Solution of large linear systems with a massive number of right-hand sides and machine learning

Yan-Fei Xiang

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 **Yanfei XIANG**

Résolution de systèmes linéaires de grande taille avec un nombre massif de second-membres et apprentissage

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

Soutenue le 7 décembre 2022

Membres du jury :

M. Eric de STURLER	Professor	Virginia Tech	Président
M. Andreas FROMMER	Professor	Bergische Universität Wuppertal	Rapporteur
M. Stéphane LANTERI	Directeur de Recherche	Inria	Examinateur
M. Michael BAUERHEIM	Associate Professor	ISAE-SUPAERO	Examinateur
M. Jayant SENGUPTA	Leader dept data-science/AI	Airbus C R & T	Examinateur
Mme. Carola KRUSE	Senior Scientist	Cerfacs	Examinatrice
M. Luc GIRAUD	Directeur de Recherche	Inria	Directeur de thèse
M. Paul MYCEK	Senior Scientist	Cerfacs	Directeur de thèse

Résolution de systèmes linéaires de grande taille avec un nombre massif de second-membres et apprentissage

Résumé : Ce travail se concentre sur la résolution itérative de grands systèmes linéaires avec des second-membres multiples qui apparaissent dans diverses applications scientifiques. Lorsque de multiples second-membres doivent être résolus, les variantes par blocs des méthodes du sous-espace de Krylov sont les méthodes de choix. La mise en œuvre de l'algorithme des blocs permettant l'utilisation de noyaux de calcul efficaces de type BLAS-3, le temps de résolution devrait être réduit. Malheureusement, ces avantages potentiels se font au prix de difficultés numériques induites par l'éventuelle différence de vitesse de convergence des second-membres. Cette caractéristique numérique est appelée convergence partielle. Pour les séquences de systèmes linéaires non symétriques, dans le chapitre 2, nous développons une nouvelle approche de résidu de norme minimale par bloc qui combine deux ingrédients principaux. Le premier composant exploite les idées de la méthode GCRO-DR, ce qui nous permet de recycler les informations spectrales d'un système linéaire à un autre. Le deuxième composant est le mécanisme numérique de gestion de la convergence partielle desecond-membres appelé mécanisme de détection breakdown incomplet dans la méthode IB-BGMRES. Étant donné que le problème de la convergence partielle se pose dans tous les types de solveurs de Krylov par blocs, dans le chapitre 3, nous étendons ce mécanisme à la variante par blocs de la méthode du résidu conjugué, une approche de résidu de norme minimale à récurrence courte pour des systèmes symétrique indéfini. Ensuite, inspirés par l'idée de réutiliser l'information spectrale afin d'accélérer la convergence, nous concevons au chapitre 4 une stratégie de recyclage du sous-espace correspondant. Plus précisément, nous appliquons les idées de "thick-restart', introduites dans la méthode de Lanczos pour le calcul des paires propres, dans l'algorithme du gradient conjugué par blocs (BCG) pour les systèmes linéaires symétriques définis positifs. Nous étudions également la possibilité d'utiliser la détection de convergence partielle dans BCG. Enfin, ces stratégies de détection de convergence partielle et de recyclage du sous-espace peuvent être combinées efficacement pour concevoir un algorithme de gradient conjugué par blocs déflatés pour les séquences de systèmes linéaires définis positifs symétriques.

Une voie alternative aux approches traditionnelles d'algèbre linéaire numérique mentionnées ci-dessus consiste à envisager l'utilisation des techniques d'apprentissage automatique. Dans le chapitre 5, nous présentons quelques façons d'hybrider les nouveaux solveurs d'apprentissage profond et les techniques d'algèbre linéaire numérique plus traditionnelles afin qu'ils puissent bénéficier les uns des autres. Dans le contexte de la résolution d'une équation de Helmholtz hétérogène, nous nous concentrons d'abord sur l'introduction de certains ingrédients mathématiques d'un solveur itératif classique dans la phase d'apprentissage d'un solveur de réseau profond de neurones récemment proposé. Le principal avantage est une amélioration significative de la phase d'apprentissage, plus robuste et plus rapide, qui s'avère également applicable au processus de test. En outre, une fois que les réseaux ont été correctement entrainés, leurs inférences peuvent être appliquées comme préconditionneur non linéaire dans les méthodes GMRES et FOM flexibles traditionnelle. Cette partie démontre que ces variantes hybrides présentent des avantages évidents par rapport à la fois à l'approche récemment introduite et au solveurs itératifs classiques de sous-espaces, tant en termes de coût de calcul que de précision des résultats.

Mots-clés : Méthode du sous-espace de Krylov, algèbre linéaire numérique, apprentissage automatique, apprentissage profond, Méthodes de sous-espace par bloc, Recyclage de sous-espace, Détection de convergence partielle, Critère d'arrêt d'erreur inverse, Calcul arithmétique mixte.

Solution of large linear systems with a massive number of right-hand sides and machine learning

Abstract: This work focuses on the iterative solution of large linear systems with multiple right-hand sides that appear in various scientific applications. When multiple right-hand sides have to be solved, block variants of Krylov subspace methods are the methods of choice. Because the block algorithm implementation enables the use of efficient BLAS-3 like computational kernels, the time for solution is expected to reduce. Unfortunately these potential advantages come at the price of numerical difficulties induced by the possible different convergence rate of the right-hand sides or linear combination of some of them. This numerical feature is referred to as partial convergence. For sequences of unsymmetric linear systems in Chapter 2, we develop novel block minimum norm residual approach that combines two main ingredients. The first component exploits ideas from GCRO-DR [80], enabling us to recycle spectral information from one linear system to the next. The second component is the numerical mechanism for managing partial convergence of the right-hand sides, referred to as the inexact breakdown detecting mechanism in IB-BGMRES [88], that enables the monitoring of the rank deficiency in the residual space basis expanded blockwise. Next, for the class of block minimum norm residual approaches, that relies on a block Arnoldi-like equality between the search space and residual space, we introduce new search space expansion policies defined on novel criteria to detect partial convergence. These novel detection criteria are tuned to the targeted stopping criterion and convergence threshold. This enables us to monitor the computational effort while ensuring the final accuracy of each individual solution. Because the partial convergence issue appears in any type of block Krylov solvers, in Chapter 3, we extend the partial convergence detecting mechanism to the block variant of the conjugate residual method [66], a minimum residual norm approach with short term recurrence scheme, for symmetric but not necessary positive definite linear systems. Then, inspired by the idea of reusing generated information to approximate spectral information for accelerating convergence, in Chapter 4, we devise a corresponding subspace recycling strategy. More precisely, we apply the thick-restart ideas [125] introduced in the Lanczos method for eigenpair calculation in the block conjugate gradient algorithm (BCG) [77] for the symmetric positive definite linear systems. We also investigate the possibility to use the partial convergence detection to the BCG algorithm as a heuristic. Finally, these partial convergence detection and subspace recycling strategies can be efficiently combined to design a deflated block conjugate gradient algorithm for sequences of symmetric positive definite linear systems.

An alternative path to the above mentioned traditional numerical linear algebra approaches is to consider using the scientific machine learning techniques [64, 75]. In Chapter 5, we presents some ways of hybridizing the newly emerging deep learning solvers and the more traditional numerical linear algebra techniques to let them benefit from each other. In the context of solving a heterogeneous Helmholtz equation, we first focus on introducing some mathematical ingredients from classical iterative solver into the training phase of a recently proposed deep neural network solver. The main benefit is a significant improvement in the training phase that is more robust and faster, which turns out to be applicable to the testing process as well. Furthermore, once the network solvers have been properly trained, their inferences can be applied as a nonlinear preconditioner in the traditional flexible GMRES and flexible FOM methods. This part demonstrates that these hybrid variants have clear advantages over both the newly emerging deep neural network approach and the classical iterative Krylov solver in terms of both computational cost and accuracy of the computed solution.

Keywords: Krylov subspace method, Numerical linear algebra, Machine learning, Deep learning, Scientific machine learning, Block subspace methods, Subspace recycling, Partial convergence detection, Backward error stopping criterion, Mixed arithmetic calculation.

Inria, CONCACE
UMR xxxx Université, 33000 Bordeaux, France.

Contents

Acknowledgements

Thanks are due to my adviser, Professor Luc Giraud, for everything I learned from you, for having the flexibility to pursue my research interests, for showing the kind patience during some challenging and hard moments, for having given me the opportunity to integrate the world of research, for always supporting me, and for the frequent and inspiring discussions we had during my Ph.D. For the same reasons, sincerely thanks are also due to my co-adviser, Professor Paul Mycek, for the intensive discussions we had on the machine learning part, for always proposing new ideas, and for the kind hospitality when I visited CERFACS.

Thanks to my jury members, Professor Eric de Sturler, Professor Andreas Frommer, Professor Stéphane Lanteri, Professor Michael Bauerheim, Professor Jayant Sengupta, Professor Carola Kruse, and my advisers for serving on my Ph.D. jury, reviewing my thesis, and providing valuable feedback.

Thanks to my current and former colleagues at the COmposabilité Numerique and parallèle pour le CAlcul haute performanCE (CONCACE) project at Inria de l'université de Bordeaux for being open to collaboration and for having contributed to the excellent work atmosphere. I thought of Marek, Martina, Romain, Alena, Emmanuel, Olivier C., Pierre, Nick, Gilles and Florent, thank you all for the cheerful moments we had together in Coffee and Goûter Time, Baby Foot game, Apéro Plage, Secret Santa, Apéro Weekend, etc. For the same reasons, also thanks to my colleagues of the TOPAL project. I thought of Esragul, Mathieu V., Jean-Francois, Xunyi, Olivier B., Lionel, Mathieu F. and Abdou. Special thanks to my advisors and Carola, I am extremely proud of the work we have done together. For the same reason, also thanks to Matthieu and Professor Jing.

Thanks to my friends Li and Kun for the rich discussions. I would also like to thanks Yetao Wang for your numerous explanations and supports.

I would like to acknowledge the support of my family. Thank you, Mom, Dad and my younger sister. Thanks to my closest friend, Lingxin Luo, for your accompany, understanding, patience, support, and for always being interested in hearing about my research. I am also grateful to Van-Gogh, our 19 monthly feline companion, who brought us so many sweet moments with her naughty, tenacity, and spiritual independence.

Last but not least, I want to thank myself, for believing in me, for encouraging me, for never quitting, and for finishing this interesting work. I want to thank myself for trying to do more right than wrong. I want to thank myself for just being me at all times.

This research was supported by the joint-funding from the CERFACS and the Inria de l'université de Bordeaux Research Institute. Experiments presented in this work were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (refer to https://www.plafrim.fr).

Résumé étendu

Le cadre de recherche de cette thèse est la conception de techniques numériques efficaces pour des simulations frontières dans des applications académiques et industrielles complexes à grande échelle. Dans ce travail, nous nous concentrons principalement sur la résolution itérative de grands systèmes linéaires avec des second-membres multiples qui apparaissent dans diverses applications scientifiques telles que la chromodynamique quantique, la mécanique de la rupture, certaines études paramétriques, et certaines formulations de problèmes inverses, pour n'en citer que quelques-unes.

Les schémas itératifs modernes sont basés sur des méthodes de projection et font très souvent appel aux sous-espaces de Krylov. Lorsque un système linéaire avec second-membres multiples doit être résolu, les variantes par blocs des méthodes classiques de sous-espaces de Krylov sont les méthodes de choix. Les méthodes par blocs définissent l'espace de recherche comme la somme des sous-espaces de Krylov associés à chacun des p second-membres, de sorte qu'à chaque itération bloc, l'espace de recherche est élargi de p directions supplémentaires. D'un point de vue numérique, comme l'espace de recherche est la somme des espaces de Krylov individuels, la convergence devrait être au moins aussi rapide que la résolution indépendante de chaque second-membre. L'implémentation d'algorithmes par blocs permettant l'utilisation de noyaux de calcul efficaces de type BLAS-3, le temps de résolution devrait être réduit. Malheureusement, ces avantages potentiels se font au prix de nouvelles difficultés numériques induites par les taux de convergence éventuellement différents des second-membres ou de la combinaison linéaire de certains second-membres. Cette caractéristique numérique est appelée convergence partielle. Cette convergence partielle se traduit par une perte de rang dans le bloc de directions p à utiliser pour élargir l'espace de recherche. Des remèdes numériques appropriés doivent être conçus pour détecter et traiter cette éventuelle perte de rang ; d'abord pour éviter la mise en défaut du solveur itératif, ensuite pour réduire éventuellement l'effort de calcul dans les itérations suivantes en ajustant correctement la taille du bloc.

Une voie alternative aux approches traditionnelles d'algèbre linéaire numérique mentionnées ci-dessus consiste à considérer l'hybridation de méthddes classiques d'algèbre linéaire numérique avec des techniques d'apprentissage automatique [64, 75].

La première partie de ce manuscrit, composée de trois chapitres (chapitre 2-4), se concentre sur la résolution des problèmes de calcul associés aux méthodes itératives par blocs pour les séquences de systèmes linéaires à second-membres multiples. Dans le chapitre 2, pour les séquences de systèmes linéaires non symétriques, nous développons une nouvelle approche bloc basée sur la minimisation du résidu de norme minimale qui combine deux ingrédients principaux. Le premier composant exploite les idées de GCRO-DR [80], nous permettant de recycler l'information spectrale d'un système linéaire à l'autre. La deuxième composante est le mécanisme numérique de gestion de la convergence partielle,

appelé inexact breakdown dans IB-BGMRES [88]. Ce mécanisme permet de contrôler la déficience de rang dans la base de l'espace des résidus étendue par blocs. Ensuite, pour la classe des approches résiduelles à norme minimale par blocs, qui repose sur une égalité de type Arnoldi par blocs entre l'espace de recherche et l'espace résiduel (par ex. Ensuite, pour la classe des approches par blocs basées sur la minimisation du résidu de norme minimale, qui repose sur une égalité de type Arnoldi qui lie l'espace de recherche des solutions et l'espace des résidus associés (par exemple, toute variante GMRES ou GCRO par blocs), nous introduisons de nouvelles méthodes d'expansion de l'espace de recherche qui exploitent de nouveaux critères pour détecter la convergence partielle. Ces nouveaux critères de détection sont adaptés au critère d'arrêt choisi et au seuil de convergence ciblé pour garantir la qualité de la solution en termes d'erreur inverse. Cela nous permet de contrôler l'effort de calcul tout en assurant la précision finale de chaque solution individuelle. Puisque le problème de convergence partielle apparaît dans tous les types de méthodes de sous-espaces, dans le chapitre 3, nous étendons le mécanisme de détection de convergence partielle à la variante en bloc de la méthode des résidus conjugués [66], une approche de norme minimale des résidus basée sur un schéma de récurrence courte, pour les systèmes linéaires symétriques mais pas nécessairement définis positifs. Ensuite, inspiré par l'idée de réutiliser l'information générée pour approximer l'information spectrale afin d'accélérer la convergence, dans le chapitre 4, nous concevons une stratégie de recyclage de sous-espace. Plus précisément, nous appliquons les idées de "thick restart" [125] introduites dans la méthode de Lanczos pour le calcul des paires propres dans l'algorithme du gradient conjugué par blocs (BCG) [77] pour les systèmes linéaires symétriques définis positifs. Cette nouvelle variante raffine périodiquement l'information spectrale sur une petite fenêtre définie lors de la résolution des systèmes linéaires, ce qui fournit un moyen pratique d'approximer l'information spectrale avec un effort de calcul modéré. Nous étudions également la possibilité d'utiliser la détection de la convergence partielle, présentée dans le contexte de méthodes de norme minimale des résidus, pour l'algorithme BCG. Ce mécanisme numérique n'est qu'une heuristique car la stratégie d'expansion de l'espace de recherche qui en résulte, repose toujours sur la minimisation de la norme des résidus, alors qu'idéalement elle devrait être basée sur un principe de minimisation de l'erreur en norme A. Malheureusement, ces directions d'erreur en norme A ne sont pas un sous-produit du BCG comme les résidus pour les méthodes par blocs considérées dans les chapitres précédents. Enfin, ces stratégies de détection de convergence partielle et de recyclage du sous-espace peuvent être combinées efficacement pour concevoir un algorithme de gradient conjugué par blocs déflatés pour des séquences de systèmes linéaires symétriques définis positifs. Des expériences numériques sont présentées pour illustrer les caractéristiques numériques et informatiques de tous ces nouveaux solveurs de Krylov par blocs.

La deuxième partie, composée du chapitre 5, présente certaines manières d'hybrider les solveurs par apprentissage profond nouvellement émergents et des techniques d'algèbre linéaire numérique plus traditionnelles afin de les faire bénéficier les uns des autres. Les techniques d'apprentissage profond ont deux ingrédients principaux. Le premier est la phase d'entrainement, qui est essentiellement une procédure d'optimisation permettant de réduire une fonction coût en ajustant les paramètres internes des couches du réseau de neurones. La deuxième phase, où le réseau entraîné, est utilisé pour prédire la solution d'un ensemble de données d'entrée, qui n'a jamais été vue lors de sa phase d'apprentissage. Dans le contexte de la résolution d'équations de Helmholtz 2D hétérogène,

nous nous concentrons d'abord sur l'introduction de nouveaux ingrédients mathématiques des solveurs itératifs classiques dans la phase d'entraînement d'un solveur récemment proposé, qui est basé sur un réseau profond de neurones . Le principal avantage est une amélioration significative de la phase d'entraînement , plus robuste et plus rapide, qui s'avère également applicable aux processus de validation. En outre, une fois que ces réseaux ont été correctement entraînés, leurs inférences peuvent être appliquées comme des préconditionnements non linéaires dans les méthodes traditionnelles GMRES et FOM flexibles. Ce chapitre démontre que ces variantes hybrides présentent de nets avantages par rapport à l'approche nouvellement émergente des réseaux neuronaux profonds et au solveur itératif classique de Krylov, tant en termes de coût de calcul que de précision de la solution calculée.

Enfin, nous tirons quelques conclusions de ces travaux de thèse et énumérons quelques piste de perspectives supplémentaires.

La partie I développe des nouvelles variantes des méthodes de sous-espace de Krylov par blocs pour les systèmes linéaires à second-membres multiples. Pour le chapitre 2, lsuivant les idées de [78], je proposerais d'analyser la stabilité des techniques de norme minimale des résidus par blocs dans un contexte d'analyse inverse des erreurs Pour les algorithmes à récurrence courte tels que BCR et BCG décrits dans le chapitre 3-4, certains écarts résiduels existent parfois empêchant la convergence de l'erreur inverse réelle, c'est-à-dire basée sur la norme des vrais résidus, alors que celle basée sur la norme des résidus itérés indique la convergence. Un remède possible pourrait être d'étendre l'analyse de l'erreur d'arrondi de [120] au cas des méthodes blocs pour estimer l'écart des résidus et éventuellement concevoir des techniques de remplacement correspondantes [16, 25]. En outre, lorsque l'on considère la stratégie de recyclage ou de déflation du sous-espace, une autre direction intéressante consiste à trouver des informations générales pour guider le choix de certains paramètres d'ajustement, comme la longueur maximale d'un cycle de raffinage, la dimension de l'espace de déflation et le nombre de second-membres. Les techniques de minimisation des normes des résidus par blocs pour les cas non symétriques décrites dans le chapitre 2 ont été intégrées dans une pile logicielle de notre équipe, à savoir *Fabulous* [1], grâce à une collaboration étroite avec un ingénieur responsable de cette bibliothèque. Fabulous est une bibliothèque C++ complète qui implémente divers solveurs de Krylov par bloc pour la résolution de systèmes linéaires. Cette intégration permet l'utilisation du solveur proposé dans le chapitre 2 dans diverses applications telles que le calcul de QCD dans le cadre du projet H2020 PRACE-6IP. Afin d'évaluer pleinement les performances de calcul des solveurs blocs présentés dans les chapitres 3 et 4, il serait intéressant de mener un tel effort d'ingénierie pour favoriser le transfert des connaissances de l'algèbre linéaire numérique vers les applications à grande échelle.

La partie II discute de deux améliorations principales dans l'hybridation des techniques d'apprentissage automatique et des solveurs d'algèbre linéaire numérique pour la résolution de systèmes linéaires avec un seul second-membre. Une direction future directe de la partie II est d'explorer l'équilibre entre la garantie d'une meilleure précision atteignable et la bonne généralisation du réseau entraîné. D'autres travaux connexes tentent d'éclairer les boîtes noires lors du développement de solveurs SciML (scientific machine learning) [3,65,118], comme le choix de l'architecture du réseau neuronal profond pour un problème physique spécifique (comme les opérateurs neuronaux de Fourier [41]),

[1] https ://gitlab.inria.fr/solverstack/fabulous/

la définition de la fonction coût basée sur un contexte physique pratique (comme les travaux récents [129]), la définition (automatique) du taux d'apprentissage qui décroît exponentiellement avec l'indice d'époque, le réglage et le test d'autres hyperparamètres (comme le choix d'un autre optimiseur, le nombre de couches cachées et de neurones par couche cachée), les moyens possibles pour résoudre le problème du gradient évanescent [9, 37, 43], le pré-traitement des ensembles de données, la conception d'autres stratégies d'hybridation et l'étude de leurs effets, etc. Pour avoir une bonne compréhension de ces options, j'envisage d'essayer d'autres modèles d'apprentissage automatique pour diverses applications. Je veux découvrir les relations composition-structure-propriété pour différents problèmes de calcul et solveurs SciML, puis les utiliser pour déterminer quels sont les modèles SciML optimaux (en termes de précision, de vitesse de convergence, d'architecture de réseau neuronal, de coût de calcul, etc. Avec des descripteurs plus rapides, j'espère créer un système théorique catégorisé pour différents problèmes de calcul SciML. Un autre problème qui m'intéresse est de trouver un moyen de réduire les coûts de l'entraînement. L'un des moyens d'y parvenir pourrait être de pré-traiter les ensembles de données ou d'introduire des informations mathématiques et physiques lors de la conception d'un solveur de réseau neuronal avec un schéma de solution cible. Un autre moyen pourrait consister à se concentrer sur le développement de nouveaux modèles d'apprentissage automatique creux à grande échelle en utilisant les connaissances d'un cadre multidisciplinaire, tel que la théorie des graphes et l'informatique, ce qui pourrait également aider les régions d'apprentissage automatique pour d'autres applications hors du champ d'application de SciML, mais qui peuvent également être assez difficiles.

Extended summary

The research framework of this thesis is the design of numerical techniques useful for performing efficiently frontier simulations arising from academic and industrial large scale, challenging, applications. In this work, we mostly focus on the iterative solution of large linear systems with multiple right-hand sides that appear in various scientific applications such as quantum chromodynamics, finite element fracture mechanics, some parametric studies, and some formulations of inverse problems, to name a few.

Modern iterative schemes are based on projection methods and very often involve Krylov subpaces. When p multiple right-hand sides have to be solved, block variants of classical Krylov subspace methods are the methods of choice. The block methods define the search space as the sum of the Krylov subspaces associated with each individual right-hand side, so that at each iteration the search space is enlarged by p additional directions. From a numerical point of view, because the search space contains the individual Krylov spaces the convergence should be at least as fast as solving independently each right-hand side. Because the block algorithm implementation enables the use of efficient BLAS-3 like computational kernels, the time to the solution is expected to be reduced. Unfortunately, these potential advantages come at the price of novel numerical difficulties induced by the possibly different convergence rates of the right-hand sides or linear combination of some right-hand sides. This numerical feature is referred to as partial convergence. This partial convergence translates in rank deficiency within the block of p directions to be used to enlarge the search space. Appropriate numerical remedies should be designed to detect and treat this possible rank deficiency; first to avoid a possible breakdown of the iterative solver, second to possibly reduce the computational effort in the next iterations by properly adjusting the block size.

An alternative path to the above mentioned traditional numerical linear algebra approaches is to consider hybridizing numerical linear algebra with scientific machine learning techniques [64, 75].

The first main part, composed of three chapters (i.e., Chapter 2-4), focuses on addressing computational challenges associated with the block iterative Krylov subspace methods for sequences of linear systems with multiple right-hand sides. In Chapter 2, for sequences of unsymmetric linear systems, we develop a new block minimum norm residual approach that combines two main ingredients. The first component exploits ideas from GCRO-DR [80], enabling us to recycle spectral information from one linear system to the next. The second component is the numerical mechanism for managing the partial convergence of the right-hand sides, referred to as the inexact breakdown detection mechanism in IB-BGMRES [88], that enables the monitoring of the rank deficiency in the residual space basis expanded blockwise. Next, for the class of block minimum norm residual approaches, that relies on a block Arnoldi-like equality between the search space and the residual space (e.g., any block GMRES or block GCRO variants), we

introduce new search space expansion policies defined on novel criteria to detect the partial convergence. These novel detection criteria are tuned to the selected stopping criterion and targeted convergence threshold to best cope with the selected normwise backward error. This enables us to monitor the computational effort while ensuring the final accuracy of each individual solution. Because the partial convergence issue appears in any type of block Krylov solvers, in Chapter 3, we extend the partial convergence detecting mechanism to the block variant of the conjugate residual method [66], a minimum residual norm approach with short term recurrence scheme, for symmetric but not necessary positive definite linear systems. Then, inspired by the idea of reusing generated information to approximate spectral information for accelerating convergence, in Chapter 4, we devise a corresponding subspace recycling strategy. More precisely, we apply the thick-restart ideas [125] introduced in the Lanczos method for eigenpair calculation in the block conjugate gradient algorithm (BCG) [77] for the symmetric positive definite linear systems. This new variant periodically refines the spectral information on a small window defined when solving the linear systems, which provides a practical way to approximate spectral information with a moderate computational effort. We also investigate the possibility to use the partial convergence detection presented in the minimum residual norm context to the BCG algorithm as a heuristic. This numerical mechanism is only a heuristic because the resulting search space expansion policy still relies on residual norm minimization, while ideally it should be based on an A-norm error minimization principle. Unfortunately, this A-norm error directions are not a by-product of BCG as the residuals for the numerical block methods considered in the previous chapters. Finally, these partial convergence detection and subspace recycling strategies can be efficiently combined to design a deflated block conjugate gradient algorithm for sequences of symmetric positive definite linear systems. Numerical experiments are reported to illustrate the numerical and computational features of all these new block Krylov solvers.

The second part that is composed by Chapter 5, presents some ways of hybridizing the newly emerging deep learning solvers and the more traditional numerical linear algebra techniques to let them benefit from each other. Deep learning techniques have two main ingredients. The first one is the training phase, that is essentially an optimization procedure enabling to reduce a loss function by adjusting the internal parameters of the neural network layers. The second phase, where the trained network is used to infer the solution for a given input data set, which is never seen during its previous training phase. In the context of the solution of a heterogeneous 2D Helmholtz equation, we first focus on introducing some mathematical ingredients from classical iterative solvers into the training phase of a recently proposed deep neural network solver. The main benefit is a significant improvement in the training phase that is more robust and faster, which turns out to be applicable to the testing processes as well. Furthermore, once these network solvers have been properly trained, their inferences can be applied as a nonlinear preconditioner in the traditional flexible GMRES and flexible FOM methods. This chapter demonstrates that these hybrid variants have clear advantages over both the newly emerging deep neural network approach and the classical iterative Krylov solver in terms of both computational cost and accuracy of the computed solution.

Part I

Block Krylov solvers for the solution of sequences of linear systems with multiple right-hand sides

Chapter 1

Context and notations

Many scientific and industrial simulations require the solution of a sequence of linear systems with multiple right-hand sides and possibly slowly changing left-hand sides. In that context, one has to solve a series of linear systems of the form

$$A^{(\ell)}X^{(\ell)} = B^{(\ell)}, \quad \ell = 1, 2, \ldots, \tag{1.1}$$

where, associated with the ℓth family, $A^{(\ell)} \in \mathbb{C}^{n\times n}$ is a square nonsingular matrix of large dimension n along the family index ℓ, $B^{(\ell)} = [b^{(\ell,1)}, b^{(\ell,2)}, \ldots, b^{(\ell,p^{(\ell)})}] \in \mathbb{C}^{n\times p^{(\ell)}}$ are simultaneously given right-hand sides of full rank with $p^{(\ell)} \ll n$, and $X^{(\ell)} = [x^{(\ell,1)}, x^{(\ell,2)}, \ldots, x^{(\ell,p^{(\ell)})}] \in \mathbb{C}^{n\times p^{(\ell)}}$ are the solutions to be computed. Both the coefficient matrix $A^{(\ell)}$ and right-hand sides $B^{(\ell)}$ change from one family to the next, and the families of linear systems are typically available in sequence.

The symbol $||\cdot||$ denotes the Euclidean norm default for both vectors and matrices, and the Frobenius norm is denoted with the subscript $_F$. The superscript ^{H} denotes the transpose conjugate and ^{T} stands for transpose. The notation $\mathbb{C}$ and $\mathbb{R}$ respectively refer to the complex and real number field. Because much notation is involved, we make certain choices to improve the readability of the chapter. The vectors are denoted by lowercase letters; matrices with multiple columns are described by uppercase letters; calligraphic uppercase letters, e.g., $\mathscr{V}$ represent matrices whose columns are augmented by multiple columns at each iteration (as commonly appearing in the block Krylov context); and uppercase blackboard bold letters, e.g., $\mathbb{V}$ refer to the block Krylov basis generated at each iteration. The superscript † refers to the Moore-Penrose inverse. For convenience of the algorithm illustration and presentation, some MATLAB notation is used. Without special note, a subscript $_j$ for a vector (in the single right-hand case) or a matrix (in the block case) is used to indicate that the vector or matrix is obtained at iteration j, and a positive subscript integer $_m$ represents the maximal iteration number of each (block) Krylov cycle. All the involved recycling subspaces of dimension k are described as a matrix with the subscript $_k$, whose columns form a basis. A matrix $C \in \mathbb{C}^{m\times\ell}$ consisting of m rows and ℓ columns sometimes is denoted as $C_{m\times\ell}$ explicitly. The identity and null matrices of dimension m are denoted, respectively, by I_m and 0_m or by just I and 0 when the dimension is evident from the context. For a matrix $C \in \mathbb{C}^{m\times\ell}$, the singular values of C are denoted by $\sigma_1(C) \geq \ldots \geq \sigma_{min(m,\ell)}(C)$ in descending order; furthermore, we denote by $\mathrm{span}(C)$ the space spanned by the columns of C.

Chapter 2

The unsymmetric case with minimum residual norm techniques

2.1 Introduction

When solving sequences of linear systems such as Equation (1.1), attractive approaches are those that can exploit information generated during the solution of a given system to accelerate the convergence for the next systems. Deflated restarting implements a similar idea between the cycles in the generalized minimum residual (GMRES) norm method [86, 93, 114]; it is realized by using a deflation subspace containing a few approximate eigenvectors deemed to hamper the convergence of the Krylov subspace methods [71–73]. An alternative technique is the subspace recycling strategy proposed in the generalized conjugate residual with inner orthogonalization (GCRO) method and deflated restarting (GCRO-DR) method [80]. This latter method can reuse information accumulated in previous cycles as well as that accumulated during the solution of the previous families. We refer to [104] for a recent survey of subspace recycling methods. Because the multiple right-hand sides of Equation (1.1) are simultaneously available, block Krylov subspace methods are often considered as suitable candidates because of their capability of sharing search subspaces that can be generated using basic linear algebra subprograms, such as level 3 BLAS-like implementation [42] that is expected to reduce the time for solution. Unfortunately these potential advantages come at the price of a common issue in block Krylov subspace methods, the rank deficiency that might appear during the expansion of the residual spaces, which is caused by the convergence of some individual solution or a linear combination of solution vectors. Such a rank deficiency problem could cause the block Arnoldi process to break down before the solutions for all the right-hand sides are found. For the sake of balancing robustness and convergence rate, Robbé and Sadkane proposed an inexact breakdown detection mechanism for the block GMRES algorithm (denoted by IB-BGMRES) [88], which could keep and reintroduce directions associated with the almost converged parts in next iteration if necessary. We refer the reader to [6, 15, 88] for relevant works on inexact breakdown detection, as well as to [34, 110–113, 126] for related variants of block Krylov subspace methods for solving linear systems with multiple right-hand sides.

The contribution of this chapter is twofold. We first show how to combine subspace recycling techniques of GCRO-DR [80], for recycling spectral information at a new cycle/family, with the inexact breakdown detection introduced by Robbé and Sadkane in IB-BGMRES [88], for handling almost rank deficient blocks generated by the block Arnoldi procedure, to develop the IB-BGCRO-DR algorithm, a new recycling block GCRO-DR variant with partial convergence detection. This is a natural extension of a previous work on IB-BGMRES-DR [6], that enables the deflated restarting strategy proposed by Morgan [73] to be applied only at restart but not when solving a sequence of linear systems. The IB-BGCRO-DR method can reuse spectral information from solutions in both the previous cycles and families thus showing obvious advantages when solving sequences of linear systems like Equation (1.1). In addition, we propose a flexible counterpart of the new algorithm, which allows the use of a mixed arithmetic computation where all steps are computed with a selected working precision except for the preconditioner which is performed with a reduced precision. The second contribution is related to the block search space expansion policies that can be further developed based on the partial convergence detection. In particular, we introduce new search space expansion policies defined on novel criteria to detect the partial convergence. These novel detection criteria are tuned to the selected stopping criterion and targeted convergence threshold to best cope with the selected normwise backward error. This enables us to monitor the computational effort while ensuring the final accuracy of each individual solution.

The remainder of this chapter is organized as follows. Section 2.2 is devoted to the development of the new algorithm and contains some background that enables us to introduce the various numerical ingredients and notation required to design our algorithm. In Section 2.2.1 we first recall the governing ideas of the minimum norm residual Krylov method GCRO in a single right-hand side setting, and in Section 2.2.2 we briefly present its block variant. Next, in Section 2.2.3 we present how the original inexact breakdown detection mechanism [88] introduced for block GMRES can be applied to block GCRO as well. These two main ingredients are combined to develop the new algorithm IB-BGCRO-DR in Section 2.2.4 and its flexible preconditioning variant referred to as IB-BFGCRO-DR in Section 2.2.6. In Section 2.3, we describe how to extend the original inexact breakdown detection mechanism to best adapt the computational effort and reach the targeted accuracy prescribed by the stopping criterion defined in terms of normwise backward errors for the individual solutions. In particular, we derive strategies for managing the situation where the different right-hand sides need to be solved with different convergence thresholds. We also present policies adapted to a stopping criterion based on normwise backward error on the right-hand side only (i.e., classical residual norm scaled by the norm of the right-hand side) or the more general one used to establish the backward stability of GMRES [78]. Section 2.4 presents some detailed remarks on computational and algorithmic aspects; the associated pseudocode of the IB-BGCRO-DR algorithm is presented as well. In Section 2.5, we extend the previous flexible preconditioning techniques to IB-BGMRES-DR, which resulting new algorithm named IB-BFGMRES-DR. In Section 2.6 we present numerical experiments that illustrate the benefits of the new algorithm with both constant and slowly varying successive linear systems with multiple right-hand sides, and we introduce as well the numerical capabilities of the novel search space expansion policies. Finally, we conclude with some detailed remarks in Section 2.7.

For simplicity and notational convenience, in the rest of this chapter we drop the superscript $^{(\ell)}$ in $B^{(\ell)}$ and $X^{(\ell)}$ whenever we consider solving the current ℓth family of

linear systems in the entire sequence of families. We indicate the superscript for a family order explicitly when necessary. That is, suppose that the current ℓth family of linear systems to be solved is

$$AX = B, \tag{2.1}$$

where, $A \in \mathbb{C}^{n\times n}$ is the current square nonsingular matrix of dimension n, $B = [b^{(1)}, b^{(2)}, \ldots, b^{(p)}] \in \mathbb{C}^{n\times p}$ are the right-hand sides given simultaneously, and $X = [x^{(1)}, x^{(2)}, \ldots, x^{(p)}] \in \mathbb{C}^{n\times p}$ are the solutions to be computed.

2.2 Block GCRO-DR with partial convergence detection

For the sake of completeness, this section contains some (possibly well-known) background which enables us to introduce the notation required to describe the new algorithm and detail its properties. In that respect, we first recall the main ingredients of the subspace recycling techniques existing in the minimum residual Krylov methods GCRO [28] and GCRO-DR [80] that are presented in the single right-hand side context. Next, we introduce the straightforward extension to the multiple right-hand sides framework, that is the block formulation of GCRO-DR (BGCRO-DR) [79, 81]. Then the driving ideas of partial convergence detection [88], along with the corresponding block Arnoldi-like recurrence equation, are derived in the block GCRO-DR context, leading to the new IB-BGCRO-DR algorithm.

2.2.1 GCRO

The background of GCRO [28] is briefly reviewed first in the case of a single right-hand side and then extended to the block case. The GCRO method relies on a given full-rank matrix $U_k \in \mathbb{C}^{n\times k}$, and on a matrix C_k as the image of U_k by A satisfying the relations

$$AU_k = C_k, \tag{2.2}$$
$$C_k^H C_k = I_k. \tag{2.3}$$

For the solution of a single right-hand side linear system $Ax = b$ and a given initial guess x_0, the governing idea is to first define $x_1 \in x_0 + \mathrm{Range}(U_k)$ that minimizes the residual norm. From x_1 and its associated residual r_1, Arnoldi iterations are performed to enlarge the nested orthonormal basis of the residual spaces. The vector

$$x_1 = \underset{x\in x_0+\mathrm{Range}(U_k)}{\operatorname{argmin}} ||b - Ax||,$$

is defined by

$$x_1 = x_0 + U_k C_k^H r_0, \text{ and } r_1 = (I - C_k C_k^H) r_0 \text{ such that } r_1 \in C_k^{\perp}. \tag{2.4}$$

Starting from the unit vector $v_1 = r_1/\|r_1\|$, the Arnoldi procedure enables us to form an orthonormal basis $V_m = [v_1, \ldots, v_m]$ of the Krylov space $\mathcal{K}_m((I - C_k C_k^H)A, v_1) = \mathrm{span}(v_1, (I - C_k C_k^H)Av_1, \ldots, ((I - C_k C_k^H)A)^{m-1}v_1)$ yielding an Arnoldi-like relation in

the matrix form as

$$(I - C_k C_k^H) A V_m = V_{m+1} \underline{H}_m, \tag{2.5}$$

where the top square part of $\underline{H}_m \in \mathbb{C}^{(m+1)\times m}$ is upper Hessenberg, and only the last entry of its last row is nonzero. Combining (2.2) and (2.5) into one matrix form allows us to write a relation quite similar to an Arnoldi equality that reads

$$A\widehat{W}_m = \widehat{V}_{m+1} \underline{G}_m,$$

where the columns of $\widehat{W}_m = [U_k, V_m]$ define a basis of the search space, columns of $\widehat{V}_{m+1} = [C_k, V_{m+1}]$ make up an orthonormal basis of the residual space and $\underline{G}_m = \begin{bmatrix} I_k & B_m \\ 0_{(m+1)\times k} & \underline{H}_m \end{bmatrix} \in \mathbb{C}^{(k+m+1)\times(k+m)}$, with $\widehat{V}_{m+1}^H \widehat{V}_{m+1} = I_{m+1}$ and $B_m = C_k^H A V_m$. The minimum residual norm solution in the affine space $x_1 + \text{Range}(\widehat{W}_m)$ can be written as $x_m = x_1 + \widehat{W}_m y_m$, where

$$y_m = \underset{y \in \mathbb{C}^{k+m}}{\text{argmin}} \, \|c - \underline{G}_m y\|$$

and $c = \widehat{V}_{m+1}^H r_1 = (0_k, \|r_1\|, 0_m)^T \in \mathbb{C}^{k+m+1}$ are the components of the residual associated with x_1 in the residual space spanned by the columns of $\widehat{V}_{m+1}$.

GCRO and GMRES [93] both belong to the family of residual norm minimization approaches and rely on an orthonormal basis of the residual space. In addition to sharing the Arnoldi procedure to form part or all of this basis, they also share the property of "happy breakdown"; that is, if the search space cannot be enlarged because the new direction computed by the Arnoldi process is the null vector, and if the r_1 in (2.4) satisfies $r_1/\|r_1\| \in \text{Range}((I - C_k C_k^H) A V_m)$, then the solution is exactly found in the search space [28, Definition 2.4]. This sharing of features extends to the block context for the solution of linear systems with multiple right-hand sides; in particular, the inexact breakdown principle introduced in [88] in the context of block GMRES can be extended to block GCRO, as discussed in what follows. The purpose of the partial convergence detection is to prevent, in an elegant and effective way, the loss of numerical rank of the search space basis; this turns out to be also a way to monitor the search space expansion according to the final target accuracy.

2.2.2 Block GCRO

The straightforward extension of the GCRO method in the block context is briefly described below. To facilitate reading, we do not use the calligraphic form in the notation but keep the same letters to denote the block counterparts of the quantities involved in the method. Starting from the block initial guess $X_0 = [x_0^{(1)}, x_0^{(2)}, \ldots, x_0^{(p)}] \in \mathbb{C}^{n\times p}$ and associated initial residual block $R_0 = B - AX_0$, one can define

$$X_1 = \underset{X \in X_0 + \text{Range}(U_k)}{\text{argmin}} \, ||B - AX||_F,$$

given by

$$X_1 = X_0 + U_k C_k^H R_0, \text{ and } R_1 = (I - C_k C_k^H) R_0 \text{ such that } R_1 \in C_k^{\perp}. \tag{2.6}$$

For the sake of simplicity, we first assume that R_1 is of full rank and denote $R_1 = \mathbb{V}_1\Lambda_1$ as its reduced QR-factorization. The orthonormal block $\mathbb{V}_1$ is then used to build the search space via m steps of the block Arnoldi procedure depicted in Algorithm 1, to generate $\mathscr{V}_m = [\mathbb{V}_1, \ldots, \mathbb{V}_m]$, whose columns form an orthonormal basis of $\mathcal{K}_m((I - C_kC_k^H)A, \mathbb{V}_1) = \bigoplus_{t=1}^{p} \mathcal{K}_m((I - C_kC_k^H)A, v_1^{(t)})$. The block Arnoldi procedure leads to the matrix equality

Algorithm 1 *Block Arnoldi procedure with deflation of the C_k space*

1: Given a nonsingular coefficient matrix $A \in \mathbb{C}^{n\times n}$, choose a matrix $\mathbb{V}_1 \in \mathbb{C}^{n\times p}$ with orthonormal columns
2: **for** $j = 1, 2, \ldots, m$ **do**
3: Compute $W_j = (I - C_kC_k^H)A\mathbb{V}_j$
4: **for** $i = 1, 2, \ldots, j$ **do**
5: $H_{i,j} = \mathbb{V}_i^H W_j$
6: $W_j = W_j - \mathbb{V}_i H_{i,j}$
7: **end for**
8: $W_j = \mathbb{V}_{j+1}H_{j+1,j}$ (reduced QR-factorization of W_j)
9: **end for**

$$(I - C_kC_k^H)A\mathscr{V}_m = \mathscr{V}_{m+1}\underline{\mathscr{H}}_m, \tag{2.7}$$

where $\underline{\mathscr{H}}_m$ is a block Hessenberg matrix with (i, j) block defined by $H_{i,j}$. Similarly to the single right-hand side case, (2.2) and (2.7) can be gathered into matrix form

$$A\widehat{\mathscr{W}_m} = \widehat{\mathscr{V}}_{m+1}\underline{\mathcal{G}}_m, \tag{2.8}$$

where $\widehat{\mathscr{W}_m} = [U_k, \mathscr{V}_m] \in \mathbb{C}^{n\times(k+mp)}$, $\widehat{\mathscr{V}}_{m+1} = [C_k, \mathscr{V}_{m+1}] \in \mathbb{C}^{n\times(k+(m+1)p)}$, and $\underline{\mathcal{G}}_m = \begin{bmatrix} I_k & \mathcal{B}_m \\ 0_{(m+1)p\times k} & \underline{\mathscr{H}}_m \end{bmatrix} = \begin{bmatrix} \mathcal{G}_m & \\ 0_{p\times(k+(m-1)p)} & H_{m+1,m} \end{bmatrix} \in \mathbb{C}^{(k+(m+1)p)\times(k+mp)}$ with $\widehat{\mathscr{V}}_{m+1}^H\widehat{\mathscr{V}}_{m+1} = I_{k+(m+1)p}$ and $\mathcal{B}_m = C_k^HA\mathscr{V}_m \in \mathbb{C}^{k\times mp}$; here $mp = m \times p$. The minimum residual norm solution in the affine space $X_1 + \mathrm{Range}(\widehat{W}_m)$ can be written as $X_m = X_1 + \widehat{\mathscr{W}_m}Y_m$, where

$$Y_m = \underset{Y\in\mathbb{C}^{(k+mp)\times p}}{\mathrm{argmin}} \|\mathcal{C} - \underline{\mathcal{G}}_m Y\|_F,$$

$\mathcal{C} = \widehat{\mathscr{V}}_{m+1}^H R_1 = (0_{p\times k}, \Lambda_1^T, 0_{p\times mp})^T \in \mathbb{C}^{(k+(m+1)p)\times p}$, and the columns of $\mathcal{C}$ are the components of the initial residual block R_1 in the residual space $\widehat{\mathscr{V}}_{m+1}$.

2.2.3 Block GCRO with partial convergence detection

When one solution or a linear combination of solutions has converged, the block Arnoldi procedure implemented to build an orthonormal basis of $\mathcal{K}_j((I - C_kC_k^H)A, \mathbb{V}_1)$ needs to be modified to account for this partial convergence. This partial convergence is characterized by a numerical rank deficiency in the new p directions that are usually introduced for enlarging the search space at the next iteration. In [88], the authors present an elegant numerical variant that enables the detection of what is referred to as inexact breakdowns. In that approach the directions that have a low contribution to the

residual block are discarded from the candidate set of vectors used to expand the search space at the next iteration, but these directions are reintroduced in iterations afterward if necessary. In this section, we try to give an insight and the main equality required to derive the IB-BGCRO-DR algorithm. We refer the reader to the original paper [88] for a detailed and complete description. For the sake of simplicity and easy cross reference, we adopt most of the notation from [6, 88].

When a partial convergence occurs, not all the space spanned by W_j is considered to build $\mathbb{V}_{j+1}$ in order to expand the search space. For the sake of simplicity, we assume that $p_1 = p$ and we denote by p_{j+1} the number of columns of the block orthonormal basis vector $\mathbb{V}_{j+1}$. Then $\mathbb{V}_{j+1} \in \mathbb{C}^{n\times p_{j+1}}, W_j \in \mathbb{C}^{n\times p_j}$ and $H_{j+1,j} \in \mathbb{C}^{p_{j+1}\times p_j}$. As a consequence the dimension of the search space $\mathcal{K}_j((I - C_kC_k^H)A, \mathbb{V}_1)$ considered at the jth iteration is no longer necessarily equal to $j \times p$ but is equal to $n_j = \sum_{i=1}^{j} p_i$, that is, the sum of the column rank of the matrices $\mathbb{V}_i$ $(i = 1, \ldots, j)$.

When no partial convergence has occurred, that is, $p_{j+1} = p_j = \cdots = p_1 = p$, the range of W_j has always been used to enlarge the search space and we obtain the block relation given by (2.8). To account for a numerical deficiency in the residual block $R_j = B - AX_j$, Robbé and Sadkane [88] proposed splitting

$$W_j = \mathbb{V}_{j+1}H_{j+1,j} + Q_j \tag{2.9}$$

such that the columns of Q_j and $\mathbb{V}_{j+1}$ are orthogonal to each other and only $\mathbb{V}_{j+1}$ is used to enlarge $\mathscr{V}_j$ to form $\mathscr{V}_{j+1}$. We can then extend (2.8) into

$$A\widehat{\mathscr{W}_j} = \widehat{\mathscr{V}_j}\mathcal{G}_j + [0_{n\times k},\ \mathcal{Q}_{j-1},\ W_j], \tag{2.10}$$

where $\mathcal{G}_j \in \mathbb{C}^{(k+n_j)\times(k+n_j)}$ is the first $k + n_j$ rows of $\underline{\mathcal{G}}_j \in \mathbb{C}^{(k+n_j+p)\times(k+n_j)}$, $\mathcal{Q}_{j-1} = [Q_1, \ldots, Q_{j-1}] \in \mathbb{C}^{n\times n_{j-1}}$ accounts for all the discarded directions. The matrix $\mathcal{Q}_{j-1}$ is rank deficient, and it reduces to the zero matrix of $\mathbb{C}^{n\times n_{j-1}}$ as long as no partial convergence has occurred.

In order to characterize a minimum norm solution in the space spanned by $\widehat{\mathscr{W}_j}$ using (2.10) we need to form an orthonormal basis of the space spanned by $[\widehat{\mathscr{V}_j}, \mathcal{Q}_{j-1}, W_j]$. This is performed by first orthogonalizing $\mathcal{Q}_{j-1}$ against $\widehat{\mathscr{V}_j}$, that is $\widetilde{\mathcal{Q}}_{j-1} = (I - \widehat{\mathscr{V}_j}\widehat{\mathscr{V}_j}^H)\mathcal{Q}_{j-1}$. Because $\mathcal{Q}_{j-1}$ is of rank deficiency, so is $\widetilde{\mathcal{Q}}_{j-1}$, which can be written as

$$\widetilde{\mathcal{Q}}_{j-1} = P_{j-1}\mathbb{G}_{j-1} \text{ with } \begin{cases} P_{j-1} \in \mathbb{C}^{n\times q_j} \text{ has orthonormal columns with } \widehat{\mathscr{V}_j}^H P_{j-1} = 0, \\ \mathbb{G}_{j-1} \in \mathbb{C}^{q_j\times n_{j-1}} \text{ is of full rank with } q_j = p - p_j. \end{cases} \tag{2.11}$$

Next, W_j that is already orthogonal to $\widehat{\mathscr{V}_j}$ is made to be orthogonal to P_{j-1} with $W_j - P_{j-1}E_j$ where $E_j = P_{j-1}^H W_j$; then one computes $\widetilde{W}_j D_j$ with $\widetilde{W}_j \in \mathbb{C}^{n\times p_j}$ and $D_j \in \mathbb{C}^{p_j\times p_j}$ by carrying out the reduced QR-factorization of the tall and skinny matrix $W_j - P_{j-1}E_j$. Eventually, the columns of the matrix $[\widehat{\mathscr{V}_j}, P_{j-1}, \widetilde{W}_j]$ form an orthonormal basis of the residual space spanned by $[\widehat{\mathscr{V}_j}, \mathcal{Q}_{j-1}, W_j]$.

With this new basis, (2.10) reads

$$\begin{aligned} A[U_k, \mathscr{V}_j] = \; & [C_k, \mathscr{V}_j] \begin{bmatrix} I & \mathcal{B}_j \\ 0 & \mathscr{L}_j \end{bmatrix} + \left[0_k, P_{j-1}\mathbb{G}_{j-1}, \left[P_{j-1}, \widetilde{W}_j\right] \begin{bmatrix} E_j \\ D_j \end{bmatrix}\right] \\ = \; & \left[C_k, \mathscr{V}_j, [P_{j-1}, \widetilde{W}_j]\right] \begin{bmatrix} I_k & & \mathcal{B}_j \\ & & \mathscr{L}_j \\ 0_{(n_j+p)\times k} & \mathbb{G}_{j-1} & E_j \\ & 0 & D_j \end{bmatrix}, \end{aligned} \tag{2.12}$$

where $\mathscr{L}_j = \begin{bmatrix} H_{1,1} & H_{1,2} & H_{1,3} & \cdots & H_{1,j} \\ H_{2,1} & H_{2,2} & H_{2,3} & \cdots & H_{2,j} \\ \mathbb{V}_3^H Q_1 & H_{3,2} & H_{3,3} & \cdots & H_{3,j} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbb{V}_j^H Q_1 & \cdots & \mathbb{V}_j^H Q_{j-2} & H_{j,j-1} & H_{j,j} \end{bmatrix} \in \mathbb{C}^{n_j \times n_j}$ is no longer upper Hessenberg as soon as one partial convergence occurs, i.e., $\exists \ell$, s.t., $Q_\ell \neq 0$.

Equation (2.12) can be rewritten in a more compact form as

$$A[U_k, \mathscr{V}_j] = \left[C_k, \mathscr{V}_j, [P_{j-1}, \widetilde{W}_j]\right] \underline{\mathscr{F}}_j,$$

so that the least squares problem to be solved to compute the minimum residual norm solution associated with the generalized Arnoldi relation (2.12) becomes

$$Y_j = \underset{Y \in \mathbb{C}^{(k+n_j)\times p}}{\operatorname{argmin}} \left\| \Lambda_j - \underline{\mathscr{F}}_j Y \right\|_F, \tag{2.13}$$

with

$$\underline{\mathscr{F}}_j = \begin{bmatrix} I_k & & \mathcal{B}_j \\ & & \mathscr{L}_j \\ 0_{(n_j+p)\times k} & \mathbb{G}_{j-1} & E_j \\ & 0 & D_j \end{bmatrix} = \begin{bmatrix} \mathscr{F}_j \\ \mathbb{H}_j \end{bmatrix} \in \mathbb{C}^{(k+n_j+p)\times(k+n_j)} \tag{2.14}$$

and $\Lambda_j = \begin{bmatrix} 0_{k\times p} \\ \Lambda_1 \\ 0_{n_j \times p} \end{bmatrix} \in \mathbb{C}^{(k+n_j+p)\times p}$, where $\mathscr{F}_j = \begin{bmatrix} I_k & \mathcal{B}_j \\ 0_{n_j\times k} & \mathscr{L}_j \end{bmatrix} \in \mathbb{C}^{(k+n_j)\times(k+n_j)}$ and $\mathbb{H}_j = \begin{bmatrix} 0_{p\times k} & \mathbb{G}_{j-1} & E_j \\ & 0 & D_j \end{bmatrix} \in \mathbb{C}^{p\times(k+n_j)}$.

The numerical mechanism for selecting $\mathbb{V}_{j+1}$ out of $\left[P_{j-1}, \widetilde{W}_j\right]$ follows the same ideas as discussed in [6,88] in the context of block GMRES. The governing idea consists of building an orthonormal basis for the directions that contribute the most to the individual residual norms and make them larger than a prescribed threshold τ.

Specifically, the singular value decomposition (SVD) is applied to the least squares residuals

$$\Lambda_j - \underline{\mathscr{F}}_j Y_j = \mathbb{U}_{1,L}\Sigma_1 \mathbb{U}_{1,R}^H + \mathbb{U}_{2,L}\Sigma_2 \mathbb{U}_{2,R}^H, \tag{2.15}$$

where Σ_1 contains the p_{j+1} singular values greater than or equal to the prescribed threshold τ. Then we decompose $\mathbb{U}_{1,L} = \begin{pmatrix} \mathbb{U}_1^{(1)} \\ \mathbb{U}_1^{(2)} \end{pmatrix}$ in accordance with $\left[[C_k, \mathscr{V}_j], [P_{j-1}, \widetilde{W}_j]\right]$,

that is, $\mathbb{U}_1^{(1)} \in \mathbb{C}^{(k+n_j)\times p_{j+1}}$ and $\mathbb{U}_1^{(2)} \in \mathbb{C}^{p\times p_{j+1}}$. Because the objective is to construct an orthonormal basis, we consider a unitary matrix $[\mathbb{W}_1, \mathbb{W}_2]$ such that $\text{Range}(\mathbb{W}_1) = \text{Range}(\mathbb{U}_1^{(2)})$. The new set of orthonormal candidate vectors used to expand the search space

$$\mathbb{V}_{j+1} = \left[P_{j-1}, \widetilde{W_j}\right] \mathbb{W}_1 \tag{2.16}$$

is the set that contributes the most to the residual norms, while

$$P_j = \left[P_{j-1}, \widetilde{W_j}\right] \mathbb{W}_2,$$

is the new set of discarded directions with orthonormal columns. Through this mechanism, directions that have been discarded at a given iteration can be reintroduced if the residual block has a large component along them. Furthermore, this selection strategy ensures that all the solutions have converged when p partial convergence has been detected. We do not give details of the calculation but instead refer the reader to Section 2.3 of [88] for a complete description; we only state that via this decomposition, the main terms that appear in Equation (2.12) can be computed incrementally.

