# Résolution de problèmes des moindres carrés non-linéaires régularisés dans l'espace dual avec applications à l'assimilation de données

Selime Gürol


# THÈSE

**En vue de l'obtention du**

## DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

**Délivré par :**
Institut National Polytechnique de Toulouse (INP Toulouse)

**Discipline ou spécialité :**

Signal, Image, Acoustique et Optimisation

---

**Présentée et soutenue par :**
Selime GÜROL
**le :** vendredi 14 juin 2013

**Titre :**

Solving regularized nonlinear least-squares problem in dual space
with application to variational data assimilation

---

**Ecole doctorale :**
Mathématiques Informatique Télécommunications (MITT)

**Unité de recherche :**
CERFACS

**Directeur(s) de Thèse :**
Serge Gratton, Professeur à l'ENSEEIHT
Philippe L. Toint, Professeur à l'Université de Namur

**Rapporteurs :**

Eric Blayo, Professeur à l'Université Joseph Fourier
Andrew Wathen, Chercheur senior à Oxford University

**Membre(s) du jury :**

Mike Fisher, Chercheur senior à ECMWF
Xavier Vasseur, Chercheur senior au CERFACS
Anthony T. Weaver, Chercheur senior au CERFACS

# THÈSE

présentée en vue de l'obtention du titre de

DOCTEUR DE L'UNIVERSITÉ DE TOULOUSE

Spécialité : Mathématiques, Informatique et Télécommunications

par

Selime GÜROL

CERFACS

## Résolution de problèmes des moindres carrés non-linéaires regularisés dans l'espace dual avec applications à l'assimilation de données

soutenue le 14 juin 2013 devant le jury composé de:

| | | |
|---|---|---|
| Serge Gratton | Directeur de thèse | IRIT et CERFACS |
| Philippe L. Toint | Co-directeur de thèse | Université de Namur |
| Eric Blayo | Rapporteur | Université Joseph Fourier |
| Andrew Wathen | Rapporteur | Oxford University |
| Mike Fisher | Examinateur | ECMWF |
| Xavier Vasseur | Examinateur | CERFACS |
| Anthony T. Weaver | Examinateur | CERFACS |

*To my family*

# Résumé

**Mots-clés:**
assimilation de données, approche duale, optimisation, précondionnement, méthode des gradients conjugués, méthode de Lanczos, méthodes de régions de confiance

Cette thèse étudie la méthode du gradient conjugué et la méthode de Lanczos pour la résolution de problèmes aux moindres carrés non-linéaires sous-déterminés et régularisés par un terme de pénalisation quadratique. Ces problèmes résultent souvent d'une approche du maximum de vraisemblance, et impliquent un ensemble de $m$ observations physiques et $n$ inconnues estimées par régression non linéaire. Nous supposons ici que $n$ est grand par rapport à $m$. Un tel cas se présente lorsque des champs tridimensionnels sont estimés à partir d'observations physiques, par exemple dans l'assimilation de données appliquée aux modèles du système terrestre.

Un algorithme largement utilisé dans ce contexte est la méthode de Gauss-Newton (GN), connue dans la communauté d'assimilation de données sous le nom d'assimilation variationnelle des données quadridimensionnelles. Le procédé GN repose sur la résolution approchée d'une séquence de moindres carrés linéaires optimale dans laquelle la fonction coût non-linéaire des moindres carrés est approximée par une fonction quadratique dans le voisinage de l'itération non linéaire en cours. Cependant, il est bien connu que cette simple variante de l'algorithme de Gauss-Newton ne garantit pas une diminution monotone de la fonction coût et sa convergence n'est donc pas garantie. Cette difficulté est généralement surmontée en utilisant une recherche linéaire (Dennis and Schnabel, 1983) ou une méthode de région de confiance (Conn, Gould and Toint, 2000), qui assure la convergence globale des points critiques du premier ordre sous des hypothèses faibles. Nous considérons la seconde de ces approches dans cette thèse. En outre, compte tenu de la grande échelle de ce problème, nous proposons ici d'utiliser un algorithme de région de confiance particulier s'appuyant sur la méthode du gradient conjugué tronqué de Steihaug-Toint pour la résolution approchée du sous-problème (Conn, Gould and Toint, 2000,

p. 133-139)

La résolution de ce sous-problème dans un espace à n dimensions (par CG ou Lanczos) est considérée comme l'approche primale. Comme alternative, une réduction significative du coût de calcul est possible en réécrivant l'approximation quadratique dans l'espace à $m$ dimensions associé aux observations. Ceci est important pour les applications à grande échelle telles que celles quotidiennement traitées dans les systèmes de prévisions météorologiques. Cette approche, qui effectue la minimisation de l'espace à m dimensions à l'aide CG ou de ces variantes, est considérée comme l'approche duale.

La première approche proposée (Da Silva *et al.*, 1995; Cohn *et al.*, 1998; Courtier, 1997), connue sous le nom de Système d'analyse Statistique de l'espace Physque (PSAS) dans la communauté d'assimilation de données, commence par la minimisation de la fonction de coût duale dans l'espace de dimension $m$ par un CG préconditionné (PCG), puis revient l'espace à $n$ dimensions. Techniquement, l'algorithme se compose de formules de récurrence impliquant des vecteurs de taille $m$ au lieu de vecteurs de taille $n$. Cependant, l'utilisation de PSAS peut être excessivement coûteuse car il a été remarqué que la fonction de coût linéaire des moindres carrés ne diminue pas monotonement au cours des itérations non-linéaires.

Une autre approche duale, connue sous le nom de méthode du gradient conjugué préconditionné restreint (RPCG), a été proposée par Gratton and Tshimanga (2009). Celle-ci génère les mêmes itérations en arithmétique exacte que l'approche primale, à nouveau en utilisant la formule de récurrence impliquant des vecteurs taille m. L'intérêt principal de RPCG est qu'il en résulte une réduction significative de la mémoire utilisée et des coûts de calcul tout en conservant la propriété de convergence souhaitée, contrairement à l'algorithme PSAS. La relation entre ces deux approches duales et la dérivation de préconditionneurs efficaces (Gratton, Sartenaer and Tshimanga, 2011), essentiels pour les problèmes à grande échelle, n'ont pas été abordées par Gratton and Tshimanga (2009).

La motivation principale de cette thèse est de répondre à ces questions. En particulier, nous nous intéressons à la conception de techniques de préconditionnement et à une généralisation des régions de confiance qui maintiennent la correspondance une-à-une entre itérations primales et duales, offrant ainsi un calcul efficace avec un algorithme globalement convergent.

# Abstract

**Keywords:**
data assimilation, dual approach, optimization, preconditioning, conjugate-gradients, Lanczos method, trust-region methods

This thesis investigates the conjugate-gradient (CG) method and Lanczos method for the solution of under-determined nonlinear least-squares problems regularized by a quadratic penalty term. Such problems often result from a maximum likelihood approach, and involve a set of $m$ physical observations and $n$ unknowns that are estimated by nonlinear regression. We suppose here that $n$ is large compared to $m$. These problems are encountered for instance when three-dimensional fields are estimated from physical observations, as is the case in data assimilation in Earth system models (Daley, 1991; Kalnay, 2003). In meteorological applications for example, the estimation field in the assimilation procedure is the initial state of the dynamical system, which is then integrated forward in time to produce a weather forecast.

A widely used algorithm in this context is the Gauss-Newton (GN) method, known in the data assimilation community under the name of *incremental four dimensional variational data assimilation* (Incremental 4D-Var) (Courtier, Thépaut and Hollingsworth, 1994). The GN method relies on the approximate solution of a sequence of linear least-squares problems in which the nonlinear least-squares cost function is approximated by a quadratic function in the neighbourhood of the current nonlinear iterate. However, it is well known that this simple variant of the Gauss-Newton algorithm does not ensure a monotonic decrease of the cost function and that convergence is not guaranteed. Removing this difficulty is typically achieved by using a line-search (Dennis and Schnabel, 1983) or trust-region (Conn, Gould and Toint, 2000) strategy, which ensures global convergence to first order critical points under mild assumptions. We consider the second of these approaches in this thesis. Moreover, taking into consideration the large-scale nature of the problem, we propose here to use a particular trust-region algorithm relying on the Steihaug-Toint

truncated conjugate-gradient method for the approximate solution of the subproblem (Conn, Gould and Toint, 2000, pp. 133-139).

Solving this subproblem in the $n$-dimensional space (by CG or Lanczos) is referred to as the *primal approach*. Alternatively, a significant reduction in the computational cost is possible by rewriting the quadratic approximation in the $m$-dimensional space associated with the observations. This is important for large-scale applications such as those solved daily in weather prediction systems. This approach, which performs the minimization in the $m$-dimensional space using CG (or Lanczos) or variants thereof, is referred to as the *dual approach* for reasons that will be discussed in Chapter 4.

The first proposed dual approach (Da Silva *et al.*, 1995; Cohn *et al.*, 1998; Courtier, 1997), known as the Physical-space Statistical Analysis System (PSAS) in the data assimilation community, starts by solving the corresponding dual cost function in $\mathbb{R}^m$ by a standard preconditioned CG (PCG), and then recovers the step in $\mathbb{R}^n$ through multiplication by an $n \times m$ matrix. Technically, the algorithm consists of recurrence formulas involving $m$-vectors instead of $n$-vectors. However, the use of PSAS can be unduly costly as it was noticed that the linear least-squares cost function does not monotonically decrease along the nonlinear iterations when applying standard termination criteria (El Akkroui, Gauthier, Pellerin and Buis, 2008).

Another dual approach has been proposed by Gratton and Tshimanga (2009) and is known as the Restricted Preconditioned Conjugate Gradient (RPCG) method. It generates the same iterates in exact arithmetic as those generated by the primal approach, again using recursion formula involving $m$-vectors. The main interest of RPCG is that it results in significant reduction of both memory and computational costs while maintaining the desired convergence property, in contrast with the PSAS algorithm. The relation between these two dual approaches and the question of deriving efficient preconditioners (Gratton, Sartenaer and Tshimanga, 2011) – essential when large-scale problems are considered – was not addressed in Gratton and Tshimanga (2009).

The main motivation for this thesis is to address these open issues. In particular, we are interested in designing preconditioning techniques and a trust-region globalization which maintain the one-to-one correspondance between primal and dual iterates, thereby offering a cost-effective computation in a globally convergent algorithm.

## Acknowledgements

This thesis would not have been possible without the unconditional and endless support of my advisors, Serge Gratton and Philippe Toint. First of all I would like to thank them for their teaching, sharing their experiences with me, broadening my perspective and more importantly, lightening my way in this challenging life.

I would also like to thank the referees, Andy Wathen and Eric Blayo, for their reading in detail and providing me their valuable suggestions. I want to thank the other members of the jury, Mike Fisher, Xavier Vasseur and Anthony Weaver for their accepting to examine my work.

I felt like at home from the beginning of my PhD thanks to warm atmosphere of Algo team at CERFACS. Thanks to Algo team for their making my PhD experience more enjoyable and fun. Special thanks to Anke Tröltzsch for her being with me in the worst and the best of times during my PhD and lovely discussions during our tea breaks. I would also like to thank my new colleague and friend Maria Monserrat Rincon Camacho for her valuable support and reminding me to enjoy the present moment.

CERFACS administration would not be that efficient without Michèle Campassens and Brigitte Yzel. Thanks to them for their incredible support in administrative procedures. They were always available to solve the problems with patience and smile.

Moreover thanks to Anthony Weaver from Climate Modelling and Global Change group of CERFACS and Andrew Moore from University of California at Santa Cruz for their collaboration in applying the algorithms into a realistic applications which gave me a great motivation during my PhD.

When I first started to my PhD, I have met three very thoughtful and generous friends, Fadela Kabeche, Benjamin François and Miriam D'Errico in the residence of Météo France. I want to thank them for their full support in my PhD, making my life easier in Toulouse and leaving me wonderful memories.

I always appreciate spending time with my dear friends, Kamer Kaya, Zafer Zeren, Tuğçe Kut Zeren, Rabia Özerkmen and Santiago Martinez Alcalde.

Thanks to them for their coloring non-academic part of my life.

I want to thank Ali Özel for his special friendship and broadening my perspective in our all discussions.

I would also like to thank Birol Gökduman for his valuable suggestions and increasing my awareness.

And my friends knowing my desire to do a PhD from the beginning and supporting me all the time to achieve my goals in this way. Thanks to Didem Akçay, Yasemin Bolat, Melike Gürün, Sezin Selçuk and Aslı Yılmaz for their support and understanding.

Thanks to my parents, Zübeyde Gürol and Derviş Gürol, for their supporting and believing me in all projects of my life. Canım annem ve babam çok emek harcadığım bu yolda bana her zaman inandığınız ve destek olduğunuz için size sonsuz teşekkür ederim.

Last but certainly not least, I want to thank Jean-Mathieu Senoner for his never-ending support, trust, encouragement and understanding.

# Contents

# List of Tables

# List of Figures

# List of Algorithms

# CHAPTER 1

## Introduction and motivation

This thesis investigates the conjugate-gradient (CG) method and Lanczos method for the solution of under-determined nonlinear least-squares problems regularized by a quadratic penalty term. Such problems often result from a maximum likelihood approach, and involve a set of $m$ physical observations and $n$ unknowns that are estimated by nonlinear regression. We suppose here that $n$ is large compared to $m$. These problems are encountered for instance when three-dimensional fields are estimated from physical observations, as is the case in data assimilation in Earth system models (Daley, 1991; Kalnay, 2003). In meteorological applications for example, the estimation field in the assimilation procedure is the initial state of the dynamical system, which is then integrated forward in time to produce a weather forecast.

A widely used algorithm in this context is the Gauss-Newton (GN) method, known in the data assimilation community under the name of *incremental four dimensional variational data assimilation* (Incremental 4D-Var) (Courtier, Thépaut and Hollingsworth, 1994). The GN method relies on the approximate solution of a sequence of linear least-squares problems in which the nonlinear least-squares cost function is approximated by a quadratic function in the neighbourhood of the current nonlinear iterate. However, it is well known that this simple variant of the Gauss-Newton algorithm does not ensure a monotonic decrease of the cost function and that convergence is not guaranteed. Removing this difficulty is typically achieved by using a line-search (Dennis and Schnabel, 1983) or trust-region (Conn, Gould and Toint, 2000) strategy, which ensures global convergence to first order critical points under mild assumptions. We consider the second of these approaches in this thesis. Moreover, taking into consideration the large-scale nature of the problem, we propose here to use a particular trust-region algorithm relying on the Steihaug-Toint truncated conjugate-gradient method for the approximate solution of the subproblem (Conn, Gould and Toint, 2000, pp. 133-139).

Solving this subproblem in the $n$-dimensional space (by CG or Lanczos) is referred to as the *primal approach*. Alternatively, a significant reduction in the computational cost is possible by rewriting the quadratic approximation in the $m$-dimensional space

associated with the observations. This is important for large-scale applications such as those solved daily in weather prediction systems. This approach, which performs the minimization in the $m$-dimensional space using CG (or Lanczos) or variants thereof, is referred to as the *dual approach* for reasons that will be discussed in Chapter 4.

The first proposed dual approach (Da Silva *et al.*, 1995; Cohn *et al.*, 1998; Courtier, 1997), known as the Physical-space Statistical Analysis System (PSAS) in the data assimilation community, starts by solving the corresponding dual cost function in $\mathbb{R}^m$ by a standard preconditioned CG (PCG), and then recovers the step in $\mathbb{R}^n$ through multiplication by an $n \times m$ matrix. Technically, the algorithm consists of recurrence formulas involving $m$-vectors instead of $n$-vectors. However, the use of PSAS can be unduly costly as it was noticed that the linear least-squares cost function does not monotonically decrease along the nonlinear iterations when applying standard termination criteria (El Akkroui, Gauthier, Pellerin and Buis, 2008).

Another dual approach has been proposed by Gratton and Tshimanga (2009) and is known as the Restricted Preconditioned Conjugate Gradient (RPCG) method. It generates the same iterates in exact arithmetic as those generated by the primal approach, again using recursion formula involving $m$-vectors. The main interest of RPCG is that it results in significant reduction of both memory and computational costs while maintaining the desired convergence property, in contrast with the PSAS algorithm. The relation between these two dual approaches and the question of deriving efficient preconditioners (Gratton, Sartenaer and Tshimanga, 2011) – essential when large-scale problems are considered – was not addressed in Gratton and Tshimanga (2009).

The main motivation for this thesis is to address these open issues. In particular, we are interested in designing preconditioning techniques and a trust-region globalization which maintain the one-to-one correspondance between primal and dual iterates, thereby offering a cost-effective computation in a globally convergent algorithm.

The outline of the thesis is as follows.

Chapter 2 gives the background material that will be useful for the following chapters. It consists of the general formulation of the nonlinear least-squares problem and explains the solution methods for these kind of problems, in particular the GN method and its variants. CG and Lanczos methods that are used for the solution of the linear systems arising in a GN method are then explained. Preconditioning is discussed, with particular emphasis on the class of Limited Memory Preconditioners (LMPs) (Gratton, Sartenaer and Tshimanga, 2011). The chapter is concluded by explaining the dual approach which will be exploited in Chapter 4.

Chapter 3 is concerned with the solution of the subproblem using the CG method. It focuses on preconditioning of a sequence of symmetric and positive definite linear systems with multiple right-hand sides, $As = b_k$, by using directions generated during the solution of the previous linear system. Each linear system is assumed to be a member of a convergent sequence. Different ways of preconditioning using LMPs are introduced in the sense that each approach requires different forms of the preconditioner and may avoid expensive matrix-vector products (when the linear system is large scale). These approaches are later analysed when used for solving a sequence of linear systems

in the form of $A_k s = b_k$.

Chapter 4 exploits the structure of the subproblem and derives an alternative dual problem to that introduced in Chapter 2. It explores the connections between the PSAS and RPCG solvers. We introduce practical preconditioners to accelerate the convergence of the latter by taking into account the fact that a sequence of slowly varying linear least-squares problems are solved in the Gauss-Newton process near convergence. In particular, a dual-space counterpart to the LMPs explained in Chapter 3 is derived and its properties analyzed. We conclude the chapter by providing the convergence theory for RPCG when used with LMPs.

Chapter 5 introduces the Lanczos version of the CG variants presented in Chapter 3 and Chapter 4. It explains also how to derive the required information for the LMPs from a Lanczos type algorithm.

Chapter 6 is concerned with solving the subproblem by using a method based on a GN technique, made globally convergent with a trust-region strategy. In particular, an extension of the Steihaug-Toint truncated CG method to the dual space, in which the preconditioning is performed using LMPs, is presented.

Chapter 7 includes a recent paper titled "$B$-preconditioned minimization algorithms for variational data assimilation with the dual formulation" to appear in Quaterly Journal of the Royal Meteorological Society. This paper presents the numerical results obtained from ocean data assimilation systems where the inner minimization in a Gauss-Newton method is performed by RPCG and its Lanczos version, the *Restricted Preconditioned Lanczos* (RPLanczos).

Finally, conclusions are drawn in Chapter 8, and future directions are discussed.

## Contributions

The main contributions in this thesis are

- to design a general LMP that can be used with RPCG and RPLanczos with the aim of accelerating the convergence of the dual algorithm as well as keeping the convergence properties of the primal approach (Chapter 4),
- to derive particular LMPs for RPCG and RPLanczos such as the quasi-Newton LMP, the Ritz-LMP and the spectral LMP (Chapter 4),
- to design a globally convergent algorithm in dual space where the preconditioning is performed with a LMP (Chapter 6),
- to derive an alternative PCG algorithm (*PCG Inverse Free algorithm* (PCGIF)) by exploiting the structure of the linear system in primal space that can avoid expensive matrix-vector products (Chapter 3),
- to design a general LMP and particular LMPs for the PCGIF Algorithm (Chapter 3),
- to derive the Lanczos version of the PCGIF Algorithm where the preconditioning is achieved by the LMPs (Chapter 5),

- to implement and validate RPCG and RPLanczos algorithms in a real life problems, i.e. two ocean data assimilation systems named NEMOVAR and ROMS (Chapter 7).

Some results from this work are published in

- S. Gratton, S. Gürol, and Ph. L. Toint. Preconditioning and globalizing conjugate gradients in dual space for quadratically penalized nonlinear-least squares problems. Comptutational Optimization and Applications, 54:125, 2013.
- S. Gürol, A. T. Weaver, A. M. Moore, A. Piacentini, H. G. Arango, and S. Gratton. $B$-preconditioned minimization algorithms for variational data assimilation with the dual formulation. 2013. Quarterly Journal of the Royal Meteorological Society. In press.
- A. M. Moore, C. A. Edwards, J. Fiechter, P. Drake, H. G. Arango, E. Neveu, S. Gürol, and A. T. Weaver. A 4D-Var analysis system for the california current: A prototype for an operational regional ocean data assimilation system. In Data Assimilation for Atmospheric, Oceanic and Hydrologic Applicatons (Vol II.), S. K. Park and L. Xu (Eds), Springer, pp. 345-366.

## Collaborations

During this work we have developed collaboration with Anthony T. Weaver from the Global Change and Climate Modelling group at CERFACS, Toulouse, France to implement RPCG in NEMOVAR, with Andy M. Moore from the Department of Ocean Sciences, University of California at Santa Cruz, Santa Cruz, CA, USA to implement RPLanczos in ROMS and Amal El Akkraoui and Ricardo Toddling from the Global Modeling and Assimilation Office (GMAO), NASA, USA to discuss minimization algorithms for 4D-Var data assimilation system and to implement the Ritz-preconditioner in primal space.

# CHAPTER 2

## Background material

This chapter consists of fundamental information that will be a reference for the following chapters. First, the problem formulation for the regularized nonlinear least-squares problem is explained, starting from a general formulation of the nonlinear least-squares problem. After defining the problem, solution methodologies are discussed taking into consideration situations where the dimension of the problem is large. We focus on the Gauss-Newton method as a solution algorithm, in which one solves a sequence of linear least-squares subproblems where each member of the sequence is a local quadratic approximation of the original nonlinear least-squares problem. The Gauss-Newton method can be improved in terms of its convergence behaviour by using line-search or trust-region strategies that we also outline in this chapter.

The solution of the linear least-squares subproblems arising in a Gauss-Newton iteration can be found by solving the corresponding linear systems. We consider well-known Krylov subspace methods, in particular the conjugate gradients and the Lanczos methods to solve those linear systems. Lanczos methods are applicable when the system (Hessian) matrix is large and symmetric, whereas conjugate gradients are applicable when the Hessian is large, symmetric and positive definite (which is the case for the application in this study). Both methods are explained in detail and the related algorithms are provided.

It is well-known that when using iterative methods, preconditioning is critical in order to accelerate the convergence. We summarize a class of Limited Memory Preconditioners (LMPs) to precondition the linear systems when solved with a Lanczos or conjugate gradient method. These preconditioners construct an approximation to the inverse Hessian matrix by using information from preceding iterations.

We conclude the chapter by introducing the dual subproblem of the original linear least-squares subproblem which will be exploited in the coming chapters.

## 2.1 The nonlinear least-squares problem

Nonlinear least-squares problems arise most commonly from data fitting and optimal control problems that take place in various applications such as physical, chemical and aerospace applications. *The main idea is to find the best model fit to the observed data in the sense that the sum of square errors between the observed data and the model prediction is minimized.*

In other words, let us assume that we observe a phenomenon at time $t_i$, $i = 1, 2, \ldots, m$ and have access to a (possibly noisy) data set $(y_i, t_i)$. Suppose also that a prediction model

$$y = \mathcal{H}(x, t),$$

is available, where $\mathcal{H}$ is nonlinear in $x \in \mathbb{R}^n$, that provide a theoretical value of $y$ knowing a state vector $x$. Our aim is to find an estimate of the parameter vector $x$ that minimizes the sum of square errors of the residual component

$$d_i = \mathcal{H}(x, t_i) - y_i,$$

preferably in suitable norms taking care of the (statistical) knowledge on the noise affecting the data (Tarantola, 2005, p. 64). The residual components of $d_i : \mathbb{R}^n \to \mathbb{R}$ can be assembled into a *residual vector*

$$d : \mathbb{R}^n \to \mathbb{R}^m,$$

by defining

$$d(x) = [d_1(x), d_2(x), \ldots, d_m(x)]^T. \tag{2.1}$$

Minimizing the sum of square errors of the residual vector $d(x)$ leads to the nonlinear least-squares problem:

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} d(x)^T R^{-1} d(x) = \frac{1}{2} \|d(x)\|^2_{R^{-1}}, \tag{2.2}$$

where the $m \times m$ symmetric positive-definite matrix $R$ is an estimate of the data-error covariance matrix. When $m \geq n$, this nonlinear least-squares problem is called *overdetermined*. Note that the nonlinearity of the problem is due to the nonlinearity of the model $\mathcal{H}$ in $x$. If the model $\mathcal{H}$ is linear in $x$, then the minimization problem (2.2) is called a *linear* least-squares problem.

The overdetermined nonlinear least-squares problem (problem (2.2) with $m \geq n$) is typically solved by well-known line-search (Dennis and Schnabel, 1983, p. 227) and trust-region strategies (Conn, Gould and Toint, 2000) which are based on the Newton and quasi-Newton approaches with modifications that consider the special structure of the cost function $f(x)$ and of its derivatives (Nocedal and Wright, 2006, p. 247).

## 2.2 The regularized nonlinear least-squares problem

When the solution of nonlinear least-squares problem (2.2) exists but is not unique, the problem has to be regularized. This may happen for instance when $m < n$.

A possible solution technique is formulated by adding a regularization term to the original problem, as in the well-known Tikhonov regularization (Björck, 1996, p. 101). This regularization leads to the following weighted regularized problem

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\|d(x)\|^2_{R^{-1}} + \frac{1}{2}\tau^2\|x - x_c\|^2_{B^{-1}}, \tag{2.3}$$

where the center vector $x_c$ is an application-dependent vector. The vector $x_c$ can be defined from some approximation of the solution or chosen as a zero vector (Eriksson, Wedin and Gulliksson, 1998). The $n \times n$ symmetric positive-definite matrix $B$ is an estimate of the center vector-error covariance matrix, and $\tau > 0$ is the regularization parameter. The problem (2.3) is also called a *damped* nonlinear least-squares problem (Björck, 1996, p. 101).

We can reformulate the regularized problem (2.3) as

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\left\|\begin{bmatrix} d(x) \\ v(x) \end{bmatrix}\right\|^2_W = \frac{1}{2}\|u(x)\|^2_W, \tag{2.4}$$

where the $W$-norm is defined by the $(m+n)\times(m+n)$ symmetric positive-definite block matrix

$$W = \begin{bmatrix} R^{-1} & \\ & B^{-1} \end{bmatrix},$$

$v : \mathbb{R}^n \to \mathbb{R}^n$ is defined as $v(x) = [v_1(x), \ldots, v_n(x)]^T$, $v_i(x) = \tau(x_i - (x_c)_i)$, $i = 1, \ldots, n$ and $u : \mathbb{R}^n \to \mathbb{R}^{m+n}$ is defined as $u(x) = [d_1(x), \ldots, d_m(x), v(x)]^T$. As a result, the overdetermined nonlinear least-squares problem can be written as in (2.2) and the methods for overdetermined nonlinear least-squares problem can be applied to solve the problem.

Before giving details on the solution methods, we first calculate the first and second order derivatives of the cost function in (2.4) which are sometimes needed by solution methods.

## 2.3 Calculating the derivatives

The problem (2.4) can be viewed as a special case of an unconstrained optimization problem which requires for its solution the computation of the cost function $f(x)$ and sometimes of its derivative $\nabla f(x)$ and its Hessian $\nabla^2 f(x)$. We start this section by deriving first and second order derivatives of the cost function $f(x)$ given in (2.4).

Let us start with the first-derivative matrix of $u(x)$ which is simply the Jacobian

matrix $J(x) \in \mathbb{R}^{(m+n)\times n}$, defined by

$$J(x) = \begin{bmatrix} \frac{\partial d_1(x)}{\partial x_1} & \cdots & \frac{\partial d_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial d_m(x)}{\partial x_1} & \cdots & \frac{\partial d_m(x)}{\partial x_n} \\ \\ \frac{\partial v_1(x)}{\partial x_1} & \cdots & \frac{\partial v_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial v_n(x)}{\partial x_1} & \cdots & \frac{\partial v_n(x)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla u_1(x)^T \\ \vdots \\ \nabla u_m(x)^T \\ \\ \nabla u_{m+1}(x)^T \\ \vdots \\ \nabla u_{m+n}(x)^T \end{bmatrix} = \begin{bmatrix} H_1 \\ \vdots \\ H_m \\ \\ e_1^T \\ \vdots \\ e_n^T \end{bmatrix},$$

where $e_i$ is the $i$-th column of the $n \times n$ identity matrix $I_n$ and for simplicity we consider $\tau = 1$. Then Jacobian matrix can be written in a compact form as

$$J(x) = \begin{bmatrix} H \\ I_n \end{bmatrix}, \tag{2.5}$$

where $H = [H_1^T, H_2^T, \ldots, H_m^T]^T \in \mathbb{R}^{m\times n}$. So the first derivative of $f(x)$ defined in (2.4) can be expressed as follows,

$$\begin{aligned} \nabla f(x) &= \sum_{j=1}^{m+n} W_j^T u(x) \nabla u_j(x) = J(x)^T W u(x) \\ &= H^T R^{-1} d(x) + B^{-1}(x - x_c), \end{aligned} \tag{2.6}$$

where $W_j$ is the $j$th column vector of $W$ matrix and $d(x)$ is defined in (2.1).

The Hessian matrix of the component $u_j(x)$ is given by

$$\nabla^2 u_j(x) = \begin{bmatrix} \frac{\partial^2 u_j(x)}{\partial^2 x_1} & \cdots & \frac{\partial^2 u_j(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 u_j(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 u_j(x)}{\partial^2 x_n} \end{bmatrix}.$$

Note that $\nabla^2 u_j(x) = 0$ for $j = m+1, \ldots, n$ since $u_j(x)$ is linear in $x$ for $j = m+1, \ldots, m+n$. Then, the second derivative of $f(x)$ is calculated as

$$\begin{aligned} \nabla^2 f(x) &= \sum_{j=1}^{m+n} \nabla u_j(x) W_j^T J(x) + \sum_{j=1}^{m+n} \left(W_j^T u(x)\right) \nabla^2 u_j(x) \\ &= J(x)^T W J(x) + \sum_{j=1}^{m} \left(W_j^T u(x)\right) \nabla^2 u_j(x) \\ &= (B^{-1} + H^T R^{-1} H) + S(x), \end{aligned} \tag{2.7}$$

where $S(x) = \sum_{j=1}^{m} \left(W_j^T u(x)\right) \nabla^2 u_j(x)$ includes second order derivatives in $\nabla^2 f(x)$.

## 2.4 Solving the regularized problem

It is very common to find the solution of the nonlinear least-squares problem of the form (2.4) by Newton's method. Starting from an initial vector $x_0$, Newton's method generates a sequence of iterates $\{x_k\}$, $k = 0, 1, \ldots, K$. This sequence is generated by computing the search direction $s$ based on the quadratic model $m_k$

$$m_k(s) = f(x_k) + s^T \nabla f(x_k) + \frac{1}{2} s^T \nabla^2 f(x_k) s. \tag{2.8}$$

This model is the Taylor approximation of the function $f(x)$ in the neighborhood of a given iterate $x_k$. Assuming that $\nabla^2 f(x_k)$ is positive definite, the minimizer of the quadratic function (2.8) is given by setting the derivative of $m_k(s)$ to zero, i.e.

$$s_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k). \tag{2.9}$$

Thus, the next iterate is updated according to

$$x_{k+1} = x_k + s_k. \tag{2.10}$$

Replacing the objective function $f(x)$, the gradient $\nabla f(x_k)$ and Hessian $\nabla^2 f(x_k)$ by (2.4), (2.6) and (2.7) respectively in (2.8) leads to

$$m_k(s) = \frac{1}{2} u_k^T W u_k + s^T (H_k^T R^{-1} d_k + B^{-1}(x_k - x_c)) + \frac{1}{2} s^T (B^{-1} + H_k^T R^{-1} H_k + S_k) s, \tag{2.11}$$

where $u_k = u(x_k)$, $d_k = d(x_k)$, $S_k = S(x_k)$ and $H_k = [\nabla u_1(x_k)^T, \ldots, \nabla u_m(x_k)^T]^T$. Similarly, replacing $\nabla f(x_k)$ and $\nabla^2 f(x_k)$ in (2.9) gives

$$s_k = (B^{-1} + H_k^T R^{-1} H_k + S_k)^{-1} (B^{-1}(x_c - x_k) - H_k^T R^{-1} d_k). \tag{2.12}$$

Therefore, from (2.10) and (2.12), the minimizer of the regularized problem (2.4) in the neighborhood of $x_k$ is given as follows

$$x_{k+1} = x_k + (B^{-1} + H_k^T R^{-1} H_k + S_k)^{-1} (B^{-1}(x_c - x_k) - H_k^T R^{-1} d_k).$$

In (2.7) usually matrix vector products with $H$ are easily available. However, this is not the case for $S(x)$ which includes second order derivatives that are usually not available. As mentioned in (Dennis and Schnabel, 1983, p. 218), [p. 263](Nocedal and Wright, 2006) there are different iterative approaches to handle this issue. For a *zero-residual* problem, i.e. $S(x^*) = 0$ where $x^*$ is the solution, or a *small-residual* problem, i.e. $S(x^*)$ is small relative to $J(x^*)^T W J(x^*)$, $S(x)$ can be omitted. It leads to the Gauss-Newton algorithm. As it can be easily seen, the Gauss-Newton method convergence depends on the problem nonlinearity. For a zero-residual problem the Gauss-Newton method is locally q-quadratically convergent whereas for the problems that are not very nonlinear (small-residual problem), the method is locally q-linearly convergent (Dennis and Schnabel, 1983, p. 225). For a *large-residual* problem, i.e.

$S(x^*)$ is large relative to $J(x^*)^T W J(x^*)$, the method may not even be locally convergent (Björck, 1996, p.343). The Gauss-Newton method convergence also depends on the starting vector $x_0$. In particular, it is not necessarily globally convergent, i.e. a sequence of iterates $\{x_k\}$ obtained by Gauss Newton method starting from an arbitrary initial vector $x_0$ may not converge to a local minimizer of $f(x)$. However global convergence can be obtained by introducing a line search or trust region approachs.

For a large-residual nonlinear problem, another alternative can be to use nonlinear optimization algorithms directly. However, in this approach the special structure of the nonlinear least-squares problem is ignored and the convergence rate of algorithm may be inferior to Gauss-Newton method for small or zero-residual problems.

A hybrid method can be another option to handle the problem. The main idea in this approach is to consider the Gauss-Newton method if the problem is a zero-residual or small residual problem, otherwise consider the approaches that handles a large residual problem. For details on hybrid methods, we refer to the references (Nocedal and Wright, 2006, p. 263-264), (Björck, 1996, p. 348-351).

We focus on the Gauss-Newton type methods throughout this study, since our application problem arises from operational data assimilation, which is a large-scale problem with a small-residual, and it is well-known that the Gauss-Newton algorithm is an efficient solver for this kind of problem.

## 2.5 The Gauss-Newton method

The Gauss-Newton method consists of solving a sequence of linearized least-squares problems

$$\min_s \frac{1}{2} \|H_k s + d_k\|^2_{R^{-1}} + \frac{1}{2} \|s + x_k - x_c\|^2_{B^{-1}}, \tag{2.13}$$

where $J$, $u$ and $W$ are defined in Section 2.2. The cost function in (2.13) is a quadratic approximation of the nonlinear cost function $f(x)$ given in (2.4) in the neighborhood of a given iterate $x_k$. On each iteration $k$, this quadratic cost function is minimized to determine a step $s_{k+1}$ starting from the iterate $x_k$. This step is then used to update the current iterate from

$$x_{k+1} = x_k + s_k. \tag{2.14}$$

Assuming that the Jacobian matrix $J_k$ (at iteration $k$) has full column rank, the solution to the problem (2.13) at the $k$-th iterate is given as

$$s_k = (B^{-1} + H_k^T R^{-1} H_k)^{-1} (B^{-1}(x_c - x_k) - H_k^T R^{-1} d_k). \tag{2.15}$$

The matrices $H_k$, $B$ and $R$ appearing in (2.15) may be so large that they can't be stored explicitly, and the information they contain may only be available through matrix-vector products. In this case it is natural to solve the linear system in (2.15) by Krylov subspace methods (Saad, 1996), whose sole access to the system matrix is via such products. Therefore, the Gauss-Newton algorithm first applies this method to the

linear system

$$(B^{-1} + H_k^T R^{-1} H_k)s = B^{-1}(x_c - x_k) - H_k^T R^{-1} d_k, \tag{2.16}$$

where $B^{-1} + H_k^T R^{-1} H_k$ is symmetric and positive definite and updates the iterate as in (2.14). The main loop of the Gauss-Newton method that generates the iterates $\{x_k\}$ is called the *outer loop*. The iterative loop of solving the linear system (2.16) by an iterative Krylov subspace method is called the *inner loop* since it is nested within the outer loop of the Gauss-Newton algorithm.

We next explain variants of the Gauss-Newton method by including line search or trust region strategies.

### 2.5.1 The damped Gauss-Newton method

The damped Gauss-Newton method (Björck, 1996, p. 343), (Dennis and Schnabel, 1983, p. 227) uses a line search strategy in a Gauss-Newton method. The key idea in a line search strategy is to control the step length along the *descent direction* by ensuring that it gives a sufficient decrease in the objective function $f(x)$, measured by some conditions such as the Armijo and Wolfe conditions (Nocedal and Wright, 2006, p. 21,33,34).

It can be shown that whenever $J_k$ has full column rank and the gradient $\nabla f(x_k)$ is nonzero, the Gauss-Newton step $s_k$ is in a *descent direction* (Nocedal and Wright, 2006, p. 254) for the cost function $f(x)$ given in (2.4). This suggests to improve the Gauss-Newton algorithm by using a line search strategy. A line search strategy in a Gauss-Newton method gives rise to the damped Gauss-Newton method which generates the iterates as follows:

$$x_{k+1} = x_k + \alpha_k s_k,$$

where $\alpha_k > 0$ is the step length. The selection of $\alpha_k$ is critical in this method. Ideally, it is selected as the solution of the problem

$$\min_{\alpha} \|u(x_k + \alpha s_k)\|_W^2 .$$

However in general it may be expensive to find the minimizer of this univariate function. Instead, an *inexact* line search method can be used, provided that the step sufficiently reduces the function $f(x)$ at minimal cost. For more details on the line search strategy we refer to (Nocedal and Wright, 2006, Chapter 3), (Björck, 1996, p. 344-346) and (Dennis and Schnabel, 1983, p. 116-129).

As expected from the property of the Gauss-Newton method, the damped Gauss-Newton method is locally convergent on zero and small residual problems. Additionally, it is locally convergent on large-residual problems or very nonlinear problems. It is not always globally convergent (Dennis and Schnabel, 1983, p. 227), (Björck, 1996, p. 346) and the rate of convergence may still be slow on large-residual problems.

Another issue is that when $J(x)$ is rank deficient, the damped Gauss-Newton method is not well defined (it is not possible to apply a line search strategy) since

the Gauss Newton direction may not be a descent direction. This problem can be overcome with the so-called trust region approach which is an alternative strategy to improve the Gauss-Newton algorithm. In the next section we give the details on the Gauss-Newton methods that uses a trust-region strategy.

### 2.5.2 The Levenberg-Marquart method

Improving the convergence properties of the Gauss-Newton method in a trust-region framework take its origins from Levenberg (1944), Morrison (1960) and Marquardt (1963). This method is named the Levenberg-Morrison-Marquardt method in Conn, Gould and Toint (2000) because of their independent contributions.

At first, Levenberg (1944) suggested to use a damping procedure in the Gauss-Newton method by adding a multiple of the identity to the Hessian matrix. Afterwards, Morrison (1960) and Marquardt (1963) independently showed that there is a link between damping the Hessian and reducing the length of the step length. They also recognised that minimizing a model with a damped Hessian is equivalent to minimizing the original model in a restricted area defined by the damping parameter (Conn, Gould and Toint, 2000, p.8).

On each iteration $k$, the Levenberg-Marquardt method solves the subproblem

$$\min_s \frac{1}{2} \|u_k + J_k s\|_W^2 + \frac{1}{2}\mu_k \|s\|_2^2 , \tag{2.17}$$

to compute the step where $\mu_k \geq 0$ is the parameter controlling the size of $s_k$ [p. 346](Björck, 1996). With this formulation $s_k$ is well-defined also in the case where $J(x)$ is rank-deficient. The problem (2.17) can be reformulated as a linear least-squares problem as follows:

$$\min_s \frac{1}{2} \left\| \begin{bmatrix} W^{1/2} J_k \\ \mu_k I_n \end{bmatrix} s + \begin{bmatrix} W^{1/2} u_k \\ 0 \end{bmatrix} \right\|_2^2 .$$

This formulation allows us again to use linear least-squares algorithms.

The local convergence behaviour of the Levenberg-Marquardt method is similar to that of the Gauss-Newton method (Dennis and Schnabel, 1983, Theorem 10.2.6). The method may be slowly locally convergent if the problem has a large-residual or is very nonlinear. Global convergence property of this method can be proved by reformulating the problem (2.17) as a trust-region problem (Nocedal and Wright, 2006, p. 261) whose global convergence is well-known from the trust-region theory.

We next explain the Gauss-Newton algorithm with a trust-region approach which can be considered as a generalization of the Levenberg-Marquardt method.

### 2.5.3 The Gauss-Newton method with a trust-region approach

The Gauss-Newton method equipped with a trust-region approach solves at each iteration the linear least-squares problem defined by (2.13) subject to a quadratic constraint,

yielding

$$\min_s \frac{1}{2} \|u_k + J_k s\|_W^2 , \quad \text{subject to } \|s\|_k^2 \leq \Delta_k \tag{2.18}$$

where $\|\cdot\|_k$ is an iteration dependent norm, and $\Delta_k$ is the radius of the *trust region*,

$$\mathcal{B}_k = \{x \in \mathbb{R}^n | \, \|x - x_k\|_k \leq \Delta_k\} ,$$

which is the region where we believe that the objective function of the regularized nonlinear-least squares problem (2.4) is adequately approximated by $\frac{1}{2} \|u_k + J_k s\|_W^2$.

After (possibly approximately) solving the subproblem (2.18), the step $s_k$ is accepted or rejected and the trust-region radius is updated accordingly. The acceptance of the trial point and the change in the trust-region radius are decided by considering the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}, \tag{2.19}$$

where $f$ is the objective function given in (2.3) and $m_k$ is its quadratic approximation which is defined in (2.18). This ratio of achieved to predicted reductions gives an indication of the quadratic model's quality. If it is larger than a given small positive constant, the step is accepted and the trust-region radius possibly enlarged, while, if it is too small or negative, the step is rejected and the trust-region radius decreased. This approach can be given by Algorithm 2.1 proposed by (Conn, Gould and Toint, 2000, p. 116) in which we define the subproblem by (2.18). We name this algorithm basic Gauss-Newton trust-region algorithm (BGNTR).

Up to now, we presented variants of Gauss-Newton method that can be used to solve a regularized nonlinear-least squares problem. We next give details on how to calculate the Gauss-Newton step $s_k$ by applying Krylov subspace methods to the linear system (2.16).

## 2.6 Solving the linearized subproblem

We pointed out that solving the regularized nonlinear least-squares problem (2.4) with a Gauss-Newton method consists of solving linear systems in the form of (2.16) sequentially. In this section we explain how to adapt the Krylov subspace methods in this context.

Let us define

$$A = B^{-1} + H_k^T R^{-1} H_k,$$

and

$$b = B^{-1}(x_c - x_k) - H_k^T R^{-1} d_k.$$

Then, the linear system (2.16) can be rewritten simply in the form

$$As = b. \tag{2.20}$$

**Algorithm 2.1: Basic Gauss-Newton trust region algorithm**

**1 Initialization:** Choose an initial point $x_0$ and an initial trust region radius $\Delta_0$. Choose the constants $\eta_1, \eta_2, \gamma_1, \gamma_2$ that satisfy

$$0 < \eta_1 \leq \eta_2 < 1 \text{ and } 0 < \gamma_1 \leq \gamma_2 < 1$$

Set $k = 0$ and calculate the function value at the initial, $f(x_0)$.

**2 Norm definition:** Define $\|\cdot\|_k$.

**3 Calculation of $s_{k+1}$ approximately:** Solve the subproblem (2.18).

**4 Acceptance of the trial point:** Compute the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + s_{k+1})}{m_k(x_k) - m_k(x_k + s_{k+1})},$$

If $\rho_k \geq \eta_1$, $x_{k+1} = x_k + s_{k+1}$; otherwise, $x_{k+1} = x_k$.

**5 Trust-region radius update:** Set

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty) & \text{if } \rho_k \geq \eta_2, \\ [\gamma_2 \Delta_k, \Delta_k) & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_1 \Delta_k, \gamma_2 \Delta_k) & \text{if } \rho_k < \eta_1, \end{cases}$$

Increment $k$ by 1 and go to step 2.

Krylov subspace methods search for an approximate solution for a linear system in the form of (2.20) in a subspace $s_0 + \mathcal{K}^l(A, r_0)$ where $s_0$ is the initial guess,

$$\mathcal{K}^l(A, r_0) = \text{span}\left\{r_0, Ar_0, A^2 r_0, \ldots, A^{l-1} r_0\right\}, \tag{2.21}$$

is the Krylov subspace of dimension $l$ and $r_0 = b - A\, s_0$ is the initial residual. Note that the subspaces given by (2.21) are nested, i.e. $\mathcal{K}^l \subseteq \mathcal{K}^{l+1}$.

To uniquely define the $l$-th iterate ($s_l$), several Krylov subspace methods impose the Petrov-Galerkin condition (Saad, 1996, p.144)

$$r_l \perp \mathcal{L}^l(A, r_0), \tag{2.22}$$

where $\mathcal{L}^l(A, r_0)$ is a $l$-dimensional subspace. The condition (2.22) is called a Galerkin condition when $\mathcal{L}^l(A, r_0) = \mathcal{K}^l(A, r_0)$, and leads to the Full Orthogonalization Method (FOM) which minimizes the residual in the $A^{-1}$ norm, i.e. $\|r\|_{A^{-1}}$. Choosing $\mathcal{L}^l(A, r_0) = A\mathcal{K}^l(A, r_0)$ gives the Generalized Minimum RESidual method (GMRES) which finds the approximation $s_l$ by minimizing the residual in Euclidean norm, i.e. $\|r\|_2$. If $A$ is symmetric, then FOM and GMRES can be optimized to reduce the memory needs of the methods. These algorithms are known as the Lanczos method and the MINimum RESidual method (MINRES), respectively. It is also well known that Conjugate Gradient (CG) is mathematically equivalent to the Lanczos(-Cholesky) method when $A$ is

symmetric and positive definite (Saad, 1996, p.176).

In this study, we are interested in the Lanczos and the CG method which are well-known and widely used for solving large-scale and symmetric positive definite linear systems. Therefore, in the rest of the thesis we concentrate on these two Krylov subspace methods.

### 2.6.1 The conjugate gradient method (CG)

The linear Conjugate Gradients (CG) was proposed by Hestenes and Stiefel (1952) and is among the most useful iterative techniques for solving large-scale symmetric positive definite linear systems. Actually, CG was originally considered as a direct method since in exact arithmetic the method converges in at most $n$ iterations, $n$ being the order of the linear system matrix. Later, it was shown that in finite arithmetic CG does not terminate at most $n$ iterations (Engeli *et al.*, 1960), and it was noticed that the convergence of the method depends on the distribution of the eigenvalues of the linear system matrix (Kaniel, 1966). This leads to the idea of transforming or *preconditioning* the linear system in a way that the eigenvalue distribution becomes more favorable and the convergence improves significantly. The CG method is used as an iterative method by Reid (1971) who renewed attention to this algorithm. As a result, CG is recognized as a useful iterative method for symmetric positive definite linear systems (Saad and van der Vorst, 2000).

CG solves the linear system of equations $As = b$ where $A$ is an $n \times n$ symmetric and positive definite matrix. Solving the linear system is equivalent to minimizing the quadratic problem [p. 102](Nocedal and Wright, 2006)

$$\min \phi(s) = \frac{1}{2} s^T A s - b^T s. \tag{2.23}$$

Problem (2.23) can be reformulated as

$$\min \phi(s) = \frac{1}{2} s^T A s - b^T s = \frac{1}{2}(s - s^*)^T A (s - s^*) - \frac{1}{2} b^T s^*,$$

where $s^* = A^{-1}b$ is the solution. Defining the $A$-norm which is also referred to as the energy norm to be $\|x\|_A^2 = x^T A x$ leads to

$$\min \frac{1}{2} \|s - s^*\|_A^2 - \frac{1}{2} \|b\|_{A^{-1}}^2 .$$

Since the second term of this problem is constant, we can see that minimizing the quadratic problem (2.23) is also equivalent to minimizing the energy norm of the error, i.e.

$$\min \frac{1}{2} \|s - s^*\|_A^2 . \tag{2.24}$$

The CG method minimizes the quadratic function $\phi(s)$ or equivalently solves the linear system $As = b$ by generating the iterates $\{s_i\}_{i=0,1,\ldots,l}$ as a linear combination of

the initial point $s_0 \in \mathbb{R}^n$ and the search directions $\{p_i\}_{i=0,1,\ldots,l-1}$

$$s_{i+1} = s_i + \alpha_i p_i,$$

where $\alpha_i$ is the one-dimensional minimizer of the quadratic function $\phi(\cdot)$ in the search direction $p_i$ which is computed as (Nocedal and Wright, 2006, p. 103)

$$\alpha_i = \frac{r_i^T p_i}{p_i^T A p_i}.$$

The new search direction is generated using only the previous direction, i.e.

$$p_i = r_i - \beta_i p_{i-1}$$

where $r_i$ is the residual and the scalar $\beta_i$ is chosen ensuring that $p_i$ and $p_{i-1}$ are conjugate with respect to $A$, i.e.

$$p_i^T A p_j = 0, \text{ for all } i \neq j.$$

Defining a new direction using only the previous one is a crucial property of the method in terms of its practical implementation.

The pseudo-code for the CG method is given in Algorithm 2.2.

**Algorithm 2.2: CG Algorithm**

$r_0 = b - A s_0$
$\rho_0 = r_0^T r_0$
$p_0 = r_0$
**for** $i = 0, 1, \ldots, l-1$ **do**
    $q_i = A p_i$
    $\alpha_i = \rho_i / q_i^T p_i$
    $s_{i+1} = s_i + \alpha_i p_i$
    $r_{i+1} = r_i - \alpha_i q_i$
    Check convergence and stop if desired accuracy is reached
    $\rho_{i+1} = r_{i+1}^T r_{i+1}$
    $\beta_{i+1} = \rho_{i+1} / \rho_i$
    $p_{i+1} = r_{i+1} + \beta_{i+1} p_i$
**end**

Note that, Algorithm 2.2 minimizes the quadratic problem over the set $s_0 + \text{span}\{p_0, p_1, \ldots, p_{l-1}\}$ which is the same as $s_0 + \text{span}\{r_0, Ar_0, \ldots, A^{l-1} r_0\}$ (Golub and Van Loan, 1996, p.523), where $l$ is the number of the performed successful iterations.

### 2.6.2 The Lanczos method

The Lanczos method was proposed initially by Lanczos (1950) to compute the eigenvalues of a symmetric matrix. The "Lanczos idea" is later applied to solve large sparse linear equation and least squares problem (Saunders, 1995; Saad, 1996). The results on the equivalence between the CG and Lanczos method for the solution of symmetric and positive definite linear systems can be found in Paige and Saunders (1975), Cullum and Willoughby (1980), Golub and Van Loan (1996, p. 528), Saad (1996, p. 176–182) and Meurant (2006, p. 46–49).

As mentioned in Section 2.6, when the Lanczos method is used for the solution of a linear system in the form $As = b$, the method searches for an approximate solution $s_l$ in a subspace $s_0 + \mathcal{K}^l(A, r_0)$ where $s_0$ is the initial guess, $r_0 = b - A\,s_0$ is the initial residual, and the Krylov subspace $\mathcal{K}^l(A, r_0)$ is given by (2.21). It imposes the Galerkin condition

$$r_l \perp \mathcal{K}^l(A, r_0),$$

which is equivalent to

$$V_l^T(b - As_l) = 0, \tag{2.25}$$

where $V_l = [v_1, v_2, ..., v_l]$ is an $n \times l$ matrix whose column vectors form an orthonormal basis for $\mathcal{K}^l(A, r_0)$. The basis vectors are known as Lanczos vectors and are constructed using the symmetric Lanczos algorithm (Saad, 1996, p. 174-175). The Lanczos vectors are by construction orthonormal vectors, i.e. $v_j^T v_i = 0$ for $i \neq j$ and $v_j^T v_i = 1$ for $i = j$. As a result, the approximate solution using the Krylov subspace $\mathcal{K}^l(A, r_0)$ is given by

$$s_l = s_0 + V_l y_l, \tag{2.26}$$

where $y_l$ is derived as follows.

Multiplying the expression (2.26) with the matrix $A$ from the left and using the resulting expression in the Galerkin condition (2.25) gives that

$$\begin{aligned} V_l^T(b - As_0 - AV_l y_l) &= 0, \\ V_l^T(r_0 - AV_l y_l) &= 0, \\ V_l^T AV_l y_l &= V_l^T r_0. \end{aligned} \tag{2.27}$$

Choosing $v_1 = r_0 / \|r_0\|_2$ and defining a $l$-dimensional vector $e_1 = [1, 0, ..., 0]^T$ we obtain that

$$V_l^T r_0 = V_l^T \|r_0\|_2 v_1 = \|r_0\|_2 e_1. \tag{2.28}$$

In (2.28) we have used the orthonormality property of the Lanczos vectors. Using the expressions (2.27) and (2.28) the vector $y_l$ is given by (Saad, 1996, p. 152-153)

$$y_l = T_l^{-1}(\|r_0\|_2 e_1), \tag{2.29}$$

where $T_l = V_l^T A V_l$. The matrix $T_l$ can be interpreted as a matrix representation of the projection of $A$ onto the Krylov subspace $\mathcal{K}^l(A, r_0)$.

The pseudo-code for the Lanczos method is given in Algorithm 2.3.

**Algorithm 2.3: Lanczos Algorithm for Linear Systems**

$r_0 = b - As_0$
$\beta_0 = \|r_0\|_2$
$\beta_1 = 0$
$v_0 = 0$
$v_1 = r_0/\beta_0$
$V_1 = v_1$
**for** $i = 1, ..., l$ **do**
    $w_i = Av_i - \beta_i v_{i-1}$
    $\alpha_i = w_i^T v_i$
    $w_i := w_i - \alpha_i v_i$
    $\beta_{i+1} = \|w_i\|_2$
    $v_{i+1} = w_i/\beta_{i+1}$
    $V_i := [V_i, v_{i+1}]$
    $(T_i)_{i,i} = \alpha_i$
    **if** $i > 1$ **then**
        $(T_i)_{i-1,i} = (T_i)_{i,i-1} = \beta_i$
    **end**
    Check convergence and stop if desired accuracy is reached
**end**
$y_l = T_l^{-1}(\beta_0 e_1)$
$s_l = s_0 + V_l y_l$

### 2.6.3 From Lanczos to CG

In this section we provide the formulas that allow one to calculate the CG iterates from the Lanczos iterates. First we derive the formula for the residual $r_i = b - As_i$, $i = 1, \ldots, l$. Replacing the expression (2.26) in the residual gives

$$r_i = b - A(s_0 + V_i y_i) = r_0 - AV_i y_i. \tag{2.30}$$

Using the following relation (Saad, 1996, p.148)

$$AV_i = V_i T_i + \beta_{i+1} v_{i+1} e_i^T, \tag{2.31}$$

where $V_i$, $T_i$, $\beta_{i+1}$, $v_{i+1}$ are defined in Algorithm 2.3 and $e_i$ is the $i$-th column of the $i \times i$ identity matrix, we obtain from (2.30) that

$$r_i = r_0 - V_i T_i y_i - \beta_{i+1} v_{i+1} e_i^T y_i.$$

Using the relation (2.29) and the definition $v_1 = r_0/\|r_0\|_2$ (line 5 of Algorithm 2.3), the formula for the residual becomes

$$\begin{aligned} r_i &= r_0 - V_i\|r_0\|_2 e_1 - \beta_{i+1} v_{i+1} e_i^T y_i, \\ &= r_0 - \|r_0\|_2 v_1 - \beta_{i+1} v_{i+1} e_i^T y_i, \\ &= -\beta_{i+1} v_{i+1} e_i^T y_i. \end{aligned} \tag{2.32}$$

The scalar $\alpha_i$ and the vector $q_i$ can be derived from the lines 7 and 8 of the CG Algorithm respectively. The resulting algorithm to compute the iterates of CG Algorithm 2.2 from those of Lanczos Algorithm 2.3 is given by Algorithm 2.4. In Algorithm 2.4, the vector $v_{i+1}$ is the $(i+1)$-th column of the matrix $V_l$ and $(T_l)_{i+1,i}$ refers to the $(i+1,i)$-th entry of the matrix $T_l$.

**Algorithm 2.4: Lanczos2CG**

1 Given $l, r_0, s_0, T_l, \beta_{l+1}, V_l$ from Algorithm 2.3
2 $(T_l)_{l+1,l} = \beta_{l+1}$
3 $\beta_0 = \|r_0\|_2$
4 $p_0 = r_0$
5 $\rho = \beta_0^2$
6 **for** $i = 1, \ldots, l$ **do**
7 $\quad y_i = T_i^{-1}(\beta_0 e_1)$
8 $\quad r_i = -(T_l)_{i+1,i}\, v_{i+1} e_i^T y_i$
9 $\quad \rho_i = r_i^T r_i$
10 $\quad \beta_i = \rho_i/\rho_{i-1}$
11 $\quad p_i = r_i + \beta_i p_{i-1}$
12 $\quad s_i = s_0 + V_i y_i$
13 $\quad \alpha_i = \|(s_i - s_{i-1})\|_2/\|p_{i-1}\|_2$
14 $\quad q_i = -(r_i - r_{i-1})/\alpha_i$
15 **end**

### 2.6.4 Eigenspectrum estimates

During the minimization process, information on the eigenspectrum of the matrix $A$ can be useful for constructing preconditioners. We explain in this section how to estimate these eigenpairs from the Lanczos and CG coefficients.

The eigenpairs of the matrix $A$ can be estimated by calculating the eigenpairs of the tridiagonal matrix $T_l$ generated in Lanczos Algorithm 2.3. Let us denote the eigenvalues of the matrix $T_l$ by $\theta_i$, $i = 1, \ldots, l$ and corresponding eigenvectors by $\overline{u}_i$, i.e.

$$T_l \overline{u}_i = \theta_i \overline{u}_i. \tag{2.33}$$

The pair given by $(\theta_i, u_i)$ where

$$u_i = V_l \overline{u}_i, \tag{2.34}$$

is known as *Ritz pair* and may be considered as approximates of the eigenpair of the matrix $A$. The eigenvalues $\theta_i$ are known as the *Ritz values* and the approximated eigenvectors $u_i$ are known as the *Ritz vectors* (Meurant, 2006, p. 8).

From the theoretical equivalence of Lanczos and CG, it is possible to construct the matrices $T_l$ and $V_l$ from the coefficients of the CG Algorithm 2.2, and hence to obtain approximate eigenvalues of the matrix $A$. In particular, denoting the entries of the tridiagonal matrix $T_l$ by $\eta_i$, $\gamma_i$ and $\eta_{i+1}$, i.e.

$$T_l = \begin{bmatrix} \gamma_1 & \eta_2 & & & & \\ \eta_2 & \gamma_2 & \eta_3 & & & \\ & \cdot & \cdot & \cdot & & \\ & & & \eta_{l-1} & \gamma_{l-1} & \eta_l \\ & & & & \eta_l & \gamma_l \end{bmatrix},$$

these entries can be obtained from Algorithm 2.2 using the relations (Saad, 1996, p.181–182)

$$\gamma_{i+1} = \begin{cases} \frac{1}{\alpha_i} & \text{if } i = 0, \\ \frac{1}{\alpha_i} + \frac{\beta_{i-1}}{\alpha_{i-1}} & \text{if } i > 0, \end{cases} \tag{2.35}$$

$$\eta_{i+1} = \frac{\sqrt{\beta_{i-1}}}{\alpha_{i-1}}. \tag{2.36}$$

The columns of the matrix $V_l$ can be generated by (Golub and Van Loan, 1996, p. 528)

$$v_{i+1} = (-1)^i \frac{r_i}{\|r_i\|_2}. \tag{2.37}$$

### 2.6.5 Preconditioning

It is well-known that a *preconditioner* is desirable when applying an iterative method to accelerate the convergence. A preconditioner typically transforms the linear system into one that has more favorable spectral properties, since these properties are related with the method convergence rate. A good preconditioner should be cheap to construct and apply (Benzi, 2002). With a good preconditioner the cost of constructing and applying the preconditioner should be less than the gain in computational cost (Benzi, 2002), (Barrett *et al.*, 1994, p.35). We recommend the references (Benzi, 2002; Saad and van der Vorst, 2000) for more information on a wide class of preconditioners.

When using CG or the Lanczos method, a suitable preconditioner $P$ can be used in the Preconditioned CG (PCG) and Preconditioned Lanczos (PLanczos) algorithms (Golub and Van Loan, 1996; Nour-Omid *et al.*, 1988; Axelsson, 1996; Chan *et al.*, 1999). These algorithms are given respectively by Algorithm 2.5 and Algorithm 2.6.

In Algorithm 2.6 the norm is defined as $\|x\|_P = \sqrt{x^T P x}$. The algorithm that generates the iterates of PCG from PLanczos is given by Algorithm 2.7 for further reference. Note that, in Algorithm 2.7, the vector $v_{i+1}$ is the $(i+1)$-th column of the matrix $V_l$ and $(T_l)_{i+1,i}$ refers to the $(i+1,i)$-th entry of the matrix $T_l$.

**Algorithm 2.5: PCG Algorithm**

$r_0 = b - As_0$
$z_0 = Pr_0$
$\rho_0 = r_0^T z_0$
$p_0 = z_0$
**for** $i = 0, 1, \ldots, l-1$ **do**
    $q_i = Ap_i$
    $\alpha_i = \rho_i / q_i^T p_i$
    $s_{i+1} = s_i + \alpha_i p_i$
    $r_{i+1} = r_i - \alpha_i q_i$
    $z_{i+1} = Pr_{i+1}$
    Check convergence and stop if desired accuracy is reached
    $\rho_{i+1} = r_{i+1}^T z_{i+1}$
    $\beta_{i+1} = \rho_{i+1} / \rho_i$
    $p_{i+1} = z_{i+1} + \beta_{i+1} p_i$
**end**

**Algorithm 2.6: PLanczos**

$r_0 = b - As_0$
$z_0 = Pr_0$
$\beta_0 = \|r_0\|_P$
$\beta_1 = 0$
$v_0 = 0$
$v_1 = r_0 / \beta_0$
$z_1 = z_0 / \beta_0$
$V_1 = v_1$
**for** $i = 1, \ldots, l$ **do**
    $w_i = Az_i - \beta_i v_{i-1}$
    $\alpha_i = w_i^T z_i$
    $w_i := w_i - \alpha_i v_i$
    $z_i := Pw_i$
    $\beta_{i+1} = \|w_i\|_P$
    $v_{i+1} = w_i / \beta_{i+1}$
    $z_{i+1} := z_i / \beta_{i+1}$
    $V_i := [V_i, v_{i+1}]$
    $(T_i)_{i,i} = \alpha_i$
    **if** $i > 1$ **then**
        $(T_i)_{i-1,i} = (T_i)_{i,i-1} = \beta_i$
    **end**
    Check convergence and stop if desired accuracy is reached
**end**
$y_l = T_l^{-1}(\beta_0 e_1)$
$s_l = s_0 + PV_l y_l$

**Algorithm 2.7: PLanczos2PCG**

Given $P, r_0, s_0, T_l, \beta_{l+1}, V_l, l$ from Algorithm 2.6
$(T_l)_{l+1,l} = \beta_{l+1}$
$z_0 = Pr_0$
$\beta_0 = \sqrt{r_0^T z_0}$
$p_0 = z_0$
$\rho = \beta_0^2$
**for** $i = 1, \ldots, l$ **do**
    $y_i = T_i^{-1}(\beta_0 e_1)$
    $r_i = -(T_l)_{i+1,i} v_{i+1} e_i^T y_i$
    $z_i = Pr_i$
    $\rho_i = r_i^T z_i$
    $\beta_i = \rho_i / \rho_{i-1}$
    $p_i = z_i + \beta_i p_{i-1}$
    $s_i = s_0 + PV_i y_i$
    $\alpha_i = \|(s_i - s_{i-1})\|_2 / \|p_{i-1}\|_2$
    $q_i = -(r_i - r_{i-1})/\alpha_i$
**end**

### 2.6.6 Preconditioning with Limited Memory Preconditioners

In this study we consider a class of Limited Memory Preconditioners (LMPs) that is designed for solving systems of equations with multiple right hand sides (Morales and Nocedal, 1999; Gratton, Sartenaer and Tshimanga, 2011). The general definition of the LMP is given as (Gratton, Sartenaer and Tshimanga, 2011, Definition 2.1):

**Definition 2.1** *Let $A$ and $M$ be symmetric and positive definite matrices of order $n$ and assume that $S$ is any $n$ by $\ell$ matrix of rank $\ell$, with $\ell \leq n$. The symmetric matrix*

$$P = [I_n - S(S^T AS)^{-1} S^T A] M [I_n - AS(S^T AS)^{-1} S^T] + S(S^T AS)^{-1} S^T \qquad (2.38)$$

*is called the Limited Memory Preconditioner (LMP).*

In the formula (2.38), the matrix $M$ is considered to cluster most eigenvalues at 1 with relatively few outliers and called *first-level preconditioner* (Gratton, Sartenaer and Tshimanga, 2011). Whereas, the preconditioner $P$ is called *second-level preconditioner* and is used to capture the directions which slows down the CG method (or the Lanczos method) that the first-level preconditioner $M$ is leaving out (Gratton, Sartenaer and Tshimanga, 2011). The main properties of the preconditioner $P$ can be summarized as (see (Gratton, Sartenaer and Tshimanga, 2011) and (Tshimanga, 2007))

- if $\ell = n$ in Definition 2.1, then $P = A^{-1}$,
- the LMP preconditioner is symmetric and positive definite,
- the LMP preconditioner clusters at least $\ell$ eigenvalues of the matrix $PA$ at 1 and does not expand the remaining part of the spectrum,

- the LMP can be applied as an operator in order to calculate matrix-vector multiplications,
- the LMP is cheap to apply.

Under the assumption that the columns of $S \in \mathbb{R}^{n\times\ell}$ are conjugate with respect to the matrix $A$, i.e. $s_i^T A s_j = 0$ for $i \neq j$, and setting $M = P_0$ the preconditioner (2.38) results in a simplified formula (Gratton, Sartenaer and Tshimanga, 2011) given by

$$P_l = \left(I_n - \sum_{i=1}^{l} \frac{s_i s_i^T}{s_i^T A s_i} A\right) P_0 \left(I_n - \sum_{i=1}^{l} \frac{A s_i s_i^T}{s_i^T A s_i}\right) + \sum_{i=0}^{l} \frac{s_i s_i^T}{s_i^T A s_i}. \tag{2.39}$$

From the formula (2.39) or (2.38) three particular instances of the LMPs: quasi-Newton LMP, spectral LMP and Ritz LMP are derived by using the relevant information obtained from a CG (or Lanczos) run while solving a linear system involving $A$.

The quasi-Newton LMP is constructed by using $l$ successive (linearly independent) search directions, $p_i$, $i = 1, \ldots, l$ obtained during a CG (or Lanczos) run. Replacing $s_i$ by $p_i$ in the formula (2.39) results in the quasi-Newton preconditioner inductively given as (Gratton, Sartenaer and Tshimanga, 2011)

$$P_l = \left(I_n - \frac{p_l p_l^T A}{p_l^T A p_l}\right) P_{l-1} \left(I_n - \frac{A p_l p_l^T}{p_l^T A p_l}\right) + \frac{p_l p_l^T}{p_l^T A p_l}. \tag{2.40}$$

This preconditioner amounts to the preconditioner proposed by Morales and Nocedal (1999).

The Ritz LMP uses the linearly independent Ritz vectors $u_i$, $i = 1, \ldots, l$ and the Ritz values $\theta_i$ generated during a CG (or Lanczos) run. Choosing $S = [u_1, \ldots, u_l]$ and $M = I_n$ in the formula (2.38) leads to the Ritz LMP matrix given as (Gratton, Sartenaer and Tshimanga, 2011)

$$P_l = I_n + S(\Theta^{-1} - I_l)S^T - S\omega v_{l+1}^T - v_{l+1}\omega^T S^T + S\omega\omega^T S^T \tag{2.41}$$

where $\Theta = diag(\theta_i)$ and $\omega = (\omega_1, \ldots, \omega_l)^T$, with

$$\omega_i = \frac{(e_l^T \overline{u}_i)\beta_{l+1}}{\theta_i}, \quad \text{for } i = 1, \ldots, l. \tag{2.42}$$

The vector $\overline{u}_i$ in (2.42) denotes the eigenvector of the tridiagonal matrix $T_l$ and $\beta_{l+1}$ is defined in Lanczos Algorithm 2.3.

The spectral LMP is generated from the independent unit eigenvectors $v_i$ of $A$ and corresponding eigenvalues $\lambda_i$. Choosing $S = [v_1, \ldots, v_l]$ and $M = I_n$ in the formula (2.38) leads to the spectral LMP matrix given as (Gratton, Sartenaer and Tshimanga, 2011)

$$P = I_n - S(\Lambda^{-1} - I_l)S^T \tag{2.43}$$

where $\Lambda = diag(\lambda_i)$. The spectral-LMP amounts to the well-known preconditioner in operational data assimilation systems proposed by Fisher (1998).

It is shown in Tshimanga (2007) that the spectral LMP generated by Ritz information behaves like the Ritz LMP when the the error on the Ritz pair is small. It is also noted in Gratton, Sartenaer and Tshimanga (2011, Theorem 4.6) that when all the available information (search directions and Ritz vectors) are used while constructing the quasi-Newton LMP and the Ritz LMP, these LMPs span the same space and hence, they are mathematically equivalent. For more details on comparisons of the mentioned preconditioners, we recommend the references (Gratton *et al.*, 2011; Tshimanga *et al.*, 2008; Tshimanga, 2007).

