems to saturate as opposed to the strategies based on stationary quantizers with a fixed value of m. Meanwhile, the average numbers of solver iterations achieved by the strategy based on deterministic grids are also reasonably small.

Local interpolations of preconditioners were introduced as an attempt to leverage the fact that, even when using distributed memory systems, each node is likely able to store a small number of preconditioners greater than one. Therefore, local interpolations based on the optimal linear span of preconditioners from small clusters of neighboring centroids were introduced in a similar way to what was proposed in [168]. This strategy was however shown to not bring any amelioration to our different strategies based on quantizers of coefficient fields.

Chapter 5

Conclusion and perspectives

This thesis was aimed at developing preconditioning strategies for the iterative solve of linear systems with multiple operators and right-hand sides which arise when discretizing stochastic ellitpic PDEs while sampling random variable coefficients by Monte Carlo methods. Here, we summarize our contributions as well as our conclusions, and we list some ideas of future works.

5.1 Summary of contributions

The following contributions were made:

- Parallel Julia implementation of the DD-KL method presented in Section 1.2.2 using P1 finite elements with unstructured triangular meshes in 2D. This implementation is parallelized with Distributed.jl. The source code can be found in the following files:
  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/Fem/KarhunenLoeveDomainDecomposition.jl`
  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/Fem/KarhunenLoeveDomainDecompositionHelper.jl`
  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/Fem/KarhunenLoevePllDomainDecomposition.jl`
- Julia implementation of preconditioners based on non-overlapping domain decomposition. The source code can be found in the following file:
  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/Fem/EllipticPdeDomainDecomposition.jl`

- Application of two limit preconditioning strategies. A first one which consists of using a constant preconditioner based on a central representative coefficient field, and a second strategy which consists of redefining the preconditioner for every sampled linear system on the basis of the Galerkin operator of the coefficient field realization. The use of these two basic strategies allowed to draw the upper and lower limits of the PCG convergence behavior within which the other preconditioning strategies lie.

- Application and Julia implementation of deflation with approximate eigenvectors for the linear solve of sequences of linear systems with varying operators and right-hand sides which arise from the discretization of stochastic elliptic PDEs when the random variable coefficients is sampled by MCMC. The source code of the deflated (preconditioned) solvers can be found in the following file:

  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/RecyclingKrylovSolvers/defcg.jl`

- Application, Julia implementation and analysis of two different projection methods used for online eigenvector approximation, namely Rayleigh-Ritz (RR) and harmonic Rayleigh-Ritz (HR) projections, as well as of different restarting strategies of the eigen-search space, namely thick-restart (TR) and locally optimal thick-restart (LO-TR). The source code of the different projection methods and restarting strategies implemented for online eigenvector approximation within deflated (preconditioned) solvers can be found in the following files:

  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/RecyclingKrylovSolvers/rrdefpcg.jl`
  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/RecyclingKrylovSolvers/hrdefpcg.jl`
  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/RecyclingKrylovSolvers/trrrdefpcg.jl`
  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/RecyclingKrylovSolvers/trhrdefpcg.jl`
  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/RecyclingKrylovSolvers/lotrrrdefpcg.jl`
  - `https://github.com/venkovic/julia-phd-krylov-spdes/blob/master/RecyclingKrylovSolvers/lotrhrdefpcg.jl`

- Development of preconditiong strategies based on the quantization of random coefficient fields with stationary quantizers of two different maps of the latent random vectors.

- Development of preconditioning strategies based on the quantization of random coeffcient field with quantizers of the latent random vector with deterministic grids.

5.2 Summary of conclusions

Several conclusions were drawn throughout the thesis. We list the most important conclusions as follows:

- The preconditioning strategy based on delfation with online eigenvector approximation works better when used with preconditioners whose application leaves a trail of well separated eigenvalues at the lower extremity of the spectrum. Consequently, this strategy is recommended when using bJ preconditioners and preconditioners based on non-overlapping domain decomposition. It is however not recommended to use with AMG preconditioners.

- Both RR and HR projections lead to similar convergence behaviors of the deflated linear solve of sequences of correlated linear systems which arise when sampling the coefficient field of stochastic PDEs by MCMC. However, unlike HR projections, RR projections require no preconditioner application when building the reduced eigenvalue problem of eigenvector approximation. Consequently, RR projections lead to more efficient runtimes.

- Restarting the eigen-search space for the online eigenvector approximation used by deflated solvers leads to significant performance improvements, especially for linear systems of larger dimension. However, the LO-TR restarting strategy only proves to be more useful than TR when it is used with RR projections, particularly with bJ preconditioners.

- When using preconditioning strategies based on stationary quantizers for serial MC simulation of stochastic PDEs, it is recommended to use the map $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$ of the latent random vector $\boldsymbol{\xi}$ as it leads to smaller expected numbers of solver iterations.

- When using preconditioning strategies based on stationary quantizers for parallel MC simulation of stochastic PDEs, it is recommended to use the map $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}F_\xi \circ \boldsymbol{\xi}$ of the latent random vector $\boldsymbol{\xi}$ as it leads to a more balanced distribution of cumulated solver iterations among the different preconditioners.

- When using preconditioning strategies based on stationary quantizers of coefficient fields, the optimal number on KL modes in the approximating coefficient field depends on the number of preconditioners. As a means to optimally tune the number of KL modes to obtain a more performant preconditioning strategy, one may have to run preliminary computations. An alternative to stationary quantizers is a quantizer based on deterministic grids in the stochastic space for which the number of KL modes increases as a function of the number of preconditioners. The approach based on deterministic grids has shown to yield average numbers of solver iterations which are comparable to the best results obtained by a stationary quantizer built with $T_2^{-1}(\boldsymbol{\xi}) = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}$. The advantage of the approach based on deterministic grids is that it shows no stagnation as a function of the number of preconditioners, as opposed to using a stationary quantizer with a fixed number of KL modes, and it circumvents the need for preliminary computations.

5.3 Ideas of future works

The work presented in this thesis can serve as a motivation for further studies. In particular, we find the following to be interesting openings:

- The preconditioning strategy based on the deflation of correlated linear systems which was presented in Chapter 3 can be applied in the context of Bayesian inference of coefficient fields also referred to as inverse modeling. That is, one may have an array of observations of the coefficient field of a deterministic PDE at different locations within the domain Ω. The objective is then to infer the KL expansion of the coefficient field defined over the entire domain Ω given the observations. This approach relies on prior information related to the unknown KL expansion. Using Bayes' formula, one can formulate a posterior distribution of the random variables of the inferred KL expansion. The posterior distribution is defined in terms of a marginal likelihood which can be highly complex to evaluate. As a means to sample the posterior distribution, it is possible to proceed by MCMC and thus circumvent the need to evaluate the marginal likelihood. Then, we end up having to solve sequences of correlated linear systems using MCMC to sample the underlying coefficient field, exactly like in Chapter 3.