2.2.4 Subspace recycling policies along with partial convergence detection

So far, we have not made any specific assumption about the definition of the recycling space U_k except that it has full column rank. In the context of subspace recycling, one key point is to specify what subspace is to be recycled at restart. At the cost of the extra storage of k vectors, block GCRO offers more flexibility than block GMRES in the choice of the recycling space. This extra storage, which enables us to remove the constraint that the search space is included in the residual space, allows us to consider any subspace to be deflated at restart. In particular, either of the two classical alternatives, the Rayleigh-Ritz procedure or the harmonic-Ritz procedure, can be considered to compute the targeted approximate eigenvectors to define U_k and C_k at restart.

Definition 1. *harmonic-Ritz projection.*
Consider a subspace $\mathcal{W}$ of $\mathbb{C}^n$. Given a general nonsingular matrix $A \in \mathbb{C}^{n\times n}$, $\lambda \in \mathbb{C}$, and $g \in \mathcal{W}$, we see that (λ, g) is a harmonic-Ritz pair of A with respect to the space $\mathcal{W}$ if and only if

$$Ag - \lambda\, g \perp A\,\mathcal{W}$$

or equivalently,

$$\forall w \in \text{Range}(A\,\mathcal{W}), \quad w^H\,(Ag - \lambda\, g) = 0.$$

The vector g is a harmonic-Ritz vector associated with the harmonic-Ritz value λ.

Definition 2. *Rayleigh-Ritz projection.*
Consider a subspace $\mathcal{W}$ of $\mathbb{C}^n$. Given a general nonsingular matrix $A \in \mathbb{C}^{n\times n}$, $\lambda \in \mathbb{C}$, and $g \in \mathcal{W}$, we see that (λ, g) is a Rayleigh-Ritz pair of A with respect to the space $\mathcal{W}$ if and only if

$$Ag - \lambda\, g \perp \mathcal{W}$$

or equivalently,

$$\forall w \in \text{Range}(\mathcal{W}) \quad w^H\,(Ag - \lambda\, g) = 0.$$

The vector g is a Rayleigh-Ritz vector associated with the Rayleigh-Ritz value λ.

Once the maximum size of the search space has been reached, we have

$$\begin{aligned} A\widehat{\mathscr{W}}_m &= \widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m = \left[C_k, \mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right]\underline{\mathscr{F}}_m, && (2.17)\\ X_m &= X_1 + \widehat{\mathscr{W}}_m Y_m, && (2.18)\\ R_m &= B - AX_m = \left[C_k, \mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right](\Lambda_m - \underline{\mathscr{F}}_m Y_m), && (2.19)\\ Y_m &= \underset{Y\in\mathbb{C}^{(k+n_m)\times p}}{\operatorname{argmin}} \|\Lambda_m - \underline{\mathscr{F}}_m Y\|_F, \quad \Lambda_m = [0_{p\times k}, \Lambda_1^T, 0_{p\times n_m}]^T. && (2.20)\end{aligned}$$

Then, a restart procedure has to be implemented to possibly refine the spectral information to be recycled during the next cycle. Based on these equalities we will compute the approximated eigen-information as shown in Proposition 1 and then use it to define the new deflation basis U_k^{new} and its orthonormal image C_k^{new} by A as described in Theorem 1.

Proposition 1. *At restart of IB-BGCRO-DR, the update of the recycling subspace for the next cycle relies on the computation of harmonic-Ritz vectors $\widehat{\mathscr{W}}_m g_i^{(HR)} \in \operatorname{span}(\widehat{\mathscr{W}}_m)$, or Rayleigh-Ritz vectors $\widehat{\mathscr{W}}_m g_i^{(RR)} \in \operatorname{span}(\widehat{\mathscr{W}}_m)$, of A with respect to $\widehat{\mathscr{W}}_m = [U_k, \mathscr{V}_m] \in \mathbb{C}^{n\times(k+n_m)}$.*

- *The harmonic-Ritz pairs $(\theta_i, \widehat{\mathscr{W}}_m g_i^{(HR)})$ to be possibly used for the next restart satisfy*

$$\underline{\mathscr{F}}_m^H \underline{\mathscr{F}}_m g_i^{(HR)} = \theta_i \underline{\mathscr{F}}_m^H \widehat{\mathscr{V}}_{m+1}^H \widehat{\mathscr{W}}_m g_i^{(HR)}, \quad \text{for } 1 \le i \le k+n_m, \tag{2.21}$$

  where $\widehat{\mathscr{V}}_{m+1}^H \widehat{\mathscr{W}}_m = \begin{bmatrix} C_k^H U_k & 0_{k\times n_m} \\ \mathscr{V}_m^H U_k & I_{n_m} \\ P_{m-1}^H U_k & \\ \widetilde{W}_m^H U_k & 0_{p\times n_m} \end{bmatrix} \in \mathbb{C}^{(k+n_m+p)\times(k+n_m)}$.

- *The Rayleigh-Ritz pairs $(\theta_i, \widehat{\mathscr{W}}_m g_i^{(RR)})$ to be possibly used for the next restart satisfy*

$$\widehat{\mathscr{W}}_m^H \widehat{\mathscr{V}}_{m+1} \underline{\mathscr{F}}_m g_i^{(RR)} = \theta_i \widehat{\mathscr{W}}_m^H \widehat{\mathscr{W}}_m g_i^{(RR)}, \quad \text{for } 1 \le i \le k+n_m$$

  where $\widehat{\mathscr{W}}_m^H \widehat{\mathscr{V}}_{m+1} = \begin{bmatrix} U_k^H C_k & U_k^H \mathscr{V}_m & U_k^H P_{m-1} & U_k^H \widetilde{W}_m \\ 0_{n_m\times k} & I_{n_m} & 0_{n_m\times p} & \end{bmatrix} \in \mathbb{C}^{(k+n_m)\times(k+n_m+p)}$ *and* $\widehat{\mathscr{W}}_m^H \widehat{\mathscr{W}}_m = \begin{bmatrix} U_k^H U_k & U_k^H \mathscr{V}_m \\ \mathscr{V}_m^H U_k & I_{n_m} \end{bmatrix} \in \mathbb{C}^{(k+n_m)\times(k+n_m)}$.

Proof. The proofs basically rely on some matrix computations as shortly described below.

- According to Definition 1, each harmonic-Ritz pair $(\theta_i, \widehat{\mathscr{W}}_m g_i^{(HR)})$ satisfies

$$\forall w \in \operatorname{Range}(A\widehat{\mathscr{W}}_m) \quad w^H (A\widehat{\mathscr{W}}_m g_i^{(HR)} - \theta_i \widehat{\mathscr{W}}_m g_i^{(HR)}) = 0,$$

  which is equivalent to

$$(A\widehat{\mathscr{W}}_m)^H (A\widehat{\mathscr{W}}_m g_i^{(HR)} - \theta_i \widehat{\mathscr{W}}_m g_i^{(HR)}) = 0.$$

Substituting Equation (2.17) into the above leads to

$$\left(\widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m\right)^H \left(\widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m g_i^{(HR)} - \theta_i \widehat{\mathscr{W}}_m g_i^{(HR)}\right) = 0. \tag{2.22}$$

Because $\widehat{\mathscr{V}}_{m+1} = [C_k, \mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]]$ generated at the end of each cycle is orthonormal, Equation (2.22) becomes

$$\underline{\mathscr{F}}_m^H \underline{\mathscr{F}}_m g_i^{(HR)} - \theta_i \underline{\mathscr{F}}_m^H \widehat{\mathscr{V}}_{m+1}^H \widehat{\mathscr{W}}_m g_i^{(HR)} = 0,$$

which gives the formulation (2.21).

- Rayleigh-Ritz pairs: using Definition 2 and similar arguments and matrix computation enable to derive the proof.

□

Depending on the region of the spectrum that is intended to be deflated (e.g., subspace associated with the smallest and/or largest eigenvalues in magnitude), a subset of k approximated eigenvectors is chosen from among the $k + n_m$ ones to define a space that will be used to span U_k^{new}. Then, we describe in Theorem 1 the update of U_k^{new} and its image C_k^{new} with respect to A at restart of IB-BGCRO-DR.

Theorem 1. *At restart of IB-BGCRO-DR, if we intend to deflate the space* $\text{span}([U_k, \mathscr{V}_m]G_k^{(*)})$, *where* $G_k^{(*)} = \left[g_1^{(*)}, \ldots, g_k^{(*)}\right]$ *with* $G_k^{(*)} = G_k^{(HR)}$ *or* $G_k^{(*)} = G_k^{(RR)}$ *is the set of vectors associated with the targeted eigenvalues, then the matrices* U_k^{new} *and* C_k^{new} *to be used for the next cycle are defined by*

$$U_k^{new} = \widehat{\mathscr{W}}_m G_k^{(*)} R^{-1} = [U_k, \mathscr{V}_m] G_k^{(*)} R^{-1}, \tag{2.23}$$

$$C_k^{new} = \widehat{\mathscr{V}}_{m+1} Q = \left[C_k, \mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] Q, \tag{2.24}$$

where Q *and* R *are the factors of the reduced* QR*-factorization of the tall and skinny matrix* $\underline{\mathscr{F}}_m G_k^{(*)}$, *which* $AU_k^{new} = C_k^{new}$ *and* $(C_k^{new})^H C_k^{new} = I_k$.

Proof. Let Q and R be the factors of the reduced QR-factorization of the tall and skinny matrix $\underline{\mathscr{F}}_m G_k^{(*)}$. Right multiplying $G_k^{(*)}$ on both sides of Equation (2.17) leads to $A\widehat{\mathscr{W}}_m G_k^{(*)} = \widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m G_k^{(*)} = \widehat{\mathscr{V}}_{m+1} QR$, that is equivalent to $A\widehat{\mathscr{W}}_m G_k^{(*)} R^{-1} = \widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m G_k^{(*)} R^{-1} = \widehat{\mathscr{V}}_{m+1} Q$ concluding the proof as $\text{span}(\widehat{\mathscr{W}}_m G_k^{(*)} R^{-1}) = \text{span}(\widehat{\mathscr{W}}_m G_k^{(*)})$, and as $\widehat{\mathscr{V}}_{m+1} Q$ is the product of two matrices with orthonormal columns, so are its columns. □

Corollary 1. *The residual block at restart,* $R_1^{new} = R_m^{old} = B - AX_1^{new}$ *with* $X_1^{new} = X_m^{old}$ *is orthogonal to* C_k^{new}.

Proof. $X_m^{old} = X_1 + \widehat{\mathscr{W}}_m Y_m$ where Y_m solves the least squares problem (2.20) so that $(\Lambda_m - \underline{\mathscr{F}}_m Y_m) \in (\text{Range}(\underline{\mathscr{F}}_m))^\perp = \text{Null}(\underline{\mathscr{F}}_m^H)$. We also have $R_m^{old} = \widehat{\mathscr{V}}_{m+1}(\Lambda_m - \underline{\mathscr{F}}_m Y_m)$,

consequently

$$\begin{aligned}(C_k^{new})^H R_m^{old} &= \left(\widehat{\mathscr{V}}_{m+1}Q\right)^H \left(\widehat{\mathscr{V}}_{m+1}\left(\Lambda_m - \underline{\mathscr{F}}_m Y_m\right)\right) \\ &= \left(\widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m G_k^{(*)} R^{-1}\right)^H \left(\widehat{\mathscr{V}}_{m+1}\left(\Lambda_m - \underline{\mathscr{F}}_m Y_m\right)\right) \\ &= R^{-H} G_k^{(*)\,H} \underbrace{\underline{\mathscr{F}}_m^H \left(\Lambda_m - \underline{\mathscr{F}}_m Y_m\right)}_{=0 \quad \text{because of (2.20)}} = 0.\end{aligned}$$

□

2.2.5 Comparison with IB-BGMRES-DR in terms of reusing information

IB-BGMRES-DR [6] is a block GMRES method that enables the deflated restarting strategy proposed by Morgan [73] for recycling spectral information at a new cycle and the partial convergence detection mechanism introduced by Robbé and Sadkane [88] for handling the issue of almost rank deficient block generated by the block Arnoldi procedure. Assume the way of approximating the spectral information is the same for the IB-BGCRO-DR and IB-BGMRES-DR methods, the major difference between these two IB variants arise from their way of reusing the generated spectral information as described in Figure 2.1, in which the content in rectangle refers to the algorithm adopted in corresponding cycle and the directed arrow illustrates generating target spectral information at the end of the jth ($j = 1, 2, \ldots$) cycle (or family) and then reusing it in the subsequent $(j+1)$th cycle for convergence acceleration. Figure 2.1 illustrates that unlike IB-BGCRO-DR which could reuse spectral information from the solutions of previous family and cycle, IB-BGMRES-DR can only reuse information from the previous cycle, which means IB-BGMRES-DR could solely solve each individual family of linear systems (1.1) separately without benefiting from information generated when solving the previous family. Note that when solving single family in (1.1) and when no subspace is augmented at the beginning, IB-BGCRO-DR and IB-BGMRES-DR can be mathematically equivalent to each other under some conditions (like the way to approximate eigen-information) as the relationship between (block) GCRO-DR and (block) GMRES-DR, while the performance of the former GCRO one overs the later GMRES one when solving subsequent related-sequence families thanks to its ability of recycling spectral information between families as described in the directed arrows of the upper pink parts of Figure 2.1, and which has been verified by the numerical results shown in Section 2.6.3.

2.2.6 A variant suited for flexible preconditioning

All the descriptions in the previous sections are naturally extended to the right preconditioning case with a fixed preconditioner M, and the central equality reads

$$A[U_k, M\mathscr{V}_m] = \left[C_k, \mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] \underline{\mathscr{F}}_m. \tag{2.25}$$

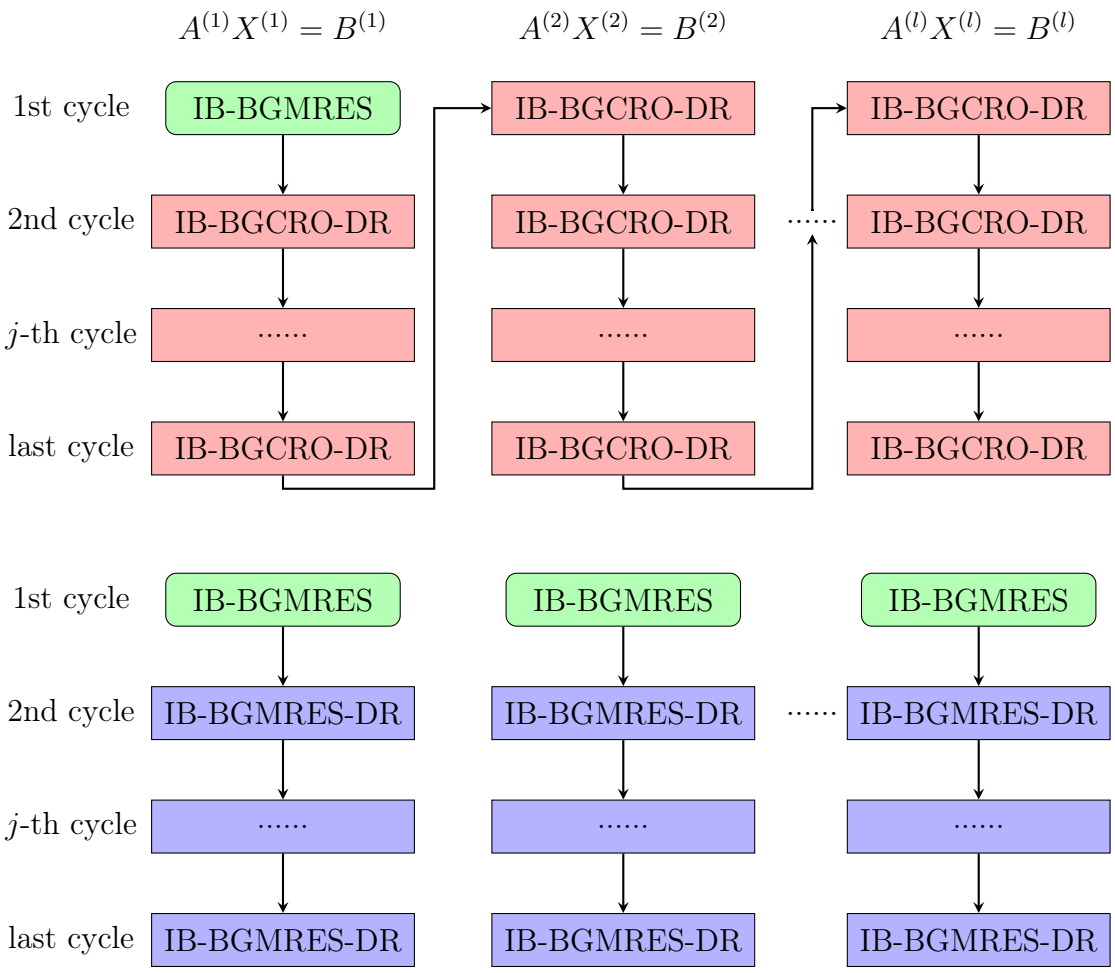


Figure 2.1 – Flowchart of reusing spectral information in the IB-BGCRO-DR (upper) and IB-BGMRES-DR (bottom) algorithms.

The least squares problem to be solved to compute the minimum norm solution becomes

$$Y_m = \underset{Y \in \mathbb{C}^{(k+n_m)\times p}}{\operatorname{argmin}} \|\Lambda_m - \underline{\mathscr{F}}_m Y\|_F \,,$$

and the solution is

$$X_m = X_1 + [U_k, M\mathscr{V}_m]Y_m.$$

If we denote by $\mathscr{M}_j$ a (possibly nonlinear) nonsingular preconditioning operator at iteration j and by $\mathscr{M}_j(\mathbb{V}_j)$ the action of $\mathscr{M}_j$ on a block vector $\mathbb{V}_j$, (2.25) translates into

$$A[U_k, \mathscr{Z}_m] = \left[C_k, \mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] \underline{\mathscr{F}}_m \text{ with } \mathscr{Z}_m = [\mathscr{M}_1(\mathbb{V}_1), \ldots, \mathscr{M}_m(\mathbb{V}_m)] \,,$$

which can be written in a more compact form as

$$A\widehat{\mathscr{Z}}_m = \widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m \text{ with } \widehat{\mathscr{Z}}_m = [U_k, \mathscr{Z}_m] \text{ and } \widehat{\mathscr{V}}_{m+1} = \left[C_k, \mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right]. \qquad (2.26)$$

The solution update is $X_m = X_1 + [U_k, \mathscr{Z}_m]Y_m$. For the sake of simplicity, we choose to keep the notation for quantities that have the same meaning as in the nonflexible case but of course they will have different values.

In the context of flexible preconditioning, many strategies for defining harmonic-Ritz vectors can be envisioned for GCRO-DR. Among those considered in [18], we follow the one with a lower computational cost required in solving the generalized eigenvalue problem, referred to as strategy C in [18]. Furthermore, it also allows us to obtain companion properties in the flexible preconditioning case that are quite similar to the ones we have shown in the nonpreconditioned case in Section 2.2.4. We refer the reader to Appendix A.1 for two other strategies for approximating targeted eigen-information. Proposition 2 indicates that with an appropriate definition of the harmonic-Ritz vectors, all the properties of IB-BGCRO-DR extend to the flexible preconditioning variant denoted as IB-BFGCRO-DR.

Proposition 2. *At the end of a cycle of the IB-BFGCRO-DR algorithm, if the deflation space is built on the harmonic-Ritz vectors $\mathcal{W}_m g_i \in \mathrm{span}(\mathcal{W}_m)$ of $A\widehat{\mathscr{Z}_m}\mathcal{W}_m^\dagger$ with respect to $\mathcal{W}_m = [\mathcal{W}_k, \mathscr{V}_m] \in \mathbb{C}^{n\times(k+n_m)}$, the following hold:*

1. *The harmonic-Ritz pairs $(\theta_i, \mathcal{W}_m g_i)$ for all restarts satisfy*

$$\underline{\mathscr{F}}_m^H \underline{\mathscr{F}}_m g_i = \theta_i \underline{\mathscr{F}}_m^H \widehat{\mathscr{V}}_{m+1}^H \mathcal{W}_m g_i, \quad \text{for } 1 \le i \le k + n_m, \tag{2.27}$$

where $\widehat{\mathscr{V}}_{m+1}^H \mathcal{W}_m = \begin{bmatrix} C_k^H \mathcal{W}_k & 0_{k\times n_m} \\ \mathscr{V}_m^H \mathcal{W}_k & I_{n_m} \\ P_{m-1}^H \mathcal{W}_k & \\ \widetilde{W}_m^H \mathcal{W}_k & 0_{p\times n_m} \end{bmatrix} \in \mathbb{C}^{(k+n_m+p)\times(k+n_m)}$.

2. *At restart, if $G_k = [g_1, \ldots, g_k]$ is associated with the k targeted eigenvalues, the matrices $\mathcal{W}_k^{new}$, U_k^{new} and C_k^{new} to be used for the next cycle are updated by*

$$\begin{aligned} \mathcal{W}_k^{new} &= \mathcal{W}_m G_k R^{-1} = [\mathcal{W}_k, \mathscr{V}_m] G_k R^{-1}, & (2.28)\\ U_k^{new} &= \widehat{\mathscr{Z}_m} G_k R^{-1} = [U_k, \mathscr{Z}_m] G_k R^{-1}, & (2.29)\\ C_k^{new} &= \widehat{\mathscr{V}}_{m+1} Q = \left[C_k, \mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] Q, \end{aligned}$$

   where Q and R are the factors of the reduced QR-factorization of the tall and skinny matrix $\underline{\mathscr{F}}_m G_k$, ensuring $AU_k^{new} = C_k^{new}$ with $(C_k^{new})^H C_k^{new} = I_k$.

3. *The residual at restart $R_1^{new} = R_m^{old} = B - AX_1^{new}$ with $X_1^{new} = X_m^{old}$ is orthogonal to C_k^{new}.*

Proof. The proofs basically rely on some matrix computations as shortly described below:

- According to Definition 1, each harmonic-Ritz pair $(\theta_i, \mathcal{W}_m g_i)$ satisfies

$$\forall w \in \mathrm{Range}(A\widehat{\mathscr{Z}_m}\mathcal{W}_m^\dagger \mathcal{W}_m) \quad w^H \left(A\widehat{\mathscr{Z}_m}\mathcal{W}_m^\dagger \mathcal{W}_m g_i - \theta_i \mathcal{W}_m g_i\right) = 0. \tag{2.30}$$

  Because $\mathcal{W}_m$ is initially set to be equal to $\mathscr{V}_m$ and then is updated by (2.28), which has full column rank, taking a left inverse for the Moore-Penrose inverse of $\mathcal{W}_m$

makes $\mathcal{W}_m^\dagger \mathcal{W}_m = I$. Therefore, the second formula of (2.30) equivalently becomes

$$(A\widehat{\mathscr{Z}_m})^H \left(A\widehat{\mathscr{Z}_m} g_i - \theta_i \mathcal{W}_m g_i\right) = 0. \tag{2.31}$$

Substituting (2.26) into the above leads to

$$\left(\widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m\right)^H \left(\widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m g_i - \theta_i \mathcal{W}_m g_i\right) = 0. \tag{2.32}$$

Because $\widehat{\mathscr{V}}_{m+1} = \left[C_k, \mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right]$ generated at the end of each cycle is orthonormal, (2.32) becomes

$$\underline{\mathscr{F}}_m^H \underline{\mathscr{F}}_m g_i - \theta_i \underline{\mathscr{F}}_m^H \widehat{\mathscr{V}}_{m+1}^H \mathcal{W}_m g_i = 0,$$

which gives the formulation (2.27).

- Let Q and R be the factors of the reduced QR-factorization of the tall and skinny matrix $\underline{\mathscr{F}}_m G_k$. Right multiplying G_k on both sides of (2.26) leads to $A\widehat{\mathscr{Z}_m} G_k = \widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m G_k = \widehat{\mathscr{V}}_{m+1} QR$, that is equivalent to $A\widehat{\mathscr{Z}_m} G_k R^{-1} = \widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m G_k R^{-1} = \widehat{\mathscr{V}}_{m+1} Q$, concluding the proof as $\mathrm{span}(\widehat{\mathscr{Z}_m} G_k R^{-1}) = \mathrm{span}(\widehat{\mathscr{Z}_m} G_k)$, and as $\widehat{\mathscr{V}}_{m+1} Q$ is the product of two matrices with orthonormal columns, so are its columns.
- The same process for proving Corollary 1.

□

A closely related numerical technique that extends IB-BGMRES-DR in the flexible preconditioning context can be derived similarly in the following Section 2.5, in which the resulting new algorithm named IB-BFGMRES-DR is detailed and its properties are described.

2.3 Search space expansion policies governed by the stopping criterion

In this section we describe a few novel policies for expanding the search space that generalize the original one considered for inexact breakdown detection [88]. In particular we first show how numerical criteria for detecting the partial convergence and expanding the search space can be tuned to ensure that a targeted threshold for a prescribed stopping criterion based on the individual backward error solution will be eventually satisfied. Second, we present how computational constraints can be taken into account and combined with any of the previous numerical criteria to best cope with the performance of the underlying computer architecture.

The partial convergence detection briefly described in Section 2.2.3 ensures that if all the singular values of the least squares residual are smaller than the threshold τ, then all the linear system residual norms are also smaller than τ (i.e., p partial convergences have occurred). This is due to the inequality

$$\forall i \quad \|b^{(i)} - Ax_j^{(i)}\| \leq \|B - AX_j\| = \|\Lambda_j - \underline{\mathscr{F}}_j Y_j\| = \sigma_{\max}(\Lambda_j - \underline{\mathscr{F}}_j Y_j) < \tau, \tag{2.33}$$

which follows from the facts that the 2-norm of a matrix is an upper bound of the 2-norm of its individual columns and that $\widehat{\mathscr{V}}_{j+1}$ has orthonormal columns.

2.3.1 Search space expansion policy governed by η_b

A classical stopping criterion for the solution of a linear system $Ax = b$ is based on backward error analysis and consists of stopping the iteration when

$$\eta_b(x_j) = \frac{\|b - Ax_j\|}{\|b\|} \leq \varepsilon. \tag{2.34}$$

This criterion was considered in [6] where it was consequently proposed to define $\tau = \varepsilon \min_{i=1,\ldots,p} \|b^{(i)}\|$. With this choice, when the iteration complies with (2.33), we have

$$\eta_b(x_j^{(i)}) \leq \frac{\|b^{(i)} - Ax_j^{(i)}\|}{\min_{i=1,\ldots,p} \|b^{(i)}\|} \leq \varepsilon. \tag{2.35}$$

When the different right-hand sides have very different norms in magnitude, the subspace expansion associated with this criterion might not be effective because the upper bound in (2.35) will not be tight. This leads to enlarging the search space with directions that are not relevant (generating useless computation). In that context a better choice would be to focus on the space expansion to reduce the residual associated with the right-hand side of the large norm. For that purpose, the idea is to perform the SVD not directly on the least squares residual but on its scaled least squares residual.

Proposition 3. *Performing the SVD of the scaled least squares residuals* $(\Lambda_j - \underline{\mathscr{F}}_j Y_j) D_{b,\varepsilon}$ *with threshold* $\tau = 1$ *and* $D_{b,\varepsilon} = \varepsilon^{-1} diag(\|b^{(1)}\|^{-1}, \ldots, \|b^{(p)}\|^{-1})$ *ensures that when* p *partial convergences have occurred, so that the search space cannot be enlarged, each of the current individual iterates complies with the stopping criterion* (2.34).

Proof. This is a direct consequence of the following inequalities

$$\max_{i=1,..,p} \frac{\|b^{(i)} - Ax_j^{(i)}\|}{\varepsilon \|b^{(i)}\|} \leq \|(B - AX_j) D_{b,\varepsilon}\| = \|(\Lambda_j - \underline{\mathscr{F}}_j Y_j) D_{b,\varepsilon}\| \leq 1$$

and implies $\forall i \ \eta_b(x_j^{(i)}) \leq \varepsilon$. □

In some applications all the solutions associated with a block of right-hand sides do not need to be solved with the same accuracy. That is, we may have to solve a family of right-hand sides $B = [b^{(1)}, \ldots, b^{(p)}]$ with individual convergence thresholds $\varepsilon^{(i)}$ for the solution associated with each right-hand side $b^{(i)}$ $(i = 1, \ldots, p)$; thus we have a more general version of (2.34),

$$\eta_{b^{(i)}}(x_j^{(i)}) = \frac{\|b^{(i)} - Ax_j^{(i)}\|}{\|b^{(i)}\|} \leq \varepsilon^{(i)}. \tag{2.36}$$

In that context, the subspace expansion policy can be easily adapted to ensure the convergence for each individual accuracy.

Corollary 2. *Performing the SVD of the scaled least squares residuals* $(\Lambda_j - \underline{\mathscr{F}}_j Y_j) D_{b,\varepsilon_i}$ *with threshold* $\tau = 1$ *and* $D_{b,\varepsilon_i} = diag((\varepsilon_1 \|b^{(1)}\|)^{-1}, \ldots, (\varepsilon_p \|b^{(p)}\|)^{-1})$ *ensures that when* p *partial convergences have occurred each of the current individual iterates complies with the stopping criterion* (2.36)*.*

2.3.2 Search space expansion policy governed by $\eta_{A,b}$

One can also adapt the expansion policy described in the previous section to the situation where the stopping criterion is based on the normwise backward error on A and b, defined by

$$\eta_{A,b}(x_j) = \frac{\|b - Ax_j\|}{\|b\| + \|A\| \, \|x_j\|} \leq \varepsilon. \tag{2.37}$$

It suffices to define accordingly the scaled least squares residuals in the SVD that is involved in the search space expansion. We notice that this type of stopping criterion will have a computational penalty as the iterates of all individual iterations have to be computed to calculate their norm.

Corollary 3. *Performing the SVD of the scaled least squares residual* $(\Lambda_j - \underline{\mathscr{F}}_j Y_j) D_{A,b,\varepsilon}$ *with threshold* $\tau = 1$ *and* $D_{A,b,\varepsilon} = \varepsilon^{-1} diag((\|A\| \|x_j^{(1)}\| + \|b^{(1)}\|)^{-1}, \ldots, (\|A\| \|x_j^{(p)}\| + \|b^{(p)}\|)^{-1})$ *ensures that when* p *partial convergences have occurred, each of the current individual iterates complies with the stopping criterion* (2.37)*.*

We do not develop further these ideas but similarly we could define expansion policies where for each solution we can select either η_b or $\eta_{A,b}$ as stopping criterion with individual threshold setting.

The occurrence of p partial convergences is a sufficient condition that ensures the convergence of the p solution vectors, but the convergence might occur earlier, and a more classical stopping criterion can be accommodated at a low computational cost. Given that the norms of true residuals are very close to those of the least squares residuals when the loss of orthogonality of the generated block Krylov basis is not too serious, one can also check the convergence by looking at the norm of the least squares residual, which is easy to compute. Let $Q_j^{LS} R_j^{LS}$ be a full QR-factorization of $\underline{\mathscr{F}}_j$ (i.e., Q_j^{LS} is unitary); then

$$\Lambda_j - \underline{\mathscr{F}}_j Y_j = Q_j^{LS} \begin{pmatrix} 0_{(n_j+k)\times p} \\ R_j^{\ell s} \end{pmatrix}, \tag{2.38}$$

where $R_j^{\ell s} \in \mathbb{C}^{p \times p}$ are the last p rows of $(Q_j^{LS})^H \Lambda_j$ so that $\|b^{(i)} - Ax_j^{(i)}\| = \|R_j^{\ell s}(:, i)\|$. Those residual norm calculations are part of the stopping criterion based on η_b or $\eta_{A,b}$

2.3.3 Search space expansion policy governed by computational performance

Based on any of these expansion policies, the discarded directions at a given iteration might be reintroduced in a subsequent one; thereby we can trade on the considered numerical policy and select for the subspace expansion only a subset of those eligible. In particular, it might be relevant to choose a prescribed block size p^{CB} (here the superscript ^{CB} stands for computational blocking) that is best suited to cope with the computational

features on a given platform rather than selecting the numerical block size p_{j+1} defined as the number of singular values greater than or equal to the prescribed threshold $\tau = 1$. In that respect, we consider a subspace expansion policy so that the block size at the end of step j is defined as $p_{j+1}^{CB} = \min(p^{CB}, p_{j+1})$. We refer to this variant as inexact breakdown block GCRO-DR with computational blocking (denoted by IB-BGCRO-DR-CB).

Note that all the subspace expansion policies discussed in Section 2.3 could be applied to any other block minimum residual norm methods equipped with the partial convergence detection such as the IB-BGMRES [88] and IB-BGMRES-DR [6] algorithms.

2.4 Computational and algorithmic remarks

The mathematical description in the previous section assumes exact calculation. In practice, the numerical behavior of the algorithms depends on the numerical algorithms selected to perform the computation in finite precision arithmetic. In particular, all the above descriptions assume the orthonormality of the residual basis; the orthonomality ensures the norm equality of the true linear system residual and their least squares counterpart which governs the numerical search space expansion policies described in the previous section. In our implementation, for the block Arnoldi procedure (See Algorithm 1), we consider the block modified Gram-Schmidt (BMGS) algorithm with reduced QR-factorization based on Householder reflections of the final tall and skinny block (referred to as (BMGS $\circ$ HouseQR) in [17]). In addition, at restart the re-orthogonalization of the recycling space C_k and of the initial block residual vector $[\mathbb{V}_1, P_0]$ in Equation (2.40) is performed a vector at a time using modified Gram-Schmidt.

2.4.1 Inexact breakdown and re-orthogonalization at restart

For the sake of simplicity, in the previous sections we made the assumption that the initial residual block was of full rank. In practice, this constraint can be removed by applying the partial convergence detection to the initial residual block. In that case, only a subspace of the space spanned by the columns of the initial residual block will be selected to define the first search space, and the discarded directions are kept in the basis of the residual space. This has the following two main consequences:

1. The first iteration needs some extra attention to set up the initial basis $\mathbb{V}_1$ and discarded directions P_0 defined in (2.11).

2. A consequence of having discarded directions in the first search space is that the projection of the initial residual block in the residual space that defines the right-hand side of the least squares residual solved at each block iteration will no longer have the nested block structure that is expanded by a $p \times p$ zero block at each block iteration as presented in (2.20).

Without loss of generality, let us present the partial convergence detection and re-orthogonalization at restart where the recycling subspaces U_k^{new} and C_k^{new} are defined by (2.23) and (2.24), respectively, so that mathematically $AU_k^{new} = C_k^{new}$ and $(C_k^{new})^H C_k^{new} = I_k$, and the initial residual block $R_1^{new} = R_1$ in Corollary 1 is orthogonal to C_k^{new}. For a prescribed stopping criterion and convergence threshold, let us denote by

D_ε the diagonal matrix used to select the space expansion described in the Section 2.3. Let

$$R_1 D_\varepsilon = [\mathbb{V}_1^{new}, P_0^{new}] \begin{bmatrix} \Sigma_{p_1} & \\ & \Sigma_{q_1} \end{bmatrix} \mathbb{V}_{R_1}^H = [\mathbb{V}_1^{new}, P_0^{new}] \hat{\Lambda}_1', \tag{2.39}$$

where $\mathbb{V}_1^{new} \in \mathbb{C}^{n\times p_1}$, $P_0^{new} \in \mathbb{C}^{n\times q_1}$ with $p_1 + q_1 = p$, and Σ_{p_1} contains the p_1 singular values of $R_1 D_\varepsilon$ greater than or equal to the prescribed τ, and Σ_{q_1} contains the ones smaller than τ.

We first perform an MGS re-orthogonalization of the columns of $[C_k^{new}, [\mathbb{V}_1^{new}, P_0^{new}]]$ that reads

$$[C_k^{new}, [\mathbb{V}_1^{new}, P_0^{new}]] = [C_k, [\mathbb{V}_1, P_0]] \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix}, \tag{2.40}$$

where all the columns of $[C_k, [\mathbb{V}_1, P_0]]$ are orthogonal to one another, and $\begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix} \in \mathbb{C}^{(k+p)\times(k+p)}$ is an upper triangular matrix with $R_{11} \in \mathbb{C}^{k\times k}$ and $R_{22} \in \mathbb{C}^{p\times p}$. Next, we update $U_k = U_k^{new} R_{11}^{-1}$ to satisfy (2.2), and $\mathscr{V}_1 = \mathbb{V}_1$ will serve to span the first search space and P_0 will be abandoned for this first block iteration that will be run as follows.

1. Form $W_1 = A\mathbb{V}_1$ and orthogonalize it (using BMGS $\circ$ HouseQR) against the set of orthonormal vectors that are part of the residual space $[C_k, \mathbb{V}_1, P_0]$ which enables the computation of the entries of $\mathcal{B}_1 = C_k^H W_1$, $\mathscr{L}_{1,1} = \mathbb{V}_1^H W_1$ and $E_1 = P_0^H W_1$.

2. The resulting block $\bar{W}_1$ formally reads $\bar{W}_1 = W_1 - C_k\mathcal{B}_1 - \mathbb{V}_1\mathscr{L}_{1,1} - P_0 E_1$ with $\bar{W}_1 = \widetilde{W}_1 D_1$ being its reduced QR-factorization.

3. In matrix form the above relations also reads

$$W_1 = A\mathbb{V}_1 = \left[C_k, \mathbb{V}_1, [P_0, \widetilde{W}_1]\right] \begin{bmatrix} \mathcal{B}_1 \\ \mathscr{L}_{1,1} \\ E_1 \\ D_1 \end{bmatrix},$$

   so that we have the first Arnoldi-like relation

$$A[U_k, \mathbb{V}_1] = \left[C_k, \mathbb{V}_1, [P_0, \widetilde{W}_1]\right] \underline{\mathscr{F}}_1 \tag{2.41}$$

   with

$$\underline{\mathscr{F}}_1 = \begin{bmatrix} I_k & \mathcal{B}_1 \\ 0_{(p_1+p)\times k} & \mathscr{L}_{1,1} \\ & \widetilde{\mathbb{H}}_1 \end{bmatrix} \in \mathbb{C}^{(k+p_1+p)\times(k+p_1)} \text{ and } \widetilde{\mathbb{H}}_1 = \begin{bmatrix} E_1 \\ D_1 \end{bmatrix} \in \mathbb{C}^{p\times p_1}.$$

4. Next, define the minimum norm solution $X_2 = X_1 + [U_k, \mathbb{V}_1]Y$, and note that R_1 belongs to the space $[C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]$ where its components in this orthogonal basis

are given by $[C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H R_1$. From Equation (2.41) we have

$$\begin{aligned}\|B - AX_2\|_F &= \|R_1 - A\,[U_k, \mathbb{V}_1]\,Y\|_F = \|R_1 - [C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]\underline{\mathscr{F}}_1 Y\|_F \\ &= \|[C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H R_1 - \underline{\mathscr{F}}_1 Y\|_F \\ &= \|[C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H [\mathbb{V}_1^{new}, P_0^{new}]\hat{\Lambda}_1 - \underline{\mathscr{F}}_1 Y\|_F,\end{aligned}$$

and then from Equation (2.39), we have

$$R_1 = [\mathbb{V}_1^{new}, P_0^{new}]\hat{\Lambda}_1' D_\varepsilon^{-1} = [\mathbb{V}_1^{new}, P_0^{new}]\hat{\Lambda}_1 \text{ with } \hat{\Lambda}_1 = \hat{\Lambda}_1' D_\varepsilon^{-1}, \tag{2.42}$$

so that from Equation (2.40), the right-hand side of the above least squares residual reads

$$\begin{aligned}\Lambda_1 &= [C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H [\mathbb{V}_1^{new}, P_0^{new}]\hat{\Lambda}_1 = [C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H [C_k R_{12} + [\mathbb{V}_1, P_0]R_{22}]\hat{\Lambda}_1 \\ &= \left([C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H C_k R_{12} + [C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H [\mathbb{V}_1, P_0]R_{22}\right)\hat{\Lambda}_1 \\ &= \begin{bmatrix} R_{12} \\ 0_{(p_1+p)\times p} \end{bmatrix}\hat{\Lambda}_1 + \begin{bmatrix} 0_{k\times p_1} & 0_{k\times q_1} \\ I_{p_1} & 0_{p_1\times q_1} \\ 0_{q_1\times p_1} & I_{q_1} \\ 0_{p_1\times p_1} & 0_{p_1\times q_1} \end{bmatrix} R_{22}\hat{\Lambda}_1 \in \mathbb{C}^{(k+p_1+p)\times p}.\end{aligned} \tag{2.43}$$

5. Compute Y_1 the solution of the first new least squares problem

$$Y_1 = \underset{Y\in\mathbb{C}^{(k+p_1)\times p}}{\operatorname{argmin}} \|\Lambda_1 - \underline{\mathscr{F}}_1 Y\|_F\,.$$

6. Execute the search space expansion policy following the IB principles

   (a) Compute the SVD of the scaled least squares residual

   $$(\Lambda_1 - \underline{\mathscr{F}}_1 Y_1)\,D_\varepsilon = \mathbb{U}_{1,L}\Sigma_1 \mathbb{V}_{1,R}^H + \mathbb{U}_{2,L}\Sigma_2 \mathbb{V}_{2,R}^H, \text{ where } \sigma_{\min}(\Sigma_1) \geq 1 > \sigma_{\max}(\Sigma_2).$$

   (b) Compute $\mathbb{W}_1$ and $\mathbb{W}_2$ such that $\text{Range}(\mathbb{W}_1) = \text{Range}(\mathbb{U}_1^{(2)}) \in \mathbb{C}^{p\times p_2}$ with $\mathbb{U}_{1,L} = \begin{pmatrix} \mathbb{U}_1^{(1)} \\ \mathbb{U}_1^{(2)} \end{pmatrix} \in \mathbb{C}^{(k+p_1+p)\times p_2}$, $[\mathbb{W}_1, \mathbb{W}_2]$ is unitary and $\mathbb{W}_2 \in \mathbb{C}^{p\times q_2}$ with $p_2 + q_2 = p$.

   (c) Compute the new orthonormal matrices $\mathbb{V}_2$ and P_1 as

   $$\mathbb{V}_2 = [P_0, \widetilde{W}_1]\mathbb{W}_1 \in \mathbb{C}^{n\times p_2},\ P_1 = [P_0, \widetilde{W}_1]\mathbb{W}_2 \in \mathbb{C}^{n\times q_2},$$

   and compute as well the last block row matrix $\mathscr{L}_{2,:}$ of $\underline{\mathscr{L}}_1$ and $\mathbb{G}_1$ as

   $$\mathscr{L}_{2,:} = \mathbb{W}_1^H \widetilde{\mathbb{H}}_1 \in \mathbb{C}^{p_2\times p_1},\ \mathbb{G}_1 = \mathbb{W}_2^H \widetilde{\mathbb{H}}_1 \in \mathbb{C}^{q_2\times p_1}.$$

7. Set $\underline{\mathscr{L}}_1 = \begin{pmatrix} \mathscr{L}_1 \\ \mathscr{L}_{2,:} \end{pmatrix} \in \mathbb{C}^{(p_1+p_2)\times p_1} = \mathbb{C}^{n_2\times p_1}$.

Whenever a partial convergence is detected in R_1, some of its components (along

P_0^{new}) are first discarded but could be reintroduced in some subsequent iterations. One of the consequences of this is that the last q_1 columns of the least squares right-hand side problem evolve from one iteration to the next, depending on how some of the P_0^{new} directions are reintroduced in the search space along the iterations. There is a way to incrementally update the least squares right-hand side, and this is discussed in the next proposition.

Proposition 4. *At each iteration of IB-BGCRO-DR, the new least squares problem reads*

$$Y_{j+1} = \underset{Y \in \mathbb{C}^{(k+n_{j+1})\times p}}{\operatorname{argmin}} \left\| \Lambda_{j+1} - \underline{\mathscr{F}}_{j+1} Y \right\|_F, \quad \Lambda_{j+1} \in \mathbb{C}^{(k+n_{j+1}+p)\times p}, \quad j = 0, 1, 2, \ldots, \tag{2.44}$$

with the updated right-hand sides being

$$\Lambda_{j+1} = \begin{bmatrix} R_{12} \\ 0_{(n_j+p+p_{j+1})\times p} \end{bmatrix} \hat{\Lambda}_1 + \begin{bmatrix} 0_{k\times p_1} & 0_{k\times q_1} \\ \begin{bmatrix} I_{p_1} \\ 0_{(n_j+p-p_1)\times p_1} \end{bmatrix} & \Phi_{j+1} \\ 0_{p_{j+1}\times p_1} & 0_{p_{j+1}\times q_1} \end{bmatrix} R_{22}\ \hat{\Lambda}_1, \tag{2.45}$$

where $\Phi_{j+1} = \begin{bmatrix} \Phi_j(1:n_j,:) \\ [\mathbb{W}_1, \mathbb{W}_2]^H \begin{bmatrix} \Phi_j(n_j+1:n_j+q_j,:) \\ 0_{p_j\times q_1} \end{bmatrix} \end{bmatrix} \in \mathbb{C}^{(n_j+p)\times q_1}$ *for* $j = 0, 1, 2, \ldots,$ *with* $\Phi_1 = \begin{bmatrix} 0_{p_1\times q_1} \\ I_{q_1} \end{bmatrix} \in \mathbb{C}^{p\times q_1}$ *and* $q_j = p - p_j (j > 0)$; $[\mathbb{W}_1, \mathbb{W}_2]$ *is unitary as defined in the search space expansion algorithm based on IB principles; and* $R_{12} \in \mathbb{C}^{k\times p}$ *and* $R_{22} \in \mathbb{C}^{p\times p}$ *are two block components of the upper triangular matrix as shown in the right-hand side of Equation* (2.40).

Proof. From (2.39), (2.40), and (2.42), the initial residual block R_1 with partial convergence detection at restart could be described as

$$\begin{aligned} R_1 &= [C_k, \mathbb{V}_1, P_0, \widetilde{W}_1][C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H R_1 \\ &= [C_k, \mathbb{V}_1, P_0, \widetilde{W}_1][C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H [\mathbb{V}_1^{new}, P_0^{new}]\hat{\Lambda}_1 \\ &= [C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]\ \left([C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H C_k R_{12} + [C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]^H [\mathbb{V}_1, P_0] R_{22} \right) \hat{\Lambda}_1 \\ &= [C_k, \mathbb{V}_1, P_0, \widetilde{W}_1]\Lambda_1 \text{ with } \Lambda_1 = \begin{bmatrix} R_{12} \\ 0_{(p_1+p)\times p} \end{bmatrix} \hat{\Lambda}_1 + \begin{bmatrix} 0_{k\times p_1} & 0_{k\times q_1} \\ I_{p_1} & 0_{p_1\times q_1} \\ 0_{q_1\times p_1} & I_{q_1} \\ 0_{p_1\times p_1} & 0_{p_1\times q_1} \end{bmatrix} R_{22}\hat{\Lambda}_1, \end{aligned}$$

by $[\mathbb{V}_1^{new}, P_0^{new}] = C_k R_{12} + [\mathbb{V}_1, P_0] R_{22}$ obtained from Equation (2.40). That can also be written as

$$\Lambda_1 = \begin{bmatrix} R_{12} \\ 0_{(p_1+p)\times p} \end{bmatrix} \hat{\Lambda}_1 + \begin{bmatrix} 0_{k\times p_1} & 0_{k\times q_1} \\ I_{p_1} & \Phi_1 \\ 0_{q_1\times p_1} & \\ 0_{p_1\times p_1} & 0_{p_1\times q_1} \end{bmatrix} R_{22}\hat{\Lambda}_1,$$

where $\Phi_1 = \begin{bmatrix} 0_{p_1\times q_1} \\ I_{q_1} \end{bmatrix} \in \mathbb{C}^{p\times q_1}$ and $q_1 + p_1 = p$.

The right-hand sides of the least squares problem at iteration $(j+1)$ for $j = 1, 2, \ldots$, are defined by

$$
\begin{aligned}
\Lambda_{j+1} &= [C_k, \mathscr{V}_{j+1}, [P_j, \widetilde{W}_{j+1}]]^H R_1 \\
&= [C_k, \mathscr{V}_j, V_{j+1}, [P_j, \widetilde{W}_{j+1}]]^H R_1 \\
&= \left[C_k, \mathscr{V}_j, [P_{j-1}, \widetilde{W}_j]\mathbb{W}_1, [P_{j-1}, \widetilde{W}_j]\mathbb{W}_2, \widetilde{W}_{j+1}\right]^H R_1 \\
&= \left[C_k, \mathscr{V}_j, [P_{j-1}, \widetilde{W}_j][\mathbb{W}_1, \mathbb{W}_2], \widetilde{W}_{j+1}\right]^H [\mathbb{V}_1^{new}, P_0^{new}]\hat{\Lambda}_1 \\
&= \left[C_k, \mathscr{V}_j, [P_{j-1}, \widetilde{W}_j][\mathbb{W}_1, \mathbb{W}_2], \widetilde{W}_{j+1}\right]^H C_k R_{12}\hat{\Lambda}_1 + \\
&\quad \left[C_k, \mathscr{V}_j, [P_{j-1}, \widetilde{W}_j][\mathbb{W}_1, \mathbb{W}_2], \widetilde{W}_{j+1}\right]^H [\mathbb{V}_1, P_0] R_{22}\hat{\Lambda}_1 \\
&= \begin{bmatrix} R_{12} \\ 0_{(n_j+p+p_{j+1})\times p} \end{bmatrix} \hat{\Lambda}_1 + \begin{bmatrix} C_k^H\mathbb{V}_1 & C_k^H P_0 \\ \mathscr{V}_j^H\mathbb{V}_1 & \mathscr{V}_j^H P_0 \\ [V_{j+1}, P_j]^H\mathbb{V}_1 & [\mathbb{W}_1, \mathbb{W}_2]^H\left[P_{j-1}, \widetilde{W}_j\right]^H P_0 \\ \widetilde{W}_{j+1}^H\mathbb{V}_1 & \widetilde{W}_{j+1}^H P_0 \end{bmatrix} R_{22}\hat{\Lambda}_1 \\
&= \begin{bmatrix} R_{12} \\ 0_{(n_j+p+p_{j+1})\times p} \end{bmatrix} \hat{\Lambda}_1 + \begin{bmatrix} 0_{k\times p_1} & 0_{k\times q_1} \\ \begin{bmatrix} I_{p_1} \\ 0_{(n_j-p_1)\times p_1} \end{bmatrix} & \Phi_j(1:n_j,:) \\ 0_{p\times p_1} & [\mathbb{W}_1, \mathbb{W}_2]^H \begin{bmatrix} P_{j-1}^H \\ \widetilde{W}_j^H \end{bmatrix} P_0 \\ 0_{p_{j+1}\times p_1} & 0_{p_{j+1}\times q_1} \end{bmatrix} R_{22}\hat{\Lambda}_1 \\
&= \begin{bmatrix} R_{12} \\ 0_{(n_j+p+p_{j+1})\times p} \end{bmatrix} \hat{\Lambda}_1 + \begin{bmatrix} 0_{k\times p_1} & 0_{k\times q_1} \\ \begin{bmatrix} I_{p_1} \\ 0_{(n_j-p_1)\times p_1} \end{bmatrix} & \Phi_j(1:n_j,:) \\ 0_{p\times p_1} & [\mathbb{W}_1, \mathbb{W}_2]^H \begin{bmatrix} \Phi_j(n_j+1:n_j+q_j,:) \\ 0_{p_j\times q_1} \end{bmatrix} \\ 0_{p_{j+1}\times p_1} & 0_{p_{j+1}\times q_1} \end{bmatrix} R_{22}\hat{\Lambda}_1 \\
&= \begin{bmatrix} R_{12} \\ 0_{(n_j+p+p_{j+1})\times p} \end{bmatrix} \hat{\Lambda}_1 + \begin{bmatrix} 0_{k\times p_1} & 0_{k\times q_1} \\ \begin{bmatrix} I_{p_1} \\ 0_{(n_j+p-p_1)\times p_1} \end{bmatrix} & \Phi_{j+1} \\ 0_{p_{j+1}\times p_1} & 0_{p_{j+1}\times q_1} \end{bmatrix} R_{22}\hat{\Lambda}_1
\end{aligned}
$$

where $\Phi_{j+1} \in \mathbb{C}^{(n_j+p)\times q_1}$ for $j = 1, 2, \ldots$. □

Based on the above discussions, the IB-BGCRO-DR algorithm with partial convergence detection in the initial residual block and updated right-hand sides of the least squares residual is presented as Algorithm 2 for solving a series of linear systems with slowly changing left-hand sides.