We summarized above the LMPs and their effects when used for preconditioning systems of equations with multiple right hand sides. It is also important to understand how to use LMPs in the context of nonlinear least-squares problem. Since the local quadratic approximations of the nonlinear least-squares problem change, the solution of the problem with an iterative method (like the Gauss-Newton method) requires solving a sequence of slightly varying systems. In this case, the LMP formula (2.38) would still be applicable provided that the new matrix $A$ is used. Note that, for this preconditioner to be considered effective, the cost of constructing and applying the preconditioner should be less than the gain obtained.

### 2.6.7 CG relation with LSQR and Generalized LSQR

In the previous sections, we focus on solving the linearized system (2.16)

$$(B^{-1} + H^T R^{-1} H)\, s = B^{-1}(x_c - x) - H^T R^{-1} d, \tag{2.44}$$

(where we have dropped the index $k$ for simplicity) by using a CG or Lanczos method. This linear system represents the normal equations

$$(J^T W J)s = -J^T W u$$

where the matrices $J$ and $W$ and the vector $u$ are defined in Section 2.3. As mentioned before this normal equation is associated with the linear least-squares problem (2.13):

$$\min_s \frac{1}{2} \|Js + u\|_W^2 \tag{2.45}$$

or equivalently given as

$$\min_s \frac{1}{2} \|Hs + d\|_{R^{-1}}^2 + \frac{1}{2} \|s + x - x_c\|_{B^{-1}}^2 . \tag{2.46}$$

The overdetermined least-squares problem (2.45) can be alternatively solved by using the well-known LSQR method (Paige and Saunders, 1982; Saunders, 1995), which is based on the Lanczos bidiagonalization procedure (Golub and Van Loan, 1996, p. 495). When considering the equivalent regularized least-squares problem (2.46), the LSQR method could not be applied directly since it requires a special structure. Defining a variable $\delta\widetilde{x} = B^{-1/2}(s + x - x_c)$ and substituting it into (2.46), the problem can also

be stated in this special structure as

$$\min_{\delta\widetilde{x}} \ \frac{1}{2}\left\| \begin{bmatrix} \widetilde{H} \\ I_n \end{bmatrix} \delta\widetilde{x} - \begin{bmatrix} \widetilde{c} \\ 0 \end{bmatrix} \right\|_2^2, \tag{2.47}$$

where $\widetilde{H} = R^{-1/2} H B^{1/2}$ is a $m$-by-$n$ matrix with $m < n$ and $\widetilde{c} = R^{-1/2} H(x - x_c) - R^{-1/2} d$. It is noted in (Saunders, 1995, Result 8) that since $\widetilde{H}^T \widetilde{H} + I_n$ is symmetric and positive definite, the LSQR method generates mathematically equivalent iterates $\delta\widetilde{x}_i$ to those of CG on the linear system

$$(\widetilde{H}^T \widetilde{H} + I_n)\ \delta\widetilde{x} = \widetilde{H}^T \widetilde{c}. \tag{2.48}$$

This linear system can be written as:

$$(B^{1/2} H^T R^{-1} H B^{1/2} + I)\ \delta\widetilde{x} = B^{1/2} H^T R^{-1}(H(x - x_c) - d). \tag{2.49}$$

Note that, at convergence the solution of the linear system (2.44) and the solution of the linear system (2.49) are related by the expression $s_k = x_c - x_k + B^{1/2}\delta\widetilde{x}_k$ (where $k$ is the index for the outer loop of the Gauss-Newton method).

The problem structure for solving a regularized least squares problem with LSQR can be generalized by using the a non-standard norm (Arioli and Orban, 2012). This problem can then be solved by using the generalized LSQR (G-LSQR) method proposed by Arioli and Orban (2012). G-LSQR generates mathematically equivalent iterates $\delta x_i$ to those of applying PCG on the linear system

$$(B^{-1} + H^T R^{-1} H)\ \delta x = H^T R^{-1}(H(x - x_c) - d), \tag{2.50}$$

with the preconditioner matrix $B$. Accordingly, it also generates the same iterates as those of CG applied on the linear system (2.49) with $\delta x_i = B^{1/2}\delta\widetilde{x}_i$. Note also that, at convergence the solution of the linear system (2.44) and the solution of the linear system (2.50) are related with $s_k = x_c - x_k + \delta x_k$.

LSQR is numerically more reliable than CG when the system matrix is ill-conditioned and many iterations are performed (Björck, 1996, p. 288, 307-309), (Paige and Saunders, 1982). However it requires extra storage which may not be afordable for large scale problems. LSQR can be preferable to CG when memory is not a problem and high accuracy is expected.

## 2.7 The dual problem

The original quadratic subproblem (2.45), considered the *primal problem*, can be converted into an alternative problem called the *dual problem*. The dual problem can be used in practice for sensitivity analysis, or to design algorithms for solving the primal problem (Nocedal and Wright, 2006, p. 343).

Let us rewrite the convex quadratic subproblem (2.45) as a constrained problem

$$\min_{r,s} \frac{1}{2}\|r\|_W^2 \quad \text{subject to} \quad Js + u - r = 0 \tag{2.51}$$

which can, in turn, be solved using duality theory. The *dual objective* function for the *primal objective* function in (2.51) can be defined as (Nocedal and Wright, 2006, p.343-349)

$$q(\lambda) = \inf_{r,s} \mathcal{L}(r, s, \lambda) \stackrel{\text{def}}{=} \inf_{r,s} \frac{1}{2} r^T W r - \lambda^T (Js + u - r), \tag{2.52}$$

where $q : \mathbb{R}^{m+n} \to \mathbb{R}$ and $\lambda \in \mathbb{R}^{m+n}$ is a Lagrange multiplier vector. The infimum is achieved when

$$\nabla_r \mathcal{L}(r, s, \lambda) = Wr + \lambda = 0,$$
$$\nabla_s \mathcal{L}(r, s, \lambda) = J^T \lambda = 0,$$

which yields that $r = -W^{-1}\lambda$. We may therefore substitute $r$ and use the equation $J^T\lambda = 0$ in the expression (2.52) and obtain the dual objective explicitly as follows:

$$q(\lambda) = -\frac{1}{2}\lambda^T W^{-1}\lambda - \lambda^T u.$$

Then the *dual problem* for the *constrained primal problem* (2.51) is given by

$$\max_{\lambda} \; q(\lambda). \tag{2.53}$$

Note that the cost function of the dual problem (2.53) is quadratic and the dual problem has no constraints. This dual problem is defined on a space of the same dimension as the primal problem. We will see in Chapter 4 that there may be different dual approaches based on the structure of the primal problem, which allows us to define the dual problem on a space of smaller dimension than the primal problem, that will provide computationally more efficient algorithms.

The relation between the constrained primal and dual problem can be given from the *weak duality* theorem (Nocedal and Wright, 2006, Theorem 12.11), i.e.

$$\boxed{\text{for any } \underline{r} \text{ feasible for (2.51) and any } \underline{\lambda} \in \mathbb{R}^{m+n}, \text{ we have } q(\underline{\lambda}) \leq f(\underline{r})}$$

where, since $f$ does not depend on $s$ in this particular case, $f(\underline{r}) = \frac{1}{2}\|\underline{r}\|_W^2$. From this theorem one can observe that the optimal value, $\lambda^*$, of the dual problem (2.53) gives a lower bound on the optimal value, $r^*$, for the constrained primal problem (2.51) (Nocedal and Wright, 2006, p. 345).

If $q(\lambda^*) = f(r^*)$, it is said that there is no *duality gap* and if $q(\lambda^*) < f(r^*)$ it is said that there is a *duality gap* (Bertsekas, 1995, Section 5.1.2). Since the constrained primal problem has a convex cost function with a linear constraint, from *strong duality theorem* (Bertsekas, 1995, Proposition 5.1.2), there is no duality gap between the constrained primal and dual problems given by (2.51) and (2.53) respectively.

We recommend the references Mangasarian (1969); Bertsekas (1995); Rockafellar (1970, 1976); Boyd and Vandenberghe (2004) for more information on duality.

# CHAPTER 3

## Solving the subproblem with the conjugate-gradient method

This chapter considers applying the *preconditioned conjugate-gradients* to a sequence of symmetric and positive definite linear systems of the form

$$A_k s = b_k,$$

where $A_k = B^{-1} + H_k^T R^{-1} H_k$ and $b_k$ are defined in (2.16), and where preconditioning is achieved by using the Limited Memory Preconditioners (LMPs) described in Section 2.6.6. These linear systems arise from a sequence of linear least-squares subproblems (2.13), of a Gauss-Newton (GN) algorithm with each member of the sequence being the least-squares problem (2.3). We recall that the matrices $B$ and $R$ appearing in the linear system are the symmetric and positive definite error-covariance matrices.

The LMP for the system matrix $A_k$ is obtained by using directions generated during solving the linear system at $(k-1)$-th outer loop of the GN algorithm. This way of preconditioning improves a preconditioner at each outer loop and is known as the *warm-start preconditioning*. We first focus on warm-start preconditioning of a sequence of linear systems with multiple right-hand sides,

$$As = b_k,$$

by using directions generated during solving the previous linear system. Note that, each linear system is part of a sequence, which is supposed to produce a convergent sequence of solutions.

We present different ways of preconditioning the linear system $As = b_k$ with a warm-start preconditioner. The first approach is to use a split preconditioning with a standard conjugate gradient method (CG Algorithm 2.2). This approach requires the square-root factorization of the preconditioner, i.e. $P = FF^T$. In a particular case when $P = B$, this approach avoids the matrix-vector products with $B^{-1}$ under the assumption that $B^{1/2}$ is available. The second alternative is to use PCG Algorithm 2.5

where the preconditioner $P$ can be directly used without requiring any square-root factorization.

Under the assumption that the $B^{-1}P$ operator is available, which is possible as shown in Section 3.1, we propose a variant of conjugate-gradients as a third alternative which uses *right-symmetric-preconditioning* that both avoids the matrix-vector products with $B^{-1}$ and the square-root factorization of the matrix $B$. This alternative can be preferable especially for large scale problems since calculating the inverse or a factorization of $B$, as required by the two first approaches, may not be affordable in terms of computational cost.

Starting from these three types of preconditioning, we present three algorithms solving a sequence of linear systems with multiple right hand-sides in which the inner minimization is performed with the corresponding conjugate gradients approach using the general formulations of the corresponding LMPs. These algorithms generate a mathematically equivalent sequence of iterates $\{s_k\}$. We continue the chapter by discussing how to use particular LMPs such as the quasi-Newton LMP, Ritz LMP and spectral LMP, for each approach.

We later analyze three variants of conjugate gradients when used for solving a sequence of linear systems in the form of $A_k s = b_k$. We conclude the chapter by providing inexpensive formulae to compute the quadratic cost function (2.13) and the norm of its gradient, which are useful quantities to stop inner iterations in a nonlinear solver.

## 3.1 Preconditioning the conjugate-gradient method with LMPs

In this section we are dealing with applying conjugate gradients to the systems of equations with multiple right hand sides,

$$(B^{-1} + H^T R^{-1} H)\, s = B^{-1}(x_c - x_k) - H^T R^{-1} d_k, \tag{3.1}$$

of the form of

$$As = b_k,$$

with a *warm-start LMP*. Each linear system is part of a sequence and is not isolated. It is assumed that this sequence produces a convergent sequence of solutions.

We next present different approaches on preconditioning of the linear system (3.1) with a symmetric and positive definite LMP defined by (2.38),

$$\boxed{P = [I_n - S(S^T AS)^{-1} S^T A] M [I_n - AS(S^T AS)^{-1} S^T] + S(S^T AS)^{-1} S^T,}$$

in the sense that *each approach requires different forms of the preconditioner and may avoid expensive matrix-vector products (when the linear system is of large-scale)*. We recall that the preconditioner $P$ is often called *second-level preconditioner* for the matrix $A$ and used to capture the directions which slow down the CG method that the *first-level preconditioner* $M$ is leaving out.

While presenting different approaches, we first explain the approach over a single linear system. We later explain how to use this method for solving a convergent sequence of linear systems. In this case using a warm-start technique improves the preconditioner along the sequence.

We start with the approach that uses a split preconditioning with the standard conjugate gradient (Algorithm 2.2) as explained below.

### Preconditioning with CG Algorithm 2.2

Solving the linear system (3.1) with a LMP $P$ formulated by (2.38) by using CG Algorithm 2.2 requires the factored form of the LMP. We recall Theorem 3.9 of Gratton, Sartenaer and Tshimanga (2011) that provides this factorization.

Let $M = LL^T$ with $L^T \in \mathbb{R}^{n\times n}$, and $S^TAS = G^TG$ with $G \in \mathbb{R}^{\ell\times\ell}$. Then the LMP matrix of $P$ can be factored as $P = FF^T$ with $F$ given by

$$\boxed{F = L - SG^{-1}G^{-T}S^TAL + SG^{-1}X^{-T}S^TL^{-T}} \tag{3.2}$$

where $S^TL^{-T}L^{-1}S = X^TX$, with $X \in \mathbb{R}^{\ell\times\ell}$, is any factorization of $S^TL^{-T}L^{-1}S$.

Using this factored form of the LMP $P$, Algorithm 2.2 solves the preconditioned system

$$F^TAF\ \widetilde{s} = F^Tb_k, \tag{3.3}$$

for the preconditioned variable $\widetilde{s}$. The approximate solution $s$ is then obtained from the relation $s = F\ \widetilde{s}$.

Let us now explain in detail how to solve a convergent sequence of linear systems (3.1) by using CG Algorithm 2.2 with a warm-start LMP formulated by (3.2). Assume that we choose an initial guess $x_1$. Assume also that the factorization of the initial preconditioner

$$\boxed{P_0 = F_0F_0^T} \tag{3.4}$$

is available. Then for $k = 1$, CG Algorithm solves the preconditioned system

$$\underbrace{F_0^TAF_0}_{\widetilde{A}_1}\ \widetilde{s} = \underbrace{F_0^Tb_1}_{\widetilde{b}_1}, \tag{3.5}$$

for the variable $\widetilde{s}$ and recovers the approximate solution from the relation $s_1 = F_0\ \widetilde{s}_1$. This approximate solution is then used to update the current iterate from $x_2 = x_1 + s_1$ as explained in Section 2.5.

While performing the CG algorithm on the linear system (3.5), the information can be stored to obtain the factored form of the LMP for the system matrix $\widetilde{A}_1$, i.e. $Y_1 = F_1F_1^T$, with $F_1$ is constructed from (3.2). Note that while constructing $F_1$ from (3.2), $M_0$ is taken as an identity matrix of order $n$ since $P_0 = F_0F_0^T$ is already acting as a first-level preconditioner.

Using the preconditioner $F_1$ for $k = 2$, CG Algorithm solves the preconditioned

system

$$\underbrace{F_1^T(F_0^T A F_0)F_1}_{\widetilde{A}_2}\ \widetilde{s} = \underbrace{F_1^T F_0^T b_2}_{\widetilde{b}_2}, \tag{3.6}$$

for the variable $\widetilde{s}$ and recover the approximate solution from $s_2 = F_0 F_1\, \widetilde{s}_2$. This approximate solution is then used to update the current iterate from $x_3 = x_2 + s_2$.

While performing the CG algorithm on the linear system (3.6) the information can be stored to obtain the factored form of the LMP for the system matrix $\widetilde{A}_2$, i.e. $Y_2 = F_2 F_2^T$, with $F_2$ is constructed from (3.2). Note that while constructing $F_2$ from (3.2), $M_1$ is taken again as an identity matrix of order $n$ since in this case the matrix $F_1 F_0 F_0^T F_1^T$ can be considered as an implicit first-level preconditioner.

The same strategy can be carried over for later steps. Using the preconditioner $F_k$, for the $(k+1)$-th linear system CG Algorithm solves the preconditioned system

$$\underbrace{F_k^T(F_{k-1}^T \dots F_1^T F_0^T A F_0 F_1 \dots F_{k-1})F_k}_{\widetilde{A}_{k+1}}\ \widetilde{s} = \underbrace{F_k^T F_{k-1}^T \dots F_1^T F_0^T b_{k+1}}_{\widetilde{b}_{k+1}}, \tag{3.7}$$

for the variable $\widetilde{s}$ and recover the approximate solution from $s_{k+1} = F_0 F_1 \dots F_{k-1} F_k \widetilde{s}_{k+1}$. This approximate solution is then used to update the current iterate from $x_{k+2} = x_{k+1} + s_{k+1}$. Algorithm 3.1 outlines this implementation.

---

**Algorithm 3.1: CG for solving a convergent sequence of linear systems with multiple right-hand sides**

---

**1 Initialization:** Choose an initial preconditioner $P_0$ and factorize the preconditioner such that $P_0 = F_0 F_0^T$. Choose an initial vector $x_1$.

**2 Calculate preconditioned right hand side $\tilde{b}_k$:**

$$\tilde{b}_k = F_{k-1}^T \dots F_1^T F_0^T [B^{-1}(x_c - x_k) - H^T R^{-1} d_k]$$

**3 Perform inner loop**: Solve the preconditioned linear system

$$\widetilde{A}_k\, \widetilde{s} = \widetilde{b}_k,$$

with CG Algorithm 2.2. During CG extract and save the information to precondition the next linear system with $F_k$ where $F_k$ is constructed from (3.2).

**4 Recover the approximate solution:**

$$s_k = F_0 F_1 \dots F_{k-1} \widetilde{s}_k$$

**5 Update the iterate:**

$$x_{k+1} = x_k + s_k$$

**6** Increment $k$ by 1 and go to step 2.

---

*This algorithm requires the factored form of the LMP, and assumes that the factorization of the initial preconditioner, i.e.* $P_0 = F_0 F_0^T$*, is available.*

When solving the $k$-th linear system in Algorithm 3.1, CG Algorithm 2.2 requires a single matrix-vector product with each of the operators $B^{-1}, H, H^T, R^{-1}$, $F_i$ and $F_i^T$ with $i = 0, \dots k-1$. Note that the cost of the matrix-vector products with the limited memory preconditioners $F_i$ and $F_i^T$, for $i > 0$ is not expensive (Tshimanga, 2007).

When there is no initial preconditioner available, the initial preconditioner can be chosen as $P_0 = B$ since it is known that this preconditioning clusters most eigenvalues at 1. Assuming that $B = B^{1/2}B^{1/2}$ is available, it can be easily seen from (3.7) that matrix vector products with $B^{-1}$ can be avoided by choosing $F_0 = B^{1/2}$, i.e. $F_0^T A F_0 = I_n + B^{1/2} H^T R^{-1} H B^{1/2}$. This can reduce the computational cost since the matrix $B$ is not diagonal in general and therefore the matrix-vector products with $B^{-1}$ may be expensive especially for large-scale problems. As a result, in this special case, when solving the $k$-th linear system in Algorithm 3.1, the CG Algorithm requires matrix-vector products with each of the operators $B^{1/2}, H, H^T, R^{-1}$, $F_i$, and $F_i^T$ with $i = 1, \dots k-1$.

We continue this section by explaining another way of preconditioning a sequence of linear systems (3.1) with PCG Algorithm 2.5.

### Preconditioning with PCG Algorithm 2.5

The main difference in solving the linear system (3.1) with PCG Algorithm 2.5 rather than CG Algorithm 2.2 with a preconditioner is that *PCG Algorithm 2.5 does not require the factored form of the preconditioner*. This can be algorithmically significant when the preconditioner factorization is not available or expensive to compute.

Therefore, another way of preconditioning can be performed by applying Algorithm 2.5 to linear system (3.1) of the form of $As = b_k$ with a LMP (2.38).

Let us now explain in detail how to solve a convergent sequence of linear systems (3.1) by using PCG Algorithm 2.5 with a warm-start LMP formulated by (2.38). Assume that we choose an initial guess $x_1$. Then for $k = 1$, PCG Algorithm solves the system

$$A\, s = b_1, \tag{3.8}$$

for the variable $s$ by using an initial preconditioner $P_0$. The current iterate is updated by using the approximate solution $s_1$ from $x_2 = x_1 + s_1$.

While performing the PCG algorithm on the linear system (3.8), the information can be stored to obtain the LMP $P_1$ for the system matrix $A$ by using the formula (2.38). This preconditioner can then be used to precondition the second linear system. Note that while constructing $P_1$ from (2.38), the first level preconditioner $M_0$ is taken as the initial preconditioner, i.e. $M_0 = P_0$.

Using the preconditioner $P_1$ for $k = 2$, PCG Algorithm solves the linear system

$$A\, s = b_2, \tag{3.9}$$

for the variable $s$. The current iterate is updated by using the approximate solution $s_2$

from $x_3 = x_2 + s_2$.

While performing the PCG algorithm on the linear system (3.9) the information can be stored to obtain the LMP $P_2$ for the system matrix $A$ by using the formula (2.38). Note that while constructing $P_2$ from (2.38), the first level preconditioner $M_1$ is taken as the preconditioner used for $k = 2$, i.e. $M_1 = P_1$.

The same strategy can be carried over for later steps. Using the preconditioner $P_k$, for the $(k+1)$-th linear system PCG Algorithm solves the preconditioned system

$$A\, s = b_{k+1}, \tag{3.10}$$

for the variable $s$. The current iterate is updated by using the approximate solution $s_{k+1}$ from $x_{k+2} = x_{k+1} + s_{k+1}$. Algorithm 3.2 outlines this implementation.

**Algorithm 3.2: PCG for solving a convergent sequence of linear systems with multiple right-hand sides**

1 **Initialization:** Choose an initial preconditioner $P_0$, and an initial vector $x_1$.

2 **Calculate right hand side** $b_k$**:**

$$b_k = B^{-1}(x_c - x_k) - H^T R^{-1} d_k$$

3 **Perform inner loop**: Using the preconditioner $P_{k-1}$ solve the linear system:

$$As = b_k$$

with PCG Algorithm 2.5. During PCG extract and save the information to precondition the next linear system with an LMP $P_k$ formulated by (2.38).

4 **Update the iterate:**

$$x_{k+1} = x_k + s_k$$

5 Increment $k$ by 1 and go to step 2.

Note that, *this algorithm does not require the factorization of the preconditioner.*

When solving the $k$-th linear system in Algorithm 3.2, PCG Algorithm 2.5 requires a single matrix-vector product with each of the operators $B^{-1}, H, H^T, R^{-1}$ and $P_{k-1}$. Note that the cost of matrix-vector products with the limited memory preconditioner $P_k$, $k > 0$ is not expensive (Tshimanga, 2007; Gratton, Sartenaer and Tshimanga, 2011).

As with Algorithm 3.1, when there is no initial preconditioner available, the initial preconditioner can be chosen as $P_0 = B$. In this special case, for the first linear system of Algorithm 3.2 during PCG the matrix vector products with $B^{-1}$ can be avoided (Derber and Rosati, 1989) as explained below.

During PCG Algorithm 2.5, the system matrix $A = B^{-1} + H^T R^{-1} H$ is applied to the search direction $p_i$ (see line 6 of Algorithm 2.5). Let us define an additional vector $\underline{p}_i$ defined by $\underline{p}_i = B^{-1} p_i$. Taking $P = B$ and multiplying line 14 of Algorithm 2.5 by

$B^{-1}$, we obtain that

$$\underline{p}_i = \begin{cases} r_0 & \text{if } i = 0, \\ r_i + \beta_i \underline{p}_{i-1} & \text{if } i > 0, \end{cases} \tag{3.11}$$

where we used the fact that $r_i = P^{-1} z_i = B^{-1} z_i$, for $i > 0$. Therefore, the matrix-vector products with $B^{-1}$ can be avoided by introducing the vector $\underline{p}_i$. As a result, in this special case ($P_0 = B$) for the first linear system of Algorithm 3.2, the PCG Algorithm requires a single matrix-vector product with each of the operators $H, H^T, R^{-1}$ and $B$.

Note that, when the same preconditioning $P_k = B$ is applied for the linear systems in sequence within Algorithm 3.2, matrix-vector products with $B^{-1}$ can also be avoided. However, using the LMP formulated by (2.38) does not allow us to use the vector $\underline{p}_i$ for the matrix-vector products of $B^{-1} p_i$ since $r_i = P^{-1} z_i \neq B^{-1} z_i$.

Is there any other way of preconditioning the linear system (3.1) with the LMP given by (2.38) that avoids matrix-vector products with $B^{-1}$ as well as the factorization of the preconditioner? In the next part, we answer this question.

### Preconditioning with PCG Inverse Free Algorithm

We can apply the symmetric and positive definite matrix $P$ formulated by (2.38) as a right symmetric preconditioner while solving the linear system (3.1). This leads to solving the linear system

$$(B^{-1} P + H^T R^{-1} H P)\, \underline{v} = B^{-1}(x_c - x_k) - H^T R^{-1} d_k, \tag{3.12}$$

equipped with the $P$-inner product for the variable $\underline{v}$. The approximate solution $s$ is then obtained from $s = P\underline{v}$.

Let us now assume that *the matrix $B^{-1}P$ is available* and define this matrix by $C = B^{-1}P$. Substituting the relation $P = BC$ into the linear system (3.12) gives that

$$(C + H^T R^{-1} H B C)\, \underline{v} = B^{-1}(x_c - x_k) - H^T R^{-1} d_k, \tag{3.13}$$

which suggests solving the linear system

$$(I_n + H^T R^{-1} H B)\, \underline{s} = B^{-1}(x_c - x_k) - H^T R^{-1} d_k, \tag{3.14}$$

equipped with the $B$-inner product for the variable $\underline{s}$ by using the preconditioner matrix $C$. Note that, the system matrix in (3.14) is symmetric with respect to the $B$-inner product. The approximate solution is recovered from $s = B\underline{s}$. This way of minimization with conjugate gradients results in PCG Inverse Free (PCGIF) Algorithm 3.3 that avoids matrix-vector multiplication with $B^{-1}$.

It can be easily shown that the vectors defined in PCG Algorithm 2.5 when applied to the linear system (3.1) with a preconditioner $P$ are related with the vectors defined

**Algorithm 3.3: PCGIF Algorithm (version 1)**

1 $\underline{r}_0 = b - AB\underline{s}_0$
2 $\underline{z}_0 = C\underline{r}_0$
3 $\underline{p}_0 = \underline{z}_0$
4 $\rho_0 = \underline{r}_0^T B\underline{z}_0$
5 **for** $i = 0, 1, \ldots, l-1$ **do**
6 $\quad \underline{q}_i = (\underline{p}_i + H^T R^{-1} H B\underline{p}_i)$
7 $\quad \alpha_i = \rho_i / \underline{q}_i^T B\underline{p}_i$
8 $\quad \underline{s}_{i+1} = \underline{s}_i + \alpha_i \underline{p}_i$
9 $\quad \underline{r}_{i+1} = \underline{r}_i - \alpha_i \underline{q}_i$
10 $\quad \underline{z}_{i+1} = C\underline{r}_{i+1}$
11 $\quad$ Check convergence and stop if desired accuracy is reached
12 $\quad \rho_{i+1} = \underline{r}_{i+1}^T B\underline{z}_{i+1}$
13 $\quad \beta_{i+1} = \rho_{i+1} / \rho_{i+1}$
14 $\quad \underline{p}_{i+1} = \underline{z}_{i+1} + \beta_{i+1}\underline{p}_i$
15 **end**

in PCGIF Algorithm with the following relations

$$\left.\begin{array}{rcl} r_i &=& \underline{r}_i \\ z_i &=& B\underline{z}_i \\ q_i &=& \underline{q}_i \\ s_i &=& B\underline{s}_i \\ p_i &=& B\underline{p}_i \end{array}\right\} \qquad (i \geq 0) \tag{3.15}$$

under the assumption that $B^{-1}P$ is available. Using the relation $p_i = B\underline{p}_i$ and $A$-conjugacy property of $p_i$ we also have the property

$$\underline{p}_i^T BAB\underline{p}_j = 0 \text{ for } i \neq j, \tag{3.16}$$

for further reference.

Algorithm 3.3 can be transformed into a cheaper form by introducing additional vectors, reducing its cost per loop to a single matrix-vector product with $B$. More precisely, consider $p$ and $z$ defined by

$$z_i = B\underline{z}_i \quad \text{and} \quad p_i = B\underline{p}_i. \tag{3.17}$$

for $i \geq 0$. If we multiply lines 3 and 14 of Algorithm 3.3 by the matrix $B$, we get

$$p_i = \begin{cases} z_0 & \text{if } i = 0, \\ z_i + \beta_i p_{i-1} & \text{if } i > 0. \end{cases} \tag{3.18}$$

Substituting these definitions into Algorithm 3.3, yields the final version of PCGIF Algorithm (Algorithm 3.4).

**Algorithm 3.4: PCGIF Algorithm**

$\underline{r}_0 = b - AB\underline{s}_0$
$\underline{z}_0 = C\underline{r}_0$
$\underline{p}_0 = \underline{z}_0$
$z_0 = B\underline{z}_0$
$\rho_0 = \underline{r}_0^T z_0$
$p_0 = z_0$
**for** $i = 0, 1, \ldots, l-1$ **do**
  $\underline{q}_i = (\underline{p}_i + H^T R^{-1} H p_i)$
  $\alpha_i = \rho_i / \underline{q}_i^T p_i$
  $\underline{s}_{i+1} = \underline{s}_i + \alpha_i \underline{p}_i$
  $\underline{r}_{i+1} = \underline{r}_i - \alpha_i \underline{q}_i$
  $\underline{z}_{i+1} = C\underline{r}_{i+1}$
  $z_{i+1} = B\underline{z}_{i+1}$
  Check convergence and stop if desired accuracy is reached
  $\rho_{i+1} = \underline{r}_{i+1}^T z_{i+1}$
  $\beta_{i+1} = \rho_{i+1} / \rho_{i+1}$
  $\underline{p}_{i+1} = \underline{z}_{i+1} + \beta_{i+1} \underline{p}_i$
  $p_{i+1} = z_{i+1} + \beta_{i+1} p_i$
**end**

Now, the question is whether it is possible to obtain the matrix $C = B^{-1}P$ as the form of a limited memory type (or in other words find a limited memory type preconditioner $C$ such that $P = BC$) where $P$ is an LMP formulated by (2.38). The following lemma provides a general formula that allows one to obtain such a matrix $C$.

**Lemma 3.1** *Let $B\underline{A}$ and $B\underline{M}$ be symmetric and positive definite matrices of order $n$. Assume also that $\underline{S}$ is any $n$ by $l$ matrix of rank $l$, with $l \leq n$. Then the $n$-by-$n$ matrix given by*

$$C = [I_n - \underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B\underline{A}]\,\underline{M}\,[I_n - \underline{A}\,\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B] + \underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B \tag{3.19}$$

*is symmetric with respect to the $B$-inner product and $BC$ is positive definite.*

*Suppose also that $M = B\underline{M}$. If we denote $S = B\underline{S}$ and $\underline{A} = AB$ then the LMP matrix $P$ given by (2.38) and the matrix $C$ defined by (3.19) satisfy $P = BC$.*

**Proof.** Multiplying $C$ on the left by $B$ gives that

$$BC = [B - B\underline{S}(\underline{S}^T B\underline{A}\underline{S})^{-1}\underline{S}^T B\underline{A}]\underline{M}[I_n - \underline{A}\underline{S}(\underline{S}^T B\underline{A}\underline{S})^{-1}\underline{S}^T B] + B\underline{S}(\underline{S}^T B\underline{A}\underline{S})^{-1}\underline{S}^T B. \tag{3.20}$$

Using the assumption that $B\underline{A}$ is symmetric, we have

$$\begin{aligned} BC =& [I_n - B\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T \underline{A}^T]\; B\underline{M}\;[I_n - \underline{A}\,\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B] \\ &+ B\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B. \end{aligned}$$

Similarly using the symmetry of $B\underline{M}$ and $B\underline{A}$ we obtain that

$$\begin{aligned} BC &= [I_n - B\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T\underline{A}^T]\,\underline{M}^T\,[B - B\underline{A}\,\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B] \\ &\quad + B\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B, \\ BC &= [I_n - B\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T\underline{A}^T]\,\underline{M}^T\,[I_n - \underline{A}^T B\,\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T]B \\ &\quad + B\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B, \\ &= C^T B, \end{aligned}$$

which proves symmetry of $C$ with respect to the $B$-inner product.

Since $B\underline{A}$ is symmetric and positive definite and $\underline{S}$ has full column rank, the matrix $\underline{S}^T B\underline{A}\,\underline{S}$ is also symmetric and positive definite (Golub and Van Loan, 1996, Theorem 4.2.1). Therefore, the matrix $\underline{S}^T B\underline{A}\,\underline{S}$ has an inverse and it can be factored such that $\underline{S}^T B\underline{A}\,\underline{S} = (\underline{S}^T B\underline{A}\,\underline{S})^{1/2}(\underline{S}^T B\underline{A}\,\underline{S})^{1/2}$. Defining the matrix $V = \underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1/2}$ and substituting it in equation (3.20), we obtain that

$$BC = [B - BVV^T B\underline{A}]\,\underline{M}\,[I_n - \underline{A}VV^T B] + BVV^T B. \tag{3.21}$$

Let us define symmetric and positive definite matrices as $M = B\underline{M}$ and $A = \underline{A}B^{-1}$, and the matrix $W = BV$ with $W^T AW = I_l$, $I_l$ is the identity matrix of order $l$. Substituting these matrices in (3.21) gives that

$$\begin{aligned} BC &= [I_n - BVV^T BA]\,M\,[I_n - ABVV^T B] + BVV^T B \\ &= [I_n - WW^T A]\,M\,[I_n - AWW^T] + WW^T. \end{aligned} \tag{3.22}$$

In Lemma 3.3 of the reference (Gratton, Sartenaer and Tshimanga, 2011), it is proved that a matrix having the form of (3.22) is positive definite under the assumptions that $A$ and $M$ are symmetric and positive definite and $W^T AW = I_l$. Applying the same results proves that $BC$ is a positive definite matrix.

Using the relations $M = B\underline{M}$, $S = B\underline{S}$ and $\underline{A} = AB$ given in the lemma, we deduce from (2.38) that

$$\begin{aligned} P &= [I_n - B\underline{S}(\underline{S}^T BABS)^{-1}\underline{S}^T BA]B\underline{M}[I_n - ABS(\underline{S}^T BABS)^{-1}\underline{S}^T B] \\ &\quad + B\underline{S}(\underline{S}^T BABS)^{-1}\underline{S}^T B \\ &= [B - B\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B\underline{A}]\,\underline{M}\,[I_n - \underline{A}\,\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B] \\ &\quad + B\underline{S}(\underline{S}^T B\underline{A}\,\underline{S})^{-1}\underline{S}^T B \\ &= BC \end{aligned}$$

which completes the proof. □

The formula (3.19) given in Lemma 3.1 can be rewritten inductively if $\underline{S} \in \mathbb{R}^{n\times\ell}$ be $\underline{A}$-conjugate with respect to the $B$-inner product. The next lemma provides such form of the preconditioner (3.19).

**Lemma 3.2** *Let $B\underline{A}$ and $B\underline{M}$ be $n \times n$ symmetric positive definite matrices. Assume also that $\underline{S} \in \mathbb{R}^{n\times\ell}$ be $\underline{A}$-conjugate with respect to the $B$-inner product. Then the matrix $C$ defined by (3.19) can be written as*

$$C_\ell = \left(I_n - \sum_{i=1}^{\ell} \frac{\underline{s}_i\, \underline{s}_i^T}{\underline{s}_i^T B\underline{A}\, \underline{s}_i} B\underline{A}\right) \underline{M} \left(I_n - \sum_{i=1}^{\ell} \underline{A} \frac{\underline{s}_i\, \underline{s}_i^T}{\underline{s}_i^T B\underline{A}\, \underline{s}_i} B\right) + \sum_{i=0}^{\ell} \frac{\underline{s}_i \underline{s}_i^T}{\underline{s}_i^T B\underline{A}\, \underline{s}_i} B, \quad (3.23)$$

*which can also be written inductively as*

$$C_\ell = \left(I_n - \frac{\underline{s}_\ell \underline{s}_\ell^T B\underline{A}}{\underline{s}_\ell^T B\underline{A}\, \underline{s}_\ell}\right) C_{\ell-1} \left(I_n - \frac{\underline{A}\, \underline{s}_\ell \underline{s}_\ell^T B}{\underline{s}_\ell^T B\underline{A}\, \underline{s}_\ell}\right) + \frac{\underline{s}_\ell \underline{s}_\ell^T B}{\underline{s}_\ell^T B\underline{A}\, \underline{s}_\ell}. \quad (3.24)$$

**Proof.** From the $\underline{A}$-conjugacy of the matrix $\underline{S}$ with respect to the $B$-inner product, $\underline{S}^T B\underline{A}\, \underline{S}$ becomes a diagonal matrix. Then, $\underline{S}(\underline{S}^T B\underline{A}\, \underline{S})^{-1}\underline{S}^T$ can be written as

$$\underline{S}(\underline{S}^T B\underline{A}\, \underline{S})^{-1}\underline{S}^T = \sum_{i=1}^{\ell} \frac{\underline{s}_i\, \underline{s}_i^T}{\underline{s}_i^T B\underline{A}\, \underline{s}_i}. \quad (3.25)$$

Substituting this expression in (3.19) gives the equation (3.23).

For the second part of the proof let us define

$$\underline{A} = AB, \quad M = B\underline{M} \quad \text{and} \quad s_i = B\underline{s}_i. \quad (3.26)$$

Multiplying the matrix $C_\ell$ given by (3.23) from the left by $B$ and substituting definitions (3.26) into $BC_\ell$ gives the same formula as (2.39). This formula is written inductively by the equation (2.40). Replacing $p_i$ by $s_i$ in the formula (2.40) and substituting definitions given in (3.26) into (2.40) gives that

$$\begin{aligned} BC_\ell &= \left(I_n - \frac{B\underline{s}_\ell \underline{s}_\ell^T B\underline{A}B^{-1}}{\underline{s}_\ell^T B\underline{A}B^{-1}B\underline{s}_\ell}\right) BC_{\ell-1} \left(I_n - \frac{\underline{A}B^{-1}B\underline{s}_\ell \underline{s}_\ell^T B}{\underline{s}_\ell^T B\underline{A}B^{-1}B\underline{s}_\ell}\right) + \frac{B\underline{s}_\ell \underline{s}_\ell^T B}{\underline{s}_\ell^T B\underline{A}B^{-1}B\underline{s}_\ell} \\ &= \left(B - \frac{B\underline{s}_\ell \underline{s}_\ell^T B\underline{A}}{\underline{s}_\ell^T B\underline{A}\, \underline{s}_\ell}\right) C_{\ell-1} \left(I_n - \frac{\underline{A}\, \underline{s}_\ell \underline{s}_\ell^T B}{\underline{s}_\ell^T B\underline{A}\, \underline{s}_\ell}\right) + \frac{B\underline{s}_\ell \underline{s}_\ell^T B}{\underline{s}_\ell^T B\underline{A}\, \underline{s}_\ell} \\ &= B\left[\left(I_n - \frac{\underline{s}_\ell \underline{s}_\ell^T B\underline{A}}{\underline{s}_\ell^T B\underline{A}\, \underline{s}_\ell}\right) C_{\ell-1} \left(I_n - \frac{\underline{A}\, \underline{s}_\ell \underline{s}_\ell^T B}{\underline{s}_\ell^T B\underline{A}\, \underline{s}_\ell}\right) + \frac{\underline{s}_\ell \underline{s}_\ell^T B}{\underline{s}_\ell^T B\underline{A}\, \underline{s}_\ell}\right], \end{aligned}$$

from which the desired result is obtained. □

Lemma 3.1 and 3.2 suggest to construct a limited memory preconditioner matrix $C$ for the system matrix $\underline{A} = AB$ of the linear system (3.14) that can be used during PCGIF Algorithm 3.4. Therefore, another alternative to precondition the linear system (3.1) is that applying PCGIF Algorithm to the linear system (3.14) of the form of

$$\underline{A}\, \underline{s} = \underline{b}_k \quad (3.27)$$

A equipped with the $B$-inner product for the variable $\underline{s}$ by using the limited memory preconditioner $C$ formulated by (3.19). The approximate solution is obtained from $s = B\underline{s}$.

We are left with the task of integrating this approach when solving a convergent sequence of linear systems with multiple right-hand sides. Assume that we choose the initial guess $x_1$. Assume also that the initial preconditioner $C_0$ such that

$$\boxed{C_0 = B^{-1}P_0} \tag{3.28}$$

is available. Then for $k = 1$, PCGIF Algorithm 3.4 solves the preconditioned system

$$\underline{A}\,\underline{s} = b_1 \tag{3.29}$$

for the variable $\underline{s}$ by using the preconditioner $C_0$ and recovers the approximate solution from the relation $s_1 = B\,\underline{s}_1$. This approximate solution is then used to update the current iterate from $x_2 = x_1 + s_1$.

While performing the PCGIF algorithm on the linear system (3.29), the information can be stored to obtain the LMP $C_1$ for the system matrix $\underline{A}$ by using the formula (3.19). This preconditioner can then be used to precondition the second linear system. Note that while constructing $C_1$ from (3.19), $\underline{M}_0$ is taken as the initial preconditioner matrix, i.e. $\underline{M}_0 = C_0$. From Lemma 3.1 we ensure that $P_1 = BC_1$ where $P_1$ is the preconditioner for the second linear system of Algorithm 3.2 assuming that $S_0 = B\underline{S}_0$.

Using the preconditioner $C_1$ for $k = 2$, PCGIF solves the preconditioned system

$$\underline{A}\,\underline{s} = b_2 \tag{3.30}$$

for the variable $\underline{s}$ and recover the approximate solution from $s_2 = B\,\underline{s}_2$. This approximate solution is then used to update the current iterate from $x_3 = x_2 + s_2$.

As it is done for $k = 1$, while performing the PCGIF algorithm on the linear system (3.30) the information can be stored to obtain the LMP $C_2$ for the system matrix $\underline{A}$ by using the formula (3.19). Note that while constructing $C_2$ from (3.19), $\underline{M}_1$ is taken as the preconditioner used for $k = 2$, $\underline{M}_1 = C_1$. From Lemma 3.1 we ensure that $P_2 = BC_2$ where $P_2$ is the preconditioner for the third linear system in Algorithm 3.2 assuming that $S_1 = B\underline{S}_1$.

The same strategy can be carried over for later steps. Using the preconditioner $C_k$, for the $(k+1)$-th linear system PCGIF Algorithm solves the preconditioned system

$$\underline{A}\,\underline{s} = b_{k+1} \tag{3.31}$$

for the variable $\underline{s}$ and recover the approximate solution from $s_{k+1} = B\underline{s}_{k+1}$. This approximate solution is then used to update the current iterate from $x_{k+2} = x_{k+1} + s_{k+1}$. Algorithm 3.5 outlines this implementation.

In Algorithm 3.5, choosing $x_1 = x_c$ the term $B^{-1}(x_c - x_k)$ can be calculated from the following equation

$$B^{-1}(x_c - x_k) = -\sum_{j=1}^{k-1} \underline{s}_j \tag{3.32}$$

**Algorithm 3.5: PCGIF for solving a convergent sequence of linear systems with multiple right-hand sides**

1 **Initialization:** Assume that $C_0 = B^{-1}P_0$ is available. Choose an initial vector $x_1$.

2 **Calculate right hand side $b_k$:**

$$b_k = B^{-1}(x_c - x_k) - H^T R^{-1} d_k.$$

3 **Perform inner loop**: Using the preconditioner $C_{k-1}$ solve the linear system:

$$\underline{A}\,\underline{s} = b_k$$

with PCGIF Algorithm 3.4. During PCGIF extract and save the information to precondition the next linear system with an LMP $C_k$ formulated by (3.19).

4 **Recover the approximate solution:**

$$s_k = B\underline{s}_k$$

5 **Update the iterate:**

$$x_{k+1} = x_k + s_k$$

6 Increment $k$ by 1 and go to step 2.

where we used the relations $s_k = B\underline{s}_k$ and $x_{k+1} = x_k + s_k$.

*Algorithm 3.5 requires neither the factorization of the preconditioner nor matrix-vector products with $B^{-1}$ under the assumption that $C_0 = B^{-1}P_0$ is available.*

When solving the $k$-th linear system in Algorithm 3.5, PCGIF Algorithm 3.4 requires a single matrix-vector product with each of the operators $H, H^T, R^{-1}, B$ and $C_{k-1}$.

## Summary

We have presented different conjugate gradient algorithms for the solution of the linear system (3.1) with the LMP given by (2.38) in the context of solving a convergent sequence of linear systems with multiple right-hand sides. These algorithms differ in the assumption they made on the preconditioner form and matrix-vector products. They are listed below:

- Approach (A): Apply CG Algorithm 2.2 to the linear system (3.7) of the form of $\widetilde{A}\widetilde{s} = \widetilde{b}_k$ with a preconditioner constructed from (3.2),
- Approach (B): Apply PCG Algorithm 2.5 to the linear system (3.1) of the form of $As = b_k$ with a preconditioner constructed from (2.38),
- Approach (C): Apply PCGIF Algorithm 3.4 to the linear system (3.14) of the form of $\underline{A}\,\underline{s} = b_k$ with a preconditioner constructed from (3.19).

The main properties of these approaches in terms of preconditioning can be listed as follows:

- The first approach (A) requires the factorization of the preconditioner. When used for solving a sequence of linear systems the factorization of initial preconditioner needs to be available.
- The second approach (B) does not require any factorization of the preconditioner.
- The third approach (C) requires neither the factorization of the preconditioner nor the matrix-vector products with $B^{-1}$ under the assumption that $B^{-1}P$ operator is available.

These three approaches are presented by Algorithms 3.1, 3.2 and 3.5 whose characteristics are listed in Table 3.1.

| Algorithm | Inner min. | LMP formula | Assumption |
|---|---|---|---|
| 3.1 | App. (A) | $F_{k+1}$ given by (3.2) with $M_k = I_n$ | $F_0$ is available |
| 3.2 | App. (B) | $P_{k+1}$ given by (2.38) with $M_k = P_k$ | |
| 3.5 | App. (C) | $C_{k+1}$ given by (3.19) with $\underline{M}_k = C_k$ and $S_k = B\underline{S}_k$ | $B^{-1}P_0$ is available |

Table 3.1: A summary of the characteristics of Algorithms 3.1, 3.2 and 3.5

When solving the $k$-th linear system, the iterates of Algorithm 3.1 are related to the iterates of Algorithm 3.2 with

$$s_i = F_0 F_1 \dots F_{k-1} \widetilde{s}_i. \tag{3.33}$$

where $i$ is the index for the inner loop.

We can rewrite the linear system (3.7) as

$$P_k^{1/2} A P_k^{1/2} = P_k^{1/2} b_{k+1} \tag{3.34}$$

where $P_k^{1/2} = F_0 \dots F_k$. From this system it can be observed that the relation between the preconditioner $Y_k = F_k F_k^T$ used during Algorithm 3.1 and the preconditioner $P_k$ used during Algorithm 3.2 can be given as

$$P_k = P_{k-1}^{1/2} Y_k P_{k-1}^{1/2}. \tag{3.35}$$

When solving the $k$-th linear system, the iterates of PCGIF Algorithm 3.4 are related to the iterates of PCG Algorithm 2.5 with

$$s_i = B\underline{s}_i, \tag{3.36}$$

under the assumption that $P_{k-1} = BC_{k-1}$. Note that, from Lemma 3.1 assuming that $P_0 = BC_0$ ensures that $P_k = BC_k$ for $k > 0$, when choosing the first level

preconditioners as given in Table 3.1 and $S_k = B\underline{S}_k$.

Let us now consider the particular case that the initial preconditioner when solving a sequence of linear systems is chosen as $P_0 = B$. Then $C_0 = B^{-1}P_0 = I_n$ is available. In this special case, we summarize the required matrix-vector products for each approach, (A), (B) and (C) for the solution of the $k$-th linear system within Algorithms 3.1, 3.2, and 3.5 in Table 3.2.

| Approach | matrix-vector products | Assumption |
|---|---|---|
| (A) | $H, H^T, R^{-1}, B^{1/2}$, $F_i$, $F_i^T$ for $i = 1, \dots k-1$ | $F_0 = B^{1/2}$ is available |
| (B) | for $k = 1 \rightarrow H, H^T, R^{-1}$ and $B$<br>for $k > 1 \rightarrow H, H^T, R^{-1}, B^{-1}$ and $P_k$ | $P_0 = B$ |
| (C) | $H, H^T, R^{-1}$, $B$ and $C_k$ | $B^{-1}P_0$ is available |

Table 3.2: A summary of the required matrix-vector products for each approach used for the solution of the $k$-th linear system (3.1) with LMPs within Algorithms 3.1, 3.2, and 3.5 under the assumption that $P_0 = B$

*As a result, we know that Algorithms 3.1 and 3.2 generate a mathematically equivalent sequence of iterates $\{s_k\}$ since it is well-known that CG Algorithm 2.2 and PCG Algorithm 2.5 are equivalent algorithms (Golub and Van Loan, 1996). Algorithms 3.2 and 3.5 whose characteristics are listed in Table 3.1 also generate a mathematically equivalent sequence of iterates $\{s_k\}$ which can be easily observed from the relations (3.15). The choice of the algorithm depends on the availability of the required form of the preconditioner, operators and their matrix-vector product cost.*

We next explain how to construct and implement particular LMPs: the quasi-Newton LMP, the Ritz LMP and the spectral LMP in the context of the given algorithms. This part will include for each preconditioner a definition of the preconditioner, its properties, an extraction methodology of the required information from the corresponding conjugate gradient algorithm, and memory and flops requirements for the application of the preconditioner.

### 3.1.1 Preconditioning with the quasi-Newton LMP

We consider in this section the use of quasi-Newton LMPs (Gratton, Sartenaer and Tshimanga, 2011) formulated by the equation (2.40) which are derived from the inverse Hessian approximations using the Limited Memory BFGS (LBFGS) updating formula (Morales and Nocedal, 1999). We adapt the quasi-Newton LMP formula given by (2.40) that can be used during Algorithms 3.1, 3.2 and 3.5.

We start with explaining how to construct a quasi-Newton LMP within Algorithm 3.2 in which the linear system in sequence is solved by Approach (B). Note that the index $(k-1)$ for the preconditioner used for the $k$-th linear system in sequence within these algorithms is dropped for the sake of simplicity throughout this section.

### Approach (B) with the quasi-Newton LMP

Approach (B) applies PCG Algorithm 2.5 to the linear system (3.1) of the form of $As = b_k$. Choosing the search directions $p_i$, $i = 0, \ldots, j$ from PCG Algorithm in the formula (2.40), the quasi-Newton LMP $P_{j+1}$ for $A = B^{-1} + H^T R^{-1} H$ for the $k$-th linear system can be constructed from

$$\boxed{P_{j+1} = (I_n - \tau_j p_j q_j^T) P_j (I_n - \tau_j q_j p_j^T) + \tau_j p_j p_j^T}, \tag{3.37}$$

where $p_j$, $\tau_j = 1/(q_j^T p_j)$ and $q_j = A p_j$ are obtained during PCG applied to the previous linear system. The initial preconditioner matrix (which can be interpreted as the first-level preconditioner) for the formula (3.37) is chosen as the preconditioner used for the previous linear system.

In (3.37) the $j$ "secant pairs" consisting of descent directions $p_i$ and associated changes in gradient $q_i$ can be chosen from different strategies. For instance, the use of the $j$ last pairs is proposed in (Nocedal and Wright, 2006, p. 177), while a uniform sampling across all generated pairs is proposed in (Morales and Nocedal, 1999). A remarkable feature of the update (3.37) is that the matrix $\Delta P_j$ defined by $\Delta P_j = P_{j+1} - P_j$ is the solution to the following minimization problem (Nocedal and Wright, 2006, pp. 138-140):

$$\min_{\Delta P_j} \|W^{1/2} \Delta P_j W^{1/2}\|_F \tag{3.38}$$

$$\text{subject to} \quad \Delta P_j = \Delta P_j^T, \quad P_{j+1} q_j = p_j, \tag{3.39}$$

where $W$ is any symmetric positive definite matrix satisfying $W p_j = q_j$.

Note that $\|W^{1/2} \Delta P_j W^{1/2}\|_F = \|\Delta P_j\|_W$, where $\|\cdot\|_W$ is a weighted Frobenius norm of weight $W$. The solution of problem (3.38)-(3.39) can be computed in close form and is given by

$$\Delta P_j = \frac{W^{-1} q_j (p_j - P_j q_j)^T + (p_j - P_j q_j) q_j^T W^{-1}}{q_j^T W^{-1} q_j} - \frac{q_j^T (p_j - P_j q_j) W^{-1} q_j q_j^T W^{-1}}{(q_j^T W^{-1} q_j)^2}. \tag{3.40}$$

Substituting the expressions $W p_j = q_j$ and $\tau_j = 1/(q_j^T p_j)$ into (3.40) gives that

$$\begin{aligned}
\Delta P_j &= \frac{p_j (p_j - P_j q_j)^T + (p_j - P_j q_j) p_j^T}{q_j^T p_j} - \frac{q_j^T (p_j - P_j q_j) p_j p_j^T}{(q_j^T p_j)^2} \\
&= \tau_j p_j (p_j - P_j q_j)^T + \tau_j (p_j - P_j q_j) p_j^T - \tau_j^2 q_j^T (p_j - P_j q_j) p_j p_j^T \\
&= \tau_j p_j p_j^T - \tau_j p_j q_j^T P_j + \tau_j p_j p_j^T - \tau_j P_j q_j p_j^T - \tau_j^2 (q_j^T p_j) p_j p_j^T + \tau_j^2 q_j^T P_j q_j p_j p_j^T \\
&= \tau_j p_j p_j^T - \tau_j p_j q_j^T P_j + \tau_j p_j p_j^T - \tau_j P_j q_j p_j^T - \tau_j p_j p_j^T + \tau_j^2 q_j^T P_j q_j p_j p_j^T \\
&= \tau_j p_j p_j^T - \tau_j p_j q_j^T P_j - \tau_j P_j q_j p_j^T + \tau_j^2 q_j^T P_j q_j p_j p_j^T,
\end{aligned}$$

from which it can be seen that $P_j + \Delta P_j$ gives (3.37).

We now explain how to construct a quasi-Newton LMP within Algorithm 3.1 in which the linear system in sequence is solved by Approach (A).

### Approach (A) with the quasi-Newton LMP

Approach (A) applies CG Algorithm 2.2 to the linear system (3.7) of the form of $\widetilde{A}_k\,\widetilde{s} = \widetilde{b}_k$ within Algorithm 3.1. Choosing the search directions $\widetilde{p}_i$, $i = 0, \ldots, j$ from CG Algorithm in the formula (2.40), the quasi-Newton LMP $Y_{j+1}$ for $\widetilde{A}_k$ in the $k$-th linear system can be constructed from

$$Y_{j+1} = (I_n - \tau_j \widetilde{p}_j \widetilde{q}_j^T) Y_j (I_n - \tau_j \widetilde{q}_j \widetilde{p}_j^T) + \tau_j \widetilde{p}_j \widetilde{p}_j^T, \tag{3.41}$$

where $\widetilde{p}_j$, $\tau_j = 1/(\widetilde{q}_j^T \widetilde{p}_j)$ and $\widetilde{q}_j = \widetilde{A}_{k-1}\widetilde{p}_j$ are obtained during CG applied to the previous linear system. Note that, the initial preconditioning matrix is chosen as the identity matrix of order $n$.

Remember that while applying the quasi-Newton LMP $Y_{j+1}$ during CG Algorithm 2.2, the square-root factorization of $Y_{j+1}$ is required. This factored form $Y_{j+1} = F_{j+1}F_{j+1}^T$ can be calculated from (Gratton, Sartenaer and Tshimanga, 2011)

$$\boxed{F_{j+1} = \left[ I_n - \widetilde{p}_j \left( \tau_j \widetilde{q}_j^T + \sqrt{\tau_j} \frac{\widetilde{r}_j^T}{\|\widetilde{r}_j\|_2} \right) \right] F_j}, \tag{3.42}$$

where $\widetilde{r}_j$ is the residual defined in CG Algorithm. Note that for the first linear system, the factorization of the preconditioner is assumed to be available.

We are left with the task of deriving the quasi-Newton LMP for Approach (C) which applies PCGIF Algorithm 3.4 on the linear system (3.14) of the form of $\underline{A}\,\underline{s} = b_k$ within Algorithm 3.5.

### Approach (C) with the quasi-Newton LMP

Let us assume that $P_0 = BC_0$, where $P_0$ is the initial preconditioner used for Algorithm 3.2 and $C_0$ is the initial preconditioner used for Algorithm 3.5. Assume also that for the $k$-th linear system $P_{k-1}$ is constructed from (2.38) and $C_{k-1}$ is constructed from (3.19) where the pairs are chosen such that $S_{k-1} = B\underline{S}_{k-1}$. Under these assumptions from Lemma 3.1, the relation $P_k = BC_k$ naturally hold (see Section 3.1).

When using the conjugate-gradient method, the pairs can be chosen as search directions. Let us now assume that for the $k$-th linear system $P_{k-1}$ is constructed from (2.38) by using the search directions $p_i$, $i = 0, \ldots, j$ that are obtained during PCG Algorithm 2.5 and $C_{k-1}$ is constructed from (3.19) by using the search directions $\underline{p}_i$, $i = 0, \ldots, j$ that are obtained during PCGIF Algorithm 3.4 while solving the $(k-1)$-th linear system. From the relation $p_i = B\underline{p}_i$ given in (3.15), the assumption $S_{k-1} = B\underline{S}_{k-1}$ is satisfied. Using the same strategy as explained in Section 3.1, it can be observed that the relation $P_k = BC_k$ hold (where the preconditioners $P_k$ and $C_k$ are constructed from the search directions obtained from PCG Algorithm and PCGIF Algorithm respectively when solving the $k$-th linear system).

We can also observe from (3.16) that when solving the $(k+1)$ linear system in Algorithm 3.5 the search directions $\underline{p}_i$ generated by PCGIF Algorithm 3.4 are $\underline{A}(= AB)$-conjugate with respect to the $B$-inner product. Using this property from Lemma 3.2,

the LMP given by (3.19) can be written as

$$C_{j+1} = \left( I_n - \frac{\underline{p}_j \underline{p}_j^T B \underline{A}}{\underline{p}_j^T B \underline{A} \underline{p}_j} \right) C_j \left( I_n - \frac{\underline{A} \underline{p}_j \underline{p}_j^T B}{\underline{p}_j^T B \underline{A} \underline{p}_j} \right) + \frac{\underline{p}_j \underline{p}_j^T B}{\underline{p}_j^T B \underline{A} \underline{p}_j}.$$

Using this formula the quasi-Newton LMP for the $k$-th linear system in Algorithm 3.5 can be constructed from

$$\boxed{C_{j+1} = \left( I_n - \tau_j \underline{p}_j \underline{q}_j^T B \right) C_j \left( I_n - \tau_j \underline{q}_j p_j^T \right) + \tau_j \underline{p}_j p_j^T} \tag{3.43}$$

where $p_i = B\underline{p}_i$, $\underline{q}_i = \underline{A}\underline{p}_i$ and $\tau_i = 1/(\underline{q}_i^T p_i)$ are obtained during PCGIF applied to the previous linear system. The initial matrix to construct the preconditioner (3.43) for the $k$-th linear system is chosen as the preconditioner that is used for the $(k-1)$-th linear system.

Note that, from Lemma 3.1 the preconditioner (3.43) is symmetric with respect to the $B$-inner product and $BC_{j+1}$ is positive definite. Using the relations from (3.15), it can be easily shown that the preconditioner $P_{j+1}$ formulated by (3.37) and the preconditioner formulated by (3.43) satisfies the relation $P_{j+1} = BC_{j+1}$.

In the formula (3.43) the vectors $\underline{p}_j$, $\underline{q}_j$ and the scalar $\tau_j$ are by-products of the PCGIF Algorithm, however the quantity $B\underline{q}_i$ is not a direct by-product of the algorithm. Therefore it may seem at first that constructing the preconditioner $C$ requires additional matrix-vector products with matrix $B$. However, the matrix-vector product $B\underline{q}_j$ can be obtained also as a by-product of PCGIF Algorithm as explained below.

Introducing a new vector $\underline{l}$ such that

$$\underline{l}_i = B\underline{r}_i,$$

suggests computing the vector $z_i$ defined in Algorithm 3.4 from the relation

$$z_i = B\underline{z}_i = BC\underline{r}_i = C^T B\underline{r}_i = C^T \underline{l}_i,$$

where we used the symmetry of the preconditioner $C$ with respect to the $B$ inner product, i.e. $BC = C^T B$. Multiplying line 11 of Algorithm 3.4 by $B$ gives that

$$\underline{l}_{i+1} = \underline{l}_i - \alpha_i B\underline{q}_i$$

from which the formula to compute the matrix-vector product $B\underline{q}_i$ can be obtained as

$$B\underline{q}_i = (\underline{l}_i - \underline{l}_{i+1})/\alpha_i.$$

Defining $t_i = B\underline{q}_i$ and using all the relations Algorithm 3.4 can be transformed into Algorithm 3.6 on the following page.

*As a result it is possible to find the corresponding quasi-Newton LMPs for variants of conjugate gradients (Approaches (A), (B) and (C)) such that when they are used within the corresponding algorithms (Algorithms 3.1, 3.2 and 3.5), these algo-*

**Algorithm 3.6: PCGIF Algorithm (version for quasi-Newton LMP)**

$\underline{r}_0 = b - As_0$
$\underline{l}_0 = B\underline{r}_0$
$\underline{z}_0 = C\underline{r}_0$
$\underline{p}_0 = \underline{z}_0$
$z_0 = C^T\underline{l}_0$
$\rho_0 = \underline{r}_0^T z_0$
$p_0 = z_0$
**for** $i = 0, 1, ..., l-1$ **do**
  $\underline{q}_i = (\underline{p}_i + H^T R^{-1} H p_i)$
  $\alpha_i = \rho_i / \underline{q}_i^T p_i$
  $\underline{s}_{i+1} = \underline{s}_i + \alpha_i \underline{p}_i$
  $\underline{r}_{i+1} = \underline{r}_i - \alpha_i \underline{q}_i$
  $\underline{l}_{i+1} = B\underline{r}_{i+1}$
  $t_i = (\underline{l}_i - \underline{l}_{i+1})/\alpha_i$
  $\underline{z}_{i+1} = C\underline{r}_{i+1}$
  $z_{i+1} = C^T\underline{l}_{i+1}$
  Check convergence and stop if desired accuracy is reached
  $\rho_{i+1} = \underline{r}_{i+1}^T z_{i+1}$
  $\beta_{i+1} = \rho_{i+1}/\rho_i$
  $\underline{p}_{i+1} = \underline{z}_{i+1} + \beta_{i+1}\underline{p}_i$
  $p_{i+1} = z_{i+1} + \beta_{i+1} p_i$
**end**

*rithms generate a mathematically equivalent sequence of iterates* $\{s_k\}$. *We showed that such preconditioners can be constructed from vectors which are by-products of the corresponding algorithms (Algorithms 2.2, 2.5 and 3.4).*

We summarize the characteristics of Algorithms 3.1, 3.2 and 3.5 when used with the quasi-Newton LMP in Table 3.3. In this table we define another approach for PCGIF Algorithm when used with the quasi-Newton LMP :

| Approach (D): Apply PCGIF Algorithm 3.6 on the linear system (3.14) of the form of $\underline{A}\,\underline{s} = \underline{b}_k$ with a preconditioner constructed from (3.43) |
|---|

| Algorithm | Inner min. | Quasi-Newton LMP formula | Assumption |
|---|---|---|---|
| 3.1 | Approach (A) | $F_j$ given by (3.42) | $F_0$ is available |
| 3.2 | Approach (B) | $P_j$ given by (3.37) | |
| 3.5 | Approach (D) | $C_j$ given by (3.43) | $B^{-1}P_0$ is available |

Table 3.3: A summary of the characteristics of Algorithms 3.1, 3.2 and 3.5 where preconditioning is achieved by using the quasi-Newton LMPs.

We now want to discuss the implementation issues for each quasi-Newton LMP. For Approach (A) the matrix-vector products with the quasi-Newton preconditioner

$F_j$ and its transpose $F_j^T$ can be calculated recursively (Tshimanga, 2007). In addition to matrix-vector products with the initial matrices, $F_0$ and $F_0^T$, these two recursions require in total $(8j+3)n$ flops. For Approach (B) matrix-vector products with $P_j$ can be calculated using the L-BFGS two-loop recursion (Nocedal and Wright, 2006, p.178, Algorithm 7.4). In addition to matrix-vector products with the initial matrix $P_0$, this recursion requires in total $8jn$ flops. For Approach (D), matrix-vector products with $C_j$ and $C_j^T$ can be calculated by adapting the L-BFGS two-loop recursion which results in Algorithm 3.7 and 3.8. Each algorithm requires $8jn$ flops and matrix-vector products with $C_0$ and $C_0^T$ respectively. We summarize the cost for applying the preconditioner and memory requirements in Table 3.4.

**Algorithm 3.7: Compute** $\underline{z}= C\underline{r}$

Given $\underline{r}$ set $v \leftarrow \underline{r}$
**for** $i = j : -1 : 1$ **do**
    $\eta_i \leftarrow \tau_i p_i^T v$
    $v \leftarrow v - \eta_i \underline{q}_i$
**end**
$\underline{z} \leftarrow C_0 v$
**for** $i = 1 : j$ **do**
    $\zeta \leftarrow \tau_i t_i^T \underline{z}$
    $\underline{z} \leftarrow \underline{z} + \underline{p}_i(\eta_i - \zeta)$
**end**

**Algorithm 3.8: Compute** $z = C^T\underline{l}$

Given $\underline{l}$ set $v \leftarrow \underline{l}$
**for** $i = j : -1 : 1$ **do**
    $\eta_i \leftarrow \tau_i \underline{p}_i^T v$
    $v \leftarrow v - \eta_i t_i$
**end**
$z \leftarrow C_0^T v$
**for** $i = 1 : j$ **do**
    $\zeta \leftarrow \tau_i \underline{q}_i^T z$
    $z \leftarrow z + p_i(\eta_i - \zeta)$
**end**

| Approach | LMP | Reference | Memory req. | Flops req. | mat-vec prod. |
|---|---|---|---|---|---|
| (A) | $F_j$ and $F_j^T$ | (3.42) | $(3n+1)j$ | $8jn+3n$ | $F_0$ and $F_0^T$ |
| (B) | $P_j$ | (3.37) | $(2n+1)j$ | $8jn$ | $P_0$ |
| (D) | $C_j$ and $C_j^T$ | (3.43) | $(4n+1)j$ | $16jn$ | $C_0$ and $C_0^T$ |

Table 3.4: Memory and cost requirements for applying the quasi-Newton LMPs during Approaches (A), (B) and (D)

From Table 3.4, we can see that applying $C$ and $C^T$ require more memory and flops comparing the others. However, for large scale problems considering Approach (D) avoids both factorization of $B$ and inverse of $B$ under the assumption that $B^{-1}P$ is available, $16jn$ flops become very small comparing the cost for matrix-vector multiplications with $B^{-1}$ and $B^{1/2}$.

### 3.1.2 Preconditioning with the Ritz LMP

We now consider preconditioning a linear system by using spectral information in particular Ritz pairs, which is shown to be an effective preconditioning technique for variants of CG or Lanczos methods (Fisher, 1998), (Gratton, Sartenaer and Tshimanga, 2011) and (Rey and Risler, 1998). In this section we focus on the Ritz LMPs (Gratton, Sartenaer and Tshimanga, 2011) formulated by the equation (2.41) and adapt this preconditioner that it can be used during Algorithms 3.1, 3.2 and 3.5.

Let us first recall the definition of the Ritz pair (Parlett, 1980):

**Definition 3.1** *A scalar $\theta$ is called a Ritz value of $A$ with respect to a subspace $\mathcal{L}$ if there exists a nonzero vector $u \in \mathcal{L}$, called Ritz vector, such that $(Au - \theta u) \perp \mathcal{L}$, where orthogonality is considered with respect to the canonical inner product. The pair $(\theta, u)$ is called a Ritz pair of $A$ with respect to $\mathcal{L}$. If $\theta$ is a Ritz value, its multiplicity is the maximum number of independent vectors $u_i$ such that $(\theta, u_i)$ is a Ritz pair of $A$ with respect to $\mathcal{L}$.*

We want to construct such Ritz pairs to construct a Ritz LMP from the information that is obtained during variants of the CG algorithm (Approaches (A), (B) and (C)). Since the spectral information is not directly available from a CG run, we first explain how to extract these Ritz pairs from the iterates of the each algorithm and later we define the Ritz-LMP.

We start with explaining how to use a Ritz LMP within Algorithm 3.1 in which the linear system in sequence is solved by Approach (A).

#### Approach (A) with the Ritz LMP

Let us consider applying CG Algorithm 2.2 to the linear system (3.7) of the form of $\widetilde{A}_k\, \widetilde{s} = \widetilde{b}_k$ within Algorithm 3.1. Assume that CG Algorithm performed $\ell$ successive inner iterations. Then the approximate solution $\widetilde{s}_\ell$ calculated by CG Algorithm lies in a subspace $\widetilde{s}_0 + \mathcal{K}^\ell(\widetilde{A}_k, \widetilde{r}_0)$ where $\widetilde{s}_0$ is the initial guess and $\widetilde{r}_0$ is the initial residual. As

explained in Section 2.6.4, the orthonormal basis for the Krylov subspace $\mathcal{K}^\ell(\widetilde{A}_k, \widetilde{r}_0)$ can be constructed from the column vectors $\widetilde{v}_i$, $i = 1, \ldots, \ell$. These vectors are extracted from

$$\widetilde{v}_{i+1} = (-1)^i \frac{\widetilde{r}_i}{\|\widetilde{r}_i\|_2}, \tag{3.44}$$

where $\widetilde{r}_i$ is the residual given in Algorithm 2.2. The projected matrix

$$T_\ell = \widetilde{V}_\ell^T \widetilde{A}_k \widetilde{V}_\ell \tag{3.45}$$

where $\widetilde{V}_\ell = [\widetilde{v}_1, \ldots, \widetilde{v}_l]$ can also be computed during CG Algorithm as explained in Section 2.6.4. Therefore, the Ritz pairs $(\theta_i, \widetilde{u}_i)$, $i = 1, \ldots, \ell$ of the matrix $\widetilde{A}_k$ with respect to subspace $\mathcal{K}^\ell(\widetilde{A}_k, \widetilde{r}_0)$ can be computed by using the eigenpairs $(\theta_i, \bar{u}_i)$ of the matrix $T_\ell$ as follows:

$$(\theta_i, \widetilde{u}_i) = (\theta_i, \widetilde{V}_\ell \bar{u}_i). \tag{3.46}$$

Choosing these Ritz pairs $(\theta_i, \widetilde{u}_i)$, $i = 1, \ldots, \ell$ obtained from CG Algorithm in the formula (2.41), the Ritz LMP for the matrix $\widetilde{A}_k$ in $k$-th linear system can be generated from (Gratton, Sartenaer and Tshimanga, 2011)

$$Y_{k-1} = I_n + \widetilde{U}(\Theta^{-1} - I_\ell)\widetilde{U}^T - \widetilde{U}\omega\widetilde{v}_{\ell+1}^T - \widetilde{v}_{\ell+1}\omega^T\widetilde{U}^T + \widetilde{U}\omega\omega^T\widetilde{U}^T, \tag{3.47}$$

where $\widetilde{U} = [\widetilde{u}_1, \ldots, \widetilde{u}_\ell]$, $\Theta = diag(\theta_i)$, and $\omega = (\omega_1, ..., \omega_\ell)^T$, with

$$\omega_i = \frac{(e_\ell^T \bar{u}_i)\beta_{\ell+1}}{\theta_i}, \quad \text{for } i = 1, ..., \ell. \tag{3.48}$$

The vector $e_\ell$ is defined as the $\ell$-th column of the identity matrix $I_\ell$ of order $\ell$. In formula (3.47), the Lanczos vector $\widetilde{v}_{\ell+1}$ can be computed from (3.44) for $i = \ell$ and the scalar $\beta_{\ell+1}$ can be obtained from formula (2.36) where $\eta_{\ell+1} = \beta_{\ell+1}$.