- As it was shown in Chapter 1, the preconditioning strategy which consists of assembling an AMG preconditioner for each realization of linear system yields excellent convergence properties. However, the cost of the naive assembly of a new AMG preconditioner from scratch on the basis of the Galerkin operator of every single realization of the coefficient field is prohibitive. In this thesis, we have used the same finite element mesh for the discretization of the PDE with all the realizations of the coefficient field. What our initial investigations have shown is that some the features built during the AMG assembly of all these Galerkin operators remain unchanged from one realization to another. As such, we believe it may be possible to leverage the fact that a single constant mesh is used for each of the Galerkin operators used for the AMG preconditioner definitions. We would like to recycle some of the features of the AMG setup, and infer others by some interpolation techniques so as to significantly decrease the AMG setup time for new realizations. The convergence behavior of the realization-dependent AMG preconditioner is so good in comparison to all the other preconditioning strategies investigated in this thesis, that speeding-up the setup time of AMG is perhaps the most promising options to build highly efficient preconditioning strategies for sotchastic elliptic PDEs.

- All the preconditioners presented in Chapter 1 are rather straightfoward to parallelize. Hence, as a means to solve much larger linear systems, the strategy based on deflation presented in Chapter 3 could be parallelized. When parallelizing these methods on distributed memory systems, the runtime is not governed anymore by computation, but rather by the communication which occurs at the different synchronization points of the parallel solver implementations. As such, we are interested in exploring communication-avoiding methods to improve the runtime of a parallel implementation, while still carrying the restart of the eigen-search space for the online eigenvector approximation. In particular, we are interested in s-step methods which consists of fusing s loop iterations of the linear solver so as to asymptotically

reduce sequential and parallel communication costs by a factor of $\mathcal{O}(s)$. A first application of s-step methods to deflated conjugate gradients was proposed by Carson et al. in [24].

Appendices

Appendix A

Practical considerations of the finite element method

The computation of a finite element approximation consists of the following tasks:

1. Input of the data on Ω and $\partial\Omega$ defining the problem to be solved.
2. Generation of a grid or mesh of elements.
3. Construction of the Galerkin system.
4. Solution of the discrete system, using a linear solver that exploits the sparsity of the finite element coefficient matrix.
5. A posteriori error estimation.

Here, we focus on the core aspect 3. of setting up the discrete Galerkin system, i.e., Eq. (1.97). The key idea in the implementation of finite element methodology is to consider everything elementwise, that is, locally one element at a time. In effect the discrete problem is broken up; for example, Eq. (1.96) is rewritten as

$$\sum_{j=1}^{n} u_j a(\phi_j, \phi_i) = \sum_{j=1}^{n} u_j \int_{\Omega} \nabla\phi_j(x) \cdot \nabla\phi_i(x) \mathrm{d}x = \sum_{j=1}^{n} u_j \left\{ \sum_{\triangle_k \in \mathcal{T}_h} \int_{\triangle_k} \nabla\phi_j(x) \cdot \nabla\phi_i(x) \mathrm{d}x \right\} \tag{A.1}$$

Notice that when forming the sum over the element in Eq. (A.1), we need only take account of those elements where the basis functions ϕ_i and ϕ_j are both nonzero. This means that entries A_{ij} and b_i in the Galerkin system can be computed by calculating contributions from each of the elements, and then gathering (or assembling) them together.

If the k-th element has n_k local degrees of freedom, then there are n_k basis functions that re not identically zero on the element. For example, in the case of a mesh made up

entirely of $\mathbb{P}_1$ triangles, we have $n_k = 3$ for all elements, so that in each $\triangle_k$ there are three element basis functions associated with the restriction of three different global basis functions ϕ_j. In all cases the local functions form an (element) basis set

$$\Xi_k := \{\psi_{k,1}, \psi_{k,2}, \ldots, \psi_{k,n_k}\}, \tag{A.2}$$

so that the solution within the element takes the form

$$u_h|_k = \sum_{i=1}^{n_k} u_i^{(k)} \psi_{k,i}. \tag{A.3}$$

Using triangular elements, for example, and localizing Eqs. (1.98) and (1.99), we need to compute a set of $n_k \times n_k$ element matrices $\mathbf{A}^{(k)}$ and a set of n_k-vectors $\mathbf{b}^{(k)}$ such that the components of $\mathbf{A}^{(k)}$ are given by

$$A_{ij}^{(k)} = \int_{\triangle_k} \nabla\phi_{k,i}(x) \cdot \nabla\phi_{k,j}(x) \mathrm{d}x, \tag{A.4}$$

and the components of $\mathbf{b}^{(k)}$ are

$$b_i^{(k)} = \int_{\triangle_k} f(x)\phi_{k,i}(x) \mathrm{d}x. \tag{A.5}$$

The matrix $\mathbf{A}^{(k)}$ is referred to as the element stiffness matrix (local stiffness matrix) associated with element $\triangle_k$. Notice that for computational convenience the essential boundary condition has not been enforced in Eq. (A.5). This is the standard implementation; essential conditions are usually imposed after the assembly of the element contributions into the Galerkin matrix has been completed. We will return to this point in the discussion of the assembly process.

A.1 Triangular element matrices

The first stage in the computation of the element stiffness matrix $\mathbf{A}^{(k)}$ is to map from a reference element $\triangle_*$ onto the given element $\triangle_k$, as illustrated in Fig. A.1. For straight sided triangles the local-global mapping is defined for all points $(x, y) \in \triangle_k$ and is given by

$$x(\xi, \eta) = x_1\chi_1(\xi, \eta) + x_2\chi_2(\xi, \eta) + x_3\chi_3(\xi, \eta) \tag{A.6}$$
$$y(\xi, \eta) = y_1\chi_1(\xi, \eta) + y_2\chi_2(\xi, \eta) + y_3\chi_3(\xi, \eta) \tag{A.7}$$

where

$$\chi_1(\xi, \eta) = 1 - \xi - \eta \tag{A.8}$$
$$\chi_2(\xi, \eta) = \xi \tag{A.9}$$
$$\chi_3(\xi, \eta) = \eta \tag{A.10}$$

are the $\mathbb{P}_1$ basis functions defined on the reference element.