Algorithm 2 IB-BGCRO-DR for slowly changing left-hand sides and massive number of right-hand sides

Require: $A \in \mathbb{C}^{n\times n}$ left-hand side of current family (not vary much compared to previous one)
Require: $B \in \mathbb{C}^{n\times p}$ the block of right-hand-sides and $X_0 \in \mathbb{C}^{n\times p}$ the block initial guess
Require: m maximum number of Arnoldi steps within a cycle
Require: p^{CB} a given constant number satisfying $1 \le p^{CB} \le p$ for computational blocking
Require: $D_\varepsilon \in \mathbb{C}^{p\times p}$ a diagonal matrix used to select the space expansion described in the Section 2.3
Require: $U_k, C_k \in \mathbb{C}^{n\times k}$ the recycling subspaces assumed to be empty for the first family and obtained after solving previous slow-changing family of linear systems

Compute $R_0 = B - AX_0$
/ Some families have already been solved ? */*
if the recycling space is not empty, $U_k \neq 0$ **then**
  Apply the reduced QR-factorization to AU_k for updating U_k and C_k for the current family such that the U_k and C_k satisfy (2.2) and (2.3). Compute R_1 and X_1 as described in (2.6)
else
  Set $R_1 = R_0$, $X_1 = X_0$, $U_k = 0$, $C_k = 0$
end if
/ Loop over the restarts */*
while the stopping criterion based on Section 2.3.1 or 2.3.2 is not met **do**
  Apply partial convergence detection in the scaled (least squares) residual block following Section 2.4.1
  / Arnoldi loop */*
  for $j = 2, 3, \ldots, m$ **do**
    Orthogonalize $A\mathbb{V}_j$ against C_k as $W_j = (I - C_kC_k^H)A\mathbb{V}_j$. Then orthogonalize W_j against previous block orthonormal vectors $\mathscr{V}_j = [\mathbb{V}_1, \ldots, \mathbb{V}_j]$ as

$$W_j = A\mathbb{V}_j - C_kC_k^HA\mathbb{V}_j - \mathscr{V}_j\mathscr{L}_{1,1:j}, \text{ where } \mathscr{L}_{1,1:j} = \mathscr{V}_j^H(W_j) = \mathscr{V}_j^H(A\mathbb{V}_j) \text{ is a block column matrix}$$

    Set $\mathscr{L}_j = \begin{bmatrix}\underline{\mathscr{L}}_{j-1}, & \mathscr{L}_{1,1:j}\end{bmatrix} \in \mathbb{C}^{n_j\times n_j}$, $\mathcal{B}_j = \begin{bmatrix}\mathcal{B}_{j-1}, & C_k^HA\mathbb{V}_j\end{bmatrix} \in \mathbb{C}^{k\times n_j}$
    Orthogonalize W_j against P_{j-1} and carry out its reduced QR-factorization as

$$\widetilde{W}_jD_j = W_j - P_{j-1}E_j, \text{ where } E_j = P_{j-1}^HW_j$$

    Compute Y_j by solving the least squares problem described in (2.13) (or (2.44)) with $\underline{\mathscr{F}}_j$ shown in (2.14) composed by $\mathscr{F}_j$ and $\mathbb{H}_j$ but with the updated right-hand side Λ_j as shown in (2.45) instead
    if the stopping criterion is met **then**
      return $X_j = X_1 + [U_k, \mathscr{V}_j]Y_j$, U_k and C_k
    end if
    Singular value decomposition of the residuals scaled by D_ε

$$(\Lambda_j - \underline{\mathscr{F}}_jY)D_\varepsilon = \mathbb{U}_{1,L}\Sigma_1\mathbb{V}_{1,R}^H + \mathbb{U}_{2,l}\Sigma_2\mathbb{V}_{2,R}^H \text{ with } \sigma_{\min}(\Sigma_1) \ge 1 > \sigma_{\max}(\Sigma_2)$$

    if Computational blocking of Section 2.3.3 is activated **then**
      $\mathbb{U}_{1,L} = \mathbb{U}_{1,L}(:, 1 : p_j^{CB})$ with $p_j^{CB} = min(p^{CB}, nl_{\Sigma_1})$, nl_{Σ_1} refers to column number of Σ_1
    end if
    Following item 6 described in Section 2.4.1 for computing $\mathbb{W}_1$ and $\mathbb{W}_2$
    Compute orthonormal matrices $\mathbb{V}_{j+1}$ and P_j, the last block row matrix $\mathscr{L}_{j+1,:}$ of $\underline{\mathscr{L}}_j$, and G_j as

$$\mathbb{V}_{j+1} = \left[P_{j-1}, \widetilde{W}_j\right]\mathbb{W}_1, P_j = \left[P_{j-1}, \widetilde{W}_j\right]\mathbb{W}_2, \mathscr{L}_{j+1,:} = \mathbb{W}_1^H\mathbb{H}_j, \mathbb{G}_j = \mathbb{W}_2^H\mathbb{H}_j, \underline{\mathscr{L}}_j = \begin{pmatrix}\mathscr{L}_j \\ \mathscr{L}_{j+1,:}\end{pmatrix}$$

  end for
  / Restart procedure */*
  Compute the solution X_m as described in (2.18) and residual R_m according to (2.19)
  Compute the targeted harmonic-Ritz vectors $G_k = [g_1, \ldots, g_k]$ by solving the generalized eigenvalue problem (2.21) described in Proposition 1
  Update the values of U_k and C_k, respectively, by (2.23) and (2.24) described in Theorem 1
  Restart with $X_1 = X_m$, $\widehat{\mathscr{V}}_{m+1}$, $R_1^{LS} = \Lambda_m - \underline{\mathscr{F}}_mY_m$ ($R_1 = R_m = \widehat{\mathscr{V}}_{m+1}R_1^{LS}$)
end while
return X_j for approximation of the current family; U_k, C_k for the next family to be solved

2.4.2 Solution of the least squares problem and cheap SVD calculation of the scaled least squares residual

The partial convergence detection mechanism allows us to extract from the residual spaces new directions to expand the search space at the next iteration of the block method. The selection consists of extracting the directions that contribute the most to the scaled residual block and is based on the SVD of the scaled least squares residual. In this section, we detail how the solution of the least squares problem (2.13) enables to compute easily and cheaply the SVD of the associated scaled (least squares) residual block. The least squares problem

$$Y_j = \underset{Y\in\mathbb{C}^{(k+n_j)\times p}}{\operatorname{argmin}} \left\|\Lambda_j - \underline{\mathscr{F}}_j Y\right\|_F, \text{ with } \underline{\mathscr{F}}_j \in \mathbb{C}^{(k+n_j+p)\times(k+n_j)} \tag{2.46}$$

is solved by using a full QR-factorization of $\underline{\mathscr{F}}_j = Q_j^{LS} R_j^{LS}$, where the superscript ^{LS} comes from least squares, $Q_j^{LS} = [Q_j^{LS(1)},\ Q_j^{LS(2)}]$ with $Q_j^{LS(1)} \in \mathbb{C}^{(k+n_j+p)\times(k+n_j)}$ and $Q_j^{LS(2)} \in \mathbb{C}^{(k+n_j+p)\times p}$, and $R_j^{LS} = \begin{bmatrix} R_j^{LS(1)} \\ 0_{p\times(k+n_j)} \end{bmatrix} \in \mathbb{C}^{(k+n_j+p)\times(k+n_j)}$ with $R_j^{LS(1)} \in \mathbb{C}^{(k+n_j)\times(k+n_j)}$ is an upper triangular matrix, from which the reduced QR-factorization of $\underline{\mathscr{F}}_j$ is formulated as $\underline{\mathscr{F}}_j = Q_j^{LS(1)} R_j^{LS(1)}$ if $Q_j^{LS(1)}$ is considered as an orthogonal basis of $\underline{\mathscr{F}}_j$. Thus, we could still formulate Y_j in a relatively economic way as

$$Y_j = (R_j^{LS(1)})^{-1}((Q_j^{LS(1)})^H \Lambda_j) \in \mathbb{C}^{(k+n_j)\times p}, \tag{2.47}$$

from which we could deduce the residual of the least squares problem described in (2.38) as follows:

$$\begin{aligned}
\Lambda_j - \underline{\mathscr{F}}_j Y_j &= \Lambda_j - Q_j^{LS} R_j^{LS} Y_j = Q_j^{LS}\left((Q_j^{LS})^H \Lambda_j - R_j^{LS} Y_j\right), \\
&= Q_j^{LS}\left(\begin{bmatrix} (Q_j^{LS(1)})^H \\ (Q_j^{LS(2)})^H \end{bmatrix} \Lambda_j - \begin{bmatrix} R_j^{LS(1)} \\ 0_{p\times(k+n_j)} \end{bmatrix} Y_j\right), \\
&= Q_j^{LS}\left(\begin{bmatrix} 0_{(k+n_j)\times(k+n_j+p)} \\ (Q_j^{LS(2)})^H \end{bmatrix} \Lambda_j\right), \\
&= Q_j^{LS}\begin{pmatrix} 0_{(k+n_j)\times p} \\ R_j^{\ell s} \end{pmatrix},
\end{aligned}$$

where $R_j^{\ell s} = (Q_j^{LS(2)})^H \Lambda_j \in \mathbb{C}^{p\times p}$ are corresponds to the last p rows of $(Q_j^{LS})^H \Lambda_j$. The SVD of the scaled residual $R_j^{\ell s} D_\varepsilon$ can be written as

$$R_j^{\ell s} D_\varepsilon = U_{\ell s} \Sigma V_{\ell s}^H,$$

so that the SVD of the scaled least squares residual is

$$\left(\Lambda_j - \underline{\mathscr{F}}_j Y_j\right) D_\varepsilon = \underbrace{Q_j^{LS}\begin{pmatrix} 0_{(n_j+k)\times p} & I_{n_j+k} \\ U_{\ell s} & 0_{p\times(n_j+k)} \end{pmatrix}}_{\text{Unitary}} \begin{pmatrix} \Sigma \\ 0_{(n_j+k)\times p} \end{pmatrix} V_{\ell s}^H.$$

In conclusion, the idea is that computing the full QR-factorization of the matrices involved in the least squares problems allows us to reuse its Q factor to compute the SVD of the least squares residual using a QR-SVD algorithm such that the actual SVD decomposition is performed on a $p \times p$ block $R_j^{\ell s} D_\varepsilon$, where $R_j^{\ell s}$ appears as in the right-hand side of (2.38), at each iteration. Note that this observation applies naturally to the IB-BGMRES [88] and IB-BGMRES-DR [6] algorithms as well.

2.5 Block flexible GMRES with partial convergence detection and deflated restarting

2.5.1 Block flexible Arnoldi with partial convergence detection

Starting from an orthonormal block vector $\mathbb{V}_1$ obtained from the reduced QR-factorization of the initial residual block (denoted as R_0 in this section)[1] $R_0 = B - AX_0 = \mathbb{V}_1\Lambda_1$, Algorithm 3 describes details about the block flexible Arnoldi process used to construct a pair of orthonormal basis. When no inexact breakdown occurs, i.e., $p_{j+1} = p_j = \ldots = p_1 = p$, the whole columns of W_j in step 10 of Algorithm 3 have been used to enlarge the search space, and then the block Arnoldi relation at the jth iteration is obtained as

$$A\mathscr{Z}_j = \mathscr{V}_j\mathscr{H}_j + [0_{n\times n_{j-1}}, \quad W_j] = \mathscr{V}_{j+1}\underline{\mathscr{H}}_j, \tag{2.48}$$

in which $\mathscr{Z}_j = [\mathscr{M}_1(\mathbb{V}_1), \ldots, \mathscr{M}_m(\mathbb{V}_j)]$, $\mathscr{V}_j = [\mathbb{V}_1, \ldots, \mathbb{V}_j] \in \mathbb{C}^{n\times n_j}$ $(n_j = j \times p)$ contains orthonormal columns and $\underline{\mathscr{H}}_j = \begin{bmatrix} \mathscr{H}_j \\ 0\ldots 0 \ H_{j+1,j} \end{bmatrix} \in \mathbb{C}^{n_{j+1}\times n_j}$ composed by square matrices $H_{j+1,j} \in \mathbb{C}^{p_j\times p_j}$ $(p_j = p)$ is a block upper Hessenberg matrix. The minimum residual norm solution in the affine space $X_0 + \text{Range}(\mathscr{Z}_j)$ can be written as $X_j = X_0 + \mathscr{Z}_j Y_j$ where

$$Y_j = \underset{Y\in\mathbb{C}^{n_j\times p}}{\text{argmin}} \|\tilde{\Lambda}_j - \underline{\mathscr{H}}_j Y\|_F$$

and $\tilde{\Lambda}_j = \mathscr{V}_{j+1}^H R_0 = (\Lambda_1, 0_{n_j\times p})^T$, the columns of $\tilde{\Lambda}_j$ are the components of the individual initial residual in the residual space $\mathscr{V}_{j+1}$.

When a partial convergence occurs up to iteration j in Algorithm 3, the dimension of the approximation space $\text{Range}(\mathscr{Z}_j)$ generated at the jth iteration is no longer equal to $j \times p$ but equal to $n_j = \sum_{i=1}^{j} p_i$ with $n_j < j \times p$. According to the partial convergence detection mechanism in IB-BGMRES [88], the block flexible Arnoldi with partial convergence detection [2] and Equation (2.9) developed by Robbé and Sadkane [88],

[1] Out of simplicity, the initial residual block in here is assumed to be of full column rank, while such assumption could be removed by introducing partial convergence detection in the initial residual block as the contents described in Section 2.4.1.

[2] The block flexible Arnoldi with partial convergence detection is obtained by changing the content in step 10 of Algorithm 2 into: Orthogonalize $A\mathscr{M}_j(\mathbb{V}_j)$ against previous block orthonormal vector $\mathscr{V}_j = [\mathbb{V}_1, \ldots, \mathbb{V}_j]$ as

$$W_j = A\mathscr{M}_j(\mathbb{V}_j) - \mathscr{V}_j\mathscr{L}_{1,1:j}, \text{ where } \mathscr{L}_{1,1:j} = \mathscr{V}_j^H(A\mathscr{M}_j(\mathbb{V}_j)) \text{ is a block column matrix.}$$

Algorithm 3 *Block flexible Arnoldi procedure with blockwise modified Gram-Schmidt orthogonalization:*

1: Given a nonsingular coefficient matrix $A \in \mathbb{C}^{n\times n}$, choose a unitary matrix $V_1 \in \mathbb{C}^{n\times p}$ with orthonormal columns
2: **for** $j = 1, 2, \ldots, m$ **do**
3: Choose a (possibly nonlinear) preconditioning operator $\mathscr{M}_j$
4: $Z_j = \mathscr{M}_j(\mathbb{V}_j)$
5: Compute $W_j = AZ_j$
6: **for** $i = 1, 2, \ldots, j$ **do**
7: $H_{i,j} = \mathbb{V}_i^H W_j$
8: $W_j = W_j - \mathbb{V}_i H_{i,j}$
9: **end for**
10: $W_j = \mathbb{V}_{j+1} H_{j+1,j}$ (reduced QR-factorization)
11: **end for**

the Equation (2.48) could be extended into

$$A\mathscr{Z}_j = \mathscr{V}_j \mathscr{H}_j + [\mathcal{Q}_{j-1}, \quad W_j], \tag{2.49}$$

where $\mathcal{Q}_{j-1} = [Q_1, \ldots, Q_{j-1}] \in \mathbb{C}^{n\times n_{j-1}}$ is rank deficient and accounts for all the abandoned directions.

In order to characterize a minimum norm solution in the space spanned by $\mathscr{Z}_j$ with Equation (2.49) we need to form an orthonormal basis of the space spanned by $[\mathscr{V}_j, \mathcal{Q}_{j-1}, W_j]$. This is performed by first orthogonalizing $\mathcal{Q}_{j-1}$ against $\mathscr{V}_j$, that is $\widetilde{\mathcal{Q}}_{j-1} = (I - \mathscr{V}_j \mathscr{V}_j^H)\mathcal{Q}_{j-1}$. Because $\mathcal{Q}_{j-1}$ is of low rank so is $\widetilde{\mathcal{Q}}_{j-1}$ that can be written as formula (2.11). Next W_j, that is already orthogonal to $\mathscr{V}_j$, is made to be orthogonal to P_{j-1} with $W_j - P_{j-1}E_j$ where $E_j = P_{j-1}^H W_j$; then one computes $\widetilde{W}_j D_j$ the reduced QR-factorization of $W_j - P_{j-1}E_j$. Eventually, the columns of the matrix $[\mathscr{V}_j, P_{j-1}, \widetilde{W}_j]$ form an orthonormal basis of the space spanned by $[\mathscr{V}_j, \mathcal{Q}_{j-1}, W_j]$.

With this new basis, Equation (2.49) writes

$$A\mathscr{Z}_j \quad = \quad \left[\mathscr{V}_j, [P_{j-1}, \widetilde{W}_j]\right] \underline{\tilde{\mathscr{F}}}_j, \tag{2.50}$$

where $\underline{\tilde{\mathscr{F}}}_j = \begin{bmatrix} \mathscr{L}_j \\ \widehat{\mathbb{H}}_j \end{bmatrix} \in \mathbb{C}^{(n_j+p)\times n_j}$ with $\widehat{\mathbb{H}}_j = \begin{bmatrix} \mathbb{G}_{j-1} & E_j \\ 0 & D_j \end{bmatrix} \in \mathbb{C}^{p\times n_j}$ (here the notation $\mathbb{H}_j$ with the wide-hat form is used for distinguishing from that already used in IB-BGCRO-DR case as appeared in Equation (2.14) and (2.41)) and $\mathscr{L}_j \in \mathbb{C}^{n_j\times n_j}$ owns the same details as described in formula (2.12), which is no longer a block upper Hessenberg as shown in the right-hand sides of Equation (2.48) as soon as a partial convergence occurs, i.e., $\exists \ell\ Q_\ell \neq 0$.

The numerical mechanism to select V_{j+1} out of $\left[P_{j-1}, \widetilde{W}_j\right]$ follows the same ideas as discussed in [6, 88] within the context of block GMRES. The governing idea consists in building the orthonormal basis for the directions that contribute the most to the individual residual norms and make them larger than the target threshold τ. Based on the SVD of the coordinate vector of the scaled least squares residual $(\tilde{\Lambda}_j - \underline{\tilde{\mathscr{F}}}_j Y_j)D_\varepsilon = \mathbb{U}_{1,L}\Sigma_1 \mathbb{V}_{1,R}^H +$

$\mathbb{U}_{2,L}\Sigma_2\mathbb{V}_{2,R}^H$ where D_ε is a diagonal matrix used to select the space expansion described in the Section 2.3, Σ_1 contains the singular values larger than the prescribed IB-threshold τ, they decompose $\mathbb{U}_{1,L} = \begin{pmatrix} \mathbb{U}_1^{(1)} \\ \mathbb{U}_1^{(2)} \end{pmatrix}$ in accordance with $\left[\mathscr{V}_j, [P_{j-1}, \widetilde{W}_j]\right]$, that is $\mathbb{U}_1^{(1)} \in \mathbb{C}^{n_j \times p_{j+1}}$ and $\mathbb{U}_1^{(2)} \in \mathbb{C}^{p \times p_{j+1}}$. Because, the objective is to construct an orthonormal basis we consider $[\mathbb{W}_1, \mathbb{W}_2]$ unitary so that $\text{Range}(\mathbb{W}_1) = \text{Range}(\mathbb{U}_1^{(2)})$. The new set of orthonormal vectors selected to expand the search space as formula (2.16), which contributes the most to the residual. We do not give the detailed calculation and refer to [88] for a complete description, but only state that via this decomposition the main terms that appear in Equation (2.50) can be computed incrementally by an alternative formulation:

$$A\mathscr{Z}_j = \mathscr{V}_{j+1}\underline{\mathscr{L}}_j + \widetilde{\mathcal{Q}}_j, \tag{2.51}$$

with $\underline{\mathscr{L}}_j = \begin{bmatrix} \mathscr{L}_\mathrm{j} \\ V_{j+1}\mathcal{Q}_{j-1}\ \mathrm{H_{j+1,j}} \end{bmatrix}$, where $\mathscr{L}_j = \begin{bmatrix} & H_{1,j} \\ \underline{\mathscr{L}}_{j-1} & \vdots \\ & H_{j,j} \end{bmatrix}$, the last block row of $\underline{\mathscr{L}}_j$ at next iteration $(j+1)$ is given by $\underline{\mathscr{L}}_{j+1,:} = \mathbb{W}_1^H\widehat{\mathbb{H}}_j$. The last block column of $\mathscr{L}_{j+1}$ results from the block flexible Arnoldi orthogonalization. The new compressed form of the abandoned direction $\widetilde{\mathcal{Q}}_j$ is given by the new orthonomal set of vectors

$$P_j = \left[P_{j-1}, \widetilde{W}_j\right] \mathbb{W}_2, \tag{2.52}$$

and the complementary part of V_{j+1} and their components in the space spanned by P_j are $\mathbb{G}_j = \mathbb{W}_2^H\widehat{\mathbb{H}}_j$.

Consequently, in one cycle of IB-BFGMRES-DR, once the maximum size of the space has been reached, we have

$$\begin{aligned}
A\mathscr{Z}_m &= \left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] \underline{\tilde{\mathscr{F}}}_m, && (2.53)\\
A\mathscr{Z}_m &= \mathscr{V}_{m+1}\underline{\mathscr{L}}_m + \widetilde{\mathcal{Q}}_m, && (2.54)\\
X_m &= X_0 + \mathscr{Z}_m Y_m, && (2.55)\\
R_m &= \left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] \left(\tilde{\Lambda}_m - \underline{\tilde{\mathscr{F}}}_m Y_m\right), && (2.56)\\
Y_m &= \underset{Y \in \mathbb{C}^{n_m \times p}}{\text{argmin}} \left\| \tilde{\Lambda}_m - \underline{\tilde{\mathscr{F}}}_m Y \right\|_F, \tilde{\Lambda}_m = [\Lambda_1^T, \quad 0_{p \times n_m}]^T.
\end{aligned}$$

2.5.2 Harmonic-Ritz vectors and residuals

We first illustrate how to compute the harmonic-Ritz vectors used for deflation as described in Proposition 5, and then discuss the relation between the linear system residuals and the residuals of harmonic-Ritz vectors at the restart of IB-BFGMRES-DR.

Proposition 5. *At the end of a cycle of IB-BFGMRES-DR, the updating of deflated restarting used in next cycle relies on the computation of k harmonic-Ritz vectors $Y_k = \mathscr{V}_m G_k$ of $A\mathscr{Z}_m\mathscr{V}_m^H$ with respect to* $\text{Range}(\mathscr{V}_m)$*, where each harmonic-Ritz pair $(\theta_i, \mathscr{V}_m g_i)$ computed at the end of cycle satisfies*

$$(\mathscr{L}_m + \mathscr{L}_m^{-H}\,\widehat{\mathbb{H}}_m^H\widehat{\mathbb{H}}_m)g_i = \theta_i\, g_i \quad \textit{for } 1 \le i \le n_m, \tag{2.57}$$

where $\mathscr{L}_m \in \mathbb{C}^{n_m \times n_m}$ *and* $\widehat{\mathbb{H}}_m \in \mathbb{C}^{p \times n_m}$.

Proof. According to Definition 1, each harmonic-Ritz pair $(\theta_i, \mathscr{V}_m g_i)$ satisfies

$$\forall w \in \text{Range}(A\mathscr{Z}_m \mathscr{V}_m^H \mathscr{V}_m) \quad w^H \left(A\mathscr{Z}_m \mathscr{V}_m^H \mathscr{V}_m g_i - \theta_i \mathscr{V}_m g_i\right) = 0,$$

which is equivalent to

$$(A\mathscr{Z}_m)^H \left(A\mathscr{Z}_m g_i - \theta_i \mathscr{V}_m g_i\right) = 0,$$

by the orthonormality of $\mathscr{V}_m$. Substituting (2.53) into the above one yields

$$\left(\left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] \underline{\tilde{\mathscr{F}}}_m\right)^H \left(\left[\mathscr{V}_i, [P_{m-1}, \widetilde{W}_m]\right] \underline{\tilde{\mathscr{F}}}_m g_i - \theta_i \mathscr{V}_m g_i\right) = 0. \tag{2.58}$$

Because of the structure of $\underline{\tilde{\mathscr{F}}}_m$ and the orthonormality of $[\mathscr{V}_m, P_{m-1}, \widetilde{W}_m]$, (2.58) becomes

$$(\mathscr{L}_m^H \mathscr{L}_m + \widehat{\mathbb{H}}_m^H \widehat{\mathbb{H}}_m) g_i = \theta_i \mathscr{L}_m^H g_i, \tag{2.59}$$

which completes the proof since $\mathscr{L}_m$ is assumed to be nonsingular. □

Assume $R_m^{LS} = \left(\tilde{\Lambda}_m - \underline{\tilde{\mathscr{F}}}_m Y_m\right) \in \mathbb{C}^{(n_m+p)\times p}$, the residual of linear system presented in Equation (2.56) could be simplified as

$$R_m = \left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] R_m^{LS} \in \mathbb{C}^{n \times p}. \tag{2.60}$$

Denote the corresponding residual of harmonic-Ritz vectors as $R_m^{(HR)}$ similarly, which owns form as

$$R_m^{(HR)} = A\mathscr{Z}_m G_k - \mathscr{V}_m G_k \text{ diag}(\theta_1, \ldots, \theta_k) \in \mathbb{C}^{n \times k}. \tag{2.61}$$

Given that both R_m and $R_m^{(HR)}$ are resided in the subspace $\text{Range}(\left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right]) \in \mathbb{C}^{n \times (n_m+p)}$ and are orthogonal to the same subspace $\text{Range}(A\mathscr{Z}_m) \in \mathbb{C}^{n \times n_m}$. Therefore, the residuals of linear system R_m and the residuals of harmonic-Ritz vectors $R_m^{(HR)}$ are in the same p-dimensional space denoted as $\text{Range}(A\mathscr{Z}_m)^\perp \cap \text{Range}(\left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right])$, which means there exists a matrix $\beta_{p\times k} \in \mathbb{C}^{p \times k}$ such that $R_m^{(HR)} = R_m \beta_{p\times k}$. According to (2.60) and (2.61), such collinear relationship between the linear system residuals and residuals of harmonic-Ritz vectors could be further described as the following formula

$$A\mathscr{Z}_m G_k \;=\; \left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] \underline{G} \begin{bmatrix} diag(\theta_1, \cdots, \theta_k) \\ \beta_{p\times k} \end{bmatrix}, \tag{2.62}$$

where $G_k = [g_1, \ldots, g_k] \in \mathbb{C}^{n_m \times k}$, $\underline{G} = \begin{bmatrix} G_k & R_m^{LS} \\ 0_{p\times k} & \end{bmatrix} \in \mathbb{C}^{(n_m+p)\times(k+p)}$, $\beta_{p\times k} = (\beta_1, \cdots, \beta_k) \in \mathbb{C}^{p\times k}$ and $\beta_i \in \mathbb{C}^p$ $(1 \le i \le k)$. Based on Equation (2.50) and the orthonormality of $\left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right]$, Equation (2.62) can be also expressed as

$$\underline{\tilde{\mathscr{F}}}_m G_k \;=\; \underline{G} \begin{bmatrix} diag(\theta_1, \cdots, \theta_k) \\ \beta_{p\times k} \end{bmatrix}, \tag{2.63}$$

which is the block form of (3.4) shown in [6, Lemma3.3].

2.5.3 Block flexible GMRES with partial convergence detection at restart

In this section, the forthcoming Theorem 2 will be presented to illustrate the flexible Arnoldi relation with partial convergence detection described in Equation (2.50) and (2.51) (or in Equation (2.53) and (2.54)) still hold at restart of IB-BFGMRES-DR. Firstly, let us denote $\underline{G} = Q_{\underline{G}} R_{\underline{G}}$ the reduced QR-factorization of $\underline{G}$ shown in Equation (2.63) and the reduced factors could be partitioned as

$$Q_{\underline{G}} = \begin{bmatrix} \Gamma_1 & \Gamma_2 \\ 0_{p\times k} & \end{bmatrix} \in \mathbb{C}^{(n_m+p)\times(k+p)}, \tag{2.64}$$

$$R_{\underline{G}} = \begin{bmatrix} \Theta_1 & \Theta_2 \\ 0_{p\times k} & \end{bmatrix} \in \mathbb{C}^{(k+p)\times(k+p)}, \tag{2.65}$$

with $\Gamma_1 = Q_{\underline{G}}(1:n_m, 1:k)$, $\Gamma_2 = Q_{\underline{G}}(:, k+1:k+p)$, $\Theta_1 = R_{\underline{G}}(1:k, 1:k)$, $\Theta_2 = R_{\underline{G}}(:, k+1:k+p)$ and

$$G_k = \Gamma_1 \Theta_1, \tag{2.66}$$

$$R_m^{LS} = Q_{\underline{G}} \Theta_2. \tag{2.67}$$

Theorem 2. *At each restart of IB-BFGMRES-DR, the initial block-flexible-Arnoldi-like relation* (2.50) *and* (2.51) *still hold in exact arithmetic as*

$$A\mathscr{Z}_1^{new} = \left[\mathscr{V}_1^{new}, [P_0, \widetilde{W}_1]^{new}\right] \underline{\tilde{\mathscr{F}}}_1^{new}, \tag{2.68}$$

$$A\mathscr{Z}_1^{new} = \mathscr{V}_2^{new} \underline{\mathscr{L}}_1^{new} + \widetilde{\mathcal{Q}}_1^{new}, \tag{2.69}$$

$$R_0^{new} = R_m = \left[\mathscr{V}_1^{new}, [P_0, \widetilde{W}_1]^{new}\right] \tilde{\Lambda}_1^{new} \text{ and } \tilde{\Lambda}_1^{new} = \Theta_2, \tag{2.70}$$

with

$$\begin{aligned}
\mathscr{Z}_1^{new} &= \mathscr{Z}_m \Gamma_1,\ \left[\mathscr{V}_1^{new}, [P_0, \widetilde{W}_1]^{new}\right] = \left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] Q_{\underline{G}}, \\
\mathscr{V}_1^{new} &= \mathscr{V}_m \Gamma_1,\ [P_0, \widetilde{W}_1]^{new} = \left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] \Gamma_2, \\
\underline{\tilde{\mathscr{F}}}_1^{new} &= \begin{bmatrix} \mathscr{L}_1^{\text{new}} \\ \widehat{\mathbb{H}}_1^{new} \end{bmatrix} \text{ and } \mathscr{L}_1^{new} = \Gamma_1^H \mathscr{L}_m \Gamma_1,\ \widehat{\mathbb{H}}_1^{new} = \Gamma_2^H \underline{\tilde{\mathscr{F}}}_m \Gamma_1, \\
\mathbb{V}_2^{new} &= [P_0, \widetilde{W}_1]^{new} \mathbb{W}_1,\ \mathscr{V}_2^{new} = [\mathscr{V}_1^{new}, \mathbb{V}_2^{new}], \\
\mathscr{L}_{2,:}^{new} &= \mathbb{W}_1^H \widehat{\mathbb{H}}_1^{new},\ \underline{\mathscr{L}}_1^{new} = \begin{bmatrix} \mathscr{L}_1^{new} \\ \mathscr{L}_{2,:}^{new} \end{bmatrix}, \\
P_1^{new} &= [P_0, \widetilde{W}_1]^{new} \mathbb{W}_2,\ \mathbb{G}_1^{new} = \mathbb{W}_2^H \widehat{\mathbb{H}}_1^{new},\ \widetilde{\mathcal{Q}}_1^{new} = P_1^{new} \mathbb{G}_1^{new},
\end{aligned}$$

where $\mathbb{W}_1$ *and* $\mathbb{W}_2$ *satisfy*

$$\text{Range}(\mathbb{W}_1) = \text{Range}(\mathbb{U}_1^{new(2)}) \text{ with } \mathbb{U}_{1,L}^{new} = \begin{bmatrix} \mathbb{U}_1^{new(1)} \\ \mathbb{U}_1^{new(2)} \end{bmatrix} \text{ and } [\mathbb{W}_1\, \mathbb{W}_2] \text{ is unitary}$$

with

$$(\tilde{\Lambda}_1^{new} - \underline{\tilde{\mathscr{F}}}_1^{new} Y_1^{new}) D_\epsilon = \mathbb{U}_{1,L}^{new} \Sigma_1^{new} \mathbb{V}_{1,R}^{new\,H} + \mathbb{U}_{2,L}^{new} \Sigma_2^{new} \mathbb{V}_{2,R}^{new\,H},$$

where $\sigma_{min}(\Sigma_1^{new}) \geq 1 \geq \sigma_{max}(\Sigma_2^{new})$, *the SVD to detect partial convergence in the restarting scaled least squares residual block where*

$$Y_1^{new} = \underset{Y \in \mathbb{C}^{n_1 \times p}}{\text{argmin}} \left\| \tilde{\Lambda}_1^{new} - \underline{\tilde{\mathscr{F}}}_1^{new} Y \right\|_F.$$

Proof. Starting from the relationship between residual and harmonic-Ritz vectors as shown in Equation (2.62), let's substitute $\underline{G}$ by these reduced factors $Q_{\underline{G}}$ in Equation (2.64) and $R_{\underline{G}}$ in Equation (2.65) obtained by its reduced QR-factorization and change G_k by relation (2.66), then we have

$$A\mathscr{Z}_m\Gamma_1 = \left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] Q_{\underline{G}} R_{\underline{G}} \begin{bmatrix} diag(\theta_1, \cdots, \theta_k) \\ \beta_{p\times k} \end{bmatrix} \Theta_1^{-1}$$

by the nonsingularity of Θ_1, which could be rewritten as

$$A\mathscr{Z}_m\Gamma_1 = \left[\mathscr{V}_m\Gamma_1, [\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]]\Gamma_2\right] R_{\underline{G}} \begin{bmatrix} diag(\theta_1, \cdots, \theta_k) \\ \beta_{p\times k} \end{bmatrix} \Theta_1^{-1} \tag{2.71}$$

because of the partition of $Q_{\underline{G}}$ shown in (2.64). Then, repeating the same processes described above, the corresponding formula (2.63) could also be reformed as

$$\underline{\tilde{\mathscr{F}}}_m\Gamma_1 = Q_{\underline{G}} R_{\underline{G}} \begin{bmatrix} diag(\theta_1, \cdots, \theta_k) \\ \beta_{p\times k} \end{bmatrix} \Theta_1^{-1},$$

from which, we have

$$R_{\underline{G}} \begin{bmatrix} diag(\theta_1, \cdots, \theta_k) \\ \beta_{p\times k} \end{bmatrix} \Theta_1^{-1} = Q_{\underline{G}}^H \underline{\tilde{\mathscr{F}}}_m\Gamma_1.$$

According to the structure of $Q_{\underline{G}}$ and $\mathscr{F}_m$ as shown in Equation (2.64) and (2.50), we obtain

$$R_{\underline{G}} \begin{bmatrix} diag(\theta_1, \cdots, \theta_k) \\ \beta_{p\times k} \end{bmatrix} \Theta_1^{-1} = \begin{bmatrix} \Gamma_1^{\mathrm{H}} \mathscr{L}_{\mathrm{m}} \Gamma_1 \\ \Gamma_2^H \underline{\tilde{\mathscr{F}}}_m \Gamma_1 \end{bmatrix}. \tag{2.72}$$

If we denote

$$\begin{aligned} \mathscr{Z}_1^{new} &= \mathscr{Z}_m\Gamma_1,\ \mathscr{V}_1^{new} = \mathscr{V}_m\Gamma_1,\ [P_0, \widetilde{W}_1]^{new} = \left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right]\Gamma_2, \\ \mathscr{L}_1^{new} &= \Gamma_1^H \mathscr{L}_m\Gamma_1,\ \widehat{\mathbb{H}}_1^{new} = \Gamma_2^H \underline{\tilde{\mathscr{F}}}_m\Gamma_1, \underline{\tilde{\mathscr{F}}}_1^{new} = \begin{bmatrix} \mathscr{L}_1^{\text{new}} \\ \widehat{\mathbb{H}}_1^{new} \end{bmatrix}, \end{aligned}$$

and substitute Equation (2.72) into (2.71), then Equation (2.68) is proven.

Next, show that equality (2.69) holds. Given $[\mathbb{W}_1\, \mathbb{W}_2]$ is unitary, we have

$$[P_0, \widetilde{W}_1]^{new} = [P_0, \widetilde{W}_1]^{new}[\mathbb{W}_1\mathbb{W}_1^H + \mathbb{W}_2\mathbb{W}_2^H],$$

and substituting this into Equation (2.68) gives

$$\begin{aligned}
A\mathscr{Z}_1^{new} &= \left[\mathscr{V}_1^{new}, [P_0, \widetilde{W}_1]^{new}[\mathbb{W}_1\mathbb{W}_1^H + \mathbb{W}_2\mathbb{W}_2^H]\right] \begin{bmatrix} \mathscr{L}_1^{\text{new}} \\ \widehat{\mathbb{H}}_1^{new} \end{bmatrix}, \\
&= \mathscr{V}_1^{new}\mathscr{L}_1^{new} + [P_0, \widetilde{W}_1]^{new}[\mathbb{W}_1\mathbb{W}_1^H + \mathbb{W}_2\mathbb{W}_2^H]\widehat{\mathbb{H}}_1^{new}, \\
&= \mathscr{V}_1^{new}\mathscr{L}_1^{new} + [P_0, \widetilde{W}_1]^{new}\mathbb{W}_1\mathbb{W}_1^H\widehat{\mathbb{H}}_1^{new} + [P_0, \widetilde{W}_1]^{new}\mathbb{W}_2\mathbb{W}_2^H\widehat{\mathbb{H}}_1^{new}, \\
&= \mathscr{V}_1^{new}\mathscr{L}_1^{new} + \mathbb{V}_2^{new}\mathscr{L}_{2,:}^{new} + P_1^{new}\mathbb{G}_1^{new}, \\
&= [\mathscr{V}_1^{new}\, V_2^{new}] \begin{bmatrix} \mathscr{L}_1^{new} \\ \mathscr{L}_{2,:}^{new} \end{bmatrix} + P_1^{new}\mathbb{G}_1^{new},
\end{aligned}$$

which is relation (2.69).

From Equation (2.60) and (2.67), at restart we have

$$\begin{aligned}
R_0^{new} &= R_m = \left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] R_m^{LS} \\
&= \left[\mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]\right] Q_{\underline{G}}\Theta_2 = \left[\mathscr{V}_1^{new}, [P_0, \widetilde{W}_1]^{new}\right] \tilde{\Lambda}_1^{new}.
\end{aligned}$$

This completes the proof. □

2.6 Numerical experiments

In the following sections we illustrate different numerical features of the novel algorithm introduced above. For the sake of comparison, in some of the experiments we also display results of closely related block methods such as BGCRO-DR [81, 103, 127] or IB-BGMRES-DR [6]. All the numerical experiments have been run using a MATLAB prototype, so that the respective performances of the algorithms are evaluated in terms of the number of matrix-vector products, denoted as *#mvps* (and preconditioner applications in the preconditioned case) required to converge.

For each set of blocks of right-hand sides, referred to as a family, the block initial guess is equal to $0 \in \mathbb{C}^{n\times p}$, where p is the number of right-hand sides. The block right-hand side $B = [b^{(1)}, b^{(2)}, \ldots, b^{(p)}] \in \mathbb{C}^{n\times p}$ is composed of p linearly independent vectors generated randomly (using the same random seed when block methods are compared). While any part of the spectrum could be considered to define the recycling space we consider for all the experiments the approximate eigenvectors associated with the k smallest approximate eigenvalues in magnitude. The maximum dimension of the search space in each cycle is set to be $m_d = 15 \times p$. To illustrate the potential benefit of IB-BGCRO-DR when compared to another block solver, we consider the overall potential gain when solving a sequence of ℓ families defined as

$$\text{Gain } (\ell) = \frac{\sum_{s=1}^{\ell} \#mvps \text{ (method)}^{(s)}}{\sum_{s=1}^{\ell} \#mvps \text{ (IB-BGCRO-DR)}^{(s)}}. \tag{2.73}$$

2.6.1 Rayleigh-Ritz and harmonic-Ritz approaches for recycling subspace

To illustrate the flexibility of subspace recycling in IB-BGCRO-DR as discussed in Section 2.2.4, both the harmonic-Ritz (HR) and Rayleigh-Ritz (RR) projections are considered to construct the recycling subspace; the correspond algorithms are referred to as IB-BGCRO-DR(HR) and IB-BGCRO-DR(RR). Following the spirit of the test examples considered in [72] we consider bidiagonal matrices of size 5000 with upper diagonal unit so that their spectrum is defined by their diagonal entries; we denote them Matrix 1 and Matrix 2. Matrix 1 has diagonal entries $0.1, 1, 2, 3, \ldots, 4999$ and Matrix 2 has diagonal entries $10.1, 10.2, \ldots, 20, 21, \ldots, 4920$. We consider experiments with a series of linear systems and each one with $p = 20$ given right-hand sides, the size of the recycled space $k = 30$, and the maximal dimension of the search space $m_d = 300$. In the left plots of Figure 2.2 we display the convergence histories of the p backward errors as a function of the number of matrix-vector products ($\#mvps$) for the first three consecutive families. On the right two plots of Figure 2.2 we depict $\#mvps$ for each of the 30 families. For Matrix 1, one can observe that the HR-projection captures a space that slows down the initial convergence rate once the first family has been solved; that is, for families 2 and 3 the converge histories do not exhibit anymore any plateau. On that example the RR-projection does not capture a recycling space that helps much the convergence as the three convergence histories exhibit very similar pattern. For Matrix 2, both RR and HR projections work pretty much the same. In Table 2.1, we report the total required $\#mvps$ for the two matrix examples for 3 and 30 families. Those results do not attempt to highlight that one projection is superior to the other one, but simply illustrate the flexibility of the GCRO approach to accommodate both. The selection or discussion of the best suited projection method is out of the scope of this manuscript.

# families	Matrix	Method	$\#mvps$
3	Matrix 1	IB-BGCRO-DR(HR)	**7182**
		IB-BGCRO-DR(RR)	7583
30	Matrix 1	IB-BGCRO-DR(HR)	**68262**
		IB-BGCRO-DR(RR)	71709
3	Matrix 2	IB-BGCRO-DR(HR)	13981
		IB-BGCRO-DR(RR)	**13430**
30	Matrix 2	IB-BGCRO-DR(HR)	138247
		IB-BGCRO-DR(RR)	**136812**

Table 2.1 – Numerical results of IB-BGCRO-DR with recycling subspace generated by RR or HR-projection for Matrix 1 and Matrix 2 ($p = 20$, $m_d = 300$ and $k = 30$).

In the rest of this chapter, only the HR projection is considered to build recycling subspace used in the GCRO-DR like methods. Besides, the bidiagonal Matrix 1 is chosen as the constant left-hand sides in the following Section 2.6.2-2.6.6, in which the related parameters are likewise set to be $p = 20$, $k = 30$ and $m_d = 300$ defaultly.

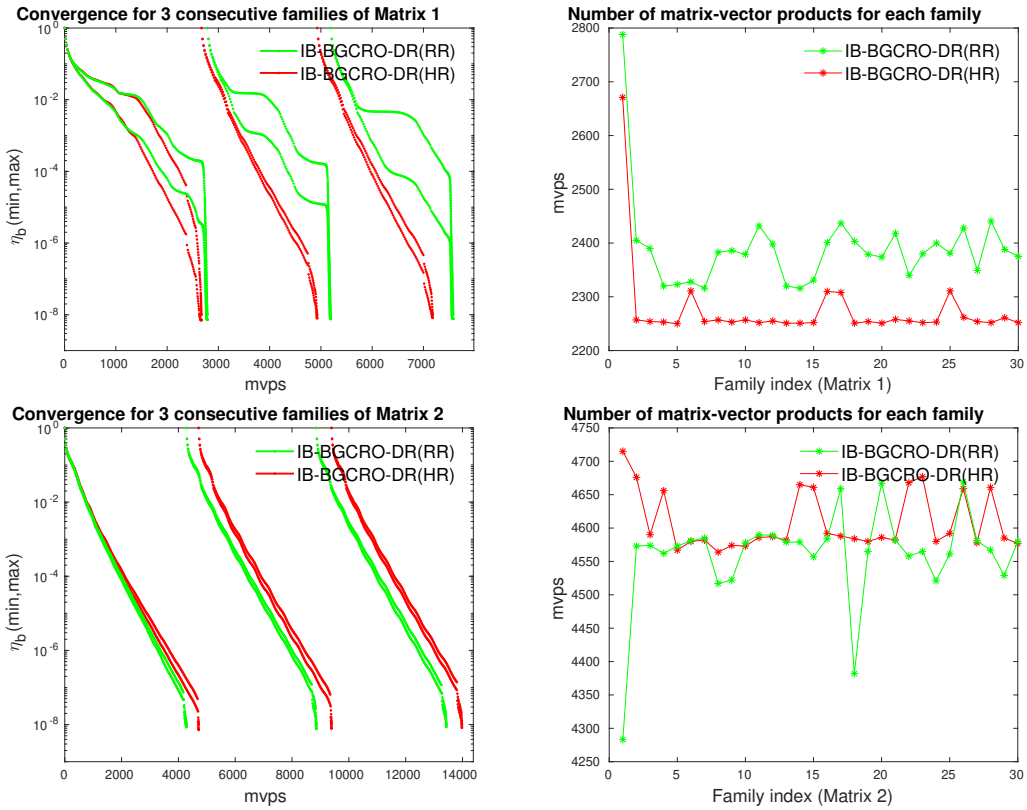


Figure 2.2 – Comparison behavior of IB-BGCRO-DR with recycling subspace developed by RR or HR-projection on bidiagonal Matrix 1 and Matrix 2 ($p = 20$, $m_d = 300$ and $k = 30$) as described in Section 2.6.1. Left: convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for 3 consecutive families. Right: consumed $\#mvps$ versus family index.

2.6.2 Comparison of two different partial convergence detection thresholds

According to the inequality (2.35) in Section 2.3.1, the partial convergence threshold we adopted in this work satisfy $\tau = 1$. Specifically, assume that the solutions corresponding to each single right-hand side $\eta_b(x_j^{(i)})$ as described in left-hand side of (2.35) converge to the same convergence threshold ε, when p partial convergences have occurred, we have,

$$\frac{||b^{(i)} - Ax_j^{(i)}||}{||b^{(i)}|| \times \varepsilon} \leq \frac{||B - AX_j||}{||b^{(i)}|| \times \varepsilon} \leq \left\| \left(\Lambda_1 - \underline{\mathscr{F}}_j Y_j\right) D_\varepsilon \right\| \leq \tau \text{ for } \forall i \in \{1, \ldots, p\}, \qquad (2.74)$$

where $D_\varepsilon = \varepsilon^{-1}$ diag $(||b^{(1)}||^{-1}, \cdots, ||b^{(p)}||^{-1}) \in \mathbb{C}^{p \times p}$ with $\tau = 1$. Comparing $\tau = 1$ appeared in the right-hand sides of (2.74) with the original version of IB-threshold $\tau = \varepsilon \min_{i=1,\ldots,p} ||b^{(i)}||$ described in the right-most part of inequality (3.17) of [6, Section 3.3] (which is specially denoted as min-IB-threshold in here for distinguishing it from $\tau = 1$), we conclude that the IB-variants (like IB-BGCRO-DR or IB-BGMRES-DR) with the IB-threshold $\tau = 1$ show clear benefit especially when the norm of each single right-hand

side $||b^{(i)}||$ is of very different magnitude, which may let the $\min_{i=1,\ldots,p} ||b^{(i)}||$ be quite different from some others that with larger norms thus further leads to the min-IB-threshold fails to detect IB of these columns effectively. This is illustrated by the results shown in Figure 2.3, in which the min-IB-BGCRO-DR refers to the IB-BGCRO-DR algorithm with the min-IB-threshold $\tau = \varepsilon \min_{i=1,\ldots,p} ||b^{(i)}||$, and IB-BGCRO-DR is the one with the IB-threshold $\tau = 1$ proposed and adopted in this chapter.

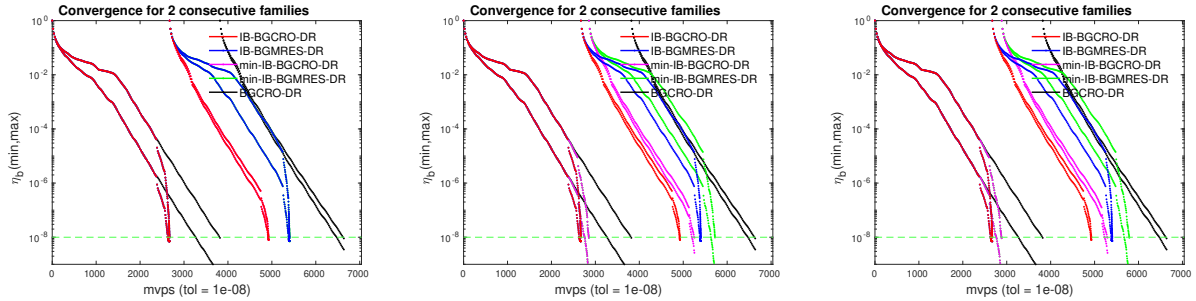


Figure 2.3 – Convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for Section 2.6.2 with different scaling size of the columns of the right-hand sides. Comparison the two IB-variants with two different form of IB-threshold by solving Matrix 1 ($p = 20$, $m_d = 300$, $\varepsilon = 10^{-8}$ and $k = 30$). Left: $B = \texttt{rand}(n, p)$. Middle: $B = \texttt{rand}(n, p)$ and then multiply 20 to the first $p/2$ columns of B. Right: the same as the Middle case except for multiplying by 50 instead.

2.6.3 Benefits of recycling between the families

To illustrate the benefits of recycling spectral information from one family to the next as well as the computational savings due to the partial convergence detection mechanism, we first report on experiments with BGCRO-DR, IB-BGCRO-DR and IB-BGMRES-DR on a series of linear systems with constant left-hand side. We consider experiments with a family size $p = 20$ and a recycled space size $k = 30$, and where the maximal dimension of the search space is $m_d = 300$.

In the left plot of Figure 2.4 we display the convergence histories for solving two consecutive families with the η_b-based stopping criterion. Several observations can be made. Because IB-BGMRES-DR, IB-BGCRO-DR and BGCRO-DR do not have a deflation space to start with for the first family, the convergence histories of these three solvers overlap as long as no partial convergence is detected. After this first partial convergence, the convergence rate of IB-BGCRO-DR and IB-BGMRES-DR becomes faster (in terms of $\#mvps$) than that of BGCRO-DR, and the former two convergence histories mostly overlap as the two IB solvers remain mathematically equivalent. For the second and subsequent families, the capability to start with a deflation space shows its benefit for BGCRO-DR and IB-BGCRO-DR. It is because IB-BGMRES-DR needs a few restarts to capture this spectral information again and to refine it in its subsequent search spaces construction process; eventually it exhibits a convergence rate similar to the BGCRO-DR counterpart. For the sake of comparison and to illustrate the benefit of the partial convergence detection we also display the convergence histories of BGCRO-DR which always requires more $\#mvps$ compared to its IB counterpart. Those extra $\#mvps$

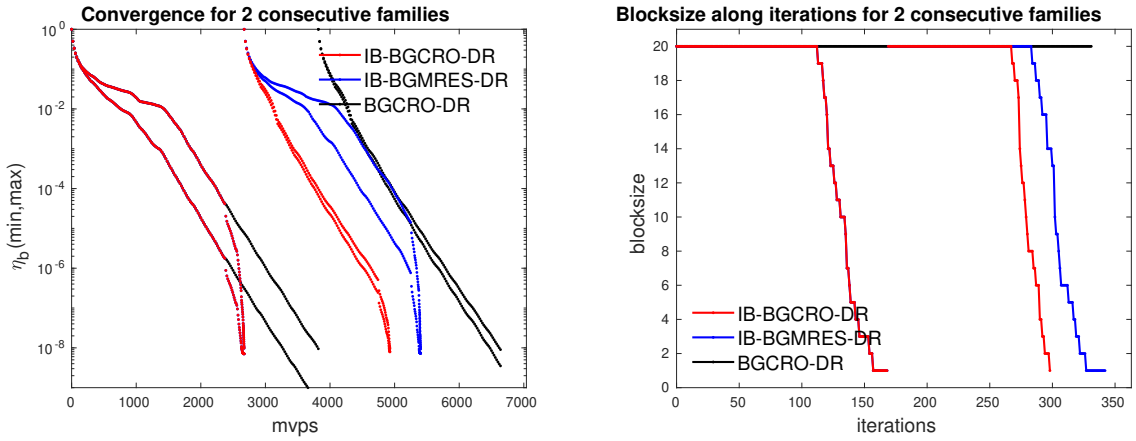


Figure 2.4 – Comparison history for Section 2.6.3. IB-BGCRO-DR with BGCRO-DR and IB-BGMRES-DR by solving Matrix 1 ($p = 20$, $m_d = 300$ and $k = 30$). Left: convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for 2 consecutive families. Right: varying blocksize (i.e., p_j) along the iterations.

mostly enable us to improve the solution quality for some right-hand sides beyond the targeted accuracy.

To visualize the effect of the partial convergence detection, we report in the right plot of Figure 2.4 the size of search space expansion p_j as a function of the iterations. Because BGCRO-DR does not implement the partial convergence detection, its search space is increased by $p = 20$ at each iteration. For the other two block IB-solvers, the block size monotonically decreases to 1. Note that the partial convergence detection is implemented in the initial (least squares) residual block in IB-BGCRO-DR, and thus its block size does not jump back to the original block size p at restart. By construction, IB-BGMRES-DR implements the partial convergence detection at restart so that the same observation applies.

# families	Method	$\#mvps$	$\#iter$
2	BGCRO-DR	6640	332
	IB-BGMRES-DR	5404	343
	IB-BGCRO-DR	**4928**	**299**
20	BGCRO-DR	56940	2847
	IB-BGMRES-DR	53772	3454
	IB-BGCRO-DR	**45652**	**2637**

Table 2.2 – Numerical results in both terms of $\#mvps$ and $\#iter$ for Section 2.6.3 with Matrix 1 ($p = 20$, $m_d = 300$ and $k = 30$).

A summary of the $\#mvps$ and the number of block iterations (referred to as $\#iter$) is given in Table 2.2 that shows the benefit of using IB-BGCRO-DR.

Note that we introduced partial convergence detection in the initial residual block for the proposed solver as described in Section 2.4.1. However, what would happen if we skip it for the first initial residual block? Let's denote the solver with partial convergence detection after the initial residual block as IBa-BGCRO-DR, where IBa stands for carrying

out inexact breakdown after the initial iteration (or without partial convergence detection in the initial residual block), for contrasting with IB-BGCRO-DR with partial convergence detection in the initial residual block, which with updating right-hand sides of the least squares problem as shown in Equation (2.44). From the pseudocode of IB-BGCRO-DR described in Algorithm 2, the corresponding pseudocode for IBa-BGCRO-DR could be deduced similarly by letting all the columns of the initial residual block be the initial Arnoldi basis $\mathbb{V}_1^{new}$ with p columns thus P_0^{new} is empty in this case, and by replacing the varying right-hand sides Λ_j shown in Proposition 4 into the one that with simple version as $\Lambda_j = \left[0_{p\times k}, \Lambda_1^T, 0_{p\times n_j}\right]^T$. Figure 2.5 displays the results of adding the performance of IBa-BGCRO-DR to Figure 2.4 to illustrate the benefit of introducing partial convergence detection in the initial residual block, i.e., reduce $\#mvps$ by avoiding the block size of search space jumps back to the original block size p at restart.

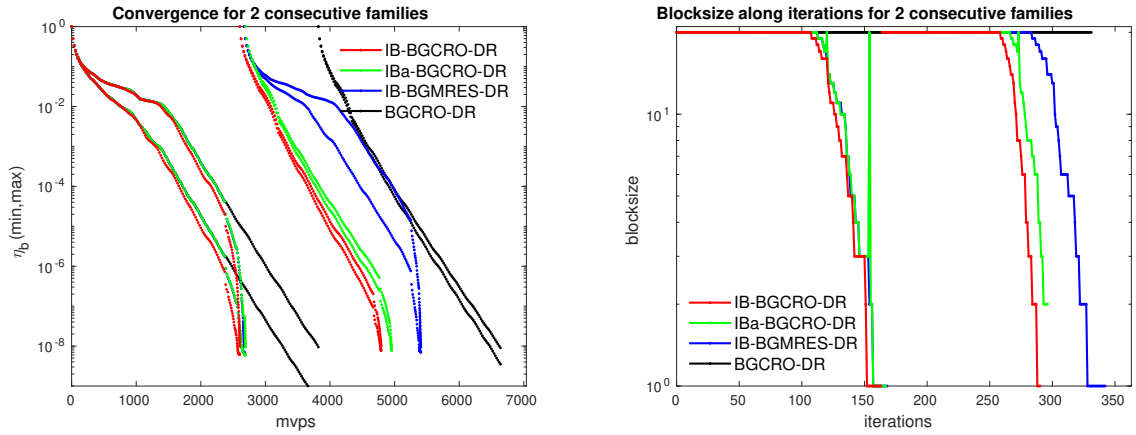


Figure 2.5 – Comparison of IB-BGCRO-DR with IBa-BGCRO-DR, BGCRO-DR and IB-BGMRES-DR by solving bidiagonal Matrix 1 ($p = 20$, $m_d = 300$ and $k = 30$). Left: convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for 2 consecutive families. Right: varying blocksize along iterations.