While applying the preconditioner during Algorithm 3.1 the factorization of $Y_{k-1}$ is required. This factored form $Y_{k-1} = F_{k-1}F_{k-1}^T$ can be calculated from (Gratton, Sartenaer and Tshimanga, 2011)

$$\boxed{F_{k-1} = I_n + \widetilde{U}(\Theta^{-1/2} - I_\ell)\widetilde{U}^T - \widetilde{U}\omega\widetilde{v}_{\ell+1}^T}. \tag{3.49}$$

Note that for the first linear system, the factorization of $P_0 = F_0F_0^T$ is assumed to be available. Algorithm 3.9 summarizes how to built the Ritz-LMP (3.49).

We now explain how to construct a Ritz LMP within Algorithm 3.2 in which the linear system in sequence is solved by Approach (B).

### Approach (B) with the Ritz LMP

Let us consider applying PCG Algorithm 2.5 to the linear system (3.1) of the form of $As = b_k$ within Algorithm 3.2. In order to define the Ritz-LMP for the matrix $A$, we use the relations of the information obtained from CG Algorithm 2.2 and PCG Algorithm 2.5 as follows.

**Algorithm 3.9: Construct the factored form of the Ritz-LMP for the matrix $\widetilde{A}_k$**

1 Given $\widetilde{V}_\ell$, $T_\ell$, $\widetilde{v}_{\ell+1}$ and $\beta_{\ell+1}$ from CG Algorithm 2.2

2 Calculate the eigenpairs $(\theta_i, \bar{u}_i)$, $i = 1, \ldots, \ell$ of the tridiagonal matrix $T_\ell$:

$$T_\ell \, \bar{U} = \Theta \, \bar{U}$$

where $\Theta = diag(\theta_i)$ and $\bar{U} = [\bar{u}_1, \ldots, \bar{u}_\ell]$.

3 Generate the vector $\omega = (\omega_1, ..., \omega_\ell)^T$ with $\omega_i$ calculated from the formula (3.48)

4 Calculate the Ritz pairs $(\theta_i, \widetilde{u}_i)$ of the matrix $\widetilde{A}_k$:

$$(\Theta, \widetilde{U}) = (\Theta, \widetilde{V}_l \bar{U})$$

where $\widetilde{U} = [\widetilde{u}_1, \ldots, \widetilde{u}_\ell]$.

5 Construct the factored form of the Ritz-LMP from the formula (3.49)

From the relation (3.35), preconditioning the linear system $\widetilde{A}_k \widetilde{s} = \widetilde{b}_k$ given by (3.7) with the preconditioner $Y_k = F_k F_k^T$ during CG Algorithm 2.2 is equivalent to preconditioning the linear system $As = b_k$ given by (3.1) with the preconditioner $P_k$. Therefore, the preliminary Ritz-LMP $P_{k-1}$ for the matrix $A$ can be derived by substituting the Ritz-LMP formula $Y_{k-1}$ given by (3.47) into (3.35) which gives that

$$\begin{aligned} P_{k-1} &= P_{k-2}^{1/2} Y_{k-1} P_{k-2}^{1/2} \\ &= P_{k-2} + P_{k-2}^{1/2} \widetilde{U} (\Theta^{-1} - I_\ell) \widetilde{U}^T P_{k-2}^{1/2} - P_{k-2}^{1/2} \widetilde{U} \omega \widetilde{v}_{l+1}^T P_{k-2}^{1/2} \\ &\quad - P_{k-2}^{1/2} \widetilde{v}_{l+1} \omega^T \widetilde{U}^T P_{k-2}^{1/2} + P_{k-2}^{1/2} \widetilde{U} \omega \omega^T \widetilde{U}^T P_{k-2}^{1/2}. \end{aligned} \tag{3.50}$$

In this formula, the Lanczos vectors and the spectral information obtained from CG Algorithm 2.2 is required. We next explain by Lemma 3.3 and 3.4 how to extract this information from PCG Algorithm 2.5, and later reformulate this LMP in terms of the vectors and spectral information obtained from PCG Algorithm.

**Lemma 3.3** *Let $\widetilde{v}_i$, $i = 1, \ldots, \ell$ be $\ell$ Lanczos vectors generated from (3.44) during CG Algorithm 2.2 applied to the preconditioned linear system $\widetilde{A}\widetilde{s} = \widetilde{b}$ where $\widetilde{A} = P^{1/2} A P^{1/2}$, $\widetilde{s} = P^{-1/2} s$ and $\widetilde{b} = P^{1/2} b$, and $P$ is symmetric and positive definite. Then the Lanczos vectors $v_i$, $i = 1, \ldots, \ell$ generated from the formula*

$$v_{i+1} = (-1)^i \frac{r_i}{\|r_i\|_P} \tag{3.51}$$

*during PCG Algorithm 2.5 applied to the linear system $As = b$ with the preconditioner $P$ satisfies the relation*

$$\widetilde{v}_i = P^{1/2} v_i. \tag{3.52}$$

*The Lanczos vectors $v_i$, $i = 1, \ldots, \ell$ construct an orthonormal basis with respect to the $P$-inner product for the Krylov subspace $\mathcal{K}^\ell(AP, r_0)$. The projected matrix given by (3.45) can be written in terms of the matrix $V_\ell = [v_1, \ldots, v_\ell]$ as*

$$T_\ell = V_\ell^T P A P V_\ell \tag{3.53}$$

*which defines an orthogonal projection of $AP$ with respect to the $P$-inner product onto the Krylov subspace $\mathcal{K}^\ell(AP, r_0)$.*

*Moreover, the vectors defined by*

$$g_i = P v_i \tag{3.54}$$

*constructs an orthonormal basis with respect to the $P^{-1}$-inner product for the Krylov subspace $\mathcal{K}^\ell(PA, Pr_0)$. The projected matrix given by (3.45) can be written in terms of the matrix $G_\ell = [g_1, \ldots, g_\ell]$ as*

$$T_\ell = G_\ell^T A G_\ell \tag{3.55}$$

*which defines an orthogonal projection of $PA$ with respect to the $P^{-1}$-inner product onto the Krylov subspace $\mathcal{K}^\ell(PA, Pr_0)$.*

**Proof.** Using the relations given in the Lemma it can be easily shown that the residuals $r_i = b - As_i$ and $\widetilde{r}_i = \widetilde{b}_i - \widetilde{A}\widetilde{s}_i$ generated by Algorithm 2.5 and Algorithm 2.2 satisfy the following relation (Golub and Van Loan, 1996, p. 533)

$$\widetilde{r}_i = P^{1/2} r_i,$$

with $i = 0, \ldots, l-1$. Substituting this relation into (3.44) gives that

$$\widetilde{v}_{i+1} = (-1)^i \frac{\widetilde{r}_i}{\|\widetilde{r}_i\|_2} = (-1)^i \frac{P^{1/2} r_i}{\|r_i\|_P} = P^{1/2} \left[ (-1)^i \frac{r_i}{\|r_i\|_P} \right] = P^{1/2} v_{i+1}$$

which proves the first part of the Lemma. The Lanczos vectors $\widetilde{v}_{i+1}$ construct an orthonormal basis for the Krylov subspace $\mathcal{K}^\ell(\widetilde{A}, \widetilde{r}_0)$ (Saad, 1996, p.147, 152). From orthonormal vectors $\widetilde{v}_{i+1}$ and the relation $\widetilde{v}_{i+1} = P^{1/2} v_{i+1}$, we have $V_\ell^T P V_\ell = I_\ell$. Using this equation with the definition (3.54) yields that $G_\ell^T P^{-1} G_\ell = I_\ell$.

We can write the Krylov subspace $\mathcal{K}^\ell(\widetilde{A}, \widetilde{r}_0)$ as

$$\begin{aligned}
\mathcal{K}^\ell(\widetilde{A}, \widetilde{r}_0) &= \text{span}\{\widetilde{r}_0, \widetilde{A}\widetilde{r}_0, \ldots, \widetilde{A}^{l-1}\widetilde{r}_0\} \\
&= \text{span}\{P^{1/2} r_0, P^{1/2} A P^{1/2} P^{1/2} r_0, \ldots, (P^{1/2} A P^{1/2})^{l-1} P^{1/2} r_0\} \\
&= P^{1/2} \text{span}\{r_0, APr_0, \ldots, (AP)^{l-1} r_0\} \\
&= P^{1/2} \mathcal{K}^\ell(AP, r_0).
\end{aligned} \tag{3.56}$$

Then, from the relations (3.52) and (3.56), the column vectors of the matrix $V_\ell$ form an orthonormal basis with respect to the $P$-inner product for the Krylov subspace $\mathcal{K}^\ell(AP, r_0)$.

Finally, $T_\ell$ matrix generated from (3.45) can be rewritten as

$$\begin{aligned} T_\ell &= \widetilde{V}_\ell^T \widetilde{A} \widetilde{V} \\ &= V_\ell^T P A P V_\ell \end{aligned} \tag{3.57}$$

where we used the relation $\widetilde{A} = P^{1/2} A P^{1/2}$ and $\widetilde{V}_\ell = P^{1/2} V_\ell$.

We now show that $g_i$ vectors form an orthornormal basis for $\mathcal{K}^\ell(PA, Pr_0)$. Multiplying the subspace $\mathcal{K}^\ell(AP, r_0)$ spanned by the vectors $v_i$ from the right with $P$ gives

$$\begin{aligned} P\mathcal{K}^\ell(AP, r_0) &= \text{span}\{Pr_0, PAPr_0, \ldots, P(AP)^{\ell-1} r_0\} \\ &= \text{span}\{Pr_0, (PA)Pr_0, \ldots, (PA)^{\ell-1} Pr_0\} \\ &= \mathcal{K}^\ell(PA, Pr_0). \end{aligned} \tag{3.58}$$

From this relation and (3.54), the columns of the matrix $G_\ell$ form an orthonormal basis with respect to the $P^{-1}$-inner product for the Krylov subspace $\mathcal{K}^\ell(PA, r_0)$.

Using the relations (3.57) and $G_\ell = PV_\ell$, the matrix $T_\ell$ (3.45) can be written as

$$\begin{aligned} T_\ell &= V_\ell^T P A P V_\ell \\ &= G_\ell^T A G_\ell \end{aligned}$$

which completes the proof. □

**Lemma 3.4** *Suppose that $(\theta_i, \widetilde{u}_i)$, $i = 1, \ldots, \ell$ be the Ritz pairs of the matrix $\widetilde{A}$ with respect to the subspace $\mathcal{K}^\ell(\widetilde{A}, \widetilde{r}_0)$, i.e. $(\theta_i, \widetilde{u}_i) = (\theta_i, \widetilde{V}_\ell \bar{u}_i)$. The matrix $\widetilde{V}_\ell$ consists of column vectors $\widetilde{v}_i$, $i = 1, \ldots, \ell$ defined in Lemma 3.3, and $\bar{u}_i$ are the eigenvectors of the matrix $T_\ell$ given by (3.45). Let's define the vectors $u_i$ such that*

$$u_i = V_\ell \bar{u}_i \tag{3.59}$$

*where $V_\ell = [v_1, \ldots, v_\ell]$ is a matrix whose column vectors are defined in Lemma 3.3. Then the vectors $u_i$ satisfy the relation*

$$\widetilde{u}_i = P^{1/2} u_i \tag{3.60}$$

*and the pairs $(\theta_i, u_i)$ define the Ritz pairs of the matrix of $AP$ with respect to the subspace $\mathcal{K}^\ell(AP, r_0)$.*

*Let us define the vectors $x_i$ such that*

$$x_i = G_\ell \bar{u}_i \tag{3.61}$$

*where $G_\ell = [g_1, \ldots, g_\ell]$ is a matrix whose column vectors are defined in Lemma 3.3. Then the vectors $x_i$ satisfy the relation*

$$x_i = P u_i \tag{3.62}$$

*and the pairs* $(\theta_i, x_i)$ *define the Ritz pairs of the matrix* $PA$ *with respect to the subspace* $\mathcal{K}^\ell(PA, Pr_0)$.

**Proof.** Using the relation (3.52), we can write the vectors $\widetilde{u}_i$ as

$$\widetilde{u}_i = \widetilde{V}_\ell \bar{u}_i = P^{1/2} V_\ell \bar{u}_i$$

from which it can be seen that the vectors defined by (3.59) satisfies the relation (3.60).

From the definition of the lemma $(\theta_i, \bar{u}_i)$ are the eigenpairs of the tridiagonal matrix $T_l$ given by (3.45). Defining $\Theta = diag(\theta_i)$ and $\bar{U} = (\overline{u}_1, \ldots, \overline{u}_\ell)$, we have

$$\bar{U}^T \, T_\ell \, \bar{U} = \Theta. \tag{3.63}$$

From Lemma 3.3 the matrix $T_\ell$ can be written as $T_\ell = V_\ell^T PAPV_\ell$ where $V_\ell$ has orthonormal columns with respect to the $P$-inner product, i.e. $V_\ell^T PV_\ell = I_\ell$ that span $\mathcal{K}^\ell(AP, r_0)$. Then, the pairs $(\theta_i, u_i) = (\theta_i, V_\ell \overline{u}_i)$ define the Ritz pairs of the matrix $AP$ with respect to the subspace $\mathcal{K}^\ell(AP, r_0)$ (Golub and Van Loan, 1996, p.402, 474-475).

We now prove the second part of the lemma. Using the relation (3.54), it can be easily seen that $G_\ell = PV_\ell$. Substituting this relation into the definition (3.61) yields that

$$x_i = G_\ell \bar{u}_i = PV_\ell \bar{u}_i = Pu_i$$

where we use the definition (3.59).

From Lemma 3.3 the matrix $T_\ell$ can be written as $T_\ell = G_\ell^T P^{-1} PAG_\ell$ where $G_\ell$ has orthonormal columns with respect to the $P^{-1}$-inner product, i.e. $G_\ell^T P^{-1} G_\ell = I_\ell$ that span $\mathcal{K}^\ell(PA, Pr_0)$. Then, the pairs $(\theta_i, x_i) = (\theta_i, G_\ell \overline{u}_i)$ define the Ritz pairs of the matrix $PA$ with respect to the subspace $\mathcal{K}^\ell(PA, Pr_0)$ (Golub and Van Loan, 1996, p.402, 474-475).

□

We now use the vectors and relations given by Lemma 3.3 and Lemma 3.4 to reformulate the LMP given by (3.50) in the sense that all the information can be obtained directly from PCG Algorithm 2.5.

Let us define the column matrix $U = [u_1, \ldots, u_\ell]$ whose columns are defined in Lemma 3.4. Substituting the relations (3.60) for $i = 1, \ldots, \ell$ and (3.52) for $i = \ell + 1$, into (3.50) we obtain the formula

$$\begin{aligned} P_{k-1} = P_{k-2} + P_{k-2} U(\Theta^{-1} - I_\ell) U^T P_{k-2} - P_{k-2} U \omega v_{l+1}^T P_{k-2} \\ - P_{k-2} v_{l+1} \omega^T U^T P_{k-2} + P_{k-2} U \omega \omega^T U^T P_{k-2}, \end{aligned} \tag{3.64}$$

where the vector $v_{l+1}$ is defined by (3.51) for $i = l$. In order to obtain the vector $\omega$ and the matrices $U$ and $\Theta$ in the formula (3.64), the matrices $V_\ell$ and $T_\ell$ defined in Lemma 3.3 and the scalar $\beta_{\ell+1}$ needs to be generated and stored. The matrix $V_\ell$ is generated from the column vectors defined by (3.51) for $i = 0, \ldots, \ell - 1$. The matrix $T_\ell$ is obtained as explained in Section 2.6.4 and the scalar $\beta_{\ell+1}$ can be obtained from

formula (2.36) where $\eta_{\ell+1} = \beta_{\ell+1}$. Once we have all the required information from PCG Algorithm 2.5, Algorithm 3.10 summarizes how to built the the Ritz-LMP (3.64).

**Algorithm 3.10: Construct the Ritz-LMP for the matrix $A$ (version 1)**

1 Given $P_{k-2}$, $V_\ell$, $T_\ell$, $v_{\ell+1}$ and $\beta_{\ell+1}$ from PCG Algorithm 2.5

2 Calculate the eigenpairs $(\theta_i, \bar{u}_i)$, $i = 1, \ldots, \ell$ of the tridiagonal matrix $T_\ell$:

$$T_\ell \, \bar{U} = \Theta \, \bar{U}$$

where $\Theta = diag(\theta_i)$ and $\bar{U} = [\bar{u}_1, \ldots, \bar{u}_\ell]$.

3 Generate the vector $\omega = (\omega_1, ..., \omega_\ell)^T$ with $\omega_i$ calculate from the formula (3.48)

4 Calculate the Ritz pairs $(\theta_i, u_i)$ of the matrix $AP$:

$$(\Theta, U) = (\Theta, V_l \bar{U})$$

where $U = [u_1, \ldots, u_\ell]$.

5 Construct the Ritz-LMP from the formula (3.64)

The Ritz-LMP $P_{k-1}$ given by (3.64) can be written in a more compact form as explained below.

Let us define a matrix $X = [x_1, \ldots, x_\ell]$ whose column vectors are defined in Lemma 3.4. Substituting the relations (3.62) for $i = 1, \ldots, \ell$, we obtain that

$$\boxed{P_{k-1} = P_{k-2} + X(\Theta^{-1} - I_\ell)X^T - X\omega g_{l+1}^T - g_{l+1}\omega^T X^T + X\omega\omega^T X^T} \tag{3.65}$$

where the vector $g_{l+1}$ is defined by (3.54) for $i = l + 1$. Alternatively, this formula can be used to construct the preconditioner $P_{k-1}$. In this case to obtain the matrix $X$, the matrix $G_\ell$ defined in Lemma 3.3 needs to be generated and stored. Once we have all the required information from PCG Algorithm 2.5, Algorithm 3.11 summarizes how to built the Ritz-LMP (3.65).

In this section we are left with the task of deriving the Ritz LMP for Approach (C).

### Approach (C) with the Ritz LMP

We now consider applying PCGIF Algorithm 3.4 to the linear system (3.14) of the form of $\underline{A}\,\underline{s} = b_k$ within Algorithm 3.5.

We suppose that $P_0 = BC_0$ where $P_0$ is the initial preconditioner for Algorithm 3.2 and $C_0$ is the initial preconditioner for Algorithm 3.5. Our aim is to find the preconditioner $C_{k-1}$ for the $k$-th linear system of Algorithm 3.5 such that $P_{k-1} = BC_{k-1}$ where $P_{k-1}$ is the Ritz-LMP constructed from (3.64) and used for the $k$-th linear system of Algorithm 3.2.

Assume that while solving the $(k-1)$-th linear systems in Algorithm 3.2 and 3.5

**Algorithm 3.11: Construct the Ritz-LMP for the matrix $A$ (version 2)**

1 Given $P_{k-2}$, $G_\ell$, $T_\ell$, $g_{\ell+1}$ and $\beta_{\ell+1}$ from PCG Algorithm 2.5

2 Calculate the eigenpairs $(\theta_i, \bar{u}_i)$, $i = 1, \ldots, \ell$ of the tridiagonal matrix $T_\ell$:

$$T_\ell \, \bar{U} = \Theta \, \bar{U}$$

where $\Theta = diag(\theta_i)$ and $\bar{U} = [\bar{u}_1, \ldots, \bar{u}_\ell]$.

3 Generate the vector $\omega = (\omega_1, ..., \omega_\ell)^T$ with $\omega_i$ calculate from the formula (3.48)

4 Calculate the Ritz pairs $(\theta_i, x_i)$ of the matrix $PA$:

$$(\Theta, X) = (\Theta, G_l \bar{U})$$

where $X = [x_1, \ldots, x_\ell]$.

5 Construct the Ritz-LMP from the formula (3.65)

the relation $P_{k-2} = BC_{k-2}$ holds. Substituting this relation into (3.64) gives that

$$\begin{aligned} P_{k-1} &= BC_{k-2} + BC_{k-2}U(\Theta^{-1} - I_\ell)U^T C_{k-2}^T B - BC_{k-2}U\omega v_{l+1}^T C_{k-2}^T B \\ &\quad - BC_{k-2}v_{l+1}\omega^T U^T C_{k-2}^T B + BC_{k-2}U\omega\omega^T U^T C_{k-2}^T B \\ &= B[C_{k-2} + C_{k-2}U(\Theta^{-1} - I_\ell)U^T C_{k-2}^T B - C_{k-2}U\omega v_{l+1}^T C_{k-2}^T B \\ &\quad - C_{k-2}v_{l+1}\omega^T U^T C_{k-2}^T B + C_{k-2}U\omega\omega^T U^T C_{k-2}^T B], \end{aligned}$$

from which we can write the preliminary formula for $C_{k-1}$ as

$$\begin{aligned} C_{k-1} &= C_{k-2} + C_{k-2}U(\Theta^{-1} - I_\ell)U^T C_{k-2}^T B - C_{k-2}U\omega v_{l+1}^T C_{k-2}^T B \\ &\quad - C_{k-2}v_{l+1}\omega^T U^T C_{k-2}^T B + C_{k-2}U\omega\omega^T U^T C_{k-2}^T B. \end{aligned} \tag{3.66}$$

This formula requires the information from PCG Algorithm 2.5. Similar to the derivation of the preconditioner $P_{k-1}$, we first explain by Lemma 3.5 and 3.6 how to extract the required information from PCGIF Algorithm 3.4 and later reformulate the LMP (3.66) in terms of vectors and matrices that are directly available from PCGIF Algorithm.

**Lemma 3.5** *Let $v_i$, $i = 1, \ldots, l$ be $\ell$ Lanczos vectors generated from (3.51) during PCG Algorithm 2.5 applied to the preconditioned linear system $As = b$ with a symmetric and positive definite matrix $P$. Then the Lanczos vectors $\underline{v}_i$, $i = 1, \ldots, l$ generated from the formula*

$$\underline{v}_{i+1} = (-1)^i \frac{\underline{z}_i}{\sqrt{\underline{r}_i^T z_i}} \tag{3.67}$$

*during PCGIF Algorithm 3.4 applied to the linear system $AB\underline{s} = b$ with the preconditioner $C$ such that $P = BC$ satisfy the relation*

$$\underline{v}_i = C v_i. \tag{3.68}$$

*Moreover, the vectors defined by*

$$g_i = B\underline{v}_i, \tag{3.69}$$

*coincide with those given in Lemma 3.3.*

**Proof.** Under the assumption that $P = BC$, the relation between the residuals of Algorithms 2.5 and 3.4 is given as

$$\underline{r}_i = r_i.$$

Multiplying (3.51) from left by $C$ and substituting the relation between the residuals gives the desired result $\underline{v}_i = Cv_i$.

Substituting the relations $P = BC$ and $\underline{v}_i = Cv_i$ in (3.54) we obtain that

$$g_i = BCv_i = B\underline{v}_i$$

which completes the proof. □

**Lemma 3.6** *Let $(\theta_i, u_i)$, $i = 1, \ldots, l$ be the Ritz pairs of the matrix $AP$ with respect to the subspace $\mathcal{K}^\ell(AP, r_0)$ defined in Lemma 3.4. Let us define the vectors $\underline{u}_i$ such that*

$$\underline{u}_i = \underline{V}_\ell \bar{u}_i \tag{3.70}$$

*where $\underline{V}_\ell = [\underline{v}_1, \ldots, \underline{v}_\ell]$ and its column vectors are defined in Lemma 3.5. Then the vectors $\underline{u}_i$ satisfy the relation*

$$\underline{u}_i = Cu_i. \tag{3.71}$$

**Proof.** Multiplying the equation (3.59) from right by C and using the relation (3.68) gives that

$$Cu_i = CV_\ell \bar{u}_i = \underline{V}_\ell \bar{u}_i$$

from which it can be seen that the vectors defined by (3.70) satisfy the relation (3.71). □

We now use the vectors and relations given by Lemma 3.5 and Lemma 3.6 to reformulate the LMP given by (3.66) in the sense that all the information can be obtained directly from PCGIF Algorithm 3.4.

Let us define the matrix $\underline{U} = [\underline{u}_1, \ldots, \underline{u}_\ell]$ whose columns are defined in Lemma 3.6. Substituting the relations (3.71) for $i = 1, \ldots, \ell$ into (3.66) gives that

$$C_{k-1} = C_{k-2} + \underline{U}(\Theta^{-1} - I_\ell)\underline{U}^T B - \underline{U}\omega\underline{v}_{l+1}^T B - \underline{v}_{l+1}\omega^T\underline{U}^T B + \underline{U}\omega\omega^T\underline{U}^T B, \tag{3.72}$$

where the vector $\underline{v}_{l+1}$ is defined by (3.68) for $i = \ell + 1$. The formula (3.72) requires matrix-vector multiplications with $B$ which may appear computationally costly. However, we can avoid this matrix-vector multiplications by observing that

$$B\underline{u}_i = BCu_i = Pu_i = x_i.$$

where $x_i$ is defined in Lemma 3.4. Substituting this relation and the vector $g_{l+1}$ defined by (3.69) for $i = l + 1$ into (3.72) yields that

$$\boxed{C_{k-1} = C_{k-2} + \underline{U}(\Theta^{-1} - I_\ell)X^T - \underline{U}\omega g_{l+1}^T - \underline{v}_{l+1}\omega^T X^T + \underline{U}\omega\omega^T X^T.} \tag{3.73}$$

In order to obtain the vector $\omega$ and the matrices $\underline{U}$, $X$ and $\Theta$ in the formula (3.73), the matrices $\underline{V}_\ell$,$G_\ell$ and $T_\ell$ and the scalar $\beta_{\ell+1}$ needs to be generated and stored. The column vectors of the matrix $V_\ell$ and $G_\ell = [g_1, \ldots, g_\ell]$ are defined in Lemma 3.5. The matrix $T_\ell$ is obtained as explained in Section 2.6.4 and the scalar $\beta_{\ell+1}$ can be obtained from formula (2.36) where $\eta_{\ell+1} = \beta_{\ell+1}$. Once we have all the required information from PCGIF Algorithm 3.4, Algorithm 3.12 summarizes how to built the the Ritz-LMP (3.73).

**Algorithm 3.12: Construct the Ritz-LMP for the matrix $\underline{A}$**

1. Given $C_{k-2}$, $\underline{V}_\ell$, $G_\ell$, $T_\ell$, $\underline{v}_{\ell+1}$, $g_{\ell+1}$ and $\beta_{\ell+1}$ from PCGIF Algorithm 3.4
2. Calculate the eigenpairs $(\theta_i, \bar{u}_i)$, $i = 1, \ldots, \ell$ of the tridiagonal matrix $T_\ell$:

$$T_\ell \, \bar{U} = \Theta \, \bar{U}$$

   where $\Theta = diag(\theta_i)$ and $\bar{U} = [\bar{u}_1, \ldots, \bar{u}_\ell]$.
3. Generate the vector $\omega = (\omega_1, ..., \omega_\ell)^T$ with $\omega_i$ calculate from the formula (3.48)
4. Generate the matrix $\underline{U} = [\underline{u}_1, \ldots, \underline{u}_\ell]$ whose column vectors are calculated from

$$\underline{u}_i = \underline{V}_l \bar{u}_i$$

5. Generate the matrix $X = [x_1, \ldots, x_\ell]$ whose column vectors are calculated from

$$x_i = G_l \bar{u}_i$$

6. Construct the Ritz-LMP from the formula (3.73)

*As a result it is possible to find the corresponding Ritz LMPs for variants of conjugate gradients (Approaches (A), (B) and (C)) such that when they are used within the corresponding algorithms (Algorithms 3.1, 3.2 and 3.5), these algorithms generate a mathematically equivalent sequence of iterates $\{s_k\}$. We showed that these preconditioners can be constructed from vectors which are by-products of the corresponding algorithms (Algorithms 2.2, 2.5 and 3.4).*

Table 3.5 summarizes the characteristics of Algorithms 3.1, 3.2 and 3.5 when used with the Ritz LMP and Table 3.6 summarizes the cost of applying the preconditioner and memory requirements.

Using the information from Table 3.6 together with the cost of $B^{1/2}$ and $B^{-1}$, it can be decided which algorithm is more efficient in terms of the computational cost and which is to be chosen to solve the sequence of linear systems (3.1).

| Algorithm | Inner min. | Ritz LMP formula | Assumption |
|---|---|---|---|
| 3.1 | Approach (A) | $F_k$ given by (3.49) | $F_0$ is available |
| 3.2 | Approach (B) | $P_k$ given by (3.65) | |
| 3.5 | Approach (C) | $C_k$ given by (3.73) | $B^{-1}P_0$ is available |

Table 3.5: A summary of the characteristics of Algorithms 3.1, 3.2 and 3.5 where preconditioning is achieved by using the Ritz LMPs.

| Approach | LMP | Ref. | Memory req. | Flops req. | mat-vec prod. |
|---|---|---|---|---|---|
| (A) | $F$ and $F^T$ | (3.49) | $(n+2)l+n$ | $\approx (10ln+5n)$ | |
| (B) | $P$ | (3.65) | $(n+2)l+n$ | $\approx (8ln+4n)$ | $P_{k-2}$ |
| (C) | $C$ and $C^T$ | (3.73) | $2(n+1)l+2n$ | $\approx 2(8ln+4n)$ | $C_{k-2}$ and $C_{k-2}^T$ |

Table 3.6: Memory and cost requirements for applying the Ritz LMPs during Approaches (A), (B) and (C)

### 3.1.3 Preconditioning with the spectral LMP

In this section, we recall the spectral LMP (2.43) and adapt this preconditioner that can be used during Algorithms 3.1, 3.2 and 3.5.

We start with explaining how to use a spectral LMP within Algorithm 3.1 in which the linear system in sequence is solved by Approach (A).

#### Approach (A) with the spectral LMP

Let us consider applying CG Algorithm 2.2 to the linear system (3.7) of the form of $\widetilde{A}_k\ \widetilde{s} = \widetilde{b}_k$ within Algorithm 3.1. Let us denote the exact eigenpairs of the system matrix $\widetilde{A}_{k-1}$ in the $(k-1)$-th linear system by $(\lambda_i, \tilde{v}_i)$, i.e.

$$\widetilde{A}_{k-1}\ \tilde{v}_i = \tilde{v}_i\ \lambda_i. \tag{3.74}$$

Then as explained in Section 2.6.5 by using the eigenpairs $(\lambda_i, \tilde{v}_i)$, the preconditioner $Y_{k-1}$ that approximates the inverse of $\widetilde{A}_{k-1}$ can be calculated as

$$Y_{k-1} = I_n - \widetilde{U}(\Lambda^{-1} - I_l)\widetilde{U}^T, \tag{3.75}$$

where $\widetilde{U} = [\widetilde{v}_1, ..., \widetilde{v}_l]$, $\Lambda = diag(\lambda_i)$.

While applying the preconditioner during Algorithm 2.2 the factorization of $Y_{k-1}$ is required. This factored form $Y_{k-1} = F_{k-1}F_{k-1}^T$ can be calculated from (Gratton, Sartenaer and Tshimanga, 2011)

$$\boxed{F_{k-1} = I_n - \widetilde{U}(\Lambda^{-1/2} - I_l)\widetilde{U}^T}. \tag{3.76}$$

We now explain how to construct a spectral LMP within Algorithm 3.2 in which

the linear system in sequence is solved by Approach (B).

### Approach (B) with the spectral LMP

Let us consider applying PCG Algorithm 2.5 to the linear system (3.1) of the form of $As = b_k$ within Algorithm 3.2. The spectral LMP $P_{k-1}$ can be derived from (3.35) and (3.75) as

$$\begin{aligned} P_{k-1} &= P_{k-2}^{1/2} Y_{k-1} P_{k-2}^{1/2} \\ &= P_{k-2} + P_{k-2}^{1/2} \widetilde{U}(\Lambda^{-1} - I_l)\widetilde{U}^T P_{k-2}^{1/2}. \end{aligned} \tag{3.77}$$

Using the relation $\widetilde{A}_{k-1} = P_{k-2}^{1/2} A P_{k-2}^{1/2}$ from (3.34) and the equation (3.74), it can be easily seen that the pairs $(\lambda_i, x_i)$ where $x_i$ is defined by

$$x_i = P_{k-2}^{1/2} \widetilde{v}_i,$$

are the eigenpairs of the matrix $P_{k-2}A$. Defining $X = [x_1, \ldots, x_\ell]$, the spectral LMP formula (3.77) can be rewritten in a more compact form as

$$\boxed{P_{k-1} = P_{k-2} + X(\Lambda^{-1} - I_l)X^T}. \tag{3.78}$$

In this section we are left with the task of deriving the spectral LMP for Algorithm 3.5.

### Approach (C) with the spectral LMP

We consider applying PCGIF Algorithm 3.4 to the linear system (3.14) of the form of $AB\,\underline{s} = b_k$ within Algorithm 3.5.

We suppose that $P_0 = BC_0$ where $P_0$ is the initial preconditioner for Algorithm 3.2 and $C_0$ is the initial preconditioner for Algorithm 3.5. Our aim is to find the preconditioner $C_{k-1}$ for the $k$-th linear system in Algorithm 3.5 such that $P_{k-1} = BC_{k-1}$ where $P_{k-1}$ is the spectral LMP constructed from (3.78) and used for the $k$-th linear system in Algorithm 3.2.

Assume that when solving the $(k-1)$-th linear system in Algorithm 3.2 and 3.5, the relation $P_{k-2} = BC_{k-2}$ holds. Let us define a matrix $\underline{U} = [\underline{u}_1, \ldots, \underline{u}_\ell]$ whose column vectors are defined by

$$\underline{u}_i = B^{-1} x_i. \tag{3.79}$$

Substituting this relation into (3.78) and using the relation $P_{k-2} = BC_{k-2}$ gives that

$$\begin{aligned} P_{k-1} &= BC_{k-2} + B\underline{U}(\Lambda^{-1} - I_l)\underline{U}^T B \\ &= B\,[C_{k-2} + \underline{U}(\Lambda^{-1} - I_l)\underline{U}^T B], \end{aligned}$$

from which the preconditioner $C_{k-1}$ can be written as

$$\boxed{C_{k-1} = C_{k-2} + \underline{U}(\Lambda^{-1} - I_l)X^T}. \tag{3.80}$$

*As a result it is possible to find the corresponding spectral LMPs for variants of conjugate gradients (Approaches (A), (B) and (C)) such that when they are used within the corresponding Algorithms 3.1, 3.2 and 3.5, these algorithms generate a mathematically equivalent sequence of iterates* $\{s_k\}$.

We summarize the characteristics of Algorithms 3.1, 3.2 and 3.5 when used with the spectral LMP in Table 3.7, and the cost of applying the spectral preconditioner with its memory requirements in Table 3.8.

| Algorithm | inner min. | spectral LMP formula | Assumption |
|---|---|---|---|
| 3.1 | App. (A) | $F_k$ given by (3.76) | $F_0$ is available |
| 3.2 | App. (B) | $P_k$ given by (3.78) | |
| 3.5 | App. (C) | $C_k$ given by (3.80) | $B^{-1}P_0$ is available |

Table 3.7: A summary of the characteristics of Algorithms 3.1, 3.2 and 3.5 where preconditioning is achieved by using the spectral LMPs.

| Approach | LMP | Ref. | Memory req. | Flops req. | mat-vec prod. |
|---|---|---|---|---|---|
| (A) | $F$ and $F^T$ | (3.76) | $(n+1)l$ | $\approx 8ln$ | |
| (B) | $P$ | (3.78) | $(n+1)l$ | $\approx 4ln$ | $P_{k-2}$ |
| (C) | $C$ and $C^T$ | (3.80) | $(2n+1)l$ | $\approx 8ln$ | $C_{k-2}$ and $C_{k-2}^T$ |

Table 3.8: Memory and cost requirements for applying the spectral LMPs during Approaches (A), (B) and (C)

The spectral LMP requires to compute the eigenpairs of the system matrix which can not be afordable for large-scale systems. For example, (Fisher, 1998) propose to replace the exact eigenpairs in the formula (3.75) by the Ritz pairs called as the inexact spectral LMP in Tshimanga, Gratton, Weaver and Sartenaer (2008). In this study, they compared two LMPs and concluded that the Ritz LMP is a general and stabilized version of the inexact spectral LMP.

### 3.1.4 Preconditioning varying systems

As mentioned before, the solution of the nonlinear least-squares problem (2.4) is found by solving a sequence of slowly varying linear systems (2.16) in the form of

$$A_k s = b_k. \tag{3.81}$$

We want to understand if it is possible to precondition also these varying linear systems by using the previous information. From $A_k = B^{-1} + H_k^T R^{-1} H_k$, we assume that $B$

and $R$ are constant and $H_k$ is changing along the outer loop $k$ due to the linearization of the nonlinear model $\mathcal{H}(x_k)$.

Let us assume that the LMP $P_{k-1}$ which is an inverse approximation of the linear system matrix $A_{k-1}$ is inherited to precondition the linear system $A_k s = b_k$ at the $k$-th outer loop of the Gauss-Newton method. Then CG applies on the preconditioned system

$$P_{k-1}^{1/2} A_k P_{k-1}^{1/2} s = P_{k-1}^{1/2} b_k. \tag{3.82}$$

Defining $A_k = A_{k-1} + E_k$, from (Golub and Van Loan, 1996, Corollary 8.1.6) we have

$$|\lambda_i(P_{k-1}^{1/2} A_k P_{k-1}^{1/2}) - \lambda_i(P_{k-1}^{1/2} A_{k-1} P_{k-1}^{1/2})| \leq \|P_{k-1}^{1/2} E_k P_{k-1}^{1/2}\|_2 \tag{3.83}$$

for $i = 1 : n$. This corollary gives us the information on the condition number of the perturbed matrix $P_{k-1}^{1/2} A_k P_{k-1}^{1/2}$ as follows.

Let us denote the eigenvalues of the matrix $P_{k-1}^{1/2} A_{k-1} P_{k-1}^{1/2}$ by $\lambda_i$ and the eigenvalues of the perturbed matrix $P_{k-1}^{1/2} A_k P_{k-1}^{1/2}$ by $\widetilde{\lambda}_i$. For each eigenvalue it follows from the inequality (3.83) that

$$\begin{aligned} |\widetilde{\lambda}_i - \lambda_i| &\leq \|P_{k-1}\|_2 \|E_k\|_2, \\ -\|P_{k-1}\|_2 \|E_k\|_2 \leq \widetilde{\lambda}_i - \lambda_i &\leq \|P_{k-1}\|_2 \|E_k\|_2, \\ \lambda_i - \|P_{k-1}\|_2 \|E_k\|_2 \leq \widetilde{\lambda}_i &\leq \lambda_i + \|P_{k-1}\|_2 \|E_k\|_2, \end{aligned}$$

Using this equality for the minimum and maximum eigenvalues gives that

$$\lambda_{min} - \|P_{k-1}\|_2 \|E_k\|_2 \leq \widetilde{\lambda}_{min} \leq \lambda_{min} + \|P_{k-1}\|_2 \|E_k\|_2, \tag{3.84}$$

$$\lambda_{max} - \|P_{k-1}\|_2 \|E_k\|_2 \leq \widetilde{\lambda}_{max} \leq \lambda_{max} + \|P_{k-1}\|_2 \|E_k\|_2. \tag{3.85}$$

Assuming that $(\lambda_{min} - \|P_{k-1}\|_2 \|E_k\|_2) > 0$, we can obtain from inequalities (3.84) and (3.85) that

$$\frac{\lambda_{max} - \|P_{k-1}\|_2 \|E_k\|_2}{\lambda_{min} + \|P_{k-1}\|_2 \|E_k\|_2} \leq \frac{\widetilde{\lambda}_{max}}{\widetilde{\lambda}_{min}} \leq \frac{\lambda_{max} + \|P_{k-1}\|_2 \|E_k\|_2}{\lambda_{min} - \|P_{k-1}\|_2 \|E_k\|_2}. \tag{3.86}$$

The inequality (3.86), therefore gives a bound on the condition number of the preconditioned perturbed matrix. Then, provided that $\|E_k\|_2$ is small enough, the inherited preconditioner can be still used as a second level preconditioner if (3.86) is satisfactory. Note that, (Tshimanga, 2007) also analyzes the effect of using inherited preconditioner to precondition the new system when $E_k$ is a first-order perturbation of the matrix $A_k$ and provide first-order bounds for the perturbed preconditioned matrix (Tshimanga, 2007, Theorem 2.5.1).

If the analysis described above is not ensuring that $P_{k-1}$ is a good preconditioner for $A_k$, we have to build a preconditioner $P_k$ using the LMP formula (2.38) with the new matrix $A_k$. In this case, considering the directions $S$ inherited from the $(k-1)$-th outer loop, a preconditioner is obtained with the new matrix $A_k$ which has the good properties (e.g. clustering most eigenvalues at 1) that any LMP preconditioner

has. This generation however requires $\ell$ matrix-vector products with the new matrix $A_k$ and is interesting if the preconditioning effect balances the cost of building the preconditioner.

## 3.2 Convergence issues

The convergence properties of CG are well known and for the detail information we refer to the references Nocedal and Wright (2006), Meurant (2006) and Axelsson (1996). When PCG is used with a class of LMPs, the convergence properties are provided in Gratton, Sartenaer and Tshimanga (2011) and Tshimanga (2007).

In this section we provide inexpensive formulas for computing the necessary information on each iteration from the recurrence relations of PCG for monitoring the convergence of the minimization.

### 3.2.1 Monitoring convergence

The values of the quadratic cost function and the norm of the cost function gradient are important for monitoring the convergence of the minimization. In addition to the total value of the quadratic cost function $Q(s)$ given in (2.46), the relative contributions to $(Q)$ from the background term $(Q_b)$ and observation term $(Q_o)$ may provide additional useful diagnostic information. Inexpensive formulae for computing all these quantities on each iteration from the recurrence relations of PCG and PCGIF (Algorithms 2.5 and 3.4) are derived in this section. Note that for the sake of simplicity, the initial guess is chosen as $s_0 = 0$ for all derivations.

#### Quadratic cost function

Using a Taylor series expansion about the initial guess $s_0 = 0$, for PCG the quadratic cost function $Q[s_i]$ in (2.46) can be expressed as (Gratton and Tshimanga, 2009)

$$Q[s_i] = Q[0] - \frac{1}{2} s_i^T r_0, \tag{3.87}$$

where $i = 0, 1, ..., l-1$, $l$ is the number of iterations performed during PCG and

$$Q[0] = f(x_k) = \frac{1}{2} d^T R^{-1} d + \frac{1}{2}(x_c - x_k)^T B^{-1}(x_c - x_k).$$

The $Q_b$ term can be calculated as

$$\begin{aligned} Q_b[s_i] &= \frac{1}{2}(s_i + x_k - x_c)^T B^{-1}(s_i + x_k - x_c) \\ &= \frac{1}{2} s_i^T B^{-1} s_i + \frac{1}{2}(x_k - x_c)^T B^{-1}(x_k - x_c). \end{aligned} \tag{3.88}$$

The $Q_o$ term can then be calculated from

$$Q_o[s_i] = Q[s_i] - Q_b[s_i]. \tag{3.89}$$

For PCGIF the same formula can be used for the quadratic cost function value $Q[s_i]$ and $Q_o[s_i]$, on the other hand $Q_b[s_i]$ can alternatively calculated from

$$Q_b[s_i] = \frac{1}{2} s_i^T f_i + \frac{1}{2}(x - x_c)^T B^{-1}(x - x_c)$$

where $f_i = B^{-1} s_i$ can be computed without the need to apply $B^{-1}$ by including an additional recurrence relation in Algorithm 3.4 as follows:

$$f_i = \begin{cases} 0 & \text{if } i = 0 \\ f_{i-1} + \alpha_{i-1} \widetilde{p}_{i-1} & \text{if } i > 0 \end{cases}$$

Taking $x_1 = x_c$ and using the vector $f_i$, the term $B^{-1}(x_k - x_c)$ can also be calculated from

$$B^{-1}(x_k - x_c) = \sum_{j=1}^{k-1} B^{-1} s_k = \sum_{j=1}^{k} f_{l-1}^{(j)}$$

where $f_{l-1}^{(j)}$ denotes the last iteration ($i = l - 1$) of PCGIF at the $j$-th outer loop of Gauss-Newton method.

### Norm of the quadratic cost function gradient

For PCG, the gradient norm is given by

$$\|\nabla Q[s_i]\|_P = \|r_i\|_P = \sqrt{r_i^T z_i}$$

where the vectors $z_i$ and $r_i$ are defined in Algorithm 2.5. The gradient norm for PCGIF can be calculated equivalently from the same formula.

# CHAPTER 4

# Conjugate gradients in dual space

In this chapter, the least-squares subproblems (2.46) solved at each iteration of the GN method is rewritten as a quadratic minimization problem subject to linear equality constraints. Contrary to the primal constrained problem introduced in Section 2.7, the structure of the subproblem (2.46) is exploited to derive an alternative dual problem.

We consider a recently-proposed CG-like method to perform the quadratic minimization arising from the alternative dual approach. This method produces, in exact arithmetic, the same iterates as those produced by a standard CG applied to the linear system (2.16). At first sight, even if the dimension may not explain everything, this algorithm might provide some advantage when the dual problem is defined on a space (named *the dual space*) of smaller dimension than that of the primal problem (named *the primal space*) as it can yield gains in terms of both memory usage and computational cost. The relation between this dual-space solver and existing dual-space techniques used in data assimilation problems is explained.

We also introduce practical warm-start preconditioners to accelerate the convergence of the dual algorithm when solving a convergent sequence of linear systems with multiple right hand sides. In particular, a dual-space counterpart to the warm-start LMPs explained in Chapter 3 is derived and its properties analyzed. We later adapt the preconditioners to the case where a sequence of slowly varying linear systems has to be solved as in the Gauss-Newton process.

We conclude the chapter by providing the convergence theory for the dual algorithm and inexpensive formulae to compute the quadratic cost function (2.13) and the norm of its gradient from by-products of the dual algorithm, that are useful quantities to monitor convergence.

## 4.1 Exploiting further the structure with the dual problem

In this section, we take into consideration the structure of the regularized subproblem (2.46) and derive its dual problem which has its Lagrange multiplier vector in $\mathbb{R}^m$,

the space spanned by the $m$-dimensional vectors associated by the data vector.

We reformulate the linear least-squares problem (2.46) as a convex quadratic problem with linear equality constraints given by

$$\begin{aligned} &\min_{s,a} \frac{1}{2}\|s + x_k - x_c\|^2_{B^{-1}} + \frac{1}{2}\|a\|^2_{R^{-1}} \\ &\text{subject to} \quad a = H_k s + d_k, \end{aligned} \tag{4.1}$$

which can, in turn, be solved using duality theory. We define the *dual objective* function

$$q : \mathbb{R}^m \to \mathbb{R}$$

for this problem as (Nocedal and Wright, 2006, p.343-349)

$$q(\lambda) = \inf_{s,a} \mathcal{L}(s,a,\lambda) \stackrel{\text{def}}{=} \inf_{s,a} \frac{1}{2}\|s + x_k - x_c\|^2_{B^{-1}} + \frac{1}{2}\|a\|^2_{R^{-1}} - \lambda^T(H_k s + d_k - a), \tag{4.2}$$

where $\lambda \in \mathbb{R}^m$ is a Lagrange multiplier vector. Note that contrary to Chapter 2, the Lagrange multiplier is not associated with the full residual of (4.1), it is associated only with the part related to the observations. The infimum is achieved when

$$\begin{aligned} \nabla_s \mathcal{L}(s,a,\lambda) &= B^{-1}(s + x_k - x_c) - H_k^T\lambda = 0 \\ \nabla_a \mathcal{L}(s,a,\lambda) &= R^{-1}a + \lambda = 0 \\ \nabla_\lambda \mathcal{L}(s,a,\lambda) &= H_k s + d_k - a = 0 \end{aligned}$$

From the first two equations we obtain that

$$\begin{aligned} s &= x_c - x_k + BH_k^T\lambda, \\ a &= -R\lambda. \end{aligned} \tag{4.3}$$

We may therefore substitute $s$ and $a$ in the expression (4.2) and obtain the quadratic dual objective explicitly as follows:

$$q(\lambda) = -\frac{1}{2}\lambda^T(H_k B H_k^T + R)\lambda + \lambda^T(H_k(x_k - x_c) - d_k),$$

Then the *dual problem* for the *constrained primal problem* (4.1) is defined as

$$\max_{\lambda} \; q(\lambda) \tag{4.4}$$

The maximum of the dual objective is given by

$$\lambda^* = (H_k B H_k^T + R)^{-1}(H_k(x_k - x_c) - d_k), \tag{4.5}$$

which is found as a stationary point. Therefore, from (4.5) and (4.3), the solution of the subproblem (2.46) can be written as

$$s_k = x_c - x_k + BH_k^T(H_k B H_k^T + R)^{-1}(H_k(x_k - x_c) - d_k). \tag{4.6}$$

Note that this solution may also be obtained directly from

$$s_k = x_c - x_k + (H_k^T R^{-1} H_k + B^{-1})^{-1} H_k^T R^{-1} (H_k(x_k - x_c) - d_k)$$

(written from (2.50)) by using the Sherman-Morrison-Woodbury formula (Nocedal and Wright, 2006, p. 612-613).

This suggests that the step $s_k$ can be obtained by solving the dual problem (4.4) in the *dual space* of Lagrange multiplier and recovering the step from (4.3). We name this approach the *dual approach*. This approach solves the dual problem by applying the Krylov subspace method to the linear system in (4.5) given by

$$(H_k B H_k^T + R)\lambda = H_k(x_k - x_c) - d_k. \tag{4.7}$$

In this chapter, we focus on solving this linear system with preconditioned conjugate gradients such that the iterates in $\mathbb{R}^n$ generated from (4.3) ensures the monotonic decrease on the quadratic cost function (2.46) along the inner-iterations.

Before explaining how to apply PCG in the context of dual approach, we next provide the link between applying CG method on the linear system (4.7) and the variants of the CRAIG method.

## 4.2 Relation with Extended CRAIG and Generalized CRAIG

The extended CRAIG proposed by Saunders (1995) suggests to extend the CRAIG method (Craig, 1955) for incompatible under determined systems where $m \ll n$ by including regularization. It is observed that the extended CRAIG solves the equivalent problem to that of (2.47) which can be stated as (Saunders, 1995)

$$\min_{\delta\widetilde{x},z} \ \frac{1}{2} \left\| \begin{bmatrix} \delta\widetilde{x} \\ z \end{bmatrix} \right\|_2^2, \tag{4.8}$$

subject to

$$\begin{bmatrix} \widetilde{H} & I \end{bmatrix} \begin{bmatrix} \delta\widetilde{x} \\ z \end{bmatrix} = \widetilde{c}$$

where $z = \widetilde{c} - \widetilde{H}\delta\widetilde{x}$. It is pointed out (Saunders, 1995, Result 9) that since $\widetilde{H}^T\widetilde{H} + I_n$ is symmetric and positive definite, the extended CRAIG iterates $\delta\widetilde{x}_i$, $i = 0, \ldots, l-1$ are related to the CG iterates for the problem

$$(\widetilde{H}\widetilde{H}^T + I_m)\widetilde{\lambda} = \widetilde{c} \tag{4.9}$$

$$by \qquad \begin{pmatrix} \delta\widetilde{x} \\ z \end{pmatrix} = \begin{pmatrix} \widetilde{H}^T \\ I \end{pmatrix} \widetilde{\lambda}$$

according to $\delta\widetilde{x}_i = \widetilde{H}^T\widetilde{\lambda}_i$ and $z = \widetilde{\lambda}$. Substituting $\widetilde{H} = R^{-1/2}HB^{1/2}$ and $\widetilde{c} = R^{-1/2}H(x - x_c) - R^{-1/2}d$ into (4.9) yields that

$$(R^{-1/2}HBH^TR^{-1/2} + I_m)\widetilde{\lambda} = R^{-1/2}(H(x - x_c) - d).$$

Note that this linear system is the same linear system as (4.7) preconditioned by $R^{-1/2}$. Therefore the iterates are related according to $R^{-1/2}\widetilde{\lambda}_i = \lambda_i$.

The extended CRAIG is generalized by the Generalized CRAIG (G-CRAIG) proposed by Arioli and Orban (2012); this method uses a regularization term different from $(\tau I)$ and a non-canonical inner product. The G-CRAIG solves the equivalent problem to (2.47) which can be given as

$$\min_{r,\delta x} \frac{1}{2} \left\| \begin{bmatrix} \delta x \\ r \end{bmatrix} \right\|_D^2,$$

subject to

$$\begin{bmatrix} H & R \end{bmatrix} \begin{bmatrix} \delta x \\ r \end{bmatrix} = c$$

where $\delta x = B^{1/2}\delta\widetilde{x}$, $r = R^{-1/2}z$ , $c = R^{1/2}\widetilde{c}$ and the $D$-norm is defined by the matrix

$$D := \begin{bmatrix} B^{-1} & \\ & R \end{bmatrix}.$$

The G-CRAIG iterates $\delta x_i$, $i = 0, \ldots, l-1$ are related to the PCG iterates for the problem

$$\begin{aligned} (HBH^T + R)\lambda &= H(x - x_c) - d \\ \begin{pmatrix} \delta x \\ r \end{pmatrix} &= \begin{pmatrix} BH^T \\ I \end{pmatrix} \lambda \end{aligned} \tag{4.10}$$

with preconditioner $R^{-1}$ according to $\delta x_i = BH^T\lambda_i$ (Arioli and Orban, 2012). Note that the linear system (4.10) is the same linear system as (4.7).

The extended CRAIG and G-CRAIG minimize the direct error

$$\left\| \begin{bmatrix} \delta x \\ r \end{bmatrix} - \begin{bmatrix} \delta x_* \\ r_* \end{bmatrix} \right\|_D,$$

where $\delta x_*$ is the exact solution and $r_* = R^{-1}(c - H\delta x_*)$.

Since the extended CRAIG and G-CRAIG are more expensive than LSQR (Saunders, 1995), (Arioli and Orban, 2012) we will not consider to use these methods within this study.

## 4.3 Solving the subproblem with Restricted PCG (RPCG)

In this section, we explain variants of dual approaches which solve the linear system (4.7) by using conjugate-gradients like methods with a preconditioner. In particular, we focus on the dual approach which produces, in exact arithmetic, the same iterates as those produced by a standard CG applied to the linear system (2.16). This method therefore preserves the monotonic decrease on the quadratic cost function (2.46). Note that within this section we have dropped the outer loop index $k$ of a Gauss Newton method for simplicity.

A straightforward alternative for solving the system (4.7) with a preconditioner is to apply PCG Algorithm 2.5 with a preconditioner matrix $R^{-1}$ and a canonical inner product. This gives the preconditioned system

$$(R^{-1/2}HBH^TR^{-1/2} + I_m)\,\widetilde{\lambda} = R^{-1/2}(H(x - x_c) - d), \tag{4.11}$$

where $\lambda = R^{-1/2}\widetilde{\lambda}$ and $I_m$ is the identity matrix of order $m$. The approximate solution in $\mathbb{R}^n$ is then computed from the relation (4.3) given by

$$s = x_c - x + BH^TR^{-1/2}\widetilde{\lambda}. \tag{4.12}$$

We mentioned in Section 4.2 that CG applied to the linear system (4.11) generates the iterates $\widetilde{\lambda}_i$ which are related to the iterates of the extended CRAIG applied to the problem (4.8) with $\delta\widetilde{x}_i = \widetilde{H}^T\widetilde{\lambda}_i$. Using this relation CG applied to the linear system (4.11) minimizes the direct error

$$\begin{aligned} q(\widetilde{\lambda}) &= < (\widetilde{H}\widetilde{H}^T + I_m)(\widetilde{\lambda} - \widetilde{\lambda}_*), (\widetilde{\lambda} - \widetilde{\lambda}_*) > \\ &= (\widetilde{\lambda} - \widetilde{\lambda}_*)^T\widetilde{H}\widetilde{H}^T(\widetilde{\lambda} - \widetilde{\lambda}_*) + (\widetilde{\lambda} - \widetilde{\lambda}_*)^T(\widetilde{\lambda} - \widetilde{\lambda}_*) \\ &= (\delta\widetilde{x} - \delta\widetilde{x}_*)^T(\delta\widetilde{x} - \delta\widetilde{x}_*) + (\widetilde{\lambda} - \widetilde{\lambda}_*)^T(\widetilde{\lambda} - \widetilde{\lambda}_*) \\ &= \|\delta x - \delta x_*\|^2_{B^{-1}} + \|\lambda - \lambda_*\|^2_R \end{aligned}$$

which can be written as

$$q(\delta x, \lambda) = \left\| \begin{bmatrix} \delta x \\ \lambda \end{bmatrix} - \begin{bmatrix} \delta x_* \\ \lambda_* \end{bmatrix} \right\|^2_D, \tag{4.13}$$

where $\widetilde{\lambda}_*$ denotes the exact solution of the linear system (4.11), $\delta\widetilde{x}_* = \widetilde{H}^T\widetilde{\lambda}_*$ denotes the exact solution of the problem (4.8), $\lambda_* = R^{-1/2}\widetilde{\lambda}_*$, $\delta x_* = B^{1/2}\delta\widetilde{x}_*$ and $\delta x = B^{1/2}\delta\widetilde{x}$. The norm $D$ is defined in Section 4.2.

This dual approach that applies CG on the linear system (4.11) in $\mathbb{R}^m$ and recovers the solution in $\mathbb{R}^n$ from the relation (4.12) is of interest in atmospheric and ocean data assimilation systems since $\mathbb{R}^m$ represents the physical space and when the number of available observations $m$ is much smaller than the number control variables $n$, this algorithm is more efficient in terms of computational cost. In ocean data assimilation this approach is known as the indirect representer method (Egbert, Bennett and Foreman,

1994) whereas in meteorological data assimilation it is called PSAS (Physical Space Assimilation System) (Courtier, 1997). Convergence properties of this dual approach is investigated by several authors (Akkraoui, Gauthier, Pellerin and Buis, 2008), (Gratton and Tshimanga, 2009) and it is shown that this dual approach produces iterates in the dual space of Lagrange multipliers (see Section 4.1), and that their corresponding primal-space counterparts (given by (4.12)) do not ensure monotonic decrease of the quadratic function (2.46) along the inner-iterations. Indeed, it turns out that this quadratic cost has an erratic behaviour, even on simple examples. From (4.13) it can also be seen that this dual approach is not minimizing the quadratic function (2.46), it minimizes the direct error for the vector $[\delta x^T, \lambda^T]^T$ in the $D$-norm. Thus, if the iterations are stopped before exact optimality is attained, there is no guarantee that the value of the quadratic cost has decreased, which may then negatively affect the convergence of the Gauss Newton method.

A better alternative is to ensure the decrease of the quadratic cost function (2.46) by using a non-canonical inner-product. It can be shown that the iterates obtained by using the $\widetilde{H}\widetilde{H}^T$-inner product within CG on the linear system (4.11):

$$(\widetilde{H}\widetilde{H}^T + I_m)\widetilde{\lambda} = \widetilde{c}$$

are related to standard CG on

$$(\widetilde{H}^T\widetilde{H} + I_n)\ \delta\widetilde{x} = \widetilde{H}^T\widetilde{c}$$

according to $\delta\widetilde{x}_i = \widetilde{H}^T\widetilde{\lambda}_i$. From this relation using CG on the linear system (4.11) with the $\widetilde{H}\widetilde{H}^T(= R^{-1/2}HBH^TR^{-1/2})$-inner product minimizes the cost function (2.47) which is equivalent to the quadratic cost function (2.46) under the assumption that the initial value satisfies $s_0 = x_c - x$. This assumption seems to be restrictive; however it can be overcome as explained in Section 4.4. Therefore, using this dual approach guarantees to have the same convergence properties as those of the primal approach.

This dual approach that uses a non-standard inner product in CG is the main idea behind the Restricted PCG (RPCG) method proposed by Gratton and Tshimanga (2009). RPCG solves the linear system

$$(R^{-1}HBH^T + I_m)\ \lambda = R^{-1}(H(x - x_c) - d) \tag{4.14}$$

with the (possibly semi-)definite $HBH^T$-(semi) inner product, in which the unsymmetric system matrix

$$\widehat{A} = R^{-1}HBH^T + I_m \tag{4.15}$$

becomes symmetric. Note that, RPCG generates the same iterates when applying standard CG Algorithm 2.2 on the linear system (4.7) with $R^{-1}$ preconditioner (applying CG Algorithm on the linear system (4.11)) using the $R^{-1/2}HBH^TR^{-1/2}$-inner product.

RPCG generates mathematically equivalent iterates to those of PCG Algorithm 2.5 applied to the linear system (2.50) with the canonical inner product and a precondi-

tioner matrix $B$. Under the assumption that $s_0 = x_c - x$, it also generates the same iterates as those obtained by applying PCG on the linear system (2.16) (*primal approach*) with a preconditioner matrix $B$. Note also that RPCG iterates are related with the iterates generated from CG on the linear system (4.11) with the $R^{-1/2}HBH^TR^{-1/2}$-inner product according to $\lambda_i = R^{-1/2}\widetilde{\lambda}_i$.

A suitable algorithm for RPCG with a preconditioner matrix $D$ can be deduced from the primal approach with a preconditioner matrix $P$ as follows.

It is shown in (Gratton and Tshimanga, 2009) that there exist $m$-dimensional vectors $\widehat{r}_i$, $\widehat{z}_i$, $\widehat{p}_i$, $\widehat{q}_i$, $\lambda_i$ that can be related to the corresponding $n$-dimensional vectors $r_i, z_i, p_i, q_i, s_i$ of the primal approach according to

$$\left.\begin{array}{rcl} r_i &=& H^T\widehat{r}_i, \\ z_i &=& BH^T\widehat{z}_i \\ p_i &=& BH^T\widehat{p}_i \\ q_i &=& H^T\widehat{q}_i \\ s_i &=& s_0 + BH^T\lambda_i \end{array}\right\} \tag{4.16}$$

where $i \geq 0$. Equations (4.16) allow all the recurrence relations in the primal approach involving vectors of dimension $n$ to be transformed directly into corresponding recurrence relations involving vectors of dimension $m$ as done in Algorithm 4.1. This derivation is detailed in (Gratton and Tshimanga, 2009) and uses two assumptions. The first assumption is related with the initial residual vector which is

**Assumption 4.1** $r_0 = H^T\widehat{r}_0$,

and holds if $s_0 = x_c - x$. Note that, the initial vector can also be chosen arbitrarily by using a generalized version of the RPCG algorithm in which augmented matrices and vectors are introduced (Gratton and Tshimanga, 2009). We postpone this discussion to Section 4.4 for the sake of simplicity. This assumption basically points out that accommodating the initial residual in the proper space is crucial when using the dual approach.

The second assumption is related to the preconditioner used in RPCG,

**Assumption 4.2** $PH^T = BH^TD$,

where $D$ is the preconditioner for RPCG and $P$ is the preconditioner for the primal approach. At first sight, this assumption may appear restrictive because such a preconditioner $D$ may not exist, in particular if, for some $P$, $PH^T$ is not included in the range of $BH^T$. However, it is shown in Gratton, Gürol and Toint (2013) that the widespread warm-start preconditioning techniques based on limited memory methods (Tshimanga, Gratton, Weaver and Sartenaer, 2008), (Morales and Nocedal, 1999) do satisfy this condition.

The mathematical equivalence of the primal approach and RPCG (under Assumptions 4.1 and 4.2) ensures the monotonic decrease of the quadratic cost (2.46) along the RPCG inner iterations. This feature makes RPCG preferable to the first alternative which reduces the direct error.

**Algorithm 4.1: PCG Algorithm in $\mathbb{R}^m$ (RPCG, version 1)**

1 $\lambda_0 = 0$
2 $\widehat{r}_0 = R^{-1}(H(x - x_c) - d)$
3 $\widehat{z}_0 = D\widehat{r}_0$
4 $\rho_0 =< \widehat{r}_0, \widehat{z}_0 >_{HBH^T}$
5 $\widehat{p}_0 = \widehat{z}_0$
6 **for** $i = 0, 1, ..., l-1$ **do**
7 $\quad \widehat{q}_i = (I_m + R^{-1}HBH^T)\widehat{p}_i$
8 $\quad \alpha_i = \rho_i / < \widehat{q}_i, \widehat{p}_i >_{HBH^T}$
9 $\quad \lambda_{i+1} = \lambda_i + \alpha_i \widehat{p}_i$
10 $\quad \widehat{r}_{i+1} = \widehat{r}_i - \alpha_i \widehat{q}_i$
11 $\quad \widehat{z}_{i+1} = D\widehat{r}_{i+1}$
12 $\quad$ Check convergence and stop if desired accuracy is reached
13 $\quad \rho_{i+1} =< \widehat{r}_{i+1}, \widehat{z}_{i+1} >_{HBH^T}$
14 $\quad \beta_i = \rho_{i+1}/\rho_i$
15 $\quad \widehat{p}_{i+1} = \widehat{z}_{i+1} + \beta_i \widehat{p}_i$
16 **end**
17 The solution is recovered from $s_l = x_c - x + BH^T\lambda_l$

However, the first version of the RPCG algorithm (stated as Algorithm 4.1) is expensive since it requires three matrix vector products involving $HBH^T$ for each inner loop. Fortunately, it can be rewritten in a much cheaper form by introducing additional dual-space vectors, reducing its cost per loop to a single matrix-vector product with $HBH^T$. More precisely, consider $w$ and $t$ defined by

$$w_i = HBH^T\widehat{z}_i \text{ and } t_i = HBH^T\widehat{p}_i, \tag{4.17}$$

where $\widehat{z}_i$ and $\widehat{p}_i$ are defined in Algorithm 4.1. If we multiply lines 5 and 15 of Algorithm 4.1 by $HBH^T$, we obtain that

$$t_i = \begin{cases} w_0 & \text{if } i = 0 \\ w_i + \beta_{i-1}t_{i-1} & \text{if } i > 0, \end{cases}$$

which yields the final version of RPCG (Algorithm 4.2).

As a summary, we outlined dual approaches which perform minimization in the data space $\mathbb{R}^m$ leading to algorithms that are computationally cheaper and require less memory comparing to that of primal approach when $m \ll n$. Once it is decided to solve the dual problem instead of the primal problem due to the computational and memory advantages, a special attention has to be paid on

- accommodating the initial residual in the right space, i.e. $\text{Im}(H^T)$, to ensure constructing the basis for the same solution subspace for dual and primal approaches,
- choosing the proper inner product, i.e. the $HBH^T$ inner product, in dual space to preserve monotonic decrease on the cost function (2.46),

**Algorithm 4.2: RPCG Algorithm**

$\lambda_0 = 0$
$\widehat{r}_0 = R^{-1}(H(x_c - x) - d)$
$\widehat{z}_0 = D\widehat{r}_0$
$\widehat{p}_0 = \widehat{z}_0$
$w_0 = HBH^T\widehat{z}_0$
$\rho_0 = \widehat{r}_0^T w_0$
$t_0 = w_0$
**for** $i = 0, 1, ..., l-1$ **do**
    $\widehat{q}_i = R^{-1}t_i + \widehat{p}_i$
    $\alpha_i = \rho_i / \widehat{q}_i^T t_i$
    $\lambda_{i+1} = \lambda_i + \alpha_i \widehat{p}_i$
    $\widehat{r}_{i+1} = \widehat{r}_i - \alpha_i \widehat{q}_i$
    $\widehat{z}_{i+1} = D\widehat{r}_{i+1}$
    $w_{i+1} = HBH^T\widehat{z}_{i+1}$
    Check convergence and stop if desired accuracy is reached
    $\rho_{i+1} = \widehat{r}_{i+1}^T w_{i+1}$
    $\beta_i = \rho_{i+1}/\rho_i$
    $\widehat{p}_{i+1} = \widehat{z}_{i+1} + \beta_i \widehat{p}_i$
    $t_{i+1} = w_{i+1} + \beta_i t_i$
**end**
The solution is recovered from $s_l = x_c - x + BH^T\lambda_l$

- choosing the proper preconditioner in dual space, i.e. a preconditioner that satisfies Assumption 4.2, to preserve one to one correspondance that keeps the desired monotonic convergence behaviour.

### 4.3.1 Preconditioning RPCG with LMPs

In Section 3.1 we present three approaches (Approaches (A), (B) and (C)) in primal space for preconditioning a sequence of linear systems given by (3.1). We assume that *this sequence produces a convergent sequence of solutions* within the corresponding algorithms (Algorithms 3.1, 3.2 and 3.5).

In this section, we explain in detail preconditioning RPCG with LMPs following the strategy of Approach (B) that does not require the factorization of the preconditioner. We later explain briefly how to use a split preconditioning in dual space. Note that, these approaches in dual space are developed such that the desired convergence properties in primal space are preserved. Since RPCG does not require matrix-vector products with $B^{-1}$, the third approach (Approach (C)) is not necessary with RPCG.

We consider two algorithms in primal space within the section, Algorithms 3.1 and 3.2. In these algorithms using the fact that the linear systems in sequence are not isolated and they produce a convergent sequence of solutions; the initial guess is zero, the right hand side is updated using the solution of the previous system, so that the minimization actually continues from the point obtained at the last iteration. By this

way along the sequence, some information is carried over from one CG run to the next, which yields faster convergence in the case where the right hand sides slowly vary along the sequence of linear systems.

We start with the dual approach that does not require the factorization of the preconditioner and generates mathematically equivalent iterates to those of Approach (B) with the zero initial guess.