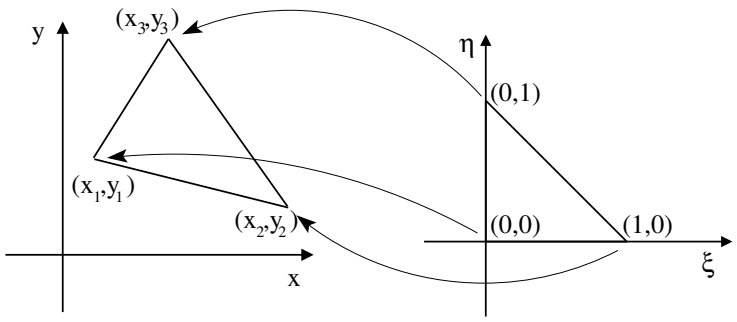

Figure A.1: Isoparametric mapping of $\mathbb{P}_1$ element.

Clearly, the map from the reference element onto $\triangle_k$ is (and has to be) differentiable. Thus, given a differentiable function $\varphi(\xi, \eta)$, we can transform derivatives via

$$\begin{bmatrix} \frac{\partial \varphi}{\partial \xi} \\ \frac{\partial \varphi}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial \varphi}{\partial x} \\ \frac{\partial \varphi}{\partial y} \end{bmatrix}. \tag{A.11}$$

The Jacobian matrix in Eq. (A.11) may be simply calculated by substituting Eqs. (A.8)–(A.10) in Eqs. (A.6)–(A.7) and differentiating to give

$$J_k = \frac{\partial(x, y)}{\partial(\xi, \eta)} = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{bmatrix}. \tag{A.12}$$

Thus in this simple case, we see that J_k is a constant matrix over the reference element, and that the determinant

$$|J_k| = \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix} = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} = 2|\triangle_k| \tag{A.13}$$

is simply the ratio of the area of the mapped element $\triangle_k$ to that of the reference element $\triangle_*$. The fact that $|J_k(\xi, \eta)| \neq 0$ for all points $(\xi, \eta) \in \triangle_*$ is very important; it ensures that the inverse mapping from $\triangle_k$ onto the reference element is uniquely defined and is differentiable. This means that the derivative transformation of Eq. (A.11) can be inverted to give

$$\begin{bmatrix} \frac{\partial \varphi}{\partial \xi} \\ \frac{\partial \varphi}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial \varphi}{\partial \xi} \\ \frac{\partial \varphi}{\partial \eta} \end{bmatrix}. \tag{A.14}$$

Thus we see that derivatives of functions defined on $\triangle_k$ satisfy

$$\frac{\partial \xi}{\partial x} = \frac{1}{|J_k|} \frac{\partial y}{\partial \eta}, \quad \frac{\partial \eta}{\partial x} = -\frac{1}{|J_k|} \frac{\partial y}{\partial \xi}, \tag{A.15}$$

$$\frac{\partial \xi}{\partial y} = -\frac{1}{|J_k|} \frac{\partial x}{\partial \eta}, \quad \frac{\partial \eta}{\partial y} = \frac{1}{|J_k|} \frac{\partial x}{\partial \xi}. \tag{A.16}$$

Given the basis functions on the master element $\psi_{*,i}$, $i = 1, \ldots, n_k$, the $\mathbb{P}_m$ element

stiffness matrix $\mathbf{A}^{(k)}$ in Eq. (A.4) is easily computed:

$$A_{ij}^{(k)} = \int_{\triangle_k} \left(\frac{\partial \psi_{k,i}}{\partial x} \frac{\partial \psi_{k,j}}{\partial x} + \frac{\partial \psi_{k,i}}{\partial y} \frac{\partial \psi_{k,j}}{\partial y} \right) \mathrm{d}x\mathrm{d}y \quad i,j = 1,\ldots,n_k \tag{A.17}$$

$$= \int_{\triangle_*} \left(\frac{\partial \psi_{*,i}}{\partial x} \frac{\partial \psi_{*,j}}{\partial x} + \frac{\partial \psi_{*,i}}{\partial y} \frac{\partial \psi_{*,j}}{\partial y} \right) |J_k| \mathrm{d}\xi\mathrm{d}\eta. \tag{A.18}$$

In the specific case of the linear mapping given by Eqs. (A.8)–(A.10), it is convenient to define the following coefficients:

$$b_1 = y_2 - y_3 \ , \quad b_2 = y_3 - y_1 \ , \quad b_3 = y_1 - y_2, \tag{A.19}$$

$$c_1 = x_3 - x_2 \ , \quad c_2 = x_1 - x_3 \ , \quad c_3 = x_2 - x_1 \tag{A.20}$$

in which case Eqs. (A.12)–(A.16) imply that

$$\begin{bmatrix} \frac{\partial \varphi}{\partial x} \\ \frac{\partial \varphi}{\partial y} \end{bmatrix} = \frac{1}{2|\triangle_k|} \begin{bmatrix} b_2 & b_3 \\ c_2 & c_3 \end{bmatrix} \begin{bmatrix} \frac{\partial \varphi}{\partial \xi} \\ \frac{\partial \varphi}{\partial \eta} \end{bmatrix}. \tag{A.21}$$

Combining Eq. (A.21) with Eq. (A.18) gives the general form of the stiffness matrix expressed in terms of the local derivatives of the element basis functions:

$$A_{ij}^{(k)} = \int_{\triangle_*} \left(b_2 \frac{\partial \psi_{*,i}}{\partial \xi} + b_3 \frac{\partial \psi_{*,i}}{\partial \eta} \right) \left(b_2 \frac{\partial \psi_{*,j}}{\partial \xi} + b_3 \frac{\partial \psi_{*,j}}{\partial \eta} \right) \frac{1}{|J_k|} \mathrm{d}\xi\mathrm{d}\eta \tag{A.22}$$

$$+ \int_{\triangle_*} \left(c_2 \frac{\partial \psi_{*,i}}{\partial \xi} + c_3 \frac{\partial \psi_{*,i}}{\partial \eta} \right) \left(c_2 \frac{\partial \psi_{*,j}}{\partial \xi} + c_3 \frac{\partial \psi_{*,j}}{\partial \eta} \right) \frac{1}{|J_k|} \mathrm{d}\xi\mathrm{d}\eta. \tag{A.23}$$

With the simplest linear approximation, that is, $\psi_{*,i} = \chi_i$ (see Eqs. (A.8)–(A.10)), the local derivatives $\partial\psi_{*,i}/\partial\xi$, $\partial\psi_{*,i}/\partial\eta$ are constant, so the local stiffness matrix is trivial to compute.