2.6.4 Subspace expansion governed by convergence criterion $\eta_{A,b}$

In this section we show the capability of the novel subspace expansion policy to drive the individual backward errors $\eta_{A,b}$ down to different accuracies and its benefit with respect to the original BGCRO-DR method. In Figure 2.6, we display the convergence histories of the IB and IB-free methods for three different convergence thresholds, from the less stringent on the left to the most stringent on the right. We can first observe that the first iteration, where the partial convergence detection starts to act, depends on the targeted accuracy as can be expected from the corresponds threshold on the singular values of the least squares residual. The second interesting observation is that IB-BGCRO-DR is able to decrease $\eta_{A,b}$ to a very low value close to the machine epsilon, that is $\mathcal{O}(10^{-16})$. This latter result mostly reveals the orthogonality quality of the residual space basis computed by (BMGS $\circ$ HouseQR) in the block Arnoldi implementation and the re-orthogonalization using MGS between all the columns of the recycling subspace C_k and the initial block Arnoldi basis at restart. This ensures that the least squares residual norms are quite close to the linear system residual ones. This latter fact ensures the relevance of the space

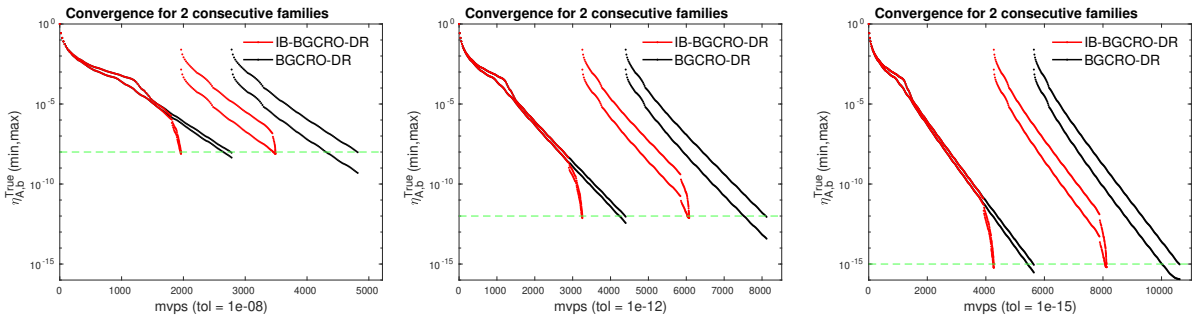


Figure 2.6 – Convergence histories of the largest/smallest $\eta_{A,b^{(i)}}(x_j^{(i)})$ at each $\#mvps$ for 2 consecutive families for Section 2.6.4 with different convergence thresholds. Comparison of IB-BGCRO-DR with BGCRO-DR by solving Matrix 1 ($p = 20$, $m_d = 300$ and $k = 30$).

expansion policy that monitors the linear system residual norms through the least squares residual ones. To illustrate the orthonormal quality of the basis $\widehat{\mathscr{V}}_{j+1} = \left[C_k, \mathscr{V}_j, [P_{j-1}, \widetilde{W}_j]\right]$, we display in Figure 2.7 the loss of orthogonality along $\#mvps$ that is defined by

$$\text{Loss-Orth} = \left\| \widehat{\mathscr{V}}_{j+1}^H \widehat{\mathscr{V}}_{j+1} - I_{j+1} \right\|. \tag{2.75}$$

In a quite similar manner to MGS-GMRES that is backward-stable [78], it can be observed that the loss of orthogonality mostly appears when the solutions of the linear systems converge. Note that without the re-orthogonalization at restart, the loss of orthogonality tends to accumulate along with restart which prevents the value of Loss-Orth to be close to the machine epsilon. We refer the reader to Figure 2.8 for the corresponding results without applying re-orthogonalization to all the columns of $[C_k, [\mathbb{V}_1, P_0]]$ at restart.

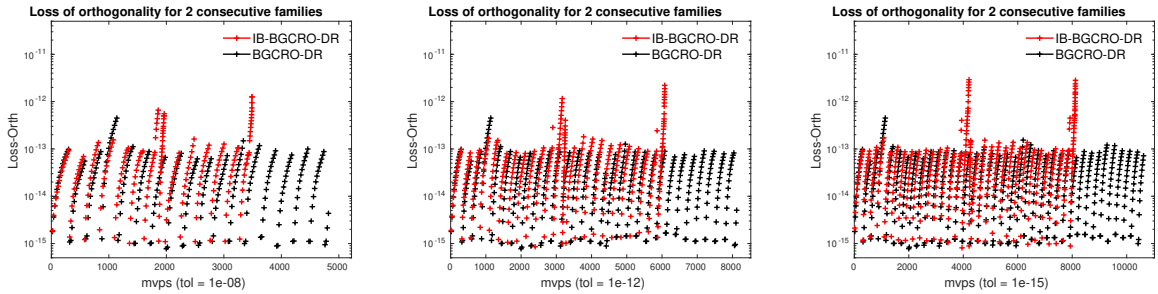


Figure 2.7 – Loss-Orth defined in Equation (2.75) of GCRO-variants with stopping criterion based on $\eta_{A,b^{(i)}}(x_j^{(i)})$ at each $\#mvps$ for 2 consecutive families for Section 2.6.4 with different convergence thresholds. Comparison of IB-BGCRO-DR with BGCRO-DR for solving Matrix 1 ($p = 20$, $m_d = 300$ and $k = 30$).

2.6.5 Subspace expansion policy for individual convergence thresholds for η_b

To illustrate this feature, we consider a family of p right-hand sides and a convergence threshold 10^{-4} for the first $p/2$ right-hand sides and 10^{-8} for the last $p/2$ ones. To estimate

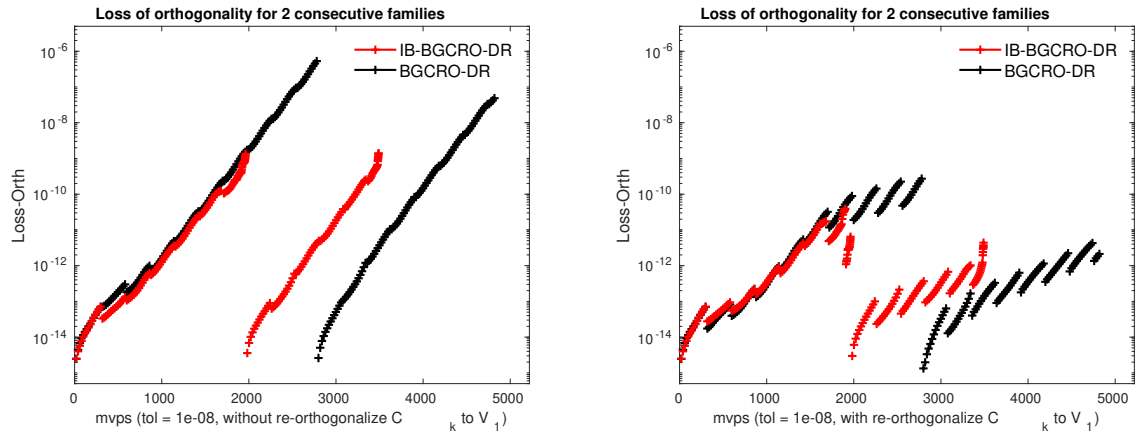


Figure 2.8 – Similar case to the left plot of Figure 2.7 but with/without re-orthogonalizing C_k to $\mathbb{V}_1$ by MGS at restart of BGCRO-DR and IB-BGCRO-DR.
*Note that the above mentioned re-orthogonalizing of C_k to $\mathbb{V}_1$ described in the right plot cannot ensure the re-orthogonality of all the columns of C_k (or $\mathbb{V}_1$) thus cannot obtain the results as shown in left plot of Figure 2.7 that with applying re-orthogonalization to all the columns of $[C_k, [\mathbb{V}_1, P_0]]$ at restart.

of the computational benefit of this feature, we also compare with calculations where all the right-hand sides are solved with the most stringent threshold, that is 10^{-8}. In the left part of Figure 2.9, we display the convergence histories for 3 successive families. The variant that controls the individual threshold is denoted as IB-BGCRO-DR-VA, where VA stands for variable accuracy. It can be seen that the numerical feature works well and that the envelope of the backward errors has the expected shape, that is, the minimum backward error decrease to 10^{-8} while the maximum one (associated with the first $p/2$ solutions) only decrease to 10^{-4}. If we compare the convergence histories of IB-BGCRO-DR and IB-BGCRO-DR-VA, it can be seen that the slope of IB-BGCRO-DR-VA is deeper than that of IB-BGCRO-DR once the first $p/2$ solutions have converged; after this point IB-BGCRO-DR-VA somehow focuses on the new directions (produced by $\#mvps$ given for the x-axis) to reduce the residual norms of the remaining $p/2$ solutions that have not yet converged. The right plot of Figure 2.9 shows the computational gain induced by the individual control of the accuracy compared to the situation where all the right-hand sides would have been solved to the most stringent stopping criterion threshold if this feature were not designed. In this case the individual monitoring of the convergence saves around 45% of $\#mvps$ in this example. Those results are summarized in Table 2.3.

# families	Method	$\#mvps$	$\#iter$
3	IB-BGCRO-DR	7182	428
	IB-BGCRO-DR-VA	**5119**	**395**
30	IB-BGCRO-DR	68263	3932
	IB-BGCRO-DR-VA	**47143**	**3566**

Table 2.3 – Numerical results of IB-BGCRO-DR with fixed/varying accuracy for each right-hand side in terms of $\#mvps$ and $\#iter$ for Section 2.6.5, where the coefficient matrix is Matrix 1 ($p = 20$, $m_d = 300$ and $k = 30$).

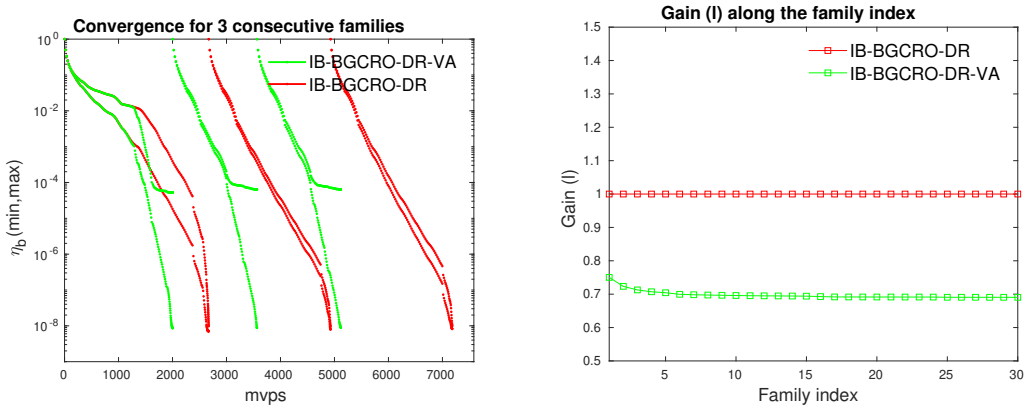


Figure 2.9 – Comparison of IB-BGCRO-DR to IB-BGCRO-DR-VA for Section 2.6.5 with Matrix 1 ($p = 20$, $m_d = 300$ and $k = 30$). Left: convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for 3 consecutive families. Right: gain (ℓ) defined in (2.73) of IB-BGCRO-DR-VA to IB-BGCRO-DR versus family index.

We refer the reader to Figure 2.10 and Table 2.4 for an illustration of extending such individual control to the block solver IB-BGMRES-DR that can also accommodate this feature.

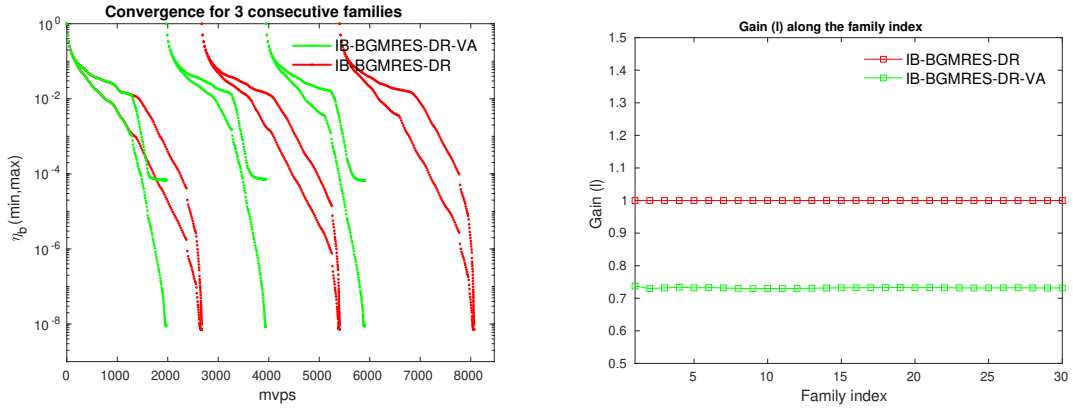


Figure 2.10 – Comparison of IB-BGMRES-DR to IB-BGMRES-DR-VA for Section 2.6.5 with Matrix 1 ($p = 20$, $m_d = 300$ and $k = 30$). Left: convergence histories of largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for 3 consecutive families. Right: gain (ℓ) defined in (2.73) of IB-BGMRES-DR-VA to IB-BGMRES-DR versus family index.

2.6.6 Expansion policy governed by computational performance

As discussed in Section 2.3.3, only a subset of the candidate directions exhibited by the partial convergence detection mechanism can be eventually selected to expand the search space at the next block iteration; we denote this maximum size as p^{CB} and refer to this variant as IB-BGCRO-DR-CB, where CB stands for computational blocking. In Table 2.5 we show the effect of this algorithmic parameter on $\#mvps$ and $\#iter$ for the

# families	Method	*#mvps*	*#iter*
3	IB-BGMRES-DR	8066	515
	IB-BGMRES-DR-VA	**5903**	**490**
30	IB-BGMRES-DR	80717	5191
	IB-BGMRES-DR-VA	**59069**	**4957**

Table 2.4 – Numerical results of IB-BGMRES-DR with fixed/varying accuracy for each right-hand side in terms of *#mvps* and *#iter* for Section 2.6.5, where the coefficient matrix is Matrix 1 ($p = 20$, $m_d = 300$ and $k = 30$)

solutions of 3 and 30 families with Matrix 1 when p^{CB} varies from 1 to 15 for a number of right-hand sides $p = 20$. Generally, the smaller p^{CB} is, the smaller *#mvps*, but the larger *#iter*. Although reported only on one example this trend has been observed in all our numerical experiments. Depending on the computational efficiency or cost of the *#mvps* with respect to the computational weight of the least squares problem and SVD of the scaled least squares residual, this gives opportunities to monitor the overall computational effort needed to complete the solution.

# families	Method	*#mvps*	*#iter*
3	IB-BGCRO-DR	7182	**428**
	IB-BGCRO-DR-CB ($p^{CB} = 15$)	**6934**	467
	IB-BGCRO-DR-CB ($p^{CB} = 10$)	6941	668
	IB-BGCRO-DR-CB ($p^{CB} = 5$)	6968	1312
	IB-BGCRO-DR-CB ($p^{CB} = 1$)	6966	6444
30	IB-BGCRO-DR	68262	**3932**
	IB-BGCRO-DR-CB ($p^{CB} = 15$)	**65364**	4303
	IB-BGCRO-DR-CB ($p^{CB} = 1$)	65823	60836

Table 2.5 – Numerical results of IB-BGCRO-DR and IB-BGCRO-DR-CB for $p^{CB} = 1, 5, 10, 15$ in terms of *#mvps* and *#iter* for Section 2.6.6, where the coefficient matrix is Matrix 1 with $p = 20$, $m_d = 300$ and $k = 30$.

As in previous subsections, we note that this subspace expansion policy is also applicable to IB-BGMRES-DR. We refer the reader to Figure 2.11 and Table 2.6 for an illustration.

2.6.7 Behavior on sequences of slowly-varying left-hand side problems

The example used in this section is from a finite element fracture mechanics problem in the field of fatigue and fracture of engineering components (denoted as the *FFEC* collection), which is fully documented in [80, Section 4.1]. Over 2000 linear systems of size 3988×3988 from the *FFEC* collection need to be solved in order to capture the fracture progression, and among them 151 (linear systems 400-550) representing a typical subset of the fracture progression in which many cohesive elements break are examined

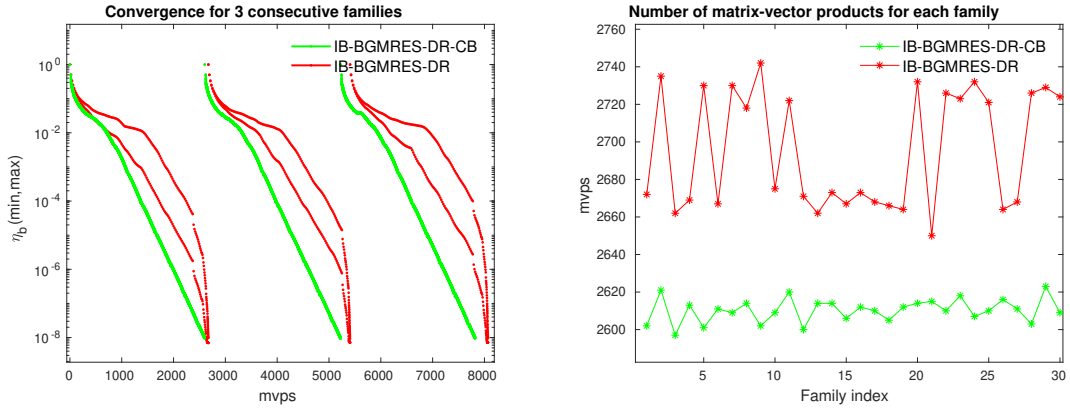


Figure 2.11 – Comparison of IB-BGMRES-DR to IB-BGMRES-DR-CB on families constructed by Matrix 1 with parameters setting as $p^{CB} = 1$, $p = 20$, $m_d = 300$ and $k = 30$. Left: convergence histories of largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for 3 consecutive families. Right: number of $\#mvps$ versus family index.

# families	Method	$\#mvps$	$\#iter$
3	IB-BGMRES-DR	8069	**515**
	IB-BGMRES-DR-CB ($p^{CB} = 15$)	7844	561
	IB-BGMRES-DR-CB ($p^{CB} = 1$)	**7820**	7250
30	IB-BGMRES-DR	80861	**5198**
	IB-BGMRES-DR-CB ($p^{CB} = 1$)	**78308**	72608

Table 2.6 – Numerical results of IB-BGMRES-DR and IB-BGMRES-DR-CB for $p^{CB} = 1, 15$ in terms of $\#mvps$ and $\#iter$ for Section 2.6.6, where the coefficient matrix is Matrix 1 with $p = 20$, $m_d = 300$ and $k = 30$.

in [80]. The solutions of these linear systems have been investigated using both GCRO-DR and GCROT (generalized conjugate residual with inner orthogonalization and outer truncation). We refer the reader to [29] for a comprehensive experimental analysis. For our numerical experiments we borrow the 10 linear systems numbered from 400 to 409 from the *FFEC* collection. For each set of linear systems we select the matrix and the corresponding right-hand sides that we expand to form a block of $p = 20$ by appending random linearly independent vectors.

We display the convergence histories for solving the first 3 consecutive families of such linear systems in the left plot of Figure 2.12. For the solution of the first linear system, the observations on the IB and DR mechanisms discussed in Section 2.6.3 apply. Even though the coefficient matrix has changed, the recycling spectral information computed for the previous family still enables a faster convergence at the beginning of the solution of the next one. Specifically, for the solution of the first family the convergence histories of the two methods fully overlap until the first partial convergence occurs, as until this step the two methods are identical. From the initial slope of the subsequent families, it can be seen that the sequence of matrices are close enough to ensure that the recycled space from one system to the next is still beneficial to the convergence. The benefit of the partial convegence detection is also illustrated on that example since IB-BGCRO-DR still

outperforms BGCRO-DR. The overall benefit in term of *#mvps* savings is illustrated in the right plot on a sequence of 10 linear systems, where the savings are more than 65% with respect to BGCRO-DR. Corresponding results are summarized in Table 2.7.

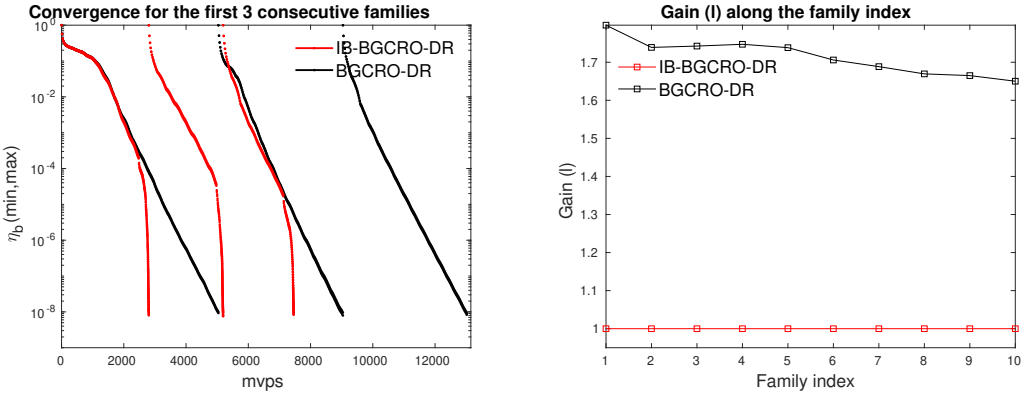


Figure 2.12 – Convergence results of IB-BGCRO-DR and BGCRO-DR on a sequence of slowly changing left-hand sides described in Section 2.6.7, where the coefficient matrices are built on the FFEC with $p = 20$, $m_d = 300$ and $k = 15$.

# families	Method	*#mvps*	*#iter*
3	BGCRO-DR	13050	651
	IB-BGCRO-DR	**7489**	**540**
10	BGCRO-DR	39935	1990
	IB-BGCRO-DR	**24200**	**1658**

Table 2.7 – Numerical results in terms of *#mvps* and *#iter* for Section 2.6.7 with $p = 20$, $m_d = 300$ and $k = 15$.

2.6.8 A variant suited for flexible preconditioning

In this section, we illustrate the numerical behavior of the flexible variant IB-BFGCRO-DR that we have derived in Section 2.2.6 and make comparison with closely related variants namely BFGCRO-DR (a straightforward block extension of FGCRO-DR [19]).

We consider a representative quantum chromodynamics (QCD) matrix from the University of Florida sparse matrix collection [27]. It is the conf5.4-00l8x8-0500 matrix denoted as B_{QCD} of size 49152×49152 with the critical parameter $\kappa_c = 0.17865$ as a model problem. Thirty families of linear systems are constructed that are defined as $A^{(\ell)} = I - \kappa_c(\ell) B_{\text{QCD}}$ with $0 \leq \kappa_c(\ell) < \kappa_c$ and $\ell = 1, 2, \ldots, 30$. We use the MATLAB function **linspace**$(0.1780, 0.1786, 30)$ to generate the parameters $\kappa_c(\ell)$ for a sequence of matrices and observe that those matrices have the same eigenvectors associated with shifted eigenvalues. A sequence of $p = 12$ successive canonical basis vectors are chosen to be the block of right-hand sides for a given left-hand side matrix following [80, Section 4.3] so that the complete set of the right-hand sides for the ℓ linear systems reduces to the first $p \times \ell$ columns of the identity matrix. This choice could be supported by the fact that

the problem of numerical simulations of QCD on a four-dimensional space-time lattice for solving QCD ab initio (cf. [80, Section 4.3]) has a 12×12 block structure, and that a system with 12 right-hand sides related to a single lattice site is often of interest to solve.

The flexible preconditioner is defined by a 32-bit $ILU(0)$ factorization of the matrix involved in the linear system. In a 64-bit calculation framework, the preconditioning consists of casting the set of directions to be preconditioned in 32-bit format, performing the forward/backward substitution in 32-bit calculation and casting back the solutions in 64-bit arithmetic. The rounding applied to the vectors, cast from 64- to 32-bit format, has a nonlinear effect that makes the preconditioner nonlinear. We refer the reader to [5,46] for more examples about the flexible preconditioner developed by mixed precision calculation and also to [91] for that developed by iterative solver itself.

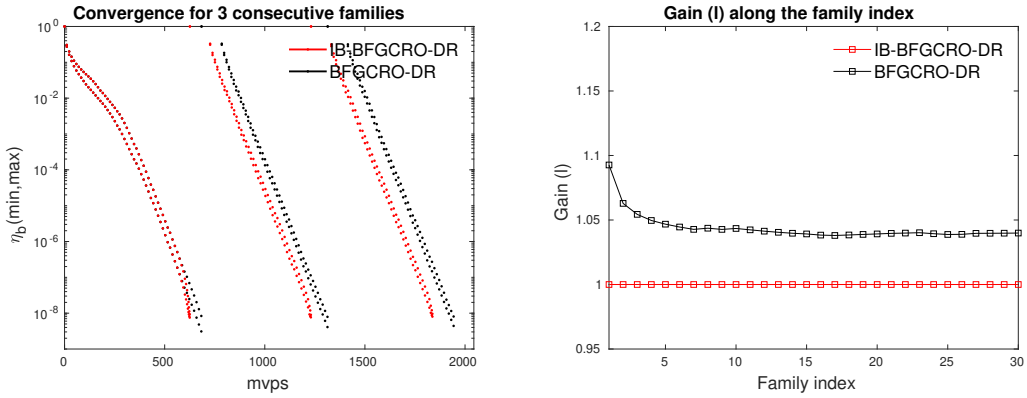


Figure 2.13 – Behavior of the BGCRO-DR-solvers with flexible preconditioner on families of QCD matrices described in Section 2.6.8 with $p = 12$, $m_d = 180$ and $k = 90$. Left: convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for 3 consecutive families. Right: gain (ℓ) of the block methods with respect to IB-BFGCRO-DR along the family index.

# families	Method	$\#mvps$	$\#iter$
3	BFGCRO-DR	1944	**147**
	IB-BFGCRO-DR	**1838**	148
30	BFGCRO-DR	18774	**1347**
	IB-BFGCRO-DR	**18054**	1350

Table 2.8 – Numerical results in terms of $\#mvps$ and $\#iter$ for Section 2.6.8 with $p = 12$, $m_d = 15 \times p = 180$ and $k = 90$.

For those experiments, we attempt to favor the recycling of the space, because the matrices share the same invariant space, so that we choose a relatively large value for k that is $k = m_d/2$. We report in the left plot of Figure 2.13, the convergence histories of the two flexible block variants. Similarly to what has already been observed the convergences are very similar on the first family and only differ when the partial convergence detection becomes active mostly in the last restart. For the second and third families, one can see that IB-BFGCRO-DR and BFGCRO-DR have identical convergence speeds. One can observe a shift in the convergence histories between the end of the solution of one family

and the beginning of the next for both IB-BFGCRO-DR and BFGCRO-DR. This shift is due to the extra k $\#mvps$ that have to be performed when the matrix changes in order to adapt the recycling space as follows

1. Compute $A^{(\ell+1)}U_k^{(\ell)} = \tilde{C}_k$
2. Compute the reduced QR-factorization of $\tilde{C}_k = C_k^{(\ell+1)}R$
3. Update the basis of the deflation space $U_k^{(\ell+1)} = U_k^{(\ell)}R^{-1}$ so that $A^{(\ell+1)}U_k^{(\ell+1)} = C_k^{(\ell+1)}$.

Because k is large, we can clearly see this shift in the left plot of Figure 2.13. For this parameter selection in this section, it can be seen that the dominating effect on the convergence improvement is due to the space recycling and not the partial convergence detection. This observation is highlighted in the right plot of Figure 2.13, where the benefit of using IB-BFGCRO-DR rather than BFGCRO-DR diminishes when compared to previous experiments and is only about 4%. Numerical details are summarized in Table 2.8.

2.7 Concluding remarks

In this chapter, we develop a new variant of the block GCRO-DR method, denoted as IB-BGCRO-DR, that inherits the appealing genes of its two parents [80, 88]. First, it inherits the capability of speeding up the convergence rate when solving sequences of linear systems by recycling spectral information from one family to the next. Second, the extended search space expansion policy enabled by the so-called partial convergence detection allows us to focus on the convergence by considering only the most important directions. Along this line, we introduce stopping-criterion driven search space expansion polices that enable us to ensure that a prescribed threshold used for the partial convergence detection will eventually lead to reaching a prescribed threshold for a backward error based stopping criterion. While introduced in the block GCRO context, those policies apply to any block minimum residual norm approach that relies on an Arnoldi-like relation and includes both block GMRES and GCRO variants. In exact arithmetic, these policies exploit the close link between the least squares residuals and the linear system residuals, which is guaranteed by the orthonormal basis of the residual space. Through numerical experiments, we show that the MGS re-orthogonalization between the columns of recycling space and initial block Arnoldi basis at restart combined with (BMGS $\circ$ HouseQR) in the block Arnoldi algorithm seems to generate a good enough orthonormal basis to ensure that such a property also holds in finite precision calculation. Following ideas from [78], future research could theoretically establish that this class of subspace augmentation algorithms is backward stable. To comply with mixed-precision calculation, the flexible preconditioning variant is also proposed, which would be of interest for emerging computing platforms where mixed-precision calculation could be a way to reduce data movement, which is foreseen as one of the major bottlenecks to reaching high performance.

Chapter 3

The symmetric case with minimum residual norm techniques

3.1 Introduction

In this chapter we present the block counterpart of the conjugate residual (CR) method introduced in [66] for matrices that are symmetric but not necessarily positive definite. The CR method is mathematically equivalent to the minimal residual (MINRES) method [92] and the generalized minimum residual (GMRES) norm method [93] as it minimizes the 2-norm of the residual on the same subspaces. It is based on ideas closely related to the conjugate gradient (CG) method [45] so that their implementations are very similar. In this chapter, we address the question of breakdown as it appears in the block conjugate gradient (BCG) method [52,77], we describe how the partial convergence detection mechanisms of the previous chapter can be extended to the block CR (BCR) context and compare its performance with a more naive but cost-free alternative. For the sake of simplicity and easy cross reference, we adopt the same notations as in Chapter 1.

The remainder of this chapter is organized as follows. We first introduce the BCR algorithm and its properties in Section 3.2. In Section 3.2.1, we respectively review the breakdown [52, 77] problem caused by rank deficiency for the BCG method that does also appear in the BCR context. Then, in Section 3.2.2, we briefly present and comment how the various policies, presented in Section 2.3, to manage the partial convergence can be applied to BCR. In particular, to alleviate their computational cost, we also present a simpler alternative to manage the partial/individual convergence. Two novel BCR variants with breakdown detecting and partial convergence mechanism are proposed. Finally, in Section 3.3, we illustrate various features of the numerical behavior of the proposed methods and we make some concluding remarks in Section 3.4.

3.2 Block conjugate residual method with partial convergence detection

For the solution of linear systems where the matrix is symmetric but not necessarily positive definite, the CR algorithm can be regarded as one general form of the CG methods as discussed by Hestenes [44]. For a single right-hand side solution, the CR method is based on the Lanczos process to generate the orthogonal basis for the Krylov subspace

$\mathcal{K}_n(A, r_0) = \text{span}\{r_0, Ar_0, \cdots, A^{n-1}r_0\}$, in which r_0 denotes the residual associated with the initial guess. In the CR method, the residual vectors are A-conjugate and the vectors Ap_j's are orthogonal to each other [92, Section 6.8] that is the opposite of CG. We refer the reader to [92, Algorithm 6.20] for the implementation of CR and to [98, Algorithm 3.1], [99, Algorithm 2] for its preconditioned variant depicted in Algorithm 4. We notice that in the preconditioned case the preconditioned residuals are A-conjugate and the Ap_j's are M-orthogonal (refer to [60, Algorithm 3] and [82, Section 6] for more discussions) assuming that the preconditioner is symmetric positive definite as for MINRES.

Algorithm 4 *Preconditioned conjugate residual method for $Ax = b$*

Require: $A \in \mathbb{C}^{n\times n}$, the left-hand side of the linear systems, and a preconditioner $M \in \mathbb{C}^{n\times n}$, an approximation of the inverse of A
Require: $b \in \mathbb{C}^n$, the right-hand side, and $x_0 \in \mathbb{C}^n$, the initial guess
Require: m maximum number of the iteration step
1: Compute $r_0 = b - Ax_0$, $z_0 = Mr_0$, $p_0 = z_0$
2: **for** $j = 0, 1, 2, \ldots, m$ **do**
3: $\quad \alpha_j = z_j^H Az_j/(MAp_j)^H Ap_j$
4: $\quad x_{j+1} = x_j + \alpha_j p_j$
5: $\quad r_{j+1} = r_j - \alpha_j(Ap_j)$
6: $\quad z_{j+1} = Mr_{j+1}$
7: $\quad \beta_j = z_{j+1}^H Az_{j+1}/z_j^H Az_j$
8: $\quad p_{j+1} = z_{j+1} + \beta_j p_j$
9: **end for**
10: **return** x_{j+1}, the computed solution

In the context of multiple right-hand sides, the original ideas introduced by O'Leary in [77] for the BCG method immediately apply to define the BCR algorithm for solving $AX = B$. Starting from a block initial guess $X_0 = [x_0^{(1)}, x_0^{(2)}, \ldots, x_0^{(p)}] \in \mathbb{C}^{n\times p}$ and the associated block initial residual $R_0 = B - AX_0$, the core loop of preconditioned BCR algorithm essentially reads

$$\begin{aligned} X_{j+1} &= X_j + P_j\alpha_j, \\ R_{j+1} &= R_j - Q_j\alpha_j, \\ Z_{j+1} &= MR_{j+1}, \\ P_{j+1} &= Z_{j+1} + P_j\beta_j, \\ Q_{j+1} &= AP_{j+1}, \end{aligned}$$

where α_j and β_j are the parameter matrices to be determined such that $Z_{j+1}^H AZ_j = 0$ and $Q_{j+1}^H MQ_j = 0$. Specifically, these parameter matrices are defined by

$$\alpha_j = (Q_j^H MQ_j)^{-1}(Q_j^H Z_j) \in \mathbb{C}^{p_j\times p}, \tag{3.1}$$

and

$$\beta_j = -(Q_j^H MQ_j)^{-1}((MQ_j)^H AZ_{j+1}) \in \mathbb{C}^{p_j\times p}, \tag{3.2}$$

in which some matrix-multiplications related to a $p \times p$ nonsingular matrix may be involved during the practical implementation to improve the stability [52,

77]. This leads to the fact that BCR generates the jth approximate solution X_j such that the Euclidean norm of the corresponding block residual R_j is minimized over the increasing subspaces $X_0 + \mathcal{K}_j(MA, MR_0)$ with $\mathcal{K}_j(MA, MR_0) = \text{span}\{MR_0, (MA)MR_0, \cdots, (MA)^{j-1}MR_0\}$, i.e.,

$$\| R_j \| = \min_{X_j \in X_0 + \mathcal{K}_j(MA, MR_0)} \| B - AX_j \| ,$$

which is the same as the case for the block minimum residual method (Block MINRES) [77, Section 3].

3.2.1 Breakdown in the block conjugate residual method

The BCR method might face similar difficulties as BCG due to the loss of rank in the block of vectors involved in the definition of the parameter matrices that become singular, i.e., the block inverse part shown in Equations (3.1)–(3.2). Hopefully, the breakdown-free remedies proposed for BCG in [52] can easily be extended to the BCR case. The main purpose of the remedies is to ensure that the columns in the block search direction P_j remains full rank so that the matrices α_j and β_j are uniquely defined. Whenever some rank deficiency appears in P_j, P_j is replaced in the algorithm by $\tilde{P}_j$ so that $\tilde{P}_j$ has orthonormal columns and spans the same space as P_j. This is denoted by $\tilde{P}_j = \texttt{orth}(P_j)$ in Algorithm 5 that depicts the BCR with breakdown-free option, where a tilde notation is used to indicate that the dimensions of these matrices may reduce in case of rank deficiency. This `orth` function can be implemented by computing the reduced singular value decomposition (SVD) of P_j and replacing it by the computed right singular vectors associated with nonzero singular values.

3.2.2 Partial convergence detection policies

Because BCR is a minimum residual norm method, the partial convergence policies described in Section 2.3 in the context of block GMRES and GCRO can be considered. However, two main differences exist. First, from a numerical point of view, as BCR is a short term recurrence, the directions that are abandoned at a given iteration cannot be reintroduced later since there is no way to keep them without destroying the short term recurrence feature of the method. Second, from a computational point of view, the algorithm does not compute a QR-factorization of the residual block, contrarily to block GMRES and GCRO where it is a byproduct of the least squares problem solution. Consequently, the partial convergence policies will require the reduced SVD calculation of a tall and skinny matrix, namely of the residual block R_j scaled by a certain diagonal matrix D_ε that depends on the selected stopping criterion and convergence threshold ε, which could be described as the following form:

$$R_j D_\varepsilon = \mathbb{U}_{1,L}\Sigma_1\mathbb{V}_{1,R}^H + \mathbb{U}_{2,L}\Sigma_2\mathbb{V}_{2,R}^H \text{ with } \sigma_{\min}(\Sigma_1) \geq \tau > \sigma_{\max}(\Sigma_2).$$

where $D_\varepsilon = \varepsilon^{-1} \text{ diag } (\|b^{(1)}\|^{-1}, \cdots, \|b^{(p)}\|^{-1}) \in \mathbb{R}^{p\times p}$, with $\tau = 1$ the prescribed IB-threshold. The vectors $(\mathbb{U}_{1,L}, \mathbb{U}_{2,L})$ and $(\mathbb{V}_{1,R}, \mathbb{V}_{2,R})$ are the left and right singular vectors of $R_j D_\varepsilon$, respectively. Once a partial convergence is detected, all the calculations but the update of the solution and residual blocks are performed on blocks of lower column

Algorithm 5 *Block preconditioned Conjugate Residual method with breakdown-free idea — BCR*

Require: $A \in \mathbb{C}^{n\times n}$ the left-hand side of the linear systems and a preconditioner $M \in \mathbb{C}^{n\times n}$ be an approximation of the inverse of A
Require: $B \in \mathbb{C}^{n\times p}$ the block of right-hand sides and $X_0 \in \mathbb{C}^{n\times p}$ the block initial guess
Require: m maximum number of the block iteration step and the maximum number of matrix-vector products is set to be $maxMvps \in \mathbb{N}^+$
Require: $\varepsilon > 0$ a threshold for the selected backward error used in stopping criterion
$R_0 = B - AX_0$ and $Z_0 = MR_0$
$\widetilde{P}_0 = \texttt{orth}(Z_0)$
for $j = 0, 1, 2, \ldots, m$ **do**
  $\widetilde{Q}_j = A\widetilde{P}_j$
  $\widetilde{\alpha}_j = (\widetilde{Q}_j^H M\widetilde{Q}_j)^{-1}(\widetilde{Q}_j^H Z_j)$
  $X_{j+1} = X_j + \widetilde{P}_j\widetilde{\alpha}_j$
  $R_{j+1} = R_j - \widetilde{Q}_j\widetilde{\alpha}_j$
  if the stopping criterion related to ε or $maxMvps$ is met **then**
    return X_{j+1}
  else
    $Z_{j+1} = MR_{j+1}$
    $\widetilde{\beta}_j = -(\widetilde{Q}_j^H M\widetilde{Q}_j)^{-1}((M\widetilde{Q}_j)^H AZ_{j+1})$
    $\widetilde{P}_{j+1} = \texttt{orth}(Z_{j+1} + \widetilde{P}_j\widetilde{\beta}_j)$
  end if
end for
return X_{j+1}

dimension, which lowers the number of matrix-vector and preconditioning applications. Let us denote $[U_j^L, W_j] = SpaceExpansion(R_j, \varepsilon)$, where $U_j^L \in \mathbb{C}^{n\times p_j}$ are the p_j left singular vectors computed by the selected partial convergence detection mechanism and $W_j = (U_j^L)^H R_j \in \mathbb{C}^{p_j\times p}$ the components of the residuals in the space spanned by U_j^L, that are the new directions to be added to the search space. The algorithm is depicted in Algorithm 6, where a bar notation is used to indicate that the dimensions of these matrices may reduce in case of partial convergence detection.

We notice that Algorithm 6 can be accommodated to implement a crude search space expansion that simply discards the columns of the block that correspond to individual solutions that have converged. In that case, U_j^L consists of the columns of the residual that have not yet converged, and $W_j \in \mathbb{R}^{p\times p}$ is a diagonal matrix with entry equal to 0 when the corresponding right-hand side has converged. This somewhat naive alternative algorithm will be referred to as IC-BCR, for Individually Converged BCR.

3.3 Numerical experiments

Numerical experiments are carried out on a set of symmetric positive definite (SPD) and symmetric indefinite matrices from the University of Florida Sparse Matrix Collection [27]. The main features of these SPD and symmetric matrices are respectively described in Table 3.1 and Table 3.2.

Algorithm 6 *Block preconditioned Conjugate Residual method with partial convergence detection (or Inexact-Breakdown) mechanism — IB-BCR*

Require: $A \in \mathbb{C}^{n\times n}$ the left-hand side of the linear systems and a preconditioner $M \in \mathbb{C}^{n\times n}$ be an approximation of the inverse of A
Require: $B \in \mathbb{C}^{n\times p}$ the block of right-hand sides and $X_0 \in \mathbb{C}^{n\times p}$ the block initial guess
Require: m maximum number of the block iteration step and the maximum number of matrix-vector products is set to be $maxMvps \in \mathbb{N}^+$
Require: $\varepsilon > 0$ a threshold for the selected backward error used in stopping criterion
$U_0^L, W_0 = SpaceExpansion(B - AX_0, \varepsilon)$ and $\overline{Z}_0 = MU_0^L$
$\overline{P}_0 = \overline{Z}_0$
for $j = 0, 1, 2, \ldots, m$ **do**
  $\overline{Q}_j = A\overline{P}_j$
  $\overline{\alpha}_j = (\overline{Q}_j^H M\overline{Q}_j)^{-1}(\overline{Q}_j^H \overline{Z}_j)$
  $X_{j+1} = X_j + \overline{P}_j\overline{\alpha}_j W_j$
  $R_{j+1} = R_j - \overline{Q}_j\overline{\alpha}_j W_j$
  if the stopping criterion related to ε or $maxMvps$ is met **then**
    return X_{j+1}
  else
    $[U_{j+1}^L, W_{j+1}] = SpaceExpansion(R_{j+1}, \varepsilon)$
    $\overline{Z}_{j+1} = MU_{j+1}^L$
    $\overline{\beta}_j = -(\overline{Q}_j^H M\overline{Q}_j)^{-1}((M\overline{Q}_j)^H A\overline{Z}_{j+1})$
    $\overline{P}_{j+1} = \overline{Z}_{j+1} + \overline{P}_j\overline{\beta}_j$
  end if
end for
return X_{j+1}

In the default setting of the experiments, the block initial guess is set to be $0 \in \mathbb{C}^{n\times p}$, where p is the number of the right-hand sides. The multiple right-hand sides $B = \texttt{randn}(n, p) = [b^{(1)}, b^{(2)}, \ldots, b^{(p)}] \in \mathbb{C}^{n\times p}$ are composed of p linearly independent vectors containing pseudo-random values drawn from the standard normal distribution (using the same seed when comparing these block methods). The search space expansion policy used in conjunction with the partial convergence detecting is based on the backward error η_b described in Section 2.3.1. Without special notes, the $maxMvps$ is set to be $5000 \times p$ for each solver run, the convergence threshold is $\varepsilon = 10^{-8}$. For all the experiments involving SPD matrices, we consider the preconditioned BCR variants, where an incomplete Cholesky factorization is employed by default as the preconditioner. For the symmetric but not positive definite ones, no preconditioner is considered. The experiments have been carried out in personal Linux (double precision floating point arithmetic) system by MATLAB (R2019a) with hardware setting as PC-Intel (R) Core (TM) i7-8665U CPU ⓐ 1.90 GHz, 8 GB RAM. In order to evaluate the robustness and efficiency of the newly proposed BCR variants, we first investigate in Section 3.3.1 their numerical behavior when the set of right-hand sides is not full rank. Next we investigate their behavior when the convergence threshold ε varies in Section 3.3.2 and when the number p of right-hand sides varies in Section 3.3.3. In Section 3.3.4, we consider examples where all the solutions do not need to be computed using the same convergence threshold. Finally, we report on

symmetric but not positive definite examples in Section 3.3.5.

Name	n	Nonzero	Origin*	Cond. number
apache1	80,800	542,184	Stru. Prob.	
bcsstk15	3,948	117,816	Stru. Prob.	6.53e+09
bcsstk16	4,884	290,378	Stru. Prob.	4.94e+09
bcsstk17	10,974	428,650	Stru. Prob.	1.29e+10
bcsstk18	11,948	149,090	Stru. Prob.	3.45e+11
bundle1	10,581	770,811	Comp. Grap./Vis. Prob.	1.00e+03
cbuckle	13,681	676,515	Stru. Prob.	3.29e+07
crankseg_1	52,804	10,614,210	Stru. Prob.	
crankseg_2	63,838	14,148,858	Stru. Prob.	
gridgena	48,962	512,084	Opti. Prob.	
gyro	17,361	1,021,159	Model Redu. Prob.	1.09e+09
Kuu	7,102	340,200	Stru. Prob.	1.57e+04
s1rmq4m1	5,489	262,411	Stru. Prob.	1.81e+06
s1rmt3m1	5,489	217,651	Stru. Prob.	2.54e+06
s2rmq4m1	5,489	263,351	Stru. Prob.	1.77e+08
s2rmt3m1	5,489	217,681	Stru. Prob.	2.49e+08
s3rmq4m1	5,489	262,943	Stru. Prob.	1.76e+10
s3rmt3m1	5,489	217,669	Stru. Prob.	2.48e+10
shallow_water1	81,920	327,680	CFD Prob.	
shallow_water2	81,920	327,680	CFD Prob.	
ted_B_unscaled	10,605	144,579	Ther. Prob.	1.27e+11

*Structural Problem, Computer Graphics/Vision Problem, Optimization Problem, Model Reduction Problem, Computational Fluid Dynamics Problem and Thermal Problem are simplified as Stru. Prob., Comp. Grap./Vis. Prob., Opti. Prob., Model Redu. Prob., CFD Prob. and Ther. Prob., respectively.

Table 3.1 – Main characteristics of the symmetric positive definite matrices

Name	n	Nonzero	Origin	Cond. number
benzene	8,219	242,669	T/QC Prob.	1.45e+03
rail_5177	5,177	35,185	Model Redu. Prob.	5.33e+04
saylr4	3,564	22,316	CFD Prob.	6.86e+06

Table 3.2 – Main characteristics of the symmetric but not positive definite matrices (The T/QC Prob. abbreviation refers to Theoretical/Quantum Chemistry Problem).

3.3.1 Partial convergence with full rank and rank deficient set of right-hand sides

In order to illustrate the benefit of using partial convergence detection, we consider the case where the right-hand sides are not full rank. For those experiments we select the

matrix Kuu with full rank right-hand sides $B = \texttt{randn}(n, p)$ or rank deficient one $B = [B_{\text{pre}}, B_{\text{pre}}\ \texttt{randn}(p/2, p/2)]$ with $B_{\text{pre}} = \texttt{randn}(n, p/2)$ and $p = 20$. The convergence histories for full rank right-hand sides are displayed in the left graph of Figure 3.1. The shape of the convergence envelop of IB-BCR is very similar to what we have observed in the previous chapters, that is the largest and smallest backward errors smoothly and simultaneously decreases to the target threshold. The more naive, but cost-free, variant IC-BCR exhibits a plateau for the largest backward error when the first right-hand side has converged, then followed by a super fast convergence. Finally, the breakdown-free BCR variant illustrates the drawback of a block solver without any partial or individual convergence detection, that is, many directions are introduced in the search space that enables some backward errors to go below the convergence threshold without special attention to the right-hand sides that converge the slowest. If we look at the block size along the iterations, displayed in the right graph of Figure 3.1, the one of BCR obviously remains equal to p, the partial convergence IB-BCR variant that monitors all the right-hand sides at once starts reducing the block size before any individual solution has converged contrary to the IC variant.

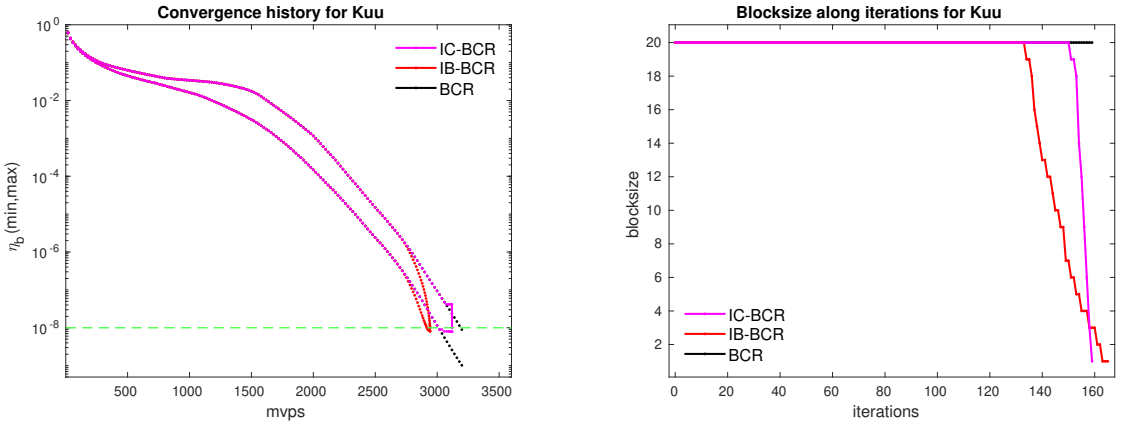


Figure 3.1 – Full rank right-hand sides with $p = 20$. Left: convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ as a function of the number of matrix-vector products ($\#mvps$). Right: block size p_j along the iterations.

We illustrate in Figure 3.2 the robustness introduced by the partial convergence detection mechanism in a fake and somehow extreme case where the rank of the p right-hand sides is $p/2$. As it can be seen in the right graph of Figure 3.2, the rank deficiency is immediately detected by the IB variant that reduces to $p/2$ block size at the very first iteration. Although no real breakdown is encountered by the other two variants, their convergence is very slow due to fact that the block size remains equal to p despite the rank deficiency (except for the very last iterations in the IC variant), which also reveals some lack of robustness. The displayed backward errors (left plot of Figure 3.2) are computed using the norm of the iterative residual, the true residual being computed only when the iterative one meets the convergence criterion. Corresponding numerical performance in terms of the number of matrix-vector products ($\#mvps$) and block iterations ($\#iter$) of these three block variants are summarized in Table 3.3.

In order to be more exhaustive, we report in Table 3.4 the numerical performances of these three variants in terms of $\#mvps$ and $\#iter$ for all matrices listed in Table 3.1.

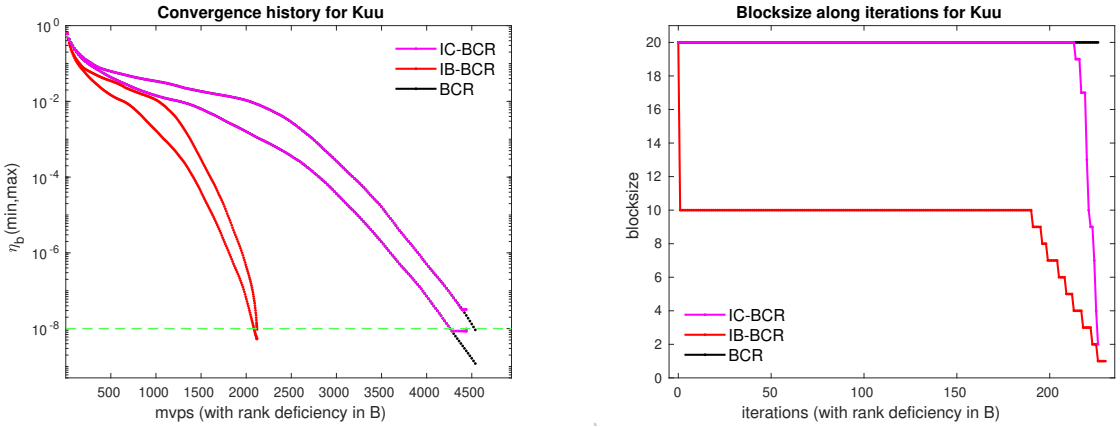


Figure 3.2 – Same case as Figure 3.1 but the right-hand sides are linearly dependent. The $B = [B_{\text{pre}},\ B_{\text{pre}}\ \texttt{randn}(p/2, p/2)]$ with $B_{\text{pre}} = \texttt{randn}(n, p/2)$.

Columns in the RHSs B	Method	$\#mvps$	$\#iter$
linearly independent	BCR	3200	160
	IB-BCR	**2944**	166
	IC-BCR	3121	160
linearly dependent	BCR	4540	227
	IB-BCR	**2121**	231
	IC-BCR	4442	227

Table 3.3 – Numerical results of BCR variants in terms of both $\#mvps$ and $\#iter$ for matrices Kuu with full rank and rank deficient set of right-hand sides.

Because some gaps may exist between the true and iterated residual norm, it is possible that some right-hand sides might not converge with respect to the backward error computed with the true residual norm, while the stopping criterion is met with the iterated residual norm. The "$*$" notation indicates that the convergence based on the iterated residual norm was obtained but not with respect to the true residual one.

3.3.2 Influence of the value of the convergence threshold

In this section, we investigate how the value of convergence threshold affects the performance and robustness of the proposed BCR variants with partial convergence detecting. The convergence thresholds for η_b are set to be $\varepsilon = 10^{-1}, 10^{-2}, 10^{-5}, 10^{-12}$ and the number of right-hand sides is set to $p = 20$. With these numerical setting, for illustration purpose, the convergence histories for the solution involving the matrix Kuu are depicted in Figure 3.3. The general trends are very similar for the various convergence thresholds. We observe that, for IB, all the right-hand sides converge simultaneously to the required accuracy, unlike IC. Although the IB variant is more effective in terms of reducing the number of matrix-vector products ($\#mvps$), both lead to the same solution quality for all the right-hand sides, while BCR compute solutions with much smaller backward error than required.

A more exhaustive set of numerical results are reported in Table 3.5 in terms of

Matrix	*#mvps* BCR / IB-BCR / IC-BCR	*#iter* BCR / IB-BCR / IC-BCR
apache1	15160 / **13938** / 14758	**758** / 870 / 772
bcsstk15	2700 / **2611** / 2668	135 / 155 / 135
bcsstk16	1060* / **989*** / **1042***	53 / 54 / 53
bcsstk17	13080* / **11689*** / 12879*	**654** / 745 / 659
bcsstk18	6440* / **6209*** / 6365*	**322** / 342 / 323
bundle1	700 / **669** / 685	35 / 37 / 35
cbuckle	14460/ **13999** / 14248	**723** / 749 / 727
crankseg_1	6200 / **5535** / 5949	**310** / 333 / 314
crankseg_2	6980 / **6480** / 6821	**349** / 370 / 351
gridgena	9140 / **8798** / 9067	457 / 463 / 457
gyro	31960* / **21221*** / 32498*	**1598** / 8075 / 9619
s1rmq4m1	2760 / **2445** / 2689	**138** / 157 / 139
s1rmt3m1	2800 / **2512** / 2739	**140** / 157 / 141
s1rmq4m1	2760 / **2445** / 2689	**138** / 157 / 139
s1rmt3m1	2800 / **2512** / 2739	**140** / 157 / 141
s2rmq4m1	4440* / **3935*** / 4354*	**222** / 244 / 223
s2rmt3m1	5660* / **5119*** / 5577*	**283** / 326 / 284
s3rmq4m1	10380* / **9199*** / 10207*	**519** / 711 / 533
s3rmt3m1	11620* / **10590*** / 11469*	**581** / 799 / 594
shallow_water1	300 / 300 / 300	15 / 15 / 16
shallow_water2	560 / 560 / 560	28 / 28 / 29
ted_B_unscaled	540* / 540* / 540*	27 / 27 / 28

Table 3.4 – Numerical results of BCR variants in terms of both *#mvps* and *#iter* for all tested matrices listed in Table 3.1 with the right-hand sides $B = \texttt{randn}(n, p)$, $p = 20$ and $\varepsilon = 10^{-8}$. The "$*$" indicates that the convergence based on the iterated residual norm was obtained but not with respect to the true residual one.