### Approach (B) in dual space

In order to keep mathematically equivalent iterates between the dual and primal algorithm (Approach (B) with zero initial guess), we first need to ensure that Assumption 4.1 holds. We show that this assumption holds by transforming the linear system (4.14) into another linear system by using change of variables.

Let us now define the dual solution $\widehat{s}_k$ such that

$$\boxed{s_k = BH^T\widehat{s}_k}. \tag{4.18}$$

Using this vector the solution at the $k$-th outer loop can be written as

$$x_{k+1} = x_k + s_k = x_1 + \sum_{j=1}^{k} s_j = x_1 + BH^T \sum_{j=1}^{k} \widehat{s}_j.$$

Taking $x_1 = x_c$ gives that

$$x_{k+1} - x_c = BH^T \sum_{j=1}^{k} \widehat{s}_j. \tag{4.19}$$

Using this expression with the relation $s_k = x_c - x_k + BH^T\lambda_k$ (see line 21 of Algorithm 4.2), we obtain that

$$\begin{aligned} s_k &= -BH^T \sum_{j=1}^{k-1} \widehat{s}_j + BH^T\lambda_k \\ &= BH^T \left( \lambda_k - \sum_{j=1}^{k-1} \widehat{s}_j \right). \end{aligned} \tag{4.20}$$

From (4.18) and (4.20) we can find the relation between the vectors $\widehat{s}_k$ and $\lambda_k$ as

$$\boxed{\widehat{s}_k = \lambda_k - \sum_{j=1}^{k-1} \widehat{s}_j}. \tag{4.21}$$

Using the change of variable $\lambda = \widehat{s} + \sum_{j=1}^{k-1} \widehat{s}_j$, and the equation (4.19) in the linear system (4.14), we have

$$(R^{-1}HBH^T + I_m)(\widehat{s} + \sum_{j=1}^{k-1} \widehat{s}_j) = R^{-1}(HBH^T \sum_{j=1}^{k-1} \widehat{s}_j - d_k),$$

which can be written as

$$\boxed{(R^{-1}HBH^T + I_m)\,\widehat{s} = -R^{-1}d_k - \sum_{j=1}^{k-1}\widehat{s}_j}. \tag{4.22}$$

Then taking $\widehat{s}_0 = 0$, the initial residual of the system (4.22) can be written as

$$\widehat{r}_0 = -R^{-1}d_k - \sum_{j=1}^{k-1}\widehat{s}_j. \tag{4.23}$$

Multiplying this expression from the left with $H^T$ and using the relations (4.18) and (4.19) gives that

$$\begin{aligned} H^T\widehat{r}_0 &= -H^TR^{-1}d_k - H^T\sum_{j=1}^{k-1}\widehat{s}_j \\ &= -H^TR^{-1}d_k - B^{-1}(x_k - x_c) \\ &= B^{-1}(x_c - x_k) - H^TR^{-1}d_k, \end{aligned}$$

which is the inital residual of the linear system (3.1) with $s_0 = 0$. This ensures that Assumption 4.1 holds, i.e.

$$\boxed{H^T\widehat{r}_0 = r_0}. \tag{4.24}$$

Therefore, under the assumption that $PH^T = BH^TD$, applying RPCG on the linear system (4.22) with the $HBH^T$-inner product and recovering the solution in $\mathbb{R}^n$ from the relation (4.18) yields mathematically equivalent iterates $\{s_k\}$ to those of applying PCG Algorithm 2.5 to the linear system (3.1) (Approach (B)) with zero initial guess. We name this variant of RPCG as *Approach (DualB)*. This approach is given by Algorithm 4.3. *Like Approach (B) this dual approach does not require a factorization of the preconditioner.*

**Algorithm 4.3: RPCG Algorithm (version for $s_0 = 0$)**

$\widehat{s}_0 = 0$
$\widehat{r}_0 = -R^{-1}d_k - \sum_{j=1}^{k}\widehat{s}_j$
$\widehat{z}_0 = D\widehat{r}_0$
$\widehat{p}_0 = \widehat{z}_0$
$w_0 = HBH^T\widehat{z}_0$
$\rho_0 = \widehat{r}_0^T w_0$
$t_0 = w_0$
**for** $i = 0, 1, ..., l-1$ **do**
    Same pseudo-code as with Algorithm 4.2 for the variable $\widehat{s}_i$
**end**
The solution is recovered from $s_l = BH^T\widehat{s}_l$

Note that the iterates of Algorithm 4.3 and the iterates of Algorithm 4.2 are related to each other with equation (4.21).

We now want to find a LMP formula for the linear system matrix

$$\boxed{\widehat{A} = I_m + R^{-1}HBH^T}, \tag{4.25}$$

satisfying Assumption 4.2 for a given general LMP (2.38). It is shown in Gratton, Gürol and Toint (2013) that it is possible to find a quasi-Newton LMP $D \in \mathbb{R}^m$ satisfying $PH^T = BH^T D$ (where $P$ is a preconditioner for the primal approach) from the pairs that are available as by-products of RPCG. The following lemma generalizes this quasi-Newton LMP for the system matrix $\widehat{A}$.

**Lemma 4.1** *Let $HBH^T\widehat{A}$ and $HBH^T\widehat{M}$ be symmetric and positive definite matrices of order $m$. Assume also that $\widehat{S}$ is any $m$ by $l$ matrix of rank $l$, with $l \leq m$. Then the $m$-by-$m$ matrix given by*

$$\begin{aligned} D =& [I_m - \widehat{S}(\widehat{S}^T HBH^T \widehat{A}\,\widehat{S})^{-1}\widehat{S}^T HBH^T\widehat{A}]\;\widehat{M}\;[I_m - \widehat{A}\,\widehat{S}(\widehat{S}^T HBH^T\widehat{A}\,\widehat{S})^{-1}\widehat{S}^T HBH^T] \\ &+ \widehat{S}(\widehat{S}^T HBH^T\widehat{A}\,\widehat{S})^{-1}\widehat{S}^T HBH^T \end{aligned} \tag{4.26}$$

*is symmetric with respect to the $HBH^T$-inner product.*

*Suppose also that $MH^T = BH^T\widehat{M}$. If we denote $S = BH^T\widehat{S}$ and $ABH^T = H^T\widehat{A}$ then the LMP matrix $P$ given by (2.38) and the matrix $D$ defined by (4.26) satisfies $PH^T = BH^T D$.*

**Proof.** Let us define $Q = HBH^T$. Multiplying $D$ on the left by $Q$ gives that

$$\begin{aligned} QD =& [Q - Q\widehat{S}(\widehat{S}^T Q\widehat{A}\,\widehat{S})^{-1}\widehat{S}^T Q\widehat{A}]\;\widehat{M}\;[I_m - \widehat{A}\,\widehat{S}(\widehat{S}^T Q\widehat{A}\,\widehat{S})^{-1}\widehat{S}^T Q] \\ &+ Q\widehat{S}(\widehat{S}^T Q\widehat{A}\,\widehat{S})^{-1}\widehat{S}^T Q \end{aligned} \tag{4.27}$$

Using the assumption that $Q\widehat{A}$ and $Q\widehat{M}$ are symmetric, we have

$$\begin{aligned} QD &= [I_m - Q\widehat{S}(\widehat{S}^T Q\widehat{A}\,\widehat{S})^{-1}\widehat{S}^T\widehat{A}^T]\;\widehat{M}^T\;[Q - \widehat{A}^T Q\,\widehat{S}(\widehat{S}^T Q\widehat{A}\,\widehat{S})^{-1}\widehat{S}^T Q] \\ &\quad + Q\widehat{S}(\widehat{S}^T Q\widehat{A}\,\widehat{S})^{-1}\widehat{S}^T Q \\ &= D^T Q, \end{aligned}$$

which proves the symmetry of $D$ in $HBH^T$.

For the second part of the proof, multiplying (2.38) from right by $H^T$ gives that

$$PH^T = [I_n - S(S^T AS)^{-1}S^T A]M[H^T - AS(S^T AS)^{-1}S^T H^T] + S(S^T AS)^{-1}S^T H^T.$$

Substituting the relations $MH^T = BH^T\widehat{M}$, $S = BH^T\widehat{S}$ and $ABH^T = H^T\widehat{A}$ into this

equation given in the lemma yields that

$$\begin{aligned} PH^T &= [I_n - BH^T\widehat{S}(\widehat{S}^THBH^T\widehat{A}\widehat{S})^{-1}\widehat{S}^THBA]MH^T[I_m - \widehat{A}\widehat{S}(\widehat{S}^THBH^T\widehat{A}\widehat{S})^{-1}\widehat{S}^THBH^T] \\ &\quad +BH^T\widehat{S}(\widehat{S}^THBH^T\widehat{A}\widehat{S})^{-1}\widehat{S}^THBH^T \\ &= BH^T[I_m - \widehat{S}(\widehat{S}^THBH^T\widehat{A}\widehat{S})^{-1}\widehat{S}^THBH^T\widehat{A}]\widehat{M}[I_m - \widehat{A}\widehat{S}(\widehat{S}^THBH^T\widehat{A}\widehat{S})^{-1}\widehat{S}^THBH^T] \\ &\quad +BH^T\widehat{S}(\widehat{S}^THBH^T\widehat{A}\widehat{S})^{-1}\widehat{S}^THBH^T \\ &= BH^T\Big\{[I_m - \widehat{S}(\widehat{S}^THBH^T\widehat{A}\widehat{S})^{-1}\widehat{S}^THBH^T\widehat{A}]\widehat{M}[I_m - \widehat{A}\widehat{S}(\widehat{S}^THBH^T\widehat{A}\widehat{S})^{-1}\widehat{S}^THBH^T] \\ &\quad +\widehat{S}(\widehat{S}^THBH^T\widehat{A}\widehat{S})^{-1}\widehat{S}^THBH^T\Big\} \\ &= BH^TD \end{aligned}$$

which completes the proof. □

In this lemma since $P$ is a symmetric positive definite preconditioner and $HPH^T = HBH^TD$, we have that when $H$ has full row rank, the matrix $HBH^TD$ is not only symmetric but also positive definite. Note that the LMP $D$ given in Lemma 4.1 is a preconditioner for $\widehat{A} = I_n + R^{-1}HBH^T$ and $HBH^TD$ is an approximation of the inverse matrix $HBH^T\widehat{A} = HBH^T + HBH^TR^{-1}HBH^T$.

The formula (4.26) given in Lemma 4.1 can be rewritten inductively if $\widehat{S} \in \mathbb{R}^{m\times l}$ is $\widehat{A}$-conjugate with respect to the $HBH^T$-inner product. The next lemma provides such form of the preconditioner.

**Lemma 4.2** *Let $HBH^T\widehat{A}$ and $HBH^T\widehat{M}$ be $m \times m$ symmetric positive definite matrices. Assume also that $\widehat{S} \in \mathbb{R}^{m\times l}$ is $\widehat{A}$-conjugate with respect to the $HBH^T$-inner product. Then the matrix $D$ defined by (4.26) can be written as*

$$\begin{aligned} D_l = &\left(I_m - \sum_{i=1}^{l}\frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^THBH^T\widehat{A}\,\widehat{s}_i}HBH^T\widehat{A}\right)\widehat{M}\left(I_m - \sum_{i=1}^{l}\widehat{A}\frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^THBH^T\widehat{A}\,\widehat{s}_i}HBH^T\right) \\ &+\sum_{i=1}^{l}\frac{\widehat{s}_i\widehat{s}_i^T}{\widehat{s}_i^THBH^T\widehat{A}\,\widehat{s}_i}HBH^T, \end{aligned} \tag{4.28}$$

*which can also be written inductively as*

$$D_l = \left(I_m - \frac{\widehat{s}_l\widehat{s}_l^THBH^T\widehat{A}}{\widehat{s}_l^THBH^T\widehat{A}\,\widehat{s}_l}\right)D_{l-1}\left(I_m - \frac{\widehat{A}\,\widehat{s}_l\widehat{s}_l^THBH^T}{\widehat{s}_l^THBH^T\widehat{A}\,\widehat{s}_l}\right) + \frac{\widehat{s}_l\widehat{s}_l^THBH^T}{\widehat{s}_l^THBH^T\widehat{A}\,\widehat{s}_l}. \tag{4.29}$$

**Proof.** From $\widehat{S} = [\widehat{s}_1, \cdots, \widehat{s}_l]$ and the $\widehat{A}$-conjugacy of the matrix $\widehat{S}$ in the $HBH^T$ inner-product, i.e. $\widehat{s}_i^THBH^T\widehat{A}\,\widehat{s}_j = 0$ for $i \neq j$, $\widehat{S}^THBH^T\widehat{A}\,\widehat{S}$ becomes a diagonal

matrix. Then, $\widehat{S}(\widehat{S}^T HBH^T \widehat{A}\,\widehat{S})^{-1}\widehat{S}^T$ can be written as

$$\begin{aligned}
\widehat{S}(\widehat{S}^T HBH^T \widehat{A}\,\widehat{S})^{-1}\widehat{S}^T &= [\widehat{s}_1, \cdots, \widehat{s}_l] \begin{pmatrix} \widehat{s}_1^T HBH^T \widehat{A}\,\widehat{s}_1 & & \\ & \ddots & \\ & & \widehat{s}_l^T HBH^T \widehat{A}\,\widehat{s}_l \end{pmatrix}^{-1} \begin{pmatrix} \widehat{s}_1^T \\ \vdots \\ \widehat{s}_l^T \end{pmatrix} \\
&= [\widehat{s}_1, \cdots, \widehat{s}_l] \begin{pmatrix} \frac{1}{\widehat{s}_1^T HBH^T \widehat{A}\,\widehat{s}_1} & & \\ & \ddots & \\ & & \frac{1}{\widehat{s}_l^T HBH^T \widehat{A}\,\widehat{s}_l} \end{pmatrix} \begin{pmatrix} \widehat{s}_1^T \\ \vdots \\ \widehat{s}_l^T \end{pmatrix} \\
&= [\widehat{s}_1, \cdots, \widehat{s}_l] \begin{pmatrix} \frac{\widehat{s}_1^T}{\widehat{s}_1^T HBH^T \widehat{A}\,\widehat{s}_1} & & \\ & \ddots & \\ & & \frac{\widehat{s}_l^T}{\widehat{s}_l^T HBH^T \widehat{A}\,\widehat{s}_l} \end{pmatrix} \\
&= \sum_{i=1}^{l} \frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T HBH^T \widehat{A}\,\widehat{s}_i}.
\end{aligned}$$

Substituting this expression in (4.26) gives the equation (4.28).

We next prove the second part of the lemma. Let us define $D_0 = \widehat{M}$ and $Q = HBH^T$. From the equation (4.28) for $\ell = 1$ we obtain that

$$D_1 = \left(I_m - \frac{\widehat{s}_1\,\widehat{s}_1^T}{\widehat{s}_1^T Q\widehat{A}\,\widehat{s}_1} Q\widehat{A}\right) D_0 \left(I_m - \widehat{A}\frac{\widehat{s}_1\,\widehat{s}_1^T}{\widehat{s}_1^T Q\widehat{A}\,\widehat{s}_1} Q\right) + \frac{\widehat{s}_1\widehat{s}_1^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_1} Q.$$

We now assume that $l > 1$ and define

$$V_l = I_m - \frac{Q\widehat{A}\widehat{s}_l\,\widehat{s}_l^T}{\widehat{s}_l^T Q\widehat{A}\,\widehat{s}_l}, \tag{4.30}$$

$$W_l = I_m - \frac{\widehat{A}\widehat{s}_l\,\widehat{s}_l^T Q}{\widehat{s}_l^T Q\widehat{A}\,\widehat{s}_l}. \tag{4.31}$$

Using these definitions, we have

$$V_l^T \sum_{i=1}^{l-1} \frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i} = \sum_{i=1}^{l-1} \frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i} - \frac{\widehat{s}_l\,\widehat{s}_l^T \widehat{A}^T Q}{\widehat{s}_l^T Q\widehat{A}\,\widehat{s}_l} \sum_{i=1}^{l-1} \frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}, \tag{4.32}$$

$$\left(\sum_{i=1}^{l-1} \frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}\right) QW_l = \sum_{i=1}^{l-1} \frac{\widehat{s}_i\,\widehat{s}_i^T Q}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i} - \left(\sum_{i=1}^{l-1} \frac{\widehat{s}_i\,\widehat{s}_i^T Q}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}\right) \frac{\widehat{A}\widehat{s}_l\,\widehat{s}_l^T Q}{\widehat{s}_l^T Q\widehat{A}\,\widehat{s}_l}. \tag{4.33}$$

Using the symmetry of $\widehat{A}$ with respect to the $HBH^T$-inner product and the assumption that the vectors $\widehat{s}_i$, $i = 1, \ldots, \ell$ are $\widehat{A}$-conjugate with respect to the $HBH^T$-inner product, the second part of equations (4.32) and (4.33) vanishes yielding

$$V_l^T \sum_{i=1}^{l-1} \frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i} = \sum_{i=1}^{l-1} \frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i} \tag{4.34}$$

$$\left(\sum_{i=1}^{l-1}\frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}\right)QW_l=\sum_{i=1}^{l-1}\frac{\widehat{s}_i\,\widehat{s}_i^T Q}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i} \tag{4.35}$$

Using the definitions (4.30) and (4.31) we also have

$$\begin{aligned}V_l^T\left(I_m-\sum_{i=1}^{l-1}\frac{\widehat{s}_i\,\widehat{s}_i^T Q\widehat{A}}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}\right) &= I_m-\sum_{i=1}^{l-1}\frac{\widehat{s}_i\,\widehat{s}_i^T Q\widehat{A}}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}-\frac{\widehat{s}_l\,\widehat{s}_l^T\widehat{A}^T Q}{\widehat{s}_l^T Q\widehat{A}\,\widehat{s}_l}\\ &\quad+\frac{\widehat{s}_l\,\widehat{s}_l^T\widehat{A}^T Q}{\widehat{s}_l^T Q\widehat{A}\,\widehat{s}_l}\sum_{i=1}^{l-1}\frac{\widehat{s}_i\,\widehat{s}_i^T Q\widehat{A}}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}\\ &= I_m-\sum_{i=1}^{l-1}\frac{\widehat{s}_i\,\widehat{s}_i^T Q\widehat{A}}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}-\frac{\widehat{s}_l\,\widehat{s}_l^T\widehat{A}^T Q}{\widehat{s}_l^T Q\widehat{A}\,\widehat{s}_l}\\ &= I_m-\sum_{i=1}^{l}\frac{\widehat{s}_i\,\widehat{s}_i^T Q\widehat{A}}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}.\end{aligned} \tag{4.36}$$

$$\begin{aligned}\left(I_m-\sum_{i=1}^{l-1}\widehat{A}\frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}Q\right)W_l &= I_m-\frac{\widehat{A}\widehat{s}_l\,\widehat{s}_l^T Q}{\widehat{s}_l^T Q\widehat{A}\,\widehat{s}_l}-\sum_{i=1}^{l-1}\widehat{A}\frac{\widehat{s}_i\,\widehat{s}_i^T Q}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}\\ &\quad+\left(\sum_{i=1}^{l-1}\widehat{A}\frac{\widehat{s}_i\,\widehat{s}_i^T Q}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}\right)\frac{\widehat{A}\widehat{s}_l\,\widehat{s}_l^T Q}{\widehat{s}_l^T Q\widehat{A}\,\widehat{s}_l}\\ &= I_m-\frac{\widehat{A}\widehat{s}_l\,\widehat{s}_l^T Q}{\widehat{s}_l^T Q\widehat{A}\,\widehat{s}_l}-\sum_{i=1}^{l-1}\widehat{A}\frac{\widehat{s}_i\,\widehat{s}_i^T Q}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}\\ &= I_m-\sum_{i=1}^{l}\widehat{A}\frac{\widehat{s}_i\,\widehat{s}_i^T Q}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}.\end{aligned} \tag{4.37}$$

Let us now write the formula (4.28) for the matrix $D_{l-1}$, $l>1$ which is given by

$$D_{l-1}=\left(I_m-\sum_{i=1}^{l-1}\frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}Q\widehat{A}\right)\widehat{M}\left(I_m-\sum_{i=1}^{l-1}\widehat{A}\frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}Q\right)+\sum_{i=1}^{l-1}\frac{\widehat{s}_i\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}Q.$$

Using this formula we have

$$\begin{aligned}V_l^T D_{l-1}W_l &= V_l^T\left(I_m-\sum_{i=1}^{l-1}\frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}Q\widehat{A}\right)\widehat{M}\left(I_m-\sum_{i=1}^{l-1}\widehat{A}\frac{\widehat{s}_i\,\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}Q\right)W_l\\ &\quad+V_l^T\sum_{i=0}^{l-1}\frac{\widehat{s}_i\widehat{s}_i^T}{\widehat{s}_i^T Q\widehat{A}\,\widehat{s}_i}QW_l.\end{aligned} \tag{4.38}$$

Substituting the equations (4.36), (4.37), (4.34), and (4.34) into (4.38) we obtain that

$$\begin{aligned} V_l^T D_{l-1} W_l + \frac{\widehat{s}_l\, \widehat{s}_l^T Q}{\widehat{s}_l^T Q \widehat{A}\, \widehat{s}_l} &= \left( I_m - \sum_{i=1}^{l} \frac{\widehat{s}_i\, \widehat{s}_i^T Q \widehat{A}}{\widehat{s}_i^T Q \widehat{A}\, \widehat{s}_i} \right) \widehat{M} \left( I_m - \sum_{i=1}^{l} \widehat{A} \frac{\widehat{s}_i\, \widehat{s}_i^T Q}{\widehat{s}_i^T Q \widehat{A}\, \widehat{s}_i} \right) \\ &\quad + \sum_{i=1}^{l-1} \frac{\widehat{s}_i\, \widehat{s}_i^T Q}{\widehat{s}_i^T Q \widehat{A}\, \widehat{s}_i} + \frac{\widehat{s}_l\, \widehat{s}_l^T Q}{\widehat{s}_l^T Q \widehat{A}\, \widehat{s}_l} \\ &= D_l, \end{aligned} \tag{4.39}$$

which gives the desired result. □

We are left with the task of integrating Approach (DualB) when solving a convergent sequence of linear systems (4.22) of the form of $\widehat{A}\widehat{s} = \widehat{b}_k$, with multiple right-hand sides where preconditioning is achieved by the warm-start LMP technique.

Assume that we choose the initial guess $x_1 = x_c$. Assume also that the initial preconditioner $D_0$ is chosen such that

$$\boxed{P_0 H^T = B H^T D_0,} \tag{4.40}$$

for instance $P_0 = B$ and $D_0 = I$. Then for $k = 1$, RPCG Algorithm 4.3 solves the linear system

$$\widehat{A}\, \widehat{s} = \widehat{b}_1 \tag{4.41}$$

for the variable $\widehat{s}$ by using the preconditioner $D_0$ and recover the approximate solution from the relation $s_1 = B H^T\, \widehat{s}_1$. This approximate solution is then used to update the current iterate from $x_2 = x_1 + s_1$.

While performing the RPCG algorithm on the linear system (4.41), the information can be stored to obtain the LMP $D_1$ for the system matrix $\widehat{A}$ by using the formula (4.26). This preconditioner can then be used to precondition the second linear system. Note that while constructing $D_1$ from (4.26), $\widehat{M}_0$ is taken as the initial preconditioner matrix, i.e. $\widehat{M}_0 = D_0$. From Lemma 4.1 we ensure that $P_1 H^T = B H^T D_1$ where $P_1$ is the preconditioner for the second linear system in Algorithm 3.2 assuming that $S_0 = B H^T \widehat{S}_0$.

Using the preconditioner $D_1$ for $k = 2$, RPCG solves the linear system

$$\widehat{A}\, \widehat{s} = \widehat{b}_2 \tag{4.42}$$

for the variable $\widehat{s}$ and recover the approximate solution from $s_2 = B H^T\, \widehat{s}_2$. This approximate solution is then used to update the current iterate from $x_3 = x_2 + s_2$.

As it is done for $k = 1$, while performing the RPCG algorithm on the linear system (4.42) the information can be stored to obtain the LMP $D_2$ for the system matrix $\widehat{A}$ by using the formula (4.26). Note that while constructing $D_2$ from (4.26), $\widehat{M}_1$ is taken as the preconditioner used for $k = 2$, $\widehat{M}_1 = D_1$. From Lemma 4.1 we ensure that $P_2 H^T = B H^T D_2$ where $P_2$ is the preconditioner for the third linear system in Algorithm 3.2 assuming that $S_1 = B H^T \widehat{S}_1$.

The same strategy can be carried over for later steps. Using the preconditioner $D_k$,

for the $(k+1)$-th linear system RPCG Algorithm solves the preconditioned system

$$\widehat{A}\,\widehat{s} = \widehat{b}_{k+1} \tag{4.43}$$

for the variable $\widehat{s}$ and recover the approximate solution from $s_{k+1} = BH^T\widehat{s}_{k+1}$. This approximate solution is then used to update the current iterate from $x_{k+2} = x_{k+1} + s_{k+1}$. Algorithm 4.4 outlines this implementation. Like Algorithm 3.2 this algorithm *does not require the factorization of the preconditioner.*

**Algorithm 4.4: RPCG for solving a sequence of linear systems with multiple right-hand sides**

1 **Initialization:** Choose an initial preconditioner $D_0$ such that Assumption 4.2 holds. Set $x_1 = x_c$.

2 **Perform inner loop**: Using preconditioner $D_{k-1}$ solve the linear system

$$(R^{-1}HBH^T + I_m)\,\widehat{s} = -R^{-1}d_k - \sum_{j=1}^{k}\widehat{s}_j,$$

with RPCG Algorithm 4.3. During RPCG extract and save relevant information to precondition the next linear system with $D_k$ formulated by (4.26). The solution in $\mathbb{R}^n$ is recovered from

$$s_{k+1} = BH^T\widehat{s}_{k+1}$$

3 **Update the iterate:**

$$x_{k+1} = x_k + s_{k+1}$$

4 Increment $k$ by 1 and go to step 2.

When solving the $k$-th linear system in Algorithm 4.4, RPCG Algorithm 4.3 requires a single matrix-vector product with each of the operators $H, H^T, R^{-1}, B$ and $D_{k-1}$.

*Algorithm 4.4 generates a mathematically equivalent iterates of $\{s_k\}$ to those of Algorithm 3.2 under the assumptions that*

- $x_1 = x_c$,
- $s_0 = 0$ *for Approach (B),*
- $P_0H^T = BH^TD_0$,

*while possibly yielding gains in terms of both memory usage and computational cost when $m \ll n$.*

We summarize the characteristics of the equivalent primal and dual algorithms by Table 4.1.

We next explain briefly how to perform *split preconditioning with RPCG.*

| Algorithm | Inner min. | LMP formula | Assumption |
|---|---|---|---|
| 3.2 | App. (B) | $P_{k+1}$ given by (2.38)<br>with $M_k = P_k$ | $x_1 = x_c$<br>$s_0 = 0$ |
| 4.4 | App. (DualB) | $D_{k+1}$ given by (4.26)<br>with $\widehat{M}_k = D_k$ and $S_k = BH^T\widehat{S}_k$ | $P_0H^T = BH^TD_0$<br>$x_1 = x_c$ |

Table 4.1: A summary of the characteristics of Algorithms 3.2 and 4.4.

### Approach (A) in the dual space

The linear system (4.22) can be equivalently solved by applying CG Algorithm 2.2 to the linear system

$$\boxed{(R^{-1/2}HBH^TR^{-1/2} + I_m)\,\widehat{v} = -R^{-1/2}d_k - R^{1/2}\sum_{j=1}^{k-1}\widehat{s}_j}, \tag{4.44}$$

with the $(R^{-1/2}HBH^TR^{-1/2})$-inner product where $\widehat{s} = R^{-1/2}\widehat{v}$. We can see this equivalence by observing that the linear system (4.44) is obtained after applying $R^{1/2}$-left preconditioning and $R^{-1/2}$-right preconditioning to the linear system (4.22). In order to preserve mathematically equivalent iterates to RPCG Algorithm 4.3 (with $D = I$), the inner product should be changed accordingly (Chan, Chow, Saad and Yeung, 1999) which becomes in this case the $(R^{-1/2}HBH^TR^{-1/2})$-inner product.

We can rewrite the linear system (4.44) in the form of

$$\boxed{R^{1/2}\widehat{A}R^{-1/2}\,\widehat{v} = R^{1/2}\widehat{b}_k}, \tag{4.45}$$

where $\widehat{A} = R^{-1}HBH^T + I_m$ and $\widehat{b}_k = R^{-1}d_k - \sum_{j=1}^{k-1}\widehat{s}_j$.

How can we construct and apply a LMP for the linear system (4.45) provided that we obtain mathematically equivalent iterates to those of Approach (DualB)? The answer is explained as follows.

Assume that $\widehat{Y} = \widehat{F}\widehat{F}^T$ is a preconditioner for the linear system matrix in (4.45), i.e. $R^{1/2}\widehat{A}R^{-1/2}$. Using this preconditioner, CG Algorithm can be applied to the linear system

$$\widehat{F}^T(R^{1/2}\widehat{A}R^{-1/2})\widehat{F}\,\widehat{v} = \widehat{F}^TR^{1/2}\widehat{b}_k, \tag{4.46}$$

for the variable $\widehat{v}$ using the $(\widehat{F}^{-1}R^{-1/2}HBH^TR^{-1/2}\widehat{F})$-inner product. The solution in dual space is recovered from $\widehat{s} = R^{-1/2}\widehat{F}\,\widehat{v}$. We name this algorithm *Approach (DualA)*.

Let us assume that

$$\boxed{D = R^{-1/2}\widehat{F}\widehat{F}^TR^{1/2}}, \tag{4.47}$$

where $D$ is the LMP used during Approach (DualB). Following the same strategy in (Gratton and Tshimanga, 2009), it can be easily shown that Approach (DualB) and

Approach (DualA) generates mathematically equivalent iterates under the assumption (4.47).

When solving a sequence of linear systems for the $(k+1)$-th linear system in sequence Approach (DualA) solves the linear system

$$\boxed{\widehat{F}_k^T(\widehat{F}_{k-1}^T \dots \widehat{F}_1^T R^{1/2} \widehat{A} R^{-1/2} \widehat{F}_1 \dots \widehat{F}_{k-1}) \widehat{F}_k \, \widehat{v} = \widehat{F}_k^T \widehat{F}_{k-1}^T \dots \widehat{F}_1^T R^{1/2} \widehat{b}_k}, \tag{4.48}$$

with the $(\widehat{F}_k^{-1} \dots \widehat{F}_1^{-1} R^{-1/2} H B H^T R^{-1/2} \widehat{F}_1 \dots \widehat{F}_k)$-inner product. The solution in dual space is recovered from $\widehat{s}_{k+1} = R^{-1/2} \widehat{F}_1 \dots \widehat{F}_k \widehat{v}_{k+1}$. In this case Assumption (4.47) can also be generalized as

$$\boxed{D_k = R^{-1/2} \widehat{F}_1 \dots \widehat{F}_k \widehat{F}_k^T \dots \widehat{F}_1^T R^{1/2}}. \tag{4.49}$$

By using this relation we can see the relation between Approach (DualA) and Approach (A) as explained below.

Substituting the relation (4.49) together with the relation (3.35) into Assumption 4.2, we obtain that

$$\begin{aligned} P_k H^T &= B H^T D_k, \\ P_{k-1}^{1/2} Y_k P_{k-1}^{1/2} H^T &= B H^T R^{-1/2} \widehat{F}_1 \dots \widehat{F}_k \widehat{F}_k^T \dots \widehat{F}_1^T R^{1/2}. \end{aligned}$$

From this equation, using the relations $P_k^{1/2} = F_0 \dots F_k$, $Y_k = F_k F_k^T$ and taking $F_0 = B^{1/2}$, we have

$$B^{1/2} F_1 \dots F_{k-1} F_k F_k^T F_{k-1}^T \dots F_1 B^{1/2} H^T = B H^T R^{-1/2} \widehat{F}_1 \dots \widehat{F}_k \widehat{F}_k^T \dots \widehat{F}_1^T R^{1/2}.$$

Multiplying this equation from left by $R^{-1/2}$ and from right by $B^{-1/2}$ yields that

$$\boxed{F_1 \dots F_{k-1} F_k F_k^T F_{k-1}^T \dots F_1 B^{1/2} H^T R^{-1/2} = B^{1/2} H^T R^{-1/2} \widehat{F}_1 \dots \widehat{F}_k \widehat{F}_k^T \dots \widehat{F}_1^T}. \tag{4.50}$$

Therefore, under this assumption on the preconditioners Approach (A) and Approach (DualA) generates a mathematically equivalent sequence of iterates $\{s_k\}$.

The formula for the LMP $\widehat{F}_k$ can be constructed by using the relation (4.47) and the LMP formula (4.26). Within this thesis preconditioning with LMPs is not explained in detail for Approach (DualA), however the necessary relations for the derivation of the LMPs are provided.

In the next section, we explain in detail preconditioning with particular LMPs: the quasi-Newton LMP, the Ritz LMP and the spectral LMP, for Approach (DualB). This approach is more efficient than Approach (DualA) in terms of computational cost for large-scale systems since it does not require the square-root factorization of the matrices $B$ and $R^{-1}$. Like Chapter 3, this part will include for each preconditioner a definition of the preconditioner, its properties, an extraction methodology of the required information from Approach (DualB), and memory and flops requirements for

the application of the preconditioner.

#### 4.3.1.1 Preconditioning with the quasi-Newton LMP

In this section we want to derive the quasi-Newton LMP to precondition the linear system (4.22). We may follow the by now familiar pattern of deriving an equivalent preconditioner to that of the quasi-Newton LMP (3.37). It is shown in Gratton, Gürol and Toint (2013) that this is indeed possible and that the resulting formula satisfies a variational property similar to that described by (3.38)-(3.39). We first focus on deriving a quasi-Newton limited memory preconditioner for dual approach satisfying Assumption 4.2.

The search directions $\widehat{p}_i$, $i = 1, \ldots, j$ generated by Algorithm 4.3 are $\widehat{A}$-conjugate with respect to the $HBH^T$-inner product, i.e.

$$\widehat{p}_i^T HBH^T \widehat{A} \widehat{p}_j = 0 \text{ for } i \neq j.$$

Therefore, choosing these directions in the formula (4.29), i.e. $\widehat{s}_i \equiv \widehat{p}_i$, $i = 1, \ldots, j$ gives that

$$D_{j+1} = \left( I_m - \frac{\widehat{p}_j \widehat{p}_j^T HBH^T \widehat{A}}{\widehat{p}_j^T HBH^T \widehat{A}\, \widehat{p}_j} \right) D_j \left( I_m - \frac{\widehat{A}\, \widehat{p}_j \widehat{p}_j^T HBH^T}{\widehat{p}_j^T HBH^T \widehat{A}\, \widehat{p}_j} \right) + \frac{\widehat{p}_j \widehat{p}_j^T HBH^T}{\widehat{p}_j^T HBH^T \widehat{A}\, \widehat{p}_j},$$

where $\widehat{A} = I_m + R^{-1} HBH^T$. Using this formula the quasi-Newton LMP for the matrix $\widehat{A}$ in the $k$-th linear system (in Algorithm 4.4) can be constructed from

$$\boxed{D_{j+1} = \left( I_m - \tau_j \widehat{p}_j \widehat{q}_j^T HBH^T \right) D_j \left( I_m - \tau_j \widehat{q}_j t_j^T \right) + \tau_j \widehat{p}_j t_j^T}, \tag{4.51}$$

where $\widehat{q}_i = \widehat{A}\widehat{p}_i$, $t_i = HBH^T \widehat{p}_i$, and $\tau_i = 1/(\widehat{q}_i^T t_i)$ are obtained during Algorithm 4.3 applied to the previous linear system. The initial preconditioning matrix for the formula (4.51) is chosen as the preconditioner that is used for the $(k-1)$-th linear system.

Using the relations (4.16), it can be easily shown that the preconditioner $P_{j+1}$ formulated by (3.37) and the preconditioner $D_{j+1}$ formulated by (4.51) satisfies the relation $P_{j+1} H^T = BH^T D_{j+1}$ (Gratton, Gürol and Toint, 2013).

We now show that the preconditioner $D$ obtained from (4.51) also satisfies variational properties in the dual space.

**Lemma 4.3** *Let $Q = HBH^T$ and suppose that $Q$ is non-singular. Then the matrix $\Delta D_j$ defined by $\Delta D_j = D_{j+1} - D_j$ where $D_{j+1}$ defined in (4.51) is the solution of*

$$\min_{\Delta D_j} \left\| W^{1/2} Q^{1/2} \Delta D_j Q^{-1/2} W^{1/2} \right\|_F \tag{4.52}$$

$$\text{subject to } \; Q \Delta D_j = \Delta D_j^T Q, \; D_{j+1} \widehat{q}_j = \widehat{p}_j,$$

*where $W$ is any symmetric positive definite matrix satisfying $W Q^{1/2} \widehat{p}_j = Q^{1/2} \widehat{q}_j$.*

**Proof.** Using the change of variables

$$\Delta X_j = Q^{1/2}\Delta D_j Q^{-1/2},\ \widehat{p}_j = Q^{-1/2}s_j \text{ and } \widehat{q}_j = Q^{-1/2}y_j \tag{4.53}$$

we can rewrite problem (4.52) as

$$\min_{\Delta X_j} \left\| W^{1/2}\Delta X_j W^{1/2} \right\|_F$$

$$\text{subject to } \Delta X_j = \Delta X_j^T, \quad X_{j+1}y_j = s_j,$$

which is structurally identical to problem (3.38). Using now (3.40) in this context yields that

$$\Delta X_j = \frac{W^{-1}y_j(s_j - X_j y_j)^T + (s_j - X_j y_j)y_j^T W^{-1}}{y_j^T W^{-1} y_j} - \frac{y_j^T(s_j - X_j y_j)W^{-1}y_j y_j^T W^{-1}}{(y_j^T W^{-1} y_j)^2}. \tag{4.54}$$

Substituting (4.53) into this solution and multiplying by $Q^{1/2}$ on the right and $Q^{-1/2}$ on the left gives that

$$\begin{aligned}\Delta D_j = &\frac{Q^{-1/2}W^{-1}Q^{1/2}\widehat{q}_j(Q^{1/2}\widehat{p}_j - Q^{1/2}D_j\widehat{q}_j)^T Q^{1/2} + (\widehat{p}_j - D_j\widehat{q}_j)\widehat{q}_j^T Q^{1/2}W^{-1}Q^{1/2}}{\widehat{q}_j^T Q^{1/2}W^{-1}Q^{1/2}\widehat{q}_j} \\ &- \frac{Q^{-1/2}\widehat{q}_j^T Q^{1/2}(Q^{1/2}\widehat{p}_j - Q^{1/2}D_j\widehat{q}_j)W^{-1}Q^{1/2}\widehat{q}_j\widehat{q}_j^T Q^{1/2}W^{-1}Q^{1/2}}{(\widehat{q}_j^T Q^{1/2}W^{-1}Q^{1/2}\widehat{q}_j)^2}.\end{aligned}$$

From the relation $WQ^{1/2}\widehat{p}_j = Q^{1/2}\widehat{q}_j$, we deduce that $Q^{1/2}\widehat{p}_j = W^{-1}Q^{1/2}\widehat{q}_j$. Substituting this expression in the solution gives

$$\Delta D_j = \frac{\widehat{p}_j(\widehat{p}_j - D_j\widehat{q}_j)^T Q + (\widehat{p}_j - D_j\widehat{q}_j)\widehat{p}_j^T Q}{\widehat{q}_j^T Q\widehat{p}_j} - \frac{\widehat{q}_j^T Q(\widehat{p}_j - D_j\widehat{q}_j)p_j\widehat{p}_j^T Q}{(\widehat{q}_j^T Q\widehat{p}_j)^2}. \tag{4.55}$$

On the other hand, we can reformulate (4.51) as:

$$\begin{aligned}D_{j+1} &= D_j - \frac{D_j\widehat{q}_j\widehat{p}_j^T Q}{\widehat{q}_j^T Q\widehat{p}_j} - \frac{\widehat{p}_j\widehat{q}_j^T QD_j}{\widehat{q}_j^T Q\widehat{p}_j} + \frac{\widehat{p}_j\widehat{q}_j^T QD_j\widehat{q}_j\widehat{p}_j^T Q}{(\widehat{q}_j^T Q\widehat{p}_j)^2} + \frac{\widehat{p}_j\widehat{p}_j^T Q}{\widehat{q}_j^T Q\widehat{p}_j} \\ &= D_j + \frac{(\widehat{p}_j - D_j\widehat{q}_j)\widehat{p}_j^T Q}{\widehat{q}_j^T Q\widehat{p}_j} - \frac{\widehat{p}_j\widehat{q}_j^T QD_j}{\widehat{q}_j^T Q\widehat{p}_j} + \frac{\widehat{p}_j\widehat{q}_j^T QD_j\widehat{q}_j\widehat{p}_j^T Q}{(\widehat{q}_j^T Q\widehat{p}_j)^2}.\end{aligned}$$

Adding and substracting the term $\frac{\widehat{p}_j\widehat{p}_j^T Q}{\widehat{q}_j^T Q\widehat{p}_j}$ gives that,

$$D_{j+1} = D_j + \frac{(\widehat{p}_j - D_j\widehat{q}_j)\widehat{p}_j^T Q}{\widehat{q}_j^T Q\widehat{p}_j} + \frac{\widehat{p}_j\widehat{p}_j^T Q}{\widehat{q}_j^T Q\widehat{p}_j} - \frac{\widehat{p}_j\widehat{p}_j^T Q}{\widehat{q}_j^T Q\widehat{p}_j} - \frac{\widehat{p}_j\widehat{q}_j^T QD_j}{\widehat{q}_j^T Q\widehat{p}_j} + \frac{\widehat{p}_j\widehat{q}_j^T QD_j\widehat{q}_j\widehat{p}_j^T Q}{(\widehat{q}_j^T Q\widehat{p}_j)^2},$$

which can be reorganized as

$$D_{j+1} = D_j + \frac{(\widehat{p}_j - D_j\widehat{q}_j)\widehat{p}_j^T Q + \widehat{p}_j(Q\widehat{p}_j - D_j^T Q\widehat{q}_j)^T}{\widehat{q}_j^T Q\widehat{p}_j} - \frac{\widehat{q}_j^T Q(\widehat{p}_j - D_j\widehat{q}_j)\widehat{p}_j\widehat{p}_j^T Q}{(\widehat{q}_j^T Q\widehat{p}_j)^2}.$$

Using the property that $QD_j = D_j^T Q$ (see Lemma 4.1), we write that

$$D_{j+1} = D_j + \frac{(\widehat{p}_j - D_j\widehat{q}_j)\widehat{p}_j^T Q + \widehat{p}_j(\widehat{p}_j - D_j\widehat{q}_j)^T Q}{\widehat{q}_j^T Q\widehat{p}_j} - \frac{\widehat{q}_j^T Q(\widehat{p}_j - D_j\widehat{q}_j)\widehat{p}_j\widehat{p}_j^T Q}{(\widehat{q}_j^T Q\widehat{p}_j)^2},$$

which is equivalent to $D_j + \Delta D_j$ where $\Delta D_j$ is given by the formula (4.55). □

Having found a suitable preconditioner (in the sense that it satisfies Assumption 4.2) and having verified that it shares desirable variational properties with its primal equivalent, we are left with the task of integrating it into the RPCG algorithm. From formula (4.51), we need to store the sequences of $\widehat{q}$, $\widehat{p}$, $t$, and $HBH^T\widehat{q}$ to obtain the successive preconditioner updates. Storing $\widehat{q}$, $\widehat{p}$ and $t$ does not require additional cost since they are already available from a run of RPCG Algorithm 4.3. On the other hand, the quantity $HBH^T\widehat{q}$ is not a by-product of the algorithm, and seems, at first sight, to require an additional matrix vector product, which may appear computationally costly. Fortunately, we can rewrite RPCG Algorithm 4.3 in a more computationally effective way by introducing a vector $l$ defined by

$$l_i = HBH^T\widehat{r}_i.$$

Since $\widehat{z}_i = D\widehat{r}_i$ and $HBH^T D$ is symmetric from Lemma 4.1, we may therefore write that

$$w_i = HBH^T D\widehat{r}_i = D^T HBH^T\widehat{r}_i = D^T l_i. \tag{4.56}$$

Moreover, multiplying the line that updates the residual of Algorithm 4.3 by $HBH^T$ gives that

$$HBH^T\widehat{q}_i = (l_i - l_{i+1})/\alpha_i$$

which is the matrix vector product $HBH^T\widehat{q}$ that we need to store. Using all these relations, we can transform RPCG Algorithm 4.3 into Algorithm 4.5.

*As a result it is possible to find the corresponding quasi-Newton LMP (LMP (4.51)) for Algorithm 4.5 such that when it is used within Algorithm 4.4, it generates a mathematically equivalent sequence of iterates $\{s_k\}$ to those of Algorithm 3.2 with the quasi-Newton LMP (3.37), while possibly yielding gains in terms of both memory usage and computational cost when $m \ll n$. We showed that this preconditioner can be constructed from vectors which are by-products of Algorithm 4.5.*

We summarize the characteristics of the equivalent primal and dual algorithms when used with the quasi-Newton LMP by Table 4.2. In this table we define another approach for RPCG Algorithm when used with the quasi-Newton LMP :

**Algorithm 4.5: RPCG Algorithm (version for quasi-Newton LMP)**

$\widehat{s}_0 = 0$
$\widehat{r}_0 = -R^{-1}d_k - \sum_{j=1}^{k} \widehat{s}_j$
$l_0 = HBH^T\widehat{r}_0$
$\widehat{z}_0 = D\widehat{r}_0$
$\widehat{p}_0 = \widehat{z}_0$
$w_0 = D^T l_0$
$\rho_0 = \widehat{r}_0^T w_0$
$t_0 = w_0$
**for** $i = 0, 1, ..., l-1$ **do**
    $\widehat{q}_i = R^{-1}t_i + \widehat{p}_i$
    $\alpha_i = \rho_i / \widehat{q}_i^T t_i$
    $\widehat{s}_{i+1} = \widehat{s}_i + \alpha_i \widehat{p}_i$
    $\widehat{r}_{i+1} = \widehat{r}_i - \alpha_i \widehat{q}_i$
    $l_{i+1} = HBH^T\widehat{r}_{i+1}$
    $\varrho_i = (l_i - l_{i+1})/\alpha_i$
    $\widehat{z}_{i+1} = D\widehat{r}_{i+1}$
    $w_{i+1} = D^T l_{i+1}$
    Check convergence and stop if desired accuracy is reached
    $\rho_{i+1} = \widehat{r}_{i+1}^T w_{i+1}$
    $\beta_i = \rho_{i+1}/\rho_i$
    $\widehat{p}_{i+1} = \widehat{z}_{i+1} + \beta_i \widehat{p}_i$
    $t_{i+1} = w_{i+1} + \beta_i t_i$
**end**
The solution is recovered from $s_l = BH^T\widehat{s}_l$

Approach (DualD): Apply RPCG Algorithm 4.5 on the linear system (4.22) with a preconditioner constructed from (4.51).

| Algorithm | inner min. | quasi-Newton LMP formula | Assumption |
|---|---|---|---|
| 3.2 | Approach (B) | $P_{j+1}$ given by (3.37) | $x_1 = x_c$, $s_0 = 0$ |
| 4.4 | Approach (DualD) | $D_{j+1}$ given by (4.51) | $P_0 H^T = BH^T D_0$<br>$x_1 = x_c$ |

Table 4.2: A summary of the characteristics of Algorithms 3.2 and 4.4 where preconditioning is achieved by using the quasi-Newton LMPs. Approach (B) is defined in Section 3.1.

We now want to discuss the implementation issues for the quasi-Newton LMP (4.51). In Algorithm 4.5, matrix-vector products with $D_j$ and $D_j^T$ can be calculated by adapting the L-BFGS two-loop recursion which results in Algorithm 4.6 and 4.7. Each algorithm requires $8jm$ flops and a matrix vector product with $D_0$.

The memory requirement for the quasi-Newton preconditioner $D$ is $(4m+1)j$. Therefore when comparing the cost and memory requirements for the primal approach

**Algorithm 4.6: Compute $\widehat{z} = D\widehat{r}$**

Given $\widehat{r}$ set $v \leftarrow \widehat{r}$
**for** $i = j : -1 : 1$ **do**
  $\eta_i \leftarrow \tau_i t_i^T v$
  $v \leftarrow v - \eta_i \widehat{q}_i$
**end**
$\widehat{z} \leftarrow D_0 v$
**for** $i = 1{:}j$ **do**
  $\zeta \leftarrow \tau_i \varrho_i^T \widehat{z}$
  $\widehat{z} \leftarrow \widehat{z} + \widehat{p}_i(\eta_i - \zeta)$
**end**

**Algorithm 4.7: Compute $w = D^T\widehat{l}$**

Given $\widehat{l}$ set $v \leftarrow \widehat{l}$
**for** $i = j : -1 : 1$ **do**
  $\eta_i \leftarrow \tau_i \widehat{p}_i^T v$
  $v \leftarrow v - \eta_i \varrho_i$
**end**
$w \leftarrow D_0^T v$
**for** $i = 1{:}j$ **do**
  $\zeta \leftarrow \tau_i \widehat{q}_i^T w$
  $w \leftarrow z + t_i(\eta_i - \zeta)$
**end**

with a quasi-Newton LMP given in Table 3.4, the cost for appying the preconditioner for the dual approach is much less than that of the primal approach when $m \ll n$.

#### 4.3.1.2 Preconditioning with the Ritz LMP

We now derive a Ritz LMP for Algorithm 4.3 with a similar idea as in the previous subsection that this preconditioner satisfies Assumption 4.2 so that one to one correspondance between the primal approach and RPCG remains.

We first explain in Lemma 4.4 and 4.5 how to extract the spectral information and the Lanczos vectors generated in $\mathbb{R}^m$ from RPCG Algorithm 4.3. Using later the relations given in these lemmas we derive the Ritz-LMP expressions in dual space.

**Lemma 4.4** *Let $v_i$, $i = 1, \ldots, l$ be $l$ Lanczos vectors generated from (3.51) during PCG Algorithm 2.5 applied to the linear system (3.1) in the form of $A\ s = b$ with a symmetric positive definite preconditioner $P$ and zero initial guess. Assume also that $D$ is a preconditioner for RPCG Algorithm 4.3 satisfying the relation $PH^T = BH^T D$.*

*Then the Lanczos vectors* $\widehat{v}_i$, $i = 1, \ldots, l$ *generated from the formula*

$$\widehat{v}_{i+1} = (-1)^i \frac{\widehat{r}_i}{\sqrt{\widehat{r}_i^T w_i}} \tag{4.57}$$

*during RPCG Algorithm 4.3 applied to the linear system (4.22) in the form of* $\widehat{A}\,\widehat{s} = \widehat{b}$ *satisfy the relation*

$$v_i = H^T \widehat{v}_i. \tag{4.58}$$

*Let us define* $g_i = P v_i$ *to be the preconditioned Lanczos vectors in* $\mathbb{R}^n$. *Then, the preconditioned Lanczos vectors in* $\mathbb{R}^m$ *defined by*

$$\widehat{g}_i = D\widehat{v}_i \tag{4.59}$$

*satisfy the relation*

$$g_i = BH^T \widehat{g}_i. \tag{4.60}$$

*The column vectors* $\widehat{g}_i$ *construct an orthonormal basis with respect to the* $HBH^T$*-inner product for the Krylov subspace* $\mathcal{K}^l(D\widehat{A}, D\widehat{r}_0)$. *The projected matrix given by (3.55) can be written in terms of the matrix* $\widehat{G}_l = [\widehat{g}_1, \ldots, \widehat{g}_l]$ *as*

$$T_l = \widehat{G}_l^T HBH^T \widehat{A} \widehat{G}_l \tag{4.61}$$

*which defines an orthogonal projection of* $\widehat{A}$ *with respect to the* $HBH^T$*-inner product onto the Krylov subspace* $\mathcal{K}^l(D\widehat{A}, D\widehat{r}_0)$.

**Proof.** From (4.24) and the assumption on the preconditioners, the residuals $r_i = b - As_i$ and $\widehat{r}_i = \widehat{b}_i - \widehat{A}\widehat{s}_i$ generated by Algorithm 2.5 and Algorithm 4.3 respectively satisfy the following relation (Gratton and Tshimanga, 2009)

$$r_i = H^T \widehat{r}_i.$$

with $i = 0, \ldots, l-1$. Substituting this relation into (3.51) and using the relation $PH^T = BH^T D$ gives that

$$v_{i+1} = (-1)^i \frac{r_i}{\|r_i\|_P} = (-1)^i \frac{H^T \widehat{r}_i}{\|H^T \widehat{r}_i\|_P} = H^T \left[ (-1)^i \frac{\widehat{r}_i}{\|\widehat{r}_i\|_{HBH^T D}} \right] = H^T \widehat{v}_{i+1}$$

which proves the first part of the Lemma. Using this relation with (4.60) and the assumption on preconditioners in the definition of $g_i$ gives that

$$g_i = PH^T \widehat{v}_i = BH^T D \widehat{v}_i = BH^T \widehat{g}_i.$$

We now show that $\widehat{g}_i$ vectors form an orthornormal basis for $\mathcal{K}^l(D\widehat{A}, D\widehat{r}_0)$. Using the relations (4.24), $PH^T = BH^T D$ and the fact that $ABH^T = H^T \widehat{A}$, the subspace

$\mathcal{K}^l(PA, Pr_0)$ spanned by the vectors $g_i$ can be written as

$$\begin{aligned}\mathcal{K}^l(PA, Pr_0) &= \text{span}\{PH^T\widehat{r}_0, PAPH^T\widehat{r}_0, \ldots, (PA)^{l-1}PH^T\widehat{r}_0\} \\ &= \text{span}\{BH^TD\widehat{r}_0, PABH^TD\widehat{r}_0, \ldots, (PA)^{l-1}BH^TD\widehat{r}_0\} \\ &= \text{span}\{BH^TD\widehat{r}_0, BH^TD\widehat{A}D\widehat{r}_0, \ldots, BH^T(D\widehat{A})^{l-1}D\widehat{r}_0\} \\ &= BH^T\mathcal{K}^l(D\widehat{A}, D\widehat{r}_0). \end{aligned} \tag{4.62}$$

From this relation and (4.60), it can be seen that the columns of the matrix $\widehat{G}_l$ form an orthonormal basis for the Krylov subspace $\mathcal{K}^l(D\widehat{A}, D\widehat{r}_0)$.

It remains to show that $T_l$ matrix (3.55) can be written as

$$\begin{aligned} T_l &= \widehat{G}_l^T HBABH^T\widehat{G}_l \\ &= \widehat{G}_l^T HBH^T\widehat{A}\widehat{G}_l \end{aligned}$$

where we used the relations (4.60) and $ABH^T = H^T\widehat{A}$. □

**Lemma 4.5** *Suppose that $(\theta_i, u_i) = (\theta_i, V_l\bar{u}_i)$, $i = 1, \ldots, l$ are the Ritz pairs given in Lemma 3.4. Let us define the vectors $\widehat{u}_i$ such that*

$$\widehat{u}_i = \widehat{V}\bar{u}_i \tag{4.63}$$

*where $\widehat{V}_l = [\widehat{v}_1, \ldots, \widehat{v}_l]$ having the column vectors $\widehat{v}_i$ defined in Lemma 4.4. Then the vectors $\widehat{u}_i$ satisfies the relation*

$$u_i = H^T\widehat{u}_i. \tag{4.64}$$

*Suppose also that $x_i$ are the vectors given in Lemma 3.4. Then the vectors defined by*

$$\widehat{x}_i = \widehat{G}_l\bar{u}_i \tag{4.65}$$

*where the matrix $\widehat{G}_l = [\widehat{g}_1, \ldots, \widehat{g}_l]$ is defined in Lemma 4.4 satisfy the relation*

$$x_i = BH^T\widehat{x}_i. \tag{4.66}$$

**Proof.** Using the equations (4.58) and (4.63), from the definition of $u_i$ given by (3.59) we deduce that

$$u_i = V_l\bar{u}_i = H^T\widehat{V}_l\bar{u}_i = H^T\widehat{u}_i. \tag{4.67}$$

The vector $x_i$ can be written from its definition (3.61) as

$$x_i = \mathrm{H}_l\bar{u}_i = BH^T\widehat{G}_l\bar{u}_i$$

where we used the relation (4.60). Therefore, it can be seen that the vectors defined by (4.65) satisfies the relation (4.66).

□

We mentioned that the PCG Algorithm 2.5 searches for an approximate solution

over the set $s_0 + \mathcal{K}^l(PA, Pr_0)$, therefore from Lemma 4.4 it can be seen that RPCG searches for the solution $s$ over the set $s_0 + BH^T\mathcal{K}^l(D\widehat{A}, D\widehat{r}_0)$. Our task is now to use the relations given in Lemma 4.4 and Lemma 4.5 to derive the Ritz preconditioner $D$ satisfying Assumption 4.2. Let us consider applying RPCG Algorithm 4.3 to the linear system (4.22) of the form of $\widehat{A}\,\widehat{s} = \widehat{b}_k$ within Algorithm 4.4.

We suppose that $P_0H^T = BH^TD_0$ where $P_0$ is the initial preconditioner for Algorithm 3.2 and $D_0$ is the initial preconditioner for Algorithm 4.4. Our aim is to find the preconditioner $D_{k-1}$ for the $k$-th linear system in Algorithm 4.4 such that $P_{k-1}H^T = BH^TD_{k-1}$ where $P_{k-1}$ is the Ritz-LMP constructed from (3.64) and used for the $k$-th linear system in Algorithm 3.2.

Assume that when solving the $(k-1)$-th linear systems within Algorithm 3.2 and Algorithm 4.4 the relation $P_{k-2}H^T = BH^TD_{k-2}$ holds. Let us define a matrix $\widehat{U} = [\widehat{u}_1, \ldots, \widehat{u}_l]$ whose columns are defined in Lemma 4.5. Multiplying the equation (3.64) on the left by $H^T$ and substituting the relation (4.64) for $i = 1, \ldots, l$, the relation (4.58) for $i = l+1$ and $P_{k-2}H^T = BH^TD_{k-2}$ into (3.64) gives that

$$\begin{aligned}
P_{k-1}H^T &= P_{k-2}H^T + P_{k-2}U(\Theta^{-1} - I_\ell)U^TP_{k-2}H^T - P_{k-2}U\omega v_{l+1}^TP_{k-2}H^T \\
&\quad -P_{k-2}v_{l+1}\omega^TU^TP_{k-2}H^T + P_{k-2}U\omega\omega^TU^TP_{k-2}H^T, \\
&= BH^TD_{k-2} + P_{k-2}H^T\widehat{U}(\Theta^{-1} - I_\ell)\widehat{U}^THP_{k-2}H^T \\
&\quad -P_{k-2}H^T\widehat{U}\omega\widehat{v}_{l+1}^THP_{k-2}H^T - P_{k-2}H^T\widehat{v}_{l+1}\omega^T\widehat{U}^THP_{k-2}H^T \\
&\quad +P_{k-2}H^T\widehat{U}\omega\omega^T\widehat{U}^THP_{k-2}H^T, \\
&= BH^TD_{k-2} + BH^TD_{k-2}\widehat{U}(\Theta^{-1} - I_\ell)\widehat{U}^TD_{k-2}^THBH^T \\
&\quad -BH^TD_{k-2}\widehat{U}\omega\widehat{v}_{l+1}^TD_{k-2}^THBH^T - BH^TD_{k-2}\widehat{v}_{l+1}\omega^T\widehat{U}^TD_{k-2}^THBH^T \\
&\quad +BH^TD_{k-2}\widehat{U}\omega\omega^T\widehat{U}^TD_{k-2}^THBH^T, \\
&= BH^T\Big[D_{k-2} + D_{k-2}\widehat{U}(\Theta^{-1} - I_\ell)\widehat{U}^TD_{k-2}^THBH^T \\
&\quad -D_{k-2}\widehat{U}\omega\widehat{v}_{l+1}^TD_{k-2}^THBH^T - D_{k-2}\widehat{v}_{l+1}\omega^T\widehat{U}^TD_{k-2}^THBH^T \\
&\quad +D_{k-2}\widehat{U}\omega\omega^T\widehat{U}^TD_{k-2}^THBH^T\Big],
\end{aligned}$$

from which we can write the preliminary formula for $D_{k-1}$ as

$$\begin{aligned}
D_{k-1} &= D_{k-2} + D_{k-2}\widehat{U}(\Theta^{-1} - I_\ell)\widehat{U}^TD_{k-2}^THBH^T - D_{k-2}\widehat{U}\omega\widehat{v}_{l+1}^TD_{k-2}^THBH^T \\
&\quad -D_{k-2}\widehat{v}_{l+1}\omega^T\widehat{U}^TD_{k-2}^THBH^T + D_{k-2}\widehat{U}\omega\omega^T\widehat{U}^TD_{k-2}^THBH^T. \qquad (4.68)
\end{aligned}$$

This formula can be written in a more compact form as explained below. Let us define a matrix $\widehat{X} = [\widehat{x}_1, \ldots, \widehat{x}_l]$ whose column vectors are defined in Lemma 4.5. From the relations (4.65), (4.59) and (4.63) for $i = 1, \ldots, \ell$, it can be easily seen that

$$D_{k-2}\widehat{U} = \widehat{X}.$$

Substituting this relation with the relation (4.59) for $i = \ell + 1$ into (4.68) gives that

$$\begin{aligned} D_{k-1} &= D_{k-2} + \widehat{X}(\Theta^{-1} - I_\ell)\widehat{X}^T HBH^T - \widehat{X}\omega\widehat{g}_{l+1}^T HBH^T - \widehat{g}_{l+1}\omega^T\widehat{X}^T HBH^T \\ &\quad + \widehat{X}\omega\omega^T\widehat{X}^T HBH^T. \end{aligned} \tag{4.69}$$

In order to obtain the successive preconditioner updates for Ritz LMP (4.69) we first need to obtain the triadiagonal matrix $T_l$ (see Sect. 2.6.4). Then the diagonal matrix $\Theta$ (having eigenvalues of the matrix $T_l$ on its diagonal) and the vector $\omega$ (defined by (3.48)) can be calculated from the eigenpairs of the matrix $T_l$ and the scalar $\beta_{\ell+1}$. This scalar can be obtained from formula (2.36) where $\eta_{\ell+1} = \beta_{\ell+1}$. The vectors $\widehat{x}_i$ are calculated from (4.65) by using the matrix $\widehat{G}_l$ whose column vectors are generated from

$$\widehat{g}_{i+1} = (-1)^i \frac{\widehat{z}_i}{\sqrt{\widehat{r}_i^T w_i}}, \tag{4.70}$$

where we used the relations (4.57) and (4.59). The vector $\widehat{g}_{l+1}$ can also be calculated from this expression for $i = l$.

In the LMP formula (4.69), the vectors $HBH^T\widehat{x}_i$ and $HBH^T\widehat{g}_{l+1}$ require matrix-vector product with $HBH^T$ which may be computationally expensive. This can be handled by storing an additional matrix and obtaining $HBH^T$ matrix-vector products from a by-product of Algorithm 4.3. This is explained as follows:

Let us define a vector $\upsilon_i$ such that

$$\upsilon_{i+1} = HBH^T\widehat{g}_{i+1} = (-1)^i \frac{w_i}{\sqrt{\widehat{r}_i^T w_i}}. \tag{4.71}$$

Then, using the relations (4.65) and (4.71) the vectors $y_i = HBH^T\widehat{x}_i$ can be calculated from

$$y_i = HBH^T\widehat{x}_i = \Upsilon_l \bar{u}_i \tag{4.72}$$

where $\Upsilon_l = [\upsilon_1, \ldots, \upsilon_l]$. The vector $\upsilon_{l+1}$ can also be obtained from (4.71) for $i = l$. Defining a matrix $Y = [y_1, \ldots, y_\ell]$ and substituting the relations (4.72) for $i = 1, \ldots, \ell$, and (4.71) into (4.69) yields that

$$\boxed{D_{k-1} = D_{k-2} + \widehat{X}(\Theta^{-1} - I_l)Y^T - \widehat{X}\,\omega\,\upsilon_{l+1}^T - \widehat{g}_{l+1}\,\omega^T\,Y^T + \widehat{X}\omega\,\omega^T Y^T}. \tag{4.73}$$

This formula requires to generate and store matrices $\widehat{G}_l$, $\Upsilon_l$ and $T_l$. Once we have all the required information from RPCG Algorithm 4.3, Algorithm 4.8 summarizes how to built the the Ritz-LMP (4.73).

*As a result it is possible to find the corresponding Ritz LMP (LMP (4.73)) for Algorithm 4.3 such that when it is used within Algorithm 4.4, it generates a mathematically equivalent sequence of iterates $\{s_k\}$ to those of Algorithm 3.2 with the Ritz LMP (3.65), while possibly yielding gains in terms of both memory usage and computational cost when $m \ll n$. We showed that this preconditioner can be constructed from vectors which are by-products of Algorithm 4.3.*

We summarize the characteristics of the equivalent primal and dual algorithms

**Algorithm 4.8: Construct the Ritz-LMP for the matrix $\widehat{A}$**

1 Given $D_{k-2}$, $\widehat{G}_\ell$, $\Upsilon_l$, $T_\ell$, $v_{\ell+1}$, $\widehat{g}_{\ell+1}$ and $\beta_{\ell+1}$ from RPCG Algorithm 4.3

2 Calculate the eigenpairs $(\theta_i, \bar{u}_i)$, $i = 1, \ldots, \ell$ of the tridiagonal matrix $T_\ell$:

$$T_\ell \, \bar{U} = \Theta \, \bar{U}$$

where $\Theta = diag(\theta_i)$ and $\bar{U} = [\bar{u}_1, \ldots, \bar{u}_\ell]$.

3 Generate the vector $\omega = (\omega_1, ..., \omega_\ell)^T$ with $\omega_i$ calculated from the formula (3.48)

4 Generate the matrix $Y = [y_1, \ldots, y_\ell]$ whose column vectors are calculated from

$$y_i = \Upsilon_l \bar{u}_i$$

5 Generate the matrix $\widehat{X} = [\widehat{x}_1, \ldots, \widehat{x}_\ell]$ whose column vectors are calculated from

$$\widehat{x}_i = \widehat{G}_l \bar{u}_i$$

6 Construct the Ritz-LMP from the formula (4.73)

when used with the Ritz LMP by Table 4.3.

| Algorithm | inner min. | Ritz LMP formula | Assumption |
|---|---|---|---|
| 3.2 | Approach (B) | $P_k$ given by (3.65) | $x_1 = x_c$, $s_0 = 0$ |
| 4.4 | Approach (DualD) | $D_k$ given by (4.73) | $P_0 H^T = B H^T D_0$ <br> $x_1 = x_c$ |

Table 4.3: A summary of the characteristics of Algorithms 3.2 and 4.4 where preconditioning is achieved by using the Ritz LMPs. Approach (A) and (DualD) are the approaches defined in Section 3.1 and 4.3.1.1 respectively.

The cost for applying preconditioner $D_{k-1}$ given by (4.73) is $(8lm + 4m)$ flops and one matrix-vector product with $D_{k-2}$. The memory requirement is $(m + 2)l + m$ scalars. Therefore when comparing the cost and memory requirements for the primal approach with a Ritz LMP given in Table 3.6, the cost for appying the preconditioner in the dual approach is much less than that of the primal approach when $m \ll n$.

#### 4.3.1.3 Preconditioning with the spectral LMP

In this section, we derive the spectral LMP to precondition the linear system (4.22) during RPCG Algorithm 4.3. This spectral LMP satisfying Assumption 4.2 is given by the following lemma.

**Lemma 4.6** *Suppose that $HBH^T\widehat{M} = \widehat{M}^T HBH^T$ where $HBH^T$ is a nonsingular matrix. Let $(\lambda_i, y_i)$ be eigenpairs of $(\widehat{A}\widehat{M})^T$, with $y_i = HBH^T\widehat{M}\,\widehat{u}_i$, $i = 1, \ldots, l$ be $l$*

*independent eigenvectors such that* $y_i^T \widehat{u}_i = \delta_{ij}$. *Then the LMP matrix given by (4.26) with* $\widehat{S} = \widehat{X}$, $\widehat{X} = [\widehat{M}\widehat{u}_1, \ldots, \widehat{M}\widehat{u}_l]$ *can be written as*

$$\boxed{D = \widehat{M} + \widehat{X}(\Lambda^{-1} - I_l)\widehat{X}^T H B H^T}, \tag{4.74}$$

*where* $\Lambda = diag(\lambda_i)$.

**Proof.** Since $y_i$ are the eigenvectors of $\widehat{M}^T\widehat{A}^T$, we can write

$$\widehat{M}^T\widehat{A}^T H B H^T \widehat{M}\widehat{U} = H B H^T \widehat{M}\widehat{U}\Lambda, \tag{4.75}$$

where $\widehat{U} = [\widehat{u}_1, \ldots, \widehat{u}_l]$. From $y_i^T\widehat{u}_i = \delta_{ij}$, we can also write that

$$\widehat{U}^T H B H^T \widehat{M}\widehat{U} = I_l.$$

These relations with the symmetry property of $H B H^T \widehat{M}$ and $H B H^T \widehat{A}$ imply that

$$\begin{aligned} (\widehat{X}^T H B H^T \widehat{A}\widehat{X})^{-1} &= \Lambda^{-1} \\ \widehat{X}(\widehat{X}^T H B H^T \widehat{A}\widehat{X})^{-1}\widehat{X}^T H B H^T \widehat{A}\widehat{M} &= \widehat{X}\widehat{X}^T H B H^T \\ \widehat{A}\widehat{X} &= \widehat{U}\Lambda \end{aligned} \tag{4.76}$$

Substituting these relations into (4.26) yields that

$$\begin{aligned} D &= (\widehat{M} - \widehat{X}\widehat{X}^T H B H^T)(I_m - \widehat{A}\widehat{X}\Lambda^{-1}\widehat{X}^T H B H^T) + \widehat{X}\Lambda^{-1}\widehat{X}^T H B H^T \\ &= \widehat{M} - \widehat{M}\widehat{A}\widehat{X}\Lambda^{-1}\widehat{X}^T H B H^T - \widehat{X}\widehat{X}^T H B H^T \\ &\quad + \widehat{X}\widehat{X}^T H B H^T \widehat{A}\widehat{X}\Lambda^{-1}\widehat{X}^T H B H^T + \widehat{X}\Lambda^{-1}\widehat{X}^T H B H^T \\ &= \widehat{M} - \widehat{X}\widehat{X}^T H B H^T - \widehat{X}\widehat{X}^T H B H^T + \widehat{X}\widehat{X}^T H B H^T + \widehat{X}\Lambda^{-1}\widehat{X}^T H B H^T \\ &= \widehat{M} + \widehat{X}(\Lambda^{-1} - I_l)\widehat{X}^T H B H^T \end{aligned}$$

which gives the desired result. □

We now show that the preconditioner $D_{k-1}$ constructed from the formula (4.74) for the $k$-th linear system in Algorithm 4.4 satisfy the relation $P_{k-1}H^T = BH^T D_{k-1}$ where $P_{k-1}$ is the spectral LMP constructed from (3.78) for the $k$-th linear system in Algorithm 3.2.