From a practical perspective, the simplest way of effecting the local transformation given by Eqs. (A.8)–(A.10) is to define local element functions using triangular or barycentric coordinates.

A.2 Assembly of the Galerkin system

The assembly of the element contributions $\mathbf{A}^{(k)}$ and $\mathbf{b}^{(k)}$ into the Galerkin system is a reversal of the localization process illustrated in Fig. A.2.

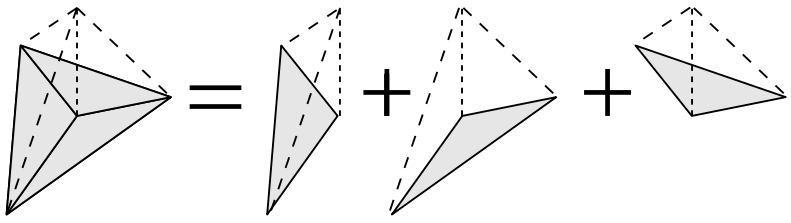

Figure A.2: Assembly of $\mathbb{P}_1$ global basis function from component element functions.

The main computational issue is the need for careful bookkeeping to ensure that the element contributions are added into the correct locations in the coefficient matrix $\mathbf{A}$ and the vector $\mathbf{b}$. The simplest way of implementing the process is to represent the mapping between local and global entities using a connectivity matrix. For example, in the case of the mesh of $\mathbb{P}_1$ triangles illustrated in Fig. A.3 we introduce the connectivity matrix defined by

$$P^T = \begin{bmatrix} 9 & 12 & 9 & 6 & 10 & 11 & 4 & 4 & 6 & 5 & 5 & 2 & 1 & 8 \\ 10 & 10 & 6 & 7 & 7 & 7 & 6 & 3 & 3 & 7 & 3 & 3 & 3 & 7 \\ 12 & 11 & 10 & 10 & 11 & 8 & 9 & 6 & 7 & 3 & 2 & 1 & 4 & 5 \end{bmatrix} \tag{A.24}$$

so that the index $j = P(k, i)$ specifies the global node number of local node i in element k, and thus identifies the coefficient $u_i^{(k)}$ in Eq. (A.3) with the global coefficient u_j in the expansion (1.94) of u_h. Given P, the matrices $\mathbf{A}^{(k)}$ and vectors $\mathbf{b}^{(k)}$ for the mesh in Fig. A.3 can be assembled into the Galerkin system matrix and vector using a set of nested loops, see Algo. 28.

Algorithm 28 Global assembly of Galerkin system for the mesh of Fig. A.3

Input: $\mathbf{A}^{(k)}$, $\mathbf{b}^{(k)}$ for $k = 1, \ldots, 14$
Output: $\mathbf{A}$, $\mathbf{b}$

1: $\mathbf{A} = \mathbf{0}$, $\mathbf{b} = \mathbf{0}$
2: **for** $k = 1, \ldots, 14$ **do**
3: **for** $j = 1, \ldots, 3$ **do**
4: **for** $i = 1, \ldots, 3$ **do**
5: $A_{P_{ki}P_{kj}} = A_{P_{ki}P_{kj}} + A_{ij}^{(k)}$
6: **end for**
7: $f_{P_{kj}} = f_{P_{kj}} + f_j^{(k)}$
8: **end for**
9: **end for**

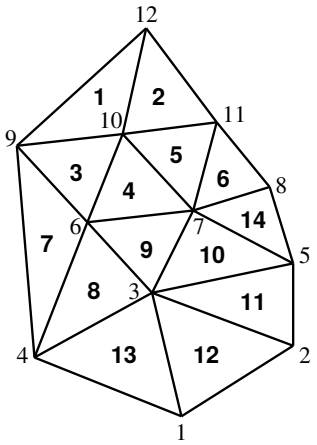

Figure A.3: Nodal and element numbering of a triangular mesh.

A few observations are appropriate here. First, in a practical implementation, the Galerkin matrix $\mathbf{A}$ will be stored in an appropriate sparse format. Second, it should be

apparent that as the elements are assembled in order above, then for any node s say, a stage will be reached when subsequent assemblies do not affect node s (i.e., the s-th row and column of the Galerkin matrix). When this stage is reached the variable is said to be fully summed; for example, variable 6 is fully summed after assembly of element 9. This observation motivates the development of specialized direct solvers (know as frontal solvers) whereby the assembly process is intertwined with Gaussian elimination. In essence, as soon as a variablebecomes fully summed, row operations can be performed to make entries below the diagonal zero and the modified row can then be saved for subsequent back-substitution.

It should also be emphasized that the intuitive element-by-element assembly embodied in the loop structure above is likely to be very inefficient; the inner loop involves indirect addressing and is too short to allow effective vectorization. The best way of generating efficient finite element code is to work with blocks of elements and to reorder the loops so that the element loop k is the innermost. For real efficiency the number of elements in a block should be set so that all required data can fit into cache memory.

We now turn our attention to the imposition of essential boundary conditions on the assembled Galerkin system. We assume here that the basis functions are Lagrangian type, that is, each basis function ϕ_j has a node $x_j \in \overline{\Omega}$ with it such that

$$\phi_j(x_j) = 1 \ , \quad \phi_j(x_i) = 0 \ \text{ for all nodes } x_i \neq x_j. \tag{A.25}$$

It follows from this assumption that for $x_j \in \partial\Omega$, $u_h(x_j) = u_j$, where the required value of u_j is interpolated from the Dirichlet boundary data.

Now consider how to impose this condition at node 5 of the mesh in Fig. A.3. Suppose that a preliminary version of the Galerkin matrix $\mathbf{A}$ is constructed via Eq. (A.4) for $1 \leq i$, $j \leq n+n_\partial$, and that in addition,all the contributions $\int_\Omega \phi_i(x) f(x) \mathrm{d}x$ have been assembled into the right-hand side vector $\mathbf{b}$. There are then two things needed to specify the system (1.97) via Eqs. (1.96) and (1.99): the given value of u_5 must be included in the definition of the vector $\mathbf{b}$ of Eq. (1.99), and the fifth and column of the preliminary Galerkin matrix must be deleted (since ϕ_5 is being removed from the space of test functions). The first step can be achieved by multiplying the fifth column of $\mathbf{A}$ by the specified boundary value u_5 and then subtracting the result from $\mathbf{b}$. An alternative technique is to retain the imposed degree of freedom in the Galerkin system by modifying the row and column (5, here) of the Galerkin matrix corresponding to the boundary node so that the diagonal value is unity and the off-diagonal entries are set to zero, and then setting the corresponding value of $\mathbf{b}$ to the boundary value u_5. Notice that the modified Galerkin matrix thus has a multiple eigenvalue of unity, with multiplicity equal to the number of nodes on the boundary.