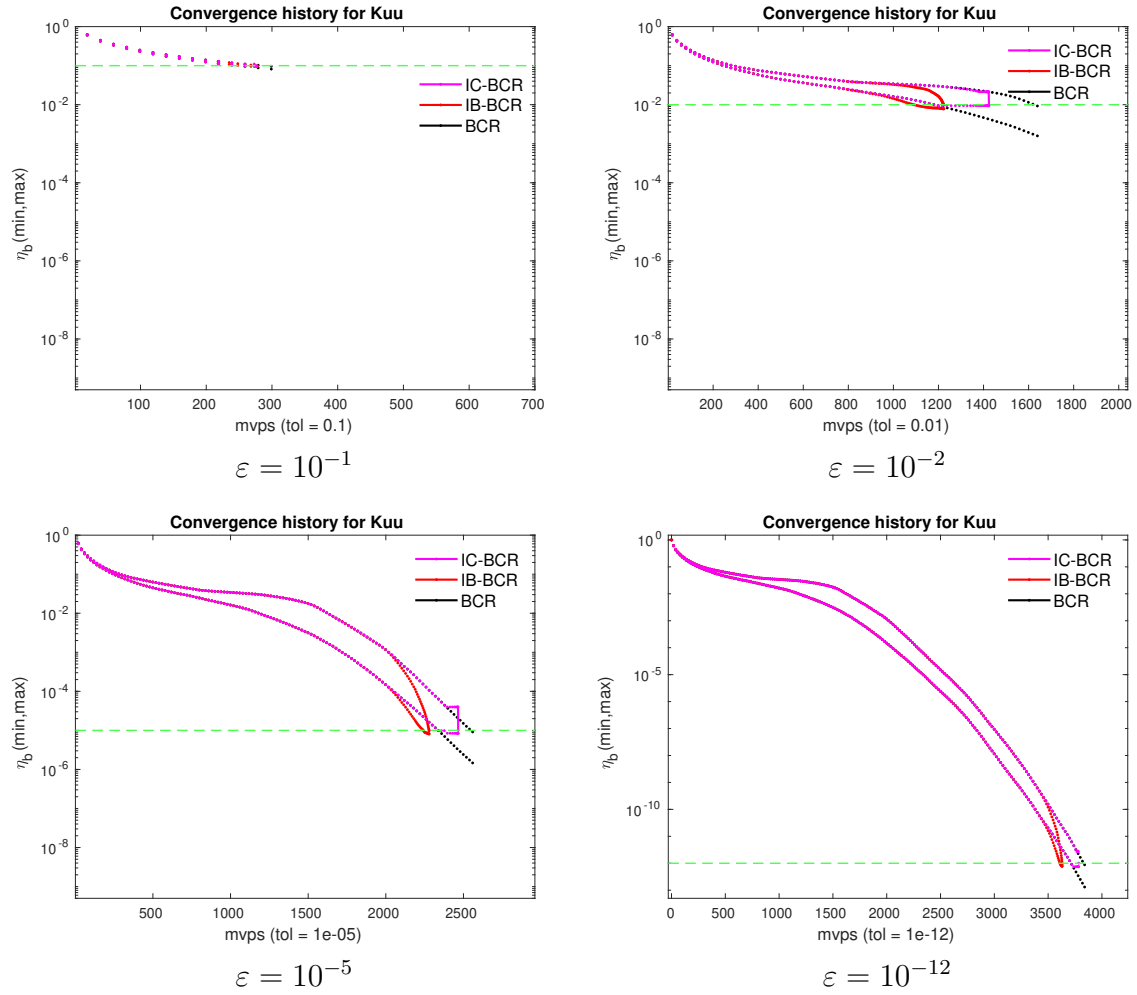


Figure 3.3 – Convergence history for solving linear systems built by Kuu ($B = \texttt{randn}(n, p)$, $p = 20$ and $maxMvps = 5000 \times p$) with different values of the convergence threshold.

$\#mvps$ and $\#iter$ for other examples. As previously indicated the "$*$" indicates that the convergence based on the iterated residual norm was obtained but not with respect to the true residual norm. This illustrates that the short term recurrence induces some residual gaps that cause trouble for stringent convergence thresholds like 10^{-12}. The general trend is that the IB and IC variants minimize $\#mvps$ while the classical BCR one minimizes $\#iter$.

3.3.3 Influence of the number of right-hand sides

In this section, we illustrate how the number of the right-hand sides interplays with the performance of the BCR variants; we vary $p = 5, 10, 30, 40$. The numerical experiments are displayed in Table 3.6, where as previously the "$*$" indicates that the convergence based on the iterated residual norm was obtained but not with respect to the true residual norm. No significant impact on the ranking of the variants can be observed. When solving for a large number of right-hand sides, it can be seen that it is preferable (when it is affordable from a memory view point) to solve all at once rather than dividing them in chunks of smaller number to be solved in sequence. For instance solving for $p = 40$ right-hand

Matrix	ε	10^{-1}		10^{-2}		10^{-5}		10^{-12}	
	Method	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$	$\#mvps$	$\#iter$
apache1	BCR	740	37	7740	**387**	12080	**604**	18960*	**948**
	IB-BCR	**725**	38	**5619**	887	**10544**	740	**17630***	1021
	IC-BCR	726	37	6082	670	11530	624	18575*	959
bcsstk18	BCR	3160	158	4500	**225**	5780	**289**	7140*	357
	IB-BCR	**2025**	212	**3986**	317	**5562**	343	**6903***	367
	IC-BCR	2501	**156**	4183	235	5672	290	7078*	357
cbuckle	BCR	3580	**179**	6960	**348**	11640	**582**	17760*	**888**
	IB-BCR	**3167**	205	**6262**	384	**11036**	606	**17231***	905
	IC-BCR	3362	180	6655	349	11429	585	17608*	890
crankseg_2	BCR	2180	**109**	2880	**144**	4960	**248**	9480*	**474**
	IB-BCR	**1063**	520	**2197**	204	**4468**	270	**9004***	496
	IC-BCR	1143	131	2642	148	4816	251	9344*	476
gridgena	BCR	220	11	5880	**294**	8500	425	9940*	497
	IB-BCR	**193**	11	5821	366	8846	803	**9595***	502
	IC-BCR	195	11	**5582**	303	**8411**	425	9849*	497
Kuu	BCR	300	15	1640	**82**	2560	128	3840	192
	IB-BCR	279	17	**1225**	107	**2280**	136	**3631**	197
	IC-BCR	**278**	15	1425	85	2467	128	3785	192

Table 3.5 – Numerical results of BCR variants in terms of $\#mvps$ and $\#iter$ for parts of matrices listed in Table 3.1 with the right-hand sides $B = \texttt{randn}(n, p)$, $p = 20$, $maxMvps = 5000 \times p$, and various convergence thresholds $\varepsilon = 10^{-1}, 10^{-2}, 10^{-5}, 10^{-12}$.

sides with the matrix apache1 by IB-BCR does not require 4 times more matrix-vector products (*#mvps*) but 2.8 compared with solving a sequence of 4 block systems with $p = 10$.

Matrix	p	5		10		30		40	
	Method	*#mvps*	*#iter*	*#mvps*	*#iter*	*#mvps*	*#iter*	*#mvps*	*#iter*
apache1	BCR	7590	**1518**	10800	**1080**	19080	**636**	22200	**555**
	IB-BCR	**7082**	1641	**9920**	1191	**17022**	752	**19784**	717
	IC-BCR	7532	1520	10538	1091	18435	653	21416	569
bcsstk18	BCR	4200*	**840**	5370*	**537**	7140*	**238**	7760*	**194**
	IB-BCR	**3963***	871	**5213***	569	**6813***	251	**7344***	206
	IC-BCR	4070*	846	5297*	540	7053*	240	7647*	195
cbuckle	BCR	6055	**1211**	10010	**1001**	17640	**588**	20080	**502**
	IB-BCR	**5892**	1236	**9573**	1019	**16924**	618	**19255**	531
	IC-BCR	6000	1212	9808	1003	17491	590	19857	506
crankseg_2	BCR	2765	**553**	4160	**416**	9390	**313**	11760	**294**
	IB-BCR	**2581**	579	**3961**	431	**8737**	337	**10856**	323
	IC-BCR	2674	559	4060	417	9195	316	11538	297
gridgena	BCR	7130	**1426**	8150	815	10080	336	10880	272
	IB-BCR	7213	1548	**8023**	879	**9525**	342	**10203**	279
	IC-BCR	**7104**	1427	8118	815	9967	336	10752	272
Kuu	BCR	1600	320	2260	226	3900	**130**	4440	111
	IB-BCR	**1476**	329	**2097**	235	**3572**	136	**4075**	118
	IC-BCR	1582	320	2224	226	3822	131	4323	111
s2rmt3m1	BCR	3910*	**782**	4690*	**469**	6240*	**208**	6800*	**170**
	IB-BCR	**3785***	868	**4435***	554	**5620***	248	**5987***	209
	IC-BCR	3863*	784	4634*	471	6137*	209	6688*	172
ted_B_unscaled	BCR	135*	27	270*	27	870*	29	1120*	**28**
	IB-BCR	135*	27	270*	27	**851***	29	1144*	29
	IC-BCR	135*	28	270*	28	840*	29	1120*	29

Table 3.6 – Numerical results of BCR variants with different number of right-hand sides ($p = 5, 10, 30, 40$) in terms of *#mvps* and *#iter* for parts of matrices listed in Table 3.1 with $B = \texttt{randn}(n, p)$, $maxMvps = 5000 \times p$ and $\varepsilon = 10^{-8}$.

3.3.4 Experiments with individual convergence threshold

As indicated in Corollary 2 in Section 2.3.1 the partial convergence mechanism can be adapted to cope with different individual convergence thresholds refer to as "variable accuracy" in the VA variant. In the BCR context, it implies to change the *SpaceExpansion* function described in Section 3.2.2. We illustrate this feature in Figure 3.4 where we consider the solution for $p = 20$ and set the convergence threshold to be $\varepsilon = 10^{-4}$ for the first $p/2$ right-hand sides and $\varepsilon = 10^{-8}$ for the last $p/2$ ones. It can be seen that it numerically works, but the computational benefit is not significant for that example. This is confirmed by other numerical experiments that are reported in Table 3.7, which exhibits a moderate positive benefit of this VA variant that was very effective in the block GCRO context presented in Section 2.6.5.

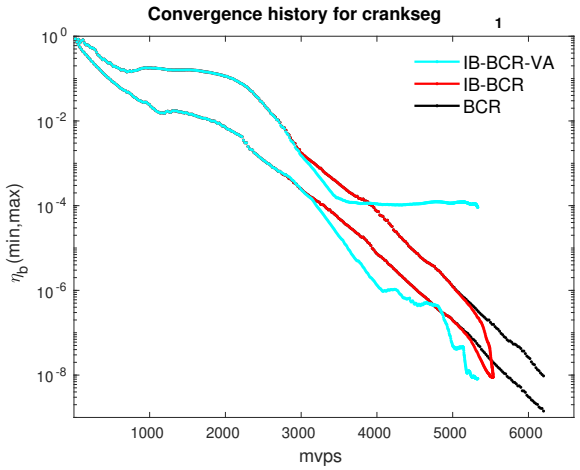


Figure 3.4 – Histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for matrix crankseg_1 with convergence threshold equals to 10^{-4} for the first $p/2$ right-hand sides and 10^{-8} for the last $p/2$ ones ($p = 20$).

Matrix	$\#mvps$ BCR / IB-BCR / IB-BCR-VA	$\#iter$ BCR / IB-BCR / IB-BCR-VA
cbuckle	14460 / **14018** / 14049	**723** / 749 / 1069
crankseg_1	6200 / 5554 / **5344**	**310** / 333 / 482
crankseg_2	6980 / **6499** / 6943	**349** / 369 / 845
Kuu	3200 / **2963** / 3609	**160** /166 / 423
shallow_water2	560 / 560 / **430**	28 / 28 / 28
ted_B_unscaled	540* / 540* / **415***	27 / 27 / 27

Table 3.7 – Numerical results of Section 3.3.4 in terms of $\#mvps$ and $\#iter$ with $B =$ `randn`(n, p), $p = 20$ and $maxMvps = 5000 \times p$.

3.3.5 Experiments with symmetric matrices

In this section, we consider testing three symmetric but not positive definite matrices, described in Table 3.2, with linearly independent right-hand sides defined as $B =$ `randn`(n, p) with $p = 20$, $m_d = 5000 \times p$, $\varepsilon = 10^{-8}$ and no preconditioner is applied. The corresponding convergence history and numerical results are respectively reported in Figure 3.5 and Table 3.8. The observations are very similar to what we have seen for symmetric positive definite matrices in the previous sections, that is IB-BCR is often the best in minimizing the number of matrix-vector products ($\#mvps$) at a possible extra cost of a few more block iterations ($\#iter$).

3.4 Concluding remarks

In this chapter, we propose new variants of the block conjugate residual method with breakdown-free and then with partial convergence detecting mechanism, which are respectively denoted as BCR, and IB-BCR as well as a cost-free alternative IC-BCR. Specifically, we extend the partial convergence detection idea to the BCR method by carrying out an extra reduced singular value decomposition (SVD) on a scaled block

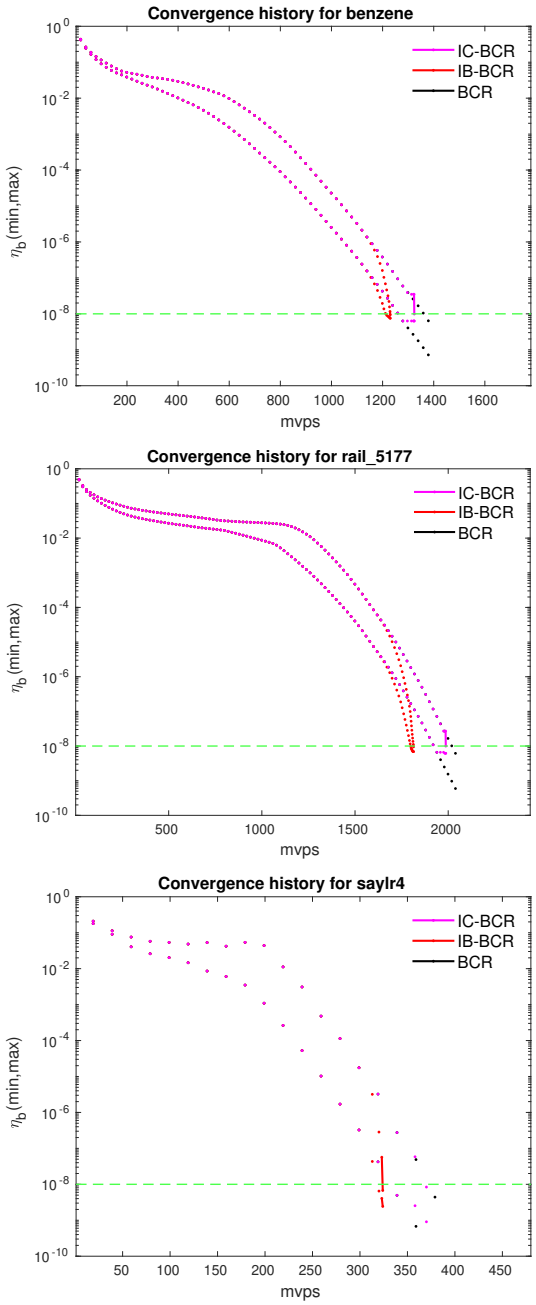


Figure 3.5 – Comparison convergence history of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ of the BCR variants by solving symmetry but not positive definite matrices listed in Table 3.2 with the right-hand sides $B = \texttt{randn}(n, p)$, $p = 20$, $maxMvps = 5000 \times p$ and $\varepsilon = 10^{-8}$.

residual (a tall and skinny matrix) for each block iteration to determine the directions to be kept and the ones to be abandoned. We point out that the breakdown free variant also requires similar SVD calculation to implement the MATLAB `orth()` function. The general observed trend is that IB-BCR performs best in terms of number of matrix-vector products, while BCR is best in terms of block iterations. Because all the variants require the reduced SVD of a tall and skinny matrix to detect the rank deficiency of the search space for the IC-BCR and BCR variants, or to detect the partial convergence through the

Matrix	Method	*#mvps*	*#iter*
benzene	BCR	1380	69
	IB-BCR	**1231**	70
	IC-BCR	1325	69
rail_5177	BCR	2040	102
	IB-BCR	**1815**	107
	IC-BCR	1988	102
saylr4	BCR	380	19
	IB-BCR	**325**	19
	IC-BCR	371	19

Table 3.8 – Numerical results of the BCR variants in terms of *#mvps* and *#iter* for matrices listed in Table 3.2 with $B = \texttt{randn}(n, p)$, $p = 20$ and $maxMvps = 5000 \times p$.

scaled block residual for the IB-BCR, the best variant in terms of time to the solution will depend on the relative cost of this reduced SVD with respect to the calculation of the preconditioning and matrix applications for a given number of right-hand sides.

Note that for the block minimum residual norm subspace solvers based on the Arnoldi basis presented in Chapter 2, a good enough orthonormal basis could be generated and thus ensuring that the true residual norm and the least squares one are close enough to monitor the convergence of the former through the latter. For short term recurrence algorithm such as BCR, some residual gaps exist, sometimes preventing the convergence of the true backward error, i.e., based on the true residual norm, even though the one based on the iterated residual norm indicates convergence. A possible remedy could be to extend the rounding error analysis from [120] to block case to estimate the block residual gap and possibly design corresponding replacement techniques [16, 25].

Chapter 4

The Hermitian positive definite case with conjugate gradient variants

4.1 Introduction

In this chapter, we consider developing new variants of the block conjugate gradient (BCG) [77] method for the solution of sequences of linear systems with multiple right-hand sides for Hermitian positive definite coefficient matrices. In that framework, we mainly study two complementary techniques to take advantage of the opportunities offered by this situation. Firstly, we investigate the possibility to use the partial convergence detection presented in the previous chapter for the BCR method as a heuristic to reduce the block size when the convergence, evaluated by a backward error criterion, takes place. In the context of BCG, this numerical mechanism is only heuristic because the associated search space expansion in BCR relies on residual norm minimization, while for BCG it would be alternatively based on A-norm error minimization, which is the quantity minimized by this numerical method. Secondly, we consider subspace recycling strategy between the successive linear systems with multiple right-hand sides to accelerate convergence rate of BCG by approximating and reusing some spectral information. More precisely, we introduce and study a Deflated Block Conjugate Gradient (D-BCG) method that periodically refines the recycling subspace along the block iterations. One of our variants reduces to the natural block counterpart of the deflated conjugate gradient method introduced in [94] if the recycling space only updates once for each linear system.

The structure of this chapter is organized as follows. In Section 4.2 we describe the design of the partial convergence idea in the framework of BCG after a short overview of its breakdown-free variant [52]. Next in Section 4.3, we present various options to extract spectral information from the D-BCG iterations to be used for the subsequent linear systems. In Section 4.4 we report on intensive numerical experiments to illustrate the numerical features of the studied algorithms. Finally, we conclude with some detailed remarks of these proposed BCG variants in Section 4.5.

4.2 Block conjugate gradient method with partial convergence detection

In this section, we first review the breakdown [52,77] issue that existed in the BCG [77] method. Then, by exploiting the algorithmic resemblance between BCG and BCR, we shortly describe the partial convergence detection mechanisms that can be designed for BCG.

4.2.1 Breakdown in the block conjugate gradient method

The block conjugate gradient (BCG) method was proposed by O'Leary in [77] for solving linear systems with a Hermitian positive definite (HPD) linear matrix with multiple right-hand sides given simultaneously. Its preconditioned version is depicted in Algorithm 7.

Algorithm 7 Block preconditioned conjugate gradient method for $AX = B$

Require: $A \in \mathbb{C}^{n\times n}$ the left-hand side of the linear systems and a preconditioner $M \in \mathbb{C}^{n\times n}$ be an approximation of the inverse of A
Require: $B \in \mathbb{C}^{n\times p}$ the of right-hand-sides and $X_0 \in \mathbb{C}^{n\times p}$ the block initial guess
Require: m maximum number of the block iteration step
1: Compute $R_0 = B - AX_0$, $Z_0 = MR_0$, $P_0 = Z_0$
2: **for** $j = 0, 1, 2, \ldots, m$ **do**
3: $\quad \alpha_j = (P_j^H AP_j)^{-1}(Z_j^H R_j)$
4: $\quad X_{j+1} = X_j + P_j\alpha_j$
5: $\quad R_{j+1} = R_j - AP_j\alpha_j$
6: $\quad Z_{j+1} = MR_{j+1}$
7: $\quad \beta_j = (Z_j^H R_j)^{-1}(Z_{j+1}^H R_{j+1})$
8: $\quad P_{j+1} = Z_{j+1} + P_j\beta_j$
9: **end for**
10: **return** X_{j+1} computed solution

As mentioned in [77], in the implementation of BCG, the block parameter matrices α_j and β_j involved in the jth iteration of BCG are respectively formulated as

$$\alpha_j = (P_j^H AP_j)^{-1}(R_j^H Z_j) \in \mathbb{C}^{p\times p},$$

and

$$\beta_j = (R_j^H Z_j)^{-1}(R_{j+1}^H Z_{j+1}) \in \mathbb{C}^{p\times p}, \text{ with } Z_{j+1} = MR_{j+1},$$

in which the inverse part of these two parameter matrices may become singular, which results in the so-called breakdown [52, 77] problem. That is, the algorithm terminates early without finding a satisfactory approximate solution. Based on the potential rank deficiency in the block search direction, Ji and Li proposed the breakdown-free block conjugate gradient method [52], where the block search directions P_j are replaced by $\widetilde{P}_j$, an orthonormal basis of the space spanned by P_j, which can be computed by considering the left singular vectors of P_j associated with non-zero singular values. In addition, the new variant proposed in [52] chooses two parameter matrices with an alternatively

formulation as

$$\alpha_j = (\widetilde{P}_j^H A \widetilde{P}_j)^{-1} (\widetilde{P}_j^H R_j) \in \mathbb{C}^{p_j \times p},$$

and

$$\beta_j = -(\widetilde{P}_j^H A \widetilde{P}_j)^{-1} (\widetilde{P}_j^H A Z_{j+1}) \in \mathbb{C}^{p_j \times p},$$

in which the two parameter matrices are always well defined because the possible singularity due to the rank deficiency of P_j (replaced by $\widetilde{P}_j$) is discarded. The resulting algorithm is detailed in Algorithm 8, where a tilde notation is used to indicate that the dimensions of these matrices may reduce in case of rank deficiency.

Algorithm 8 Block preconditioned Conjugate Gradient method with breakdown-free idea — BCG

Require: $A \in \mathbb{C}^{n \times n}$ the left-hand side of the linear systems and a preconditioner $M \in \mathbb{C}^{n \times n}$ that is an approximation of the inverse of A
Require: $B \in \mathbb{C}^{n \times p}$ the of right-hand-sides and $X_0 \in \mathbb{C}^{n \times p}$ the block initial guess
Require: m maximum number of block iteration steps
Compute $R_0 = B - AX_0$, $Z_0 = MR_0$, $P_0 = \texttt{orth}(Z_0)$
for $j = 0, 1, 2, \ldots, m$ **do**
  $\widetilde{Q}_j = A\widetilde{P}_j$
  $\widetilde{\alpha}_j = (\widetilde{P}_j^H \widetilde{Q}_j)^{-1} (\widetilde{P}_j^H R_j)$
  $X_{j+1} = X_j + \widetilde{P}_j \widetilde{\alpha}_j$
  $R_{j+1} = R_j - \widetilde{Q}_j \alpha_j$
  $Z_{j+1} = MR_{j+1}$
  $\widetilde{\beta}_j = -(\widetilde{P}_j^H \widetilde{Q}_j)^{-1} (\widetilde{Q}_j^H Z_{j+1})$
  $\widetilde{P}_{j+1} = \texttt{orth}(Z_{j+1} + \widetilde{P}_j \widetilde{\beta}_j)$
end for
return X_{j+1} computed solution

4.2.2 Partial convergence detection policies

Although the conjugate gradient (CG) [44] and BCG [77] methods minimize the A-norm of the forward error, the most commonly used stopping criterion relies on a backward error that is mostly based on the residual norm. This observation motivates us to adapt the partial convergence mechanism presented in the previous chapter in the context of the BCR method. Because the two algorithms are built on the same computational kernels, the development of corresponding IB-BCG counterpart is fairly straightforward and depicted in Algorithm 9. We notice that this algorithm might lack robustness for two main reasons. As in the BCR case, the abandoned directions cannot be introduced in later iterations, contrary to the minimum residual norm algorithms presented in Chapter 2, because it would destroy the short term recurrence nature of the conjugate gradient algorithms. Secondly, BCG minimizes the A-norm of the errors so that controlling the search space expansion through heuristic on the residuals does not perfectly comply with this nice numerical feature. Finally, the corresponding IC (Individually Converged) variant can also be considered; of does not suffer from the two previously possible flaws of IB-BCG as it mostly consists in stopping iterating on the iterates that have converged.

Algorithm 9 Block preconditioned Conjugate Gradient method with partial convergence (or Inexact-Breakdown detection) mechanism — IB-BCG

Require: $A \in \mathbb{C}^{n\times n}$ the left-hand side of the linear systems and a preconditioner $M \in \mathbb{C}^{n\times n}$ that is an approximation of the inverse of A
Require: $B \in \mathbb{C}^{n\times p}$ the block of right-hand-sides and $X_0 \in \mathbb{C}^{n\times p}$ the block initial guess
Require: m maximum number of the block iteration step and the maximum number of matrix-vector products ($\#mvps$) is set to be $maxMvps \in N^+$
Require: $\varepsilon > 0$ a threshold for the selected backward error used in stopping criterion
$U_0^L, W_0 = SpaceExpansion(B - AX_0, \varepsilon)$ and $\overline{Z}_0 = MU_0^L$
$\overline{P}_0 = (\overline{Z}_0)$
for $j = 0, 1, 2, \ldots, m$ **do**
  $\overline{Q}_j = A\overline{P}_j$
  $\overline{\alpha}_j = (\overline{P}_j^H \overline{Q}_j)^{-1}(\overline{P}_j^H \overline{R}_j)$
  $X_{j+1} = X_j + \overline{P}_j \overline{\alpha}_j W_j$
  $R_{j+1} = R_j - \overline{Q}_j \overline{\alpha}_j W_j$
  if the stopping criterion related to ε or $maxMvps$ is met **then**
    return X_{j+1}
  else
    $[U_{j+1}^L, W_{j+1}] = SpaceExpansion(R_{j+1}, \varepsilon)$
    $\overline{Z}_{j+1} = MU_{j+1}^L$
    $\overline{\beta}_j = -(\overline{P}_j^H \overline{Q}_j)^{-1}(\overline{Q}_j^H \overline{Z}_{j+1})$
    $\overline{P}_{j+1} = \texttt{orth}(\overline{Z}_{j+1} + \overline{P}_j \overline{\beta}_j)$
  end if
end for
return X_{j+1} for approximation of the linear systems

We illustrate the numerical performance of the BCG variants with partial convergence detection and individual convergence detection in Section 4.4 and will describe in the next sections complementary methodological tools that might be adopted to speed-up the convergence of the subsequent linear systems when a sequences of them have to be solved.

4.3 Deflated block conjugate gradient variants

One of the guiding ideas of the GCRO [28] method, the block variants were presented in Chapter 2, is to first compute the approximated solution in a low-dimensional subspace and then to force the remaining iterations to work in the complementary orthogonal space where the solution is more amenable for iterative scheme. Similar ideas have been developed in the HPD case that led to various variants of the CG algorithm [32,36,94,108]. In this chapter, we will focus on the deflated CG variant [94] with an alternative projection way introduced in [126] for realizing deflation. We will present its block counterpart with a periodically refining deflation subspace, denoted as D-BCG, suited for the solution of multiple right-hand sides. We will also consider various alternatives for extracting spectral information along the D-BCG iterations that is used as the deflation space for the solution of the next right-hand side block.

4.3.1 The deflated block conjugate gradient algorithm

The implementation of projected variant of the deflated block conjugate gradient algorithm [126] (denoted as D-BCG in this manuscript) is given in Algorithm 10. Notice that the A-orthogonal projector is applied to the block search direction P_j rather than the block residual R_j which is generally used in the standard deflated BCG algorithm [20] to mitigate the gradually vanishing orthogonality between the block residual vectors and the deflation space. In Algorithm 10, $W_k \in \mathbb{C}^{n\times k}(k < n)$ is a nonsingular tall and skinny matrix, whose columns span the deflation space. The parameter matrices, i.e., α_j and β_j, used in [52] are adopted to generate an orthonormal search basis per iteration to handle the possible breakdown issue caused by rank deficiency. The number of columns of P_j is denoted as p_j which may become smaller than the original block size p, i.e., $(p_j \leq p)$, when the columns of P_j become rank deficient.

Algorithm 10 Projected variant of the Deflated Block Conjugate Gradient algorithm — D-BCG

Require: $A \in \mathbb{C}^{n\times n}$ the left-hand side of the linear systems and a preconditioner $M \in \mathbb{C}^{n\times n}$
Require: $B \in \mathbb{C}^{n\times p}$ the block of right-hand-sides and $X_{-1} \in \mathbb{C}^{n\times p}$ the block initial guess
Require: $\varepsilon > 0$ a targeted backward error used in stopping criteria
Require: $W_k = [w_1, \cdots, w_k]$ a possibly initial tall and skinny matrix with k linearly independent columns
Require: m the maximum number of block iteration step, and the maximum number of matrix-vector products is set to be $maxMvps \in N^+$
Compute $R_{-1} = B - AX_{-1}$
$X_0 = X_{-1} + W_k(W_k^H AW_k)^{-1}W_k^H R_{-1}$
$R_0 = R_{-1} - AW_k(W_k^H AW_k)^{-1}W_k^H R_{-1}$
$Z_0 = MR_0$
$\hat{P}_0 = \texttt{orth}(Z_0) \in \mathbb{C}^{p\times p_0}$
for $j = 0, 1, 2, \ldots, m$ **do**
  $P_j = \left(I - W_k(W_k^H AW_k)^{-1}(AW_k)^H\right)\hat{P}_j$
  $Q_j = AP_j$
  $\alpha_j = (P_j^H Q_j)^{-1}(P_j^T R_j)$
  $X_{j+1} = X_j + P_j\alpha_j$
  $R_{j+1} = R_j - Q_j\alpha_j$
  if the stopping criterion related to ε or $maxMvps$ is met **then**
    return X_{j+1}
  else
    $Z_{j+1} = MR_{j+1}$
    $\beta_j = -(P_j^H Q_j)^{-1}(Q_j^T Z_{j+1})$
    $\hat{P}_{j+1} = \texttt{orth}(Z_{j+1} + P_j\beta_j) \in \mathbb{C}^{n\times p_{j+1}}$
  end if
end for
return X_{j+1} for approximation of the linear systems

In order to reduce the condition number of the projected linear system eventually solved by D-BCG, one often tries to use good approximations of a few eigenvectors

associated with external eigenvalues in order to define the deflation basis W_k. In a practical implementation, this spectral information is computed by exploiting the relationship between the conjugate gradient and the Lanczos method for the calculation of eigenpairs. We present a few algorithms that compute eigenvalues and eigenvectors of an HPD matrix while solving a linear system of equations with the BCG variants. There are many values, all the CG iteration vectors could be saved and recombined using the eigenvectors of the tridiagonal projection matrix; this is theoretically equivalent to unrestarted Lanczos. Our algorithms attempt to capitalize on the iteration vectors produced by BCG to update only a small window of vectors that approximates the eigenvectors. While this window is restarted using various ways, the BCG algorithm for the linear system is unaffected. The base line approach is the so called thick-restart approach introduced in [125].

4.3.2 Eigenvector computation from D-BCG iterations

4.3.2.1 The Lanczos thick-restart strategy for eigencalculation

In a classical Lanczos context for the solution of eigenproblems, various techniques have been introduced to limit the memory footprint of the Krylov subspace basis while maintaining good convergence. Among the possible policies, we consider in this work the so called thick-restart method introduced in [125]. The governing idea of the thick-restart method is to expand the search eigenspace up to a maximum dimension and then to somehow compress the interesting information into a fixed size k-dimensional subspace before restarting and expand the search space again. Let us briefly describe the governing ideas. Let

$$AV_m = V_m T_m + \beta_m v_{m+1} e_m^T$$

denote the Lanczos equality when a first search space of size m has been built. At this stage m Ritz pairs (λ_j, w_j) can be computed such that $w_j = V_m \hat{w}_j$, where $T_m \hat{w}_j = \lambda_j \hat{w}_j$.

Their residuals are:

$$Aw_i = \lambda_i w_i + \beta_m e_m^T \hat{w}_i v_{m+1}.$$

Denote $(w_1, \cdots w_k)$, the target set of vectors to be improved/refined, the residual equations can be written in matrix form as follows

$$A[w_1, \cdots w_k] = [w_1, \cdots w_k, v_{m+1}] \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \\ \beta_m e_m^T \hat{w}_1 & \cdots & \beta_m e_m^T \hat{w}_k \end{pmatrix}.$$

The Lanczos iterations can be continued from this point, denoting $v_1^{new} = v_{m+1}$, until a new search space of dimension $m + k$ is obtained and the new Lanczos equality becomes:

$$A[W_k, V_m^{new}] = [W_k, V_{m+1}^{new}] \begin{pmatrix} T_{11} & T_{12} \\ T_{12}^H & T_{22} \\ \beta_{m+k} e_{m+k}^T \end{pmatrix} = [W_k, V_{m+1}^{new}] G, \tag{4.1}$$

where $T_{11} = diag(\lambda_1, \cdots, \lambda_k) \in \mathbb{R}^{k \times k}$, T_{22} is a regular tridiagonal Lanczos matrix, and T_{12}^H is zero everywhere except its first row that is given by $(\beta_m e_m^T \hat{w}_1 \cdots \beta e_m^T \hat{w}_k)$. Notice that the V_m^{new} vectors correspond to the directions that would have been generated by

the "full" Lanczos method so that $[W_k, V_m]$ span a subspace of the Krylov subspace that would have been spanned by the unrestarted Lanczos method. A Raleigh-Ritz procedure with respect to the space spanned by $[W_k, V_m^{new}]$ to compute new k eigenpairs that will serve to update W_k and the Lanczos algorithm is continued with restart every other m steps until convergence.

Exploiting the relationship between the Lanczos and the CG method, it appears that all the quantities involved in the thick-restart procedure are a byproduct of the CG iterations. Consequently, spectral information can be extracted using this thick-restart technique if it is embedded in the CG iterations. Notice that the CG convergence is not affected, only extra storage and computation are allocated for the embedded spectral computation. In the short presentation above, at each cycle of the thick-restart, the spectral information is computed using a Raleigh-Ritz procedure. This elegantly induces a nice structure in the G matrix in Equation (4.1), that is essentially tridiagonal with a spike on its $(k+1)^{th}$ row and column. Furthermore all the entries of G are essentially byproducts of the CG iterations and Raleigh-Ritz procedure. At a cost of extra computation and storage, the Raleigh-Ritz procedure can be replaced by other alternative techniques using different spaces for the projection. In particular, following [94] we will consider a projection space built using the p_j descent directions from CG.

All the ideas briefly presented above in the context of CG for a single right-hand side naturally extend to BCG. In the sequel the thick-restart will be performed every m block iterations using the space spanned by

$$\mathscr{Z} = [W_k, \mathscr{P}_m] \tag{4.2}$$

where W_k the space spanned by the previously selected k eigenvector approximations, and the columns of $\mathscr{P}_m$ are the last m block descent directions P_j. In that context, the size of the search space is $m_d = k + m \times p$. In the sequel, we will refer to this as a spectral thick-restart, the update of the spectral information every m BCG iterations. These m BCG iterations will also be called a *refining cycle*.

4.3.2.2 Rayleigh-Ritz projection for thick-restart spectral update strategy

According to the Definition 2 of the Rayleigh-Ritz (RR) projection with respect to the space defined by Equation (4.2), computing the Ritz pairs reduces to solve the following generalized eigenvalue problem:

$$G^{(RR)} y_i - \theta_i F^{(RR)} y_i = 0, \text{ compute for } i = 1, \cdots, k,$$

where $\theta_1, \cdots, \theta_k$ are the k target eigenvalues (could be smallest and/or largest ones [26, 36]), computed at the thick-restart,

$$G^{(RR)} = \mathscr{Z}^H A \mathscr{Z} = \begin{bmatrix} W_k^H A W_k & W_k^H A \mathscr{P}_m \\ (A\mathscr{P}_m)^H W_k & \mathscr{D}_m \end{bmatrix} \in \mathbb{C}^{m_d \times m_d}, \tag{4.3}$$

where $\mathscr{D}_m = diag\{P_1^H A P_1, \cdots, P_m^H A P_m\}$ with $\mathscr{P}_m = [P_1, \cdots, P_m]$ the last m block descents directions computed by BCG for solving the first linear system. Note that the

Equation (4.3) can be reduced to

$$G^{(RR)} = \begin{bmatrix} W_k^H A W_k & 0 \\ 0 & \mathscr{D}_m \end{bmatrix} \in \mathbb{C}^{m_d \times m_d}, \tag{4.4}$$

when solving subsequent linear systems by D-BCG with the orthogonal property $W_k^H A \mathscr{P}_m = 0$. Furthermore,

$$F^{(RR)} = \mathscr{Z}^H \mathscr{Z} = \begin{bmatrix} W_k^H W_k & W_k^H \mathscr{P}_m \\ \mathscr{P}_m^H W_k & \mathscr{P}_m^H \mathscr{P}_m \end{bmatrix} \in \mathbb{C}^{m_d \times m_d}.$$

Then the new deflation space to be used for next refining cycle is updated by

$$W_k^{new} = [W_k, \mathscr{P}_m] Y \in \mathbb{C}^{n \times m_d},\ Y = [y_1, \cdots, y_k] \in \mathbb{C}^{m_d \times k}. \tag{4.5}$$

4.3.2.3 Harmonic-Ritz projection for thick-restart spectral update strategy

In this section, the Harmonic-Ritz (HR) projection technique is used to approximate eigenvectors at the thick-restart. According to Definition 1 and the search space described in Equation (4.2), the harmonic Ritz method leads to the solution of the generalized eigenproblem:

$$G^{(HR)} y_i - \theta_i F^{(HR)} y_i = 0, \text{ compute for } i = 1, \cdots, k,$$

where $\theta_1, \cdots, \theta_k$ are the corresponding k target (smallest and/or largest) eigenvalues,

$$G^{(HR)} = (A\mathscr{Z})^H A \mathscr{Z} = \begin{bmatrix} (AW_k)^H A W_k & (AW_k)^H A \mathscr{P}_m \\ (A\mathscr{P}_m)^H A W_k & (A\mathscr{P}_m)^H A \mathscr{P}_m \end{bmatrix} \in \mathbb{C}^{m_d \times m_d}$$

and

$$F^{(HR)} = \mathscr{Z}^H A \mathscr{Z} = \begin{bmatrix} W_k^H A W_k & W_k^H A \mathscr{P}_m \\ (A\mathscr{P}_m)^H W_k & \mathscr{D}_m \end{bmatrix} \in \mathbb{C}^{m_d \times m_d},$$

which will similarly reduce to the form described in the right-hand side of Equation (4.4) in the D-BCG case. Then the new deflation space to be used for next refining cycle is updated by Equation (4.5) as well.

4.3.2.4 Locally optimal thick-restart spectral update strategy

Locally optimal thick-restart (LO-TR) of the eigen-search space [122, Algorithm 3.2] is a special case of the thick-restart technique discussed in Section 4.3.2.1. Numerical experiments shown in [121, 122] indicate that the LO-TR outperforms the thick-restart Lanczos procedures for eigenvalue approximation in the single right-hand side context. Its extension to the block case is straightforward. Assuming that k is even, $k/2$ eigenvectors are computed using either RR or HR projection with respect to the space $\mathscr{Z} = [W_k, \mathscr{P}_m]$, they are defined by $Y_{k/2} = [y_1, \cdots, y_{k/2}] \in \mathbb{C}^{m_d \times k/2}$. Other $k/2$ eigenvectors are computed using either RR or HR projection with respect to the space $\bar{\mathscr{Z}} = [W_k, \mathscr{P}_{m-1}]$, they are denoted $\bar{Y}_{k/2} = [\bar{y}_1, \cdots, \bar{y}_{k/2}] \in \mathbb{C}^{m_d \times k/2}$. If we form, $Y = \texttt{orth}([Y_{k/2}, \tilde{Y}_{k/2}])$ where $\tilde{Y}_{k/2} = \begin{bmatrix} \bar{Y}_{k/2} \\ 0_{p \times k/2} \end{bmatrix}$, the LO-TR procedure computes k eigenpairs using RR or HR with

respect to the space $[W_k, \mathscr{P}_m]Y$.

Note that applying such refining strategy to the Init-CG [32, Algorithm 2] algorithm with RR projection is equivalent to the idea described in eigCG [108]. We refer the reader to [54, 55, 106, 107, 109] for more applications of LO-TR technique to the restart Lanczos-type methods.

According to Section 4.3.2.3-4.3.2.4, the implementation of recycling D-BCG with periodically refining deflation space computed by the Rayleigh-Ritz (RR) projection with LO-TR of the eigen-search space is denoted as D-BCG (RR) and presented in Algorithm 13 of Appendix A.2, its harmonic-Ritz (HR) counterpart is denotes as D-BCG (HR) and presented in Algorithm 14. Finally, the corresponding IB-D-BCG(RR/HR) variants with IB-mechanism [88] for partial convergence detecting are respectively described in Algorithm 15 and Algorithm 16.

4.4 Numerical experiments

In this section we report on numerical experiments carried out to illustrate the numerical behavior of the various techniques introduced in the previous sections. The default setting of the experiments is as follows: the block initial guess is set to be $0 \in \mathbb{C}^{n\times p}$, where p is the number of the right-hand sides. The right-hand sides $B = \texttt{randn}(n, p) = [b^{(1)}, b^{(2)}, \ldots, b^{(p)}] \in \mathbb{C}^{n\times p}$ are composed of p linearly independent vectors containing pseudo-random values drawn from the standard normal distribution (using the same seed when comparing these block methods). Unless otherwise indicated, the search space expansion policy used in conjunction with the partial convergence is based on the backward error η_b described in Section 2.3.1, the convergence threshold is set to be $\varepsilon = 10^{-8}$. Regarding the programming environment, in Section 4.4.1-4.4.4 MATLAB (R2019a) has been used, while in Section 4.4.5 it was Python 3.8. All the experiments were run on a personal Linux (double precision (Digits = 64) floating point arithmetic) PC-Intel (R) Core (TM) i7-8665U CPU ⓐ 1.90 GHz, 8 GB RAM.

4.4.1 Partial convergence with full rank and rank deficient set of right-hand sides

In Figure 4.1, we display on the left graph the convergence histories for the three variants of block CG, namely BCG, IB-BCG and IC-BCG, for solving single linear system built by the Kuu matrix. Similar observations as those made for BCR in Section 3.3, that is, IB-BCG converges the fastest in terms of number of matrix-vector products ($\#mvps$) and all the right-hand sides converge at the target accuracy at the same time. The BCG method converges for some right-hand sides to an accuracy below the target one at the price of additional $\#mvps$. The IC-BCG convergence exhibits a small plateau once the first right-hand side followed by a very fast converge. The size of the block p_j along the block iterations j are displayed in the right graph. As expected, the block size remains constant for BCG, while it reduces progressively for IB-BCG and more abruptly for IC-BCG as most of the right-hand sides converge at the same iterations for that example. In Table 4.1, we report the number of iterations ($\#iter$) and associated $\#mvps$ for all the matrices considered in this study. Similar comments can be made on those examples as what have been mentioned above for the Kuu matrix.

Matrix	*#mvps* BCG / IB-BCG / IC-BCG	*#iter* BCG / IB-BCG / IC-BCG
apache1	15420 / **13969** / 15017	**771** / 837 / 791
bcsstk15	2740 / **2614** / 2693	137 / 153 / 137
bcsstk16	1080 / **993** / 1047	54 / 55 / 54
bcsstk17	13540 / **12085** / 13284	**677** / 740 / 683
bcsstk18	6540 / **6303** / 6481	**327** / 343 / 329
bundle1	720 / **682** / 691	36 / 39 / 36
cbuckle	15080 / **14681** / 14920	**754** / 785 / 758
crankseg_1	6380 / **5714** / 6117	**319** / 337 / 324
crankseg_2	7200 / **6696** / 7055	**360** / 377 / 363
gridgena	9180 / **8846** / 9115	459 / 462 / 459
gyro	8600* / **7424*** / 8013	**430** / 475 / 452
s1rmq4m1	2800 / **2462** / 2720	140 / 153 / 140
s1rmt3m1	2840 / **2522** / 2764	142 / 150 / 142
s2rmq4m1	4480 / **3964** / 4399	**224** / 240 / 225
s2rmt3m1	5760 / **5190** / 5671	**288** / 318 / 289
s3rmq4m1	15660* / **9796*** / 11237*	**783** / 791 / 680
s3rmt3m1	19100* / **16325*** / 40424*	**956** / 5979 / 3470
shallow_water1	300 / 300 / 300	15 / 15 / 15
shallow_water2	560 / 560 / 560	28 / 28 / 28
ted_B_unscaled	560 / 543 / **541**	28 / 28 / 28

Table 4.1 – Number of matrix-vector product (*#mvps*) and block iterations (*#iter*) for all matrices listed in Table 3.1 for full rank right-hand sides, $p = 20$ and $\varepsilon = 10^{-8}$. The "$*$" indicates that the convergence based on the iterated residual norm was obtained but not with respect to the true residual norm.

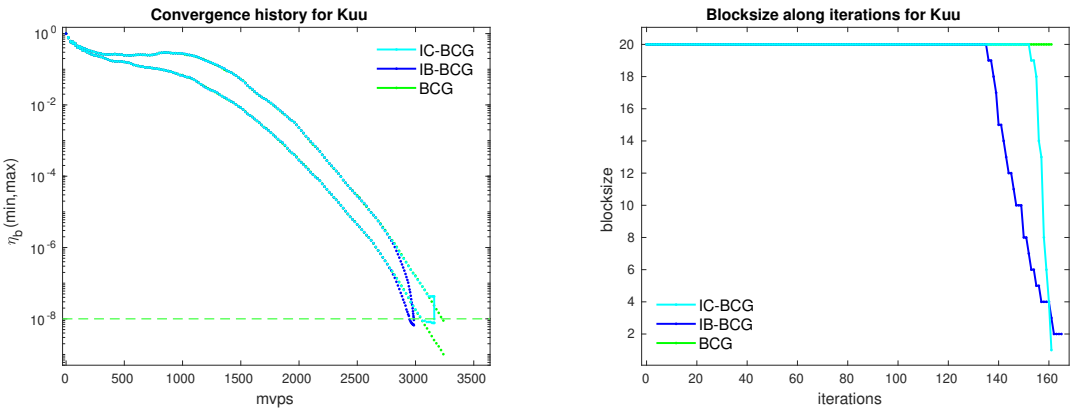


Figure 4.1 – Full rank right-hand sides with $B = \texttt{randn}(n, p)$ and $p = 20$. Left: convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ as a function of the number of matrix-vector products $\#mvps$. Right: block size p_j along iterations.

We now investigate the situation where the set of right-hand sides is rank deficient. For that purpose, we define $B = [B_{\text{pre}}, B_{\text{pre}}\ \texttt{randn}(p/2, p/2)]$ with $B_{\text{pre}} = \texttt{randn}(n, p/2)$, so that the rank is $p/2$. We display in the left graph of Figure 4.2 the convergence history of the three BCG variants. The first observation is that they are all robust to this rank deficiency; that they all detect if by reducing the block size to $p/2$ at the initial block iteration as it can be seen in the right graph. However a surprising behavior appears for the block size that suddenly increases for BCG and IC-BCG. This growth is actually caused by the internal threshold set in the MATLAB `orth` function that essentially return the left singular vectors of the truncated singular value decomposition (SVD) where the threshold is set to the unit round off of the working precision ($\approx 10^{-16}$ in our calculation). Because of the finite precision, some perturbations are gradually introduced to the residual blocks and their 10^{-16} numerical rank increases leading to the observed increase of the block size p_j. If the threshold of this `orth` function is changed into $\varepsilon = 10^{-8}$ with norm of B, we get a strictly decreasing behavior for the p_j as it can be seen in the right graph of Figure 4.3. These numerical results indicate that the breakdown-free mechanism proposed in [52] to prevent the rank deficiency base on a 10^{-16} numerical rank is too stringent and can be relaxed to reduce the computational cost, which is similar to the idea of previous discussed IB variants. In the rest of this chapter, we kept the default setup of the `orth` function unchanged and do not apply the tuning of the truncated SVD to the target accuracy; our objective is to make a comparison of the IB variants with the state-of-the-art of breakdown-free BCG.

4.4.2 Influence of the value of the convergence threshold

In this section, we illustrate how the value of the convergence threshold effects the performance and robustness of the BCG variants. The convergence thresholds are set as $\varepsilon = 10^{-1}, 10^{-2}, 10^{-5}, 10^{-12}$ and the number of right-hand sides is kept to $p = 20$. With these numerical settings, the convergence histories of solving matrix Kuu are depicted in Figure 4.4. A more exhaustive set of numerical results are reported in Table 4.2 in terms of $\#mvps$ and $\#iter$ for other test examples. We will not elaborate much on these results as the observations are essentially the same as those discussed in Section 3.3.2 for the

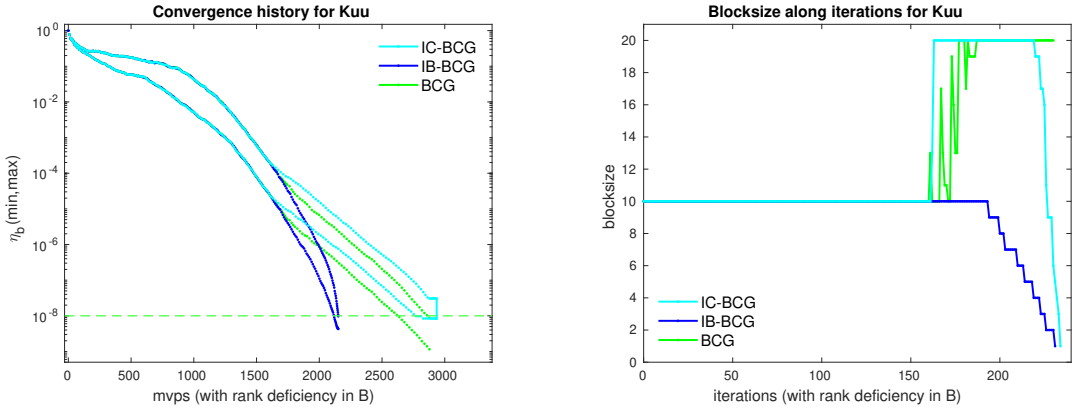


Figure 4.2 – Rank deficient set of right-hand sides. $B = [B_{\text{pre}}, B_{\text{pre}}\ \texttt{randn}(p/2, p/2)]$ with $B_{\text{pre}} = \texttt{randn}(n, p/2)$, $p = 20$. Left: convergence histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ as a function of $\#mvps$. Right: block size p_j along iterations..

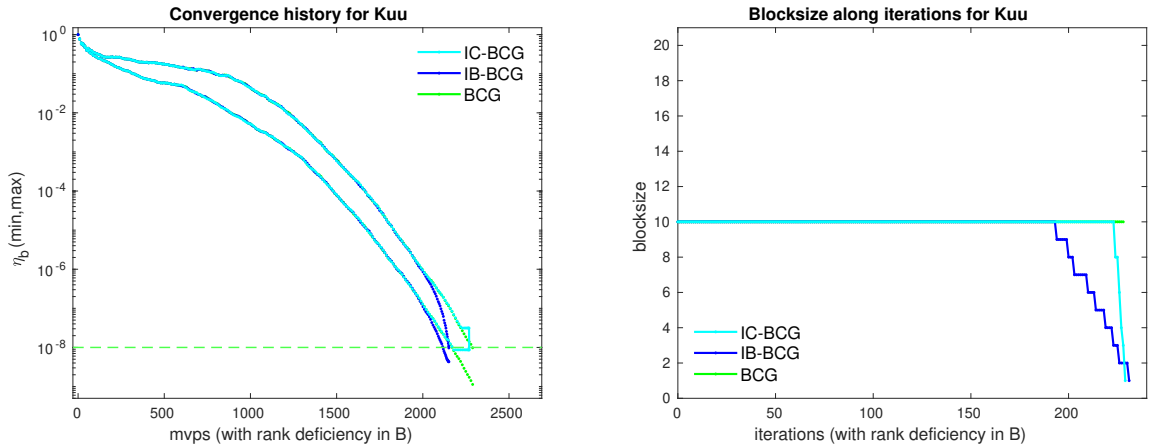


Figure 4.3 – Same case as Figure 4.2 but using $\varepsilon = 10^{-8}$ as internal threshold for the truncated SVD of the MATLAB function `orth`.

BCR variants. That is, the IB-BCG performs generally the best in terms of number of $\#mvps$ and BCG in terms of block $\#iter$.

4.4.3 Influence of the number of right-hand sides

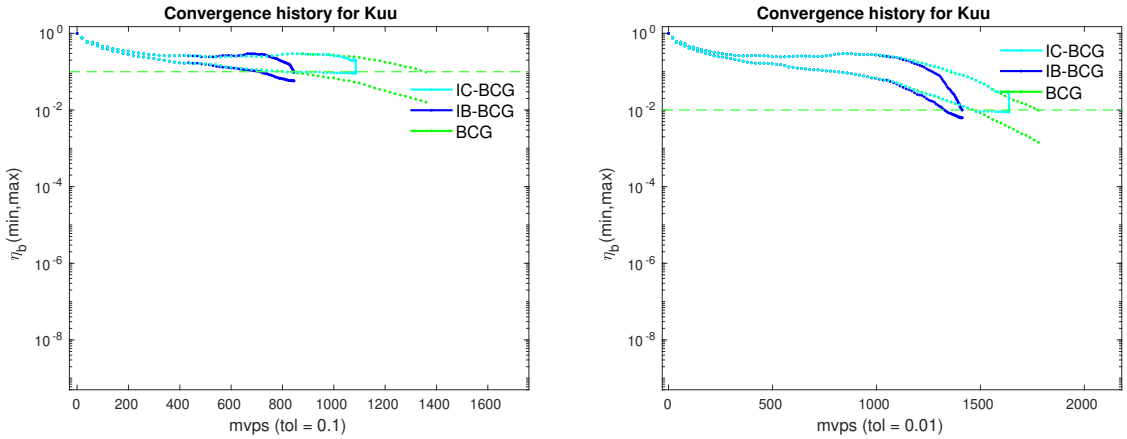
In this section, we illustrate the impact of the number of the right-hand sides on the performance of the BCG variants by varying $p = 5, 10, 30, 40$. The performances in terms of $\#mvps$ and $\#iter$ are reported in Table 4.3 that shows the benefit of the IB variant. The convergence histories for solving matrix apache1 with $p = 5$ and $p = 40$ are respectively described in the left and right graph of Figure 4.5, from which it is easy to notice that the larger p, the clearer the gap between the smallest and largest backward errors.