We suppose that $P_0 H^T = BH^T D_0$ where $P_0$ is the initial preconditioner for Algorithm 3.2 and $D_0$ is the initial preconditioner for Algorithm 4.4. Assume that for the $(k-1)$-th linear system in Algorithm 3.2, the preconditioner $P_{k-2}$ is generated from the formula (3.78) where $(\lambda_i, x_i) = (\lambda_i, P_{k-2}u_i)$, $i = 1, \ldots, l$ are eigenpairs of the matrix $P_{k-2}A$, i.e.

$$P_{k-2}Ax_i = x_i\lambda_i. \tag{4.77}$$

and $(\lambda_i, u_i)$ are eigenpairs of the matrix $AP_{k-2}$, i.e.

$$AP_{k-2}u_i = u_i\lambda_i. \tag{4.78}$$

Assume also that $P_{k-2}H^T = BH^T D_{k-2}$ where $D_{k-2} = \widehat{M}$ is defined in Lemma (4.6). Multiplying equation (4.76) from left by $H^T$ gives that

$$H^T \widehat{A} D_{k-2} \widehat{U} = H^T \widehat{U} \Lambda$$

from which we obtain that

$$AP_{k-2}H^T \widehat{U} = H^T \widehat{U} \Lambda, \tag{4.79}$$

by using the relations $P_{k-2}H^T = BH^T D_{k-2}$ and $BH^T A = H^T \widehat{A}$. From (4.78) and (4.79) it can be easily seen that $U = H^T \widehat{U}$ where $U = [u_1, \ldots, u_l]$. Using this relation in (4.75), we get $X = BH^T \widehat{X}$, $X = [x_1, \ldots, x_l]$. Therefore, from Lemma 4.1 the relation $P_{k-1}H^T = BH^T D_{k-1}$ is satisfied.

*As a result it is possible to find the corresponding spectral LMP (LMP (4.74)) for Algorithm 4.3 such that when it is used within Algorithm 4.4, it generates a mathematically equivalent sequence of iterates $\{s_k\}$ to those of Algorithm 3.2 with the spectral LMP (3.78).*

We summarize the characteristics of the equivalent primal and dual algorithms when used with the spectral LMP by Table 4.4.

| Algorithm | inner min. | spectral LMP formula | Assumption |
|---|---|---|---|
| 3.2 | Approach (B) | $P_k$ given by (3.78) | $x_1 = x_c$, $s_0 = 0$ |
| 4.4 | Approach (DualD) | $D_k$ given by (4.74) | $P_0 H^T = BH^T D_0$<br>$x_1 = x_c$ |

Table 4.4: A summary of the characteristics of Algorithms 3.2 and 4.4 where preconditioning is achieved by using the spectral LMPs. Approach (A) and (DualD) are the approaches defined in Section 3.1 and 4.3.1.1 respectively.

The cost for applying the spectral preconditioner given by (4.74) is $(2m+1)\ell$ flops and one matrix-vector product with $\widehat{M}$.

## 4.4 Solving the subproblem with Augmented RPCG

In this section, we present the final version of the RPCG algorithm that allows for an arbitrary initial guess. This generalized algorithm is proposed by (Gratton and Tshimanga, 2009) and relies on augmentation. We name this algorithm *Augmented RPCG*.

By using augmentation the linear system (3.1) is transformed into an equivalent linear system, i.e.

$$(B^{-1} + \underline{H}_k^T \underline{R}^{-1} \underline{H}_k) s = B^{-1} s_0 + \underline{H}_k^T \underline{d}_k, \tag{4.80}$$

having an initial residual in the range of an augmented matrix $\underline{H}_k^T \in \mathbb{R}^{(m+1)\times n}$, i.e.

$$r_0 = \underline{H}_k^T (\underline{d}_k - \underline{R}_k^{-1} \underline{H}_k s_0), \tag{4.81}$$

(satisfying the assumption on the initial residual). For instance, $\underline{H}_k$, $\underline{R}^{-1} \in \mathbb{R}^{(m+1)\times n}$ and the vector $\underline{d}_k \in \mathbb{R}^{m+1}$ are defined as

$$\underline{H}_k = \begin{bmatrix} H_k \\ (x_c - x_k - s_0)^T B^{-1} \end{bmatrix}, \quad \underline{R}_k^{-1} = \begin{bmatrix} R^{-1} & \\ & 0 \end{bmatrix}, \quad \underline{d}_k = \begin{bmatrix} -R^{-1} d_k \\ 1 \end{bmatrix}.$$

As a result applying PCG Algorithm 2.5 on the augmented linear system generates the same iterates as those of PCG applied to the linear system (3.1). Therefore, an equivalent algorithm (Algorithm 4.9) to PCG on the augmented system consisting the recurrence relations in $\mathbb{R}^{m+1}$ can be obtained (Gratton and Tshimanga, 2009) under the following assumption on the preconditioner:

**Assumption 4.3** *There exists a preconditioner* $\underline{D} \in \mathbb{R}^{(m+1)\times(m+1)}$ *for a given preconditioner* $P$ *used during PCG satisfying*

$$P\underline{H}_k^T = B\underline{H}_k^T \underline{D}.$$

Note that this assumption is an adaptation of Assumption 4.2 when augmentation is introduced.

**Algorithm 4.9: Augmented RPCG Algorithm**

Choose an initial estimate $s_0$
$\lambda_0 = 0$
$\underline{r}_0 = \underline{d} - \underline{R}^{-1}\underline{H} s_0$
$\underline{z}_0 = \underline{D}\, \underline{r}_0$
$\underline{p}_0 = \underline{z}_0$
$\underline{w}_0 = \underline{H} B \underline{H}^T \underline{z}_0$
$\rho_0 = \underline{r}_0^T \underline{w}_0$
$\underline{t}_0 = \underline{w}_0$
**for** $i = 0, 1, ..., l-1$ **do**
    $\underline{q}_i = \underline{R}^{-1}\underline{t}_i + \underline{p}_i$
    $\alpha_i = \rho_i / \underline{q}_i^T \underline{t}_i$
    $\lambda_{i+1} = \lambda_i + \alpha_i \underline{p}_i$
    $\underline{r}_{i+1} = \underline{r}_i - \alpha_i \underline{q}_i$
    $\underline{z}_{i+1} = \underline{D}\, \underline{r}_{i+1}$
    $w_{i+1} = \underline{H} B \underline{H}^T \underline{z}_{i+1}$
    Check convergence and stop if desired accuracy is reached
    $\rho_{i+1} = \underline{r}_{i+1}^T \underline{w}_{i+1}$
    $\beta_i = \rho_{i+1} / \rho_i$
    $\underline{p}_{i+1} = \underline{z}_{i+1} + \beta_i \underline{p}_i$
    $\underline{t}_{i+1} = \underline{w}_{i+1} + \beta_i \underline{t}_i$
**end**
The solution is recovered from $s_l = s_0 + B\underline{H}^T \lambda_l$

Defining the vectors $a$ and a scalar $\sigma$ (Gratton and Tshimanga, 2009)

$$a = H_k(x_c - x_k - s_0)$$
$$\sigma = (x_c - x_k - s_0)^T B^{-1}(x_c - x_k - s_0)$$

the matrix $\underline{H}_k B \underline{H}_k^T$ can be written as

$$\underline{H}_k B \underline{H}_k^T = \begin{bmatrix} H_k \\ (x_c - x_k - s_0)^T B^{-1} \end{bmatrix} B \left[ H_k^T, B^{-1}(x_c - x_k - s_0) \right] = \begin{bmatrix} H_k B H_k^T & a \\ a^T & \sigma \end{bmatrix}$$

Substituting these definitions into Algorithm 4.9, the algorithm can be decomposed in Algorithm 4.10 (Gratton and Tshimanga, 2009, Algorithm 8) which avoids explicit reference to the augmented matrices.

**Algorithm 4.10: Augmented RPCG Algorithm**

Choose an initial estimate $s_0$
$\lambda_0 = 0$
Compute the vector $a$
Compute the scalar $\sigma$
$\underline{r}_0(1:m) = d - R^{-1}Hs_0$
$\underline{r}_0(m+1) = 1$
$\underline{z}_0 = \underline{D}\,\underline{r}_0$
$\underline{p}_0 = \underline{z}_0$
$\underline{w}_0(1:m) = HBH^T\underline{z}_0(1:m) + \underline{z}_0(m+1)a$
$\underline{w}_0(m+1) = a^T\underline{z}_0(1:m) + \sigma\underline{z}_0(m+1)$
$\rho_0 = \underline{r}_0^T\underline{w}_0$
$\underline{t}_0 = \underline{w}_0$
**for** $i = 0, 1, ..., l-1$ **do**
    $\underline{q}_i(1:m) = R^{-1}\underline{t}_i(1:m) + \underline{p}_i(1:m)$
    $\underline{q}_i(m+1) = \underline{p}_i(m+1)$
    $\alpha_i = \rho_i / \underline{q}_i^T\underline{t}_i$
    $\lambda_{i+1} = \lambda_i + \alpha_i\underline{p}_i$
    $\underline{r}_{i+1} = \underline{r}_i - \alpha_i\underline{q}_i$
    $\underline{z}_{i+1} = \underline{D}\,\underline{r}_{i+1}$
    $\underline{w}_{i+1}(1:m) = HBH^T\underline{z}_{i+1}(1:m) + \underline{z}_{i+1}(m+1)a$
    $\underline{w}_{i+1}(m+1) = a^T\underline{z}_{i+1}(1:m) + \sigma\underline{z}_{i+1}(m+1)$
    Check convergence and stop if desired accuracy is reached
    $\rho_{i+1} = \underline{r}_{i+1}^T\underline{w}_{i+1}$
    $\beta_i = \rho_{i+1}/\rho_i$
    $\underline{p}_{i+1} = \underline{z}_{i+1} + \beta_i\underline{p}_i$
    $\underline{t}_{i+1} = \underline{w}_{i+1} + \beta_i\underline{t}_i$
**end**
The solution is recovered from $s_l = s_0 + BH^T\lambda_l(1:m) + \lambda_l(m+1)(x_c - x_k - s_0)$

In Algorithm 4.10, the notation is adapted similar to that of MATLAB, for instance $\underline{r}_0(1:m)$ refers to the first $m$ entries of the vector $\underline{r}_0 \in \mathbb{R}^{m+1}$ and $\underline{r}_0(m+1)$ refers to

the last (augmented) entry of the vector.

The comparison on required operations during the inner loops of RPCG Algorithm 4.3 and Augmented RPCG Algorithm 4.10 is given by Table 4.5. From this table it can be seen that the computational cost of Augmented RPCG Algorithm 4.10 is very close to that of RPCG Algorithm 4.3.

| Algorithm | flops req. | matvec pr. | precond op. |
|---|---|---|---|
| RPCG | 12m - 2 | $B$, $H^T$, $R^{-1}$ and $H$ | matvec with $D$ |
| A-RPCG | 16m + 13 | $B$, $H^T$, $R^{-1}$ and $H$ | matvec with $\underline{D}$ |

Table 4.5: Summary of operations for each inner iteration $i$ of RPCG and Augmented RPCG (A-RPCG): flops requirements, matrix-vector product, required operations to apply preconditioner.

Algorithm 4.10 can be integrated into a Gauss-Newton algorithm given by Algorithm 4.11. This algorithm finds the solution of a sequence of varying linear systems (3.1) by performing minimization in $\mathbb{R}^{m+1}$.

**Algorithm 4.11: Gauss-Newton method with Augmented RPCG**

1 **Initialization:** Choose an initial preconditioner $\underline{D}_0$ such that Assumption 4.3 holds. Choose an initial estimate $s_0$

2 **Perform inner loop**: Solve the linear system (3.1) with Augmented RPCG Algorithm 4.10. During Augmented RPCG extract and save relevant information to precondition the next linear system using $\underline{D}_k$ such that $P_k \underline{H}_{k+1}^T = B \underline{H}_{k+1}^T \underline{D}_k$.

3 **Update the iterate:**

$$x_{k+1} = x_k + s_k$$

4 Increment $k$ by 1 and go to step 2.

### 4.4.1 Preconditioning Augmented RPCG with LMPs

In this section we are interested in preconditioning the linear system at the $k$-th outer loop of Algorithm 4.11 with a preconditioner inherited from the previous outer loop. So the question is whether we may derive a preconditioner $\underline{D} \in \mathbb{R}^{(m+1)\times(m+1)}$ from by-products of Algorithm 4.10 satisfying Assumption 4.3.

We want to find a formula for the LMP so that Augmented RPCG would behave the same way as non augmented RPCG whenever augmentation is not needed, i.e. there exists a vector $c_k \in \mathbb{R}^m$ such that $B^{-1}(x_c - x_k - s_0) = H^T c_k$. The following lemma provides such a preconditioner.

**Lemma 4.7** *Suppose that* $\underline{H}_k$ *is defined as*

$$\underline{H}_k = \begin{bmatrix} H \\ (x_c - x_k - s_0)^T B^{-1} \end{bmatrix}$$

*where* $H \in \mathbb{R}^{m\times n}$ *has full row rank. Suppose also that there exists a vector* $c_k \in \mathbb{R}^m$ *such that* $B^{-1}(x_c - x_k - s_0) = H^T c_k$. *Then the matrix* $\underline{D}$ *defined by*

$$\underline{D} = \begin{bmatrix} D & Dc_k - c_k \\ 0 & 1 \end{bmatrix} \tag{4.82}$$

*where the matrix* $HBH^T D$ *is symmetric and positive definite satisfies* $\underline{H}B\underline{H}^T\underline{D} = \underline{D}^T\underline{H}B\underline{H}^T$.

*If* $PH^T = BH^T D$ *holds for a given matrix* $P \in \mathbb{R}^{n\times n}$, *then the matrix* $\underline{D}$ *defined by (4.82) satisfies* $P\underline{H}^T = B\underline{H}^T\underline{D}$.

**Proof.** Replacing the relation $B^{-1}(x_c - x_k - s_0) = H^T c_k$ in $\underline{H}$ and multiplying $\underline{D}$ on the left gives that

$$\underline{H}B\underline{H}^T\underline{D} = \begin{bmatrix} HBH^T D & HBH^T Dc_k \\ c_k^T HBH^T D & c_k^T HBH^T Dc_k \end{bmatrix}.$$

Using the symmetry of $D$ in $HBH^T$, we deduce that

$$\begin{aligned} \underline{H}B\underline{H}^T\underline{D} &= \begin{bmatrix} D^T HBH^T & D^T HBH^T c_k \\ c_k^T D^T HBH^T & c_k^T D^T HBH^T c_k \end{bmatrix} \\ &= \underline{D}^T\underline{H}B\underline{H}^T. \end{aligned}$$

The second part of the proof is as follows. Using the assumption $PH^T = BH^T D$ given in the lemma, we obtain that

$$\begin{aligned} P\underline{H}^T &= \begin{bmatrix} PH^T & PH^T c_k \end{bmatrix} \\ &= \begin{bmatrix} BH^T D & BH^T Dc_k \end{bmatrix} \\ &= B\underline{H}^T\underline{D}, \end{aligned}$$

which completes the proof. □

After defining a preconditioner in $\mathbb{R}^{m+1}$ by Lemma 4.7, we are left with finding such a vector $c_k$ and a matrix $D$ given in Lemma 4.7 from by-products of Algorithm 4.10. Let us start with explaining how to obtain the vector $c_k$.

We showed in Subsection 3.1.1 that $(x_k - x_c)$ can be written as

$$x_k - x_c = BH^T \sum_{j=1}^{k-1} \widehat{s}_j \tag{4.83}$$

under the assumption that $x_1 = x_c$ where $\widehat{s}_j$ is the solution of Algorithm 4.4 at the $j$-th outer loop. Therefore taking $s_0 = 0$, and choosing the vector $c_k$ as

$$c_k = -\sum_{j=1}^{k-1} \widehat{s}_j \tag{4.84}$$

satisfies the assumption in Lemma 4.7.

We showed in Section 4.3 that it is possible to derive a preconditioner $D$ from RPCG Algorithm 4.3 such that $HBH^TD$ is symmetric and positive definite. It is also shown that this preconditioner also satisfies the relation $PH^T = BH^TD$ for a given $P$ constructed from PCG Algorithm. Therefore, it is possible to derive a preconditioner $\underline{D}$ from the pairs obtained during RPCG Algorithm 4.3 that satisfies the relation $P\underline{H}^T = B\underline{H}^T\underline{D}$. The question is now how to obtain these $m$-dimensional pairs from Augmented RPCG. The answer is as follows.

When solving the $k$-th linear systems within Algorithms 3.2 and 4.4 the relations between the $m$-dimensional vectors of RPCG Algorithm 4.3 and $n$-dimensional vectors of PCG Algorithm 2.5 applied to the linear system (3.1) with zero initial guess are given as

$$\left.\begin{aligned} r_i &= H^T\widehat{r}_i, \\ z_i &= BH^T\widehat{z}_i \\ p_i &= BH^T\widehat{p}_i \\ q_i &= H^T\widehat{q}_i \\ s_i &= BH^T\widehat{s}_i \end{aligned}\right\} \tag{4.85}$$

with $i \geq 0$ under the assumption that $PH^T = BH^TD$.

The relation between the $(m+1)$-dimensional vectors of Augmented RPCG Algorithm 4.9 and those of PCG Algorithm can be similarly given as

$$\left.\begin{aligned} r_i &= \underline{H}_k^T\underline{r}_i \\ z_i &= B\underline{H}_k^T\underline{z}_i \\ p_i &= B\underline{H}_k^T\underline{p}_i \\ q_i &= \underline{H}_k^T\underline{q}_i \\ s_i &= B\underline{H}_k^T\lambda_i \end{aligned}\right\} \tag{4.86}$$

with $i \geq 0$ under the assumption that $P\underline{H}_k^T = B\underline{H}_k^T\underline{D}$. From the definition of $c_k$ and the relation (4.83), $\underline{H}_k^T$ can be written as

$$\underline{H}_k^T = \begin{bmatrix} H^T & H^Tc_k \end{bmatrix}.$$

Substituting this definition into the relations given in (4.86) and using the equality of right hand sides of (4.85) and (4.86), $m$-dimensional vectors obtained during RPCG Algorithm can be generated from the $(m+1)$-dimensional vectors obtained during

Augmented RPCG by the following relations:

$$\left.\begin{array}{rcl} \widehat{r}_i &=& \underline{r}_i(1:m) + c_k \underline{r}_i(m+1) \\ \widehat{z}_i &=& \underline{z}_i(1:m) + c_k \underline{z}_i(m+1) \\ \widehat{p}_i &=& \underline{p}_i(1:m) + c_k \underline{p}_i(m+1) \\ \widehat{q}_i &=& \underline{q}_i(1:m) + c_k \underline{q}_i(m+1) \\ \widehat{s}_i &=& \lambda_i(1:m) + c_k \lambda_i(m+1) \end{array}\right\} \tag{4.87}$$

The vector $\underline{t}_i = \underline{H}_k B \underline{H}_k^T \underline{p}_i$ defined in Augmented RPCG Algorithm 4.10 can be written also as

$$\underline{t}_i = \underline{H}_k p_i = \begin{bmatrix} H p_i \\ c_k^T H p_i \end{bmatrix} = \begin{bmatrix} H B H^T \widehat{p}_i \\ c_k^T H p_i \end{bmatrix} = \begin{bmatrix} t_i \\ c_k^T H p_i \end{bmatrix}$$

where $t_i$ is defined in RPCG Algorithm 4.3. Therefore the vector $t_i \in \mathbb{R}^m$ can be obtained from the vector $\underline{t}_i \in \mathbb{R}^{m+1}$ by using the relation

$$t_i = \underline{t}_i(1:m).$$

Similarly, the vector $\underline{w}_i = \underline{H}_k B \underline{H}_k^T \underline{z}_i$ defined in Augmented RPCG Algorithm 4.10 can be written as

$$\underline{w}_i = \underline{H}_k z_i = \begin{bmatrix} H z_i \\ c_k^T H z_i \end{bmatrix} = \begin{bmatrix} H B H^T \widehat{z}_i \\ c_k^T H z_i \end{bmatrix} = \begin{bmatrix} w_i \\ c_k^T H z_i \end{bmatrix}$$

where $w_i$ is defined in RPCG Algorithm 4.3. The vector $w_i \in \mathbb{R}^m$ can be obtained from the vector $\underline{w}_i \in \mathbb{R}^{m+1}$ by using the relation

$$w_i = \underline{w}_i(1:m).$$

Therefore, these relations allow to construct the limited memory preconditioner $D \in \mathbb{R}^{m \times m}$ at the $(k+1)$-th outer loop of Algorithm 4.11 (with the same $H_k$ along the outer loops) from the recurrence relations of Augmented RPCG that satisfies the relation $PH^T = BH^T D$. Then using the preconditioner $D$, a preconditioner $\underline{D} \in \mathbb{R}^{(m+1)\times(m+1)}$ for the $(k+1)$-th outer loop of Algorithm 4.11 can be generated from (4.82) which satisfies the relation $P\underline{H}_{k+1}^T = BH_{k+1}^T \underline{D}$.

In case of using quasi-Newton LMP Augmented RPCG, Algorithm 4.9 can be rewritten in a more computationally effective way like RPCG that does not require to perform additional matrix vector products to calculate $HBH^T\widehat{q}$. Introducing a vector

$$\underline{l}_i = \underline{H} B \underline{H}^T \underline{r}_i \tag{4.88}$$

and using the symmetry of $\underline{D}$ in $\underline{H}B\underline{H}^T$ from Lemma (4.7), Algorithm 4.9 can be transformed into Algorithm 4.12 by using the similar relations used to transform Algorithm 4.3 to Algorithm 4.5. While constructing the quasi-Newton LMP, we need to save the vectors $\varrho_i = HBH^T\widehat{q}_i$ defined in Algorithm 4.5. These vectors can be

obtained from vectors $\underline{\varrho}_i$ as explained below.

The vector $\underline{\varrho}_i = \underline{H} B \underline{H}^T \underline{q}_i$ defined in Augmented RPCG Algorithm 4.10 can be written as

$$\underline{\varrho}_i = \underline{H}_k q_i = \begin{bmatrix} Hq_i \\ c_k^T H z_i \end{bmatrix} = \begin{bmatrix} HBH^T \widehat{q}_i \\ c_k^T H q_i \end{bmatrix} = \begin{bmatrix} \varrho_i \\ c_k^T H q_i \end{bmatrix}$$

Then, the vector $\varrho_i \in \mathbb{R}^m$ can be obtained from the vector $\underline{\varrho}_i \in \mathbb{R}^{m+1}$ by using the relation

$$\varrho_i = \underline{\varrho}_i(1:m).$$

**Algorithm 4.12: Augmented RPCG (version for quasi-Newton LMP)**

Choose an initial estimate $s_0$
$\lambda_0 = 0$
$\underline{r}_0 = \underline{d}_k - \underline{R}^{-1} \underline{H}_k s_0$
$\underline{l}_0 = \underline{H} B \underline{H}^T \underline{r}_0$
$\underline{z}_0 = \underline{D}\, \underline{r}_0$
$\underline{p}_0 = \underline{z}_0$
$\underline{w}_0 = \underline{D}^T \underline{l}_0$
$\rho_0 = \underline{r}_0^T \underline{w}_0$
$\underline{t}_0 = \underline{w}_0$
**for** $i = 0, 1, ..., l-1$ **do**
    $\underline{q}_i = \underline{R}^{-1} \underline{t}_i + \underline{p}_i$
    $\alpha_i = \rho_i / \underline{q}_i^T \underline{t}_i$
    $\lambda_{i+1} = \lambda_i + \alpha_i \underline{p}_i$
    $\underline{r}_{i+1} = \underline{r}_i - \alpha_i \underline{q}_i$
    $\underline{l}_{i+1} = \underline{H} B \underline{H}^T \underline{r}_{i+1}$
    $\underline{\varrho}_i = (\underline{l}_i - \underline{l}_{i+1}) / \alpha_i$
    $\underline{z}_{i+1} = \underline{D}\, \underline{r}_{i+1}$
    $\underline{w}_{i+1} = \underline{D}^T \underline{l}_{i+1}$
    Check convergence and stop if desired accuracy is reached
    $\rho_{i+1} = \underline{r}_{i+1}^T \underline{w}_{i+1}$
    $\beta_i = \rho_{i+1} / \rho_i$
    $\underline{p}_{i+1} = \underline{z}_{i+1} + \beta_i \underline{p}_i$
    $\underline{t}_{i+1} = \underline{w}_{i+1} + \beta_i \underline{t}_i$
**end**
The solution is recovered from $s_l = s_0 + B \underline{H}^T \lambda_l$

### 4.4.2 Preconditioning varying systems

In this section we are interested in solving a sequence of slowly varying linear systems (4.14) in the form of

$$\widehat{A}_k \lambda = \widehat{b}_k \tag{4.89}$$

with conjugate gradients equipped with an LMP. Our aim is not only solving this linear system, but also to preserve the desired convergence properties of the primal approach. For this reason, for varying linear systems Algorithm 4.10 has to be used since this algorithm accomodates the initial residual in the proper space which is a requirement to ensure monotonic decrease on the cost function (2.13). In case of preconditioning another requirement is to find an LMP $\underline{D}$ that satisfies Assumption 4.3 (the augmented version of Assumption 4.2).

When $\widehat{A}_k$ is constant (the case of $H_k$ is constant) along the outer loops, an LMP $\underline{D}$ that satisfies $P\underline{H}_k^T = BH_k^T\underline{D}$ is given by Lemma 4.7. Now, we are interested in using this preconditioner $\underline{D}$ when $H_k$ slowly varies along the outer loops. In this case depending on the change along the outer loops *the preconditioner* $\underline{D}$ *may loose its symmetry according the inner product* $\underline{H}_k B\underline{H}_k^T$ *since* $D$ *may loose its symmetry in* $H_kBH_k^T$. This may happen since $D$ is inherited from the preceding outer loop $(k-1)$ and used in the current outer loop $k$. This symmetry problem can be handled by regenerating the required pairs for the LMP by using the current inner product. For instance, for a quasi-Newton preconditioner, the preconditioner $D$ can be derived from the formula (4.51) as

$$D_j = \left(I_m - \tau_j \widehat{p}_j \widehat{q}_j^T H_k B H_k^T\right) D_{j-1} \left(I_m - \tau_j \widehat{q}_j t_j^T\right) + \tau_j \widehat{p}_j t_j^T, \tag{4.90}$$

where $\widehat{p}_i \in \mathbb{R}^m$, $i = 0, 1, ..., j$ are the search directions generated from the relation given in (4.87) at the $(k-1)$-th outer loop of Algorithm 4.11, $\widehat{q}_i = (I_m + R^{-1}H_kBH_k^T)\widehat{p}_i$, $t_i = H_kBH_k^T\widehat{p}_i$ and $\tau_i = 1/(\widehat{q}_i^T t_i)$. Therefore replacing $H_{k-1}BH_{k-1}^T$ with $H_kBH_k^T$ in the formula makes the preconditioner $D(= D_j)$ be symmetric with respect to the current inner product $H_kBH_k^T$. Generating this preconditioner requires $2j$ matrix-vector products with $H_kBH_k^T$ which may be computationally expensive if the acceleration attained with the preconditioner is not strong enough.

In this situation a measure on the symmetry of the preconditioner $\underline{D}$ can be defined and used as a criterion to precondition the next linear system with $\underline{D}$ or not. A possible necessary condition for symmetry could be $\eta_k < \epsilon$, where

$$\eta_k = \frac{\|\underline{H}_k B\underline{H}_k^T\underline{D}\, x - \underline{D}^T\underline{H}_k B\underline{H}_k^T x\|}{\|\underline{H}_k B\underline{H}_k^T\underline{D}\, x\|}, \tag{4.91}$$

$x \in \mathbb{R}^{m+1}$ is a random vector, and $\epsilon$ is a user defined threshold. This measure requires two more $\underline{H}_k B\underline{H}_k^T$ matrix inner product.

## 4.5 Convergence issues

In this section we present the convergence properties of the variants of RPCG. First the re-orthogonalization will be introduced which is an important issue in the convergence of the conjugate-gradient-like methods in finite arithmetic. Afterwards the convergence theory for RPCG will be provided and the section will be concluded by providing inexpensive formulas for computing the necessary information on each iteration from

the recurrence relations of RPCG algorithms for monitoring the convergence of the minimization.

### 4.5.1 Re-orthogonalization

It is known that round-off errors typically cause loss of orthogonality between successive residuals (Meurant, 2006), a central property ensuring fast convergence of the conjugate-gradient methods in exact arithmetic. As a result the rate of convergence might be substantially deteriorated. A possible cure for this problem is to consider reorthogonalization of the residuals, either explicitly (Roux, 1989) or in the form of the mathematically equivalent Full-Orthogonalization-Method (FOM) (van der Vorst, 2003). We present the reorthogonalization of the residuals using a modified Gram-Schmidt (MGS) procedure (Saad, 1996, p. 11-12) in this subsection.

With reference to Algorithm 2.5 for PCG, the re-orthogonalization procedure acts on the vectors $r_i$. Making use of the orthogonality relationship $r_i^T P r_j = 0$, for $i \neq j$, MGS re-orthogonalization can be described in compact notation by

$$r_i \leftarrow \left[\prod_{j=i-1}^{1}\left(I_n - \frac{r_j r_j^T P}{r_j^T P r_j}\right)\right] r_i, \tag{4.92}$$

$$= \left[\prod_{j=i-1}^{1}\left(I_n - \frac{r_j z_j^T}{r_j^T z_j}\right)\right] r_i. \tag{4.93}$$

Equations (4.92) and (4.93) require the storage of the residuals $r_j$ from all previous iterations $j = 1, \ldots, i-1$. Equation (4.92) can be less practical than Equation (4.93) when matrix vector products with $P$ is expensive. In this case, the vectors $z_j$ can also be stored during PCG and used in Equation (4.93).

The need to store and manipulate the sequence of $n$-dimensional vectors $r_j$ and $z_j$ can lead to significant computational overhead. In this respect, the dual algorithms, which involve sequences of $m$ (or $(m+1)$)-dimensional vectors, are clearly an attractive alternative when $m \ll n$. From the relationship $r_i = H^T \widehat{r}_i$ and $PH^T = BH^T D$ the orthogonality condition for the residuals can be written as (Gratton and Tshimanga, 2009)

$$r_i^T P r_j = \widehat{r}_i^T H P H^T \widehat{r}_j = \widehat{r}_i^T H B H^T D \widehat{r}_j = \widehat{r}_i^T w_i \tag{4.94}$$

for $i \neq j$. Combining Eq. (4.94) with Eqs (4.92) and (4.93) leads to the MGS re-orthogonalization scheme for RPCG as

$$\widehat{r}_i \leftarrow \left[\prod_{j=i-1}^{1}\left(I_m - \frac{\widehat{r}_j \widehat{r}_j^T D^T H B H^T}{\widehat{r}_j^T H B H^T D \widehat{r}_j}\right)\right] \widehat{r}_i,$$

$$= \left[\prod_{j=i-1}^{1}\left(I_m - \frac{\widehat{r}_j w_j^T}{\widehat{r}_j^T w_j}\right)\right] \widehat{r}_i.$$

Note that, the corresponding reorthogonalization scheme for Augmented RPCG

follows the similar derivation.

*It is remarkable that re-orthogonalization of residuals may be considered as costly in terms of memory and computational cost in the primal setting, turns out to be much cheaper in the dual frameworks since the dimension of the residual vectors in dual space is (typically much) smaller than the dimension of the residual vectors in primal space.*

Our observation is that reorthogonalization in the dual space has a good effect on the convergence in the presence of round-off errors. Comparing this reorthogonalization with what is done in the primal space remains an open question.

### 4.5.2 Convergence theory

In this subsection we consider bounds on the efficiency of the dual-space preconditioners and compare them to those that can be derived for its primal-space equivalents. For this purpose, we start by recalling known properties of the preconditioned conjugate-gradient method. From the construction of the PCG method, the iteration error $e_k$ can be related to the initial error $e_0 = s^* - s_0$ by (Axelsson, 1996, p. 560)

$$e_k = (\mathcal{P}_l(PA)PA + I_n)e_0$$

where $s^*$ is the solution of the linear system (3.1) in $\mathbb{R}^n$, $A = B^{-1} + H^T R^{-1} H$ is a symmetric positive definite matrix, $P$ is a symmetric positive definite preconditioner, and $\mathcal{P}_l$ is a polynomial defined by

$$\mathcal{P}_l(PA) = a_0 I + a_1 PA + ... + a_l (PA)^l.$$

Taking $s_0 = 0$, this method implicitly computes the coefficients of the polynomial $\mathcal{P}_l^*(PA)$ that solves the minimization problem (Axelsson, 1996, p. 560)

$$\min_{\mathcal{P}_l} \|(\mathcal{P}_l(PA)PA + I_n)s^*\|_A^2 . \tag{4.95}$$

If $PA$ has eigenvalues $\mu_1 \leq \mu_2 \leq ... \leq \mu_n$, the PCG algorithm (Golub and Van Loan, 1989, p. 534) with zero initial starting vector ensures the inequality

$$\|s_{l+1} - s^*\|_A \leq 2 \left( \frac{\sqrt{\mu_n} - \sqrt{\mu_1}}{\sqrt{\mu_n} + \sqrt{\mu_1}} \right)^l \|s^*\|_A \tag{4.96}$$

(see (Conn, Gould and Toint, 2000, p. 89), for instance). Note that, when a dual approach is used, the iterates all belong to the affine subspace $\mathrm{IM}(BH^T)$ for RPCG or $\mathrm{IM}(B\underline{H}^T)$ for Augmented RPCG. This information can be taken into account to obtain a better bound on the convergence rate. We first find this bound for RPCG Algorithm 4.3 shown in the next lemma.

**Lemma 4.8** *Suppose that $HBH^T$ is non-singular, and that $D$ is a preconditioner satisfying Assumption 4.2. If $D(I_m + R^{-1}HBH^T)$ has eigenvalues $\nu_1 \leq \nu_2 \leq ... \leq \nu_m$, then the RPCG Algorithm 4.3 and its primal equivalent with zero initial starting vector*

*ensure the inequality*

$$\|s_{l+1} - s^*\|_A \leq 2\left(\frac{\sqrt{\nu_m} - \sqrt{\nu_1}}{\sqrt{\nu_m} + \sqrt{\nu_1}}\right)^l \|s^*\|_A \leq 2\left(\frac{\sqrt{\mu_n} - \sqrt{\mu_1}}{\sqrt{\mu_n} + \sqrt{\mu_1}}\right)^l \|s^*\|_A . \tag{4.97}$$

**Proof.** From (4.18) the solution of the linear system (3.1) can be written as $s^* = BH^T\widehat{s}^*$, where $\widehat{s}^*$ is the solution of the linear system (4.22). Substituting this form for the solution in the objective function of (4.95) then yields the new form

$$\left\| \left( \sum_{i=0}^{l} a_i \, (PA)^{i+1} BH^T + BH^T \right) \widehat{s}^* \right\|_A^2$$

and this objective is minimized over all choices of the coefficients $\{a_i\}_{i=0}^l$. Using the fact that $(B^{-1} + H^T R^{-1} H)BH^T = H^T(I_m + R^{-1}HBH^T)$, which we can simply write as $ABH^T = H^T\widehat{A}$, we obtain that our objective may now be written as

$$\left\| \left( \sum_{i=0}^{l} a_i \, (PA)^{i} PH^T \widehat{A} + BH^T \right) \widehat{s}^* \right\|_A^2$$

Using Assumption 4.2, we obtain the further form

$$\left\| \left( \sum_{i=0}^{l} a_i \, (PA)^{i} BH^T D\widehat{A} + BH^T \right) \widehat{s}^* \right\|_A^2 .$$

Substituting the term $ABH^T$ with $H^T\widehat{A}$ and using Assumption 4.2 then yields an objective of the form

$$\begin{aligned} \left\| \left( BH^T \sum_{i=0}^{l} a_i \, (D\widehat{A})^{i+1} + I_m \right) \widehat{s}^* \right\|_A^2 &= \left\| BH^T(\mathcal{P}_l(D\widehat{A})D\widehat{A} + I_m)\widehat{s}^* \right\|_A^2 \\ &= \left\| (\mathcal{P}_l(D\widehat{A})D\widehat{A} + I_m)\widehat{s}^* \right\|_{HBABH^T}^2 \\ &= \left\| (\mathcal{P}_l(D\widehat{A})D\widehat{A} + I_m)\widehat{s}^* \right\|_{HBH^T\widehat{A}}^2 \end{aligned} \tag{4.98}$$

Performing the change of variables $\widetilde{A} = HBH^T\widehat{A}$ and $\widetilde{P} = D(HBH^T)^{-1}$ in (4.98), we may write the minimization problem in dual space as:

$$\min_{\mathcal{P}_l} \left\| (\mathcal{P}_l(\widetilde{P}\widetilde{A})\widetilde{P}\widetilde{A} + I_m)\widehat{s}^* \right\|_{\widetilde{A}}^2 \tag{4.99}$$

Using Assumption 4.2, we then write $(HBH^T)^{-1}HPH^T(HBH^T)^{-1} = D(HBH^T)^{-1}$ which shows that the matrix $\widetilde{P}$ is symmetric positive definite. On the other hand, $\widetilde{A} = HBH^T\widehat{A} = HBH^T + HBH^T R^{-1} HBH^T$ is also a symmetric positive definite matrix. Therefore, from (4.95) and (4.96), if $\widetilde{P}\widetilde{A}$ has eigenvalues $\nu_1 \leq \nu_2 \leq \ldots \leq \nu_m$,

the RPCG Algorithm 4.3 ensures the inequality

$$\|\widehat{s}_{l+1} - \widehat{s}^*\|_{\widetilde{A}} \leq 2 \left( \frac{\sqrt{\nu_m} - \sqrt{\nu_1}}{\sqrt{\nu_m} + \sqrt{\nu_1}} \right)^l \|\widehat{s}^*\|_{\widetilde{A}} \,. \tag{4.100}$$

One also has that

$$\|\widehat{s}^*\|_{HBH^T\widehat{A}} = \|\widehat{s}^*\|_{HBABH^T} = \left\|BH^T\widehat{s}^*\right\|_A = \|s^*\|_A \,. \tag{4.101}$$

$$\|\widehat{s}_{l+1} - \widehat{s}^*\|_{HBH^T\widehat{A}} = \|\widehat{s}_{l+1} - \widehat{s}^*\|_{HBABH^T} = \left\|BH^T(\widehat{s}_{l+1} - \widehat{s}^*)\right\|_A = \|s_{l+1} - s^*\|_A \,. \tag{4.102}$$

Finally, substituting $\widetilde{A}$ with $HBH^T\widehat{A}$ and $\widetilde{P}$ with $D(HBH^T)^{-1}$ in $\widetilde{P}\widetilde{A}$, and using the relations (4.101) and (4.102) in (4.100) proves the first part of the inequality (4.97).

For the second part of the inequality, we start from the equality $ABH^T = H^T\widehat{A}$. If we multiply both sides of this equality from the left by $P$, we obtain $PABH^T = PH^T\widehat{A}$ and using Assumption 4.2, we deduce that $(PA)BH^T = BH^T(D\widehat{A})$. This equality tells us that $BH^T$ spans an invariant subspace of $PA$, from which we may deduce that every eigenvalue of $D\widehat{A}$ is an eigenvalue of $PA$. So, $\mu_1 \leq \nu_1$ and $\mu_n \geqslant \nu_m$. From these two inequalities we obtain that

$$\frac{\sqrt{\mu_1}}{\sqrt{\mu_n}} \leq \frac{\sqrt{\nu_1}}{\sqrt{\nu_m}}.$$

Using the fact that the function $\frac{1-x}{1+x}$ is decreasing for $x \geqslant 0$, we then deduce the desired result and complete the proof. □

This lemma shows that the condition number of $PA$ is generally worse than that of $D\widetilde{A}$ and we now show that it can be arbitrarily worse. For example, taking $m = n-2$, $B$ as the identity matrix, $R$ is a diagonal matrix, $H = [I\ 0]$, $P = [D\ 0;\ 0\ \mathrm{diag}(\xi, 1)]$ where $\mathrm{diag}(\xi, 1)$ is the $2 \times 2$ diagonal matrix with the diagonal entries $\xi$ and 1 and $D = (I_m + R^{-1}HH^T)^{-1}$ we easily verify that Assumption 4.2 holds. Then the diagonal preconditioned system matrices are

$$PA = \mathrm{diag}(d_i) \text{ where } d_i = 1 \text{ for } 1 \leq i \leq m \text{ and } i = n,$$
$$d_i = \xi \text{ for } i = n-1.$$

$$D\widehat{A} = \mathrm{diag}(d_i) \text{ where } d_i = 1 \text{ for } 1 \leq i \leq m.$$

We conclude that $PA$ is ill-conditioned with a condition number of $1/\xi$ whereas $D\widehat{A}$ has a condition number of 1. Therefore, for a given preconditioner $D$ in dual space, we can find a preconditioner $P$ in primal space satisfying Assumption 4.2 that is arbitrarily ill-conditioned, showing the relevance of the bound (4.97) in terms of the $\nu$'s. Note that, in this case the iterates of the dual and primal algorithms are the same and convergence takes place in one iteration both in dual and primal spaces.

Note that we assumed $HBH^T$ to be non-singular for Lemma 4.8, an assumption which has been used to solve the minimization problem (4.99) on the space of polyno-

mials. Lemma 4.8 can nevertheless be generalized to the case where $HBH^T$ is singular, which results in the following improvement on Lemma 4.8.

**Lemma 4.9** *Suppose that $HBH^T$ is singular, and that $H$ is a preconditioner satisfying Assumption 4.2. Then, it is possible to find a $r \times n$ matrix of rank $r$ $\check{H}$ and a $r \times r$ matrix of rank $r$ $\check{R}^{-1}$ satisfying*

$$B^{-1} + H^T R^{-1} H = B^{-1} + \check{H}^T \check{R}^{-1} \check{H} \tag{4.103}$$

*where $r < m$, and a $r \times r$ matrix $\check{D}$ satisfying*

$$P\check{H}^T = B\check{H}^T \check{D}. \tag{4.104}$$

*Moreover, the RPCG Algorithm 4.3 and its primal equivalent ensure the inequality*

$$\|s_{l+1} - s^*\|_A \leq 2 \left( \frac{\sqrt{\nu_r} - \sqrt{\nu_1}}{\sqrt{\nu_r} + \sqrt{\nu_1}} \right)^l \|s^*\|_A \leq 2 \left( \frac{\sqrt{\mu_n} - \sqrt{\mu_1}}{\sqrt{\mu_n} + \sqrt{\mu_1}} \right)^l \|s^*\|_A , \tag{4.105}$$

*where $\nu_1 \leq \nu_2 \leq ... \leq \nu_r$ are the eigenvalues of the full rank matrix $\check{D}(I_r + \check{R}^{-1}\check{H}B\check{H}^T)$.*

**Proof.** If the singular value decomposition for $H$ is given by

$$H = [U_1 \;\; U_2] \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}, \tag{4.106}$$

a possible theoretical choice for $\check{H}$ could be

$$\check{H} = \Sigma_r V_1^T.$$

Denoting $\check{R} = \left(U_1^T R^{-1} U_1\right)^{-1}$, direct computations show that

$$B^{-1} + H^T R^{-1} H = B^{-1} + \check{H}^T \check{R}^{-1} \check{H}.$$

Using Assumption 4.2 and denoting $\check{D} = U_1^T D U_1$, we also obtain that $P\check{H}^T = B\check{H}^T \check{D}$.

The matrix $\check{H}$ is now a $r \times n$ matrix of rank $r$ and Lemma 4.8 can then be applied using $r$, $\check{R}$, $\check{H}$ and $\check{D}$ instead of $m$, $R$, $H$ and $D$, yielding the desired result where $\nu_1, ..., \nu_r$ the eigenvalues of $\check{D}(I_r + \check{R}^{-1}\check{H}B\check{H}^T)$, replace those of $D(I_m + R^{-1}HBH^T)$ in (4.97). We next investigate the relations between these two sets of eigenvalues.

Using the relation on $\check{H}$, $\check{R}$ and $\check{D}$, it can be shown that

$$[Z\widehat{A}]U_1 = U_1[\check{D}\check{A}]$$

where $Z = U_1 U_1^T D U_1 U_1^T$ and $\widehat{A} = I_m + R^{-1}HBH^T$. This says that $U_1$ is an invariant subspace of $Z\widehat{A}$ and every eigenvalue of $\check{D}\check{A}$ is an eigenvalue of $Z\widehat{A}$. Therefore, the nonzero eigenvalues of $Z\widehat{A}$ are equal to the eigenvalues of $\check{D}\check{A}$ using the fact that $Z\widehat{A}$ has $(m-r)$ null eigenvalues. We now consider the relations between the eigenvalues of $Z\widehat{A}$ and $D\widehat{A}$, and start by rewriting these matrices blockwise.

Using the relation (4.106), it can be shown that

$$HBH^T = U_1 \Sigma_r V_1^T B V_1 \Sigma_r U_1^T.$$

Defining

$$U_1 = \begin{bmatrix} U_r \\ 0 \end{bmatrix}, \tag{4.107}$$

$HBH^T$ can thus be rewritten in a block matrix form as

$$HBH^T = \begin{bmatrix} U_r \Sigma_r V_1^T \\ 0 \end{bmatrix} B \begin{bmatrix} V_1 \Sigma_r U_r^T & 0 \end{bmatrix} = \begin{bmatrix} M_r & 0 \\ 0 & 0 \end{bmatrix}, \tag{4.108}$$

where $M_r = U_r \Sigma_r V_1^T B V_1 \Sigma_r U_r^T$ has full rank $r$. Using the equality $PH^T = BH^T D$, we can write that

$$HPH^T = HBH^T D.$$

Hence, $HBH^T D$ is symmetric due to the symmetry of $HPH^T$. Using the relation (4.108) and defining

$$D = \begin{bmatrix} D_r & D_2 \\ D_3 & D_{m-r} \end{bmatrix},$$

where $D_r$ is $r \times r$ matrix and $D_{m-r}$ is a $(m-r) \times (m-r)$ matrix, we can write $HBH^T D$ as

$$HBH^T D = \begin{bmatrix} M_r D_r & M_r D_2 \\ 0 & 0 \end{bmatrix}. \tag{4.109}$$

From the symmetry of $HBH^T D$ given by (4.109), $M_r D_2 = 0$ which implies that $D_2 = 0$ since $M_r$ is a full rank matrix. Thus, $D$ has the form

$$D = \begin{bmatrix} D_r & 0 \\ D_3 & D_{m-r} \end{bmatrix}. \tag{4.110}$$

We next derive a block matrix form of $\widehat{A}$. Defining $R^{-1}$ as

$$R^{-1} = \begin{bmatrix} R_r^{-1} & R_2^{-1} \\ R_3^{-1} & R_{m-r}^{-1} \end{bmatrix},$$

and using (4.108) and the definition of $\widehat{A}$, we can write $\widehat{A}$ as

$$\widehat{A} = I + \begin{bmatrix} R_r^{-1} & R_2^{-1} \\ R_3^{-1} & R_{m-r}^{-1} \end{bmatrix} \begin{bmatrix} M_r & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} A_r & 0 \\ R_3^{-1} M_r & I \end{bmatrix}, \tag{4.111}$$

where $A_r = I + R_r^{-1} M_r$. From (4.107), (4.110) and (4.111), we deduce that

$$D\widehat{A} = \begin{bmatrix} D_r A_r & 0 \\ D_3 A_r + D_{m-r} R_3^{-1} M_r & D_{m-r} \end{bmatrix} \text{ and } Z\widehat{A} = \begin{bmatrix} D_r A_r & 0 \\ 0 & 0 \end{bmatrix}. \tag{4.112}$$

From (4.112), the eigenvalues of $D\widehat{A}$ are the eigenvalues of $D_r A_r$ and the eigenvalues of $D_{m-r}$. Also, the eigenvalues of $Z\widehat{A}$ are the eigenvalues of $D_r A_r$ and $(m-r)$ null eigenvalues. Therefore, the nonzero eigenvalues of $Z\widehat{A}$ which are equal to the eigenvalues of $\check{D}\check{A}$ form a subset of the eigenvalues of $D\widehat{A}$. As a result, the eigenvalues of $\check{D}(I_r + \check{R}^{-1}\check{H}B\check{H}^T)$ can be used in (4.97) instead of those $D(I_m + R^{-1}HBH^T)$, which completes the proof. □

### 4.5.3 Monitoring convergence

Inexpensive formulae for computing the quadratic cost function and the norm of the cost function gradient on each iteration from the recurrence relations of RPCG and Augmented RPCG (Algorithms 4.3 and 4.10) are derived in this section. Note that for the sake of simplicity, the initial guess is chosen as $s_0 = 0$ for all derivations.

#### Quadratic cost function

For RPCG, the quadratic cost function is calculated from Eq. (3.87) using the relations $s_i = BH^T\widehat{s}_i$ and $r_0 = H^T\widehat{r}_0$ :

$$Q[s_i] = Q[0] - \frac{1}{2}\widehat{s}_i^T HBH^T\widehat{r}_0$$

Using the symmetry of $D$ in $HBH^T$ in RPCG Algorithm 4.3 the vector $w_0$ can be calculated also from $w_0 = D^T HBH^T\widehat{r}_0$. Therefore defining $l_0 = HBH^T\widehat{r}_0$ the inexpensive formula for the quadratic cost function can be given as

$$Q[s_i] = Q[0] - \frac{1}{2}\widehat{s}_i^T l_0.$$

Using the realtion $s_i = BH^T\widehat{s}_i$, the $Q_b$ term can be written as

$$Q_b[s_i] = \frac{1}{2}\widehat{s}_i^T\widehat{f}_i + \frac{1}{2}(x_k - x_c)^T B^{-1}(x_k - x_c)$$

where $\widehat{f}_i = HBH^T\widehat{s}_i$ can be computed without the need to apply $HBH^T$ by including an additional recurrence relation in Algorithm 4.3 as follows:

$$\widehat{f}_i = \begin{cases} 0 & \text{if } i = 0 \\ \widehat{f}_{i-1} + \alpha_{i-1}t_{i-1} & \text{if } i > 0 \end{cases}$$

Taking $x_1 = x_c$ the term $B^{-1}(x_k - x_c)$ can also be calculated without the need to apply $B^{-1}$ from Eq. (4.19). The $Q_o$ term can then be computed from Eq. (3.89).

For Augmented RPCG, the formula can be obtained similarly following the same derivation by using augmented matrices and vectors. While replacing the term $B^{-1}(x_k - x_c)$, since $H_k$ varies along the outer loop, the following equation has to be used instead

of the Eq. (4.19)

$$B^{-1}(x_k - x_c) = \sum_{j=0}^{k-1} H_j^T \lambda_j.$$

### Norm of the quadratic cost function gradient

For RPCG, the norm is obtained from

$$\|\nabla Q[s_i]\|_P = \|H^T \widehat{r}_i\|_P = \|\widehat{r}_i\|_{HBH^T D} = \sqrt{\widehat{r}_i^T w_i}$$

where the vectors $\widehat{r}_i$ and $w_i$ are defined in Algorithm 4.3. Similarly for Augmented RPCG the norm is obtained from

$$\|\nabla Q[s_i]\|_P = \sqrt{\underline{r}_i^T \underline{w}_i}$$

where the vectors $\underline{r}_i$ and $\underline{w}_i$ are defined in Algorithm 4.9.

# CHAPTER 5

## Solving the subproblem with a Lanczos method

This chapter considers applying a *preconditioned Lanczos method* to a sequence of symmetric and positive-definite linear systems of the form of $A_k s = b_k$, where $A_k$ and $b_k$ are defined in (2.16), and where preconditioning is achieved by using the Limited Memory Preconditioners (LMPs) described in Chapter 3.

Like Chapter 3, we first focus on warm-start preconditioning of a sequence of linear systems with multiple right-hand sides,

$$As = b_k$$

by using directions generated by the Lanczos algorithm applied on the previous linear system. Note that, each linear system is part of a sequence, which is supposed to produce a convergent sequence of solutions. We discuss three different ways of preconditioning the linear system $As = b_k$ with a warm-start preconditioner. These approaches can be considered as the Lanczos version of the variants of the CG algorithm described in Section 3.1.

We later explain how to extract the information from Lanczos-type algorithms to construct particular LMPs such as the quasi-Newton LMP, Ritz LMP and spectral LMP. The adaptation of these preconditioners to the case of solving a sequence of slowly varying linear systems in a Gauss-Newton process follows the same results as explained in Section 3.1.4.

This chapter then introduces a Lanczos-type method in dual space. This dual method produces, in exact arithmetic, the same iterates as those produced by a standard Lanczos method applied to the linear system (2.16) and RPCG. Like RPCG, this algorithm might yield gains in terms of both memory usage and computational cost compared to primal algorithms when $m \ll n$.

We then explain how to use practical warm-start LMPs described in Chapter 4 to accelerate the convergence of the dual Lanczos algorithm when solving a sequence of linear systems with multiple right hand sides. The adaptation of these preconditioners when used for a sequence of slowly varying linear systems is similar to what is explained

in Section 4.4.2.

We conclude the chapter by providing inexpensive formulae to compute the quadratic cost function (2.13) and the norm of its gradient, which are useful quantities to monitor convergence.

## 5.1 Preconditioning the Lanczos method with LMPs

In this section we are dealing with applying the Lanczos method to the systems of equations (3.1) with multiple right hand sides of the form of

$$As = b_k,$$

with a *warm-start LMP* technique. Each linear system is part of a sequence and is not isolated. It is assumed that this sequence produces a convergent sequence of solutions.

We explain different approaches on preconditioning the linear system (3.1) with a symmetric and positive definite LMP defined by (2.38). This chapter follows the same structure as Section 3.1 and the algorithms introduced in this section are the Lanczos version of the CG algorithms presented in Section 3.1.

We start with the approach that uses a split preconditioning with the Lanczos method (Algorithm 2.3) as explained below.

### Preconditioning with Lanczos Algorithm 2.3

This approach can be considered as the Lanczos version of Approach (A) presented in Section 3.1. It considers solving the linear system (3.1) with a LMP $P$ formulated by (2.38) by using Lanczos Algorithm 2.3 on the preconditioned linear system

$$F^T A F\, \widetilde{s} = F^T b_k, \tag{5.1}$$

where $F$ is the factored form the LMP $P$, i.e. $P = FF^T$, and formulated by equation (3.2). The approximate solution $s$ is obtained from the relation $s = F\,\widetilde{s}$. We name this approach *Approach (E)*.

The same strategy as with solving a convergent sequence of linear systems (3.1) by Approach (A) can be used to solve these linear systems with Approach (E). Therefore, an algorithm similar to Algorithm 3.1 can be used to solve a sequence of linear systems (3.1) with Lanczos Algorithm 2.3, by performing the inner minimization of Algorithm 3.1 with the Lanczos algorithm instead of the CG algorithm.

The characteristics of the Lanczos version of Algorithm 3.1 summarized in Section 3.1 (for instance, the need for the factored form of the LMP, matrix-vector products in particular case $P_0 = B$) remain the same as those of Algorithm 3.5. For further details we refer to Section 3.1.

We now present an alternative algorithm for preconditioning a sequence of linear systems (3.1) with Lanczos Algorithm 2.6.

### Preconditioning with PLanczos Algorithm 2.6

This approach can be considered as the Lanczos version of Approach (B) presented in Section 3.1. Similar to the case of the CG solvers, the main difference in solving the linear system (3.1) with PLanczos Algorithm 2.6 rather than Lanczos Algorithm 2.3 with a preconditioner is that *Lanczos Algorithm 2.3 does not require the factored form of the preconditioner*. This can be algorithmically significant when the preconditioner factorization is not available or expensive to compute.

Therefore, another preconditioning technique results from applying Lanczos Algorithm 2.6 to the linear system $As = b_k$ preconditioned with the LMP (2.38). We name this approach *Approach (F)*.

The same strategy for solving a convergent sequence of linear systems (3.1) with Approach (B) can be used to solve these linear systems by Approach (F). This yields Algorithm 3.2 in which the inner minimization is performed by Approach (F).

The characteristics of the Lanczos version of Algorithm 3.2 summarized in Section 3.1 (for instance, matrix-vector products in particular case $P_0 = B$) remain the same as those of Algorithm 3.5. For further details we refer to Section 3.1.

We are left with explaining the third preconditioning approach which avoids matrix-vector products with $B^{-1}$ as well as the factorization of the preconditioner by using Algorithm PCGIF 3.4. We next introduce the Lanczos version of Approach (C) presented in Section 3.1.

### Preconditioning with PLanczos Inverse Free Algorithm

Remember that the idea in this approach is to apply the symmetric and positive definite matrix $P$ formulated by (2.38) as a *right symmetric preconditioner* while solving the linear system (3.1). Under the assumption that *the matrix $B^{-1}P$ is available*, this strategy yields solving the linear system (3.14), which is given by

$$(I_n + H^T R^{-1} H B)\, \underline{s} = B^{-1}(x_c - x_k) - H^T R^{-1} d_k,$$

equipped with the $B$-inner product for the variable $\underline{s}$ by using the LMP matrix $C$ formulated by (3.19). The approximate solution is recovered from $s = B\underline{s}$. This results in the PLanczos Inverse Free (PLanczosIF) algorithm (Algorithm 5.1) that avoids matrix-vector multiplication with $B^{-1}$. We name this approach *Approach (G)*.

Algorithm 5.1 can be transformed into a cheaper form by introducing an additional vector as $z_i = B\underline{z}_i$, reducing its cost per loop to a single matrix-vector product with $B$. Substituting this definition into Algorithm 5.1, yields the final version of PLanczosIF Algorithm (Algorithm 5.2).

Therefore, as a third alternative to preconditioning, the linear system (3.14) can be solved by Approach (G). This approach can be applied on a convergent sequence of linear systems (3.1) by using Algorithm 3.5.

The characteristics of the Lanczos version of Algorithm 3.5 summarized in Section 3.1 remain the same as those of Algorithm 3.5. So, this algorithm requires neither

**Algorithm 5.1: PLanczosIF (version 1)**

$\underline{r}_0 = b - AB\underline{s}_0$
$\underline{z}_0 = C\underline{r}_0$
$\beta_0 = \sqrt{\underline{r}_0^T B \underline{z}_0}$
$\beta_1 = 0$
$\underline{v}_0 = 0$
$\underline{v}_1 = \underline{r}_0/\beta_0$
$\underline{z}_1 = \underline{z}_0/\beta_0$
$\underline{V}_1 = \underline{v}_1$
**for** $i = 1, \ldots, l$ **do**
    $\underline{w}_i = (\underline{z}_i + H^T R^{-1} H B \underline{z}_i) - \beta_i \underline{v}_{i-1}$
    $\alpha_i = \underline{w}_i^T B \underline{z}_i$
    $\underline{w}_i := \underline{w}_i - \alpha_i \underline{v}_i$
    $\underline{z}_i := C\underline{w}_i$
    $\beta_{i+1} = \sqrt{\underline{w}_i^T B \underline{z}_i}$
    $\underline{v}_{i+1} = \underline{w}_i/\beta_{i+1}$
    $\underline{z}_{i+1} := \underline{z}_i/\beta_{i+1}$
    $\underline{V}_i := [\underline{V}_i, \underline{v}_{i+1}]$
    $(T_i)_{i,i} = \alpha_i$
    **if** $i > 1$ **then**
        $(T_i)_{i-1,i} = (T_i)_{i,i-1} = \beta_i$
    **end**
    Check convergence and stop if desired accuracy is reached
**end**
$y_l = T_l^{-1}(\beta_0 e_1)$
$\underline{s}_l = \underline{s}_0 + C\underline{V}_l y_l$

**Algorithm 5.2: PLanczosIF**

$\underline{r}_0 = b - AB\underline{s}_0$
$\underline{z}_0 = C\underline{r}_0$
$z_0 = B\underline{z}_0$
$\beta_0 = \sqrt{\underline{r}_0^T z_0}$
$\beta_1 = 0$
$\underline{v}_0 = 0$
$\underline{v}_1 = \underline{r}_0/\beta_0$
$\underline{z}_1 = \underline{z}_0/\beta_0$
$z_1 = z_0/\beta_0$
$Z_1 = z_1$;
**for** $i = 1, \ldots, l$ **do**
    $\underline{w}_i = (\underline{z}_i + H^T R^{-1} H z_i) - \beta_i \underline{v}_{i-1}$
    $\alpha_i = \underline{w}_i^T z_i$
    $\underline{w}_i := \underline{w}_i - \alpha_i \underline{v}_i$
    $\underline{z}_i := C\underline{w}_i$
    $z_i = B\underline{z}_i$
    $\beta_{i+1} = \sqrt{\underline{w}_i^T z_i}$
    Check convergence and stop if desired accuracy is reached
    $\underline{v}_{i+1} = \underline{w}_i/\beta_{i+1}$
    $\underline{z}_{i+1} := \underline{z}_i/\beta_{i+1}$
    $z_{i+1} := z_i/\beta_{i+1}$
    $Z_i := [Z_i, z_{i+1}]$;
    $(T_i)_{i,i} = \alpha_i$
    **if** $i > 1$ **then**
        $(T_i)_{i-1,i} = (T_i)_{i,i-1} = \beta_i$
    **end**
**end**
$y_l = T_l^{-1}(\beta_0 e_1)$
$s_l = s_0 + Z_l y_l$

the factorization of the preconditioner nor matrix-vector products with $B^{-1}$ under the assumption that $C_0 = B^{-1}P_0$ is available.

### Summary

We have presented different Lanczos algorithms for the solution of the linear system (3.1) with the LMP given by (2.38) in the context of solving a convergent sequence of linear systems with multiple right-hand sides. These algorithms differ in the assumption they made on the preconditioner form and matrix-vector products. They are listed below:

- Approach (E): Apply Lanczos Algorithm 2.3 to the linear system (3.7) of the form of $\widetilde{A}\widetilde{s} = \widetilde{b}_k$ with a preconditioner constructed from (3.2),
- Approach (F): Apply Lanczos Algorithm 2.6 to the linear system (3.1) of the form of $As = b_k$ with a preconditioner constructed from (2.38),
- Approach (G): Apply PLanczosIF Algorithm 5.2 to the linear system (3.14) of the form of $\underline{A}\,\underline{s} = b_k$ with a preconditioner constructed from (3.19).

As in the variants of the conjugate gradients method, the iterates of Lanczos Algorithm 2.3 are related to the iterates of PLanczos Algorithm 2.6 with (3.33). Under the assumption that $P = BC$, the iterates of PLanczosIF Algorithm 3.4 are related to the iterates of PLanczos Algorithm 2.5 with (3.36). These three variants of the Lanczos algorithms search for an approximate solution $s_\ell$ in a subspace $s_0 + \mathcal{K}(Pr_0, PA)$ with

$$\mathcal{K}(Pr_0, PA) = \text{span}\left\{Pr_0, (PA)Pr_0, \ldots, (PA)^{l-1}Pr_0\right\} \tag{5.2}$$

where $r_0 = As_0 - b$ is the initial residual and $\ell$ is the number of inner iterations.

Note that these approaches are mathematically equivalent to the approaches (A), (B) and (C) described in Section 3.1. The main properties of these approaches in terms of preconditioning are listed in Section 3.1.

These three approaches can be used within Algorithms 3.1, 3.2 and 3.5 for solving a sequence of linear systems with multiple right hand sides. In this case, the characteristics of these algorithms are listed in Table 5.1.

| Algorithm | inner min. | LMP formula | Assumption |
|---|---|---|---|
| 3.1 | App. (E) | $F_{k+1}$ given by (3.2) with $M_k = I_n$ | $F_0$ is available |
| 3.2 | App. (F) | $P_{k+1}$ given by (2.38) with $M_k = P_k$ | |
| 3.5 | App. (G) | $C_{k+1}$ given by (3.19) with $\underline{M}_k = C_k$ and $S_k = B\underline{S}_k$ | $B^{-1}P_0$ is available |

Table 5.1: A summary of the characteristics of Algorithms 3.1, 3.2 and 3.5 when used with Lanczos-type algorithms

Considering the particular case that the initial preconditioner is chosen as $P_0 = B$ when solving a sequence of linear systems, the required matrix-vector products for each

approach, is the same as with Approaches (A), (B) and (C) given in Table 3.2.

*As a result, Algorithms 3.1, 3.2 and 3.5 when used with Lanczos type methods whose characteristics are listed in Table 5.1 generate a mathematically equivalent sequence of iterates $\{s_k\}$. The choice of the algorithm depends on the availability of the required form of the preconditioner, operators and their matrix-vector product cost.*

We next explain how to construct particular LMPs: the quasi-Newton LMP, the Ritz LMP and the spectral LMP described in Chapter 3 in the context of the given Lanczos-type algorithms. This part will include for each preconditioner an extraction methodology of the required information from the Lanczos algorithm.

### 5.1.1 Preconditioning with the quasi-Newton LMP

We consider in this section the use of the quasi-Newton LMPs described in Section 3.1.1 when solving a convergent sequence of linear systems 3.1 with a Lanczos-type method. We start with explaining how to construct a quasi-Newton LMP within Algorithm 3.2 in which the linear system in sequence is solved by Approach (E).

#### Approach (E) with the quasi-Newton LMP

Let us first consider to apply Lanczos Algorithm 2.3 on the linear system (3.7) of the form of $\widetilde{A}_k\,\widetilde{s} = \widetilde{b}_k$ within Algorithm 3.1. Remember that while applying the quasi-Newton LMP $Y_{j+1}$ formulated by (3.41) during Approach (A) the square-root factorization of $Y_{j+1}$ is required. This factored form $Y_{j+1} = F_{j+1}F_{j+1}^T$ is calculated from the formula (3.42) given as

$$\boxed{F_{j+1} = \left[I_n - \widetilde{p}_j \left(\tau_j \widetilde{q}_j^T + \tau_j \frac{\widetilde{r}_j^T}{\|\widetilde{r}_j\|_2}\right)\right] F_j}$$

where $F_0 = I_n$, $\widetilde{p}_i$, $i = 0, \ldots, j$ are the search directions, $\tau_j = 1/(\widetilde{q}_j^T \widetilde{p}_j)$ and $\widetilde{q}_j = \widetilde{A}\widetilde{p}_j$ obtained during the CG run when solving the $(k-1)$-th linear system.

The pairs required in the formula (3.42) are not by-products of Lanczos Algorithm 2.3. Can we construct the pairs required in this formula from the recurrence relations of Lanczos Algorithm 2.3? The answer is provided by Algorithm 2.4 in which the iterates of CG Algorithm 2.2 are obtained from those of Lanczos Algorithm 2.3. Therefore, *when using Lanczos Algorithm 2.3 with the quasi-Newton LMP (3.42) during Algorithm 3.1, before starting the next inner loop, Algorithm 2.4 has to be run to obtain the required pairs for constructing the quasi-Newton LMP.*

We now explain how to construct a quasi-Newton LMP within Algorithm 3.2 in which the linear system in sequence is solved by Approach (F).

#### Approach (F) with the quasi-Newton LMP

Let us now consider solving the system of linear equations (3.1) of the form of $As = b_k$ with PLanczos Algorithm 2.6 within Algorithm 3.2. In this case the iterates of PCG

Algorithm 2.5 can be obtained from the iterates of PLanczos Algorithm 2.6 by using Algorithm 2.7. This allows to construct the quasi-Newton LMP for the matrix $A$ in the $k$-th linear system from formula (3.37) given as

$$\boxed{P_{j+1} = (I_n - \tau_j p_j q_j^T) P_j (I_n - \tau_j q_j p_j^T) + \tau_j p_j p_j^T},$$

where $p_i$, $i = 1, \ldots, j$, $q_j = Ap_j$ and $\tau_j = 1/(q_j^T p_j)$. These pairs are extracted from the iterates of PLanczos Algorithm obtained when solving the $(k-1)$-th linear system by using Algorithm 2.7. Note that, the initial preconditioning matrix (which can be interpreted as the first-level preconditioner) for the formula (3.37) is chosen as the preconditioner used for the previous linear system.

We are left with the task of deriving the quasi-Newton LMP for Approach (G) that is used within Algorithm 3.5.

### Approach (G) with the quasi-Newton LMP

Let us consider solving the system of linear equations (3.14) of the form of $\underline{A}\underline{s} = b_k$ with PLanczosIF Algorithm 5.2 within Algorithm 3.5. Remember that the quasi-Newton LMP for the system matrix $\underline{A}$ is constructed from the formula (3.43) given as

$$\boxed{C_{j+1} = \left(I_n - \tau_j \underline{p}_j \underline{q}_j^T B\right) C_j \left(I_n - \tau_j \underline{q}_j p_j^T\right) + \tau_j \underline{p}_j p_j^T}$$

where $\underline{p}_i$, $i = 1, \ldots, j$, $p_i = B\underline{p}_i$, $\underline{q}_i = \underline{A}\underline{p}_i$ and $\tau_i = 1/(\underline{q}_i^T p_i)$ are obtained when solving the $(k-1)$-th linear system with PCGIF Algorithm 3.6. We remind that the initial matrix in (3.43) is chosen as the preconditioner used for the previous linear system.

We are left with the task of obtaining the iterates of PCGIF algorithm from those of PLanczosIF. Using the relations given in Section 2.6.3 and adapting them to Algorithms PCGIF and PLanczosIF results in Algorithm 5.3. This algorithm extracts the iterates of PCGIF algorithm from the iterates of PLanczosIF Algorithm.

*As a result it is possible to find the corresponding quasi-Newton LMPs for variants of Lanczos algorithms (Approaches (E), (F) and (G)) such that when they are used within the corresponding algorithms (Algorithms 3.1, 3.2 and 3.5), all these algorithms generate mathematically equivalent iterates $\{s_k\}$. We showed that such preconditioners can be constructed from vectors which can be extracted from by-products of the corresponding algorithms (Algorithms 2.3, 2.6 and 5.2).*

We summarize the characteristics of Algorithms 3.1, 3.2 and 3.5 when used with the Lanczos method in which the preconditioner is achieved by the quasi-Newton LMP in Table 5.2. We assume that, for Algorithm 3.1 $F_0$ is available and that, for Algorithm 3.5 $B^{-1}P_0$ is available.