Bibliography

[1] Simon Abraham, Mehrdad Raisee, Ghader Ghorbaniasl, Francesco Contino, and Chris Lacor. A robust and efficient stepwise regression method for building sparse polynomial chaos expansions. *Journal of Computational Physics*, 332:461–474, 2017.

[2] Peter Arbenz, Daniel Kressner, and DME Zürich. Lecture notes on solving large scale eigenvalue problems. *D-MATH, EHT Zurich*, 2, 2012.

[3] Walter Edwin Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics*, 9(1):17–29, 1951.

[4] Christophe Audouze and Prasanth B Nair. Sparse approximate solutions to stochastic Galerkin equations. *Comptes Rendus Mathematique*, 357(6):561–570, 2019.

[5] Ivo Babuška, Fabio Nobile, and Raúl Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 45(3):1005–1034, 2007.

[6] Ivo Babuška, Raúl Tempone, and Georgios E Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 42(2):800–825, 2004.

[7] Feng Bai and Yi Wang. Reduced-order modeling based on hybrid snapshot simulation. *International Journal of Computational Methods*, 18(01):2050029, 2021.

[8] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *Journal of machine learning research*, 6(Oct):1705–1749, 2005.

[9] Andrea Barth, Christoph Schwab, and Nathaniel Zollinger. Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients. *Numerische Mathematik*, 119:123–161, 2011.

[10] Joakim Beck, Raul Tempone, Fabio Nobile, and Lorenzo Tamellini. On the optimal polynomial approximation of stochastic PDEs by Galerkin and collocation methods. *Mathematical Models and Methods in Applied Sciences*, 22(09):1250023, 2012.

[11] WN Bell, LN Olson, and JB Schroder. PyAMG: Algebraic multigrid solvers in Python v2.0. *http://www.pyamg.org*, 2011.

[12] Peter Benner, Mario Ohlberger, Albert Cohen, and Karen Willcox. *Model reduction and approximation: theory and algorithms*. SIAM, 2017.

[13] A Benveniste, M Métivier, and B Priouret. Algorithmes adaptatifs et approximations stochastiques, collection techniques stochastiques, 1987.

[14] Wolfgang Betz, Iason Papaioannou, and Daniel Straub. Numerical methods for the discretization of random fields by means of the Karhunen–Loève expansion. *Computer Methods in Applied Mechanics and Engineering*, 271:109–129, 2014.

[15] Géraud Blatman and Bruno Sudret. Efficient computation of global sensitivity indices using sparse polynomial chaos expansions. *Reliability Engineering & System Safety*, 95(11):1216–1229, 2010.

[16] Géraud Blatman and Bruno Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of computational Physics*, 230(6):2345–2367, 2011.

[17] Leon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. *Advances in neural information processing systems*, 7, 1994.

[18] A Brandt, S McCormick, and J Ruge. Algebraic multigrid (AMG) for automatic algorithm design and problem solution. *Report,. Comp. Studies, Colorado State University, Ft. Collins*, 1982.

[19] Achi Brandt. Algebraic multigrid (AMG) for sparse matrix eqations. *Sparsity and its Applications*, pages 257–284, 1984.

[20] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[21] Luc Brun and M Mokhtari. Two high speed color quantization algorithms. In *Proceedings of the 1st international conference on color in graphics and image processing*, pages 116–121, 2000.

[22] Russel E Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta numerica*, 7:1–49, 1998.

[23] Daniela Calvetti, Lothar Reichel, and Danny Chris Sorensen. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis*, 2(1):21, 1994.

[24] Erin Carson, Nicholas Knight, and James Demmel. An efficient deflation technique for the communication-avoiding conjugate gradient method. *Electronic Transactions on Numerical Analysis*, 43(125141):09, 2014.

[25] M Emre Celebi. Improving the performance of k-means for color quantization. *Image and Vision Computing*, 29(4):260–271, 2011.

[26] Tony F Chan, Tarek P Mathew, et al. Domain decomposition algorithms. *Acta numerica*, 3(1):61–143, 1994.

[27] Cédric Chevalier and François Pellegrini. PT-Scotch: A tool for efficient parallel graph ordering. *Parallel computing*, 34(6-8):318–331, 2008.

[28] Albert Cohen, Ronald DeVore, and Christoph Schwab. Convergence rates of best N-term Galerkin approximations for a class of elliptic sPDEs. *Foundations of Computational Mathematics*, 10(6):615–646, 2010.

[29] Patrick R Conrad and Youssef M Marzouk. Adaptive Smolyak pseudospectral approximations. *SIAM Journal on Scientific Computing*, 35(6):A2643–A2670, 2013.

[30] Paul G Constantine, Michael S Eldred, and Eric T Phipps. Sparse pseudospectral approximation method. *Computer Methods in Applied Mechanics and Engineering*, 229:1–12, 2012.

[31] Andres A Contreras, Paul Mycek, Olivier P Le Maître, Francesco Rizzi, Bert Debusschere, and Omar M Knio. Parallel domain decomposition strategies for stochastic elliptic equations. Part A: Local Karhunen–Loève representations. *SIAM Journal on Scientific Computing*, 40(4):C520–C546, 2018.

[32] Olivier Coulaud, Luc Giraud, Pierre Ramet, and Xavier Vasseur. Deflation and augmentation techniques in Krylov subspace methods for the solution of linear systems. *arXiv preprint arXiv:1303.5692*, 2013.

[33] Roy Cranley and Thomas NL Patterson. Randomization of number theoretic methods for multiple integration. *SIAM Journal on Numerical Analysis*, 13(6):904–914, 1976.

[34] Hussam Al Daas, Tyrone Rees, and Jennifer Scott. Two-level Nyström–Schur preconditioner for sparse symmetric positive definite matrices. *arXiv preprint arXiv:2101.12164*, 2021.