Matrix	ε	10^{-1}		10^{-2}		10^{-5}		10^{-12}	
	Method	*#mvps*	*#iter*	*#mvps*	*#iter*	*#mvps*	*#iter*	*#mvps*	*#iter*
apache1	BCG	7000	350	8520	**426**	12340	**617**	19040*	**952**
	IB-BCG	3072	685	**6670**	739	**10752**	718	**17997***	1036
	IC-BCG	**1742**	**123**	7727	454	11788	632	18706*	965
bcsstk18	BCG	4160	**208**	4860	**243**	5940	**297**	7660*	383
	IB-BCG	**3720**	304	**4396**	297	**5728**	329	**7029***	**372**
	IC-BCG	3962	219	4728	248	5850	299	7307*	378
cbuckle	BCG	7220	**361**	8860	**443**	12580	**629**	18380*	**919**
	IB-BCG	**6541**	372	**8253**	456	**12024**	646	**17857***	928
	IC-BCG	6994	363	8683	448	12325	634	18190*	921
crankseg_2	BCG	2040	**102**	3040	**152**	5220	**261**	9700*	**485**
	IB-BCG	**1257**	133	**2503**	180	**4707**	281	**9225***	499
	IC-BCG	1659	110	2830	156	5057	265	9567*	487
gridgena	BCG	6620	331	7740	387	8540	427	10100*	**505**
	IB-BCG	**5112**	354	7799	643	**8182**	433	**9690***	525
	IC-BCG	5761	**316**	**7535**	387	8469	427	9964*	506
Kuu	BCG	1360	**68**	1780	**89**	2620	**131**	3860	**193**
	IB-BCG	**846**	84	**1414**	102	**2345**	138	**3662**	197
	IC-BCG	1087	72	1639	90	2535	132	3810	194

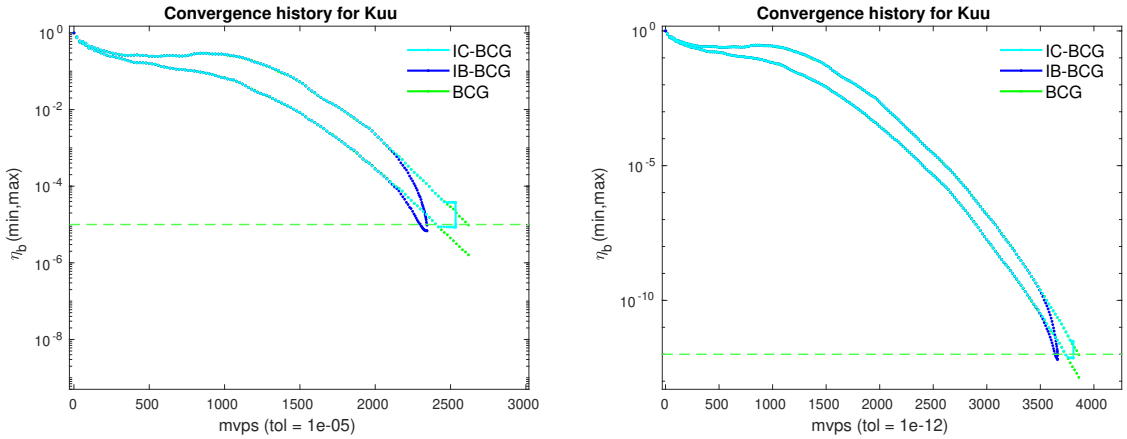
Table 4.2 – Numerical results of BCG variants in terms of both *#mvps* and *#iter* for a selection of the of testing matrices listed in Table 3.1 with the right-hand sides $B = \texttt{randn}(n, p)$, $p = 20$ and $maxMvps = 5000 \times p$ and various convergence thresholds ε.

Matrix	p	5		10		30		40	
	Method	*#mvps*	*#iter*	*#mvps*	*#iter*	*#mvps*	*#iter*	*#mvps*	*#iter*
apache1	BCG	7730	**1546**	10900	**1090**	19440	**648**	22320	**558**
	IB-BCG	**7175**	1639	**10030**	1168	**17277**	756	**20001**	666
	IC-BCG	7667	1547	10667	1103	18694	667	21695	577
bcsstk18	BCG	4395	**879**	5510	**551**	7260	**242**	7880	**197**
	IB-BCG	**4130**	895	**5345**	582	**6959**	254	**7473**	209
	IC-BCG	4260	883	5449	553	7178	243	7773	199
cbuckle	BCG	6525	1305	10570	**1057**	18450	**615**	20920	**523**
	IB-BCG	**6376**	1312	**10209**	1073	**17755**	633	**20130**	547
	IC-BCG	6469	**1299**	10423	1060	18305	617	20683	525
crankseg_2	BCG	2845	**569**	4290	**429**	9780	**326**	12160	**304**
	IB-BCG	**2642**	586	**4088**	437	**9053**	341	**11242**	321
	IC-BCG	2755	576	4188	431	9521	328	11943	308
gridgena	BCG	7395	1479	8180	818	10110	**337**	10960	274
	IB-BCG	**7222**	1493	**7997**	820	**9593**	341	**10266**	278
	IC-BCG	7358	**1478**	8151	818	10023	338	10820	274
Kuu	BCG	1625	**325**	2290	**229**	3960	132	4480	**112**
	IB-BCG	**1517**	335	**2136**	234	**3622**	137	**4115**	117
	IC-BCG	1614	326	2262	230	3860	132	4363	113
s2rmt3m1	BCG	3955	**791**	4750	**475**	6360	**212**	6960	**174**
	IB-BCG	**3772**	821	**4434**	511	**5696**	241	**6092**	201
	IC-BCG	3909	793	4693	477	6259	213	6837	177
ted_B	BCG	135	27	270	27	870	29	1160	29
	IB-BCG	135	27	270	27	864	33	1134	30
	IC-BCG	135	27	270	27	**844**	29	**1128**	29

Table 4.3 – Numerical results of BCG variants with different number of right-hand sides ($p = 5, 10, 30, 40$) in both terms of *#mvps* and *#iter* for parts of testing matrices listed in Table 3.1 with $B =$ `randn`(n, p), $maxMvps = 5000 \times p$ and $\varepsilon = 10^{-8}$.

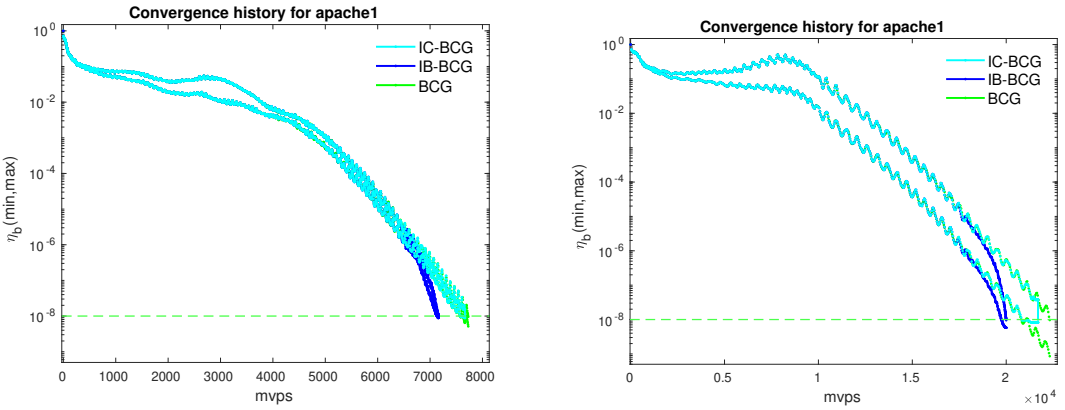


$\varepsilon = 10^{-1}$ $\varepsilon = 10^{-2}$



$\varepsilon = 10^{-5}$ $\varepsilon = 10^{-12}$

Figure 4.4 – Convergence history for solving linear systems built by Kuu ($B = \texttt{randn}(n, p)$, $p = 20$ and $maxMvps = 5000 \times p$) with different value of the convergence threshold.



$p = 5$ $p = 40$

Figure 4.5 – Histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ of for matrix apache1 with different number of right-hand sides p under the setting as $B = \texttt{randn}(n, p)$, $maxMvps = 5000 \times p$ and $\varepsilon = 10^{-8}$.

4.4.4 Experiments with individual convergence threshold

In this section, we illustrate that the partial convergence mechanism can also be adapted to cope with different individual convergence thresholds in the BCG context. We illustrate this feature in Figure 4.6 where we consider the solution for matrix crankseg_1 with $p = 20$ right-hand sides and set the convergence threshold to $\varepsilon = 10^{-4}$ for the first $p/2$ right-hand sides and $\varepsilon = 10^{-8}$ for the last $p/2$ ones. It can be seen that numerically it works and enables some saving in terms of $\#mvps$. More numerical results in terms of $\#mvps$ and block $\#iter$ are summarized in Table 4.4.

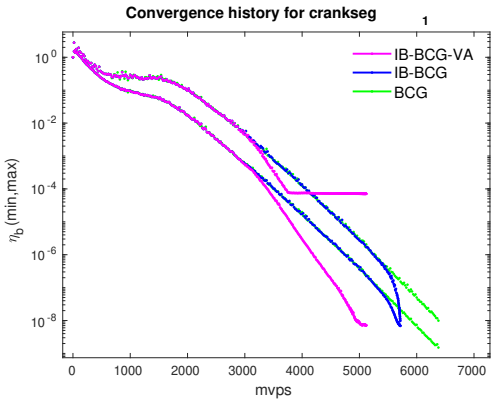


Figure 4.6 – Histories of the largest/smallest backward errors $\eta_{b^{(i)}}$ at each $\#mvps$ for matrix crankseg_1 with convergence threshold equal to 10^{-4} for the first $p/2$ right-hand sides and 10^{-8} for the last $p/2$ ones ($p = 20$).

Matrix	$\#mvps$ BCG / IB-BCG / IB-BCG-VA	$\#iter$ BCG / IB-BCG / IB-BCG-VA
cbuckle	15080 / 14681 / **14145**	**754** / 785 / 980
crankseg_1	6380 / 5714 / **5124**	**319** / 337 / 482
crankseg_2	7200 / 6696 / **6183**	**360** / 377 / 548
Kuu	3240 / **2987** / 3100	**162** / 166 / 326
shallow_water2	560 / 560 / **420**	28 / 28 / 28
ted_B_unscaled	560 / 543 / **411**	28 / 28 / 28

Table 4.4 – Numerical performances in terms of $\#mvps$ and $\#iter$ with $B = \texttt{randn}(n, p)$, $p = 20$ and $maxMvps = 5000 \times p$.

Before moving to results for deflation techniques, we briefly report on a numerical observation on the A-norm of the error that we did not expect nor can we clearly explain it. Its monotonically decreasing trend is not affected by the selection of the directions used to expand the search space as long as the A-conjugacy is retained, as can be observed in Figure 4.7. What was not expected is to have a similar behavior as the one observed for the minimum norm residual methods, that is, all the convergence history eventually converge to the same (or very close) value.

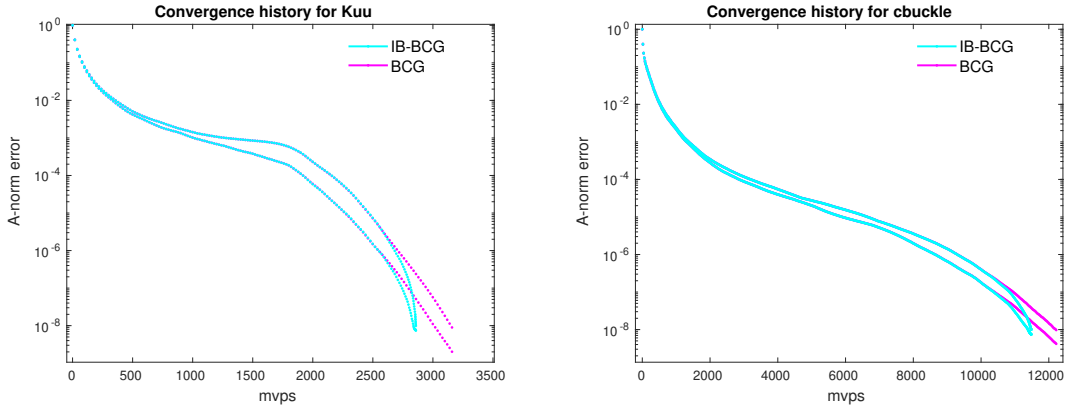


Figure 4.7 – Envelop of the A-norm of the error along the iterations, $p = 20$ for the matrices Kuu and cbuckle.

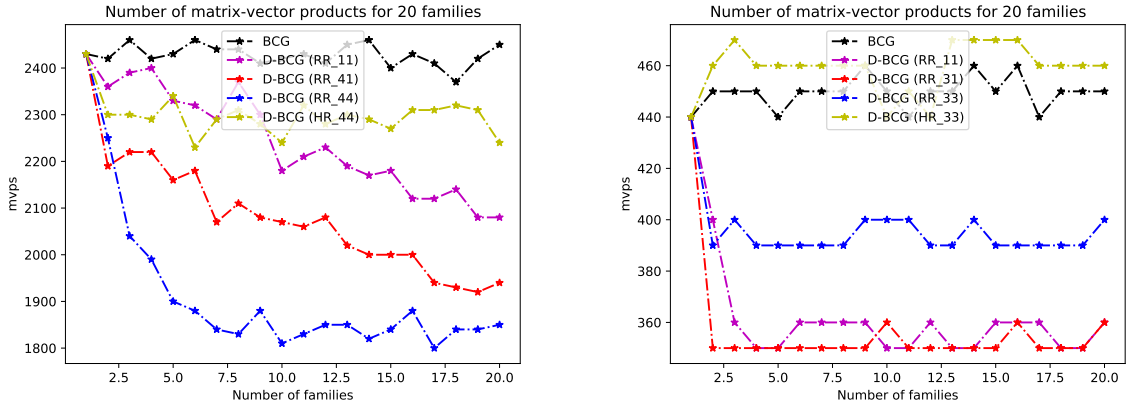
4.4.5 Benefits of refining the deflation space between the families

In this section, we report on numerical experiments where the LO-TR policy described in Section 4.3.2.4 is used and combined with the partial convergence management. While the LO-TR spectral calculation can be performed as long as the block iterations have not converged, it might not pay off since the spectral quality might not be significantly improved. Many options exist and we only consider very few in this section. We distinguish the first family solution where we start without deflation space from the subsequent ones. We denote by (RR, ℓ_1, ℓ_2) and (HR, ℓ_1, ℓ_2) the option where LO-TR is used based on the RR projection and the HR projection, respectively, where ℓ_1 denotes the number of thick-restarts that are performed when solving the first family and ℓ_2 the number thick-restarts for the subsequent families. For all the experiments, we target eigenvectors associated with the k smallest approximate eigenvalues in magnitude. We use as preconditioner either the simple Jacobi preconditioner or a state-of-the-art Algrebraic Multigrid (AMG) preconditioner implmented in [7]. The number of families is $\ell = 20$, the number of right-hand sides of each family is $p = 10$, the size of the deflation space is $k = 20$, and the length of refining cycle is $m_d = 100$.

We report in Table 4.5, the number of matrix-vector products ($\#mvps$) and the number of block iterations ($\#iter$) for the different variants of BCG using or not the partial convergence management mechanism. The first comment is that the benefit of the IB mechanism is less significant compared to what we observed in the non HPD case. The second observation is that the RR projection is more effective than the HR one, possibly because the smallest eigenvalues are external for HPD matrices. Although it would deserve a more exhaustive study, the third comment is that it seems to be more important capturing good spectral information when solving for the first family rather than trying to refine it later when solving the subsequent ones. Lastly, the benefit of the deflation is larger when combined with a simple preconditioner such as Jacobi rather than when combined with a scalable preconditioner such as AMG that has already clustered all the eigenvalues around one. Those observations can possibly appear clearer when looking at the Figure 4.8 and 4.9.

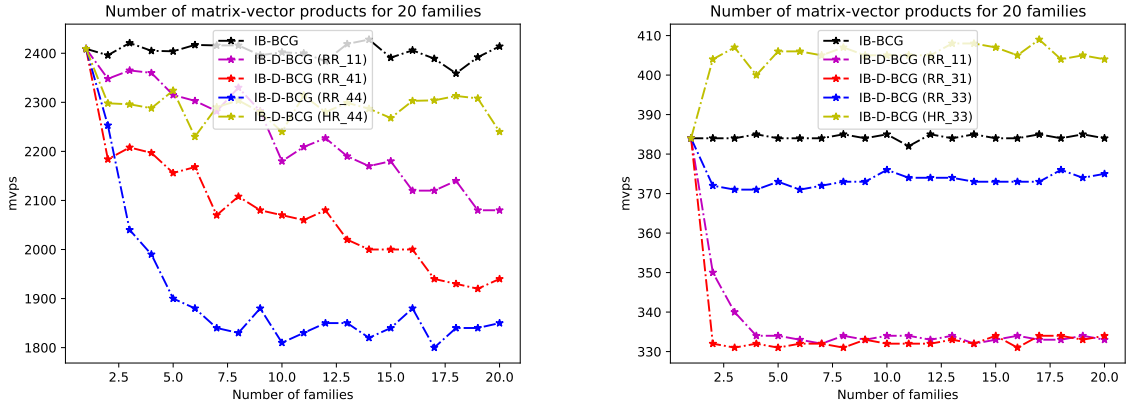
			with IB		w/o IB	
# families	Matrix	Method	*#mvps*	*#iter*	*#mvps*	*#iter*
3	Kuu (Jacobi)	BCG	7243	736	7350	735
		D-BCG (HR,4,4)	7034	692	7080	692
		D-BCG (RR,1,1)	7170	725	7290	725
		D-BCG (RR,4,1)	6812	683	6870	683
		D-BCG (RR,4,4)	**6721**	**660**	**6760**	**660**
20	Kuu (Jacobi)	BCG	48064	4867	48560	4856
		D-BCG (HR,4,4)	45875	4445	48560	4856
		D-BCG (RR,1,1)	44690	4454	44890	4451
		D-BCG (RR,4,1)	41540	4126	41620	4124
		D-BCG (RR,4,4)	**38232**	**3675**	**38250**	**3673**
3	Kuu (AMG)	BCG	1153	140	1360	136
		D-BCG (HR,3,3)	1194	128	1380	126
		D-BCG (RR,1,1)	1076	122	1230	119
		D-BCG (RR,3,1)	**1047**	**114**	**1160**	**112**
		D-BCG (RR,3,3)	1131	115	1250	113
20	Kuu (AMG)	BCG	7685	930	8990	899
		D-BCG (HR,3,3)	8089	822	9180	804
		D-BCG (RR,1,1)	6741	694	7240	686
		D-BCG (RR,3,1)	**6699**	**684**	**7120**	**674**
		D-BCG (RR,3,3)	7475	687	7910	677
3	cbuckle (AMG)	BCG	7860	820	8160	816
		D-BCG (HR,3,3)	7747	802	8120	800
		D-BCG (RR,1,1)	7840	815	8160	812
		D-BCG (RR,31)	**7665**	802	**8050**	801
		D-BCG (RR,3,3)	7672	**796**	8060	**794**
3	crankseg (AMG)	BCG	2675	324	3150	315
		D-BCG (HR,3,3)	2679	308	3110	299
		D-BCG (RR,1,1)	2659	310	3050	301
		D-BCG (RR,3,1)	**2559**	**291**	**2870**	**283**
		D-BCG (RR,3,3)	2631	294	2970	285

Table 4.5 – Numerical of IB-BCG and IB-D-BCG variants in terms of *#mvps* and *#iter*.



Jacobi preconditioner AMG preconditioner

Figure 4.8 – Number of $\#mvps$ for a sequence of 20 families using $p = 10$, k=20, $m_d = 100$ for the Deflated variants without partial convergence management on the Kuu matrix.



Jacobi preconditioner AMG preconditioner

Figure 4.9 – Number of $\#mvps$ for a sequence of 20 families using $p = 10$, k=20, $m_d = 100$ for the Deflated variants with partial convergence management on the Kuu matrix.

4.5 Concluding remarks

In this chapter, we first investigate the possibility to use the partial convergence detection presented in the minimum residual norm context to the BCG algorithm [77] as a heuristic for the symmetric positive definite linear systems. This numerical mechanism is only a heuristic because the resulting search space expansion policy still relies on residual norm minimization, while ideally it should be based on an A-norm error minimization principle. Unfortunately, this A-norm error directions are not a by-product of BCG as the residuals for the numerical block methods considered in the previous chapters. Then, inspired by the idea of reusing generated information to approximate spectral information for accelerating convergence, we devise a corresponding subspace recycling strategy. More precisely, we apply the thick-restart ideas [125] introduced in the Lanczos method for eigenpair calculation in the BCG algorithm. This new variant periodically

refines the spectral information on a small window defined when solving the linear systems, which provides a practical way to approximate spectral information with a moderate computational effort. Finally, these partial convergence detection and subspace recycling strategies can be efficiently combined to design a deflated block conjugate gradient algorithm for sequences of symmetric positive definite linear systems.

Part II

Hybridization of machine learning and numerical linear algebra techniques for scientific computing

Chapter 5

Learned minimum residual solvers for the Helmholtz equations

5.1 Introduction

In the past two decades, machine learning techniques, particularly deep learning techniques based on deep neural networks (DNN) have had great success in applications, such as image recognition [43], speech recognition [47], computer vision [56,57]. They have also been successfully applied in simulating equations from diverse fields such as climate analytics [58], weather forecasting [24, 30], earth system [30] and ocean science [100]. Research on scientific machine learning [64, 75] based on DNN has been furthermore increasingly applied to scientific computing and computational engineering, particularly for problems related to partial differential equations (PDEs) [22,31,40,48,59,61,62]. This is mainly due to their ability to effectively approximate complex functions arising in diverse scientific disciplines, such as nonlinear PDEs [85] and high-dimensional PDEs [97] that could be challenging for the traditional iterative methods. One of the earlier directions focuses on devising recommendation systems based on machine learning algorithms for classification to assist traditional solvers in the simulation process of specific PDEs, such as the SALSA and Lighthouse projects [10, 51, 101, 102]. Such approaches mainly rely on supervised machine learning techniques. Examples include applying random forest and K-nearest neighbors for auto-selecting the best solvers for transport problems [21], exploiting reinforcement learning to adaptively choose the best restart parameter from a range of values to improve the performance of restarted GMRES [83], and suggesting specific preconditioners for iterative solvers for a target system [2, 39, 95, 128]. A second direction is devising data-driven DNN approaches to build a solver directly for the solution of PDEs, such as the recent developments on Poisson solvers [90, 115, 118]. Another important example of this trend is the physics-informed neural networks (PINNs) [14, 50, 63–65, 70, 74, 96, 123], that are mesh-free, and use mostly unsupervised training with a physics-based loss function that does not require labeled data, and could be applied to different types of PDEs for solving inverse and forward problems. Once trained, those solvers usually exhibit a rather poor accuracy in the inference phase. We refer the reader to Adcock and Dexter's recent work [3, 38, 76] for more details about the challenges and mathematical interpretation of methods based on DNN. The idea of

hybridizing machine learning and traditional solvers has also been investigated. We may mention for instance the works in [23, 68, 87, 116, 130]. For example, the authors in [49] proposed a hybrid fluid solver scheme for the Poisson equation. In that work, the DNN solver introduced in [118] computes an initial guess for classical Jacobi iterations that generally converge quickly thanks to the good enough estimate of the solution. Another example is the data-driven models trained in the reference [119] to predict the deflation subspaces [71,72] for accelerating the convergence of the GMRES method for the solution of frequency-domain Maxwell's equations.

Many research activities have addressed the solution of the Helmholtz equation [8,13, 33,84,117,124], which could be trained with a physics-based loss function by unsupervised learning because of the practical physics background or be trained by supervised learning if the true solution is available. In particular, a data-driven learned iterative solver built using DNN with a modified UNet [89] architecture is presented in [105]. A way of hybridizing the DNN approach with traditional ones is described in [87], which presents an idea of interspersing a Krylov subspace iterative solver with NN corrections to improve the convergence for solving the Helmholtz equation with a fixed frequency and source location. Except for such straightforward combinations with traditional solvers for the Helmholtz equations, the idea of using extra physical knowledge to improve the performance of DNN based approaches could be noticed in [4, 11]. On the other hand, the DNN approach has been used to improve the performance of traditional solvers, for example, data-driven models are trained in [119] to predict the deflation subspace [71, 72] to accelerate the convergence of the traditional GMRES method for the solution of frequency-domain Maxwell's equations, and [67] describes a way to accelerate GMRES by deep learning for the Poisson equation.

In this chapter, we focus on the DNN solver introduced in [105] for the solution of Helmholtz equations for which we discuss two options for improvement. In Section 5.3.1 we propose to change the numerical scheme used in the training phase. We introduce a relaxation parameter in the nonlinear fixed point scheme. The major benefit of this modification is that the training of the DNN becomes more robust and consequently faster to converge. In Section 5.3.2, we investigate the use of the trained network as a nonlinear preconditioner for two flexible Krylov subspace solvers [92], namely the flexible full orthogonalization method (FFOM) and flexible GMRES (FGMRES) [91]. Finally, in Section 5.5, we summarize our work and discuss possible future directions.

5.2 Summary of related work

Our work follows and extends the study presented in [105] for the solution of the 2D Helmholtz equation with variable speed-of-sound fields denoted by c. The problem is given by

$$\nabla^2 u + \varpi(c)u = f \tag{5.1}$$

with appropriate absorbing conditions to bound the computational domain. After discretization on a 2D Cartesian grid (96×96), the solution of the discrete problem reduces to the solution of a linear system

$$A(c)x(c) = b \tag{5.2}$$

where $A(c) \in \mathbb{C}^{n\times n}$, $x(c)$ and $b \in \mathbb{C}^n$, where n denotes the number of degrees of freedom. We refer the reader to [105] for the numerical details related to the boundary conditions and the discretization techniques. In the following discussion, we omit the dependence on c for the sake of simplicity of exposition.

5.2.1 Quick introduction to neural network

Before describing the detailed simulation processes of such neural network (NN) based iterative solvers for the Helmholtz equation, some machine learning glossaries are briefly summarized in the following.

- **Supervised learning and unsupervised learning**: The former machine learning model requires labeled input and output data during the training phase, while the later one processes with raw and unlabeled training data.
- **Gradient descent**: An iterative learning algorithm that uses the training data set to update a NN model.
- **Batch**: The batch size is a hyperparameter of gradient descent that controls the number of training samples to work through before the model's internal parameters are updated. The definition of batch size includes three cases:
  - Batch Gradient Descent $\longleftrightarrow$ batch size = size of training dataset,
  - Stochastic Gradient Descent (SGD) $\longleftrightarrow$ batch size = 1,
  - Mini-batch Gradient Descent $\longleftrightarrow$ 1 < batch size < size of training set.
- **Epoch**: A full training pass over the entire dataset such that each example has been seen once.
- **Gradient update step**: One iteration of gradient descent.
- **Loss function**: An user-defined metric for evaluating how well the NN fits a data set, i.e., how far the value predicted by the NN is from the true value. The better the prediction, the lower the output of the loss function, referred to as the loss.
- **Training loss**: A measure of how well the NN fits the training data set, which is a subset of the full data set used during training. The gradient descent algorithm aims at minimizing this loss.
- **Validation loss**: A measure of how well the NN fits the validation data set, which is a subset of the full data set that is disjoint from the training data set. The validation loss is used during training to determine whether the NN needs further training.
- **Testing loss**: A measure of how well the NN fits the testing data set, which is a subset of the full data set that is disjoint from the training and validation data sets. The testing loss is used to evaluate the prediction capabilities of the trained NN on inputs that were never seen before by the NN.

5.2.2 The basic nonlinear fixed point iteration scheme

Based on the notations above, the DNN is trained using a reinforced unsupervised technique, as no knowledge of the true solution is available. The training is based on the following nonlinear Richardson iteration:

$$r_j = b - Ax_j, \tag{5.3}$$
$$(\Delta x_j, h_{j+1}^{NN}) = f_\theta(x_j, r_j, h_j^{NN}), \tag{5.4}$$
$$x_{j+1} = x_j + \Delta x_j, \tag{5.5}$$

where f_θ is the NN with a modified UNet [89] architecture and learnable internal parameters θ (i.e., the weights and biases of the DNN), r_j is the residual associated with the current iterate x_j, and Δx_j is the nonlinear update. Note that the NN uses recurrent hidden states h^{NN} that contain information from all preceding actions and observations to give enough information to the network to specify the next update [69]. For the sake of simplicity, these will be omitted in the inference equation (5.4), which will simply be written as $\Delta x_j = f_\theta(x_j, r_j)$. The network is trained using the mini-batch gradient descent algorithm with batch size equal to 32 for updating the NN internal parameters using a residual-based loss function. We refer the reader to the original paper [105] for a detailed and complete description of the NN architecture and its training processes. We also mention that a similar work is presented in [87], in which the training is performed using a supervised learning technique thanks to the known/labeled solutions and using an UNet architecture.

5.3 Improving the training and the inference phase

5.3.1 Accelerating the training phase

The iteration scheme described in Equation (5.3)-(5.5) can be viewed as a fixed point Richardson iteration [12], with the nonlinear correction that is iteratively learned by the NN. The inputs of the NN are the current iterate solution, its associated scaled residual, and a learned hidden state. Based on this information, we denote the NN solver [105] as the R(R-NN) algorithm, where the term R denotes a Richardson-like iteration scheme, and R-NN stands for corresponding NN-inference. The resulting algorithm is presented in Algorithm 11.

A possible weakness of the R-NN scheme is that the update along the direction computed by the NN might be large in the first initial steps of the training. One possible remedy is simply to introduce a relaxation parameter, so that the step size is chosen to minimize the norm of the residual associated with the next iterate. Consequently in each iteration, we perform the update $x_{j+1} = x_j + \omega_j \Delta x_j$ where $\omega_j = \text{argmin}_\omega \|r_j - \omega(A\Delta x_j)\|$. This 4D (batch size, 2, 96, 96) optimization has a simple solution that reads

$$\omega_j = \frac{(A\Delta x_j)^H r_j}{\|A\Delta x_j\|^2}.$$

Algorithm 11 *Learned Richardson iterative solver – R(R-NN)*

Require: Nonsingular coefficient matrix A
Require: The NN $f_\theta(x, r)$
Require: The zero initial guess x_0
Require: The corresponding initial residual $r_0 = b - Ax_0$
Require: A scaling factor $\alpha = 10^3$ to be used on the input and output of the NN
for $j = 0, 1, 2, \ldots, m$ **do**
  if normalize the input residual **then**
    $z = f_\theta(x_j, \alpha r_j/\|r_j\|)/\alpha$
  else
    $z = f_\theta(x_j, \alpha r_j)/\alpha$
  end if
  / Update the iterate and residual */*
  $x_{j+1} = x_j + z$
  $r_{j+1} = b - Ax_{j+1}$
end for
return x_{j+1}, r_{j+1}

The new iteration can be described in equation form as

$$r_j = b - Ax_j, \tag{5.6}$$
$$\Delta x_j = f_\theta(x_j, r_j), \tag{5.7}$$
$$\omega_j = \underset{\omega}{\operatorname{argmin}} \|r_j - \omega A\Delta x_j\| = \frac{(A\Delta x_j)^H r_j}{\|A\Delta x_j\|^2}, \tag{5.8}$$
$$x_{j+1} = x_j + \omega_j \Delta x_j. \tag{5.9}$$

We refer to this new scheme as MRR(MRR-NN), for Minimum Residual Richardson.

Based on Equations (5.6)–(5.9), the pseudocode of MRR(MRR-NN) is described in Algorithm 12, which is sketched in Figure 5.1 in a similar manner as R(R-NN) in [105].

5.3.2 Subspace solver with flexible neural network preconditioner

The inference of the trained NN presented in the previous section returns the correction for a pair of vectors being an approximated solution x and associated residual r for a linear system of the form (5.2) arising from the discretization of any Helmholz problem as (5.1). The inference can consequently be seen as the approximated inverse of A applied to r for knowing x. In that respect, it can be used as a nonlinear preconditioner in a flexible subspace method such as FFOM or FGMRES [91]. These two flexible subspace methods rely on the following extended Arnoldi relation

$$AZ_j = V_j H_j + h_{j+1,j} v_{j+1} e_j^T = V_{j+1} \underline{H}_j$$

with $V_{j+1} = [v_1, v_2 \ldots, v_{j+1}]$, $V_{j+1}^H V_{j+1} = I_{j+1}$ and $H_j \in \mathbb{C}^{j \times j}$ is an upper Hessenberg matrix. The FGMRES method characterizes the approximate solution $x_j = Z_j y_j$ in the space spanned by Z_j where $y_j = \operatorname{argmin}_y \|\beta e_1 - \underline{H}_j y\|$ with $\beta = \|r_0\|$. And then the residual $r_j = V_{j+1}(\beta e_1 - \underline{H}_j y_j)$. Similarly, FFOM computes the approximate solution

Algorithm 12 *Learned minimum residual Richardson iterative solver – MRR(MRR-NN)*

Require: Nonsingular coefficient matrix A
Require: The NN $f_\theta(x, r)$
Require: The zero initial guess x_0 and κ_ω
Require: The corresponding initial residual $r_0 = b - Ax_0$
Require: A scaling factor $\alpha = 10^3$ or $\alpha = 1$ to be used in the input and output of the NN
for $j = 0, 1, 2, \ldots, m$ **do**
  if normalize the input residual **then**
    $z = f_\theta(x_j, \alpha r_j / \|r_j\|)/\alpha$
  else
    $z = f_\theta(x_j, \alpha r_j)/\alpha$
  end if
  / Update the iterate */*
  $\omega_j = \underset{\omega}{\operatorname{argmin}} \|r_j - \omega Az\| = \dfrac{(Az)^H r_j}{\|Az\|^2}$
  if $10^{-8} \leq |\omega_j| \leq 10^8$ **then**
    $x_{j+1} = x_j + \omega_j z$
  else
    $x_{j+1} = x_j + z$, and $\kappa_\omega = \kappa_\omega + 1$
  end if
  $r_{j+1} = b - Ax_{j+1}$
end for
return x_{j+1}, r_{j+1}, κ_ω

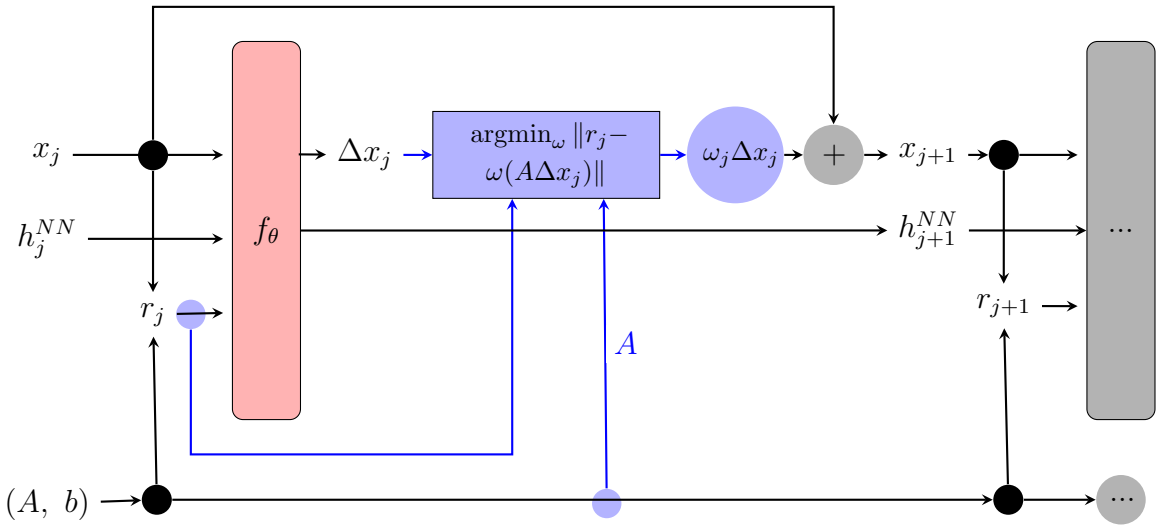


Figure 5.1 – Solution-update architecture of MRR(MRR-NN): $x_{j+1} = x_j + \omega_j \Delta x_j$, and f_θ is the NN with a modified Unet architecture.

$x_j = Z_j y_j$ in the space spanned by Z_j where $H_j y_j = \beta e_1$ so that

$$r_j = b - Ax_j = h_{j+1,j}(e_j^T y_j) v_{j+1}. \tag{5.10}$$

The above observations suggest different ways to use the inference of the trained NN as a nonlinear preconditoner for FFOM and FGMRES, where we explicitly set the scaling factor α in Algorithm 12 to 1:

Strategy 1: "Krylov driven" $z_j \approx A^{-1} v_j$ compute $z_j = f_\theta(0, v_j)$.

Strategy 2: "NN driven" compute $z_j = f_\theta(x_{j-1}, r_{j-1}/\|r_{j-1}\|)$. In FFOM, this reduces to $z_j = f_\theta(x_{j-1}, v_j)$ because of equation (5.10), that corresponds also to $z_j \approx A^{-1} v_j$. In FGMRES, the resulting search space Z_j is spanned by the successive corrections computed by the trained NN for the sequence of FGMRES iterates.

We notice that the first strategy aims at computing $z_j \approx A^{-1} v_j$ following the original spirit of the flexible preconditioners, that is using the Krylov basis v_j as input. Strategy 2 uses the trained NN in the same manner as the training NN is exploited in the fixed point iteration schemes, that is using the current iterate solution and associated residual as input.

For a fast overview of the proposed improvements, we refer the reader to Figure A.1 in Appendix A.3 for the skeleton of these two main contributions described in this section.

5.4 Numerical experiments

For the numerical experiments we consider the same data set as in [105, Section 2.3]. The training data set is composed of 9000 matrices corresponding to 9000 speed-of-sound fields ϖ (refer to Equation (5.1)) and the validation data set includes 1000 other matrices. The two NNs, namely R-NN and MRR-NN, have been trained on a machine with 2x20-core Skylake Intel Xeon Gold 6148 CPUs, 2 NVIDIA Volta V100 GPUs and 1TB memory,

under single precision (i.e., float-32 (fp32)) with batch size number equal to 32, and have used the Adam optimizer [53] with constant learning rate equal to 10^{-3}.

This section is organized as follows. In Section 5.4.1 we investigate the impact of normalizing the residual vector that is an input parameter of the NN. Next, in Section 5.4.2 we study the performance of the trained networks both in terms of convergence speed and in terms of attainable accuracy when they are used in a fixed point iteration scheme or as a preconditioner in a subspace solver.

All the numerical experiments presented in this chapter have been obtained using PyTorch, which does not support complex arithmetic.

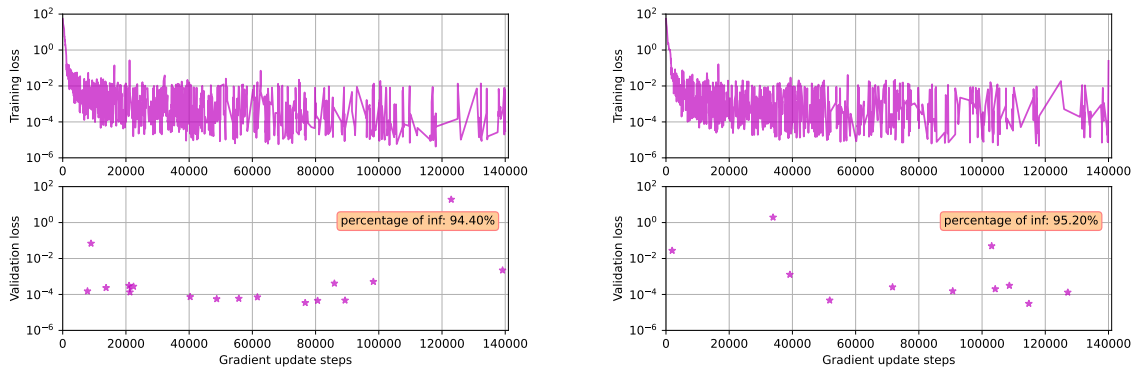
5.4.1 Training of NN solvers with different numerical settings

In this section, we illustrate some details exhibited during the training and validation processes. Specifically, we discuss the effects of normalizing the residuals and setting the hand-tuned α parameter in Section 5.4.1.1 and Section 5.4.1.2, respectively.

5.4.1.1 Benefit of the normalized residual

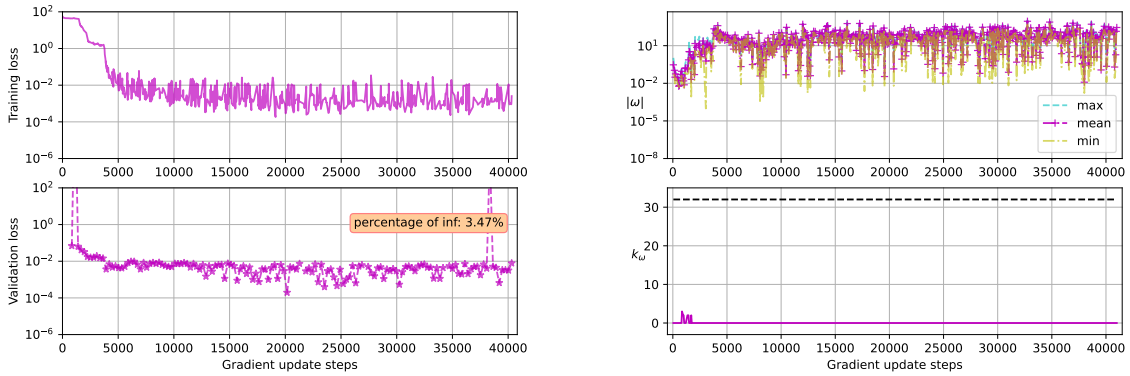
We first use the same setting as in [105]. In the top graph of Figure 5.2 (a) we plot the value of the loss function at each update of the gradient, that is at each iteration of the training. It can be seen that the training loss tends to converge but exhibits large variation. In the bottom graph of Figure 5.2 (a), we display the validation loss, that is the evaluation of the loss function on a data set not seen in the training. The validation loss is periodically computed to estimate the capability of the network to predict valuable outputs. It can be seen that many values (94.4%) of the validation loss are missing, because they were either infinite (i.e., Inf) or not a number (i.e., Nan), which are useless results. We think that this is due to the fact that the residual might have a large norm at the beginning of the training phase so that some of its entries become out of range in 32-bit calculation. Consequently, instead of using a constant scaling factor α to scale the residual vector in the input of the NN, we used the normalized residual to train the network. The results for R-NN are displayed in Figure 5.2 (b), where unfortunately no benefit is made.

We performed the same experiments for training the MRR-NN, that uses a relaxation parameter. The results are reported in Figure 5.3. Several comments can be made for the results displayed in the left graphs. The first one is related to the robustness of the MRR-NN in the validation phase, where most of the validation loss are finite numbers. The major consequence is that the convergence of the training process is faster and less computing demanding. The second observation is that the normalization of the input residual enables to reduce by two orders of magnitude the validation loss at the possible price of larger variations of the training loss. On the right part of this figure, we display the values of the relaxation parameter along the training iterations, where we used an upper and lower thresholds to keep the y-range bounded. These thresholds were set to be 10^{-8} and 10^{8} and the values out of this range were set to one. We plot below the graphs reporting the modulus of the ω, the number of values out of range along the iterations that is denoted k_ω. The number of out of range values is bounded by the batch size that is 32 in our case. It can be seen that only a few values are out of range and only early in the training.

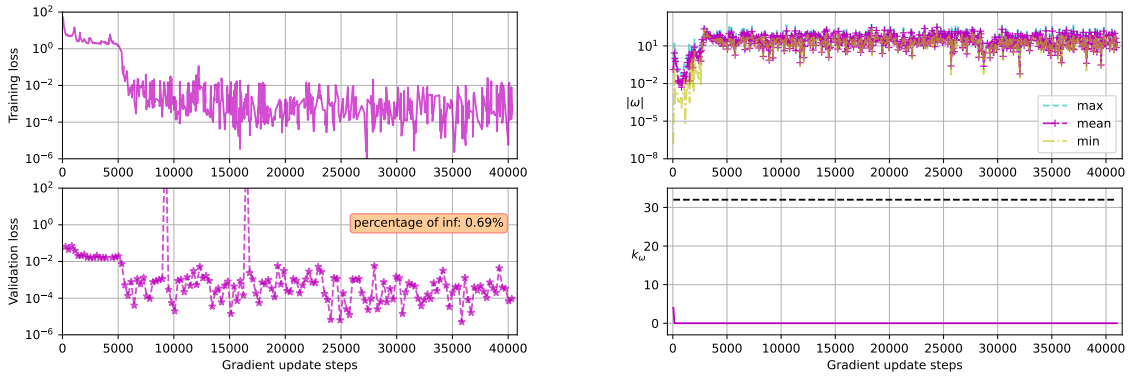


(*a*) non-normalized residual as input (*b*) normalized residual as input

Figure 5.2 – Training and validation loss of R-NN with $\alpha = 1000$ and non-normalized/normalized residual as input.



(*a*) : non-normalized residual as input

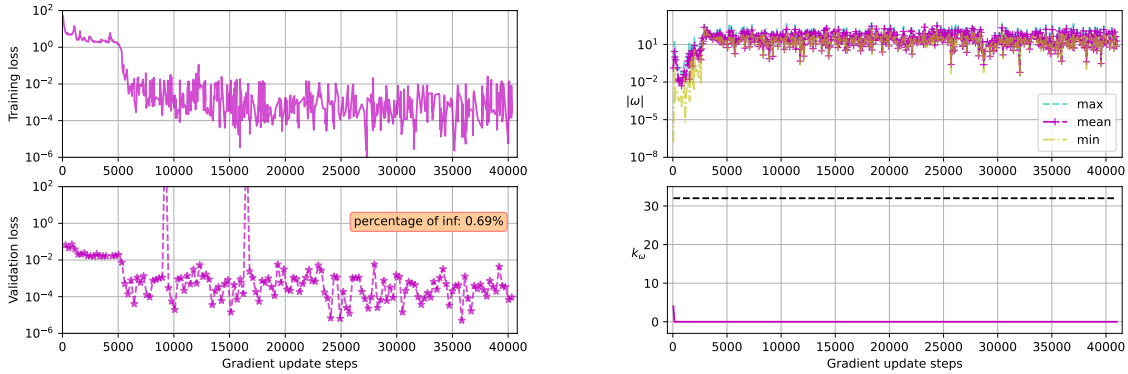


(*b*) : normalized residual as input

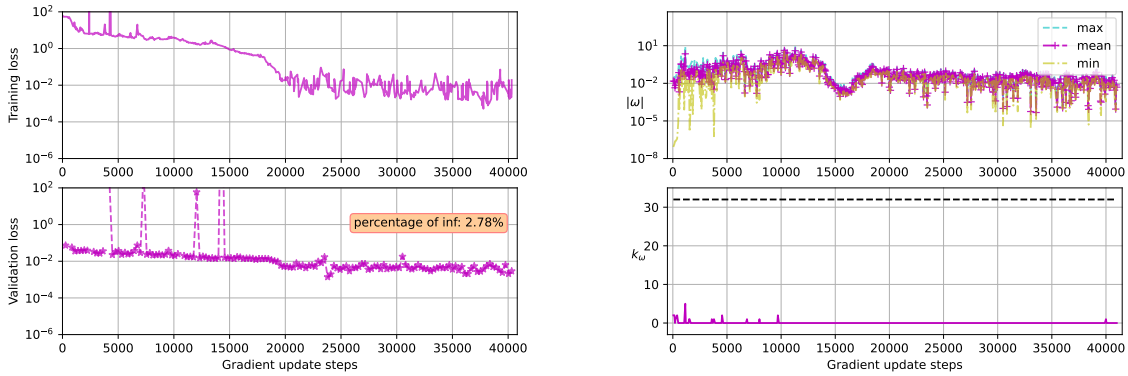
Figure 5.3 – Training of MRR-NN with normalized/non-normalized residual and $\alpha = 1000$. (Left): Training and validation loss along the training iterations (Right): Value of $|\omega|$ and k_ω along the training iterations. The black dashed line shows the batch size = 32 that is an upper bound of k_ω.

5.4.1.2 Influence of the α-scaling parameter

In Figure 5.4, we vary the values of the hand-tuned α parameter for the MRR-NN training. It can be seen that a regular normalized residual (i.e., $\alpha = 1$) leads to a less oscillatory convergence of both the training and validation loss, but with a worse asymptotic value for the training and validation loss.



(a) : $\alpha = 1000$ for scaling normalized residual as input



(b) : $\alpha = 1$ for scaling normalized residual as input

Figure 5.4 – Training of MRR-NN with different α-scaling parameter with normalized residual. (Left): Training and validation loss along the training iteration. (Right): Value of $|\omega|$ and k_ω along the training iterations. The black dashed line shows the batch size = 32 that is an upper bound of k_ω.

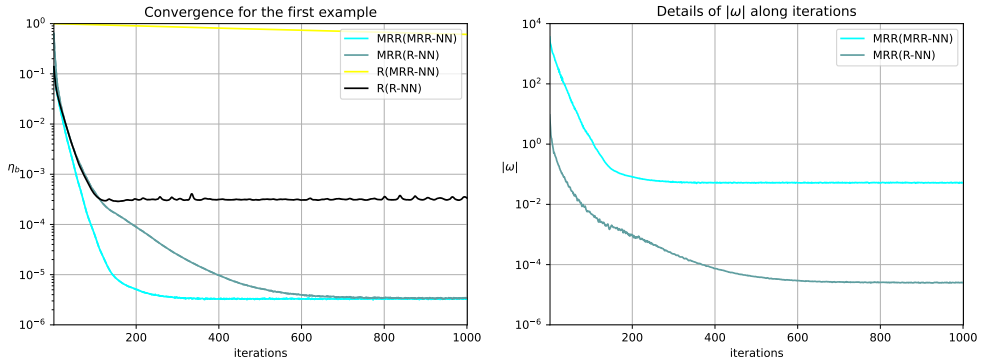
5.4.2 Testing the trained NNs solvers

In this section we investigate the numerical efficiency of the trained NNs used either as fixed point iteration or as flexible preconditioner for FFOM and FGMRES. The numerical efficiency is considered both in terms of convergence speed and attainable accuracy with respect to the backward error η_b defined in formula (2.34). For the sake of comparison with the results presented in [105] (i.e., with a non-normalized residual as input of the R-NN and $\alpha = 1000$), we use in this section the hyper parameters (i.e., the inner trained weights and biases of networks) downloaded from their GitHub project [1].

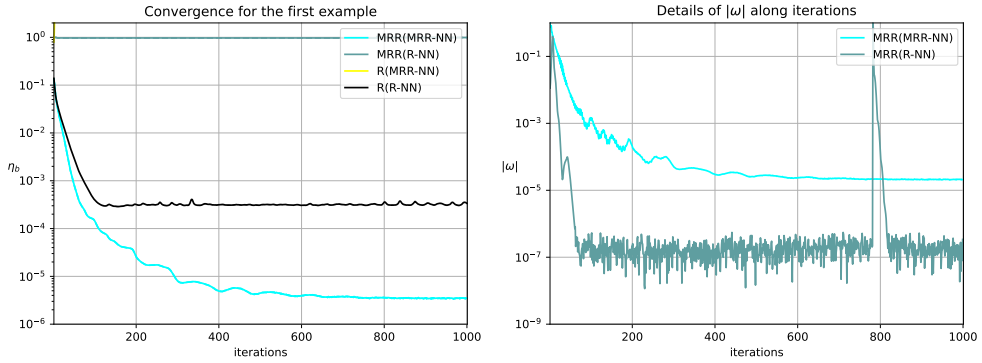
5.4.2.1 The trained Minimum Residual Richardson solver: MRR(MRR-NN)

We display in the left graphs of Figure 5.5 the convergence history of the various fixed point iterations that can be designed using the trained R-NN and MRR-NN. In order to illustrate the possible impact of the α scaling for MRR-NN we consider two different values of α. The first observation is that α does not have a significant impact on the convergence of MRR(MRR-NN), it is slightly faster for $\alpha = 1000$ but the attainable accuracy is the same and close to the round-off unit of the working precision. The second observation is that MRR(MRR-NN) outperforms R(R-NN) both in terms of convergence speed and in terms of attainable accuracy (possibly more important). Lastly, these graphs illustrate the lack of robustness of the R(MRR-NN) and MRR(R-NN). On both examples, R(MRR-NN) does not converge, while MRR(R-NN) converges in one case but not in the other one. For this reason, we do not discuss of these two combinations further.

On the right part of this figure, we plot the modulus of the relaxation parameter ω that goes quickly towards small values for the MRR(R-NN) solver.



(a): $\alpha = 1000$ for MRR-NN



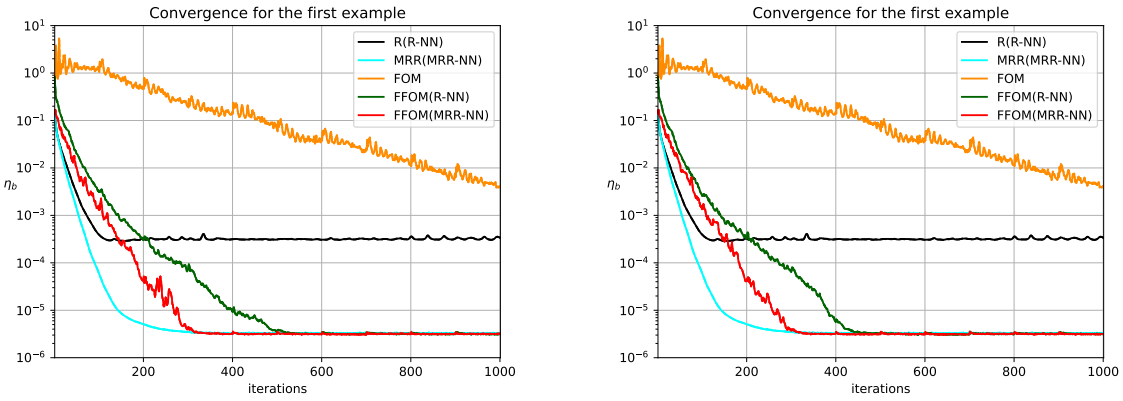
(b): $\alpha = 1$ for MRR-NN

Figure 5.5 – Performance of the R and MRR solver with nonlinear correction function R-NN and MRR for two values of α for MRR-NN. (Left) convergence history. (Right) modulus of the relaxation parameter. All calculations are performed in 32-bit arithmetic.

In the following section we investigate the main features of the various possible usages of the trained NN as a flexible preconditioner for FFOM and FGMRES.