The cost for applying the preconditioner and the memory requirements of the algorithms of the present section are summarized in Table 3.4. To obtain the quasi-Newton preconditioner, Lanczos type methods require additional $O(n)$ operations when compared with conjugate-gradients type methods. Note that, the algorithms listed in Table 5.2 are mathematically equivalent to the corresponding algorithms listed in

**Algorithm 5.3: PLanczosIF2PCGIF**

Given $B, C, \underline{r}_0, s_0, T_l, \beta_{l+1}, \underline{V}_l, Z_l, l$ from Algorithm 5.2
$(T_l)_{l+1,l} = \beta_{l+1}$
$l_0 = B\underline{r}_0$
$\underline{z}_0 = C\underline{r}_0$
$\underline{p}_0 = \underline{z}_0$
$z_0 = C^T l$
$\beta_0 = \sqrt{r_0^T z_0}$
$p_0 = z_0$
$\rho = \beta_0^2$
**for** $i = 1, \ldots, l$ **do**
    $y_i = T_i^{-1}(\beta_0 e_1)$
    $\underline{r}_i = -(T_l)_{i+1,i}(V_l)_{:,i+1} e_i^T y_i$
    $l_i = B\underline{r}_i$
    $\underline{z}_i = C\underline{r}_i$
    $z_i = C^T l_i$
    $\rho_i = \underline{r}_i^T z_i$
    $\beta_i = \rho_i / \rho_{i-1}$
    $\underline{p}_i = \underline{z}_i + \beta_i \underline{p}_{i-1}$
    $p_i = z_i + \beta_i p_{i-1}$
    $s_i = s_0 + Z_i y_i$
    $\alpha_i = \|(s_i - s_{i-1})\|_2 / \|p_{i-1}\|_2$
    $\underline{q}_i = -(\underline{r}_i - \underline{r}_{i-1})/\alpha_i$
    $t_i = -(l_i - l_{i-1})/\alpha_i$
**end**

| Algorithm | inner min. | pairs computation | quasi-Newton LMP |
|---|---|---|---|
| 3.1 | Approach (E) | Algorithm 2.4 | $F_{j+1}$ given by (3.42) |
| 3.2 | Approach (F) | Algorithm 2.7 | $P_{j+1}$ given by (3.37) |
| 3.5 | Approach (G) | Algorithm 5.3 | $C_{j+1}$ given by (3.43) |

Table 5.2: A summary of the characteristics of Algorithms 3.1, 3.2 and 3.5 where the inner minimization is performed with corresponding Lanczos algorithms and preconditioning is achieved by using the quasi-Newton LMPs. Approaches (E), (F) and (G) are defined in Section 5.1.

Table 3.3.

### 5.1.2 Preconditioning with the Ritz LMP

In this section we consider preconditioning the linear system (3.1) by using Ritz pairs computed during Lanczos type methods (Approaches (E), (F) and (G)). Ritz LMP formulas for each conjugate gradient type method (Approaches (A), (B) and (C)) are already provided in Section 3.1.2. The pairs needed to construct these Ritz LMPs are directly available from a Lanczos type method therefore there is no need to generate these pairs as done in conjugate gradients. We next explain shortly how to construct the Ritz LMPs for the Lanczos type methods.

#### Approach (E) with the Ritz LMP

While using Approach (E) for the $k$-th linear system in sequence within Algorithm 3.1, the Ritz LMP formulated by (3.49) can be constructed from Algorithm 3.9. Note that, the required inputs to Algorithm 3.9 (the matrices $\widetilde{V}_\ell$ and $T_\ell$, the vector $\widetilde{v}_{\ell+1}$, and the scalar $\beta_{\ell+1}$) are directly available from Approach (E).

#### Approach (F) with the Ritz LMP

When applying Approach (F) on the $k$-th linear system in sequence within Algorithm 3.2, the Ritz LMP (3.65) can be constructed from Algorithm 3.11. The required inputs (the matrices $P_{k-2}$, $T_\ell$, $G_\ell$, the vector $g_{\ell+1}$ and the scalar $\beta_{\ell+1}$) to Algorithm 3.11 are directly available from Approach (F). The column vectors $g_i$, $i = 1, \ldots, \ell$ of matrix $G_\ell$ and the vector $g_{\ell+1}$ are the normalized vectors $z_i$, $i = 1, \ldots, \ell+1$ that is generated during PLanczos Algorithm 2.6. These vectors then need to be stored to construct the Ritz-LMP by using Algorithm 3.11.

#### Approach (G) with the Ritz LMP

When applying Approach (G) on the $k$-th linear system in sequence within Algorithm 3.5, the Ritz LMP (3.73) can be constructed from Algorithm 3.12. The required inputs (the matrices $C_{k-2}$, $T_\ell$, $G_\ell$, $\underline{V}_\ell$, the vectors $g_{\ell+1}$ and $\underline{v}_{\ell+1}$, and the scalar

$\beta_{\ell+1}$) to Algorithm 3.11 are directly available from Approach (G). The column vectors $g_i$, $i = 1, \ldots, \ell$ of matrix $G_\ell$ and the vector $g_{\ell+1}$ are the normalized vectors $z_i$, $i = 1, \ldots, \ell+1$ that is generated during PLanczosIF Algorithm 5.2, i.e. $G_\ell = Z_\ell$ and $g_{\ell+1} = z_{\ell+1}$. Note that, the column vectors $\underline{v}_i$, $i = 1, \ldots, \ell$ of matrix $\underline{V}_\ell$ and the vector $\underline{v}_{\ell+1}$ in Algorithm 3.11 are the normalized vectors $\underline{z}_i$, $i = 1, \ldots, \ell+1$ that is generated during PLanczosIF Algorithm 5.2, i.e. $\underline{V}_\ell = \underline{Z}_\ell$ and $\underline{v}_{\ell+1} = \underline{z}_{\ell+1}$.

*As a result it is possible to find the corresponding Ritz LMPs for variants of Lanczos algorithms (Approaches (E), (F) and (G)) such that when they are used within the corresponding algorithms (Algorithms 3.1, 3.2 and 3.5), all these algorithms generate mathematically equivalent iterates $\{s_k\}$. We showed that such preconditioners can be constructed from vectors which are by-products of the corresponding algorithms (Algorithms 2.3, 2.6 and 5.2).*

We summarize the characteristics of Algorithms 3.1, 3.2 and 3.5 when used with the Lanczos method in which the preconditioner is achieved by the Ritz LMP in Table 5.3. We assume that, for Algorithm 3.1 $F_0$ is available and that, for Algorithm 3.5 $B^{-1}P_0$ is available.

| Algorithm | inner min. | Ritz-LMP calculation | Ritz-LMP formula |
|---|---|---|---|
| 3.1 | Approach (E) | Algorithm 3.9 | $F_k$ given by (3.49) |
| 3.2 | Approach (F) | Algorithm 3.11 | $P_k$ given by (3.65) |
| 3.5 | Approach (G) | Algorithm 3.12 | $C_k$ given by (3.73) |

Table 5.3: A summary of the characteristics of Algorithms 3.1, 3.2 and 3.5 where the inner minimization is performed with corresponding Lanczos algorithms and preconditioning is achieved by using the Ritz LMPs. Approaches (E), (F) and (G) are defined in Section 5.1.

The cost for applying the preconditioner and memory requirements is summarized in Table 3.6. The algorithms listed in Table 5.3 are mathematically equivalent to the corresponding algorithms listed in Table 3.5.

### 5.1.3 Preconditioning with the spectral LMP

The spectral LMPs for each conjugate gradient algorithm are given in Section 3.1.3. The same formulas can be applied to the Lanczos type methods. We refer to Section 3.1.3 for details in deriving these formulas. In this section we listed the corresponding spectral LMP formula for each Lanczos type method in Table 5.4.

### 5.1.4 Preconditioning varying systems

The same results explained in Section 3.1.4 apply for Lanczos-type methods when solving a sequence of varying linear systems (3.1) preconditioned by an LMP.

| Algorithm | inner min. | spectral LMP formula | Assumption |
|---|---|---|---|
| 3.1 | Approach (E) | $F_k$ given by (3.76) | $F_0$ is available |
| 3.2 | Approach (F) | $P_k$ given by (3.78) | |
| 3.5 | Approach (G) | $C_k$ given by (3.80) | $B^{-1}P_0$ is available |

Table 5.4: A summary of the characteristics of Algorithms 3.1, 3.2 and 3.5 where the inner minimization is performed with corresponding Lanczos algorithms and preconditioning is achieved by using the spectral LMPs. Approaches (E), (F) and (G) are defined in Section 5.1.

## 5.2 Solving the subproblem with Restricted Preconditioned Lanczos (RPLanczos)

We mentioned in Section 4.1 that the solution of the linearized subproblem (2.46) can be alternatively found by using the dual approach. We focused on the dual approach called RPCG (Gratton and Tshimanga, 2009) which solves the linear system (4.14),

$$(R^{-1}HBH^T + I_m)\,\lambda = R^{-1}(H(x - x_c) - d)$$

with the (possibly semi-)definite $HBH^T$-(semi) inner product in which the unsymmetric system matrix $\widehat{A} = R^{-1}HBH^T + I_m$ becomes symmetric. Note that we have dropped the outer loop index $k$ for the sake of simplicity. As discussed by Gratton and Tshimanga (2009), RPCG generates, in exact arithmetic, the same iterates as those generated by the primal approach (PCG on the linear system (2.16)).

*In the same way that RPCG is the dual equivalent of PCG, there exists a Lanczos algorithm that is the dual equivalent of PLanczos.* We call this algorithm *Restricted PLanczos* (*RPLanczos*). *It produces identical iterates, in exact arithmetic, to RPCG, PCG and PLanczos.* RPLanczos simply solves the linear system (4.14) using a Lanczos algorithm equipped with the (possibly semi-definite) $HBH^T$-inner product. The solution is then recovered from Eq. (4.3). RPLanczos is a variant of the Restricted FOM (RSFOM) algorithm (Gratton, Toint and Tshimanga, 2009) adapted to symmetric and positive-definite matrices. The derivation of RPLanczos is as follows.

We refer the primal approach within this section as applying PLanczos Algorithm 2.6 on the linear system (2.16). Assume for now that the initial guess for primal approach is taken as $s_0 = x_c - x$. Taking this initial guess to satisfy Assumption 4.1,

$$r_0 = H^T\widehat{r}_0 \tag{5.3}$$

where $r_0 \in \mathbb{R}^n$ is the initial residual for primal approach and $\widehat{r}_0 \in \mathbb{R}^m$ is the initial residual for RPLanczos. We assume also that RPLanczos uses the preconditioner $D \in \mathbb{R}^{m\times m}$ such that Assumption 4.2,

$$PH^T = BH^TD \tag{5.4}$$

holds where $P$ is the preconditioner for the primal approach.

Let us define the vectors in $\mathbb{R}^m$ such that

$$\widehat{z}_0 = D\widehat{r}_0, \tag{5.5}$$
$$\widehat{v}_0 = 0,$$
$$\widehat{z}_1 = \widehat{z}_0/\beta_0, \tag{5.6}$$
$$\widehat{v}_1 = \widehat{r}_0/\beta_0. \tag{5.7}$$

From the definitions (5.3) - (5.7), it is possible to derive the following relations from PLanczos:

$$z_0 = BH^T\widehat{z}_0, \tag{5.8}$$
$$z_1 = BH^T\widehat{z}_1, \tag{5.9}$$
$$v_1 = H^T\widehat{v}_1, \tag{5.10}$$
$$\begin{aligned} w_1 &= (B^{-1} + H^T R^{-1} H) z_1 - \alpha_1 v_1, \\ &= (H^T\widehat{z}_1 + H^T R^{-1} H B H^T \widehat{z}_1) - H^T \alpha_1 \widehat{v}_1, \\ &= H^T \left[ (I_m + R^{-1} H B H^T)\widehat{z}_1 - \alpha_1 \widehat{v}_1 \right]. \end{aligned} \tag{5.11}$$

Equation (5.11) can be written as

$$w_1 = H^T\widehat{w}_1, \tag{5.12}$$

where $\widehat{w}_1 = (I_m + R^{-1}HBH^T)\widehat{z}_1 - \alpha_1\widehat{v}_1$.

We now prove by induction that the relations given by (5.9), (5.10) and (5.12), also hold for $i > 1$. First, assume that the relations are satisfied for a given $i$:

$$z_i = BH^T\widehat{z}_i, \tag{5.13}$$
$$v_i = H^T\widehat{v}_i, \tag{5.14}$$
$$w_i = H^T\widehat{w}_i. \tag{5.15}$$

We show that the relations hold for $i+1$. From lines 15 and 16 of Algorithm 2.6 for PLanczos, we have, using Eqs (5.15) and (5.4),

$$v_{i+1} = w_i/\beta_{i+1} = H^T\widehat{w}_i/\beta_{i+1}, \tag{5.16}$$
$$z_{i+1} = Pw_i/\beta_{i+1} = PH^T\widehat{w}_i/\beta_{i+1} = BH^T D\widehat{w}_i/\beta_{i+1}. \tag{5.17}$$

Defining

$$\widehat{v}_{i+1} = \widehat{w}_i/\beta_{i+1},$$
$$\widehat{z}_{i+1} = D\widehat{w}_i/\beta_{i+1},$$

we obtain from (5.16) and (5.17) that

$$v_{i+1} = H^T \widehat{v}_{i+1}, \tag{5.18}$$

$$z_{i+1} = BH^T \widehat{z}_{i+1}. \tag{5.19}$$

Similarly, from lines 10 and 12 of Algorithm 2.6 and using the relations (5.14), (5.18) and (5.19), we have

$$\begin{aligned} w_{i+1} &= (B^{-1} z_{i+1} + H^T R^{-1} H z_{i+1}) - \beta_i v_i - \alpha_{i+1} v_{i+1}, \\ &= (H^T \widehat{z}_{i+1} + H^T R^{-1} H B H^T \widehat{z}_{i+1}) - \beta_{i+1} H^T \widehat{v}_i - \alpha_{i+1} H^T \widehat{v}_{i+1}, \\ &= H^T \left[ (I_m + R^{-1} H B H^T) \widehat{z}_{i+1} - \beta_{i+1} \widehat{v}_i - \alpha_{i+1} \widehat{v}_{i+1} \right], \\ &= H^T \widehat{w}_{i+1}, \end{aligned} \tag{5.20}$$

where

$$\widehat{w}_{i+1} = (I_m + R^{-1} H B H^T) \widehat{z}_{i+1} - \beta_{i+1} \widehat{v}_i - \alpha_{i+1} \widehat{v}_{i+1}, \tag{5.21}$$

which completes the proof.

Equations (5.13)-(5.15) allow all the recurrence relations in PLanczos Algorithm 2.6 involving vectors of dimension $n$ to be transformed directly into corresponding recurrence relations involving vectors of dimension $m$. This yields the preliminary version of RPLanczos algorithm given by Algorithm 5.4.

The cost of Algorithm 5.4 per loop can be reduced by introducing a new vector $t$ defined by

$$t = HBH^T \widehat{z}, \tag{5.22}$$

which yields the final version of the RPLanczos (Algorithm 5.5). As with PLanczos, one loop of RPLanczos requires a single matrix-vector product with each of the operators $H$, $H^T$, $R^{-1}$ and $B$, but with the important difference that all vectors are defined in $\mathbb{R}^m$ instead of $\mathbb{R}^n$. The two algorithms are mathematically equivalent.

As discussed in Section 5.1, PLanczos searches for the solution $s_\ell$ in the subspace $s_0 + \mathcal{K}^\ell(PA, Pr_0)$ where the Krylov subspace $\mathcal{K}^\ell(PA, Pr_0)$ is given by (5.2). Replacing $r_0$ in (5.2) with Eq. (5.3) yields

$$\mathcal{K}^\ell(PA, PH^T \widehat{r}_0) = \text{ span}\left\{ PH^T \widehat{r}_0, \ldots, (PA)^{l-1} PH^T \widehat{r}_0 \right\}. \tag{5.23}$$

Using the relations $ABH^T = H^T \widehat{A}$, (where $\widehat{A} = I_m + R^{-1} H B H^T$, $A = B^{-1} + H^T R^{-1} H$) and $PH^T = BH^T D$ in (5.23) we obtain that

$$\begin{aligned} \mathcal{K}^\ell(PA, PH^T \widehat{r}_0) &= \text{span}\left\{ BH^T D \widehat{r}_0, \ldots, BH^T D \widehat{A}^{l-1} D \widehat{r}_0 \right\} \\ &= BH^T \mathcal{K}^\ell(D\widehat{A}, D\widehat{r}_0) \end{aligned}$$

**Algorithm 5.4: Modified PLanczos Algorithm**

$\widehat{r}_0 = R^{-1}(Hs_0 - d)$
$\widehat{z}_0 = D\widehat{r}_0$
$\beta_0 = \sqrt{\widehat{z}_0^T HBH^T \widehat{r}_0}$
$\widehat{v}_1 = \widehat{r}_0/\beta_0$
$\widehat{z}_1 = \widehat{z}_0/\beta_0$
$\beta_1 = 0$
$\widehat{v}_0 = 0$
$\widehat{V} = \widehat{v}_1$
**for** $i = 1, 2, ..., l$ **do**
    $\widehat{q}_i = (I_m + R^{-1}HBH^T)\widehat{z}_i - \beta_i \widehat{v}_{i-1}$
    $\alpha_i = \widehat{w}_i^T HBH^T \widehat{z}_i$
    $\widehat{w}_i = \widehat{q}_i - \alpha_i \widehat{v}_i$
    $\widehat{z}_{i+1} = D\widehat{w}_i$
    $\beta_{i+1} = \sqrt{\widehat{z}_{i+1}^T HBH^T \widehat{w}_i}$
    $\widehat{v}_{i+1} = \widehat{w}_i/\beta_{i+1}$
    $\widehat{z}_{i+1} = \widehat{z}_{i+1}/\beta_{i+1}$
    $\widehat{V} = [\widehat{V}, \widehat{v}_{i+1}]$
    $(T_i)_{i,i} = \alpha_i$
    **if** $i > 1$ **then**
        $(T_i)_{i-1,i} = (T_i)_{i,i-1} = \beta_i$
    **end**
    Check convergence and stop if desired accuracy is reached
**end**
$y_\ell = T_\ell^{-1}\beta_0 e_1$
$s_\ell = s_0 + BH^T D\widehat{V}_\ell y_\ell$

where

$$\mathcal{K}^\ell(D\widehat{A}, D\widehat{r}_0) = \text{span}\Big\{D\widehat{r}_0, (D\widehat{A})D\widehat{r}_0, \ldots, (D\widehat{A})^{l-1}D\widehat{r}_0\Big\}.$$

From these relations it can be easily seen that RPLanczos searches for the solution $s_l$ from the subspace $s_0 + \mathcal{K}^\ell(D\widehat{A}, D\widehat{r}_0)$.

### 5.2.1 Preconditioning RPLanczos with LMPs

In this section we are interested in preconditioning RPLanczos with LMPs such that *the one-to-one correspondance between primal and dual iterates are preserved.* The term "primal approach" denotes here solving a sequence of linear systems (3.1) with multiple right-hand sides with PLanczos Algorithm 2.6, where the preconditioning is achieved by the LMP formulated by (2.38) (Approach (F)). Note that, the dual version of Approach (E) follows similar strategy as explained in Section 4.3.1 and Approach (G) is not necessary since RPLanczos does not require matrix-vector products with $B^{-1}$.

**Algorithm 5.5: RPLanczos Algorithm**

$\widehat{r}_0 = R^{-1}(Hs_0 - d)$
$\widehat{z}_0 = D\widehat{r}_0$
$t_0 = HBH^T\widehat{z}_0$
$\beta_0 = \sqrt{t_0^T\widehat{r}_0}$
$\widehat{v}_1 = \widehat{r}_0/\beta$
$\widehat{z}_1 = \widehat{z}_0/\beta$
$t_1 = t_0/\beta$
$\beta_1 = 0$
$\widehat{v}_0 = 0$
$\widehat{V} = \widehat{v}_1$
**for** $i = 1, 2, ..., l$ **do**
    $\widehat{q}_i = (\widehat{z}_i + R^{-1}t_i) - \beta_i\widehat{v}_{i-1}$
    $\alpha_i = \widehat{w}_i^T t_i$
    $\widehat{w}_i = \widehat{q}_i - \alpha_i\widehat{v}_i$
    $\widehat{z}_{i+1} = D\widehat{w}_i$
    $t_{i+1} = HBH^T\widehat{z}_{i+1}$
    $\beta_{i+1} = \sqrt{t_{i+1}^T\widehat{w}_i}$
    $\widehat{v}_{i+1} = \widehat{w}_i/\beta_{i+1}$
    $\widehat{z}_{i+1} = \widehat{z}_{i+1}/\beta_{i+1}$
    $t_{i+1} = t_{i+1}/\beta_{i+1}$
    $\widehat{V} = [\widehat{V}, \widehat{v}_{i+1}]$
    $(T_i)_{i,i} = \alpha_i$
    **if** $i > 1$ **then**
        $(T_i)_{i-1,i} = (T_i)_{i,i-1} = \beta_i$
    **end**
    Check convergence and stop if desired accuracy is reached
**end**
$y_\ell = T_\ell^{-1}\beta_0 e_1$
$s_\ell = s_0 + BH^T D\widehat{V}_\ell y_\ell$

In this section we focus on finding an algorithm that generates a mathematically equivalent sequence of iterates $\{s_k\}$ to those of Algorithm 3.2 where the inner minimization is performed by PLanczos, while working in the dual space.

In Algorithm 3.2, the initial guess is chosen as a zero vector because of the same reason explained in Section 4.3.1. In this case, RPLanczos is applied to the linear system (4.22) given as

$$\boxed{(R^{-1}HBH^T + I_m)\,\widehat{s} = -R^{-1}d_k - \sum_{j=1}^{k-1}\widehat{s}_j}.$$

whose initial residual $\widehat{r}_0$ with $\widehat{s} = 0$ satisfies Assumption 4.2 for the initial residual of the linear system (3.1) with $s_0 = 0$ (see subsection 4.3.1). This algorithm is provided by Algorithm 5.6. We name this approach *Approach (DualF)*.

**Algorithm 5.6: RPLanczos Algorithm (version for $s_0 = 0$)**

$\widehat{r}_0 = -R^{-1}d - \sum_{j=1}^{k} \widehat{s}_j$
$\widehat{z}_0 = D\widehat{r}_0$
$t_0 = HBH^T\widehat{z}_0$
$\beta_0 = \sqrt{t_0^T\widehat{r}_0}$
$\widehat{v}_1 = \widehat{r}_0/\beta$
$\widehat{z}_1 = \widehat{z}_0/\beta$
$t_1 = t_0/\beta$
$\beta_1 = 0$
$\widehat{v}_0 = 0$
$\widehat{V} = \widehat{v}_1$
**for** $i = 1, 2, ..., l$ **do**
    same pseudocode as Algorithm 5.5
**end**
$y_\ell = T_\ell^{-1}\beta_0 e_1$
$s_\ell = BH^T D\widehat{V}_\ell y_\ell$

Therefore, this algorithm can be applied to a convergent sequence of linear systems (4.22) by using Algorithm 4.4 whose inner loop is performed by RPLanczos Algorithm 5.6, and where the preconditioning is achieved by the LMP $D$ formulated by (4.26).

*As a result, the Lanczos version of Algorithm 4.4 generates a mathematically equivalent iterates of $\{s_k\}$ to those of the Lanczos version of Algorithm 3.2 under the assumptions that $P_0H^T = BH^TD_0$ and $x_1 = x_c$, while possibly yielding gains in terms of both memory usage and computational cost when $m \ll n$.*

We summarize the characteristics of the equivalent primal and dual Lanczos algorithms by Table 5.5.

| Algorithm | inner min. | LMP formula | Assumption |
|---|---|---|---|
| 3.2 | App. (F) | $P_{k+1}$ given by (2.38) with $M_k = P_k$ | $x_1 = x_c$, $s_0 = 0$ |
| 4.4 | App. (DualF) | $D_{k+1}$ given by (4.26) with $\widehat{M}_k = D_k$ and $S_k = BH^T\widehat{S}_k$ | $P_0H^T = BH^TD_0$ $x_1 = x_c$ |

Table 5.5: A summary of the characteristics of Algorithms 3.2 and 4.4. Approach (E) is defined in Section 3.1.

We next explain how to construct particular LMPs: the quasi-Newton LMP, the Ritz LMP and the spectral LMP described in Chapter 4.3 in the context of Algorithm 4.4 when used with RPLanczos. This part will include for each preconditioner an extraction methodology of the required information from the RPLanczos algorithm.

#### 5.2.1.1 Preconditioning with the quasi-Newton LMP

In this section, we are interested in preconditioning the linear system (4.22) with the quasi-Newton LMP given by (4.51). The pairs to construct the quasi-Newton LMP are not available from a Lanczos-type algorithm, however, they can be extracted from the iterates of Lanczos algorithm as explained in subsection 5.1.1. Following the similar strategy for deriving the iterates of CG Algorithm from those of Lanczos Algorithm as explained in Section 2.6.3, we can derive an algorithm to obtain the iterates of RPCG Algorithm 4.3 from those of RPLanczos Algorithm 5.6. This algorithm is provided by Algorithm 5.7.

**Algorithm 5.7: RPLanczos2RPCG**

Given $\ell, r_0, s_0, T_\ell, \beta_{\ell+1}, V_\ell$ from Algorithm 4.3
$(T_\ell)_{\ell+1,\ell} = \beta_{\ell+1}$
$\widehat{z}_0 = D\widehat{r}_0$
$l_0 = HBH^T\widehat{r}_0$
$w_0 = D^T l_0$
$\beta_0 = \sqrt{\widehat{r}_0^T w_0}$
$\widehat{p}_0 = \widehat{z}_0$
$t_0 = w_0$
$\rho = \beta_0^2$
**for** $i = 1, \ldots, \ell$ **do**
    $y_i = T_i^{-1}(\beta_0 e_1)$
    $\widehat{r}_i = -(T_l)_{i+1,i}\, v_{i+1} e_i^T y_i$
    $l_i = -(T_l)_{i+1,i}\, HBH^T v_{i+1} e_i^T y_i$
    $\widehat{z}_i = D\widehat{r}_i$
    $w_i = D^T l_i$
    $\rho_i = \widehat{r}_i^T w_i$
    $\beta_i = \rho_i/\rho_{i-1}$
    $\widehat{p}_i = \widehat{z}_i + \beta_i \widehat{p}_{i-1}$
    $t_i = w_i + \beta_i t_{i-1}$
    $\widehat{s}_i = s_0 + DV_i y_i$
    $\alpha_i = \|(\widehat{s}_i - \widehat{s}_{i-1})\|_2/\|\widehat{p}_{i-1}\|_2$
    $\widehat{q}_i = -(\widehat{r}_i - \widehat{r}_{i-1})/\alpha_i$
    $\varrho_i = -(l_i - l_{i-1})/\alpha_i$
**end**

Algorithm 5.7 requires matrix-vector products with $HBH^T$ at each loop. By introducing a new vector $l_i = HBH^T\widehat{v}_i$, Algorithm 5.6 can be transformed into Algorithm 5.8 in which the matrix-vector products, $HBH^T\widehat{v}_i$, can be obtained as by-products of RPLanczos Algorithm.

**Algorithm 5.8: RPLanczos Algorithm (version for quasi-Newton LMP)**

$\widehat{r}_0 = -R^{-1}d - \sum_{j=1}^{k} \widehat{s}_j$
$\widehat{z}_0 = D\widehat{r}_0$
$l_0 = HBH^T\widehat{r}_0$
$t_0 = D^T l_0$
$\beta_0 = \sqrt{t_0^T \widehat{r}_0}$
$\widehat{v}_1 = \widehat{r}_0/\beta_0$
$\widehat{z}_1 = \widehat{z}_0/\beta_0$
$l_1 = l_0/\beta_0$
$t_1 = t_0/\beta_0$
$\beta_1 = 0$
$\widehat{v}_0 = 0$
$\widehat{V} = \widehat{v}_1$
$L = l_1$
**for** $i = 1, 2, ..., \ell$ **do**
    $\widehat{q}_i = (\widehat{z}_i + R^{-1}t_i) - \beta_i \widehat{v}_{i-1}$
    $\alpha_i = \widehat{w}_i^T t_i$
    $\widehat{w}_i = \widehat{q}_i - \alpha_i \widehat{v}_i$
    $\widehat{z}_{i+1} = D\widehat{w}_i$
    $l_{i+1} = HBH^T\widehat{w}_i$
    $t_{i+1} = D^T l_{i+1}$
    $\beta_{i+1} = \sqrt{t_{i+1}^T \widehat{w}_i}$
    $\widehat{v}_{i+1} = \widehat{w}_i/\beta_{i+1}$
    $\widehat{z}_{i+1} = \widehat{z}_{i+1}/\beta_{i+1}$
    $t_{i+1} = t_{i+1}/\beta_{i+1}$
    $l_{i+1} = l_{i+1}/\beta_{i+1}$
    $\widehat{V} = [\widehat{V}, \widehat{v}_{i+1}]$
    $L = [L, l_{i+1}]$
    $(T_i)_{i,i} = \alpha_i$
    **if** $i > 1$ **then**
        $(T_i)_{i-1,i} = (T_i)_{i,i-1} = \beta_i$
    **end**
    Check convergence and stop if desired accuracy is reached
**end**
$y_\ell = T_\ell^{-1}\beta_0 e_1$
$s_\ell = BH^T D\widehat{V}_\ell y_\ell$

Therefore, when solving the system of linear equations (4.22) of the form of $\widehat{A}\widehat{s} = \widehat{b}_k$ with RPLanczos Algorithm 5.8 (at the $k$-th outer loop of Algorithm 4.4 with a constant $\widehat{A}$), the quasi-Newton LMP for the system matrix $\widehat{A}$ can be constructed from the

formula (4.51) given as

$$\boxed{D_{j+1} = \left(I_m - \tau_j \widehat{p}_j \widehat{q}_j^T HBH^T\right) D_j \left(I_m - \tau_j \widehat{q}_j t_j^T\right) + \tau_j \widehat{p}_j t_j^T}. \tag{5.24}$$

Note that, $\widehat{p}_i$, $i = 1, \ldots, j$, $\widehat{q}_i = \widehat{A}\widehat{p}_i$, $t_i = HBH^T\widehat{p}_i$, and $\tau_i = 1/(\widehat{q}_i^T t_i)$ are extracted from the iterates of RPLanczos Algorithm 5.8 obtained at the $(k-1)$-th outer loop by using Algorithm 5.7. We remind that the initial matrix to construct the preconditioner (4.51) at each outer loop $k$ is chosen as the preconditioner that is generated from the previous outer loop $(k-1)$. *To obtain this preconditioner with RPLanczos requires additional $O(m)$ operations when compared with RPCG.*

*As a result it is possible to find the corresponding quasi-Newton LMP (LMP (4.51)) for Algorithm 5.8 such that when it is used within the Lanczos version of Algorithm 4.4, it generates a mathematically equivalent sequence of iterates $\{s_k\}$ to those of the Lanczos version of Algorithm 3.2 with the quasi-Newton LMP (3.37), while possibly yielding gains in terms of both memory usage and computational cost when $m \ll n$. We showed that the quasi-Newton LMP can be constructed from vectors which can be extracted from by-products of Algorithm 5.8.*

We summarize the characteristics of the equivalent primal and dual Lanczos algorithms when used with the quasi-Newton LMP by Table 5.6. We assume that for Algorithm 3.2 $x_1 = x_c$ and $s_0 = 0$, and for Algorithm 4.4, $P_0 H^T = BH^T D_0$ and $x_1 = x_c$. In this table we define another approach for RPLanczos Algorithm when used with the quasi-Newton LMP:

| Approach (H): Apply RPLanczos Algorithm 5.8 on the linear system (4.22) with the $HBH^T$-inner product. |
|---|

| Algorithm | inner min. | pairs computation | quasi-Newton LMP |
|---|---|---|---|
| 3.2 | Approach (F) | Algorithm 2.7 | $P_{j+1}$ given by (3.37) |
| 4.4 | Approach (H) | Algorithm 5.7 | $D_{j+1}$ given by (4.51) |

Table 5.6: A summary of the characteristics of Algorithms 3.2 and 4.4 where the inner minimization is performed with corresponding Lanczos algorithms and preconditioning is achieved by using the quasi-Newton LMPs. Approach (F) is defined in Section 5.1.

#### 5.2.1.2 Preconditioning with the Ritz LMP

In this section we explain how to construct the Ritz-LMP (4.73), from the pairs obtained during RPLanczos Algorithm 5.8. As mentioned before, the information to construct a Ritz-LMP is available as by-products of a Lanczos algorithm.

Therefore, when applying RPLanczos on the linear system (4.22) at the $k$-th outer loop of Algorithm 4.4 (in which the linear system matrix is constant), the Ritz-LMP (4.73) can be constructed from Algorithm 4.8. The inputs, the matrices $D_{k-2}$ and $T_\ell$ and the scalar $\beta_{\ell+1}$, to Algorithm 4.8 are directly available from RPLanczos

Algorithm 5.8. The inputs, the vectors $v_i$ and $\widehat{g}_i$, $i = 1, \ldots, \ell+1$, to Algorithm 4.8 can be obtained as explained below.

From the definition of the vector $\widehat{g}_i = D\widehat{v}_i$ (see Lemma 4.4), it can be easily seen that $\widehat{g}_i = \widehat{z}_i$ where $\widehat{z}_i$ are defined in RPLanczos Algorithm. The vectors $v_i = HBH^T\widehat{z}_i$ can be obtained from

$$HBH^T\widehat{z}_i = D_{k-2}^T HBH^T\widehat{v}_i = D_{k-2}^T l_i,$$

where we used the symmetry of $D_{k-2}$ in the $HBH^T$-inner product (see Section 4.3.1).

*As a result it is possible to find the corresponding Ritz LMP (LMP (4.73)) for Algorithm 5.8 such that when it is used within the Lanczos version of Algorithm 4.4, it generates a mathematically equivalent sequence of iterates $\{s_k\}$ to those of the Lanczos version of Algorithm 3.2 with the Ritz LMP (3.65), while possibly yielding gains in terms of both memory usage and computational cost when $m \ll n$. We showed that the Ritz LMP can be constructed from vectors available from Algorithm 5.8.*

We summarize the characteristics of the equivalent primal and dual Lanczos algorithms when used with the Ritz LMP by Table 5.7. We assume that for Algorithm 3.2, $x_1 = x_c$ and $s_0 = 0$, and for Algorithm 4.4 $P_0H^T = BH^TD_0$ and $x_1 = x_c$.

| Algorithm | inner min. | Ritz LMP calculation | Ritz LMP formula |
|---|---|---|---|
| 3.2 | Approach (F) | Algorithm 3.11 | $P_k$ given by (3.65) |
| 4.4 | Approach (H) | Algorithm 4.8 | $D_k$ given by (4.73) |

Table 5.7: A summary of the characteristics of Algorithms 3.2 and 4.4 where the inner minimization is performed with corresponding Lanczos algorithms and preconditioning is achieved by using the quasi-Newton LMPs. Approaches (F) and (H) are defined in Section 5.1 and 5.2.1.1 respectively.

The cost and memory requirements for the Ritz LMP (4.73) are given in subsection 4.3.1.2.

#### 5.2.1.3 Preconditioning with the spectral LMP

The spectral LMP for RPCG is given by Lemma 4.6. The same formula can be applied to RPLanczos Algorithm. We refer to subsection 4.3.1.3 for details in deriving these formulas. In this section we summarize the characteristics of the equivalent primal and dual Lanczos algorithms when used with the spectral LMP by Table 5.8.

| Algorithm | inner min. | spectral LMP formula | Assumption |
|---|---|---|---|
| 3.2 | Approach (F) | $P_k$ given by (3.78) | $x_1 = x_c$, $s_0 = 0$ |
| 4.4 | Approach (DualF) | $D_k$ given by (4.74) | $P_0 H^T = B H^T D_0$<br>$x_1 = x_c$ |

Table 5.8: A summary of the characteristics of Algorithms 3.2 and 4.4 where the inner minimization is performed with corresponding Lanczos algorithms and preconditioning is achieved by using the spectral LMPs. Approaches (F) and (DualF) are the approaches defined in Section 3.1 and 4.3.1 respectively.

## 5.3 Solving the subproblem with Augmented RPLanczos

This section presents the final version of RPLanczos that allows for an arbitrary initial guess. The generalized algorithm relies on augmentation like augmented RPCG. We name this algorithm augmented RPLanczos, which is mathematically equivalent to augmented RPCG.

Under Assumption 4.3 on the preconditioner, the augmented RPLanczos algorithm can be derived from PLanczos Algorithm 2.6 on the linear system (4.80) following the basic approach previously used (to derive RPLanczos Algorithm 5.5 from PLanczos Algorithm 2.6 on the linear system (3.1)). The augmented RPLanczos algorithm is given by Algorithm 5.9. This algorithm can be applied to a convergent sequence of linear systems (4.14) by using Algorithm 4.11.

The preconditioner $\underline{D} \in \mathbb{R}^{(m+1)\times(m+1)}$ derived in Lemma 4.7 can be used for the augmented RPLanczos algorithm as explained in subsections 4.4.1 and 4.4.2. The required pairs for this preconditioner can be found following the same approach for the augmented RPCG.

### 5.3.1 Preconditioning varying systems

The same results explained in subsection 4.4.2 apply for RPLanczos when solving a sequence of varying linear systems (4.14) preconditioned by an LMP.

## 5.4 Convergence issues

The convergence behaviour of the Lanczos algorithm is well known and for details we refer to (Meurant, 2006; Golub and Van Loan, 1996; Saad, 1996). When Lanczos method is used with one of the particular LMPs, the convergence properties follow similar properties to PCG with an LMP. The convergence properties of RPLanczos with an LMP also follow similar properties to RPCG since these algorithms are mathematically equivalent.

In this section we explain how to re-orthogonalize the Lanczos vectors generated by the RPLanczos algorithm in presence of round-off errors. The section will be concluded by providing inexpensive formulas for computing the values of quadratic cost

**Algorithm 5.9: Augmented RPLanczos Algorithm**

Choose an initial state $s_0$
$\underline{r}_0 = \underline{d} - \underline{R}^{-1}\underline{H}s_0$
$\underline{z}_0 = \underline{D}\,\underline{r}_0$
$\underline{t}_0 = \underline{H}B\underline{H}^T\underline{z}_0$
$\beta_0 = \sqrt{\underline{t}_0^T\underline{r}_0}$
$\underline{v}_1 = \underline{r}_0/\beta$
$\underline{z}_1 = \underline{z}_0/\beta$
$\underline{t}_1 = \underline{t}_0/\beta$
$\beta_1 = 0$
$\underline{v}_0 = 0$
$\underline{V} = \underline{v}_1$
**for** $i = 1, 2, ..., \ell$ **do**
    $\underline{q}_i = (\underline{z}_i + \underline{R}^{-1}\underline{t}_i) - \beta_i\underline{v}_{i-1}$
    $\alpha_i = \underline{w}_i^T\underline{t}_i$
    $\underline{w}_i = \underline{q}_i - \alpha_i\underline{v}_i$
    $\underline{z}_{i+1} = \underline{D}\,\underline{w}_i$
    $t_{i+1} = \underline{H}B\underline{H}^T\underline{z}_{i+1}$
    $\beta_{i+1} = \sqrt{\underline{t}_{i+1}^T\underline{w}_i}$
    $\underline{v}_{i+1} = \underline{w}_i/\beta_{i+1}$
    $\underline{z}_{i+1} = \underline{z}_{i+1}/\beta_{i+1}$
    $\underline{t}_{i+1} = \underline{t}_{i+1}/\beta_{i+1}$
    $\underline{V} = [\underline{V}, \underline{v}_{i+1}]$
    $(T_i)_{i,i} = \alpha_i$
    **if** $i > 1$ **then**
        $(T_i)_{i-1,i} = (T_i)_{i,i-1} = \beta_i$
    **end**
    Check convergence and continue if necessary
**end**
$y_\ell = T_\ell^{-1}\beta_0 e_1$
$s_\ell = s_0 + B\underline{H}^T\underline{D}\,\underline{V}_\ell y_\ell$

function (2.13) and its gradient norm on each iteration from the recurrence relations of PLanczos and RPLanczos algorithms.

### 5.4.1 Re-orthogonalization

The Lanczos vectors in PLanczos (like the residuals in PCG) are, in exact arithmetic, mutually orthogonal. Round-off errors can result in a loss of the orthogonality that can significantly hinder the convergence of these methods. It is possible to alleviate this problem by re-orthogonalizing the Lanczos vectors on each iteration using a modified Gram-Schmidt (MGS) procedure (Saad, 1996, p. 11-12).

Considering Algorithm 2.6, the re-orthogonalization procedure acts on the vectors $w_i$. Making use of the orthogonality relationship $v_i^T P v_j = 0$, for $i \neq j$, the MGS

re-orthogonalization can be written in compact notation by

$$w_i \leftarrow \left[\prod_{j=i-1}^{1}(I_n - v_j v_j^T P)\right] w_i, \tag{5.25}$$

$$= \left[\prod_{j=i-1}^{1}(I_n - v_j z_j^T)\right] w_i. \tag{5.26}$$

Equations (5.25) and (5.26) require the storage of the Lanczos vectors $v_j$ from all previous iterations $j = 1, \ldots, i-1$. Equation (5.25) can be less practical than Eq. (5.26) when matrix-vector product with $P$ is expensive. In this case, the vectors $z_j$ can also be stored during PLanczos and used in Eq. (5.26).

The need to store and manipulate the sequence of $n$-dimensional vectors $v_j$ and $z_j$ can lead to significant computational overhead. In this respect, the dual algorithms, which involve sequences of $m$-dimensional vectors, are clearly an attractive alternative when $m \ll n$. From the relations $v_i = H^T \widehat{v}_i$ and $PH^T = BH^T D$, the orthogonality condition for the Lanczos vectors can be written as

$$v_i^T P v_j = \widehat{v}_i^T H P H^T \widehat{v}_j = \widehat{v}_i^T H B H^T D \widehat{v}_j = 0 \tag{5.27}$$

for $i \neq j$. Combining Eq. (5.27) with Eqs (5.25) and (5.26) leads to the MGS re-orthogonalization scheme for RPLanczos as

$$\widehat{w}_i \leftarrow \left[\prod_{j=i-1}^{1}(I_m - \widehat{v}_j \widehat{v}_j^T D^T H B H^T)\right] \widehat{w}_i,$$

$$= \left[\prod_{j=i-1}^{1}(I_m - \widehat{v}_j t_j^T)\right] \widehat{w}_i.$$

Note that the corresponding re-orthogonalization scheme for the augmented RPLanczos follows the similar derivation.

*It is remarkable that re-orthogonalization of residuals may be considered as costly in terms of memory and computational cost in the primal setting, turns out to be much cheaper in the dual frameworks since the dimension of the residual vectors in dual space is (typically much) smaller than the dimension of the residual vectors in primal space.*

### 5.4.2 Monitoring convergence

The values of the quadratic cost function and the norm of the cost function gradient are important for monitoring the convergence of the minimization. Cheap formulae for computing all these quantities on each iteration from the recurrence relations of the PLanczos and RPLanczos algorithms (Algorithms 2.6, 5.2, and 5.6, 5.9) are derived in this section. Note that for the sake of simplicity, the initial guess is chosen as $s_0 = 0$ for all derivations.

### Quadratic cost function

The quadratic cost function can be expressed in terms of quantities from PLanczos by substituting the relation $s_i = Z_i y_i$ into Equation (3.87) as

$$Q[s_i] = Q[0] - \frac{1}{2} y_i^T Z_i^T r_0, \tag{5.28}$$

where $Z_i = PV_i$, $i = 0, 1, ..., \ell - 1$, $\ell$ is the number of iterations performed during PLanczos and

$$Q[0] = f(x_k) = \frac{1}{2} d^T R^{-1} d + \frac{1}{2}(x_c - x_k)^T B^{-1}(x_c - x_k). \tag{5.29}$$

The background term $Q_b$ can be calculated from (3.88) as

$$Q_b[s_i] = \frac{1}{2} y_i^T Z_i^T B^{-1} Z_i y_i + \frac{1}{2}(x_k - x_c)^T B^{-1}(x_k - x_c). \tag{5.30}$$

The observation term, $Q_o$, can then be calculated from Equation (3.89).

The expressions for $Q$ and $Q_b$ depend on the vectors $y_i$. These vectors can be computed on each iteration by solving the (inexpensive for small $i$) tridiagonal system $T_i y_i = \beta_0 e_1$.

For PLanczosIF the same formula can be used for the quadratic cost function value $Q[s_i]$ and the term $Q_o[s_i]$. The term $Q_b[s_i]$ can alternatively calculated from

$$\begin{aligned} Q_b[s_i] &= \frac{1}{2} y_i^T Z_i^T B^{-1} Z_i y_i + \frac{1}{2}(x_k - x_c)^T B^{-1}(x_k - x_c) \\ &= \frac{1}{2} y_i^T Z_i^T B^{-1} B \underline{Z}_i y_i + \frac{1}{2}(x_k - x_c)^T B^{-1}(x_k - x_c) \\ &= \frac{1}{2} y_i^T Z_i^T \underline{Z}_i y_i + \frac{1}{2}(x_k - x_c)^T B^{-1}(x_k - x_c) \end{aligned}$$

where $\underline{Z}_i = [\underline{z}_1, \ldots, \underline{z}_i]$ can be stored during PLanczosIF. Taking $x_1 = x_c$ and using the relation $s_i = B\underline{Z}_i y_i$, the term $B^{-1}(x_k - x_c)$ can also be calculated from

$$B^{-1}(x_k - x_c) = \sum_{j=0}^{k-1} B^{-1} s_j = \sum_{j=0}^{k-1} \underline{Z}_l^{(j)} y_l^{(j)}$$

where $\underline{Z}_l^{(j)}$ denotes the last iteration ($i = l$) of PLanczosIF at the $j$-th outer loop of Gauss-Newton method.

The values of $Q$ and $Q_b$ can be evaluated in terms of quantities from RPLanczos Algorithm 5.6 by using the relations $r_0 = H^T \widehat{r}_0$, and the relation for $z_i$ given by Equation (5.13) in the expressions (5.28) and (5.30). This yields

$$\begin{aligned} Q[s_i] &= Q[0] - \frac{1}{2} y_i^T Z_i^T H^T \widehat{r}_0 \\ &= Q[0] - \frac{1}{2} y_i^T \widehat{Z}_i^T H B H^T \widehat{r}_0 \end{aligned}$$

and

$$\begin{aligned} Q_b[s_i] &= \frac{1}{2} y_i^T Z_i^T B^{-1} Z_i y_i + \frac{1}{2}(x_k - x_c)^T B^{-1}(x_k - x_c) \\ &= \frac{1}{2} y_i^T \widehat{Z}_i^T H B B^{-1} B H^T \widehat{Z}_i y_i + \frac{1}{2}(x_k - x_c)^T B^{-1}(x_k - x_c) \\ &= \frac{1}{2} y_i^T \widehat{Z}_i^T H B H^T \widehat{Z}_i y_i + \frac{1}{2}(x_k - x_c)^T B^{-1}(x_k - x_c) \end{aligned}$$

where $Q[0]$ is given by Eq. (5.29). Using the symmetry of $D$ with respect to the $HBH^T$-(semi) inner product in RPLanczos Algorithm 5.6 , the vector $t_0$ can be calculated also from $t_0 = D^T H B H^T \widehat{r}_0$. Similarly the vector $t_i$, $i > 1$ can be calculated also from $t_i = D^T H B H^T \widehat{w}_i$. Therefore, defining $l_0 = H B H^T \widehat{r}_0$, and $l_i = H B H^T \widehat{w}_i$ the inexpensive formulas for $Q[s_i]$ and $Q_b[s_i]$ can be written as

$$Q[s_i] = Q[0] - \frac{1}{2} y_i^T \widehat{Z}_i^T l_0$$

and

$$Q_b[s_i] = \frac{1}{2} y_i^T \widehat{Z}_i^T D^T L_i y_i + \frac{1}{2}(x_k - x_c)^T B^{-1}(x_k - x_c)$$

Taking $x_0 = x_c$ the term $B^{-1}(x_k - x_c)$ can also be calculated without the need to apply $B^{-1}$ from Eq. (4.19). The $Q_o$ term can then be computed from Eq. (3.89). The matrices $\widehat{Z}_i$ and $L_i$ contain the $j = 1, \ldots, i$ column vectors $\widehat{z}_j$ and $l_j$.

For Augmented RPLanczos, the formula can be obtained similarly following the same derivation by using augmented matrices and vectors. While replacing the term $B^{-1}(x_k - x_c)$, since $H_k$ varies along the outer loop, the following equation has to be used instead of the Eq. (4.19)

$$B^{-1}(x_k - x_c) = \sum_{j=0}^{k-1} H_j^T \widehat{Z}_l^{(j)} y_l^{(j)}.$$

### Norm of the quadratic cost function gradient

The residual vector of the approximate solution $s_i$ computed by PLanczos satisfies (Saad, 1996, p.153)

$$r_i = b - A s_i = -\beta_{i+1} e_i^T y_i v_{i+1},$$

where $e_i$ is the $i$-th column of the $i \times i$ identity matrix, and $\beta_{i+1}$, $y_i$ and $v_i$ are as defined in Algorithm 2.6. The $P$-norm of the cost function gradient can be readily computed as

$$\|\nabla Q[s_i]\|_P = \|r_i\|_P = \beta_{i+1} |e_i^T y_i|.$$

The gradient norm for PCGIF, RPLanczos and the augmented RPLanczos can be calculated equivalently from the same formula.

# CHAPTER 6

## Towards globally convergent algorithms

In this chapter we consider solving the regularized nonlinear least-squares problem (2.3) by using a method based on a Gauss-Newton technique, made globally convergent with a trust-region strategy. The sequence of linear least-squares problems involved in the method is iteratively solved by a conjugate-gradients method, appropriately truncated by the Steihaug-Toint strategy, and which is accelerated by limited memory preconditioners.

We present numerical experiments on a test problem based on the heat equation which can be considered as an idealized data assimilation system.

### 6.1 The Steihaug-Toint truncated conjugate gradient method

When using the simple Gauss-Newton approach described at Section 2.5 for more than mildly non-linear cost functions, the iterations can unfortunately diverge, and the function value can increase with the Gauss-Newton step computed from (2.46), see for instance (Kelley, 1999, p.39). This problem is not purely theoretical, and is also discussed in a real life problem in (Tshimanga, Gratton, Weaver and Sartenaer, 2008), where the necessity for global minimization is emphasized. As indicated above, global convergence can be ensured by inserting the Gauss-Newton strategy in a trust-region framework. For the nonlinear least-squares problem (2.3), trust-region methods amount to solving approximately a sequence of quadratic problems

$$\min_{s_k} \frac{1}{2} \|s_k + x_k - x_c\|^2_{B^{-1}} + \frac{1}{2} \|H_k s_k + d_k\|^2_{R^{-1}} \tag{6.1}$$

$$\text{subject to } \|s_k\|_{P_k^{-1}} \leq \Delta_k, \tag{6.2}$$

where $\Delta_k$ is the radius of the "trust region", which is the region where we believe that the objective function (2.3) of our nonlinear problem is adequately approximated by that of (6.1). It is important to note that preconditioning appears in this problem as

the norm $\|\cdot\|_{P_k^{-1}}$ used in (6.2).

After solving this subproblem, as explained in Section 2.5.3 the step $s_k$ is accepted or rejected and the trust region radius is updated accordingly. The acceptance of the trial point and trust region radius update are decided by considering the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$$

where $f$ is the objective function (2.4) and $m_k$ is its quadratic approximation (6.1). This ratio of achieved to predicted reductions gives an indication of the model's quality. If it is larger than some constant, the step is accepted and the trust-region radius possibly enlarged, while, if it is too small or negative, the step is rejected and the trust-region radius decreased. We refer the reader to (Conn, Gould and Toint, 2000, p. 116) for a more complete description. For large scale instances, the subproblem (6.1)-(6.2) can be solved approximately using iterative techniques, for instance the Large-Scale Trust-Region Subproblem (LSTRS) algorithm (Lampe, Rojas, Sorensen and Voss, 2011; Rojas, Santos and Sorensen, 2008), the Generalized Lanczos Trust-Region method (GLTR) (Gould, Lucidi, Roma and Toint, 1999; Conn, Gould and Toint, 2000), or the Steihaug-Toint truncated conjugate-gradient method (Conn, Gould and Toint, 2000, p. 205). We consider below an efficient implementation of the Steihaug-Toint truncated conjugate gradient method. In Section 6.2, we briefly explain the GLTR method. A similar adaptation of LSTRS is out of the scope of this thesis.

In the Steihaug-Toint truncated conjugate-gradient technique, the model (6.1) is approximately minimized using PCG until the boundary of the trust region (6.2) is encountered. More specifically, (dropping again the outer-iterations index $k$ for simplicity) three different cases may occur when applying PCG to (6.1) (Conn, Gould and Toint, 2000, pp. 202-204):

1. the curvature $\langle p_i, Ap_i\rangle$ remains positive at each inner iteration, and the PCG iterates remain inside the trust region (the standard PCG stopping rule depending on the relative gradient norm then applies);

2. the curvature $\langle p_i, Ap_i\rangle$ remains positive at each inner iteration, and the PCG iterates leave the trust region, in which case the iterates are stopped when the trust region boundary is met;

3. the curvature $\langle p_i, Ap_i\rangle$ is negative at some PCG step, in which case, the associated descent direction is followed until the trust region boundary is met.

This strategy can be shown to yield a sufficient decrease condition (Nocedal and Wright, 2006, p. 33),(Conn, Gould and Toint, 2000, p. 205) which guarantees global convergence of the iterates.

Again the same question arises: may we derive an equivalent dual-space version of this method? In particular, how easy is it to compute a final iterate on the boundary of the trust region following a descent direction form a given inner iterate? For answering these questions, we start by rewriting the Steihaug-Toint algorithm described in (Conn, Gould and Toint, 2000, p. 205) in terms of the vectors defined in Algo-

rithm 4.9, using the relations (Gratton and Tshimanga, 2009) $r_i = \underline{H}^T \underline{r}_i$, $p_i = B\underline{H}^T \underline{p}_i$, $s_i = s_0 + B\underline{H}^T \lambda_i$, $z_i = B\underline{H}^T \underline{z}_i$, $q_i = \underline{H}^T \underline{q}_i$ and the equality $P\underline{H}^T = B\underline{H}^T \underline{D}$ where $\underline{D} \in \mathbb{R}^{(m+1)\times(m+1)}$ is the preconditioner. This gives a first version of the Steihaug-Toint truncated conjugate gradient algorithm (Algorithm 6.1) for the dual approach.

**Algorithm 6.1: The Steihaug-Toint truncated CG method in dual space (version 1)**

$\lambda_0 = 0$
$\underline{r}_0 = \underline{d} - \underline{R}^{-1}\underline{H}s_0$
$\underline{z}_0 = \underline{D}\,\underline{r}_0$
$\underline{p}_0 = \underline{z}_0$
**for** $i = 0, 1, ..., l-1$ **do**
  $\underline{q}_i = (\underline{R}^{-1}\underline{H}B\underline{H}^T + I_{m+1})\underline{p}_i$
  $\mathcal{K}_i = \underline{p}_i^T \underline{H}B\underline{H}^T \underline{q}_i$
  **if** $\mathcal{K}_i \leq 0$ **then**
    compute $\alpha_i$ as the positive root of $\|\lambda_i + \alpha_i \underline{p}_i\|_{\underline{H}B\underline{H}^T\underline{D}^{-1}} = \Delta$
    $\lambda_{i+1} = \lambda_i + \alpha_i \underline{p}_i$
    return
  **end**
  $\alpha_i = \langle \underline{r}_i, \underline{z}_i \rangle_{\underline{H}B\underline{H}^T} / \mathcal{K}_i$
  **if** $\|\lambda_i + \alpha_i \underline{p}_i\|_{\underline{H}B\underline{H}^T\underline{D}^{-1}} \geqslant \Delta$ **then**
    compute $\alpha_i$ as the positive root of $\|\lambda_i + \alpha_i \underline{p}_i\|_{\underline{H}B\underline{H}^T\underline{D}^{-1}} = \Delta$
    $\lambda_{i+1} = \lambda_i + \alpha_i \underline{p}_i$
    return
  **end**
  $\lambda_{i+1} = \lambda_i + \alpha_i \underline{p}_i$
  $\underline{r}_{i+1} = \underline{r}_i - \alpha_i \underline{q}_i$
  $\underline{z}_{i+1} = \underline{D}\,\underline{r}_{i+1}$
  Check convergence and continue if necessary
  $\beta_i = \langle \underline{r}_{i+1}, \underline{z}_{i+1} \rangle_{\underline{H}B\underline{H}^T} / \langle \underline{r}_i, \underline{z}_i \rangle_{\underline{H}B\underline{H}^T}$
  $\underline{p}_{i+1} = \underline{z}_{i+1} + \beta_i \underline{p}_i$
**end**
The solution is recovered from $s_l = s_0 + B\underline{H}^T \lambda_l$

As before, this version of the algorithm turns out to be very expensive in terms of $\underline{H}B\underline{H}^T$ matrix-vector products, and we introduce new vectors to transform it into a computationally efficient method. From

$$\|\lambda_i + \alpha_i \underline{p}_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}} = \|\lambda_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}} + 2\alpha_i \langle \lambda_i, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i \rangle + \alpha_i^2 \|\underline{p}_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}},$$

the positive root of $\|\lambda_i + \alpha_i \underline{p}_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}} = \Delta$ is given by

$$\alpha_i = \frac{-\langle \lambda_i, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i \rangle + \sqrt{\langle \lambda_i, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i \rangle^2 + \|\underline{p}_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}} \left(\Delta^2 - \|\lambda_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}}\right)}}{\|\underline{p}_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}}}.$$

We now explain how to compute the terms given in this expression recursively. Let us define $a_{i+1} = \|\underline{p}_{i+1}\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}}$. From line 24 of Algorithm 6.1 we can write

$$a_{i+1} = \underline{z}_{i+1}^T \underline{H}B\underline{H}^T \underline{r}_{i+1} + 2\,\beta_i\, \underline{p}_i^T \underline{H}B\underline{H}^T \underline{r}_{i+1} + \beta_i^2\, \|\underline{p}_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}}$$

Since the vectors $\underline{r}_{i+1}$ and $\underline{p}_i$ are orthogonal with respect to $\underline{H}B\underline{H}^T$, we obtain that

$$a_{i+1} = \langle \underline{r}_{i+1}, \underline{z}_{i+1} \rangle_{\underline{H}B\underline{H}^T} + \beta_i^2\, a_i.$$

The term $b_{i+1} = \langle \lambda_{i+1}, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_{i+1} \rangle$ can be calculated by using line 24 of Algorithm 6.1 and the relation $\lambda_{i+1} = \sum_{j=0}^{i} \alpha_j \underline{p}_j$ (see line 19 of Algorithm 6.1) as follows

$$\begin{aligned}
b_{i+1} &= \left(\sum_{j=0}^{i} \alpha_j \underline{p}_j\right)^T (\underline{H}B\underline{H}^T \underline{r}_{i+1} + \beta_i\, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i) \\
&= \left(\sum_{j=0}^{i} \alpha_j \underline{p}_j\right)^T \underline{H}B\underline{H}^T \underline{r}_{i+1} + \left(\sum_{j=0}^{i} \alpha_j \underline{p}_j\right)^T \beta_i\, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i \\
&= \left(\sum_{j=0}^{i-1} \alpha_j \underline{p}_j\right)^T \beta_i\, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i + \alpha_i\, \underline{p}_i^T \beta_i\, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i \\
&= \lambda_i^T \beta_i\, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i + \beta_i\alpha_i\, \underline{p}_i^T\, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i \\
&= \beta_i \left(\langle \lambda_i, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i \rangle + \alpha_i \|\underline{p}_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}}\right) \\
&= \beta_i\, (b_i + \alpha_i a_i)\,.
\end{aligned}$$

Finally, defining $c_{i+1} = \|\lambda_{i+1}\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}}$, this term can be calculated from

$$\begin{aligned}
c_{i+1} &= \|\lambda_i + \alpha_i \underline{p}_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}} \\
&= \|\lambda_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}} + 2\,\alpha_i \langle \lambda_i, \underline{H}B\underline{H}^T\underline{D}^{-1}\underline{p}_i \rangle + \alpha_i^2\, \|\underline{p}_i\|^2_{\underline{H}B\underline{H}^T\underline{D}^{-1}} \\
&= c_i + 2\,\alpha_i\, b_i + \alpha_i^2\, a_i.
\end{aligned}$$

We may now use these scalars to calculate $\alpha_i$, which yields

$$\alpha_i = \frac{-b_i + \sqrt{b_i^2 + a_i(\Delta^2 - c_i)}}{a_i}. \tag{6.3}$$

Introducing this change, we now obtain Algorithm 6.2 on the next page. This last version requires a single $\underline{H}B\underline{H}^T$ matrix-vector product and products with $\underline{D}$ and $\underline{D}^T$

in each inner loop.

**Algorithm 6.2: The Steihaug-Toint truncated CG method in dual space**

$\lambda_0 = 0$;
$\underline{r}_0 = \underline{d} - \underline{R}^{-1}\underline{H}s_0$
$l_0 = \underline{H}B\underline{H}^T\underline{r}_0$
$\underline{z}_0 = \underline{D}\underline{r}_0$
$\underline{p}_0 = \underline{z}_0$
$w_0 = \underline{D}^T l_0$
$t_0 = w_0$
$\rho_0 = w_0^T\underline{r}_0$
$a_0 = \rho_0$
$b_0 = 0$
$c_0 = 0$
**for** $i = 0, 1, ...$ **do**
    $\underline{q}_i = \underline{R}^{-1}t_i + \underline{p}_i$
    $\mathcal{K}_i = t_i^T\underline{q}_i$
    **if** $\mathcal{K}_i \leq 0$ **then**
        Calculate $\alpha_i$ from the formula (6.3)
        return
    **end**
    $\alpha_i = \rho_i/\mathcal{K}_i$
    $\gamma = \sqrt{c_i + 2\,\alpha_i\,b_i + \alpha_i^2\,a_i}$
    **if** $\gamma \geqslant \Delta$ **then**
        Calculate $\alpha_i$ from the formula (6.3)
        $\lambda_{i+1} = \lambda_i + \alpha_i\underline{p}_i$
        return
    **end**
    $\lambda_{i+1} = \lambda_i + \alpha_i\underline{p}_i$
    $\underline{r}_{i+1} = \underline{r}_i - \alpha_i\underline{q}_i$
    $l_{i+1} = \underline{H}B\underline{H}^T\underline{r}_{i+1}$
    $\varrho_i = (l_i - l_{i+1})/\alpha_i$
    $\underline{z}_{i+1} = \underline{D}\underline{r}_{i+1}$
    $w_{i+1} = \underline{D}^T l_{i+1}$
    Check convergence and stop if desired accuracy is reached
    $\rho_{i+1} = w_{i+1}^T\underline{r}_{i+1}$
    $\beta_i = \rho_{i+1}/\rho_i$
    $\underline{p}_{i+1} = \underline{z}_{i+1} + \beta_i\underline{p}_i$
    $t_{i+1} = w_{i+1} + \beta_i t_i$
    $c_{i+1} = c_i + 2\alpha_i b_i + \alpha_i^2 a_i$
    $b_{i+1} = \beta_i(b_i + \alpha_i a_i)$
    $a_{i+1} = \rho_{i+1} + \beta_i^2\,a_i$
**end**
The solution is recovered from $s_l = s_0 + B\underline{H}^T\lambda_l$

Inserting this algorithm in a Gauss Newton method in the framework of trust region yields Algorithm 6.3. This is the algorithm which we recommend for solving truly nonlinear instances of our original problem (2.3) when $m \ll n$ and under Assumption 4.3.

**Algorithm 6.3: Gauss-Newton trust region algorithm for dual approach**

1 **Initialization:** Choose an initial point $x_0$ and an initial trust region radius $\Delta_0$. Choose a first level preconditioner $\underline{D}_0$ such that Assumption 4.3 holds. Choose an initial estimate $s_0$. Choose the constants $\eta_1, \eta_2, \gamma_1, \gamma_2$ that satisfy

$$0 < \eta_1 \leq \eta_2 < 1 \text{ and } 0 < \gamma_1 \leq \gamma_2 < 1$$

Set $k = 0$ and calculate the function value at the initial, $f(x_0)$.

2 **Calculation of $s_k$ approximately:** Solve the subproblem (2.46) by using Steihaug-Toint CG Algorithm 6.2.

3 **Acceptance of the trial point:** Compute the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)},$$

If $\rho_k \geq \eta_1$, $x_{k+1} = x_k + s_k$; otherwise, $x_{k+1} = x_k$.

4 **Trust-region radius update:** Set

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty) & \text{if } \rho_k \geq \eta_2, \\ [\gamma_2 \Delta_k, \Delta_k) & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_1 \Delta_k, \gamma_2 \Delta_k) & \text{if } \rho_k < \eta_1, \end{cases}$$

Increment $k$ by 1 and go to step 2.

When an LMP defined in Lemma 4.7 used in Algorithm 6.3 with a symmetry measure as explained in Section 4.4.2, it is clear that *efficient preconditioning depends on the choice of threshold value*. When this value is not chosen properly, *the conjugate gradient method may be in trouble with a preconditioner that is not symmetric and positive definite*. Moreover, to be used in the trust region framework, the trust region may not be defined since the norm is defined by the preconditioner. Therefore, in this case we could not use Algorithm 6.3 directly. We need to adapt this algorithm as detailed in the next section.

### 6.1.1 Flexible truncated conjugate gradients

In this section we adapt the Steihaug-Toint truncated conjugate gradients for the dual approach given by Algorithm 6.2 when the trust region could not be defined with the standard approach, since the norm is not defined. As we mentioned this may happen when the preconditioner that is inherited from previous iteration may not be symmetric in the current inner product and may not be positive-defnite in the full dual space.

Therefore we need adapt the strategy in the trust region algorithm.

We first ensure that the preconditioner is positive definite along the steepest descent direction $-g_k(=-\nabla_x f(x_k))$. Then we search for the Cauchy step $s_c^{(k)}$ by using Steihaug-Toint truncated CG, that reduces the model along the steepest descent direction while satisfying the bound $\|s_c^{(k)}\| \leq \Delta_k$. The resulting point is known as the Cauchy point $x_k^C$ (Conn, Gould and Toint, 2000), i.e

$$x_k^C = x_k - t_k^C g_k = \arg \min_{\substack{t \geq 0 \\ x_k - t g_k \in \mathcal{B}_k}} m_k(x_k - t g_k). \tag{6.4}$$

From the Cauchy point, we continue with Augmented RPCG to get $x_k^{CG} = x_k + s_k$. The iteration is stopped either when the residual norm reduces by a factor of a given tolerance or the maximum number of iterations is reached. Note that, we do not impose the trust-region bound for searching the step after the Cauchy step. Once we obtain the trial step, the step beyond the Cauchy step is accepted if

$$f(x_k^{CG}) < f(x_k^C),$$

otherwise we choose the trial point as the Cauchy point. This strategy leads to the magical step strategy explained in Conn, Gould and Toint (2000, p. 387) where the step determined within the trust region is taken as the Cauchy step and the magical step is calculated from Algorithm 4.9. We next check the change from $x_k$ to $x_k^{(CG)}$ including the contribution of the objective function since the improvement from $s_k^C$ to $s_k^{CG}$ does not depend on any prediction using the model approximation (Conn, Gould and Toint, 2000, p. 388). If this change is larger than some small constant, the step is accepted and the trust-region radius is possibly enlarged, while, if it is too small or negative, the step is rejected and the trust-region radius is decreased. We outline this strategy by Algorithm 6.5 in which the subproblem is solved by Algorithm 6.4.

**Algorithm 6.4: Flexible truncated CG for dual approach**

1 Initialization
2 Compute $\underline{r}_0^{(k)}$
3 Check the positive-definiteness of the preconditioner along $\underline{r}_0^{(k)}$
4 Compute the Cauchy step $s_k^C$ by using Algorithm 6.2
5 Compute the step beyond the Cauchy step $s_k^{CG}$ with the augmented RPCG algorithm (Algorithm 4.9)

**Algorithm 6.5: Flexible trust region algorithm for dual approach**

**1 Initialization:** Choose an initial point $x_0$ and an initial trust region radius $\Delta_0$. Choose a first level preconditioner $\underline{D}_0$ such that Assumption 4.3 holds. Choose an initial estimate $s_0$. Choose the constants $\eta_1, \eta_2, \gamma_1, \gamma_2$ that satisfy

$$0 < \eta_1 \leq \eta_2 < 1 \text{ and } 0 < \gamma_1 \leq \gamma_2 < 1$$

Set $k = 0$ and calculate the function value at the initial, $f(x_0)$.

**2 Norm definition:** Define $\|\cdot\|_k$.

**3 Preconditioner construction:** Decide which pairs to use according to the success of the previous inner loop

**4 Calculation of $s_k$ approximately:** Solve the subproblem (2.46) by using flexible conjugate gradients Algorithm 6.4. During Algorithm 6.4 extract and save relevant information to precondition the next linear system

**5 Acceptance of the step beyond the Cauchy step:** If

$$f(x_k^{CG}) < f(x_k^C),$$

$s_k = s_k^{CG}$; otherwise, $s_k = s_k^C$.

**6 Acceptance of the trial point:** Compute the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k^C) + f_k(x_k^C) - f_k(x_k + s_k)},$$

If $\rho_k \geq \eta_1$, $x_{k+1} = x_k + s_k$; otherwise, $x_{k+1} = x_k$.

**7 Trust-region radius update:** Set

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty) & \text{if } \rho_k \geq \eta_2, \\ [\gamma_2 \Delta_k, \Delta_k) & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_1 \Delta_k, \gamma_2 \Delta_k) & \text{if } \rho_k < \eta_1, \end{cases}$$

Increment $k$ by 1 and go to step 2.

Note that Algorithm 6.5 requires the function evaluation both at $x_k^C$ and $x_k^{CG}$. Note also that in Algorithm 6.5, finite number of iterations are allowed. Convergence theory of trust region with magical steps is given in (Conn, Gould and Toint, 2000, p. 388-391).