[35] Yann-Hervé De Roeck and P LeTallec. Analysis and test of a local domain decomposition. In *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, YA Kuznetsov, GA Meurant, J. Périaux, and OB Widlund, eds., Philadelphia*, pages 112–128, 1991.

[36] Manas K Deb, Ivo M Babuška, and J Tinsley Oden. Solution of stochastic partial differential equations using Galerkin finite element techniques. *Computer Methods in Applied Mechanics and Engineering*, 190(48):6359–6372, 2001.

[37] James W Demmel. *Applied numerical linear algebra*. SIAM, 1997.

[38] Josef Dick, Frances Y Kuo, and Ian H Sloan. High-dimensional integration: the quasi-Monte Carlo way. *Acta Numerica*, 22:133–288, 2013.

[39] Colin R Dietrich and Garry N Newsam. A fast and exact method for multidimensional Gaussian stochastic simulations. *Water Resources Research*, 29(8):2861–2869, 1993.

[40] Iain S Duff, Albert M Erisman, C William Gear, and John K Reid. Sparsity structure and Gaussian elimination. *ACM Signum newsletter*, 23(2):2–8, 1988.

[41] Marie Duflo. *Algorithmes stochastiques*, volume 23. Springer, 1996.

[42] Howard C Elman, David J Silvester, and Andrew J Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press, USA, 2014.

[43] Jocelyne Erhel and Frédéric Guyomarc'h. An augmented conjugate gradient method for solving consecutive symmetric positive definite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1279–1299, 2000.

[44] Robert D Falgout and Ulrike Meier Yang. hypre: A library of high performance preconditioners. In *International Conference on Computational Science*, pages 632–641. Springer, 2002.

[45] Aurélie Fischer. Quantization and clustering with Bregman divergences. *Journal of Multivariate Analysis*, 101(9):2207–2221, 2010.

[46] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, 21:768–769, 1965.

[47] Bela A Frigyik, Santosh Srivastava, and Maya R Gupta. Functional Bregman divergence and Bayesian estimation of distributions. *IEEE Transactions on Information Theory*, 54(11):5130–5139, 2008.

[48] Dani Gamerman and Hedibert F Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC press, 2006.

[49] S Geman. Gibbs distribution, and the Bayesian restoration of images. *IEEE Proc. Pattern Analysis and Machine Intelligence*, 6:774–778, 1984.

[50] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[51] Michael Gervautz and Werner Purgathofer. A simple method for color quantization: Octree quantization. In *New trends in computer graphics*, pages 219–231. Springer, 1988.

[52] Roger G Ghanem and Pol D Spanos. *Stochastic finite elements: a spectral approach*. Courier Corporation, 2003.

[53] Michael B Giles. Multilevel Monte Carlo methods. *Acta numerica*, 24:259–328, 2015.

[54] L Giraud and RS Tuminaro. Algebraic domain decomposition preconditioners. *Mesh partitioning techniques and domain decomposition methods*, pages 187–216, 2006.

[55] Luc Giraud, Daniel Ruiz, and Ahmed Touhami. A comparative study of iterative solvers exploiting spectral information for SPD systems. *SIAM Journal on Scientific Computing*, 27(5):1760–1786, 2006.

[56] Siegfried Graf and Harald Luschgy. *Foundations of quantization for probability distributions*. Springer, 2007.

[57] Ivan G Graham, Frances Y Kuo, Dirk Nuyens, Robert Scheichl, and Ian H Sloan. Quasi-Monte Carlo methods for elliptic PDEs with random coefficients and applications. *Journal of Computational Physics*, 230(10):3668–3694, 2011.

[58] Anne Greenbaum. Comparison of splittings used with the conjugate gradient algorithm. *Numerische Mathematik*, 33(2):181–193, 1979.

[59] Anne Greenbaum. *Iterative methods for solving linear systems*. SIAM, 1997.

[60] Anne Greenbaum and Leonid Gurvits. Max-min properties of matrix factor norms. *SIAM Journal on Scientific Computing*, 15(2):348–358, 1994.

[61] Laura Grigori, Frédéric Nataf, and Soleiman Yousef. *Robust algebraic Schur complement preconditioners based on low rank corrections*. PhD thesis, INRIA, 2014.

[62] Marcus J Grote and Thomas Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997.

[63] Martin H Gutknecht. Spectral deflation in Krylov solvers: A theory of coordinate space based methods. *Electron. Trans. Numer. Anal*, 39:156–185, 2012.

[64] Abdul-Lateef Haji-Ali, Fabio Nobile, and Raúl Tempone. Multi-index Monte Carlo: when sparsity meets sampling. *Numerische Mathematik*, 132:767–806, 2016.

[65] J. Hammersley and M. Clifford. Markov fields on finite graphs and lattices. *http://www.statslab.cam.ac.uk/ grg/books/hammfest/hamm-cliff.pdf*, 1970.

[66] W Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. 1970.

[67] Michael A Heroux, Phuong Vu, and Chao Yang. A parallel preconditioned conjugate gradient package for solving sparse linear systems on a Cray Y-MP. *Applied Numerical Mathematics*, 8(2):93–115, 1991.

[68] Claes Johnson. *Numerical solution of partial differential equations by the finite element method*. Courier Corporation, 2012.

[69] Galin L Jones et al. On the Markov chain central limit theorem. *Probability surveys*, 1(299-320):5–1, 2004.

[70] Mark T Jones and Paul E Plassmann. The efficient parallel iterative solution of large sparse linear systems. In *Graph theory and sparse matrix computation*, pages 229–245. Springer, 1993.

[71] M. Kac and A.J.F. Siegert. An explicit representation of a stationary Gaussian process. *Ann. Math. Stat.*, 18:438–442, 1947.

[72] K Kahl and H Rittich. The deflated conjugate gradient method: Convergence, perturbation and accuracy. *Linear Algebra and its Applications*, 515:111–129, 2017.

[73] K Karhunen. Über lineare methoden in der wahrscheinlichkeitsrechnung. *Annal. Acad. Sci. Fennicae*, 37:1–79, 1947.

[74] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[75] J Kieffer. Exponential rate of convergence for Lloyd's method I. *IEEE Transactions on Information Theory*, 28(2):205–210, 1982.

[76] Andrew V Knyazev. A preconditioned conjugate gradient method for eigenvalue problems and its implementation in a subspace. In *Numerical Treatment of Eigenvalue Problems Vol. 5/Numerische Behandlung von Eigenwertaufgaben Band 5*, pages 143–154. Springer, 1991.