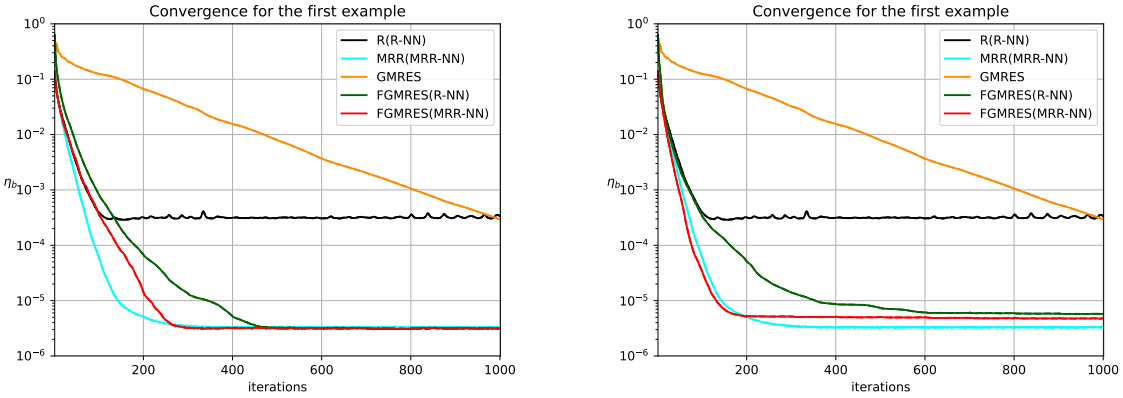
5.4.2.2 NNs as preconditioner: different strategies and mixed arithmetic

In Figure 5.6 we display the convergence history for the two preconditioning policies used for FFOM and FGMRES for the solution of the first example of the testing data set. For those experiments all the calculations are performed in 32-bit arithmetic (fp32). The first observation is that the two flexible strategies behave the same for FFOM and FGMRES, and the MRR-NN leads to a faster convergence compared to R-NN; as it was already the case in a fixed point iteration context. It can be observed that all methods but R(R-NN) and FOM/GMRES have an attainable accuracy close to the unit round-off of fp32. As a general comment regarding FFOM versus FGMRES, we can observe that the classical monotonic decrease trend of the scaled residual of FGMRES and the more erratic one of FFOM.



(*a*): Strategy 1 (*b*): Strategy 2

Experiments with FFOM

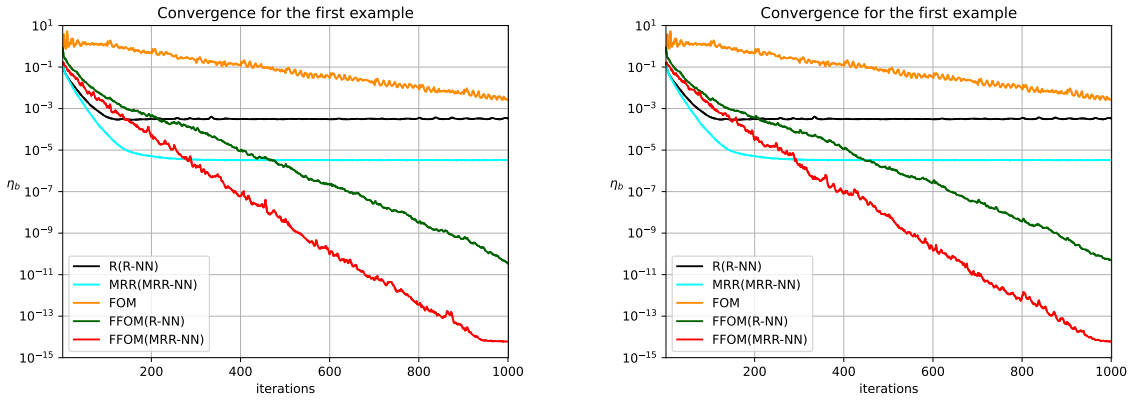


(*c*): Strategy 1 (*d*): Strategy 2

Experiments with FGMRES

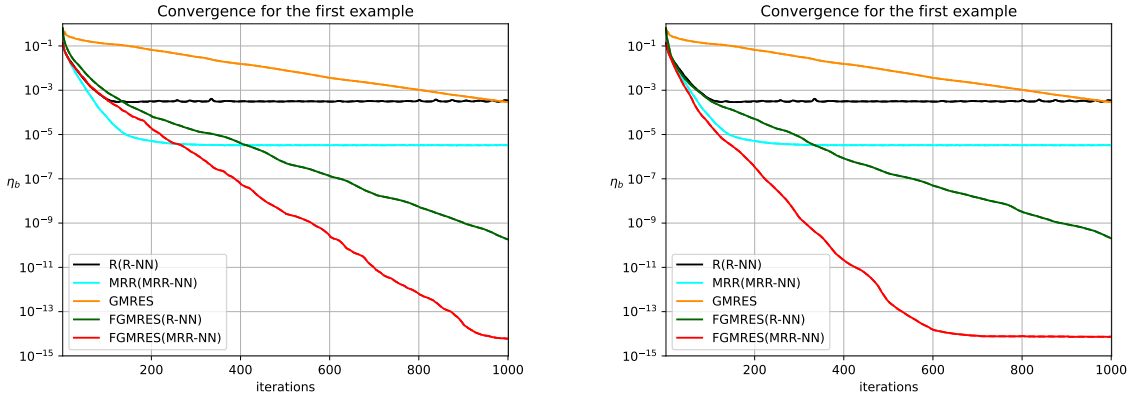
Figure 5.6 – The different variants of NN as a flexible preconditioner for FFOM and FGMRES.

In Figure 5.7 we report on similar experiments as in Figure 5.6 but using mixed arithmetic calculation. It means that the NNs have been trained in fp32 arithmetic and used in calculations where all computations but the NNs inference are performed in float-64 (fp64). For the FFOM and FGMRES, it does fit their capabilities as the additional nonlinear truncation caused by casting fp64 into fp32 is encompassed in the flexible



(*a*): Strategy 1 (*b*): Strategy 2

Experiments with FFOM



(*c*): Strategy 1 (*d*): Strategy 2

Experiments with FGMRES

Figure 5.7 – The different variants of NN as a flexible preconditioner for FFOM and FGMRES in mixed arithmetic calculation.

preconditioner framework. For the fixed point iteration, the scheme reduces to classical iterative refinement in mixed arithmetic calculation where the residual is computed in fp64 and the correction is computed in fp32.

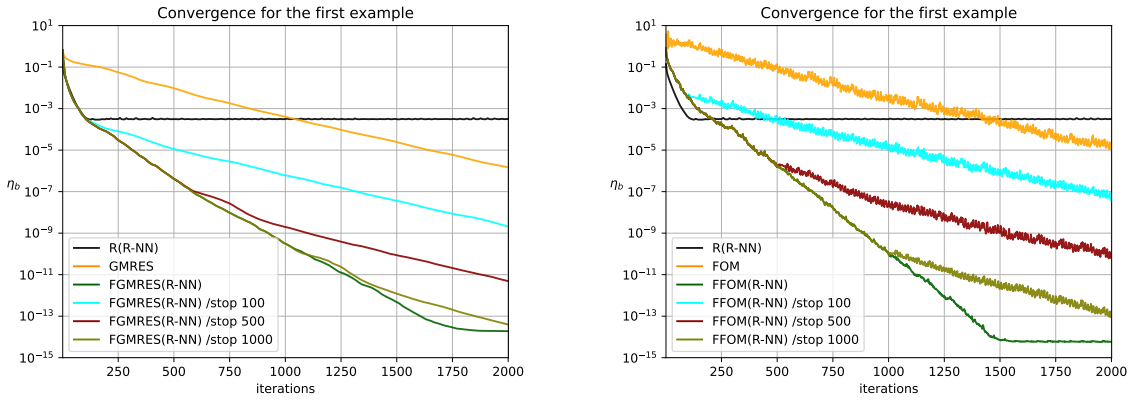
The first observation is that both FFOM and FGMRES are able to compute solutions with an accuracy close to the unit round-off error of fp64. The iterative refinement methods MRR(MRR-NN) and R(R-NN) fail to improve the solution accuracy to the fp64 unit round-off error. The second observation, regarding the different strategies, there is no real trend as Strategy 2 seems to be more effective for FGMRES than for FFOM, while Strategy 1 seems to act similarly on the two flexible subspace solvers. Refer to Table 5.1 for details of the final attainable accuracy and implementation time of solving the first example by 1000 iterations of the involved solvers in fp32 and mixed arithmetic calculation.

To clearly show that the faster convergence rate observed for both FFOM and FGMRES is due to the use of the trained NN as flexible preconditioner, we display in Figure 5.8 the convergence history of the two types of flexible preconditioning solvers and

# example	Method	η_b(fp32 / fp32&fp64)	$\#time(s)$
1st	R(R-NN)	3.36e-04 / 3.36e-04	7.37 / 9.68
	MRR(MRR-NN)	3.27e-06 / 3.27e-06	10.83 / 10.08
	GMRES	2.95e-04 / 2.88e-04	18.95 / 23.69
	FGMRES(R-NN)	6.05e-06 / 2.16e-10	24.72 / 30.75
	FGMRES(MRR-NN)	4.73e-06 / 7.34e-15	25.55 / 29.50

Table 5.1 – Numerical results in terms of η_b and $\#time(s)$ for solving the first example by 1000 iterations of NN solvers and FGMRES(R-NN) and FGMRES(MRR-NN) with Strategy 2 (notice that the R(R-NN) and MRR(MRR-NN) stagnate around 200 iterations in different attainable accuracy).

switch to unpreconditioned FOM (GMRES) at different prescribed iteration numbers. It can be seen that as soon as we switch off the preconditioner the convergence rate becomes colinear to one of the regular FOM (respectively GMRES).



(a): FGMRES with Strategy 2 (b): FFOM with Strategy 1

Figure 5.8 – Convergence histories of FGMRES and FFOM with R-NN as preconditioner; no preconditioning is applied after a given number of iterations.

To better see the impact of the mixed arithmetic calculation, we display in the same graph in Figure 5.9 all the convergence history (in terms of backward error $\eta_{A,b}$ defined in Equation (2.37)) of the FGMRES solve. It can be seen that for a given NN inference, the curves associated with fp32 and mixed arithmetic calculation overlap up to a value close to the unit round-off of fp32. After this point the fp32 curves stagnate while the mixed arithmetic ones go down to values close to the fp64 unit round-off.

In order to further illustrate the numerical behavior of the novel schemes for the solution of the first example on the one hand, and the hardest one (i.e., the 865th one) to solve on the other hand, in the testing data set, we report on their performance in Figure 5.10. Finally, we report on the performance of the novel solvers for the solution of the first 32 examples of the testing data set as well as the average and median convergence history for the full testing data set in Figure 5.11. The qualitative observations made previously on the first example do extend to the average on the full data set.

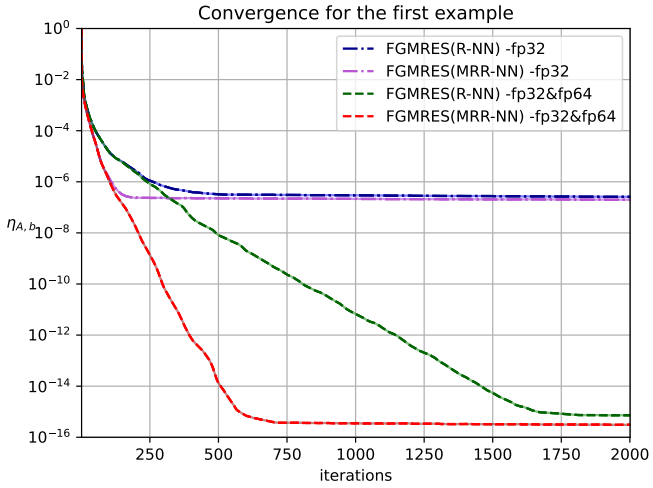


Figure 5.9 – Convergence history of FGMRES(R-NN) and FGMRES(MRR-NN) using Strategy 2 in fp32 and mixed arithmetic calculation for the solution of the first example.

To conclude the experimental part with a bit of colored science we display in Figure 5.12 the iterated solution viewed as the wavefield of a Helmholtz problem, but will not make any particular scientific comments on them.

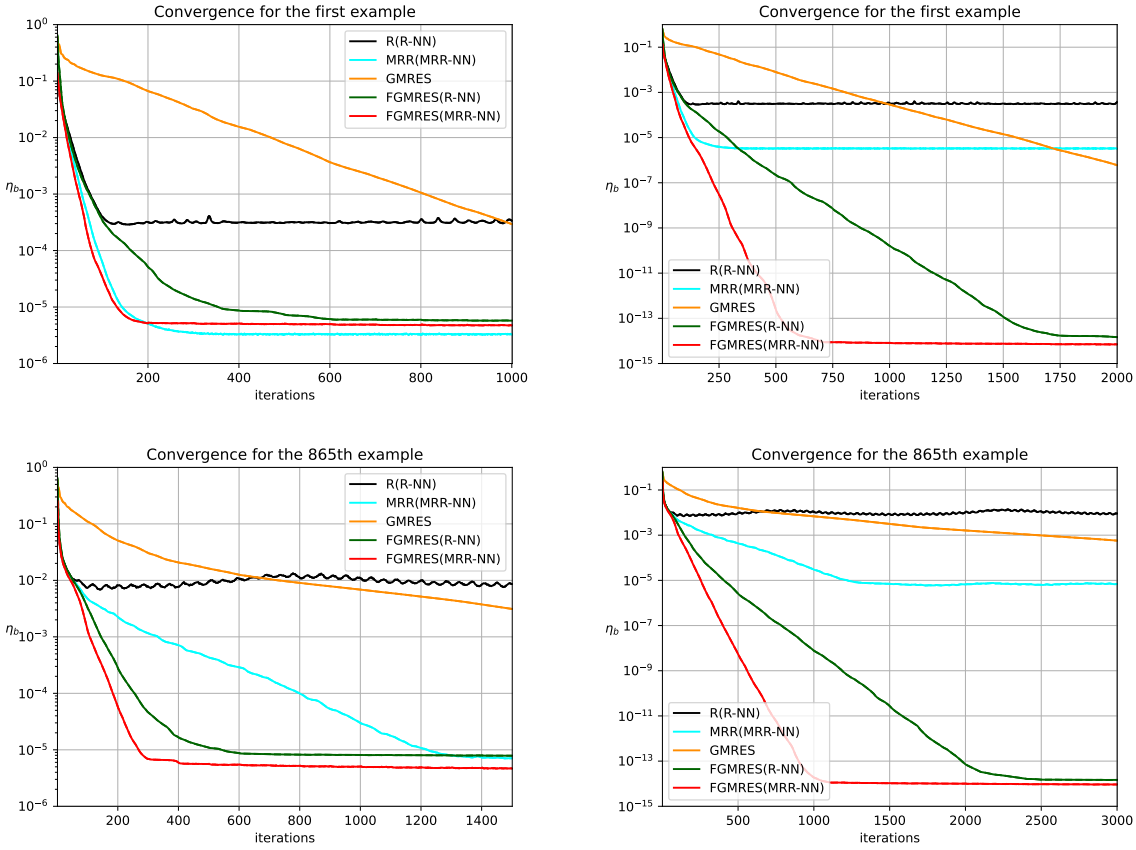
5.4.2.3 Network generalizability

To mimic the processes discussed in [105, Section 3.3], in this section, we focus on investigating the generalization capabilities of the proposed MRR-NN inference. Specifically, the two trained NNs inferences are used to solving the following three examples that out of the training, validation and test data set. We shortly describe the information of these three examples as below, and we refer the reader to [105, Section 3.3] and its references for more details.

- **The rectangle example**: A rectangular region with a background sound speed of 2 m/s on the 96×96 grid points rather than the idealized skull examples with circular or elliptic shape with a background sound speed of 1 m/s on the 96×96 grid points used during the training phases. This is used for testing the ability of the trained networks to deal with the region in different geometric shape.

- **The large example**: A large speed of sound distribution with 480×480 grid points was created by patching together 24 distributions from the test data set having 96×96 grid points. This is used for testing the ability of the trained networks to generalize to much larger domains.

- **The skull example**: A large 512×512 speed of sound distribution generated from a transverse CT slice from an adult skull. The source distribution was defined as a focused transducer represented by a 1D arc (recall that the network has only seen single-point sources in a fixed position during training). The transducer aperture diameter and radius of curvature were set to 60 mm, and the source frequency to 490 kHz. This is used for testing whether the networks still converge to a satisfactory

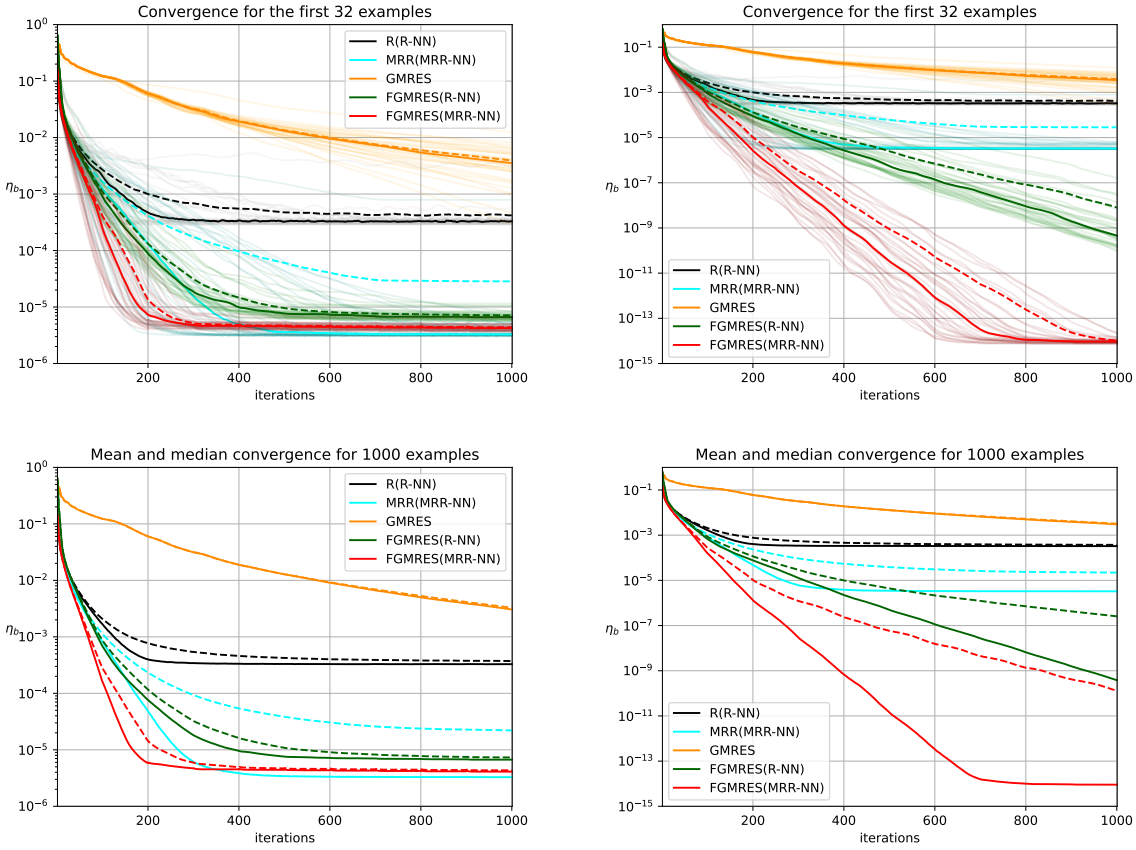


fp32 calculation mixed arithmetic calculation

Figure 5.10 – Convergence histories of solving the first and 865th example picked from the testing data set using fp32 and mixed arithmetic calculation. FGMRES nonlinear preconditioner is based on Strategy 2. Upper two: The first example. Lower two: The 865th example (the hardest one to be solved in the testing data test).

solution with an arbitrary background sound speed, source distribution, different geometric shape and different domain sizes.

For the rectangle example, Figure 5.13-5.14 show the reference solution calculated using the involved solvers with the trained NNs, the details of $|\omega|$ in the MRR variants (i.e., MRR(MRR-NN) and MRR(R-NN)), and the evolution of wavefield for displaying a satisfactory solution. We note that this example exhibits a moderate positive benefit of these MRR-NN variants (i.e., MRR(MRR-NN) and FGMRES(MRR-NN)) in reaching better attainable accuracy that was very effective in previous Section 5.4.2.1-5.4.2.2. The MRR variants (especially MRR(R-NN)) reach and stagnate at a similar attainable accuracy as the R(R-NN) one. So do the final accuracy of FGMRES(R-NN) and FMRES(MRR-NN). This information becomes much more clear when analyzing the results of the large example and the skull one as described in Figure 5.15-5.16 and Figure 5.17-5.18, respectively. That is the MRR-NN variants shown in these figures finally stagnate at a larger attainable accuracy than the R(R-NN) one, while the MRR(R-NN) and FGMRES(R-NN) exhibit its obvious advantages. This observation illustrates that the generalization capabilities of the MRR-NN inference is not as good as the R-NN one.



fp32 calculation mixed arithmetic calculation

Figure 5.11 – Convergence histories for the novel solvers in fp32 and mixed arithmetic calculation. FGMRES nonlinear preconditioner is based on Strategy 2. Upper two: The first 32 examples (the dashed and solid lines correspond to the mean and the median backward error η_b of all the 32 examples). Lower two: Present the mean with dashed line and the median with solid line of the backward errors for the full testing data set (1000 examples).

Note that here we only consider comparing the optimal R-NN inference from [1] to the MRR-NN one with the hyper parameters saved in epoch 69 rather than the ones saved in epoch 255 adopted in Section 5.4.2.1-5.4.2.2. Since the MRR-NN variants with hyper parameters saved in epoch 225 fail to solve the large and skull examples, even though they can solve the rectangle example (refer to Appendix A.4 for the numerical details). This illustrates that the sub-performance of MRR-NN in network generalizability may be caused by the over-training. While further efforts are definitely required to have a more satisfying interpretation of this. On the other hand, the trade-off between better attainable accuracy and good network generalizability is a well-known challenge in the machine learning region, which is currently beyond the scope of this manuscript. However, it is worthy to going on further efforts for balancing this two points.

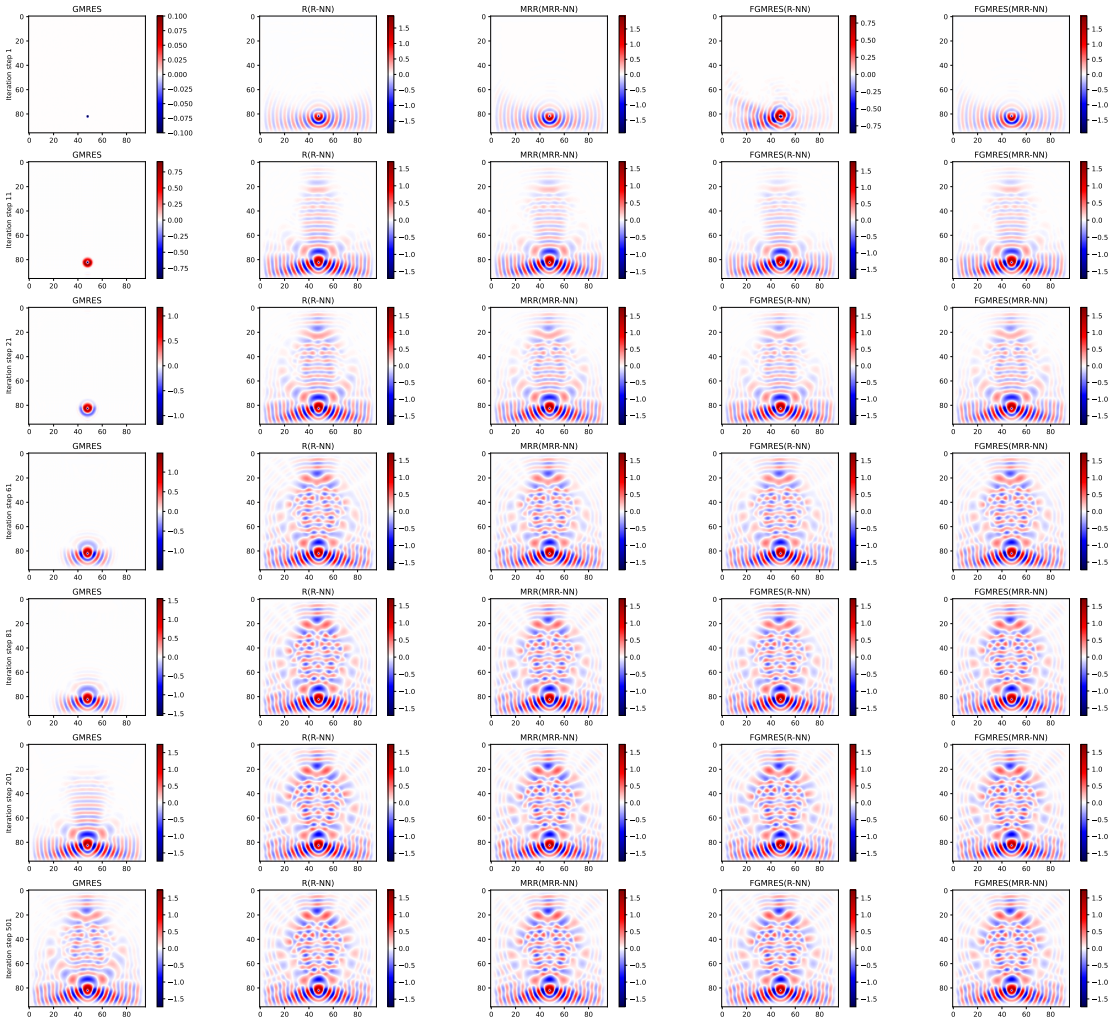
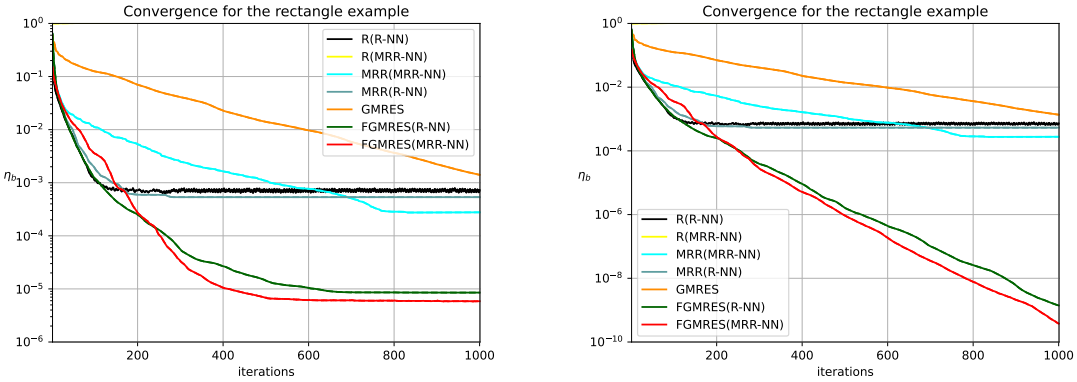


Figure 5.12 – Simulation process of the 865th example (the hardest one) (the real part of wavefield/solution is shown) by the involved five solvers with fp32.

5.5 Concluding remarks

The work presented in this chapter is a follow-up of the study described in [105], and we greatly benefited from the codes and data set made available by the authors on their GitHub project [1]. We propose two main improvements related to both the training of the NN and the use of the NN inference once the training has been performed. Regarding the training, we introduce an optimal relaxation parameter in the fixed iteration scheme that enables us to minimize the residual norm along the direction provided by the NN. The use of this relaxation parameter makes the training more robust as it reduces the number of Inf/Nan in the validation phase. We conjecture that the relaxation parameter avoids exploring some "random" regions of the NN parameter space that are irrelevant for learning the solution of Hemholtz linear systems.

The second contribution is the hybridization of machine learning and classical numerical linear algebra. In that context we propose to use the trained NN, not the fixed point iteration scheme exploited for the training, as a nonlinear preconditioner for subspace solvers such as flexible FOM (FFOM) or flexible GMRES (FGMRES). The idea



(a): fp32 calculation (b): mixed arithmetic calculation

Figure 5.13 – Convergence histories of NNs with $\alpha = 1000$ and FGMRES with Strategy 2 for the rectangle example (note that the MRR-NN with hyper parameters saved in epoch 69).

is to view the output of the NN as the inverse of a Helmholtz matrix applied to an input vector. We propose two practical strategies to implement this idea in order to design flexible preconditioners for FFOM or FGMRES. Through numerical experiments we show that those strategies have similar merits and none of the two outperforms the other.

We believe that our proposed hybrid solvers, which combine machine learning techniques and classical numerical linear algebra, allow us to benefit from two worlds:

1. improvement of the training phase with more robustness and faster convergence,
2. ability of reaching higher attainable accuracy of the NN solver,
3. fast calculation of the preconditioner application through the NN inference,
4. arbitrary precision of the computed solution thanks to the flexible subspace iterations.

This work is very much a preliminary attempt to hybridizes machine learning and scientific computing for the solution of linear algebra problem.

From Section 5.4.2.3, a straightforward future work is to explore the balance of ensuring the better attainable accuracy and the good generalizability of network. Other possible future directions could be tuning and testing other neural network architectures (like the Fourier neural operators [41]), the definition of loss function based on a practical physical background, the (automatically) defining of the learning rate that decays exponentially with the index of epoch, the tuning and testing of other hyper-parameters (like choosing of other optimizer, or the number of hidden layers and nodes per hidden layer), the possible ways to addressing the vanishing gradient issue [9, 37, 43], the pre-processing of the data sets, and devising other hybridization strategies and studying their effects, etc.

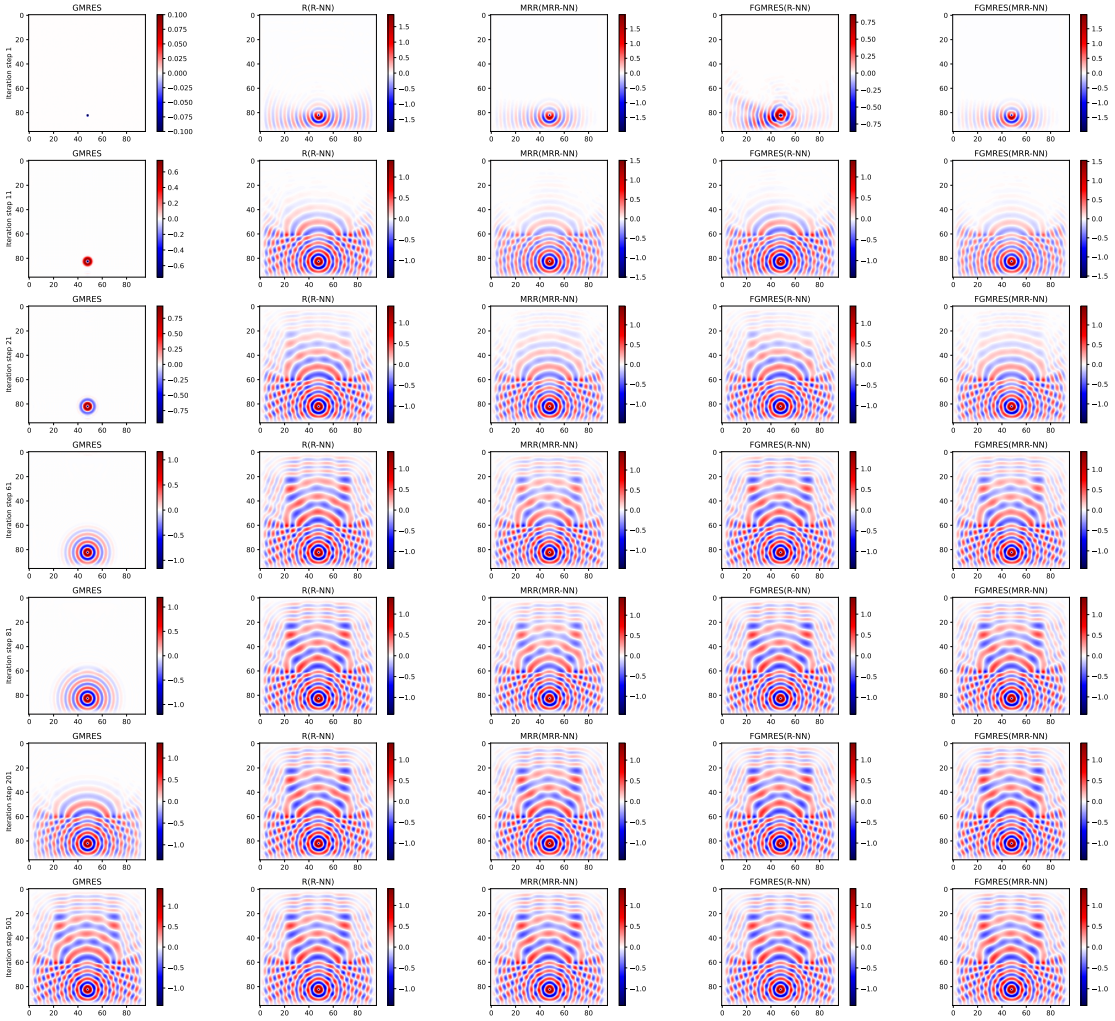
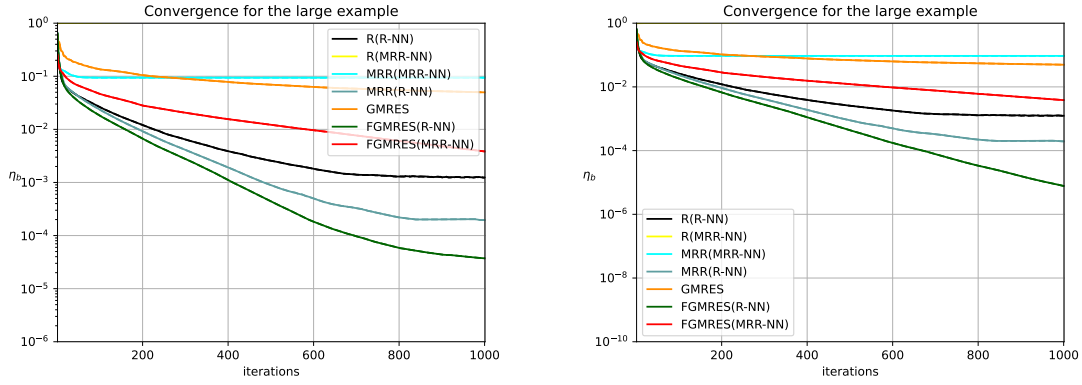


Figure 5.14 – Simulation process of the rectangle example by the involved five solvers with fp32.



(a): fp32 calculation (b): mixed arithmetic calculation

Figure 5.15 – Convergence histories of NNs with $\alpha = 1000$ and FGMRES with Strategy 2 for the large example.

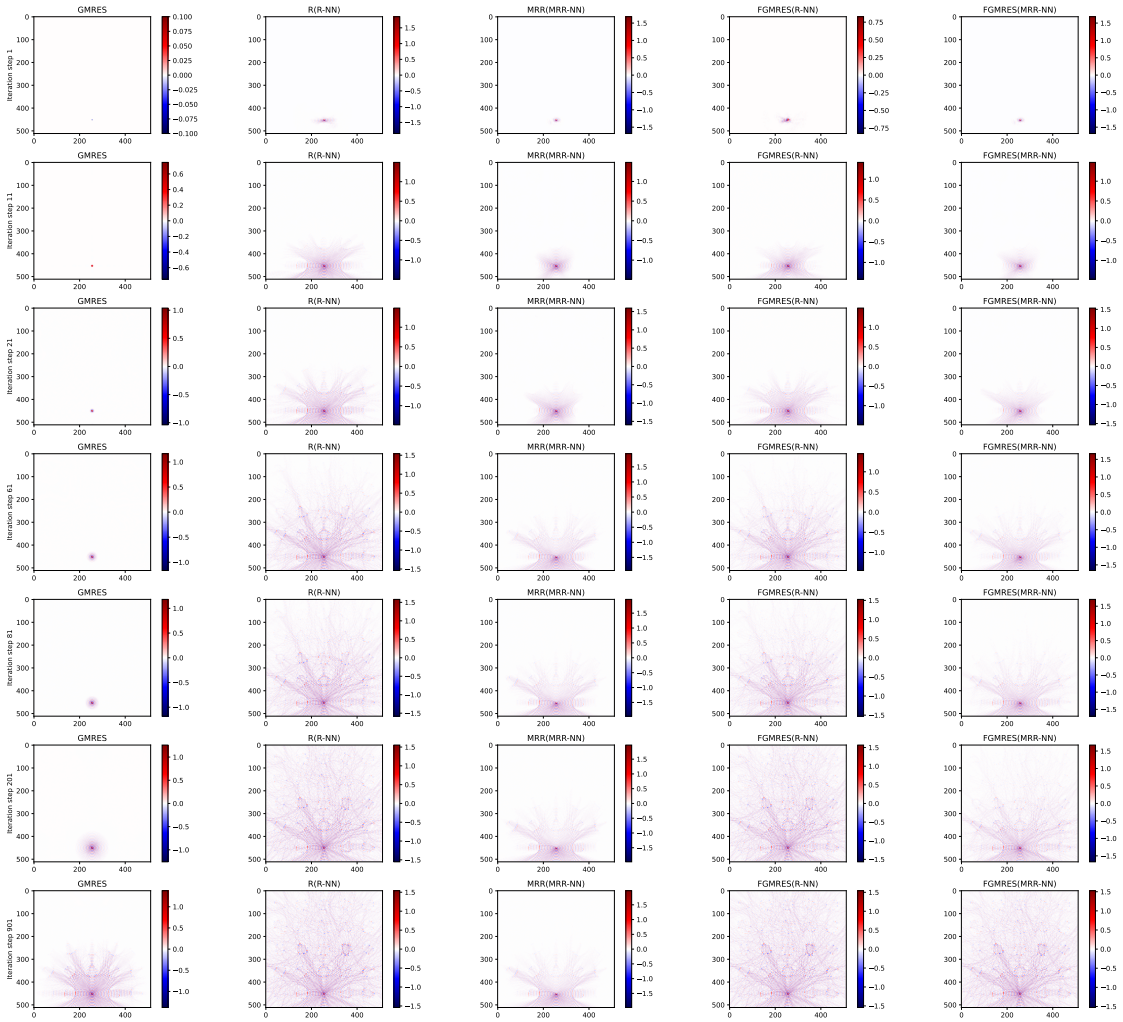
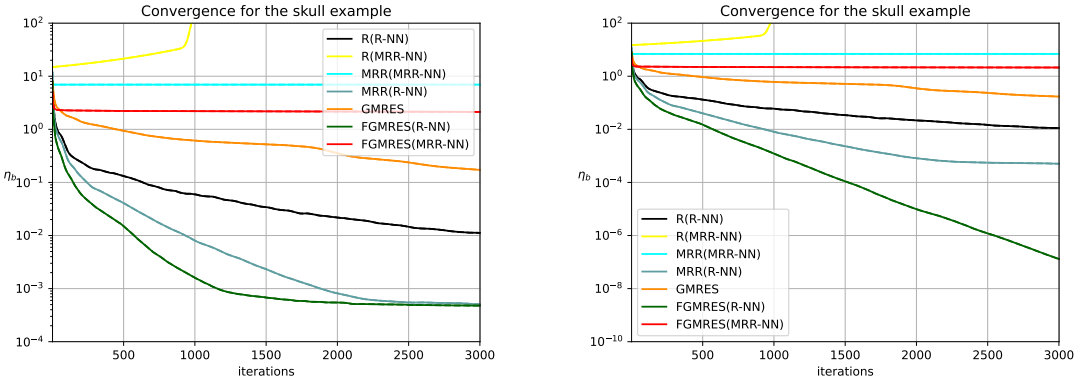


Figure 5.16 – Simulation process of the large example by the involved five solvers with fp32.



(a): fp32 calculation (b): mixed arithmetic calculation

Figure 5.17 – Convergence histories of NNs with $\alpha = 1000$ and FGMRES with Strategy 2 for the skull example.

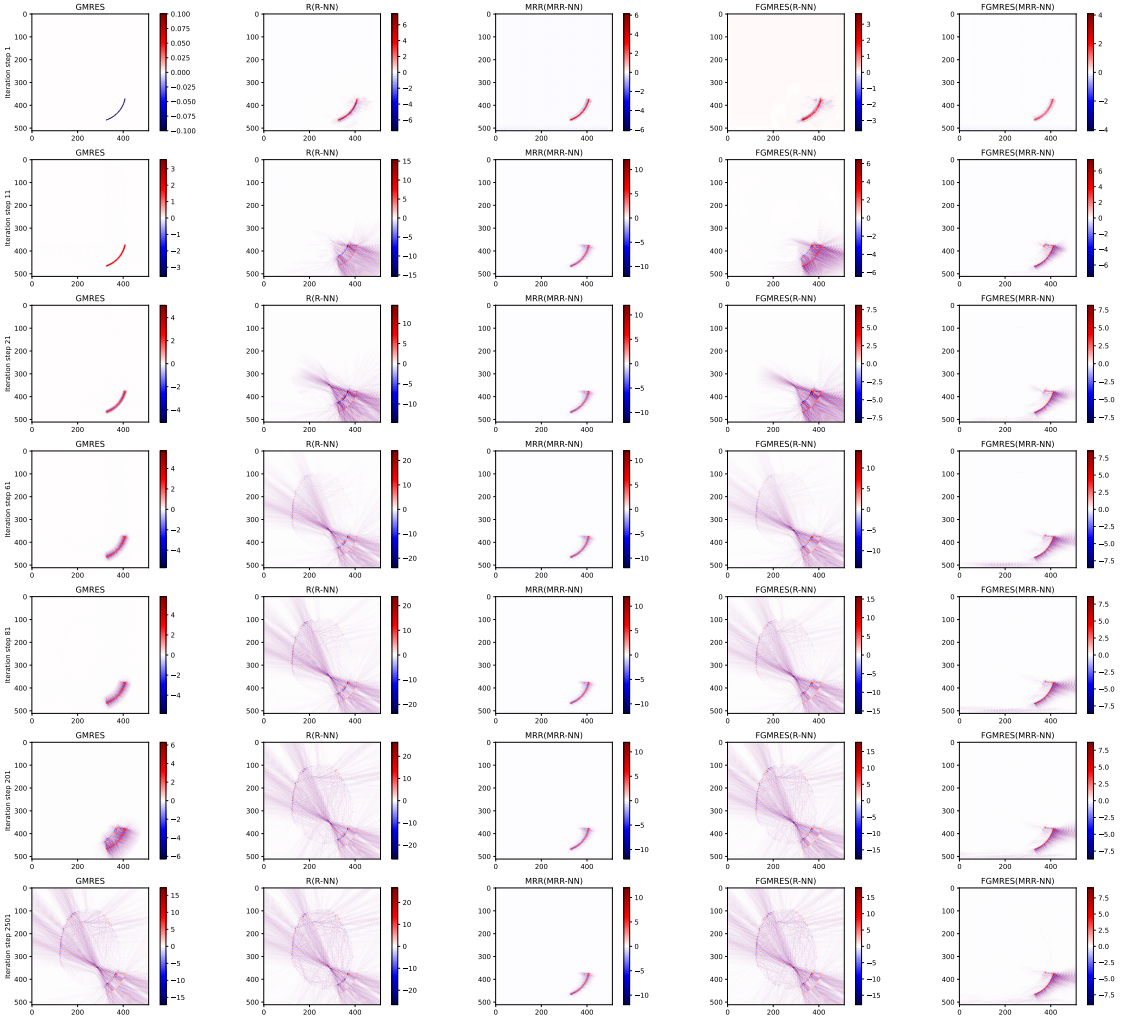


Figure 5.18 – Simulation process of the skull example by the involved five solvers with fp32.

Part III

Further perspectives

Develop new potentials for block Krylov subspace methods. Part I develops some new variants of the block Krylov subspace methods for linear systems with multiple right-hand sides. For Chapter 2 the block minimum residual norm techniques with long term recurrence, following ideas from [78], future research could theoretically establish that this class of subspace augmentation algorithms is backward stable. For the short term recurrence algorithms such as BCR and BCG described in Chapter 3-4, some residual gaps exist sometimes preventing the convergence of the true backward error, i.e., based on the true residual norm, while the one based on the iterated residual norm indicates convergence. A possible remedy could be to extend the rounding error analysis from [120] to block case to estimate the block residual gap and possibly design corresponding replacement techniques [16, 25]. Besides, when considering the subspace recycling or deflation strategy, an interesting further direction is to find some general information to guide the choosing of some hand-tuning parameters, like the maximal length of one refining cycle, the dimension of deflation space, and the number of right-hand sides.

Integration in the solver stack of the Inria team. The block minimum residual norm techniques for unsymmetric case described in Chapter 2 have been integrated into one of our team's solver stack namely *Fabulous* [1] thanks to a close collaboration with an engineer (Matthieu Simonin) in charge of this library. Fabulous is a fully featured C++ library that implements various block Krylov solvers for the solutions of linear systems. This integration enables the use of IB-BGCRO-DR in various applications such as for QCD computation in the framework of the H2020 project PRACE-6IP. To fully assess the computational performances of the block solvers presented in Chapter 3 and 4, it would be worth conducting such an engineering effort to foster the transfer of knowledge from numerical linear algebra to large scale applications.

Develop new potentials for optimal scientific machine learning models. Part II discusses two main improvements in hybridizing machine learning techniques and numerical linear algebra solvers for linear system with single right-hand side. A straightforward future direction from the details described in Section 5.4.2.3 of Part II is to explore the balance of ensuring the better attainable accuracy and the good generalizability of network. Further related work is to try to *shed some lights* into the black boxes when developing scientific machine learning (SciML) solvers [3, 65, 118], like the choosing of deep neural network architecture for specific physical problem (like the Fourier neural operators [41]), the definition of loss function based on practical physical background (like the recent work [129]), the (automatically) defining of learning rate that decays exponentially with the index of epoch, the tuning and testing of other hyper-parameters (like the choose of other optimizer, the number of hidden layers and nodes in per hidden layer), the possible ways to addressing the vanishing gradient issue [9, 37, 43], the pre-processing of the the data sets, and devising other hybridization strategies and studying their effects, etc. To have a clear understanding of these options, I envision to try other machine learning models for variety of applications. I want to discover the composition-structure-property relationships for varying computing problems and SciML solvers, and then use them to figure out what is the optimal SciML models (in terms of attainable accuracy, convergence speed, lightweight neural network architecture, and computational cost, etc.) for target simulation. With faster descriptors, I hope to create a categorized theory system for different SciML computing problems.

[1] https://gitlab.inria.fr/solverstack/fabulous/

Devise sparse machine learning models. Based on the efforts described in Part II as well as its straightforward further directions illustrated in the previous paragraph, another problem I am interested in is to devise a way to reduce the training costs. One way for this could be pre-processing data sets or introducing mathematical and physical information when devising neural network solver with target solution scheme. Another way could be switching the focus to developing new large-scale sparse machine learning models using knowledge from a multidisciplinary framework, such as graph theory and computer science, which in turn can also help the machine learning regions for other applications out of the SciML scopes, but which can be pretty challenging as well.

Bibliography

[1] A Helmholtz equation solver using unsupervised learning. `https://github.com/ucl-bug/helmnet`.

[2] J. Ackmann, P. D. Düben, T. N. Palmer, and P. K. Smolarkiewicz. Machine-Learned Preconditioners for Linear Solvers in Geophysical Fluid Flows. *arXiv*, 2020. https://doi.org/10.48550/arXiv.2010.02866.

[3] B. Adcock and N. Dexter. The Gap between Theory and Practice in Function Approximation with Deep Neural Networks. *SIAM J. Math. Data. Sci.*, 3(2):624–655, 2021. https://doi.org/10.1137/20M131309X.

[4] J. Adler and O. Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017. https://doi.org/10.1088/1361-6420/aa9581.

[5] E. Agullo, F. Cappello, S. Di, L. Giraud, X. Liang, and N. Schenkels. Exploring variable accuracy storage through lossy compression techniques in numerical linear algebra: A first application to flexible GMRES. Research report, Inria, Bordeaux Sud-Ouest, 2020. https://hal.inria.fr/hal-02572910v2/.

[6] E. Agullo, L. Giraud, and Y.-F. Jing. Block GMRES Method with Inexact Breakdowns and Deflated Restarting. *SIAM J. Matrix Anal. Appl.*, 35(4):1625–1651, 2014. https://doi.org/10.1137/140961912.

[7] N. Bell, L. N. Olson, and J. Schroder. PyAMG: Algebraic Multigrid Solvers in Python. *Journal of Open Source Software*, 7(72):4142, 2022. https://doi.org/10.21105/joss.04142.

[8] M. Benedetti, J. Realpe-Gómez, and A. Perdomo-Ortiz. Quantum-assisted Helmholtz machines: A quantum-classical deep learning framework for industrial datasets in near-term devices. *Quantum Sci. Technol.*, 3(3):034007, 2018. https://doi.org/10.1088/2058-9565/aabd98.

[9] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. https://doi.org/10.1109/72.279181.

[10] S. Bhowmick, V. Eijkhout, Y. Freund, E. Fuentes, and D. Keyes. Application of Machine Learning in Selecting Sparse Linear Solvers. *J. High Perf. Comput.*, 2006. https://icl.utk.edu/files/publications/2006/icl-utk-287-2006.pdf.

[11] F. E. Bock, S. Keller, N. Huber, and B. Klusemann. Hybrid Modelling by Machine Learning Corrections of Analytical Model Predictions towards High-Fidelity Simulation Solutions. *Materials*, 14(8), 2021. https://doi.org/10.3390/ma14081883.

[12] M. A. Botchev, V. Grimm, and M. Hochbruck. Residual, Restarting, and Richardson Iteration for the Matrix Exponential. *SIAM J. Sci. Comput.*, 35(3):A1376–A1397, 2013. https://doi.org/10.1137/110820191.

[13] N. Boullé, C. J. Earls, and A. Townsend. Data-driven discovery of Green's functions with human-understandable deep learning. *arXiv*, 2021. https://doi.org/10.48550/arXiv.2105.00266.

[14] S.-Z. Cai, Z.-C. Wang, L. Lu, T. A. Zaki, and G. E. Karniadakis. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *arXiv*, 2020. https://doi.org/10.48550/arXiv.2009.12935.

[15] H. Calandra, S. Gratton, R. Lago, X. Vasseur, and L. M. Carvalho. A modified block flexible GMRES method with deflation at each iteration for the solution of non-Hermitian linear systems with multiple right-hand sides. *SIAM J. Sci. Comput.*, 35:S345–S367, 2013. https://doi.org/10.1137/120883037.

[16] E. Carson and J. Demmel. A Residual Replacement Strategy for Improving the Maximum Attainable Accuracy of s-Step Krylov Subspace Methods. *SIAM J. Matrix Anal. Appl.*, 35(1):22–43, 2014.

[17] E. Carson, K. Lund, M. Rozloznik, and S. Thomas. Block Gram-Schmidt algorithms and their stability properties. *Linear Algebra Appl.*, 638:150–195, 2022. https://doi.org/10.1016/j.laa.2021.12.017.

[18] L. M. Carvalho, S. Gratton, R. Lago, and X. Vasseur. A Flexible Generalized Conjugate Residual Method with Inner Orthogonalization and Deflated Restarting. Technical Report TR/PA/10/10, CERFACS, Toulouse, France, 2010. https://hal.archives-ouvertes.fr/hal-00650239v2.

[19] L. M. Carvalho, S. Gratton, R. Lago, and X. Vasseur. A Flexible Generalized Conjugate Residual Method with Inner Orthogonalization and Deflated Restarting. *SIAM J. Matrix Anal. Appl.*, 32(4):1212–1235, 2011. https://doi.org/10.1137/100786253.

[20] J. Chen. A deflated version of the block conjugate gradient algorithm with an application to Gaussian process maximum likelihood estimation. 2011. Preprint ANL/MCS-P1927-0811, Argonne National Laboratory, Argonne, https://jiechenjiechen.github.io/pub/dbpcg.pdf.

[21] J.-Z. Chen, J. K. Patel, and R. Vasques. Solver Recommendation For Transport Problems in Slabs Using Machine Learning. *arXiv*, 2019. https://doi.org/10.48550/arXiv.1906.08259.

[22] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. *arXiv*, 2018. https://doi.org/10.48550/arXiv.1806.07366.

[23] L. Cheng, E. A. Illarramendi, G. Bogopolsky, M. Bauerheim, and B. Cuenot. Using neural networks to solve the 2D Poisson equation for electric field computation in plasma fluid simulations. *arXiv*, 2021. https://doi.org/10.48550/arXiv.2109.13076.

[24] M. Clare, O. Jamil, and C. Morcrette. A computationally efficient neural network for predicting weather forecast probabilities. *arXiv*, abs/2103.14430, 2021.

[25] S. Cools, E. F. Yetkin, E. Agullo, L. Giraud, and W. Vanroose. Analyzing the Effect of Local Rounding Error Propagation on the Maximal Attainable Accuracy of the Pipelined Conjugate Gradient Method. *SIAM J. Matrix Anal. Appl.*, 39(1):426–450, 2018.

[26] O. Coulaud, L. Giraud, P. Ramet, and X. Vasseur. Deflation and augmentation techniques in krylov subspace methods for the solution of linear systems. Research Report 8265, Inria, Bordeaux Sud-Ouest, 2013. https://hal.inria.fr/hal-00803225.

[27] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38:1, 2011.

[28] E. de Sturler. Nested Krylov methods based on GCR. *J. Comput. Appl. Math.*, 67(1):15–41, 1996. https://doi.org/10.1016/0377-0427(94)00123-5.

[29] E. de Sturler. Truncation Strategies for Optimal Krylov Subspace Methods. *SIAM J. Numer. Anal.*, 36(3):864–889, 1999. https://doi.org/10.1137/S0036142997315950.

[30] P. Dueben, U. Modigliani, A. Geer, S. Siemen, F. Pappenberger, P. Bauer, A. Brown, M. Palkovič, B. Raoult, N. Wedi, and V. Baousis. Machine learning at ECMWF: A roadmap for the next 10 years. *ECMWF*, 2021. http://dx.doi.org/10.21957/ge7ckgm.

[31] V. Dwivedi and B. Srinivasan. A Normal Equation-Based Extreme Learning Machine for Solving Linear Partial Differential Equation. *ASME. J. Comput. Inf. Sci. Eng.*, 22(1):014502, 2022. https://doi.org/10.1115/1.4051530.

[32] J. Erhel and F. Guyomarc'H. An Augmented Conjugate Gradient Method for Solving Consecutive Symmetric Positive Definite Linear Systems. *SIAM J. Matrix Anal. Appl.*, 21(4):1279–1299, 2000. https://doi.org/10.1137/S0895479897330194.

[33] Y-W. Fan and L-X. Ying. Solving Inverse Wave Scattering with Deep Learning. *arXiv*, 2019. https://doi.org/10.48550/arXiv.1911.13202.

[34] A. Frommer, K. Lund, and D. B. Szyld. Block Krylov Subspace Methods for Functions of Matrices II: Modified Block FOM. *SIAM J. Matrix Anal. Appl.*, 41(2):804–837, 2020. https://doi.org/10.1137/19M1255847.

[35] L. Giraud, S. Gratton, X. Pinel, and X. Vasseur. Flexible GMRES with Deflated Restarting. *SIAM J. Sci. Comput.*, 32:1858–1878, 2010. https://doi.org/10.1137/080741847.

[36] L. Giraud, D. Ruiz, and A. Touhami. A Comparative Study of Iterative Solvers Exploiting Spectral Information for SPD Systems. *SIAM J. Sci. Comput.*, 27(5):1760–1786, 2006. https://doi.org/10.1137/040608301.

[37] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256. PMLR, 2010. https://proceedings.mlr.press/v9/glorot10a.html.