## 6.2 The Generalized Lanczos Trust-Region method

As indicated before, global convergence of a Gauss Newton method can be ensured by using a trust-region method. A globally convergent algorithm stated as the Steihaug-Toint truncated conjugate gradient method is presented in Section 6.1. This technique minimizes the model (6.1) approximately by using PCG and stops when the boundary of the trust region (6.2) is encountered. Till the trust region boundary is met, this method searches the solution in a subspace $s_0 + \mathcal{K}^\ell(PA, Pr_0)$ where the basis for the Krylov subspace $\mathcal{K}^\ell(PA, Pr_0)$ is spanned by search directions.

Gould, Lucidi, Roma and Toint (1999) investigates the way of continuing the process that approximates the solution by using the generated Krylov subspace also when the trust region boundary is met. They proposed the Generalized Lanczos Trust-Region method (GLTR) which searches for the approximate solution from the same subspace $s_0+\mathcal{K}^\ell(PA, Pr_0)$ where the basis for the Krylov subspace $\mathcal{K}^\ell(PA, Pr_0)$ is formed by the preconditioned Lanczos vectors $\{Pv_1, \ldots, Pv_\ell\}$. Using the matrix $Z_l = [Pv_1, \ldots, Pv_\ell]$ the approximate solution is given by $s_\ell = Z_l y_l$ where $s_l$ solves the problem

$$\min_{s \in s_0+\mathcal{K}^\ell(PA,Pr_0)} \langle s, r_0\rangle + \frac{1}{2}\langle s, As\rangle \tag{6.5}$$

$$\text{subject to } \|s\|_{P^{-1}} \leq \Delta \tag{6.6}$$

which can be written as (Conn, Gould and Toint, 2000, p. 222)

$$\min_{y\in\mathbb{R}^{\ell+1}} \langle y, \beta_0 e_1\rangle + \frac{1}{2}\langle y, T_\ell y\rangle \tag{6.7}$$

$$\text{subject to } \|y\|_2 \leq \Delta, \tag{6.8}$$

where $T_l = Z_l^T A Z_l$ is the tridiagonal projected matrix. The solution is then found from the relation $s_\ell = Z_l y_l$. The GLTR Algorithm (Conn, Gould and Toint, 2000, p. 228) is similar to the Steihaug-Toint truncated conjugate gradient algorithm inside the trust region and when the boundary is met or $A$ is indefinite, it solves the problem (6.7) with the constraint (6.8).

Hence, when using the GLTR Algorithm, inside the trust region the Steihaug-Toint truncated conjugate gradient algorithms described in Section 6.1 can be used and when the boundary is met or $A$ is indefinite the Lanczos algorithms (Algorithm 2.6 and Algorithm 5.9) can used to solve problem (6.7) with the constraint (6.8).

When there is a trouble with a preconditioner that is, when it fails to be symmetric and positive definite in dual space, Algorithm 6.5 has to be used. Note that within this algorithm one can use also the augmented RPLanczos algorithm to compute the step beyond the Cauchy step.

## 6.3 Numerical experiments

In this section we provide a brief description of the academic test problem introduced in (Gratton and Tshimanga, 2009) based on heat equation and considered as an idealized data assimilation system. The numerical experiments are performed using this test problem, and are designed to analyze the effect of second-level preconditioners in dual space when used for slowly varying linear systems in a Gauss-Newton method, made globally convergent with a Steihaug-Toint truncated CG method.

### 6.3.1 The test problem

This test problem consists in finding the best initial temperature distribution using *a priori* initial temperature distribution, $x_c$, and a given set of approximate temperature distributions, $y(t_j)$, at different times $t_j$. The evolution model (dynamical model) is considered to be the nonlinear heat equation defined by (Gratton and Tshimanga, 2009)

$$\frac{\partial x}{\partial t} - \frac{\partial^2 x}{\partial u^2} - \frac{\partial^2 x}{\partial v^2} + f[x] = 0 \text{ in } \Omega \times (0, \infty) \tag{6.9}$$

$$x[u, v, t] = 0 \text{ on } \partial\Omega \times (0, \infty) \tag{6.10}$$

where the temperature variable $x[u, v, t]$ depends both on time $t$ and position given by spatial coordinates $u$ and $v$. The function $f[x]$ is defined by

$$f[x] = \exp[\eta x]. \tag{6.11}$$

An uniform 5-point finite difference scheme with $n = 14 \times 14 = 196$ nodes, and consequently, a space step length $h = 1/33$ is used. The implicit Euler time scheme is used for the integration over time with a time step length $\tau = 2 \cdot 10^{-4}$.

Experimental data ($x_c \in \mathbb{R}^n$ and $y(t_j) \in \mathbb{R}^{m_j}$) are produced using twin experiments, where the data are perturbed according the given covariance diagonal matrices $B \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ where $m = \sum_{j=1}^{N} m_j$ (for details we refer to (Gratton and Tshimanga, 2009)).

The prediction model $\mathcal{H}_j$ that maps the model fields to the observed data space is chosen as a restriction operator which selects 1 point every 16 points on the discretization grid and applies the operator C which is the diagonal matrix of the eigenvalues of the 5-point discrete Laplacian of order $m_j$.

We assume that observations are available at times $t_j = j \cdot \tau$ with $j = 1, \ldots, 4$. Then, we have $m = 4 \times m_j = 64$ and $n = 196$, i.e $m < n$.

### 6.3.2 Results

We run experiments for different values of $\eta$ given in the source term (6.11). The values of $\eta$ affects the degree of nonlinearity and accordingly the changes along a sequence of linear systems (2.16). As discussed in Chapter 4 depending on the change

along the sequence, a CG method in dual space may be in trouble since the inherited preconditioner may not be symmetric with respect to the $HBH^T$-inner product.

We first show the numerical results for the primal and dual approach when a Gauss-Newton method is used without a trust-region strategy to solve a sequence of varying systems (2.16) for different values of $\eta$. For the primal approach, PCG Algorithm 2.5 is applied on the linear system (2.16), where the preconditioning is achieved by the quasi-Newton LMP (3.37). In order to see the effect of second-level preconditioning we also run experiments with $P_k = B$. For the dual approach, Algorithm 4.11 is used where the preconditioning is achieved by the formula (4.82) in which the LMP $D$ is constructed from the quasi-Newton LMP (4.51).

For all experiments, the maximum number of outer iterations is set to 4 and the maximum number of inner iterations is set to 10, which somehow reflects a common practice in data assimilation (Tshimanga, Gratton, Weaver and Sartenaer, 2008). To construct the quasi-Newton LMP, all the pairs inherited from the previous iteration are used.

We present these experiments by Figure 3.1, 3.2 and 3.3 for $\eta = 1$, $\eta = 2$, and $\eta = 3$ respectively. First of all from the experiments perfomed with the primal approach, we can see the effect of second-level preconditioning. When $\eta = 1$ and $\eta = 2$ applying second-level preconditioner in dual space gives very similar results to that of primal space. When $\eta = 3$, augmented RPCG has trouble because of the loss of symmetry of the preconditioner with respect to the $HBH^T$-inner product.

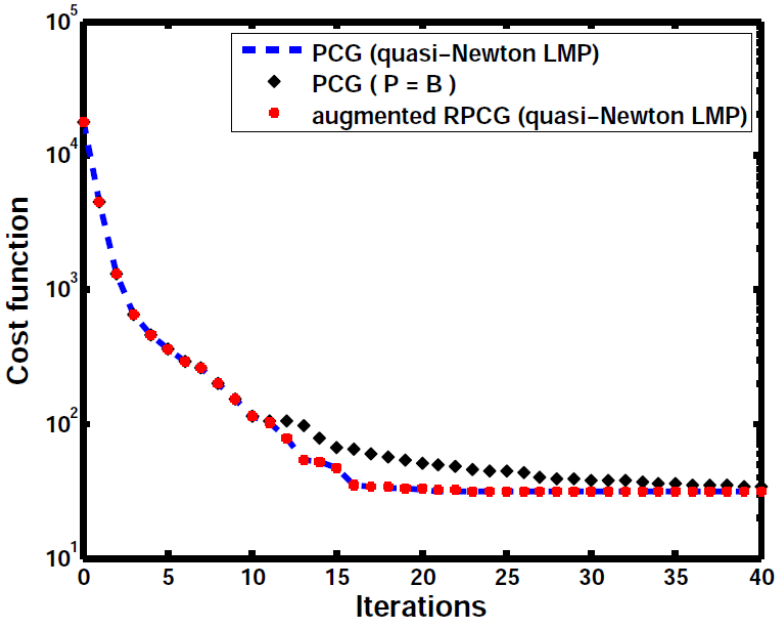


Figure 3.1: The quadratic cost function (2.13) values versus the inner iterations for $\eta = 1$.

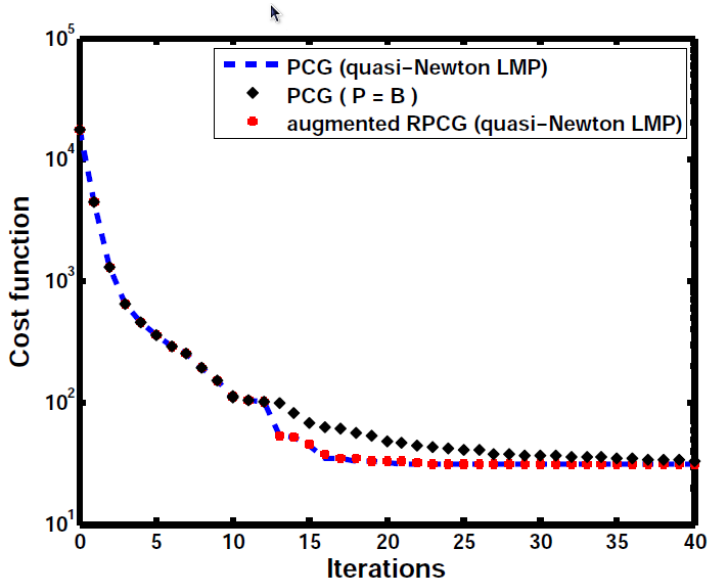


Figure 3.2: The quadratic cost function (2.13) values versus the inner iterations for $\eta = 2$.

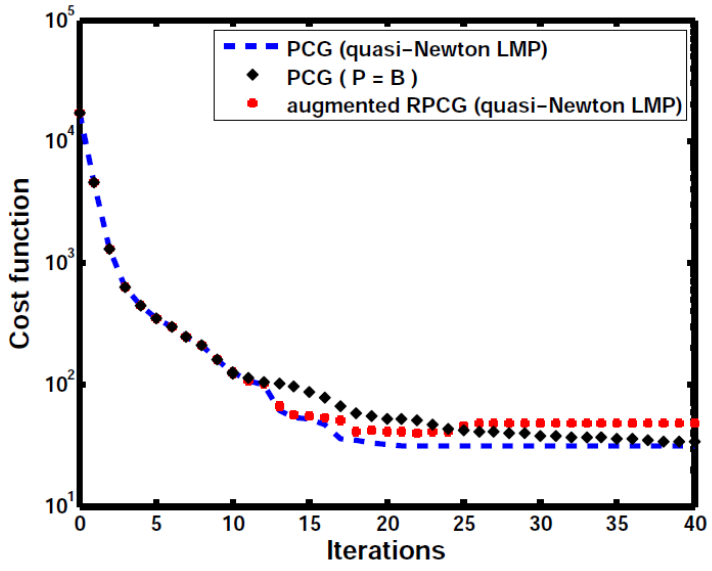


Figure 3.3: The quadratic cost function (2.13) values versus the inner iterations for $\eta = 3$.

We next present the results for $\eta = 3$ when the Gauss-Newton method is used with a trust-region strategy by Figure 3.4.

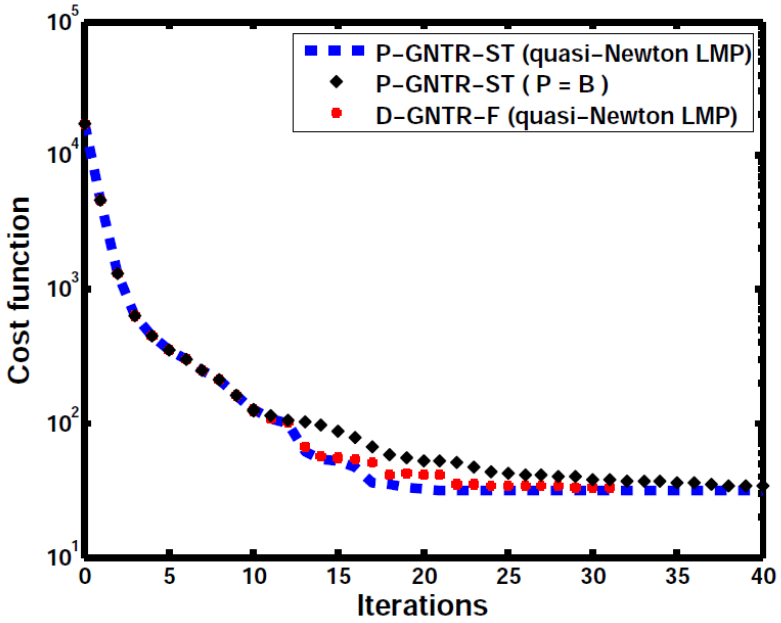


Figure 3.4: The quadratic cost function (2.13) values versus the inner iterations for $\eta = 3$ when the Gauss-Newton method is used with a trust-region strategy.

In Figure 3.4, **P-GNTR-ST** refers the primal approach which solves a sequence of slowly varying systems (2.16) with the Steihaug-Toint truncated CG algorithm (Conn, Gould and Toint, 2000, p. 205). **D-GNTR-F** refers the dual approach which solves a sequence of slowly varying systems (2.16) with the flexible trust-region Algorithm 6.5 in which the subproblem is solved by the flexible truncated CG algorithm 6.4 in dual space.

As it can be seen from Figure 3.4 that when $\eta = 3$, using Algorithm 6.5 the global convergence can be ensured for the dual approach. Algorithm 6.5 accepts the step that is calculated by Algorithm 6.4 for the first two outer loops, and the third outer loop the algorithm only accepts the Cauchy point. Note that this algorithm requires an additional function evaluation at the Cauchy point for each outer loop.

With the numerical experiments presented in this section, we see that the dual solver performs well on mildly nonlinear systems and can also be safely used in severe cases using the proposed flexible algorithm. This algorithm enables a global convergence disregarding the starting point of the iterations, at the cost of one function evaluation per outer loop. However the robustness that is gained with this algorithm is especially important for operational systems.

# CHAPTER 7

## Application to variational ocean data assimilation

Data assimilation is a key technique to improve the estimation of physical parameters arising for instance in oceanography, atmospheric sciences and also land studies. In this thesis; one of the efficient solution approximation for these systems, four dimensional variational (4D-Var) data assimilation method, is considered with an application to ocean data assimilation systems.

Variational assimilation solves *a regularized nonlinear least-squares problem* for determining a model state that optimally fits both observational information and *a priori* information in the form of a model background state. The fit is quantified by a cost function that measures the sum of the weighted squared differences between the available information (observations and background state) and the corresponding model-predicted fields. The weights are defined by matrix operators that define the error statistics (inverse error covariance) of the information.

The main characteristic of the variational data assimilation problems arising in oceanography and meteorology is that these problems are nonlinear large-scale problems. Hence, variational assimilation is implemented using an iterative technique based on the incremental approach (Courtier *et al.*, 1994), which in optimization theory is known as a Truncated Gauss-Newton (TGN) method (Gratton *et al.*, 2007). This approach is also widely used in oceanographic applications (Weaver *et al.*, 2003; Moore *et al.*, 2011). In some applications one can also observe that $m \ll n$, where $m$ is the number of observations and $n$ is the dimension of the control variable.

There is a lot of interest in data assimilation community to develop efficient algorithms that accelerate the convergence of TGN while possibly reducing the computational and memory cost. In this chapter, we show that the algorithms proposed in this thesis contribute in developing such efficient algorithms for large-scale problems where $m \ll n$.

This chapter includes a recent paper that presents the numerical results obtained from ocean data assimilation systems where the inner minimization in a Gauss-Newton method is performed by RPCG and RPLanczos with first-level preconditioners.

This paper illustrates the benefits of RPCG and RPLanczos in the case of a single outer loop of the incremental algorithm using a first-level preconditioning by the *a priori* error-covariance matrix $B$. Extensions of this work to account for multiple outer-loop iterations and second-level preconditioners based on LMPs such as quasi-Newton and Ritz (Gratton *et al.*, 2011) will be described in a future publication.

RMetS
Royal Meteorological Society

# B-preconditioned minimization algorithms for variational data assimilation with the dual formulation

S. Gürol,[a]* A. T. Weaver,[b] A. M. Moore,[c] A. Piacentini,[a] H. G. Arango[d] and S. Gratton[a,e]

[a]*CERFACS, Toulouse, France*
[b]*CERFACS/SUC URA 1875, Toulouse, France*
[c]*Department of Ocean Sciences, University of California at Santa Cruz, CA, USA*
[d]*Institute of Marine and Coastal Sciences, Rutgers University of New Jersey, NJ, USA*
[e]*ENSEEIHT, Toulouse, France*
*Correspondence to: Selime Gürol, CERFACS, 42 avenue Coriolis, 31057 Toulouse, France.
E-mail: gurol@cerfacs.fr

**Variational data assimilation problems in meteorology and oceanography require the solution of a regularized nonlinear least-squares problem. Practical solution algorithms are based on the incremental (truncated Gauss–Newton) approach, which involves the iterative solution of a sequence of linear least-squares (quadratic minimization) sub-problems. Each sub-problem can be solved using a primal approach, where the minimization is performed in a space spanned by vectors of the size of the model control vector, or a dual approach, where the minimization is performed in a space spanned by vectors of the size of the observation vector. The dual formulation can be advantageous for two reasons. First, the dimension of the minimization problem with the dual formulation does not increase when additional control variables are considered, such as those accounting for model error in a weak-constraint formulation. Second, whenever the dimension of observation space is significantly smaller than that of the model control space, the dual formulation can reduce both memory usage and computational cost.**

**In this article, a new dual-based algorithm called Restricted B-preconditioned Lanczos (RBLanczos) is introduced, where B denotes the background-error covariance matrix. RBLanczos is the Lanczos formulation of the Restricted B-preconditioned Conjugate Gradient (RBCG) method. RBLanczos generates mathematically equivalent iterates to those of RBCG and the corresponding B-preconditioned Conjugate Gradient and Lanczos algorithms used in the primal approach. All these algorithms can be implemented without the need for a square-root factorization of B. RBCG and RBLanczos, as well as the corresponding primal algorithms, are implemented in two operational ocean data assimilation systems and numerical results are presented. Practical diagnostic formulae for monitoring the convergence properties of the minimization are also presented. Copyright © 2013 Royal Meteorological Society**

*Key Words:* 3D-Var; 4D-Var; PSAS; conjugate gradient method; Lanczos method; dual approach; ocean data assimilation

*Received 4 November 2012; Revised 29 January 2013; Accepted 10 March 2013; Published online in Wiley Online Library*

*Citation:* Gürol S, Weaver AT, Moore AM, Piacentini A, Arango HG, Gratton S. 2013. B-preconditioned minimization algorithms for variational data assimilation with the dual formulation. *Q. J. R. Meteorol. Soc.* DOI:10.1002/qj.2150

## 1. Introduction

Variational assimilation seeks to solve a regularized nonlinear least-squares problem to determine a model state that optimally fits both observational information and *a priori* information in the form of a model background state. The fit is quantified by a cost function that measures the sum of the weighted squared differences between the available information (observations and background state) and the corresponding model-predicted fields. The weights are defined by matrix operators that define the error statistics (inverse error covariance) of the information.

The basic problem involves the optimization of a set of $n$ control variables given $m$ physical observations and a background estimate for each control variable. Iterative techniques must be used to identify an approximate minimum of the cost function when $n$ is large. In meteorological applications, variational assimilation is implemented using an iterative technique based on the incremental approach (Courtier *et al.*, 1994), which in optimization theory is known as a Truncated Gauss–Newton (TGN) method (Lawless *et al.*, 2005; Gratton *et al.*, 2007). This approach is also widely used in oceanographic applications (Weaver *et al.*, 2003; Moore *et al.*, 2011a).

The incremental approach solves a sequence of linear least-squares (quadratic minimization) problems where each member of the sequence is a local quadratic approximation of the original nonlinear least-squares problem. Conjugate Gradient (CG) or Lanczos methods, which belong to the general class of Krylov subspace methods, are effective for solving quadratic minimization problems when $n$ is large and when the system (Hessian) matrix is symmetric and positive definite, and only available in operator form (i.e. as a matrix-vector product). When the quadratic minimization is performed directly in $\mathbb{R}^n$, the method is referred to as the *primal approach*. Alternatively, the solution can be found using the so-called *dual approach* (Egbert *et al.*, 1994; Da Silva *et al.*, 1995; Courtier, 1997; Cohn *et al.*, 1998; Daley and Barker, 2001; Bennett, 2002) which performs the quadratic minimization in $\mathbb{R}^m$. The solution in $\mathbb{R}^m$ is then mapped to $\mathbb{R}^n$ through the application of an $n \times m$ matrix operator that defines the background-error covariances of the model-equivalent of the observations with the control variables.

The dual approach can provide a significant reduction in the computational cost and storage when $m \ll n$ since all the recurrence formulae in the minimization algorithm involve $m$-dimensional vectors instead of $n$-dimensional vectors as in the primal approach. For weak-constraint variational assimilation problems (Courtier, 1997; Bennett, 2002; Trémolet, 2006), $n$ can be very large since the control vector includes a time-sequence of corrective terms or model states in order to account for model error. For those problems, the primal approach may become intractable.

The dual approach of Courtier (1997) consists of solving the quadratic minimization problem in $\mathbb{R}^m$ with a first-level preconditioner given by the inverse of the observation-error covariance matrix ($\mathbf{R}^{-1}$). The preconditioner was applied using a factorized form involving $\mathbf{R}^{-1/2}$. The method is commonly referred to as PSAS (Physical-space Statistical Analysis System), where the dual approach was first used for operational meteorological applications (Da Silva *et al.*, 1995; Cohn *et al.*, 1998). However, it has been shown by El Akkraoui *et al.* (2008), Gratton and Tshimanga (2009) (hereafter referred to as GT09) and El Akkraoui and Gauthier (2010) that this version of the dual approach does not generate corresponding $n$-dimensional iterates that reduce monotonically the quadratic cost function in $\mathbb{R}^n$. A prohibitively large number of iterations may then be required to obtain an acceptable solution, which should be measured by the reduction of the cost function in primal space (El Akkraoui and Gauthier, 2010). If the minimization is terminated after a limited number of iterations, it can even yield a result that is inferior to the initial guess. El Akkraoui and Gauthier (2010) demonstrate experimentally that this problem can be alleviated by minimizing the ($\mathbf{R}^{-1}$-preconditioned) cost function in $\mathbb{R}^m$ using the MINimum RESidual (MINRES) method, although no mathematical proof was provided to support this result.

Another dual algorithm known as the Restricted Preconditioned Conjugate Gradient (RPCG) method has been proposed by GT09. This method again performs the minimization in $\mathbb{R}^m$, but contrary to PSAS, it generates mathematically equivalent iterates to those of the primal approach in which the cost function is minimized using a CG method. This allows the dual approach to benefit from the computational savings when $m < n$, while preserving the desired convergence properties of the primal approach.

In this article, we derive a new dual algorithm called the Restricted **B**-preconditioned Lanczos method (RBLanczos) where **B** is the background-error covariance matrix. RBLanczos generates mathematically identical iterates to the **B**-preconditioned Lanczos algorithm often used in primal approaches. This algorithm can also be interpreted as the Lanczos version of a special case of RPCG in which the corresponding primal first-level preconditioner is **B**. We call this specific algorithm Restricted **B**-preconditioned Conjugate Gradient (RBCG). The Lanczos vectors, which are directly computed by RBLanczos, are important for preconditioning (Tshimanga *et al.*, 2008; Fisher *et al.*, 2009; Desroziers and Berre, 2012) and for quantifying the performance of the data assimilation system (Cardinali *et al.*, 2004; Gelaro and Zhu, 2009; Moore *et al.*, 2011c).

A practical demonstration of the benefits of RBCG and RBLanczos is provided using two operational variational data assimilation systems for the ocean. RBCG is implemented in a three-dimensional variational assimilation (3D-Var) system for a global configuration of the NEMO (Nucleus for European Modelling of the Ocean) model. This system, called NEMOVAR, is used for operational monthly/seasonal forecasting and ocean reanalysis at the European Centre for Medium Range Weather Forecasts (ECMWF) (Mogensen *et al.*, 2012; Balmaseda *et al.*, 2013). RBLanczos is implemented in a four-dimensional variational assimilation (4D-Var) system for a California Current configuration of the ROMS (Regional Ocean Modeling Systems) model. This system, which includes a weak-constraint formulation of 4D-Var, is used for regional ocean forecasting and reanalysis applications (Moore *et al.*, 2011a,b,c).

The outline of the article is as follows. In section 2 the variational assimilation problem is formulated and the Lanczos algorithm for the primal approach is introduced. The RBLanczos algorithm is given in the same section. Practical formulae for diagnosing quantities needed for monitoring the convergence of the algorithm are also derived. Numerical experiments with NEMOVAR and ROMS are presented in section 3. Conclusions and

future directions are given in section 4. Appendix A provides simplified versions of the BLanczos and RBLanczos algorithms that do not include re-orthogonalization. Appendix B provides a derivation of the relations between the vectors defined in the primal and dual formulations of the Lanczos algorithm.

## 2. Lanczos method with the primal and dual approaches

### 2.1. Problem formulation

In its standard formulation, 4D-Var is designed to estimate the initial state of a dynamical system by combining observations over a given time window with an *a priori* estimate of the initial conditions called the background state. This approach has its origin in maximum likelihood estimation under a Gaussian assumption (Tarantola, 2005, pp 24–32). It leads to the minimization of the cost function

$$\mathcal{J}[\mathbf{u}] = \frac{1}{2}(\mathbf{u}-\mathbf{u}_\mathrm{b})^\mathrm{T}\mathbf{B}^{-1}(\mathbf{u}-\mathbf{u}_\mathrm{b}) + \frac{1}{2}\sum_{j=0}^{N_t}\{H_j(\mathbf{x}_j)-\mathbf{y}_j\}^\mathrm{T}\mathbf{R}_j^{-1}\{H_j(\mathbf{x}_j)-\mathbf{y}_j\} \quad (1)$$

with respect to a control vector $\mathbf{u}=\mathbf{x}(t_0)$, chosen to be the initial state of the dynamical model at time $t_0$. In this formulation, observations are given by an $m_j$-dimensional vector $\mathbf{y}_j$ at time $t_j$, and the background state is given by an $n$-dimensional vector $\mathbf{u}_\mathrm{b}=\mathbf{x}_\mathrm{b}(t_0)$. The $n\times n$ matrix $\mathbf{B}$ is an estimate of the background-error covariance matrix, and the $m_j\times m_j$ matrix $\mathbf{R}_j$ is an estimate of the observation-error covariance matrix at time $t_j$, with the observation errors assumed here to be uncorrelated in time. The inverse of these matrices defines the weighting matrix of the quadratic terms in (1). In order to calculate the model counterpart of the observation vector at $t_j$, first the state $\mathbf{x}_j=\mathbf{x}(t_j)=M_{0,j}\{\mathbf{x}(t_0)\}$ is estimated by propagating the initial state to time $t_j$ using the dynamical model operator $M_{0,j}\equiv M(t_0,t_j)$, and then the state is mapped to observation space using the observation operator $H_j$. Note that this formulation of 4D-Var assumes that the model operator $M_{0,j}$ is perfect. Adopting the terminology of Sasaki (1970), it is referred to as strong-constraint 4D-Var.

The incremental approach (Courtier *et al.*, 1994) is a practical algorithm for solving the 4D-Var problem when the system is weakly nonlinear and of large dimension $n$. Incremental 4D-Var consists of solving a sequence of linear least-squares approximations of the nonlinear least-squares problem (1). On each iteration $(k)$, a quadratic cost function

$$J[\delta\mathbf{u}^{(k)}] = \frac{1}{2}\left(\mathbf{u}^{(k-1)}-\mathbf{u}_\mathrm{b}+\delta\mathbf{u}^{(k)}\right)^\mathrm{T}\mathbf{B}^{-1}\left(\mathbf{u}^{(k-1)}-\mathbf{u}_\mathrm{b}+\delta\mathbf{u}^{(k)}\right) + \frac{1}{2}\sum_{j=0}^{N_t}\left(\mathbf{H}_j^{(k-1)}\mathbf{M}_{0,j}^{(k-1)}\delta\mathbf{u}^{(k)}-\mathbf{d}_j^{(k-1)}\right)^\mathrm{T} \times\mathbf{R}_j^{-1}\left(\mathbf{H}_j^{(k-1)}\mathbf{M}_{0,j}^{(k-1)}\delta\mathbf{u}^{(k)}-\mathbf{d}_j^{(k-1)}\right) \quad (2)$$

is minimized to determine a correction (increment) $\delta\mathbf{u}^{(k)}=\delta\mathbf{x}^{(k)}(t_0)$ to the initial state estimate $\mathbf{u}^{(k-1)}=\mathbf{x}^{(k-1)}(t_0)$, such that the updated estimate is

$$\mathbf{u}^{(k)}=\mathbf{u}^{(k-1)}+\delta\mathbf{u}^{(k)}.$$

The initial estimate is usually taken to be the background state, $\mathbf{u}^{(0)}=\mathbf{u}_\mathrm{b}$. In the quadratic formulation (2), $\mathbf{M}_{0,j}^{(k-1)}=\mathbf{M}^{(k-1)}(t_0,t_j)$ is the tangent-linear (TL) of the nonlinear model $M_{0,j}$ defined with respect to the time sequence of reference states $\left\{\mathbf{x}_i^{(k-1)}\right\}$, $i=1,\ldots,j$, and $\mathbf{H}_j^{(k-1)}$ is the TL of the observation operator $H_j^{(k)}$ defined with respect to $\mathbf{x}_j^{(k-1)}$. In practice, the TL operators are often approximated. The vector $\mathbf{d}_j^{(k-1)}=\mathbf{y}_j-H_j(\mathbf{x}_j^{(k-1)})$ is the difference between the observation vector and the corresponding model-predicted values at time $t_j$. The main loop of the incremental 4D-Var algorithm that generates the time sequence of states $\left\{\mathbf{x}_j^{(k-1)}\right\}$ and difference vectors $\left\{\mathbf{d}_j^{(k-1)}\right\}$, for $j=1,\ldots,N_t$, is called the *outer loop*. For large problems, an iterative method is used to solve the quadratic minimization problem (2). The iterative loop of the quadratic minimization problem is called the *inner loop* since it is nested within the outer loop.

This article focuses on minimization algorithms for the inner loop. For clarity and without loss of generality, we consider a single outer iteration ($k=1$) starting from the background state $\mathbf{u}_\mathrm{b}$. For this special case, the quadratic cost function of the inner-loop problem can be written as

$$J[\delta\mathbf{u}] = \underbrace{\frac{1}{2}\delta\mathbf{u}^\mathrm{T}\mathbf{B}^{-1}\delta\mathbf{u}}_{J_\mathrm{b}} + \underbrace{\frac{1}{2}(\mathbf{G}\,\delta\mathbf{u}-\mathbf{d})^\mathrm{T}\mathbf{R}^{-1}(\mathbf{G}\,\delta\mathbf{u}-\mathbf{d})}_{J_\mathrm{o}}, \quad (3)$$

which is a compact form of the linearized problem (2) with $k=1$ and $\mathbf{u}^{(0)}=\mathbf{u}_\mathrm{b}$. The underbraces highlight the background term $J_\mathrm{b}$ and observation term $J_\mathrm{o}$. The outer iteration counter $(k)$ has been dropped for clarity of notation. In (3), the generalized observation operator $\mathbf{G}$ is a $m\times n$ matrix of concatenated operators $\mathbf{H}_j\mathbf{M}_{0,j}$ over time where $m=\sum_{j=0}^{N_t}m_j$, $\mathbf{R}$ is a $m\times m$ block-diagonal matrix whose $j$th block is $\mathbf{R}_j$, and $\mathbf{d}$ is the vector of concatenated difference vectors $\mathbf{d}_j$.

The exact solution, $\delta\mathbf{u}^*=\mathrm{argmin}\,J[\delta\mathbf{u}]$, is obtained by setting the gradient of the cost function to zero which yields

$$\delta\mathbf{u}^*=(\mathbf{B}^{-1}+\mathbf{G}^\mathrm{T}\mathbf{R}^{-1}\mathbf{G})^{-1}\,\mathbf{G}^\mathrm{T}\mathbf{R}^{-1}\mathbf{d}. \quad (4)$$

Since the matrices appearing in (4) are large and typically only available in operator form (i.e. as a matrix-vector product), an approximate solution is usually found by solving the $n\times n$ linear system

$$(\mathbf{B}^{-1}+\mathbf{G}^\mathrm{T}\mathbf{R}^{-1}\mathbf{G})\,\delta\mathbf{u}=\mathbf{G}^\mathrm{T}\mathbf{R}^{-1}\mathbf{d}, \quad (5)$$

using a Krylov subspace iterative method (Saad, 1996, Chapter 6). Equation (5), which involves directly optimizing in control space to determine $\delta\mathbf{u}$, is referred to as the *primal* problem.

Alternatively, the solution (4) can be expressed as (Courtier, 1997)

$$\delta\mathbf{u}^*=\mathbf{B}\mathbf{G}^\mathrm{T}(\mathbf{G}\mathbf{B}\mathbf{G}^\mathrm{T}+\mathbf{R})^{-1}\mathbf{d}, \quad (6)$$

by using the Sherman–Morrison–Woodbury formula (Nocedal and Wright, 2006, pp 612–613) or from duality theory (Gratton *et al.*, 2013).

An approximate solution of (6) can be obtained by applying a Krylov subspace method to the $m \times m$ linear system

$$(\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}} + \mathbf{R})\boldsymbol{\lambda} = \mathbf{d}, \tag{7}$$

and then transforming the solution as

$$\delta\mathbf{u} = \mathbf{B}\mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda}. \tag{8}$$

In ocean data assimilation, this approach has been referred to as the indirect representer method (Egbert *et al.*, 1994) whereas in meteorological data assimilation it has been called PSAS (Da Silva *et al.*, 1995; Courtier, 1997; Cohn *et al.*, 1998). Equations (7) and (8), which involve optimizing the $m$-dimensional vector $\boldsymbol{\lambda}$, constitute the *dual* problem. The dual problem can be preferable to the primal problem for computational reasons when the dimension ($m$) of observation space is much smaller than the dimension ($n$) of control space.

For some problems it may be desirable to include additional variables in the control vector $\mathbf{u}$. For example, extra control variables can be included to account for errors in the dynamical model $M_{0,j}$, leading to the so-called weak-constraint formulation of 4D-Var (Sasaki, 1970), or to account for errors in boundary conditions. The dimension of the primal problem is determined by the size of the control vector. In contrast, the dimension of the dual problem is determined by the size of the observation vector and thus does not change by including additional control variables. For weak-constraint 4D-Var, the number of extra variables can be so large that the primal problem becomes impractical. In such cases, the dual problem is particularly appealing.

### 2.2. *Solving the linearized problem: primal approaches*

Equation (5) is a problem of the standard form

$$\mathbf{A}\,\delta\mathbf{u} = \mathbf{b},$$

where

$$\mathbf{A} = \nabla^2 J = \mathbf{B}^{-1} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G} \tag{9}$$

is the (symmetric and positive-definite) Hessian or system matrix, and

$$\mathbf{b} = -\nabla J[\mathbf{0}] = \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d} \tag{10}$$

is the negative of the cost function gradient evaluated at $\delta\mathbf{u} = \mathbf{0}$. Krylov subspace methods search for an approximate solution $\delta\mathbf{u}_l$ in a subspace $\delta\mathbf{u}_0 + \mathcal{K}^l(\mathbf{A}, \mathbf{r}_0)$ where $\delta\mathbf{u}_0$ is the initial guess,

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\delta\mathbf{u}_0 = -\nabla J[\delta\mathbf{u}_0] \tag{11}$$

is the initial residual (the negative of the gradient evaluated at $\delta\mathbf{u} = \delta\mathbf{u}_0$), and

$$\mathcal{K}^l(\mathbf{A}, \mathbf{r}_0) = \mathrm{span}\left\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{l-1}\mathbf{r}_0\right\}$$

is the Krylov subspace of dimension $l$.

To define the $l$th iterate uniquely, several Krylov subspace methods impose the Petrov–Galerkin condition (Saad, 1996, p 144)

$$\mathbf{r}_l \perp \mathcal{L}^l(\mathbf{A}, \mathbf{r}_0), \tag{12}$$

where $\mathbf{r}_l$ is the residual of the $l$th iterate, and $\mathcal{L}^l(\mathbf{A}, \mathbf{r}_0)$ is an $l$-dimensional subspace. The choice of the subspace $\mathcal{L}^l(\mathbf{A}, \mathbf{r}_0)$ and properties of the matrix $\mathbf{A}$ yield different Krylov subspace methods. For instance, when $\mathcal{L}^l(\mathbf{A}, \mathbf{r}_0) = \mathcal{K}^l(\mathbf{A}, \mathbf{r}_0)$, the condition (12) is called a Galerkin condition. It leads to the Lanczos method when $\mathbf{A}$ is symmetric, which finds an approximate solution $\delta\mathbf{u}_l$ by minimizing the $\mathbf{A}^{-1}$-norm of the residual; $\|\mathbf{r}_l\|_{\mathbf{A}^{-1}} = \sqrt{\mathbf{r}_l^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{r}_l}$. On the other hand, choosing $\mathcal{L}^l(\mathbf{A}, \mathbf{r}_0) = \mathbf{A}\,\mathcal{K}^l(\mathbf{A}, \mathbf{r}_0)$ with $\mathbf{A}$ symmetric gives the Minimum RESidual method (MINRES) (Paige and Saunders, 1975). It finds the approximate solution $\delta\mathbf{u}_l$ by minimizing the Euclidean norm of the residual; $\|\mathbf{r}_l\|_2 = \sqrt{\mathbf{r}_l^{\mathrm{T}}\mathbf{r}_l}$. Furthermore, if $\mathbf{A}$ is symmetric and positive definite then the Lanczos method is mathematically equivalent to the CG method (Saad, 1996, p 176).

This article focuses on the Lanczos method which is widely used in meteorological and ocean variational data assimilation (Tshimanga *et al.*, 2008; Fisher *et al.*, 2009; Moore *et al.*, 2011a). As mentioned above, the Lanczos method imposes the Galerkin condition which implies

$$\mathbf{V}_l^{\mathrm{T}}(\mathbf{b} - \mathbf{A}\delta\mathbf{u}_l) = 0\,,$$

where $\mathbf{V}_l = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_l]$ is an $n \times l$ matrix whose column vectors $\mathbf{v}_i$, $i = 1, \ldots, l$, form an orthonormal basis for $\mathcal{K}^l(\mathbf{A}, \mathbf{r}_0)$. The basis vectors are known as Lanczos vectors and are constructed by using a simplified version of Arnoldi's algorithm when $\mathbf{A}$ is symmetric (Saad, 1996, pp 174–175). They can be shown to be related to the normalized residual (gradient) vectors generated by CG on the same iterations (Paige and Saunders, 1975). Using this orthonormal basis, the solution on the $l$th iteration becomes

$$\delta\mathbf{u}_l = \delta\mathbf{u}_0 + \mathbf{V}_l\mathbf{s}_l\,, \tag{13}$$

where $\mathbf{s}_l$ is the solution of the linear system

$$\mathbf{T}_l\mathbf{s}_l \;=\; \mathbf{V}_l^{\mathrm{T}}\mathbf{r}_0 \tag{14}$$

involving the $l \times l$ tridiagonal matrix

$$\mathbf{T}_l \;=\; \mathbf{V}_l^{\mathrm{T}}\mathbf{A}\mathbf{V}_l\,. \tag{15}$$

The tridiagonal system (14) which is of size $l$, $l$ being typically small, can be easily solved using standard factorization methods (Golub and Van Loan, 1996, pp 138–139). The eigenpairs of the matrix $\mathbf{A}$ can be approximated by the Ritz pairs, $(\theta_i, \mathbf{V}_l\boldsymbol{\xi}_i)$, $i = 1, \ldots, l$, defined with respect to the subspace $\mathcal{K}^l(\mathbf{A}, \mathbf{r}_0)$, where $(\theta_i, \boldsymbol{\xi}_i)$ are the eigenpairs of $\mathbf{T}_l$ (Parlett, 1980; Golub and Van Loan, 1996; Saad, 1996). The right-hand side of (14) can be simplified by recalling that the vectors $\mathbf{v}_i$, $i = 1, \ldots, l$, form an orthonormal basis for $\mathcal{K}^l(\mathbf{A}, \mathbf{r}_0)$ and that $\mathbf{v}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|_2$, so that

$$\mathbf{V}_l^{\mathrm{T}}\mathbf{r}_0 = \|\mathbf{r}_0\|_2\;\mathbf{e}_1\,, \tag{16}$$

where $\|\mathbf{r}_0\|_2 = \sqrt{\mathbf{r}_0^{\mathrm{T}}\mathbf{r}_0}$ and $\mathbf{e}_1 = [1, 0, \ldots, 0]^{\mathrm{T}}$.

When applying the Lanczos or CG method, a *preconditioner* is desirable to accelerate the convergence. A preconditioner typically transforms the linear system into one with 'better' spectral properties; e.g. more clustered eigenvalues. At the very least, a *first-level* preconditioner is needed to non-dimensionalize the control vector when it involves more than one physical variable. In this way, results obtained using the standard implementations of Lanczos or CG (with the canonical inner product) will be independent of the choice of physical units. This can be easily done by separating out the units using the factorization $\mathbf{B} = \mathbf{D}^{1/2}\mathbf{C}\mathbf{D}^{1/2}$ where $\mathbf{D}^{1/2}$ is a diagonal matrix of (dimensional) standard deviations, and $\mathbf{C}$ is a (dimensionless) correlation matrix, with 1s on the diagonal. Equation (5) can then be transformed as

$$\left(\mathbf{C}^{-1} + \mathbf{D}^{1/2}\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\,\mathbf{D}^{1/2}\right)\delta\widetilde{\mathbf{u}} = \mathbf{D}^{1/2}\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d} \qquad (17)$$

and solved for $\delta\widetilde{\mathbf{u}} = \mathbf{D}^{-1/2}\delta\mathbf{u}$ using Lanczos or CG equipped with the canonical inner product. The solution in control space is then obtained from $\delta\mathbf{u} = \mathbf{D}^{1/2}\delta\widetilde{\mathbf{u}}$.

Equation (17) is poorly conditioned in general since typical correlation matrices $\mathbf{C}$ used in variational data assimilation contain a wide range of eigenvalues (Lorenc, 1997). It also requires specification of the inverse correlation operator $\mathbf{C}^{-1}$ which is difficult in practice. To alleviate these problems, the full background-error covariance matrix $\mathbf{B}$ is used as a first-level preconditioner (Lorenc, 1988). Assuming $\mathbf{B}$ can be factored as $\mathbf{B} = \mathbf{U}\mathbf{U}^{\mathrm{T}}$, where $\mathbf{U}$ is a $n \times n'$ left square-root matrix of $\mathbf{B}$, with $n'$ possibly different from $n$, then the preconditioned system becomes

$$(\mathbf{I}_{n'} + \mathbf{U}^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\mathbf{U})\,\delta\widetilde{\mathbf{u}} = \mathbf{U}^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d}\,, \qquad (18)$$

where $\mathbf{I}_{n'}$ is the $n' \times n'$ identity matrix. Equation (18) is solved for $\delta\widetilde{\mathbf{u}}$ using Lanczos or CG formulated with the canonical inner product, and the solution in control space is obtained from $\delta\mathbf{u} = \mathbf{U}\,\delta\widetilde{\mathbf{u}}$. The eigenvalue spectrum of the system matrix in (18) is bounded below by 1, and has a cluster at 1 of size at least $\max(0, n' - m)$. A *second-level* preconditioner that approximates the inverse of the linear system matrix can also be used to further accelerate the convergence. Tshimanga *et al.* (2008) and Gratton *et al.* (2011b) discuss second-level preconditioning techniques within the context of multi-outer-loop iterations of incremental 4D-Var. However, in this article, only first-level preconditioning with $\mathbf{B}$ will be considered. Furthermore, the initial guess $\delta\mathbf{u}_0$ will be assumed to be zero, as is typically the case in applications of 3D-Var and 4D-Var that involve a single outer-loop iteration.

Notice that the dimension of minimization space for solving (18) is determined by $n'$, the dimension of $\delta\widetilde{\mathbf{u}}$. The case $n' < n$ arises with reduced-rank formulations of $\mathbf{B}$ where, for example, $\mathbf{U}$ is defined by a limited number of ensemble perturbations or some other appropriately chosen basis vectors (Evensen, 2009; Gratton *et al.*, 2011a). Square-root preconditioning is clearly advantageous when $n' \ll n$. The case $n' > n$ arises when $\mathbf{B}$ is formulated as a weighted sum of two or more matrices. This situation can arise with general covariance models constructed from spectral or grid-point filters (Fisher, 2003; Purser *et al.*, 2003; Weaver and Mirouze, 2013) and with *hybrid* ensemble-variational $\mathbf{B}$ formulations (Buehner, 2005; Wang *et al.*, 2007, 2008). Square-root preconditioning can be less convenient in these latter cases.

The preconditioned linear system (18) can be written in the standard form

$$\widetilde{\mathbf{A}}\,\delta\widetilde{\mathbf{u}} = \widetilde{\mathbf{b}}\,, \qquad (19)$$

where

$$\begin{aligned} \widetilde{\mathbf{A}} &= \mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{U} \\ \text{and} \quad \widetilde{\mathbf{b}} &= \mathbf{U}^{\mathrm{T}}\mathbf{b}\,, \end{aligned}$$

with $\mathbf{A}$ and $\mathbf{b}$ given by (9) and (10). The Lanczos algorithm applied to (19) searches for a solution in the Krylov subspace $\mathcal{K}^l(\widetilde{\mathbf{A}}, \widetilde{\mathbf{r}}_0)$, where $\widetilde{\mathbf{r}}_0 = \mathbf{U}^{\mathrm{T}}\mathbf{r}_0$. Applying the standard version of the Lanczos algorithm described by (13)–(15) to the system (19), with the assumption $\delta\mathbf{u}_0 = \mathbf{0}$, gives an approximate solution on the $l$th iteration as

$$\delta\widetilde{\mathbf{u}}_l = \widetilde{\mathbf{V}}_l\mathbf{s}_l\,, \qquad (20)$$

where

$$\left.\begin{aligned} \mathbf{T}_l\mathbf{s}_l &= \widetilde{\mathbf{V}}_l^{\mathrm{T}}\widetilde{\mathbf{r}}_0\,, \\ \text{and} \quad \mathbf{T}_l &= \widetilde{\mathbf{V}}_l^{\mathrm{T}}\widetilde{\mathbf{A}}\widetilde{\mathbf{V}}_l\,. \end{aligned}\right\} \qquad (21)$$

The Lanczos algorithm applied to the preconditioned linear system (19) is based on the availability of a factored form for $\mathbf{B}$ (Fisher *et al.*, 2009). Hereafter we use the term Lanczos to refer to this particular form of the Lanczos algorithm. Alternatively, $\mathbf{B}$-preconditioning of the linear system (5) can be achieved by employing $\mathbf{B}$ as a *right* symmetric preconditioner (Nour-Omid *et al.*, 1988; Axelsson, 1996; Chan *et al.*, 1999). This leads to

$$(\mathbf{I}_n + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\,\mathbf{B})\,\delta\overline{\mathbf{u}} \;=\; \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d}. \qquad (22)$$

The system matrix in (22) is symmetric (self-adjoint) with respect to the $\mathbf{B}$-inner product. The solution in control space itself is obtained from $\delta\mathbf{u} = \mathbf{B}\,\delta\overline{\mathbf{u}}$. This approach generates mathematically equivalent iterates to those of the Lanczos approach using $\mathbf{U}$ and $\mathbf{U}^{\mathrm{T}}$, but is more general since it does not require $\mathbf{B}$ to be factored, this factorization being particularly inconvenient when $n' \gg n$ as mentioned above.

The Lanczos algorithm that solves the linear system (22) with the $\mathbf{B}$-inner product is given in Algorithm 1. For future reference, we refer to it as BLanczos.

BLanczos searches for a solution in the Krylov subspace (Chan *et al.*, 1999)

$$\begin{aligned} &\mathcal{K}^l(\mathbf{B}\mathbf{A}, \mathbf{B}\mathbf{r}_0) \\ &\quad = \mathrm{span}\left\{\mathbf{B}\mathbf{r}_0, (\mathbf{B}\mathbf{A})\mathbf{B}\mathbf{r}_0, \ldots, (\mathbf{B}\mathbf{A})^{l-1}\mathbf{B}\mathbf{r}_0\right\}. \end{aligned} \qquad (23)$$

It can be easily shown that the matrix $\mathbf{V}_l$ constructed by BLanczos is related to the matrix $\widetilde{\mathbf{V}}_l$ via the relation $\widetilde{\mathbf{V}}_l = \mathbf{U}^{\mathrm{T}}\mathbf{V}_l$. From this relation, (20) and (21), and the definitions of $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{r}}_0$, it is straightforward to show that the solution on the $l$th iteration of BLanczos can be written as

$$\delta\mathbf{u}_l = \mathbf{Z}_l\mathbf{s}_l\,, \qquad (24)$$

where

$$\left.\begin{aligned} \mathbf{Z}_l &= \mathbf{B}\mathbf{V}_l \\ \mathbf{T}_l\,\mathbf{s}_l &= \mathbf{Z}_l^{\mathrm{T}}\mathbf{r}_0, \\ \text{and} \quad \mathbf{T}_l &= \mathbf{Z}_l^{\mathrm{T}}\mathbf{A}\mathbf{Z}_l. \end{aligned}\right\} \qquad (25)$$

**Algorithm 1:** BLanczos with re-orthogonalization

$\mathbf{v}_0 = \mathbf{0}$
$\mathbf{r}_0 = \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d}$
$\mathbf{t}_0 = \mathbf{B}\,\mathbf{r}_0$
$\beta_0 = \sqrt{\mathbf{t}_0^{\mathrm{T}}\mathbf{r}_0}$
$\mathbf{v}_1 = \mathbf{r}_0/\beta_0$
$\mathbf{z}_1 = \mathbf{t}_0/\beta_0$
$\beta_1 = 0$
$\mathbf{Z}_1 = [\,\mathbf{z}_1\,]$
$\mathbf{V}_1 = [\,\mathbf{v}_1\,]$
**for** $i = 1, 2, \ldots, l$ **do**
  $\mathbf{q}_i = (\mathbf{v}_i + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\,\mathbf{z}_i) - \beta_i\mathbf{v}_{i-1}$
  $\alpha_i = \mathbf{q}_i^{\mathrm{T}}\mathbf{z}_i$
  $\mathbf{w}_i = \mathbf{q}_i - \alpha_i\mathbf{v}_i$
  Re-orthogonalize $\mathbf{w}_i$ using $\mathbf{V}_i$ and $\mathbf{Z}_i$
  $\mathbf{t}_i = \mathbf{B}\,\mathbf{w}_i$
  $\beta_{i+1} = \sqrt{\mathbf{t}_i^{\mathrm{T}}\mathbf{w}_i}$
  $\mathbf{v}_{i+1} = \mathbf{w}_i/\beta_{i+1}$
  $\mathbf{z}_{i+1} = \mathbf{t}_i/\beta_{i+1}$
  $\mathbf{Z}_i := [\,\mathbf{Z}_i,\ \mathbf{z}_{i+1}\,]$
  $\mathbf{V}_i := [\,\mathbf{V}_i,\ \mathbf{v}_{i+1}\,]$
  $(\mathbf{T}_i)_{i,i} = \alpha_i$
  **if** $i > 1$ **then**
    $(\mathbf{T}_i)_{i-1,i} = (\mathbf{T}_i)_{i,i-1} = \beta_i$
  **end**
**end**
Solve $\mathbf{T}_l\mathbf{s}_l = \beta_0\,\mathbf{e}_1$
$\delta\mathbf{u}_l = \mathbf{Z}_l\,\mathbf{s}_l$

From the orthonormality of the column vectors of $\widetilde{\mathbf{V}}_l$, i.e. $\widetilde{\mathbf{V}}_l^{\mathrm{T}}\widetilde{\mathbf{V}}_l = \mathbf{I}_l$, and the relation $\widetilde{\mathbf{V}}_l = \mathbf{U}^{\mathrm{T}}\mathbf{V}_l$, we obtain that $\mathbf{V}_l^{\mathrm{T}}\mathbf{B}\mathbf{V}_l = \mathbf{I}_l$, i.e. that the column vectors of $\mathbf{V}_l$ are **B**-orthonormal. From the **B**-orthonormality of the vectors $\mathbf{v}_i$, $i = 1, \ldots, l$, and the fact that $\mathbf{v}_1 = \mathbf{r}_0/\beta_0$, where $\beta_0 = \|\mathbf{r}_0\|_{\mathbf{B}} = \sqrt{\mathbf{r}_0^{\mathrm{T}}\mathbf{B}\mathbf{r}_0}$, the right-hand side of the second expression in (25) can be simplified as

$$\mathbf{Z}_l^{\mathrm{T}}\mathbf{r}_0 = \beta_0\,\mathbf{e}_1.$$

Each iteration of BLanczos requires only one matrix-vector multiplication with each of the matrices **G**, $\mathbf{G}^{\mathrm{T}}$, $\mathbf{R}^{-1}$ and **B**. The $n \times l$ matrix $\mathbf{Z}_l$ and the two vectors consisting of the diagonal and off-diagonal entries of the $l \times l$ symmetric matrix $\mathbf{T}_l$ need to be stored for computing the solution with (24). As mentioned earlier, these matrices contain approximate information on the eigenspectrum of the Hessian matrix which is valuable for diagnostic studies and for building second-level preconditioners. The matrix $\mathbf{Z}_l$ is also required for re-orthogonalization with the matrix $\mathbf{V}_l$, as will be explained in section 2.5.

When re-orthogonalization is not necessary, the solution can also be found using recurrence relationships that do not require the matrices $\mathbf{Z}_l$ and $\mathbf{T}_l$ (Papadrakakis and Smerou, 1990; Saad, 1996; Chien and Chang, 2003). Therefore, the memory requirements can be reduced by using this version of the Lanczos algorithm (known also as the direct version of the Lanczos algorithm (Saad, 1996, p 177)) which is derived from the LU factorization of $\mathbf{T}_l$. This direct version of BLanczos is given in Appendix A.

As mentioned earlier, another well-known method for solving the linear system (5) is CG. CG is mathematically equivalent to the Lanczos method in exact arithmetic when the system matrix **A** is symmetric and positive definite. A **B**-preconditioned CG algorithm for solving the linear system (5), which we call BCG, is given in Algorithm 2. Like BLanczos, BCG only requires **B** for preconditioning (not its factorization or inverse) and only one application per iteration of the matrix operators **G**, $\mathbf{G}^{\mathrm{T}}$, $\mathbf{R}^{-1}$ and **B** (Derber and Rosati, 1989).

**Algorithm 2:** BCG

$\delta\mathbf{u}_0 = \mathbf{0}$
$\mathbf{f}_0 = \mathbf{0}$
$\mathbf{r}_0 = \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d}$
$\mathbf{z}_0 = \mathbf{B}\mathbf{r}_0$
$\mathbf{p}_0 = \mathbf{z}_0$
$\mathbf{h}_0 = \mathbf{r}_0$
**for** $i = 0, 1, \ldots, l-1$ **do**
  $\mathbf{q}_i = \mathbf{h}_i + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\mathbf{p}_i$
  $\alpha_i = \mathbf{r}_i^{\mathrm{T}}\mathbf{z}_i/\mathbf{q}_i^{\mathrm{T}}\mathbf{p}_i$
  $\delta\mathbf{u}_{i+1} = \delta\mathbf{u}_i + \alpha_i\mathbf{p}_i$
  $\mathbf{f}_{i+1} = \mathbf{f}_i + \alpha_i\mathbf{h}_i$ } for diagnosing $J_{\mathrm{b}}$
  $\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i\mathbf{q}_i$
  Re-orthogonalize $\mathbf{r}_{i+1}$
  $\mathbf{z}_{i+1} = \mathbf{B}\mathbf{r}_{i+1}$
  $\beta_i = \mathbf{r}_{i+1}^{\mathrm{T}}\mathbf{z}_{i+1}/\mathbf{r}_i^{\mathrm{T}}\mathbf{z}_i$
  $\mathbf{p}_{i+1} = \mathbf{z}_{i+1} + \beta_i\mathbf{p}_i$
  $\mathbf{h}_{i+1} = \mathbf{r}_{i+1} + \beta_i\mathbf{h}_i$
**end**

From the theoretical equivalence of BLanczos and BCG, it is possible to construct the tridiagonal matrix $\mathbf{T}_l$ from the coefficients $\alpha_i$ and $\beta_i$ generated by BCG (Saad, 1996, pp 181–182), and hence to obtain approximate eigenvalues of the Hessian matrix from the eigenvalues $\mathbf{T}_l$. In particular, the diagonal and off-diagonal entries of $\mathbf{T}_l$ can be obtained from the relations

$$(\mathbf{T}_l)_{i,i} = \begin{cases} \dfrac{1}{\alpha_{i-1}} & \text{if } i = 1, \\ \dfrac{1}{\alpha_{i-1}} + \dfrac{\beta_{i-2}}{\alpha_{i-2}} & \text{if } i > 1, \end{cases} \tag{26}$$

and

$$(\mathbf{T}_l)_{i+1,i} = (\mathbf{T}_l)_{i,i+1} = \frac{\sqrt{\beta_{i-1}}}{\alpha_{i-1}}. \tag{27}$$

Another way to avoid specifying a factorization of **B** is to treat (22) as a non-symmetric system (with respect to the canonical inner product) and to solve it with the Full Orthogonalization Method (FOM), which is a generalization of the Lanczos method that does not require **A** to be symmetric but is more computationally expensive. El Akkraoui *et al.* (2012) used a BiConjugate Gradient (BiCG) algorithm* (Saad, 1996, pp 211–212) to solve the non-symmetric system (22) with second-level preconditioning. BiCG is less robust numerically than PCG (Golub and Van Loan, 1996, p 551), although this problem can be handled by using the BiCG Stabilized method (Saad, 1996, p 217). When using only first-level (**B**)-preconditioning, El Akkraoui *et al.* (2012) show that BiCG is mathematically equivalent to BCG or 'double CG' as referred to in their article.

*BiCG is referred to as BCG in Saad (1996) and should not be confused with the BCG algorithm described here.

### 2.3. Solving the linearized problem: dual approaches

Alternatively, the solution of the linearized problem (3) can be found by solving the dual problem using a CG or Lanczos method. By analogy with the use of $\mathbf{B}$ as a preconditioner for the primal problem, one possibility is to use $\mathbf{R}^{-1}$ as a preconditioner for the dual problem. This is the basic approach taken in PSAS. For example, employing $\mathbf{R}^{-1}$ *left* symmetric preconditioning on (7) leads to solving the system

$$(\mathbf{R}^{-1}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}} + \mathbf{I}_m)\,\boldsymbol{\lambda} \;=\; \mathbf{R}^{-1}\mathbf{d} \tag{28}$$

using a CG or Lanczos algorithm formulated with the $\mathbf{R}$-inner product. In many data assimilation systems, $\mathbf{R}$ is assumed to be diagonal in which case applying $\mathbf{R}$ and $\mathbf{R}^{-1}$ is trivial. Conventional implementations of PSAS (Courtier, 1997; El Akkraoui *et al.*, 2008; El Akkraoui and Gauthier, 2010) employ $\mathbf{R}^{-1}$-preconditioning via a square-root matrix $\mathbf{R}^{-1/2}$, where $\mathbf{R}^{-1} = \mathbf{R}^{-1/2}\mathbf{R}^{-1/2}$ (assuming $\mathbf{R}$ is diagonal), and solve

$$(\mathbf{R}^{-1/2}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1/2} + \mathbf{I}_m)\,\widetilde{\boldsymbol{\lambda}} \;=\; \mathbf{R}^{-1/2}\mathbf{d}\,, \tag{29}$$

where $\boldsymbol{\lambda} = \mathbf{R}^{-1/2}\widetilde{\boldsymbol{\lambda}}$, using CG or Lanczos furnished with the canonical inner product.

As illustrated by GT09 and El Akkraoui and Gauthier (2010), PSAS has a non-monotonic convergence behaviour when viewed in terms of the reduction of the quadratic cost function (3). This can yield a quadratic cost value that is larger than the initial value when the minimization is terminated after a few iterations. As a remedy, El Akkraoui and Gauthier (2010) suggested the use of MINRES to solve the linear system (29). Their experimental results showed that MINRES could produce a monotonically decreasing quadratic cost function (3), although no mathematical theory was provided to guarantee this behaviour.

A better alternative is the Restricted Preconditioned Conjugate Gradient (RPCG) method (GT09) which also solves (28) using a CG method but equipped with the (possibly semi-definite) $\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}$-inner product instead of the $\mathbf{R}$-inner product. As discussed by GT09, RPCG generates, in exact arithmetic, the same iterates as those generated by PCG. This can also be achieved by solving (29) with the $\mathbf{R}^{-1/2}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1/2}$-inner product instead of the canonical inner product as in PSAS.

Since only first-level preconditioning with $\mathbf{B}$ is considered in this study, we restrict our attention to RBCG which is the dual equivalent of BCG and a special case of RPCG. For reference, the RBCG algorithm, which is based on Algorithm 5 of GT09, is provided in Algorithm 3. (GT09 and Gratton *et al.* (2013) provide a more general presentation of the algorithm that allows for second-level preconditioning.)

As with BCG, each loop of the RBCG algorithm requires one matrix-vector multiplication with $\mathbf{G}$, $\mathbf{G}^{\mathrm{T}}$, $\mathbf{R}^{-1}$ and $\mathbf{B}$. Note that, the tridiagonal matrix $\mathbf{T}_l$ can be generated using (26) and (27) since the coefficients $\alpha_i$ and $\beta_i$ are equivalent in BCG and RBCG.

In the same way that RBCG is the dual equivalent of BCG, there exists a Lanczos algorithm that is the dual equivalent of BLanczos. We call this algorithm Restricted BLanczos (RBLanczos). It produces identical iterates, in exact arithmetic, to RBCG, BCG and BLanczos. RBLanczos simply solves the linear system (28) using a Lanczos algorithm equipped with the (possibly semi-definite) $\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}$-inner product. The solution in control space is then recovered from (8). RBLanczos is a variant of the Restricted FOM (RSFOM) algorithm (Gratton *et al.*, 2009) adapted to symmetric and positive-definite matrices. The derivation of RBLanczos is sketched in the next section, with more technical details provided in Appendix B.

**Algorithm 3:** RBCG

$\boldsymbol{\lambda}_0 = \mathbf{0}$
$\mathbf{c}_0 = \mathbf{0}$
$\widehat{\mathbf{r}}_0 = \mathbf{R}^{-1}\mathbf{d}$
$\widehat{\mathbf{p}}_0 = \widehat{\mathbf{r}}_0$
$\mathbf{w}_0 = \mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0$
$\mathbf{t}_0 = \mathbf{w}_0$
**for** $i = 0, 1, \ldots l-1$ **do**
  $\widehat{\mathbf{q}}_i = \mathbf{R}^{-1}\mathbf{t}_i + \widehat{\mathbf{p}}_i$
  $\alpha_i = \mathbf{w}_i^{\mathrm{T}}\widehat{\mathbf{r}}_i / \widehat{\mathbf{q}}_i^{\mathrm{T}}\mathbf{t}_i$
  $\boldsymbol{\lambda}_{i+1} = \boldsymbol{\lambda}_i + \alpha_i\widehat{\mathbf{p}}_i$
  $\mathbf{c}_{i+1} = \mathbf{c}_i + \alpha_i\mathbf{t}_i$ } for diagnosing $J_{\mathrm{b}}$
  $\widehat{\mathbf{r}}_{i+1} = \widehat{\mathbf{r}}_i - \alpha_i\widehat{\mathbf{q}}_i$
  Re-orthogonalize $\widehat{\mathbf{r}}_{i+1}$
  $\mathbf{w}_{i+1} = \mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_{i+1}$
  $\beta_i = \mathbf{w}_{i+1}^{\mathrm{T}}\widehat{\mathbf{r}}_{i+1} / \mathbf{w}_i^{\mathrm{T}}\widehat{\mathbf{r}}_i$
  $\widehat{\mathbf{p}}_{i+1} = \widehat{\mathbf{r}}_{i+1} + \beta_i\widehat{\mathbf{p}}_i$
  $\mathbf{t}_{i+1} = \mathbf{w}_{i+1} + \beta_i\mathbf{t}_i$
**end**
$\delta\mathbf{u}_l = \mathbf{B}\mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda}_l$

### 2.4. RBLanczos algorithm

Following GT09 and Gratton *et al.* (2009), we assume that the initial residual $\mathbf{r}_0 \in \mathbb{R}^n$ is in the range of the operator $\mathbf{G}^{\mathrm{T}}$. This assumption is valid when the initial guess $\delta\mathbf{u}_0$ is taken to be zero, which is the case here. When the initial guess is different from zero, the assumption still holds but a generalized algorithm with augmented vectors and matrices is required (GT09).

The residual can be written as

$$\mathbf{r}_0 = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0\,, \tag{30}$$

where $\widehat{\mathbf{r}}_0 = \mathbf{R}^{-1}\mathbf{d} \in \mathbb{R}^m$. From the basic assumption (30), it is shown in Appendix B that there exist $m$-dimensional vectors $\widehat{\mathbf{v}}_i$, $\widehat{\mathbf{z}}_i$, $\widehat{\mathbf{q}}_i$, $\widehat{\mathbf{w}}_i$ and $\widehat{\mathbf{t}}_i$ that can be related to the corresponding $n$-dimensional vectors $\mathbf{v}_i, \mathbf{z}_i, \mathbf{q}_i, \mathbf{w}_i$ and $\mathbf{t}_i$ of BLanczos Algorithm 1 according to

$$\left.\begin{aligned} \mathbf{G}\mathbf{t}_i &= \widehat{\mathbf{t}}_i, \\ \mathbf{v}_i &= \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{v}}_i \end{aligned}\right\} \tag{31}$$

for $i \geq 0$, and

$$\left.\begin{aligned} \mathbf{G}\mathbf{z}_i &= \widehat{\mathbf{z}}_i, \\ \mathbf{q}_i &= \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{q}}_i, \\ \mathbf{w}_i &= \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{w}}_i \end{aligned}\right\} \tag{32}$$

where $i \geq 1$.

Equations (30)–(32) allow all the recurrence relations in BLanczos Algorithm 1 involving vectors of dimension $n$ to be transformed directly into corresponding recurrence relations involving vectors of dimension $m$. This yields the RBLanczos algorithm given by Algorithm 4.

As with BLanczos, one loop of RBLanczos requires a single matrix-vector product with each of the operators $\mathbf{G}$,

**Algorithm 4:** RBLanczos with re-orthogonalization

$\widehat{\mathbf{r}}_0 = \mathbf{R}^{-1}\mathbf{d}$
$\widehat{\mathbf{t}}_0 = \mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0$
$\beta_0 = \sqrt{\widehat{\mathbf{t}}_0^{\mathrm{T}}\widehat{\mathbf{r}}_0}$
$\widehat{\mathbf{v}}_1 = \widehat{\mathbf{r}}_0/\beta_0$
$\widehat{\mathbf{z}}_1 = \widehat{\mathbf{t}}_0/\beta_0$
$\beta_1 = 0$
$\widehat{\mathbf{v}}_0 = \mathbf{0}$
$\widehat{\mathbf{Z}}_1 := [\,\widehat{\mathbf{z}}_1\,]$
$\widehat{\mathbf{V}}_1 := [\,\widehat{\mathbf{v}}_1\,]$
**for** $i = 1, 2, \ldots, l$ **do**
  $\widehat{\mathbf{q}}_i = (\widehat{\mathbf{v}}_i + \mathbf{R}^{-1}\widehat{\mathbf{z}}_i) - \beta_i\widehat{\mathbf{v}}_{i-1}$
  $\alpha_i = \widehat{\mathbf{q}}_i^{\mathrm{T}}\widehat{\mathbf{z}}_i$
  $\widehat{\mathbf{w}}_i = \widehat{\mathbf{q}}_i - \alpha_i\widehat{\mathbf{v}}_i$
  Re-orthogonalize $\widehat{\mathbf{w}}_i$ using $\widehat{\mathbf{V}}_i$ and $\widehat{\mathbf{Z}}_i$
  $\widehat{\mathbf{t}}_i = \mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{w}}_i$
  $\beta_{i+1} = \sqrt{\widehat{\mathbf{t}}_i^{\mathrm{T}}\widehat{\mathbf{w}}_i}$
  $\widehat{\mathbf{v}}_{i+1} = \widehat{\mathbf{w}}_i/\beta_{i+1}$
  $\widehat{\mathbf{z}}_{i+1} := \widehat{\mathbf{t}}_i/\beta_{i+1}$
  $\widehat{\mathbf{Z}}_i := [\,\widehat{\mathbf{Z}}_i, \widehat{\mathbf{z}}_{i+1}\,]$
  $\widehat{\mathbf{V}}_i := [\,\widehat{\mathbf{V}}_i, \widehat{\mathbf{v}}_{i+1}\,]$
  $(\mathbf{T}_i)_{i,i} = \alpha_i$
  **if** $i > 1$ **then**
    $(\mathbf{T}_i)_{i-1,i} = (\mathbf{T}_i)_{i,i-1} = \beta_i$
  **end**
**end**
Solve $\mathbf{T}_l\mathbf{s}_l = \beta_0\mathbf{e}_1$
$\delta\mathbf{u}_l = \mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{V}}_l\mathbf{s}_l$

$\mathbf{G}^{\mathrm{T}}$, $\mathbf{R}^{-1}$ and $\mathbf{B}$, but with the important difference that all vectors are defined in $\mathbb{R}^m$ instead of $\mathbb{R}^n$. The two algorithms are mathematically equivalent.