[77] Andrew V Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing*, 23(2):517–541, 2001.

[78] Teuvo Kohonen. Learning vector quantization. In *Self-organizing maps*, pages 175–189. Springer, 1995.

[79] Katerina Konakli and Bruno Sudret. Global sensitivity analysis using low-rank tensor approximations. *Reliability Engineering & System Safety*, 156:64–83, 2016.

[80] Boris Kozintsev. *Computations with Gaussian random fields*. PhD thesis, 1999.

[81] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of Monte Carlo methods*. John Wiley & Sons, 2013.

[82] Frances Y Kuo and Dirk Nuyens. Application of quasi-Monte Carlo methods to elliptic PDEs with random diffusion coefficients: a survey of analysis and implementation. *Foundations of Computational Mathematics*, 16:1631–1696, 2016.

[83] Thomas Laloë. L1-quantization and clustering in Banach spaces. *Mathematical Methods of Statistics*, 19(2):136–150, 2010.

[84] Alan J Laub. *Matrix analysis for scientists and engineers*, volume 91. SIAM, 2005.

[85] Olivier Le Maître and Omar M Knio. *Spectral methods for uncertainty quantification: with applications to computational fluid dynamics*. Springer Science & Business Media, 2010.

[86] Patrick Le Tallec, Yann-Hervé De Roeck, and Marina Vidrascu. Domain decomposition methods for large linearly elliptic three-dimensional problems. *Journal of Computational and Applied Mathematics*, 34(1):93–117, 1991.

[87] Pierre L'Ecuyer. Uniform random number generation. *Annals of Operations Research*, 53(1):77–120, 1994.

[88] Pierre L'Ecuyer. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47(1):159–164, 1999.

[89] Pierre L'Ecuyer. Software for uniform random number generation: Distinguishing the good and the bad. In *Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01CH37304)*, volume 1, pages 95–105. IEEE, 2001.

[90] RB Lehoucq, DC Sorensen, and C Yang. ARPACK users' guide: Solution of large scale eigenvalue problems with implicitly restarted Arnoldi methods.

[91] C. Lemieux. *Monte Carlo and quasi-Monte Carlo sampling*. Springer-Verlag, New York, 2009.

[92] Ruipeng Li and Yousef Saad. Low-rank correction methods for algebraic domain decomposition preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 38(3):807–828, 2017.

[93] Tamás Linder. Learning-theoretic methods in vector quantization. In *Principles of nonparametric learning*, pages 163–210. Springer, 2002.

[94] Jun S Liu. *Monte Carlo strategies in scientific computing*, volume 10. Springer, 2001.

[95] Stuart Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[96] Michel Loève. Functions aléatoires du second ordre. *Processus stochastique et mouvement Brownien*, pages 366–420, 1948.

[97] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297, 1967.

[98] Jan Mandel. Balancing domain decomposition. *Communications in numerical methods in engineering*, 9(3):233–241, 1993.

[99] Pierre Maréchal and Jane J Ye. Optimizing condition numbers. *SIAM Journal on Optimization*, 20(2):935–947, 2009.

[100] Pierre Matalon. *Fast solvers for robust discretizations in computational fluid dynamics*. PhD thesis, Université Montpellier; Friedrich-Alexander-Universität Erlangen-Nürnberg, 2021.

[101] J. Matoušek. On the L2-discrepancy for anchored boxes. *Journal of Complexity*, 14(4):527–556, 1998.

[102] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[103] Ronald B Morgan. Computing interior eigenvalues of large matrices. *Linear Algebra and its Applications*, 154-156:289–309, 1991.

[104] Ronald B Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Mathematics of Computation*, 65(215):1213–1230, 1996.

[105] Ronald B Morgan and Min Zeng. A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity. *Linear algebra and its applications*, 415(1):96–113, 2006.

[106] Peter Neal and Gareth Roberts. Optimal scaling for partially updating MCMC algorithms. *The Annals of Applied Probability*, 16(2):475–515, 2006.

[107] Roy A Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis*, 24(2):355–365, 1987.

[108] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. SIAM, 1992.

[109] Fabio Nobile, Raúl Tempone, and Clayton G Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46(5):2309–2345, 2008.

[110] Anthony Nouy. Generalized spectral decomposition method for solving stochastic finite element equations: invariant subspace problem and dedicated algorithms. *Computer Methods in Applied Mechanics and Engineering*, 197(51-52):4718–4736, 2008.

[111] Anthony Nouy. Proper generalized decompositions and separated representations for the numerical solution of high dimensional stochastic problems. *Archives of Computational Methods in Engineering*, 17(4):403–434, 2010.

[112] Anthony Nouy and Olivier P Le Maître. Generalized spectral decomposition for stochastic nonlinear problems. *Journal of Computational Physics*, 228(1):202–235, 2009.

[113] Wolfgang Nowak. *Geostatistical methods for the identification of flow and transport parameters in the subsurface*. 2005.

[114] L. N. Olson and J. B. Schroder. PyAMG: Algebraic multigrid solvers in Python v4.0, 2018. Release 4.0.

[115] Art B Owen. Randomly permuted (t, m, s)-nets and (t, s)-sequences. In *Monte Carlo and quasi-Monte Carlo methods in scientific computing*, pages 299–317. Springer, 1995.

[116] Art B Owen. Scrambled net variance for integrals of smooth functions. *The Annals of Statistics*, 25(4):1541–1562, 1997.

[117] Gilles Pagès. A space quantization method for numerical integration. *Journal of computational and applied mathematics*, 89(1):1–38, 1998.

[118] Gilles Pagès and Jacques Printems. Optimal quadratic quantization for numerics: the Gaussian case. *Monte Carlo methods and applications*, 9(2):135–165, 2003.

[119] Chris C Paige, Beresford N Parlett, and Henk A Van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numerical linear algebra with applications*, 2(2):115–133, 1995.

[120] Christopher C Paige. Computational variants of the Lanczos method for the eigenproblem. *IMA Journal of Applied Mathematics*, 10(3):373–381, 1972.

[121] Christopher Conway Paige. *The computation of eigenvalues and eigenvectors of very large sparse matrices.* PhD thesis, University of London, 1971.

[122] Michael L Parks, Eric De Sturler, Greg Mackey, Duane D Johnson, and Spandan Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5):1651–1674, 2006.

[123] Beresford N Parlett. A new look at the Lanczos algorithm for solving symmetric systems of linear equations. *Linear algebra and its applications*, 29:323–346, 1980.