[38] N. Gottschling, V. Antun, B. Adcock, and A. C. Hansen. The troublesome kernel: why deep learning for inverse problems is typically unstable. *arXiv*, abs/2001.01258, 2020. https://arxiv.org/pdf/2001.01258.pdf.

[39] M. Götz and H. Anzt. Machine Learning-Aided Numerical Linear Algebra: Convolutional Neural Networks for the Efficient Preconditioner Generation. In *2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (scalA)*. IEEE, 2018. https://doi.org/10.1109/ScalA.2018.00010.

[40] X.-X. Guo, W. Li, and F. Iorio. Convolutional Neural Networks for Steady Flow Approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. https://doi.org/10.1145/2939672.2939738.

[41] J. K. Gupta and J. Brandstetter. Towards Multi-spatiotemporal-scale Generalized PDE Modeling. *arXiv*, 2022. https://doi.org/10.48550/arXiv.2209.15616.

[42] M. H. Gutknecht. Block Krylov space methods for linear systems with multiple right-hand sides: An introduction. In I. S. Duff, A. H. Siddiqi, and O. Christensen, editors, *Modern Mathematical Models, Methods and Algorithms for Real World Systems*, pages 420–447. Anamaya Publishers, New Delhi, India, 2006.

[43] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *arXiv*, 2015. https://doi.org/10.48550/arXiv.1512.03385.

[44] M. Hestenes. The conjugate gradient method for solving linear systems. *PSAM*, VI, Numerical Analysis:83–102, 1956.

[45] M. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[46] N. Higham and T. Mary. Mixed precision algorithms in numerical linear algebra. *Acta Numerica*, 31:347–414, 2022. https://doi.org/10.1017/S0962492922000022.

[47] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Process. Mag.*, 29(6):82–97, 2012. https://doi.org/10.1109/MSP.2012.2205597.

[48] J.-T. Hsieh, S.-J. Zhao, S. Eismann, L. Mirabella, and S. Ermon. Learning neural PDE solvers with Convergence Guarantees. *arXiv*, 2019. https://doi.org/10.48550/arXiv.1906.01200.

[49] E. A. Illarramendi, A. Alguacil, M. Bauerheim, A. Misdariis, B. Cuenot, and E. Benazera. Towards an hybrid computational strategy based on Deep Learning for incompressible flows. In *AIAA AVIATION 2020 FORUM*. American Institute of Aeronautics and Astronautics, 2020. https://doi.org/10.2514/6.2020-3058.

[50] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proc. R. Soc. A*, 476(2239):20200334, 2020. https://doi.org/10.1098/rspa.2020.0334.

[51] E. Jessup, P. Motter, B. Norris, and K. Sood. Performance-Based Numerical Solver Selection in the Lighthouse Framework. *SIAM J. Sci. Comput.*, 38(5):S750–S771, 2016. https://doi.org/10.1137/15M1028406.

[52] H. Ji and Y.-H. Li. A breakdown-free block conjugate gradient method. *BIT*, 57:379–403, 2017. http://dx.doi.org/10.1007/s10543-016-0631-z.

[53] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. https://doi.org/10.48550/arXiv.1412.6980.

[54] A. 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*, volume 96, pages 143–154. Birkhäuser Basel, 1991. https://doi.org/10.1007/978-3-0348-6332-2_11.

[55] A. V. Knyazev. Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method. *SIAM J. Sci. Comput.*, 23(2):517–541, 2006. https://doi.org/10.1137/S1064827500366124.

[56] A. V. Knyazev. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. https://doi.org/10.1145/3065386.

[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25, 26th Annual Conf*, volume 1, pages 1097–1105. Neural Information Processing Systems Foundation Inc, Neural Information Processing Systems 2012, Lake Tahoe, 2012.

[58] T. Kurth, S. Treichler, J. Romero, M. Mudigonda, N. Luehr, E. Phillips, A. Mahesh, M. Matheson, J. Deslippe, M. Fatica, Prabhat, and M. Houston. Exascale Deep Learning for Climate Analytics. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 649–660. IEEE, 2018. https://doi.org/10.1109/SC.2018.00054.

[59] I. Lagaris, A. Likas, and D. G. Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–9, 2000. http://dx.doi.org/10.1109/72.870037.

[60] M. Levonyak, C. Pacher, and W. N. Gansterer. Scalable Resilience Against Node Failures for Communication-Hiding Preconditioned Conjugate Gradient and Conjugate Residual Methods. In *Proceedings of the 2020 SIAM*

Conference on Parallel Processing for Scientific Computing, pages 81–92, 2020. https://doi.org/10.1137/1.9781611976137.

[61] H. Li, J. Schwab, S. Antholzer, and M. Haltmeier. NETT: solving inverse problems with deep neural networks. *Inverse Problems*, 36(6):065005, 2020. https://doi.org/10.1088/1361-6420/ab6d57.

[62] Z.-C. Long, Y.-P. Lu, X.-Z. Ma, and B. Dong. PDE-Net: Learning PDEs from Data. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3208–3216. PMLR, 2018. https://proceedings.mlr.press/v80/long18a.html.

[63] L. Lu, P.-Z. Jin, and G. E. Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv*, 2019. https://doi.org/10.48550/arXiv.1910.03193.

[64] L. Lu, X.-H. Meng, Z.-P. Mao, and G. E. Karniadakis. DeepXDE: A Deep Learning Library for Solving Differential Equations. *SIAM Rev.*, 63(1):208–228, 2021. https://doi.org/10.1137/19M1274067.

[65] L. Lu, R. Pestourie, W.-J. Yao, Z.-C. Wang, F. Verdugo, and S. G. Johnson. Physics-Informed Neural Networks with Hard Constraints for Inverse Design. *SIAM J. Sci. Comput.*, 43(6):B1105–B1132, 2021. https://doi.org/10.1137/21M1397908.

[66] D. Luenberger. The Conjugate Residual Method for Constrained Minimization Problems. *SIAM J. Numer. Anal.*, 7(3):390–398, 1970. https://hal.archives-ouvertes.fr/hal-01350543.

[67] K. Luna, K. Klymko, and J. P. Blaschke. Accelerating GMRES with Deep Learning in Real-Time. *arXiv*, 2021. https://doi.org/10.48550/arXiv.2103.10975.

[68] S. Markidis. The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers? *arXiv*, 2021. https://doi.org/10.48550/arXiv.2103.09655.

[69] R. A. McCallum. Hidden state and reinforcement learning with instance-based state identification. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 26(3):464–473, 1996. https://doi.org/10.1109/3477.499796.

[70] S. Mishra and R. Molinaro. Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating PDEs. *arXiv*, 2020. https://doi.org/10.48550/arXiv.2006.16144.

[71] R. B. Morgan. A Restarted GMRES Method Augmented with Eigenvectors. *SIAM J. Matrix Anal. Appl.*, 16(4):1154–1171, 1995. https://doi.org/10.1137/S0895479893253975.

[72] R. B. Morgan. GMRES with Deflated Restarting. *SIAM J. Sci. Comput.*, 24(1):20–37, 2002. https://doi.org/10.1137/S1064827599364659.

[73] R. B. Morgan. Restarted block-GMRES with deflation of eigenvalues. *Appl. Numer. Math.*, 54(2):222–236, 2005. https://doi.org/10.1016/j.apnum.2004.09.028.

[74] M. A. Nabian, R. J. Gladstone, and H. Meidan. Efficient training of physics-informed neural networks via importance sampling. *Comput. Aided. Civ. Inf.*, 36:962–977, 2021. https://doi.org/10.1111/mice.12685.

[75] B. Nathan, A. Frank, B. Timo, H. Aric, K. Yannis, N. Habib, P. Manish, P. Abani, S. James, W. Stefan, W. Karen, and L. Steven. Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence. Technical report, U.S. DOE Office of Science, Washington, DC, 2019. https://doi.org/10.2172/1478744.

[76] M. Neyra-Nesterenko and B. Adcock. Stable, accurate and efficient deep neural networks for inverse problems with analysis-sparse models. *arXiv*, 2022. https://doi.org/10.48550/arXiv.2203.00804.

[77] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra Appl.*, 29:293–322, 1980. https://doi.org/10.1016/0024-3795(80)90247-5.

[78] C. C. Paige, M. Rozložník, and Z. Strakoš. Modified Gram-Schmidt (MGS), least squares, and backward stability of MGS-GMRES. *SIAM J. Matrix Anal. Appl.*, 28(1):264–284, 2006. https://doi.org/10.1137/050630416.

[79] M. L. Parks. The Iterative Solution of a Sequence of Linear Systems Arising from Nonlinear Finite Element Analysis. Ph.D. dissertation UIUCDCS-R-2005-2497, University of Illinois at Urbana-Champaign, 2005.

[80] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti. Recycling Krylov Subspaces for Sequences of Linear Systems. *SIAM J. Sci. Comput.*, 28(5):1651–1674, 2006. https://doi.org/10.1137/040607277.

[81] M. L. Parks, K. M. Soodhalter, and D. B. Szyld. A block Recycled GMRES method with investigations into aspects of solver performance. *arXiv*, 2016. https://doi.org/10.48550/arXiv.1604.01713.

[82] L. F. Pavarino. Preconditioned conjugate residual methods for mixed spectral discretizations of elasticity and Stokes problems. *Comput. Methods Appl. Mech. Engrg.*, 146(1-2):19–30, 1997. https://doi.org/10.1016/S0045-7825(96)01224-8.

[83] L. Peairs and T.-Y. Chen. Using reinforcement learning to vary the m in GMRES(m). *Procedia Computer Science*, 4:2257–2266, 2011. International Conference on Computational Science, ICCS 2011, https://doi.org/10.1016/j.procs.2011.04.246.

[84] A. Perdomo-Ortiz, M. Benedetti, J. Realpe-Gómez, and R. Biswas. Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. *Quantum Sci. Technol.*, 3(3):030502, 2018. https://doi.org/10.1088/2058-9565/aab859.

[85] M. Raissi. Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations. *J. Mach. Learn. Res.*, 19(25):1–24, 2018. http://jmlr.org/papers/v19/18-046.html.

[86] L. G. Ramos, R. Kehl, and R. Nabben. Projections, Deflation, and Multigrid for Nonsymmetric Matrices. *SIAM J. Matrix Anal. Appl.*, 41(1):83–105, 2020. https://doi.org/10.1137/18M1180268.

[87] G. Rizzuti, A. Siahkoohi, E. Chow, and F. J. Herrmann. Learned Iterative Solvers for the Helmholtz Equation. In *81st EAGE Conference and Exhibition 2019*. European Association of Geoscientists & Engineers, 2019. https://doi.org/10.3997/2214-4609.201901542.

[88] M. Robbé and M. Sadkane. Exact and inexact breakdowns in the block GMRES method. *Linear Algebra Appl.*, 419(1):265–285, 2006. https://doi.org/10.1016/j.laa.2006.04.018.

[89] O. Ronneberger, F. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Lecture Notes in Computer Science*. Springer International Publishing, 2015. https://doi.org/10.1007/978-3-319-24574-4_28.

[90] D. Ruda, S. Turek, D. Ribbrock, and P. Zajac. Very fast finite element Poisson solvers on lower precision accelerator hardware: A proof of concept study for Nvidia Tesla V100. *Internat. J. High Performance Computing Appl.*, 36(4):1–16, 2022. http://dx.doi.org/10.1177/10943420221084657.

[91] Y. Saad. A Flexible Inner-Outer Preconditioned GMRES Algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993. https://doi.org/10.1137/0914028.

[92] Y. Saad. *Iterative Methods for Sparse Linear Systems, 2nd ed.* SIAM, Philadelphia, 2003.

[93] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986. https://doi.org/10.1137/0907058.

[94] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'H. A Deflated Version of the Conjugate Gradient Algorithm. *SIAM J. Sci. Comput.*, 21(5):1909–1926, 2000. https://doi.org/10.1137/S1064829598339761.

[95] J. Sappl, L. Seiler, M. Harders, and W. Rauch. Deep Learning of Preconditioners for Conjugate Gradient Solvers in Urban Water Related Problems. *arXiv*, 2019. https://doi.org/10.48550/arXiv.1906.06925.

[96] Y. Shin, J. Darbon, and G. E. Karniadakis. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. *arXiv*, 2020. https://doi.org/10.48550/arXiv.2004.01806.

[97] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *J. Comp. Phys.*, 375(15):1339–1364, 2018. https://doi.org/10.1016/j.jcp.2018.08.029.

[98] T. Sogabe. Extensions of the Conjugate Residual Method. Ph.D. dissertation, University of Tokyo, Japan, 2003. http://www.ist.aichi-pu.ac.jp/person/sogabe/thesis.pdf.

[99] T. Sogabe, M. Sugihara, and S. L. Zhang. An extension of the conjugate residual method to nonsymmetric linear systems. *Journal of Computational and Applied Mathematics*, 226(1):103–113, 2009. https://doi.org/10.1016/j.cam.2008.05.018.

[100] M. Sonnewald, R. Lguensat, D. C. Jones, P. D. Dueben, J. Brajard, and V. Balaji. Bridging observations, theory and numerical simulation of the ocean using machine learning. *Environ. Res. Lett.*, 16(7), 2021. https://doi.org/10.1088/1748-9326/ac0eb0.

[101] K. Sood. Iterative Solver Selection Techniques for Sparse Linear Systems. Ph.D. dissertation, University of Oregon, 2019. https://www.cs.uoregon.edu/Reports/PHD-201905-Sood.pdf.

[102] K. Sood, B. Norris, and E. R. Jessup. Lighthouse: a taxonomy-based solver selection tool. In *Proceedings of the 2nd International Workshop on Software Engineering for Parallel Systems*. ACM, 2015. http://dx.doi.org/10.1145/2837476.2837485.

[103] K. M. Soodhalter. Krylov Subspace Methods with Fixed Memory Requirements: Nearly Hermitian Linear Systems and Subspace Recycling. Ph.D. dissertation, Temple University, Philadelphia, 2012.

[104] K. M. Soodhalter, E. de Sturler, and M. E. Kilmer. A survey of subspace recycling iterative methods. *Special Issue: Topical Issue Applied and Numerical Linear Algebra - Part II*, 43(4), 2020. https://doi.org/10.1002/gamm.202000016.

[105] A. Stanziola, S. R. Arridge, B. T. Cox, and B. E. Treeby. A Helmholtz equation solver using unsupervised learning: Application to transcranial ultrasound. *J. Comp. Phys.*, 441(2):110430, 2021. http://dx.doi.org/10.1016/j.jcp.2021.110430.

[106] A. Stathopoulos. Nearly Optimal Preconditioned Methods for Hermitian Eigenproblems under Limited Memory. Part I: Seeking One Eigenvalue. *SIAM J. Sci. Comput.*, 29(2):481–514, 2007. https://doi.org/10.1137/050631574.

[107] A. Stathopoulos and J. R. McCombs. Nearly Optimal Preconditioned Methods for Hermitian Eigenproblems Under Limited Memory. Part II: Seeking Many Eigenvalues. *SIAM J. Sci. Comput.*, 29(5):2162–2188, 2007. https://doi.org/10.1137/060661910.

[108] A. Stathopoulos and K. Orginos. Computing and Deflating Eigenvalues While Solving Multiple Right-Hand Side Linear Systems with an Application to Quantum Chromodynamics. *SIAM J. Sci. Comput.*, 32(1):439–462, 2010. https://doi.org/10.1137/080725532.

[109] A. Stathopoulos and Y. Saad. Restarting techniques for (Jacobi-)Davidson symmetric eigenvalue methods. *Electron. Trans. Numer. Anal.*, 7:163–181, 1998.

[110] D.-L. Sun, B. Carpentieri, T.-Z. Huang, and Y.-F. Jing. A spectrally preconditioned and initially deflated variant of the restarted block GMRES method for solving multiple right-hand sides linear systems. *Internat. J. Mech. Sci.*, 144:775–787, 2018. https://doi.org/10.1016/j.ijmecsci.2018.06.033.

[111] D.-L. Sun, T.-Z. Huang, B. Carpentieri, and Y.-F. Jing. Flexible and deflated variants of the block shifted GMRES method. *J. Comput. Appl. Math.*, 345:168–183, 2019. https://doi.org/10.1016/j.cam.2018.05.053.

[112] D.-L. Sun, T.-Z. Huang, B. Carpentieri, and Y.-F. Jing. A new shifted block GMRES method with inexact breakdowns for solving multi-shifted and multiple right-hand sides linear systems. *J. Sci. Comput.*, 78:746–769, 2019. https://doi.org/10.1007/s10915-018-0787-6.

[113] D.-L. Sun, T.-Z. Huang, Y.-F. Jing, and B. Carpentieri. A block GMRES method with deflated restarting for solving linear systems with multiple shifts and multiple right-hand sides. *Numer. Linear Algebra Appl.*, 25(5), 2018. https://doi.org/10.1002/nla.2148.

[114] A. Tajaddini, G. Wu, F. Saberi-Movahed, and N. Azizizadeh. Two New Variants of the Simpler Block GMRES Method with Vector Deflation and Eigenvalue Deflation for Multiple Linear Systems. *J. Sci. Comput.*, 86:9, 2021. https://doi.org/10.1007/s10915-020-01376-w.

[115] W. Tang, T. Shan, X. W. Dang, M. K. Li, F. Yang, S. H. Xu, and J. Wu. Study on a Poisson's equation solver based on deep learning technique. *IEEE Electrical Design of Advanced Packaging and Systems*, pages 1–3, 2017. https://doi.org/10.1109/EDAPS.2017.8277017.

[116] N. Tathawadekar, N. A. K. Doan, C. F. Silva, and N. Thuerey. Hybrid Neural Network PDE Solvers for Reacting Flows. *arXiv*, 2021. https://doi.org/10.48550/arXiv.2111.11185.

[117] Y. S. Teh, S. Ghosh, and K. Bhattacharya. Machine-learned prediction of the electronic fields in a crystal. *Mechanics of Materials*, 163:104070, 2021. https://doi.org/10.1016/j.mechmat.2021.104070.

[118] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. Accelerating Eulerian Fluid Simulation with Convolutional Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3424–3433. PMLR, 2017. https://proceedings.mlr.press/v70/tompson17a.html.

[119] R. Trivedi, L. Su, J. Lu, M. F. Schubert, and J. Vuckovic. Data-driven acceleration of photonic simulations. *Sci. Rep.*, 9:19728, 2019. https://doi.org/10.1038/s41598-019-56212-5.

[120] H. A. van der Vorst and Q. Ye. Residual Replacement Strategies for Krylov Subspace Iterative Methods for the Convergence of True Residuals. *SIAM J. Sci. Comput.*, 22(3):835–852, 2000. https://doi.org/10.1137/S1064827599353865.

[121] N. Venkovic, P. Mycek, L. Giraud, and O. Le Maître. Comparative study of harmonic and Rayleigh-Ritz procedures with applications to deflated conjugate gradients. Research report, CERFACS, 2020. https://hal.archives-ouvertes.fr/hal-02434043.

[122] N. Venkovic, P. Mycek, L. Giraud, and O. Le Maître. Recycling Krylov subspace strategies for sequences of sampled stochastic elliptic equations. Research report, Inria Bordeaux - Sud Ouest, 2021. https://hal.archives-ouvertes.fr/hal-03366966/document.

[123] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu. Towards Physics-informed Deep Learning for Turbulent Flow Prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp Data Mining*. ACM, 2020. https://doi.org/10.1145/3394486.3403198.

[124] Z.-M. Wang, T. Cui, and X.-S. Xiang. A Neural Network with Plane Wave Activation for Helmholtz Equation. *arXiv*, 2020. https://doi.org/10.48550/arXiv.2012.13870.

[125] K. S. Wu and H. Simon. Thick-Restart Lanczos Method for Large Symmetric Eigenvalue Problems. *SIAM J. Matrix Anal. Appl.*, 22(2):602–616, 2000. https://doi.org/10.1137/S0895479898334605.

[126] Y.-F. Xiang, Y.-F. Jing, and T.-Z. Huang. A New Projected Variant of the Deflated Block Conjugate Gradient Method. *J. Sci. Comput.*, 80:1116–1138, 2019. https://doi.org/10.1007/s10915-019-00969-4.

[127] F. Xue and H. C. Elman. Fast inexact subspace iteration for generalized eigenvalue problems with spectral transformation. *Linear Algebra Appl.*, 435(3):601–622, 2011. https://doi.org/10.1016/j.laa.2010.06.021.

[128] K. Yamada, T. Katagiri, H. Takizawa, M. Kazuo, M. Yokokawa, T. Nagai, and M. Ogino. Preconditioner Auto-Tuning Using Deep Learning for Sparse Iterative Algorithms. In *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE, 2018. https://doi.org/10.1109/CANDARW.2018.00055.

[129] J. Yu, L. Lu, X. Meng, and G. E. Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Comput. Methods Appl. Mech. Engrg.*, 393(1):114823, 2022. https://doi.org/10.1016/j.cma.2022.114823.

[130] J.-W. Zhuang, D. Kochkov, Y. Bar-Sinai, M. P. Brenner, and S. Hoyer. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Phys. Rev. Fluids*, 6:064605, 2021. https://doi.org/10.1103/PhysRevFluids.6.064605.

Appendix A

Appendix

A.1 Other two alternative strategies for approximating the eigen-information

Proposition 6. *(Strategy A [18]) At the end of a cycle of the IB-BFGCRO-DR algorithm, if the deflation space is built on the harmonic-Ritz vectors* $\widehat{\mathscr{Z}_m} g_i \in \text{span}(\widehat{\mathscr{Z}_m})$ *of* A *with respect to* $\widehat{\mathscr{Z}_m} = [U_k,\ \mathscr{Z}_m] \in \mathbb{C}^{n\times(k+n_m)}$*, the following hold:*

1. *The harmonic-Ritz pairs* $(\theta_i, \widehat{\mathscr{Z}_m} g_i)$ *for all restarts satisfy*

$$\underline{\mathscr{F}}_m^H \underline{\mathscr{F}}_m g_i = \theta_i \underline{\mathscr{F}}_m^H \widehat{\mathscr{V}}_{m+1}^H \widehat{\mathscr{Z}_m} g_i, \quad \text{for } 1 \le i \le k+n_m, \tag{A.1}$$

where

$$\widehat{\mathscr{V}}_{m+1}^H \widehat{\mathscr{Z}_m} = \begin{bmatrix} C_k^H U_k & C_k^H \mathscr{Z}_m \\ \mathscr{V}_m^H U_k & \mathscr{V}_m^H \mathscr{Z}_m \\ P_{m-1}^H U_k & P_{m-1}^H \mathscr{Z}_m \\ \widetilde{W}_m^H U_k & \widetilde{W}_m^H \mathscr{Z}_m \end{bmatrix} \in \mathbb{C}^{(k+n_m+p)\times(k+n_m)}. \tag{A.2}$$

2. *At restart, if* $G_k = [g_1, \ldots, g_k]$ *is associated with the* k *targeted eigenvalues, the matrices* U_k^{new} *and* C_k^{new} *to be used for the next cycle are updated by*

$$\begin{aligned} U_k^{new} &= \widehat{\mathscr{Z}_m} G_k R^{-1} = [U_k, \mathscr{Z}_m] G_k R^{-1}, \\ C_k^{new} &= \widehat{\mathscr{V}}_{m+1} Q = [C_k, \mathscr{V}_m, P_{m-1}, \widetilde{W}_m] Q, \end{aligned}$$

where Q *and* R *are the factors of the reduced* QR*-factorization of the tall and skinny matrix* $\underline{\mathscr{F}}_m G_k$*, ensuring* $A U_k^{new} = C_k^{new}$ *with* $(C_k^{new})^H C_k^{new} = I_k$*.*

3. *The residual at restart* $R_1^{new} = R_m^{old} = B - A X_1^{new}$ *with* $X_1^{new} = X_m^{old}$ *is orthogonal to* C_k^{new}*.*

Proof. The proofs basically relay on some matrix computations as shortly described below.

- According to Definition 1, each harmonic-Ritz pair $(\theta_i, \widehat{\mathscr{Z}_m} g_i)$ satisfies

$$\forall w \in \text{Range}(A\widehat{\mathscr{Z}_m}) \quad w^H \left(A\widehat{\mathscr{Z}_m} g_i - \theta_i\, \widehat{\mathscr{Z}_m} g_i\right) = 0,$$

which equivalently becomes

$$(A\widehat{\mathscr{Z}_m})^H\,(A\widehat{\mathscr{Z}_m}g_i - \theta_i\,\widehat{\mathscr{Z}_m}g_i) = 0.$$

Substituting (2.26) into the above leads to

$$\left(\widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m\right)^H\,\left(\widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m g_i - \theta_i\,\widehat{\mathscr{Z}_m}g_i\right) = 0. \tag{A.3}$$

Because $\widehat{\mathscr{V}}_{m+1} = [C_k, \mathscr{V}_m, [P_{m-1}, \widetilde{W}_m]]$ generated at the end of cycle is orthonormal, (A.3) becomes

$$\underline{\mathscr{F}}_m^H\,\underline{\mathscr{F}}_m g_i - \theta_i \underline{\mathscr{F}}_m^H \widehat{\mathscr{V}}_{m+1}^H \widehat{\mathscr{Z}_m} g_i = 0,$$

which gives the formulation (A.1).

- Let Q and R be the factors of the reduced QR-factorization of the tall and skinny matrix $\underline{\mathscr{F}}_m G_k$. Right multiplying G_k on both sides of (2.26) leads to $A\widehat{\mathscr{Z}_m}G_k = \widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m G_k = \widehat{\mathscr{V}}_{m+1}QR$, that is equivalent to $A\widehat{\mathscr{Z}_m}G_kR^{-1} = \widehat{\mathscr{V}}_{m+1}\underline{\mathscr{F}}_m G_kR^{-1} = \widehat{\mathscr{V}}_{m+1}Q$ concluding the proof as $\mathrm{span}(\widehat{\mathscr{Z}_m}G_kR^{-1}) = \mathrm{span}(\widehat{\mathscr{Z}_m}G_k)$, and $\widehat{\mathscr{V}}_{m+1}Q$ is the product of two matrices with orthonormal columns, so are its columns.

- The proof essentially follows the same arguments as the ones developed for Corollary 1, the details are omitted here.

□

Proposition 7. *(Strategy B [18]) At the end of a cycle of the IB-BFGCRO-DR algorithm, if the deflation space is built on the harmonic-Ritz vectors $\widehat{\mathscr{V}}_m g_i \in \mathrm{span}(\widehat{\mathscr{V}}_m)$ of $A\widehat{\mathscr{Z}_m}\widehat{\mathscr{V}}_m^H$ with respect to $\widehat{\mathscr{V}}_m = [C_k,\ \mathscr{V}_m] \in \mathbb{C}^{n\times(k+n_m)}$, the following hold:*

1. *The harmonic-Ritz pairs $(\theta_i, \widehat{\mathscr{V}}_m g_i)$ for all restarts satisfy*

$$\underline{\mathscr{F}}_m^H\underline{\mathscr{F}}_m g_i \;=\; \theta_i\,\mathscr{F}_m^H g_i \quad \textit{for } 1 \le i \le k+n_m,$$

2. *At restart, if $G_k = [g_{i_1}, \ldots, g_{i_k}]$ is associated with the k targeted eigenvalues, the matrices U_k^{new} and C_k^{new} to be used for the next cycle are defined by*

$$\begin{aligned} U_k^{new} &= \widehat{\mathscr{Z}_m}G_kR^{-1} = [U_k, \mathscr{Z}_m]G_kR^{-1}, \\ C_k^{new} &= \widehat{\mathscr{V}}_{m+1}Q = [C_k, \mathscr{V}_m, P_{m-1}, \widetilde{W}_m]Q, \end{aligned}$$

   where Q and R are the factors of the reduced QR-factorization of the tall and skinny matrix $\underline{\mathscr{F}}_m G_k$, ensuring $AU_k^{new} = C_k^{new}$ with $(C_k^{new})^H\,C_k^{new} = I_k$.

3. *The residual at restart $R_1^{new} = R_m^{old} = B - AX_1^{new}$ with $X_1^{new} = X_m^{old}$ is orthogonal to C_k^{new}.*

Proof. Given the proof essentially follows the same arguments as the ones developed for Proposition 6 or 2, the details are omitted here. □

Although the strategy A depicted in Proposition 6 is the most efficient way among the possible three strategies described in [18] for approximating the eigen-information of the coefficient matrix A, the computational cost of the last n_m columns of $\widehat{\mathscr{V}}_{m+1}^H \widehat{\mathscr{Z}}_m$ as shown in the right-hand side of (A.2) is too heavy especially with larger n_m. Therefore, another possible alternatives are considered to reduce the computational cost of solving such general eigen-solving problem. Inspired from the way of computing eigen-information under the context of flexible GMRES with deflated restarting (FGMRES-DR) as shown in [35, Proposition 1], strategy B shown in Proposition 7 is described for the IB-BFGCRO-DR, while which turns out to be not that suitable under the GCRO-DR context by numerical results shown in Table A.1. Thus, the strategy C is devised and described in Proposition 2, which has the same sense as strategy A but with a lower computational cost of solving the general eigen-solving problem as shown in (2.27). From Table A.1, it is easy to observed that the numerical result of IB-BFGCRO-DR with strategy C is approximate to that with strategy A even though the later one costs the fewest $\#mvps$ and $\#iter$.

Number of families	Method	$\#mvps$	$\#iter$
3	IB-BFGCRO-DR (strategy A)	**1807**	**144**
	IB-BFGCRO-DR (strategy B)	2074	177
	IB-BFGCRO-DR (strategy C)	1838	148

Table A.1 – Numerical results of IB-BFGCRO-DR with three kinds of strategies in terms of $\#mvps$ and $\#iter$, in which the involving parameters for QCD matrix are set to be $p = 12$, $m_d = 15 \times p = 180$ and $k = 90$.

A.2 The pseudocode of the D-BCG variants

Algorithm 13 Recycling deflated BCG with refining deflation space computed by Raleigh-Ritz projection with LO-TR of the eigen-search space — D-BCG (RR)

Require: As the first four requires stated in Algorithm 10 (i.e., D-BCG)
Require: $\widehat{m}$ maximum number of the block iteration step
Require: $s \in N^+$ the maximal number of refining cycle, $s_m \in N^+$ (initialized as $s_m = 0$) the current index of refining cycle
Require: m the maximal iteration step of a single refining cycle, $m_d = k + m \times p$ the maximal length of a refining cycle, where k is the dimension of deflation space to be refined, and let $m_{dpre} = k + (m-1) \times p$
Require: $\mathscr{D}_m = [\quad]$, $\mathscr{P}_m = [\quad]$ initial storage for refining deflation space
Step 1-5 of Algorithm 10 (i.e., D-BCG)
for $j = 0, 1, 2, \ldots, \widehat{m}$ **do**
  Step 7-18 of Algorithm 10
  / Refining deflation space */*
  if $s_m < s$ **then**
    $s_{mj} = j - s_m \times m$
    if $s_{mj} \leq m - 1$ **then**
      $\mathscr{D}_m(s_{mj} \times p + 1 : (s_{mj}+1) \times p, s_{mj} \times p + 1 : (s_{mj}+1) \times p) = P_j^H Q_j$
      $\mathscr{P}_m = [\mathscr{P}_m, P_j]$
    end if
    if $s_{mj} = m - 1$ **then**
      $G^{(RR)}(1:k, 1:k) = W_k^H A W_k,\ G^{(RR)}(k+1:m_d, k+1:m_d) = \mathscr{D}_m$
      $F^{(RR)}(1:k, 1:k) = W_k^H W_k,\ F^{(RR)}(k+1:m_d, k+1:m_d) = \mathscr{P}_m^H \mathscr{P}_m$
      $F^{(RR)}(1:k, k+1:m) = W_k^H \mathscr{P}_m$
      $F^{(RR)}(k+1:m, 1:k) = F^{(RR)}(1:k, k+1:m)^H$
      $G_{pre}^{(RR)} = G^{(RR)}(1:m_{dpre}, 1:m_{dpre}),\ F_{pre}^{(RR)} = F^{(RR)}(1:m_{dpre}, 1:m_{dpre})$
      Carry out the LO-TR [122, Algorithm 3.2] with eigen-search space: $[W_k, \mathscr{P}_m]$, i.e., solve the general eigen-problem $G^{(RR)} Y_{k/2} = Y_{k/2} F^{(RR)}$ ($Y_{k/2} \in \mathbb{C}^{m_d \times k/2}$) and $G_{pre}^{(RR)} \bar{Y}_{k/2} = \bar{Y}_{k/2} F_{pre}^{(RR)}$ ($\bar{Y}_{k/2} \in \mathbb{C}^{m_{dpre} \times k/2}$), a block zero vector is appended to $\bar{Y}_{k/2}$ as: $\begin{bmatrix} \bar{Y}_{k/2} \\ 0_{p \times k/2} \end{bmatrix} \in \mathbb{C}^{m_d \times k/2}$, $Q = \texttt{orth}([Y_{k/2}, \tilde{Y}_{k/2}])$ for returning an orthogonal basis of $Y = [Y_{k/2}, \tilde{Y}_{k/2}]$, then compute $W_k^{new} = [W_k, \mathscr{P}_m] Q$ and its image AW_k^{new}
      Let $W_k = W_k^{new},\ AW_k = AW_k^{new},\ \mathscr{D}_m = [\quad],\ \mathscr{P}_m = [\quad]$
      $s_m = s_m + 1$
    end if
  end if
end for
return X_{j+1} for approximation of the current family, W_k and AW_k for next one

Algorithm 14 Recycling deflated BCG with refining deflation space computed by harmonic-Ritz projection with LO-TR of the eigen-search space — D-BCG (HR)

Require: As the first four requires stated in Algorithm 13 (i.e., D-BCG (RR))
Require: $\mathscr{D}_m = [\quad], \mathscr{P}_m = [\quad], \mathscr{Q}_m = [\quad]$ initial storage for refining deflation space
Step 1-5 of Algorithm 10
for $j = 0, 1, 2, \ldots, \widehat{m}$ **do**
  Step 7-18 of Algorithm 10
  / Refining deflation space */*
  if $s_m < s$ **then**
    $s_{mj} = j - s_m \times m$
    if $s_{mj} \leq m-1$ **then**
      $\mathscr{D}_m(s_{mj} \times p + 1 : (s_{mj}+1) \times p, s_{mj} \times p + 1 : (s_{mj}+1) \times p) = P_j^H Q_j$
      $\mathscr{P}_m = [\mathscr{P}_m, P_j]$
      $\mathscr{Q}_m = [\mathscr{Q}_m, AP_j] = [\mathscr{Q}_m, Q_j]$
    end if
    if $s_{mj} = m-1$ **then**
      $G^{(HR)}(1:k, 1:k) = (AW_k)^H AW_k,\ G^{(HR)}(k+1:m_d, k+1:m_d) = \mathscr{Q}_m^H \mathscr{Q}_m$
      $G^{(HR)}(1:k, k+1:m) = \mathscr{Q}_m^H AW_k$
      $G^{(HR)}(k+1:m, 1:k) = G^{(HR)}(1:k, k+1:m)^H$
      $F^{(HR)}(1:k, 1:k) = W_k^H AW_k,\ F^{(HR)}(k+1:m_d, k+1:m_d) = \mathscr{D}_m$
      $G_{pre}^{(HR)} = G^{(HR)}(1:m_{dpre}, 1:m_{dpre}),\ F_{pre}^{(HR)} = F^{(HR)}(1:m_{dpre}, 1:m_{dpre})$
      Step 16 of Algorithm 13 but with $G^{(HR)}, F^{(HR)}, G_{pre}^{(HR)}, F_{pre}^{(HR)}$
      Let $W_k = W_k^{new},\ AW_k = AW_k^{new},\ \mathscr{D}_m = [\quad],\ \mathscr{P}_m = [\quad],\ \mathscr{Q}_m = [\quad]$
      $s_m = s_m + 1$
    end if
  end if
end for
return X_{j+1} for approximation of the current family, W_k and AW_k for next one

Algorithm 15 Recycling IB-Deflated-BCG with partial convergence detection and refining deflation space computed by RR projection with LO-TR of the eigen-search space — IB-D-BCG (RR)

Require: As require stated in Algorithm 13 (i.e., D-BCG (RR))
1: Step 1-5 of Algorithm 10 (i.e., D-BCG)
2: **for** $j = 0, 1, 2, \ldots, \widehat{m}$ **do**
3: Step 7-11 of Algorithm 10
4: **if** the stopping criterion related to ε or $maxMvps$ is met **then**
5: stop and return final results, W_k and AW_k
6: **else**
7: Step 11-14 of Algorithm 9 (i.e., IB-BCG)
8: **end if**
/ Refining deflation space */*
9: **if** $s_m < s$ and *the column number of* β_j = *the row number of* $\alpha_j = p$ **then**
10: Step 5-19 of Algorithm 13
11: **end if**
12: **end for**
13: **return** X_{j+1} for approximation of the current family, W_k and AW_k for next one

Algorithm 16 Recycling IB-Deflated-BCG with partial convergence detection and refining deflation space computed by HR projection with LO-TR of the eigen-search space — IB-D-BCG (HR)

Require: As require stated in Algorithm 14 (i.e., D-BCG (HR))
1: Step 1-5 of Algorithm 10 (i.e., D-BCG)
2: **for** $j = 0, 1, 2, \ldots, \widehat{m}$ **do**
3: Step 7-11 of Algorithm 10
4: **if** the stopping criterion related to ε or $maxMvps$ is met **then**
5: Stop and return final results, W_k and AW_k
6: **else**
7: Step 11-14 of Algorithm 9 (i.e., IB-BCG)
8: **end if**
/ Refining deflation space */*
9: **if** $s_m < s$ and *the column number of* β_j = *the row number of* $\alpha_j = p$ **then**
10: Step 5-20 of Algorithm 14
11: **end if**
12: **end for**
13: **return** X_{j+1} for approximation of the current family, W_k and AW_k for next one

A.3 Skeleton of two main contributions in hybridizing machine learning and numerical linear algebra

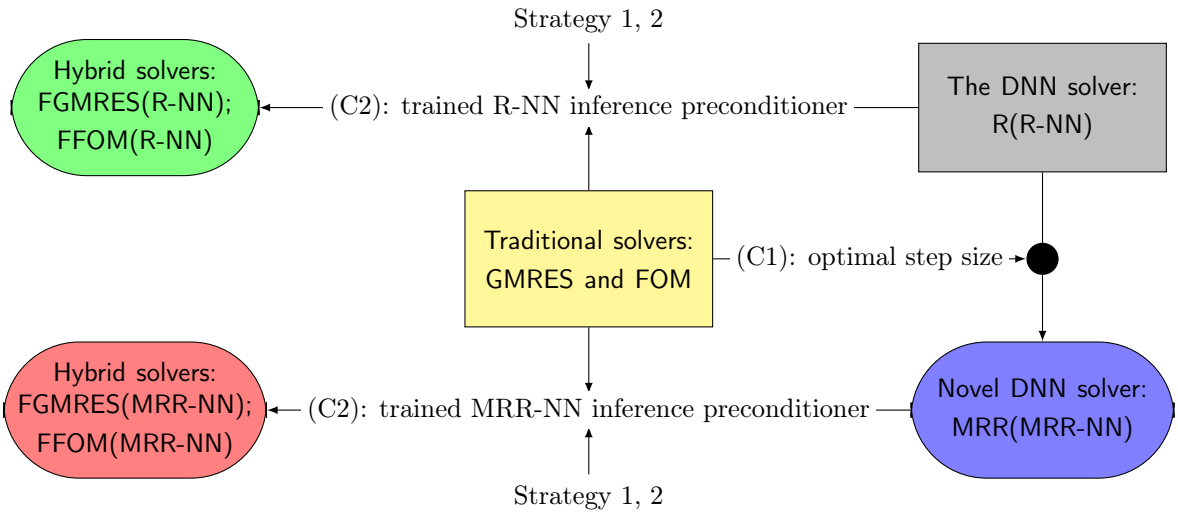
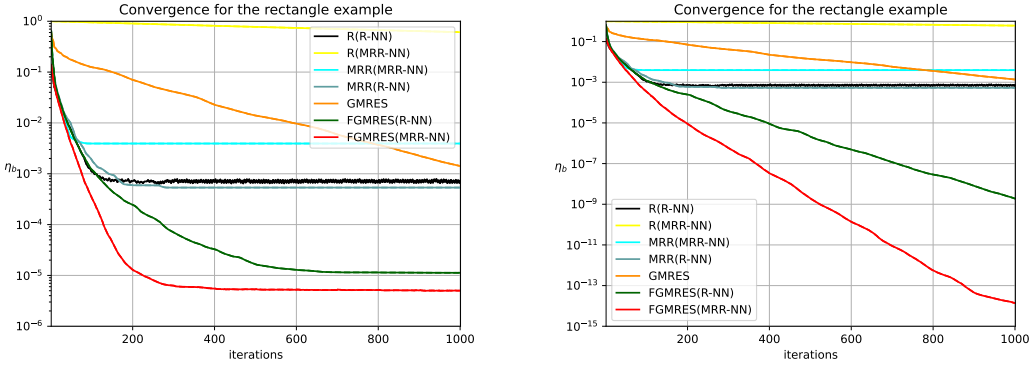


Figure A.1 – Two main contributions (simplified as C1 and C2) in hybridizing machine learning (ML) and numerical linear algebra (NLA) techniques (refer to Section 5.3.2 for the details of Strategy 1, 2).
*Note that the rectangular shapes indicate the existing traditional/DNN solver, the ellipses refer to the novel DNN solver and the hybrid variants proposed in Chapter 5, and the involved solvers are identified by varying colors that labeled each solver in the experimental Section 5.4.2.

A.4 Network generalizability: MRR-NN in epoch 255



(a): fp32 calculation (b): mixed arithmetic calculation

Figure A.2 – Convergence histories of NNs with $\alpha = 1000$ and FGMRES with Strategy 2 for the rectangle example.

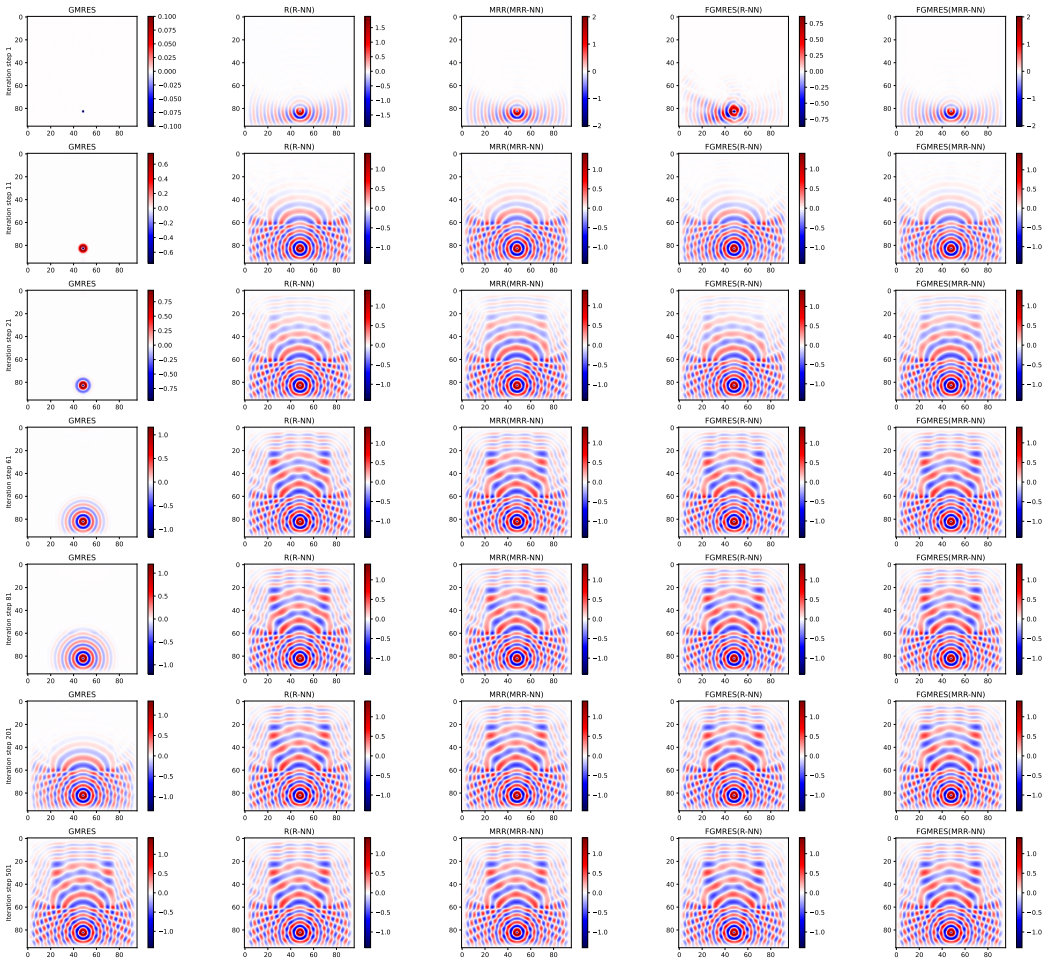
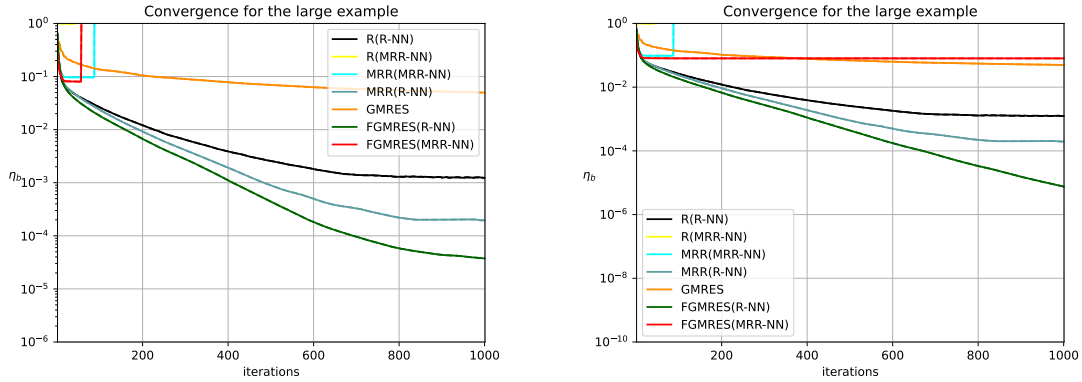


Figure A.3 – Simulation process of the rectangle example by the involved five solvers with fp32.



(a): fp32 calculation (b): mixed arithmetic calculation

Figure A.4 – Convergence histories of NNs with $\alpha = 1000$ and FGMRES with Strategy 2 for the large example.

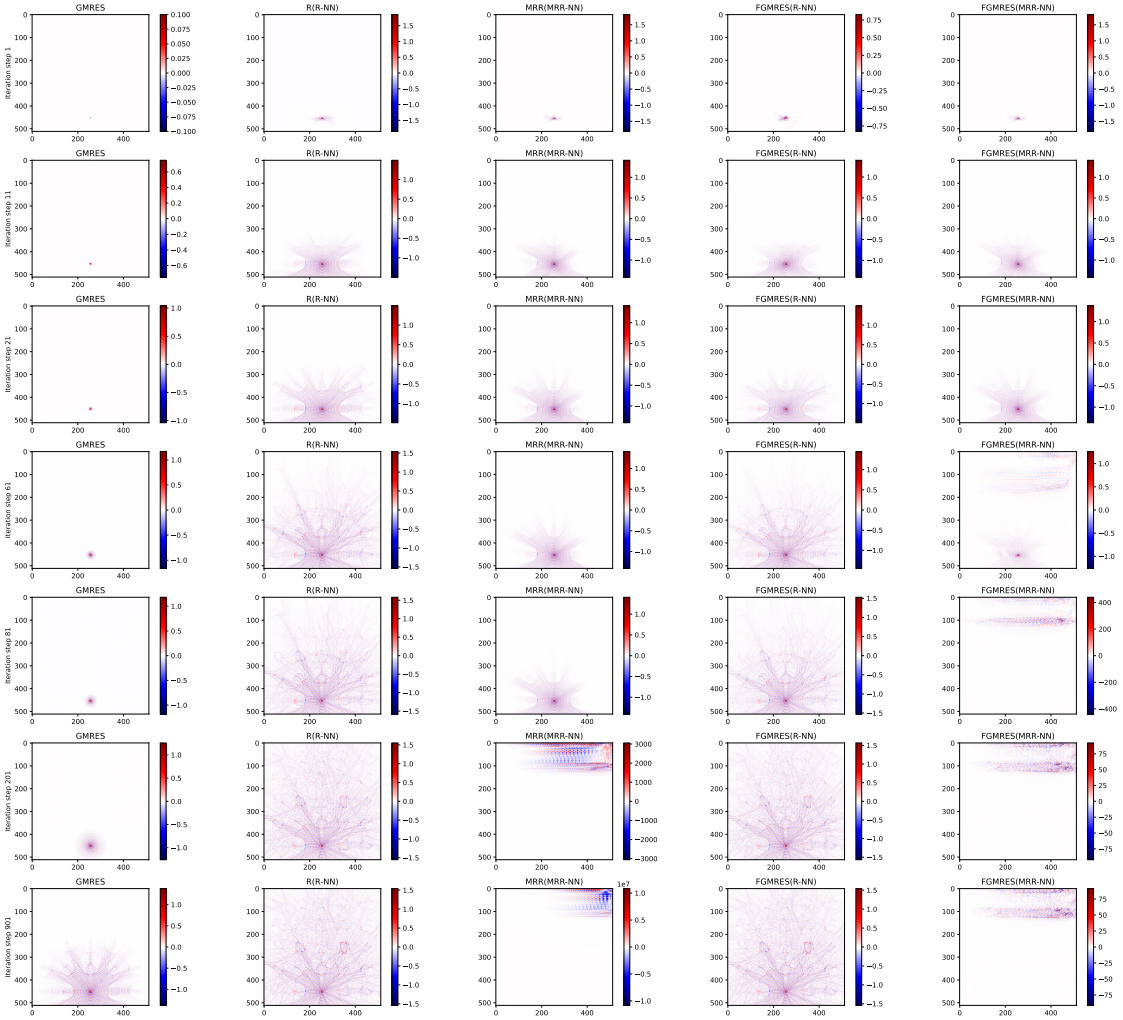
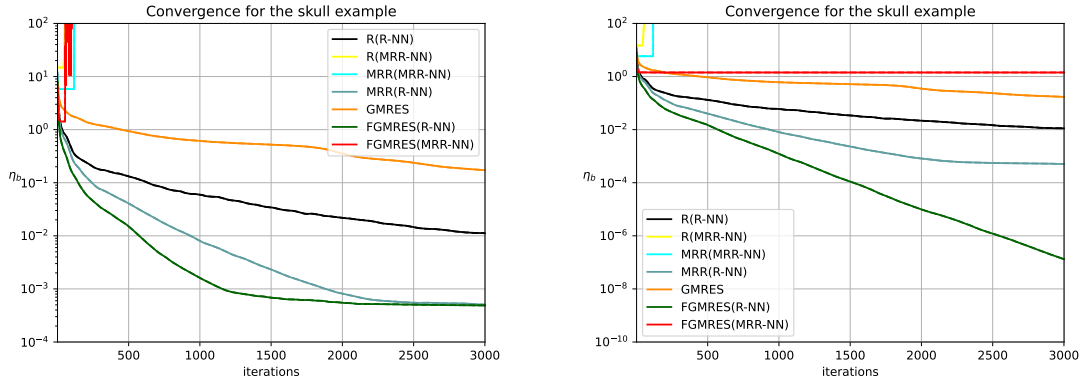


Figure A.5 – Simulation process of the large example by the involved five solvers with fp32.



(a): fp32 calculation (b): mixed arithmetic calculation

Figure A.6 – Convergence histories of NNs with $\alpha = 1000$ and FGMRES with Strategy 2 for the skull example.

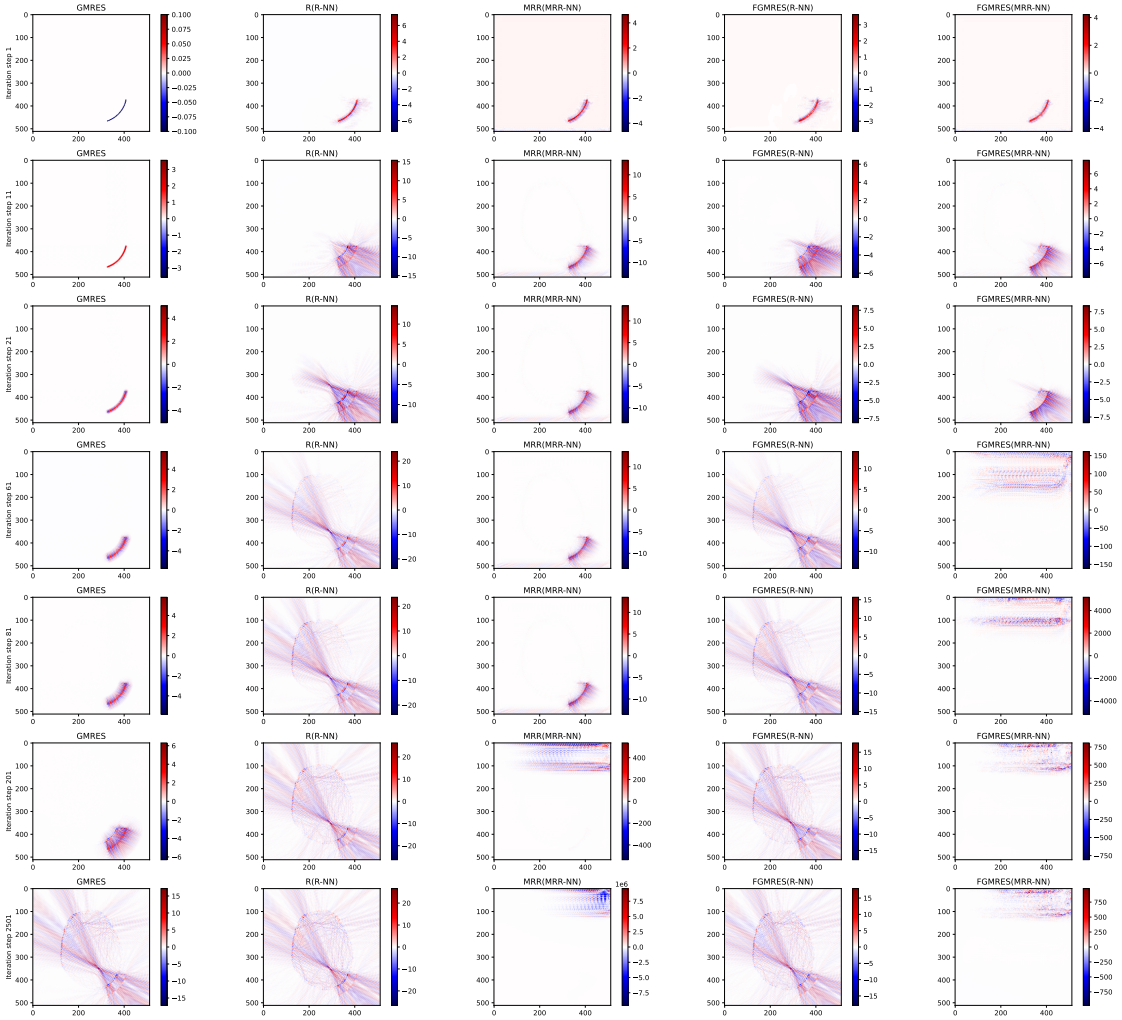


Figure A.7 – Simulation process of the skull example by the involved five solvers with fp32.