[124] Beresford N Parlett. *The symmetric eigenvalue problem*, volume 20. SIAM, 1998.

[125] Beresford N Parlett and David S Scott. The Lanczos algorithm with selective orthogonalization. *Mathematics of computation*, 33(145):217–238, 1979.

[126] Louis Poirel. *Algebraic domain decomposition methods for hybrid (iterative/direct) solvers.* PhD thesis, Université de Bordeaux, 2018.

[127] Brian David Ripley. The lattice structure of pseudo-random number generators. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 389(1796):197–204, 1983.

[128] Pieterjan Robbe, Dirk Nuyens, and Stefan Vandewalle. A multi-index quasi–Monte Carlo algorithm for lognormal diffusion problems. *SIAM Journal on Scientific Computing*, 39(5):S851–S872, 2017.

[129] Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

[130] Gareth O Roberts, Andrew Gelman, Walter R Gilks, et al. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The annals of applied probability*, 7(1):110–120, 1997.

[131] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

[132] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical science*, 16(4):351–367, 2001.

[133] Gareth O Roberts and Richard L Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

[134] GO Roberts and RL Tweedie. Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. *Biometrika*, pages 95–110, 1996.

[135] Karl F Roth. On irregularities of distribution. *Mathematika*, 1(2):73–79, 1954.

[136] John W Ruge and Klaus Stüben. Algebraic multigrid. In *Multigrid methods*, pages 73–130. SIAM, 1987.

[137] Yousef Saad. On the Lanczos method for solving symmetric linear systems with several right-hand sides. *Mathematics of computation*, 48(178):651–662, 1987.

[138] Yousef Saad. *Iterative methods for sparse linear systems*, volume 82. SIAM, 2003.

[139] Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition*, volume 66. SIAM, 2011.

[140] Yousef Saad, Manshung Yeung, Jocelyne Erhel, and Frédéric Guyomarc'h. A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing*, 21(5):1909–1926, 2000.

[141] Michael Sabin and Robert Gray. Global convergence and empirical consistency of the generalized Lloyd algorithm. *IEEE Transactions on information theory*, 32(2):148–155, 1986.

[142] Luis A Salomón, Jean-Claude Fort, and Li-Vang Lozada-Chang. Average competitive learning vector quantization. *Communications in Statistics-Simulation and Computation*, 43(6):1288–1303, 2014.

[143] Martin Schlather. An introduction to positive definite functions and to unconditional simulation of random fields. Technical report, Technical report st 99-10, Dept. of Mathematics and Statistics, Lancaster ..., 1999.

[144] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[145] Jonathan Richard Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Workshop on Applied Computational Geometry*, pages 203–222. Springer, 1996.

[146] Horst D Simon. The Lanczos algorithm with partial reorthogonalization. *Mathematics of computation*, 42(165):115–142, 1984.

[147] Barry F Smith. Domain decomposition methods for partial differential equations. In *Parallel Numerical Algorithms*, pages 225–243. Springer, 1997.

[148] Andreas Stathopoulos and Konstantinos Orginos. Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics. *SIAM Journal on Scientific Computing*, 32(1):439–462, 2010.

[149] Hugo Steinhaus et al. Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci*, 1(804):801, 1956.

[150] Gilbert W Stewart. A Krylov–Schur algorithm for large eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 23(3):601–614, 2002.

[151] J. M. Tang, R. Nabben, C. Vuik, and Y. A. Erlangga. Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods. *Journal of Scientific Computing*, 39(3):340–370, 2009.

[152] Kunkun Tang, Pietro M Congedo, and Rémi Abgrall. Adaptive surrogate modeling by ANOVA and sparse polynomial dimensional decomposition for global sensitivity analysis in fluid simulation. *Journal of Computational Physics*, 314:557–589, 2016.

[153] Aretha Leonore Teckentrup. *Multilevel Monte Carlo methods and uncertainty quantification*. PhD thesis, University of Bath, 2013.

[154] Shu Tezuka. *Uniform random numbers: Theory and practice*, volume 315. Springer Science & Business Media, 2012.

[155] Andrea Toselli and Olof Widlund. *Domain decomposition methods-algorithms and theory*, volume 34. Springer Science & Business Media, 2006.

[156] Henk A Van der Vorst. An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A. *Journal of Computational and Applied Mathematics*, 18(2):249–263, 1987.

[157] Henk A van der Vorst. ICCG and related methods for 3D problems on vector computers. *Computer Physics Communications*, 53(1-3):223–235, 1989.

[158] Petr Vaněk, Jan Mandel, and Marian Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996.

[159] Nicolas Venkovic, Paul Mycek, Luc Giraud, and Olivier Le Maitre. Comparative study of harmonic and Rayleigh-Ritz procedures with applications to deflated conjugate gradients. *Technical Report of Cerfacs*, TR-PA-20-3, 2020.

[160] Nicolas Venkovic, Paul Mycek, Luc Giraud, and Olivier Le Maitre. Recycling Krylov subspace strategies for sequences of sampled stochastic elliptic equations. *Research Report of Inria Bordeaux-Sud Ouest*, RR-9425, 2021.

[161] Michel Verleysen. Les principaux modèles de réseaux de neurones artificiels. In *Les réseaux de neurones en finance: conception et applications*, 1995.

[162] Justin Winokur, Daesang Kim, Fabrizio Bisetti, Olivier P Le Maître, and Omar M Knio. Sparse pseudo spectral projection methods with directional adaptation for uncertainty quantification. *Journal of Scientific Computing*, 68:596–623, 2016.

[163] Rafi Witten and Emmanuel Candes. Randomized algorithms for low-rank matrix factorizations: sharp performance bounds. *Algorithmica*, 72(1):264–281, 2015.

[164] Andrew TA Wood and Grace Chan. Simulation of stationary Gaussian processes in $[0, 1]^d$. *Journal of computational and graphical statistics*, 3(4):409–432, 1994.

[165] Kesheng Wu and Horst Simon. Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 22(2):602–616, 2000.

[166] Zhigang Xiang. Color image quantization by minimizing the maximum intercluster distance. *ACM Transactions on Graphics (TOG)*, 16(3):260–276, 1997.

[167] Dongbin Xiu. *Numerical methods for stochastic computations: a spectral method approach*. Princeton university press, 2010.

[168] Olivier Zahm and Anthony Nouy. Interpolation of inverse operators for preconditioning parameter-dependent equations. *SIAM Journal on Scientific Computing*, 38(2):A1044–A1074, 2016.