As discussed in section 2.2, BLanczos searches for the solution $\delta\mathbf{u}_l$ in the Krylov subspace (23). It is instructive to determine the corresponding Krylov subspace for RBLanczos. Replacing $\mathbf{r}_0$ in (23) with (30) yields

$$\mathcal{K}^l(\mathbf{BA}, \mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0) = \mathrm{span}\left\{\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0, \ldots, (\mathbf{BA})^{l-1}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0\right\}. \quad (33)$$

It can be easily shown that

$$\mathbf{A}\mathbf{B}\mathbf{G}^{\mathrm{T}} = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{A}},$$

where $\widehat{\mathbf{A}} = \mathbf{I}_m + \mathbf{R}^{-1}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}$. Equation (33) can then be written as

$$\begin{aligned}\mathcal{K}^l(\mathbf{BA}, \mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0) &= \mathrm{span}\left\{\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0, \ldots, \mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{A}}^{l-1}\widehat{\mathbf{r}}_0\right\}\\ &= \mathbf{B}\mathbf{G}^{\mathrm{T}}\mathcal{K}^l(\widehat{\mathbf{A}}, \widehat{\mathbf{r}}_0),\end{aligned}$$

where

$$\mathcal{K}^l(\widehat{\mathbf{A}}, \widehat{\mathbf{r}}_0) = \mathrm{span}\left\{\widehat{\mathbf{r}}_0, \widehat{\mathbf{A}}\widehat{\mathbf{r}}_0, \ldots, \widehat{\mathbf{A}}^{l-1}\widehat{\mathbf{r}}_0\right\}$$

is the Krylov subspace generated by RBLanczos.

When re-orthogonalization is not required, as with BLanczos, RBLanczos can be transformed into an equivalent algorithm that does not require storing the matrices $\mathbf{T}_l$ and $\widehat{\mathbf{V}}_l$ to find the solution. This algorithm is provided in Appendix A.

### 2.5. *Re-orthogonalization*

The Lanczos vectors in BLanczos (or the residuals in BCG) are, in exact arithmetic, mutually orthogonal with respect to the **B**-inner product. Round-off errors can result in a loss of **B**-orthogonality and can significantly hinder the convergence of these methods as a consequence. It is possible to alleviate this problem by re-orthogonalizing the Lanczos vectors (or the CG residuals) on each iteration using a modified Gram–Schmidt (MGS) procedure (Saad, 1996, pp 11–12).

With reference to Algorithm 1 for BLanczos, the re-orthogonalization procedure acts on the vectors $\mathbf{w}_i$. Making use of the orthogonality relationship $\mathbf{v}_i^{\mathrm{T}}\mathbf{B}\mathbf{v}_j = 0$, for $i \neq j$, MGS re-orthogonalization can be described in compact notation by

$$\mathbf{w}_i \leftarrow \left[\prod_{j=i-1}^{1}(\mathbf{I}_n - \mathbf{v}_j\mathbf{v}_j^{\mathrm{T}}\mathbf{B})\right]\mathbf{w}_i\,, \quad (34)$$

$$= \left[\prod_{j=i-1}^{1}(\mathbf{I}_n - \mathbf{v}_j\mathbf{z}_j^{\mathrm{T}})\right]\mathbf{w}_i\,. \quad (35)$$

Equations (34) and (35) require the storage of the Lanczos vectors $\mathbf{v}_j$ from all previous iterations $j = 1, \ldots, i-1$. Equation (34) is clearly less practical than (35) since it requires $i-1$ additional, and generally expensive, applications of **B** to compute the **B**-inner product of $\mathbf{v}_j$ and $\mathbf{w}_i$. This expensive matrix-vector product can be avoided by storing the vectors $\mathbf{z}_j$ and using them in (35).

The need to store and manipulate the sequence of $n$-dimensional vectors $\mathbf{v}_j$ and $\mathbf{z}_j$ can lead to significant computational overhead, as will be illustrated in the experiments in section 3. In this respect, the dual algorithms, which involve sequences of $m$-dimensional vectors, are clearly an attractive alternative when $m \ll n$. From the relationship $\mathbf{v}_i = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{v}}_i$, the orthogonality condition for the Lanczos vectors can be written as

$$\mathbf{v}_i^{\mathrm{T}}\mathbf{B}\mathbf{v}_j = \widehat{\mathbf{v}}_i^{\mathrm{T}}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{v}}_j = 0 \quad (36)$$

for $i \neq j$. Combining (36) with (34) and (35) leads to the MGS re-orthogonalization scheme for RBLanczos in Algorithm 4:

$$\widehat{\mathbf{w}}_i \leftarrow \left[\prod_{j=i-1}^{1}(\mathbf{I}_m - \widehat{\mathbf{v}}_j\widehat{\mathbf{v}}_j^{\mathrm{T}}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}})\right]\widehat{\mathbf{w}}_i\,,$$

$$= \left[\prod_{j=i-1}^{1}(\mathbf{I}_m - \widehat{\mathbf{v}}_j\widehat{\mathbf{z}}_j^{\mathrm{T}})\right]\widehat{\mathbf{w}}_i\,.$$

### 2.6. *Monitoring convergence*

The values of the quadratic cost function and the norm of the cost function gradient are important for monitoring the convergence of the minimization. In addition to the total value of the cost function, $J$, the relative contributions to $J$ from the background term $J_{\mathrm{b}}$ and observation term $J_{\mathrm{o}}$ provide additional useful diagnostic information. Inexpensive formulae for computing all these quantities on each iteration from the recurrence relations in BLanczos,

RBLanczos, BCG and RBCG (Algorithms 1, 4, 2 and 3) are derived in this section. Similar formulae can also be derived for the corresponding direct Lanczos algorithms (Algorithms 5 and 6) given in Appendix A.

*2.6.1. Quadratic cost function*

**BLanczos**

Using a Taylor series expansion about the initial guess $\delta\mathbf{u}_0 = \mathbf{0}$, the quadratic cost function $J[\delta\mathbf{u}_i]$ in (3) can be expressed as

$$J[\delta\mathbf{u}_i] = J[\mathbf{0}] + \delta\mathbf{u}_i^{\mathrm{T}}\nabla J[\mathbf{0}] + \frac{1}{2}\delta\mathbf{u}_i^{\mathrm{T}}(\nabla^2 J)\,\delta\mathbf{u}_i.$$

Replacing $\nabla^2 J$ by $\mathbf{A}$ (Eq. (9)) and $\nabla J[\mathbf{0}]$ by $-\mathbf{r}_0$ (Eq. (11)), and using the relations (24) and (25), the quadratic cost function can be expressed in terms of quantities from BLanczos as

$$\begin{aligned} J[\delta\mathbf{u}_i] &= J[\mathbf{0}] - \mathbf{s}_i^{\mathrm{T}}\mathbf{Z}_i^{\mathrm{T}}\mathbf{r}_0 + \frac{1}{2}\mathbf{s}_i^{\mathrm{T}}\mathbf{Z}_i^{\mathrm{T}}\mathbf{A}\mathbf{Z}_i\mathbf{s}_i \\ &= J[\mathbf{0}] - \mathbf{s}_i^{\mathrm{T}}\mathbf{Z}_i^{\mathrm{T}}\mathbf{r}_0 + \frac{1}{2}\mathbf{s}_i^{\mathrm{T}}\mathbf{T}_i\mathbf{s}_i \\ &= J[\mathbf{0}] - \mathbf{s}_i^{\mathrm{T}}\mathbf{Z}_i^{\mathrm{T}}\mathbf{r}_0 + \frac{1}{2}\mathbf{s}_i^{\mathrm{T}}\mathbf{Z}_i^{\mathrm{T}}\mathbf{r}_0 \\ &= J[\mathbf{0}] - \frac{1}{2}\mathbf{s}_i^{\mathrm{T}}\mathbf{Z}_i^{\mathrm{T}}\mathbf{r}_0, \end{aligned} \tag{37}$$

where

$$J[\mathbf{0}] = \frac{1}{2}\mathbf{d}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d}. \tag{38}$$

Using again (24) and (25), the $J_{\mathrm{b}}$ term can be evaluated as

$$\begin{aligned} J_{\mathrm{b}}[\delta\mathbf{u}_i] &= \frac{1}{2}\delta\mathbf{u}_i^{\mathrm{T}}\mathbf{B}^{-1}\delta\mathbf{u}_i \\ &= \frac{1}{2}\mathbf{s}_i^{\mathrm{T}}\mathbf{Z}_i^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{Z}_i\mathbf{s}_i \\ &= \frac{1}{2}\mathbf{s}_i^{\mathrm{T}}\mathbf{V}_i^{\mathrm{T}}\mathbf{Z}_i\mathbf{s}_i. \end{aligned} \tag{39}$$

The expressions for $J$ and $J_{\mathrm{b}}$ depend on the vectors $\mathbf{s}_i$. These vectors can be computed on each iteration by solving the (inexpensive for small $i$) tridiagonal system $\mathbf{T}_i\mathbf{s}_i = \beta_0\mathbf{e}_1$.

The $J_{\mathrm{o}}$ term is expensive to compute directly since it requires evaluating the term $\mathbf{G}\delta\mathbf{u}_i$ which is not directly available on each iteration of BLanczos. It also requires application of $\mathbf{R}^{-1}$ which may be costly when correlated observation error is accounted for in $\mathbf{R}$. Fortunately, given $J$ and $J_{\mathrm{b}}$, $J_{\mathrm{o}}$ can be deduced simply from their difference:

$$J_{\mathrm{o}}[\delta\mathbf{u}_i] = J[\delta\mathbf{u}_i] - J_{\mathrm{b}}[\delta\mathbf{u}_i]. \tag{40}$$

However, in some applications, additional constraints are included in the cost function via a penalty term $J_{\mathrm{c}}$, in which case (40) would allow only the sum of $J_{\mathrm{o}}$ and $J_{\mathrm{c}}$ to be recovered. A typical example is when a digital filter is employed as a weak constraint in 4D-Var in order to penalize high-frequency noise in the solution trajectory (Gauthier and Thépaut, 2001). In order to isolate $J_{\mathrm{o}}$, the $J_{\mathrm{c}}$ term would then need to be evaluated explicitly on each iteration.

**RBLanczos**

The values of $J$ and $J_{\mathrm{b}}$ can be evaluated in terms of quantities from RBLanczos by using the relation (30) and the relations for $\mathbf{v}_i$ and $\mathbf{z}_i$ in (31) and (32) in the expressions (37) and (39). This yields

$$\begin{aligned} J[\delta\mathbf{u}_i] &= J[\mathbf{0}] - \frac{1}{2}\mathbf{s}_i^{\mathrm{T}}\mathbf{Z}_i^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0 \\ &= J[\mathbf{0}] - \frac{1}{2}\mathbf{s}_i^{\mathrm{T}}\widehat{\mathbf{Z}}_i^{\mathrm{T}}\widehat{\mathbf{r}}_0, \end{aligned}$$

and

$$\begin{aligned} J_{\mathrm{b}}[\delta\mathbf{u}_i] &= \frac{1}{2}\mathbf{s}_i^{\mathrm{T}}\widehat{\mathbf{V}}_i^{\mathrm{T}}\mathbf{G}\mathbf{Z}_i\mathbf{s}_i \\ &= \frac{1}{2}\mathbf{s}_i^{\mathrm{T}}\widehat{\mathbf{V}}_i^{\mathrm{T}}\widehat{\mathbf{Z}}_i\mathbf{s}_i, \end{aligned}$$

where $J[\mathbf{0}]$ is again given by (38). The $J_{\mathrm{o}}$ term can then be evaluated from (40). The matrices $\widehat{\mathbf{V}}_i$ and $\widehat{\mathbf{Z}}_i$ contain the $j = 1, \ldots, i$ column vectors $\widehat{\mathbf{v}}_j$ and $\widehat{\mathbf{z}}_j$. These matrices are also needed for re-orthogonalization.

**BCG**

For BCG, the quadratic cost function can be calculated from (37) using the relation (24):

$$J[\delta\mathbf{u}_i] = J[\mathbf{0}] - \frac{1}{2}\delta\mathbf{u}_i^{\mathrm{T}}\mathbf{r}_0\,, \tag{41}$$

where $J[\mathbf{0}]$ is given by Eq. (38).

The $J_{\mathrm{b}}$ term can be calculated as

$$\begin{aligned} J_{\mathrm{b}}[\delta\mathbf{u}_i] &= \frac{1}{2}\delta\mathbf{u}_i^{\mathrm{T}}\mathbf{B}^{-1}\delta\mathbf{u}_i \\ &= \frac{1}{2}\delta\mathbf{u}_i^{\mathrm{T}}\mathbf{f}_i\,, \end{aligned}$$

where $\mathbf{f}_i = \mathbf{B}^{-1}\delta\mathbf{u}_i$ can be computed without the need to apply $\mathbf{B}^{-1}$ by including an additional recurrence relation in Algorithm 2. The $J_{\mathrm{o}}$ term can then be computed from (40).

**RBCG**

For RBCG, the quadratic cost function is calculated from (41) using the relations (8) and (30):

$$\begin{aligned} J[\delta\mathbf{u}_i] &= J[\mathbf{0}] - \frac{1}{2}\boldsymbol{\lambda}_i^{\mathrm{T}}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0 \\ &= J[\mathbf{0}] - \frac{1}{2}\boldsymbol{\lambda}_i^{\mathrm{T}}\mathbf{w}_0, \end{aligned}$$

where $\mathbf{w}_0 = \mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0$ is available from Algorithm 3. Using (8), the $J_{\mathrm{b}}$ term can be written as

$$\begin{aligned} J_{\mathrm{b}}[\delta\mathbf{u}_i] &= \frac{1}{2}\boldsymbol{\lambda}_i^{\mathrm{T}}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda}_i \\ &= \frac{1}{2}\boldsymbol{\lambda}_i^{\mathrm{T}}\mathbf{c}_i\,, \end{aligned}$$

where $\mathbf{c}_i = \mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda}_i$ can be diagnosed at little extra cost by including an additional recurrence relation in Algorithm 3. The $J_{\mathrm{o}}$ term can then be computed from (40).

*2.6.2. Norm of the quadratic cost function gradient*

**BLanczos**

The residual vector of the approximate solution $\delta\mathbf{u}_i$ computed by BLanczos satisfies (Saad, 1996, p 153)

$$\mathbf{r}_i = \mathbf{b} - \mathbf{A}\delta\mathbf{u}_i = -\beta_{i+1}\mathbf{e}_i^{\mathrm{T}}\mathbf{s}_i\mathbf{v}_{i+1}\,,$$

where $\mathbf{e}_i$ is the $i$th column of the $i \times i$ identity matrix, and $\beta_{i+1}$, $\mathbf{s}_i$ and $\mathbf{v}_i$ are as defined in Algorithm 1. Since $\mathbf{r}_i = -\nabla J[\delta\mathbf{u}_i]$, the $\mathbf{B}$-norm of the cost function gradient can be readily computed as

$$\|\nabla J[\delta\mathbf{u}_i]\|_{\mathbf{B}} = \|\mathbf{r}_i\|_{\mathbf{B}} = \beta_{i+1}|\mathbf{e}_i^{\mathrm{T}}\mathbf{s}_i|. \tag{42}$$

**RBLanczos**

Formula (42) can also be used to compute the gradient norm with RBLanczos.

**BCG**

For BCG, the gradient norm is given by

$$\|\nabla J[\delta\mathbf{u}_i]\|_{\mathbf{B}} = \|\mathbf{r}_i\|_{\mathbf{B}} = \sqrt{\mathbf{r}_i^{\mathrm{T}}\mathbf{z}_i}\,,$$

where the vectors $\mathbf{z}_i$ and $\mathbf{r}_i$ are defined in Algorithm 2.

**RBCG**

For RBCG, the norm is obtained from

$$\|\nabla J[\delta\mathbf{u}_i]\|_{\mathbf{B}} = \|\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_i\|_{\mathbf{B}} = \|\widehat{\mathbf{r}}_i\|_{\mathbf{GBG}^{\mathrm{T}}} = \sqrt{\widehat{\mathbf{r}}_i^{\mathrm{T}}\mathbf{w}_i}$$

where the vectors $\widehat{\mathbf{r}}_i$ and $\mathbf{w}_i$ are defined in Algorithm 3.

## 3. Numerical experiments

### *3.1. Experimental framework*

This section provides a brief description of the NEMOVAR and ROMS data assimilation systems and the framework in which the experiments are performed.

#### *3.1.1. NEMOVAR*

NEMOVAR is a variational data assimilation system (Mogensen *et al.*, 2009) for the NEMO ocean model (Madec, 2008). It is designed as an incremental 4D-Var algorithm. 3D-Var is also supported, using the First-Guess at Appropriate Time (FGAT) approach. 3D-Var FGAT follows the basic formulation of incremental 4D-Var but replaces the linearized model propagator with the identity operator ($\mathbf{M}(t_j, t_0) \equiv \mathbf{I}_n$). Dynamical information is still incorporated into the 3D-Var analysis through a balance operator, which relates the control vector to the initial state vector (see later).

A close variant of the operational version of NEMOVAR used at ECMWF for seasonal forecasting and ocean reanalysis (Mogensen *et al.*, 2012) is used for the experiments presented here. The system is based on 3D-Var FGAT. The configuration is global, with 42 vertical levels and approximately 1° resolution in the Extratropics. In the Tropics, the meridional resolution is refined, reaching a minimum value of 0.3° directly at the Equator.

Both BCG and RBCG have been implemented in NEMOVAR, following very similar technical specifications. NEMOVAR also includes a version of Lanczos (the Lanczos method on the linear system (19) with the canonical inner product). Here the experiments focus on comparing BCG and RBCG, and will be presented in section 3.2. These experiments are conducted on a single 3D-Var analysis cycle where the time period covered by the cycle is 10 days. The background initial conditions are taken from the ECMWF ORAS4 ocean reanalysis (Balmaseda *et al.*, 2013) on 1 January 2006. The model state variables comprise potential temperature ($T$), salinity ($S$), sea-surface height ($\eta$) and the horizontal components ($u, v$) of velocity. The background-state trajectory on the 10-day window is generated by integrating forward the NEMO model using the same daily surface forcing fluxes and relaxation strategies to sea-surface temperature and climatology as used in ORAS4 for the same period. The observations that are assimilated on this cycle are the same as those used in ORAS4 on the same cycle. They consist of temperature and salinity ($T/S$) profiles from the quality-controlled EN3_v2a dataset[†] and along-track sea-level anomaly (SLA) data from altimeter database at AVISO (Archiving, Validation and Interpretation of Satellite Oceanographic data)[‡]. The SLA data are referenced to a mean dynamic topography defined as the time-mean sea level from an ocean reanalysis similar to ORAS4 but assimilating only $T/S$ profiles. The total number of individual observations assimilated on the 10-day 3D-Var cycle from 1 to 11 January 2006 is approximately $5.0 \times 10^5$, with $2.1 \times 10^5$ for $T$, $1.6 \times 10^5$ for $S$, and $1.3 \times 10^5$ for SLA.

The variables in the control vector $\delta\mathbf{u}$ consist of increments $\delta T$, $\delta S_{\mathrm{U}}$ and $\delta\eta_{\mathrm{U}}$ where the subscript U denotes an 'unbalanced' component of that variable (Weaver *et al.*, 2005). The control variables are assumed to be mutually uncorrelated. A linearized balance operator $\mathbf{K}$ relates $\delta\mathbf{u}$ to the increments $\delta T$, $\delta S$, $\delta\eta$, $\delta u$ and $\delta v$ comprising the initial state $\delta\mathbf{x}(t_0)$. The balance operator acts as a multivariate constraint in the linearized generalized observation operator $\mathbf{G}$ in the observation term of the cost function (3). In this experiment, the horizontal velocity increments $\delta u$ and $\delta v$ are determined entirely by geostrophic constraints in $\mathbf{K}$ (Mogensen *et al.*, 2012). The total number of *active* control variables is approximately $4.6 \times 10^6$.

For computational convenience, a land–ocean mask with values of 0 (land) and 1 (ocean) is used in NEMO, as in many ocean models, to account for the complex geometry of land–ocean boundaries. As a consequence, (passive) land points are included along with (active) ocean points in the model-variable arrays in the computer code. By including the land points, the size of the control vector is roughly $9.2 \times 10^6$. Half of the memory allocation in a control vector is thus associated with land points. A control vector in this experiment requires 19 times more memory than an observation vector.

[†] http://www.metoffice.ogv.uk/hadobs/en3

[‡] http://www.aviso.oceanobs.com/en/data/products/sea- surface-height-products/global/sla/index.html

*3.1.2. ROMS 4D-Var*

ROMS is a hydrostatic, primitive equation, Boussinesq ocean general circulation model designed primarily for coastal applications. Terrain-following vertical coordinates are employed allowing for greater vertical resolution in shallow water and regions with complex bathymetry (Marchesiello *et al.*, 2001; Shchepetkin and McWilliams, 2003, 2005; Haidvogel *et al.*, 2008). ROMS supports both a primal and dual formulation of incremental 4D-Var. The primal form follows very closely that of Weaver *et al.* (2003) and solves (19) using the standard Lanczos algorithm which is equivalent to solving (5) using the BLanczos algorithm.

Two forms of dual 4D-Var are available in ROMS, which differ in the choice of linearization strategy on the outer loop (Moore *et al.*, 2011a,b). One follows the incremental approach (Courtier *et al.*, 1994), while the other is based on the indirect representer approach of Egbert *et al.* (1994). For the current study, there is no distinction between these approaches since the experiments are conducted with a single outer loop where the linearization trajectory is the background.

As noted earlier, one of the primary advantages of the dual approach to 4D-Var is in the weak-constraint formulation. Since (29) is solved in a space spanned by vectors whose size is determined by the number of observations, the dimension of the problem is not changed by the addition of control variables to account for model error. Conversely, in the primal form embodied by (5), the dimension of the control vector swells very quickly as additional control variables are added.

Four options are available in ROMS for solving the quadratic problem (3), namely Lanczos (the Lanczos method on the linear system (19) with the canonical inner product), dual-Lanczos (the Lanczos method on the linear system (29) with the canonical inner product), dual-MINRES (MINRES on the linear system (29) with the canonical inner product), and RBLanczos. A comparison of the four approaches under the assumption of both strong and weak constraint will be presented in section 3.2.

The configuration of ROMS used for the experiments presented here is for the California Current System (CCS), an eastern boundary current characterized by a pronounced seasonal cycle of upwelling and by energetic mesoscale circulations (Hickey, 1998; Checkley and Barth, 2009), providing a challenge for linear-based data assimilation methods such as incremental 4D-Var. The ROMS CCS domain and circulation is described by Veneziani *et al.* (2009a,b) and Broquet *et al.* (2009a,b). It spans the region 134–116°W, 31–48°N, with 10 km resolution in the horizontal and 42 $\sigma$-levels. The minimization experiments described here are conducted for a single 4D-Var analysis cycle spanning the 7-day period 29 March–4 April 2003. The background initial conditions are taken from a 4D-Var sequence described in Moore *et al.* (2011b), and the model control variables are the same as those of NEMOVAR, with the addition of unbalanced variables $\delta\mathbf{u}_{\mathrm{U}}$ and $\delta\mathbf{v}_{\mathrm{U}}$ for horizontal velocity. The background trajectory is generated using forcing derived from daily averaged output of atmospheric boundary-layer fields from the Naval Research Laboratory's Coupled Ocean–Atmosphere Mesoscale Prediction System (COAMPS) (Doyle *et al.*, 2009). The ocean surface fluxes were derived using the bulk formulations of Liu *et al.* (1979) and Fairall *et al.* (1996a,b), and represent the background surface forcing for 4D-Var. The model domain has open boundaries at the northern, southern, and western edges, and at these boundaries the tracer and velocity fields were prescribed, while the free surface and vertically integrated flow are subject to Chapman (1985) and Flather (1976) boundary conditions respectively. The prescribed open-boundary solution was taken from the Estimating the Circulation and Climate of the Ocean (ECCO) global data assimilation product (Wunsch and Heimbach, 2007), and represents the background open-boundary conditions for 4D-Var.

In the strong-constraint 4D-Var experiments, the control vector comprises increments to the model initial conditions, the ocean surface forcing (namely the surface wind stress, and the surface fluxes of heat and freshwater), and open boundary conditions. The number of control variables is $7.3 \times 10^6$. In the weak-constraint 4D-Var experiments, the control vector is augmented to include corrections for model error as well, which take the form of space–time corrections to the model equations of motion. The size of the control vector in this case is $5.2 \times 10^7$. The implementation of the ROMS weak-constraint 4D-Var used here is the same as that described in Moore *et al.* (2011b).

The implementation of ROMS weak-constraint 4D-Var follows the dual approach of Chua and Bennett (2001) where a correction term for model error enters as a forcing on the right-hand side of the ROMS TL equations in the inner loop, and in the ROMS nonlinear equations in the outer loop. The ROMS formulation is described in detail by Moore *et al.* (2011b). While the structure of the cost function is unchanged in this approach, the background term $J_{\mathrm{b}}$ now contains additional terms that penalize departures of the model-error correction terms from a zero model-error background. The background-error covariance matrix $\mathbf{B}$ is expanded and contains block-diagonal entries $\mathbf{Q}$ that describe the model-error covariance. Similarly, the structure of the Hessian matrix does not change and would have the same structure implied by the system matrix in (5) but with $\mathbf{B}$ now expanded to include the $\mathbf{Q}$ matrix blocks. Therefore, the problem can be preconditioned by $\mathbf{B}$ in the usual way, although the addition of $\mathbf{Q}$ to the expanded $\mathbf{B}$ will undoubtedly influence the conditioning of the Hessian.

Since we do not have a primal weak-constraint algorithm for ROMS, we cannot explore directly the change in properties of the Hessian due to the addition of $\mathbf{Q}$. However, the influence of $\mathbf{Q}$ on the condition number of the preconditioned stabilized representer matrix $\mathbf{R}^{-1/2}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1/2} + \mathbf{I}_m$ in (29) has been examined. The condition number of the preconditioned Hessian matrix $\mathbf{I}_{n'} + \mathbf{U}^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\mathbf{U}$ in (18) can then be estimated since this matrix has the same eigenvalues as the preconditioned stabilized representer matrix, together with an additional $n' - m$ eigenvalues equal to 1 (Courtier, 1997; Horn and Johnson, 1999, pp 53–54). A one-year sequence of strong- and weak-constraint 4D-Var simulations using a lower-resolution version of the California Current ROMS reveals that the addition of $\mathbf{Q}$ to $\mathbf{B}$ generally raises the condition number of the Hessian matrix, although the difference in condition number between the strong- and weak-constaint cases is typically less than an order of magnitude (P. J. Smith, personal communication).

The observations assimilated are the same as those described in Moore *et al.* (2011b) and include: gridded sea-surface height analyses in the form of dynamic topography

from AVISO at approximately $1/3^\circ$ resolution every 7 days; a blended sea-surface temperature (SST) product with 10 km resolution, available daily from National Oceanic and Atmospheric Administration (NOAA) CoastWatch and consisting of 5-day means derived from the Goddard Earth Observing System (GEOS), Advanced Very High Resolution Radiometer (AVHRR) and Moderate Resolution Imaging Spectroradiometer (MODIS) satellite instruments; and *in situ* hydrographic observations extracted from the quality-controlled EN3_v1d dataset. The total number of observations is of the order $10^5$ ($1.0 \times 10^5$ for $T$, $5.0 \times 10^2$ for $S$, and $2.4 \times 10^3$ for SSH).

### 3.2. Results

This section describes the results from numerical experiments with NEMOVAR and ROMS that are designed to compare the various primal and dual algorithms discussed in this article. The distinguishing characteristics of these algorithms are summarized in Table 1. The different experiments are summarized in Table 2.

#### 3.2.1. NEMOVAR

The experiments with NEMOVAR are based on 3D-Var FGAT with a single outer-loop iteration and 40 inner-loop iterations. The properties of the minimization are summarized in Figure 1. Figure 1(a) shows the iterative evolution of the quadratic cost function (3) using the primal formulation (BCG) and dual formulation (RBCG) of the CG algorithm. The curves are indistinguishable as expected from the theory. Numerical results with Lanczos (not shown) also produce indistinguishable results from those of BCG and RBCG.

The results also illustrate that, for this 3D-Var experiment, there is no visible improvement in the cost function minimization when using re-orthogonalization. However, a positive impact from re-orthogonalization is seen after iteration 22 in the evolution of the gradient norm (Figure 1(b)). With re-orthogonalization, the gradient norms in the BCG_O and RBCG_O experiments are indistinguishable (when the norm is defined with respect to the appropriate metric: $\mathbf{B}$ in BCG, and $\mathbf{GBG}^{\mathrm{T}}$ in RBCG).

While Figure 1 shows that the algorithms for the primal and dual formulations have near-identical convergence properties, Figure 2 illustrates that the memory requirements of the algorithms can be very different, particularly with re-orthogonalization. This figure shows the memory usage as a function of execution time, where the initial point has been chosen just before entering the minimization loop. The jump in the curves near the start corresponds to static memory allocation within the CG algorithms ($\sim$2.5 Gb for BCG; $\sim$2 Gb for RBCG). Thereafter, the total memory usage remains constant if re-orthogonalization is not activated, resulting in a saving of $\sim$0.5 Gb with RBCG. Re-orthogonalization requires storing an extra pair of vectors on each iteration as discussed in section 2.5. The required memory is allocated dynamically in NEMOVAR, which explains the steady increase of the curves BCG_O and RBCG_O. Re-orthogonalization produces only small memory overhead with RBCG, with the total memory being comparable to that required by BCG *without* re-orthogonalization. On the other hand, re-orthogonalization with BCG requires a significantly larger amount of extra memory ($\sim$6 Gb). With higher-resolution model

Table 1. A summary of the distinguishing characteristics of the different minimization algorithms used in the data assimilation experiments for solving either the primal problem (5) or the dual problem (7).

| Algorithm | Formulation | Description | Reference |
|---|---|---|---|
| BCG | Primal | CG on the linear system (22) with the **B**-inner product | Algorithm 2 |
| Lanczos | Primal | Lanczos on the linear system (18) with the canonical inner product | – |
| (BLanczos) | (Primal) | (Lanczos on the linear system (22) with the **B**-inner product) | (Algorithm 1) |
| RBCG | Dual | CG on the linear system (28) with the $\mathbf{GBG}^{\mathrm{T}}$-inner product | Algorithm 3 |
| RBLanczos | Dual | Lanczos on the linear system (28) with the $\mathbf{GBG}^{\mathrm{T}}$-inner product | Algorithm 4 |
| dual-Lanczos | Dual | Lanczos on the linear system (29) with the canonical inner product | – |
| dual-MINRES | Dual | MINRES on the linear system (29) with the canonical inner product | – |

Table 2. A list of the experiments performed using data assimilation systems with the NEMO and ROMS models.

| Experiment | Description |
|---|---|
| BCG_O | NEMO, Global, 3D-Var, BCG with re-orthogonalization |
| BCG | NEMO, Global, 3D-Var, BCG without re-orthogonalization |
| RBCG_O | NEMO, Global, 3D-Var, RBCG with re-orthogonalization |
| RBCG | NEMO, Global, 3D-Var, RBCG without re-orthogonalization |
| BLanc_OS | ROMS, Regional, Strong-constraint 4D-Var, Lanczos with re-orthogonalization |
| DLanc_OS | ROMS, Regional, Strong-constraint 4D-Var, dual-Lanczos with re-orthogonalization |
| DMin_OS | ROMS, Regional, Strong-constraint 4D-Var, dual-MINRES with re-orthogonalization |
| RBLanc_OS | ROMS, Regional, Strong-constraint 4D-Var, RBLanczos with re-orthogonalization |
| DLanc_OW | ROMS, Regional, Weak-constraint 4D-Var, dual-Lanczos with re-orthogonalization |
| DMin_OW | ROMS, Regional, Weak-constraint 4D-Var, dual-MINRES with re-orthogonalization |
| RBLanc_OW | ROMS, Regional, Weak-constraint 4D-Var, RBLanczos with re-orthogonalization |
| DMin_W | ROMS, Regional, Weak-constraint 4D-Var, dual-MINRES without re-orthogonalization |
| RBLanc_W | ROMS, Regional, Weak-constraint 4D-Var, RBLanczos without re-orthogonalization |

Naming convention: The first few letters are shorthand notation for the minimization algorithm used (see Table 1).
O indicates that re-orthogonalization is used (otherwise it is not used), S indicates strong-constraint 4D-Var, and W indicates weak-constraint 4D-Var.

 

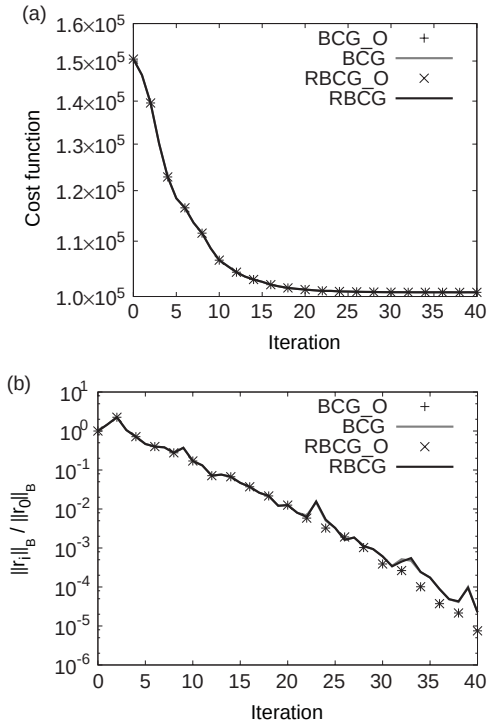


**Figure 1.** (a) The value of the cost function as a function of the CG iteration counter, and (b) the value of the norm of the cost function gradient relative to the norm of the initial cost function gradient, both as a function of the CG iteration counter. The gradient norm is measured with respect to the **B** metric. The experiments are as described in Table 2. The curves in (a), and the curves for BCG_O and RBCG_O in (b), are indistinguishable. A logarithmic vertical scale is used in both panels.

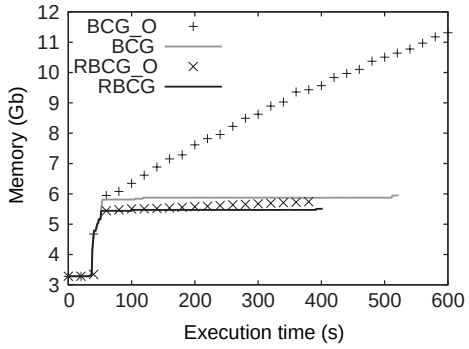


**Figure 2.** Memory usage (gigabytes) as a function of execution time in the minimization loop using CG algorithms based on the primal and dual formulations. Table 2 gives a description of the experiments, which have been performed on a Dell T5500 using a single-CPU processor.

configurations or with applications, such as 4D-Var, that involve additional control variables, the memory requirements could easily become prohibitive. This figure also illustrates that the cost of algebraic computations, such as scalar products, is noticeably cheaper with the dual approach than the primal approach. The total execution time with RBCG is roughly equivalent with or without re-orthogonalization, and is noticeably shorter than with BCG. This is especially noticeable when re-orthogonalization is activated where the computational saving with RBCG is about 22%.

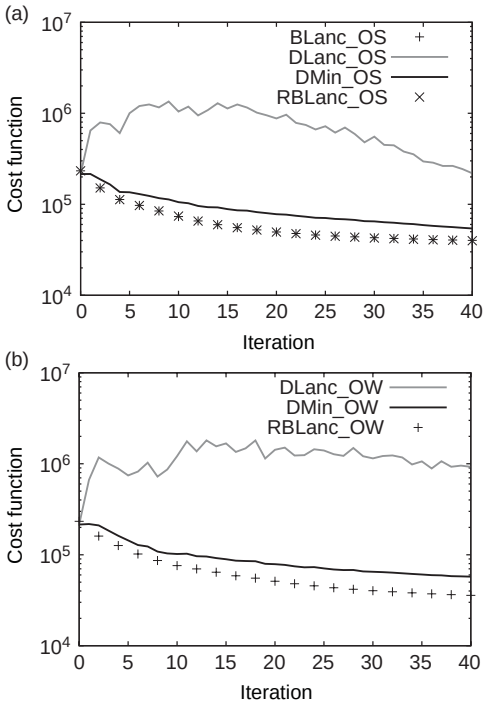


**Figure 3.** The value of the cost function as a function of the Lanczos iteration counter for (a) strong-constraint 4D-Var and (b) weak-constraint 4D-Var, using the Lanczos algorithms based on the primal and dual formulations with re-orthogonalization. Table 2 gives a description of the experiments. A logarithmic vertical scale is used in both panels.

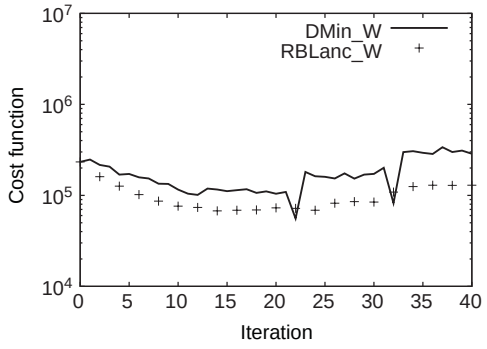


**Figure 4.** As Figure 3, but for the case of weak-constraint 4D-Var without re-orthogonalization of the Lanczos vectors.

### *3.2.2. ROMS 4D-Var*

In all of the experiments presented here, ROMS 4D-Var was run, as was NEMO 3D-Var, with a single outer-loop iteration and 40 inner-loop iterations. Figures 3 and 4 summarize the results of various experiments employing either strong- or weak-constraint 4D-Var. These figures show the behaviour of the quadratic cost function (3) as a function of the number of inner-loop iterations.

The results for the strong-constraint 4D-Var are shown in Figure 3(a). The curves correspond to experiments with the primal solver BLanczos and the dual solvers RBLanczos, dual-MINRES and dual-Lanczos with re-orthogonalization. As anticipated, RBLanczos and BLanczos yield the same sequence of cost function values. Clearly the performance of RBLanczos is superior to both dual-MINRES and

dual-Lanczos. Figure 3(a) illustrates the poor convergence properties of dual-Lanczos, in agreement with results from GT09 and El Akkraoui and Gauthier (2010) who employed CG to the linear system (29) with a canonical inner product (which is mathematically equivalent to applying dual-Lanczos to system (29)). In fact, approximately 100 inner-loop iterations are generally required to achieve an acceptable level of convergence with dual-Lanczos, as shown by Moore *et al.* (2011b). The results of three corresponding weak-constraint dual 4D-Var calculations are shown in Figure 3(b), revealing again the superior convergence properties of RBLanczos. The convergence of weak-constraint 4D-Var with the dual-Lanczos algorithm required more than 200 inner-loop iterations to yield an acceptable level of convergence (Moore *et al.*, 2011b). Note that this plot does not include results for the primal approach since it was not feasible to perform weak-constraint 4D-Var experiments with BLanczos.

The importance of re-orthogonalization with the dual approach is illustrated in Figure 4 which shows, for the case of weak-constraint 4D-Var, the results from applying dual-MINRES and RBLanczos without re-orthogonalization (cf. Figure 3). After an initial period of convergence, the cost function diverges. This is in stark contrast to the NEMO 3D-Var experiment where re-orthogonalization had a relatively minor impact on convergence. The need for re-orthogonalization in weak-constraint 4D-Var places an enormous memory burden on the primal approach. Indeed, in the ROMS weak-constraint 4D-Var experiments, only the dual approach was feasible. Unlike NEMO 3D-Var, the computational cost of ROMS 4D-Var, with strong- or weak-constraints, is comparable for both the primal and dual formulations for the same number of inner-loop iterations since the majority of the computational burden comes from the integration of the TL and adjoint models.

## 4. Summary and conclusions

Variational data assimilation involves the solution of a nonlinear least-squares problem to estimate an $n$-dimensional control vector given an $m$-dimensional observation vector and an $n$-dimensional background estimate of the control vector. A nonlinear generalized observation operator relates the model control vector to the observation vector. Variational data assimilation problems in meteorology and oceanography are commonly solved using an incremental or truncated Gauss–Newton approach. This approach requires minimizing a sequence of quadratic cost functions, where each member of the sequence is derived by linearizing the generalized observation operator about a recent estimate.

The quadratic minimization problems can be solved using a primal or dual approach. Primal approaches search for a solution directly in an $n$-dimensional space associated with the model control variables. Dual approaches exploit the fact that the solution of the quadratic minimization problem is in the range of the adjoint of the generalized observation operator to search for the solution in an $m$-dimensional space associated with the observations. The dual approach becomes especially attractive from a computational viewpoint when $m \ll n$ which is a typical situation in ocean data assimilation and with weak-constraint formulations of 4D-Var in general.

CG and Lanczos methods, which belong to the general class of Krylov subspace iterative methods, are well suited for solving quadratic minimization problems when $n$ and $m$ are large. Conventional implementations of incremental 4D-Var employ a primal approach in conjunction with a CG or Lanczos algorithm. The system is preconditioned by the background-error covariance matrix, **B**, which in practice is achieved through a variable transformation that requires a square-root factorization of **B**. **B**-preconditioning of the CG and Lanczos algorithms can also be achieved without the need to factor **B**. This property, which is convenient with general **B** formulations, is shared by the BCG and BLanczos primal algorithms which were presented in this article. Both algorithms are equivalent in exact arithmetic.

The dual equivalent of BCG is the RPCG algorithm of GT09 with no second-level preconditioning. We called this specific algorithm RBCG to distinguish it from RPCG which allows for general preconditioners. Numerical experiments comparing the performance of RBCG and BCG in a global-ocean 3D-Var assimilation system for the NEMO model were presented. In the experiments, the control vector was roughly 19 times larger than the observation vector. As expected from theory, and as demonstrated in an idealized framework by GT09, RBCG and BCG produce identical iterates to within machine precision. However, the memory requirements and computational costs of the algorithms were very different, particularly when a re-orthogonalization scheme was used to compensate for the effects of round-off error. While RBCG was relatively insensitive to re-orthogonalization in terms of memory overhead, BCG with re-orthogonalization required nearly twice as much total memory as BCG without re-orthogonalization. RBCG was also nearly 22% faster than BCG.

The dual equivalent of BLanczos is RBLanczos which is a new algorithm which was derived in this article. The derivation of RBLanczos follows directly from BLanczos by exploiting the fundamental property that, under the assumption of a zero initial guess, the initial value of the cost function gradient is in the range of the adjoint of the generalized observation operator. As a result, all recurrence relations involving $n$-dimensional vectors in BLanczos can be transformed into recurrence relations involving $m$-dimensional vectors. The resulting algorithm requires the same number of matrix-vector products as BLanczos.

Numerical experiments comparing the performance of RBLanczos and Lanczos (the equivalent form of BLanczos which employs the factorized form of **B** as a preconditioner) in a regional-ocean 4D-Var assimilation system for the ROMS model were presented. As expected from theory, RBLanczos and Lanczos produce identical iterates to within machine precision. In experiments which employed both strong- and weak-constraint formulations of 4D-Var, the convergence properties of RBLanczos were also clearly superior to those of two other dual algorithms (dual-Lanczos and dual-MINRES) which have been proposed in the literature. In all algorithms, re-orthogonalization was found to be crucial for convergence and produced much smaller memory overhead with the dual algorithms than with the primal algorithms. It is also important to remark that only the dual approach was feasible in the weak-constraint 4D-Var experiment due to the very large size of the control vector.

The two ocean models and data assimilation systems used here differ significantly. On the one hand, NEMO

is configured as a coarse-resolution model of the global ocean and captures the dominant large-scale features of the global circulation. ROMS, on the other hand, is configured as an eddy-resolving model of the west coast of North America, and captures the energetic mesoscale circulation environment associated with the CCS. In addition, in this study, NEMO employs 3D-Var while ROMS employs 4D-Var with the model applied as either a strong or weak constraint. It is therefore noteworthy that RBCG and RBLanczos are robust in two very different circulation environments, and across a range of data assimilation approaches within the variational framework.

This work has illustrated the benefits of RBCG and RBLanczos from different perspectives. In this study, we considered only a single outer loop of the incremental algorithm and only first-level preconditioning using $\mathbf{B}$. Extensions of this work to account for multiple outer-loop iterations and second-level preconditioners based on limited-memory preconditioners such as quasi-Newton and Ritz (Gratton *et al.*, 2011b) will be described in a future publication. In view of the importance of re-orthogonalization in the 4D-Var experiments, it would also be of interest to investigate the use of FOM, which is more stable with respect to loss of orthogonality since it includes naturally an explicit full re-orthogonalization of the Krylov subspace basis.

**Acknowledgements**

This work is a contribution to the ADTAO and FILAOS projects which are financed by the RTRA STAE foundation. Additional support from the French National Research Agency (ANR) COSINUS programme (VODA project, no. ANR-08-COSI-016) and the French LEFE programme is also acknowledged. The authors also gratefully acknowledge the support of the US Office of Naval Research (N00014-10-1-0476, N00014-10-1-0322) and the National Science Foundation (OCE-1061434). Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the National Science Foundation. Kristian Mogensen is thanked for his help in implementing the minimization algorithms in NEMOVAR. Two anonymous reviewers provided helpful remarks for improving the article.

## Appendix A

### BLanczos and RBLanczos without re-orthogonalization

This appendix provides the direct versions of BLanczos and RBLanczos which can be used instead of Algorithms 1 and 4 to reduce memory requirements when re-orthogonalization is not desired.

The direct BLanczos algorithm is given in Algorithm 5. The direct RBLanczos algorithm can be derived following the basic approach used to derive RBLanczos Algorithm 4 from BLanczos Algorithm 1. This is done by using the relations between the vectors defined in $\mathbb{R}^n$ and $\mathbb{R}^m$ given by (30)–(32) and the additional equations

$$\mathbf{p}_i = \mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{p}}_i, \tag{A1}$$

$$\delta\mathbf{u}_i = \mathbf{B}\mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda}_i, \tag{A2}$$

that relate $m$-dimensional vectors $\widehat{\mathbf{p}}_i$ and $\boldsymbol{\lambda}_i$ with the vectors $\mathbf{p}_i$ and $\delta\mathbf{u}_i$ in Algorithm 5, for $i \geq 0$. It can be shown that relations (A1) and (A2) are satisfied by choosing

$$\widehat{\mathbf{p}}_i = (\widehat{\mathbf{v}}_i - \beta_i\widehat{\mathbf{p}}_{i-1})/\eta_i,$$
$$\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_{i-1} + \zeta_i\widehat{\mathbf{p}}_i,$$

where $\widehat{\mathbf{p}}_0 = \boldsymbol{\lambda}_0 = \mathbf{0}$ and the scalars $\beta_i$, $\eta_i$ and $\zeta_i$ are defined as

$$\beta_i = \|\widehat{\mathbf{w}}_{i-1}\|_{\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}} = \sqrt{\widehat{\mathbf{w}}_{i-1}^{\mathrm{T}}\mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{w}}_{i-1}}\,,$$
$$\eta_i = \alpha_i - (\beta_i/\eta_{i-1})\beta_i\,,$$
$$\zeta_i = -(\beta_i/\eta_{i-1})\zeta_{i-1}\,,$$

for $i > 1$. Direct substitution of these recurrence relations in BLanczos Algorithm 5 yields Algorithm 6.

**Algorithm 5:** BLanczos without re-orthogonalization

$\mathbf{v}_0 = \mathbf{0}$
$\delta\mathbf{u}_0 = \mathbf{0}$
$\mathbf{r}_0 = \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d}$
$\mathbf{t}_0 = \mathbf{B}\,\mathbf{r}_0$
$\beta_0 = \sqrt{\mathbf{t}_0^{\mathrm{T}}\mathbf{r}_0}$
$\zeta_1 = \beta_0$
$\mathbf{v}_1 = \mathbf{r}_0/\beta_0$
$\mathbf{z}_1 = \mathbf{t}_0/\beta_0$
$\beta_1 = 0$
$\gamma_1 = 0$
$\mathbf{p}_0 = 0$
**for** $i = 1, 2, \ldots, l$ **do**
  $\mathbf{q}_i = (\mathbf{v}_i + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\,\mathbf{z}_i) - \beta_i\mathbf{v}_{i-1}$
  $\alpha_i = \mathbf{q}_i^{\mathrm{T}}\mathbf{z}_i$
  **if** $i > 1$ **then**
    $\gamma_i = \beta_i/\eta_{i-1}$
    $\zeta_i = -\gamma_i\zeta_{i-1}$
  **end**
  $\eta_i = \alpha_i - \gamma_i\beta_i$
  $\mathbf{p}_i = (\mathbf{z}_i - \beta_i\mathbf{p}_{i-1})/\eta_i$
  $\delta\mathbf{u}_i = \delta\mathbf{u}_{i-1} + \zeta_i\mathbf{p}_i$
  $\mathbf{w}_i = \mathbf{q}_i - \alpha_i\mathbf{v}_i$
  $\mathbf{t}_i = \mathbf{B}\,\mathbf{w}_i$
  $\beta_{i+1} = \sqrt{\mathbf{t}_i^{\mathrm{T}}\mathbf{w}_i}$
  $\mathbf{v}_{i+1} = \mathbf{w}_i/\beta_{i+1}$
  $\mathbf{z}_{i+1} = \mathbf{t}_i/\beta_{i+1}$
**end**

## Appendix B

### Relationships between vectors in RBLanczos and BLanczos

Following GT09 and Gratton *et al.* (2009) in their derivation of RPCG, we suppose that

$$\mathbf{r}_0 \in \mathrm{range}(\mathbf{G}^{\mathrm{T}})$$

which is satisfied under the assumption that the initial guess $\delta\mathbf{u}_0 = \mathbf{0}$. In particular,

$$\mathbf{r}_0 = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0\,, \tag{B1}$$

where $\widehat{\mathbf{r}}_0 = \mathbf{R}^{-1}\mathbf{d} \in \mathbb{R}^m$.

**Algorithm 6:** RBLanczos without re-orthogonalization

1 $\widehat{\mathbf{r}}_0 = \mathbf{R}^{-1}\mathbf{d}$
2 $\boldsymbol{\lambda}_0 = \mathbf{0}$
3 $\widehat{\mathbf{t}}_0 = \mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\,\widehat{\mathbf{r}}_0$
4 $\beta_0 = \sqrt{\widehat{\mathbf{t}}_0^{\mathrm{T}}\widehat{\mathbf{r}}_0}$
5 $\zeta_1 = \beta_0$
6 $\widehat{\mathbf{v}}_1 = \widehat{\mathbf{r}}_0/\beta_0$
7 $\widehat{\mathbf{z}}_1 = \widehat{\mathbf{t}}_0/\beta_0$
8 $\beta_1 = 0$
9 $\widehat{\mathbf{v}}_0 = \mathbf{0}$
10 $\gamma_1 = 0$
11 $\widehat{\mathbf{p}}_0 = 0$
12 **for** $i = 1, 2, \ldots, l$ **do**
13 $\quad\widehat{\mathbf{q}}_i = (\widehat{\mathbf{v}}_i + \mathbf{R}^{-1}\widehat{\mathbf{z}}_i) - \beta_i\widehat{\mathbf{v}}_{i-1}$
14 $\quad\alpha_i = \widehat{\mathbf{q}}_i^{\mathrm{T}}\widehat{\mathbf{z}}_i$
15 $\quad$**if** $i > 1$ **then**
16 $\quad\quad\gamma_i = \beta_i/\eta_{i-1}$
17 $\quad\quad\zeta_i = -\gamma_i\zeta_{i-1}$
18 $\quad$**end**
19 $\quad\eta_i = \alpha_i - \gamma_i\beta_i$
20 $\quad\widehat{\mathbf{p}}_i = (\widehat{\mathbf{v}}_i - \beta_i\widehat{\mathbf{p}}_{i-1})/\eta_i$
21 $\quad\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_{i-1} + \zeta_i\widehat{\mathbf{p}}_i$
22 $\quad\widehat{\mathbf{w}}_i = \widehat{\mathbf{q}}_i - \alpha_i\widehat{\mathbf{v}}_i$
23 $\quad\widehat{\mathbf{t}}_i = \mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{w}}_i$
24 $\quad\beta_{i+1} = \sqrt{\widehat{\mathbf{t}}_i^{\mathrm{T}}\widehat{\mathbf{w}}_i}$
25 $\quad\widehat{\mathbf{v}}_{i+1} = \widehat{\mathbf{w}}_i/\beta_{i+1}$
26 $\quad\widehat{\mathbf{z}}_{i+1} := \widehat{\mathbf{t}}_i/\beta_{i+1}$
27 **end**
28 $\delta\mathbf{u}_l = \mathbf{B}\mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda}_l$

Let us define the vectors in $\mathbb{R}^m$ such that

$$\widehat{\mathbf{v}}_0 = \mathbf{0},$$
$$\widehat{\mathbf{t}}_0 = \mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{r}}_0\,, \tag{B2}$$
$$\widehat{\mathbf{v}}_1 = \widehat{\mathbf{r}}_0/\beta_0\,, \tag{B3}$$
$$\widehat{\mathbf{z}}_1 = \widehat{\mathbf{t}}_0/\beta_0\,, \tag{B4}$$

where $\beta_0 = \sqrt{\widehat{\mathbf{r}}_0^{\mathrm{T}}\widehat{\mathbf{t}}_0}$. From the definitions (B1)–(B4), it is possible to derive the following relations from BLanczos:

$$\mathbf{G}\mathbf{t}_0 = \widehat{\mathbf{t}}_0\,,$$
$$\mathbf{v}_1 = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{v}}_1\,, \tag{B5}$$
$$\mathbf{G}\mathbf{z}_1 = \widehat{\mathbf{z}}_1\,, \tag{B6}$$
$$\begin{aligned}\mathbf{q}_1 &= \mathbf{v}_1 + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\mathbf{z}_1\\ &= \mathbf{G}^{\mathrm{T}}\big(\widehat{\mathbf{v}}_1 + \mathbf{R}^{-1}\widehat{\mathbf{z}}_1\big)\,,\end{aligned} \tag{B7}$$
$$\begin{aligned}\mathbf{w}_1 &= \mathbf{q}_1 - \alpha_1\mathbf{v}_1\\ &= \mathbf{G}^{\mathrm{T}}\big[(\widehat{\mathbf{v}}_1 + \mathbf{R}^{-1}\widehat{\mathbf{z}}_1) - \alpha_1\widehat{\mathbf{v}}_1\big]\,,\end{aligned} \tag{B8}$$
$$\begin{aligned}\mathbf{t}_1 &= \mathbf{B}\mathbf{w}_1\\ &= \mathbf{B}\mathbf{G}^{\mathrm{T}}\big[(\widehat{\mathbf{v}}_1 + \mathbf{R}^{-1}\widehat{\mathbf{z}}_1) - \alpha_1\widehat{\mathbf{v}}_1\big]\,.\end{aligned} \tag{B9}$$

Equations (B7)–(B9) can be written as

$$\mathbf{q}_1 = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{q}}_1, \tag{B10}$$
$$\mathbf{w}_1 = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{w}}_1, \tag{B11}$$
$$\mathbf{G}\mathbf{t}_1 = \widehat{\mathbf{t}}_1, \tag{B12}$$

where $\widehat{\mathbf{q}}_1 = \widehat{\mathbf{v}}_1 + \mathbf{R}^{-1}\widehat{\mathbf{z}}_1$, $\widehat{\mathbf{w}}_1 = \widehat{\mathbf{q}}_1 - \alpha_1\widehat{\mathbf{v}}_1$, $\alpha_1 = \widehat{\mathbf{q}}_1^{\mathrm{T}}\widehat{\mathbf{z}}_1$ and $\widehat{\mathbf{t}}_1 = \mathbf{G}\mathbf{B}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{w}}_1$.

We now prove by induction that the relations given by (B5)–(B6) and (B10)–(B12) also hold for $i > 1$. First, assume that the relations are satisfied for a given $i$:

$$\mathbf{v}_i = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{v}}_i, \tag{B13}$$
$$\mathbf{G}\mathbf{z}_i = \widehat{\mathbf{z}}_i\,,$$
$$\mathbf{q}_i = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{q}}_i\,,$$
$$\mathbf{w}_i = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{w}}_i\,, \tag{B14}$$
$$\mathbf{G}\mathbf{t}_i = \widehat{\mathbf{t}}_i\,. \tag{B15}$$

We now show that the relations hold for $i+1$. From lines 17 and 18 of Algorithm 1 for BLanczos, we have, using (B14) and (B15),

$$\mathbf{v}_{i+1} = \mathbf{w}_i/\beta_{i+1} = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{w}}_i/\beta_{i+1}\,, \tag{B16}$$
$$\mathbf{G}\mathbf{z}_{i+1} = \mathbf{G}\mathbf{t}_i/\beta_{i+1} = \widehat{\mathbf{t}}_i/\beta_{i+1}\,. \tag{B17}$$

Defining

$$\widehat{\mathbf{v}}_{i+1} = \widehat{\mathbf{w}}_i/\beta_{i+1}\,,$$
$$\widehat{\mathbf{z}}_{i+1} = \widehat{\mathbf{t}}_i/\beta_{i+1}\,,$$

we obtain from (B16) and (B17) that

$$\mathbf{v}_{i+1} = \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{v}}_{i+1}\,, \tag{B18}$$
$$\mathbf{G}\mathbf{z}_{i+1} = \widehat{\mathbf{z}}_{i+1}\,. \tag{B19}$$

From line 13 of Algorithm 1 and using the relations (B13), (B18) and (B19), we have

$$\begin{aligned}\mathbf{q}_{i+1} &= (\mathbf{v}_{i+1} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\mathbf{z}_{i+1}) - \beta_{i+1}\mathbf{v}_i\\ &= (\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{v}}_{i+1} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\widehat{\mathbf{z}}_{i+1}) - \beta_{i+1}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{v}}_i\\ &= \mathbf{G}^{\mathrm{T}}\big[(\widehat{\mathbf{v}}_{i+1} + \mathbf{R}^{-1}\widehat{\mathbf{z}}_{i+1}) - \beta_{i+1}\widehat{\mathbf{v}}_i\big]\\ &= \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{q}}_{i+1},\end{aligned} \tag{B20}$$

where

$$\widehat{\mathbf{q}}_{i+1} = (\widehat{\mathbf{v}}_{i+1} + \mathbf{R}^{-1}\widehat{\mathbf{z}}_{i+1}) - \beta_{i+1}\widehat{\mathbf{v}}_i\,.$$

Similarly, from line 13 of Algorithm 1 and using the relations (B18) and (B20), we have

$$\begin{aligned}\mathbf{w}_{i+1} &= \mathbf{q}_{i+1} - \alpha_{i+1}\mathbf{v}_{i+1}\\ &= \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{q}}_{i+1} - \alpha_{i+1}\mathbf{G}^{\mathrm{T}}\widehat{\mathbf{v}}_{i+1}\\ &= \mathbf{G}^{\mathrm{T}}(\widehat{\mathbf{q}}_{i+1} - \alpha_{i+1}\widehat{\mathbf{v}}_{i+1})\\ &= \mathbf{G}^{\mathrm{T}}\widehat{\mathbf{w}}_{i+1}\,,\end{aligned}$$

where

$$\widehat{\mathbf{w}}_{i+1} = \widehat{\mathbf{q}}_{i+1} - \alpha_{i+1}\widehat{\mathbf{v}}_{i+1}\,,$$

which completes the proof.

## References

Axelsson O. 1996. *Iterative Solution Methods*. Cambridge University Press: Cambridge, UK.

Balmaseda MA, Mogensen K, Weaver AT. 2013. Evaluation of the ECMWF ocean reanalysis ORAS4. *Q. J. R. Meteorol. Soc.* DOI: 10.1002/qj.2063.

Bennett AF. 2002. *Inverse Modelling of the Ocean and Atmosphere*. Cambridge University Press: Cambridge, UK.

Broquet G, Edwards CA, Moore AM, Powell BS, Veneziani M, Doyle JD. 2009a. Application of 4D-variational data assimilation to the California Current System. *Dyn. Atmos. Oceans* **48**: 69–91.

Broquet G, Moore AM, Arango HG, Edwards CA, Powell BS. 2009b. Ocean state and surface forcing correction using the ROMS-IS 4DVAR data assimilation system. *Mercator Ocean Quarterly Newsletter* **34**: 5–13.

Buehner M. 2005. Ensemble-derived stationary and flow-dependent background-error covariances: evaluation in a quasi-operational NWP setting. *Q. J. R. Meteorol. Soc.* **131**: 1013–1043.

Cardinali C, Pezzulli S, Andersson E. 2004. Influence-matrix diagnostic of a data assimilation system. *Q. J. R. Meteorol. Soc.* **130**: 2767–2786.

Chan TF, Chow E, Saad Y, Yeung MC. 1999. Preserving symmetry in preconditioned Krylov subspace methods. *SIAM J. Sci. Comput.* **20**: 568–581.

Chapman DC. 1985. Numerical treatment of cross-shelf open boundaries in a barotropic coastal ocean model. *J. Phys. Oceanogr.* **15**: 1060–1075.

Checkley DM, Barth JA. 2009. Patterns and process in the California Current system. *Prog. Oceanogr.* **83**: 49–64.

Chien CS, Chang SL. 2003. Application of the Lanczos algorithm for solving the linear systems that occur in continuation problems. *Numer. Linear Algebra Appl.* **10**: 335–355. DOI: 10.1002/nla.306.

Chua BS, Bennett AF. 2001. An inverse ocean modeling system. *Ocean Model.* **3**: 137–165.

Cohn S, Da Silva A, Guo J, Sienkiewicz M, Lamich D. 1998. Assessing the effects of data selection with the DAO physical-space statistical analysis system. *Mon. Weather. Rev.* **126**: 2913–2926.

Courtier P. 1997. Dual formulation of four-dimensional variational assimilation. *Q. J. R. Meteorol. Soc.* **123**: 2449–2461.

Courtier P, Thépaut J-N, Hollingsworth A. 1994. A strategy for operational implementation of 4D-Var using an incremental approach. *Q. J. R. Meteorol. Soc.* **120**: 1367–1388.

Da Silva A, Pfaendtner J, Guo J, Sienkiewicz M, Cohn S. 1995. Assessing the effects of data selection with the DAO physical-space statistical analysis system. In: *Proceedings of the 2nd WMO Symposium on assimilation of observations in meteorology and oceanography*. World Meteorological Organization: Geneva, Switzerland. 273–278.

Daley R, Barker E. 2001. NAVDAS: Formulation and diagnostics. *Mon. Weather Rev.* **120**: 869–883.

Derber J, Rosati A. 1989. A global oceanic data assimilation system. *J. Phys. Oceanogr.* **19**: 1333–1347.

Desroziers G, Berre L. 2012. Accelerating and parallelizing minimizations in ensemble and deterministic variational assimilations. *Q. J. R. Meteorol. Soc.* DOI: 10.1002/qj.1886 In press.

Doyle JD, Jiang Q, Chao Y, Farrara J. 2009. High-resolution atmospheric modeling over the Monterey Bay during AOSN II. *Deep Sea Res. II* **56**: 87–99.

Egbert GD, Bennett AF, Foreman MCG. 1994. TOPEX/POSEIDON tides estimated using a global inverse method. *J. Geophys. Res.* **99**: 24821–24852.

El Akkraoui A, Gauthier P. 2010. Convergence properties of the primal and dual forms of variational data assimilation. *Q. J. R. Meteorol. Soc.* **136**: 107–115.

El Akkraoui A, Gauthier P, Pellerin S, Buis S. 2008. Intercomparison of the primal and dual formulations of variational data assimilation. *Q. J. R. Meteorol. Soc.* **134**: 1015–1025.

El Akkraoui A, Trémolet Y, Todling R. 2012. Preconditioning of variational data assimilation and the use of a bi-conjugate gradient method. *Q. J. R. Meteorol. Soc.* DOI:10.1002/qj.1997. In press.

Evensen G. 2009. *Data Assimilation: The Ensemble Kalman Filter*. Springer Verlag: Berlin.

Fairall CW, Bradley EF, Godfrey JS, Wick GA, Ebson JB, Young GS. 1996a. Cool-skin and warm layer effects on the sea surface temperature. *J. Geophys. Res.* **101**: 1295–1308.

Fairall CW, Bradley EF, Rogers DP, Ebson JB, Young GS. 1996b. Bulk parameterization of air-sea fluxes for tropical ocean global atmosphere coupled-ocean atmosphere response experiment. *J. Geophys. Res.* **101**: 3747–3764.

Fisher M. 2003. Background-error covariance modelling. In: *Proceedings of Seminar on Recent Developments in Data Assimilation for Atmosphere and Ocean*. ECMWF: Reading, UK. 87–101.

Fisher M, Nocedal J, Trémolet Y, Wright SJ. 2009. Data assimilation in weather forecasting: A case study in PDE-constrained optimization. *Optim. Eng.* **10**: 409–426.

Flather RA. 1976. A tidal model of the northwest European continental shelf. *Mem. Soc. R. Sci. Liege* **10**: 141–164.

Gauthier P, Thépaut JN. 2001. Impact of the digital filter as a weak constraint in the pre-operational 4D-VAR assimilation system of Météo-France. *Mon. Weather Rev.* **129**: 2089–2102.

Gelaro R, Zhu Y. 2009. Examination of observation impacts derived from observing system experiments (OSEs) and adjoint models. *Tellus* **61A**: 179–193.

Golub GH, Van Loan CF. 1996. *Matrix Computations*. (third ed.) Johns Hopkins University Press: Baltimore, MA.

Gratton S, Tshimanga J. 2009. An observation-space formulation of variational assimilation using a Restricted Preconditioned Conjugate-Gradient algorithm. *Q. J. R. Meteorol. Soc.* **135**: 1573–1585.

Gratton S, Lawless A, Nichols NK. 2007. Approximate Gauss-Newton methods for nonlinear least-squares problems. *SIAM J. Optim.* **18**: 106–132.

Gratton S, Toint PL, Tshimanga J. 2009. 'Inexact range-space Krylov solvers for linear systems arising from inverse problems'. Technical Report 09/20, Department of Mathematics, FUNDP, University of Namur: Belgium.

Gratton S, Laloyaux P, Sartenaer A, Tshimanga J. 2011a. A reduced- and limited-memory preconditioned approach for the 4D-Var data-assimilation problem. *Q. J. R. Meteorol. Soc.* **137**: 452–466.

Gratton S, Sartenaer A, Tshimanga J. 2011b. On a class of limited-memory preconditioners for large-scale linear systems with multiple right-hand sides. *SIAM J. Optim.* **21**: 912–935.

Gratton S, Gürol S, Toint PL. 2013. Preconditioning and globalizing conjugate gradients in dual space for quadratically penalized nonlinear least-squares problems. *Comput. Optim. Appl.* **54**: 1–25.

Haidvogel DB, Arango HG, Budgell WP, Cornuelle BD, Curchitser E, Di Lorenzo E, Fennel K, Geyer WR, Hermann AJ, Lanerolle L, Levin J, McWilliams JC, Miller AJ, Moore AM, Powell TM, Shchepetkin AF, Sherwood CR, Signell RP, Warner JC, Wilkin J. 2008. Ocean forecasting in terrain-following coordinates: Formulation and skill assessment of the Regional Ocean Modeling System. *J. Comput. Phys.* **227**: 3595–3624.

Hickey BM. 1998. Coastal oceanography of western North America from the tip of Baja California to Vancouver Island. *The Sea* **11**: 345–393.

Horn RA, Johnson CR. 1999. *Matrix Analysis*. Cambridge University Press: Cambridge, UK.

Lawless AS, Gratton S, Nichols NK. 2005. Approximate iterative methods for variational data assimilation. *Int. J. Numer. Meth. Fluids.* **47**: 1129–1135.

Liu WT, Katsaros KB, Businger JA. 1979. Bulk parameterization of the air–sea exchange of heat and water vapor including the molecular constraints at the interface. *J. Atmos. Sci.* **36**: 1722–1735.

Lorenc AC. 1988. Optimal nonlinear objective analysis. *Q. J. R. Meteorol. Soc.* **114**: 205–240.

Lorenc AC. 1997. Development of an operational variational assimilation scheme. *J. Meteorol. Soc. Japan.* **75**: 339–346.

Madec G. 2008. 'NEMO ocean engine'. Technical Report 27, Note du Pôle de modélisation, Institut Pierre-Simon Laplace (IPSL), France. http://www.nemo-ocean.eu.

Marchesiello P, McWilliams JC, Shchepetkin AF. 2001. Open boundary conditions for long-term integration of regional oceanic models. *Ocean Model.* **3**: 1–20.

Mogensen K, Balmaseda MA, Weaver AT, Martin M, Vidard A. 2009. 'NEMOVAR: A variational data assimilation system for the NEMO model'. *ECMWF Newsletter* **120**: 17–22.

Mogensen K, Balmaseda MA, Weaver AT. 2012. 'The NEMOVAR ocean data assimilation system as implemented in the ECMWF ocean analysis for System 4'. Tech. Report 668, ECMWF: Reading, UK.

Moore AM, Arango HG, Broquet G, Powell BS, Zavala-Garay J, Weaver AT. 2011a. The Regional Ocean Modeling System (ROMS) 4-dimensional variational data assimilation systems. Part I: System overview and formulation. *Prog. Oceanogr.* **91**: 34–49.

Moore AM, Arango HG, Broquet G, Edwards C, Veneziani M, Powell B, Foley D, Costa D, Robinson P. 2011b. The Regional Ocean Modeling system (ROMS) 4-dimensional variational data assimilation systems. Part II: Performance and application to the California Current System. *Prog. Oceanogr.* **91**: 50–73.

Moore AM, Arango HG, Broquet G, Edwards C, Veneziani M, Powell B, Foley D, Doyle JD, Costa D, Robinson P. 2011c. The Regional Ocean Modeling system (ROMS) 4-dimensional variational data assimilation systems. Part III: Observation impact and observation sensitivity in the California Current System. *Prog. Oceanogr.* **91**: 74–94.

 

Nocedal J, Wright SJ. 2006. *Numerical Optimization*. *Series in Operations Research*, Springer Verlag: Berlin.

Nour-Omid B, Parlett BN, Raefsky A. 1988. Comparison of Lanczos with conjugate gradient using element preconditioning. In: *Proceedings of First International Symposium on Domain Decomposition Methods for Partial Differential Equations*. SIAM: Philadelphia, PA. 250–260.

Paige CC, Saunders MA. 1975. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.* **12**: 617–629.

Papadrakakis M, Smerou S. 1990. A new implementation of the Lanczos method in linear problems. *Int. J. Numer. Meth. Eng.* **29**: 141–159. DOI: 10.1002/nme.1620290110.

Parlett BN. 1980. *The symmetric eigenvalue problem*. Prentice-Hall: Englewood Cliffs, NJ, USA. Reprinted (1998) as *Classics in Applied Mathematics* **20**: SIAM: Philadelphia, PA.

Purser RJ, Wu WW, Parrish DF, Roberts NM. 2003. Numerical aspects of the application of recursive filters to variational statistical analysis. Part I: Spatially homogeneous and isotropic Gaussian covariances. *Mon. Weather Rev.* **131**: 1524–1535.

Saad Y. 1996. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company: Boston, MA.

Sasaki YK. 1970. Some basic formalisms in numerical variational analysis. *Mon. Weather. Rev.* **98**: 875–883.

Shchepetkin AF, McWilliams JC. 2003. A method for computing horizontal pressure-gradient force in an oceanic model with a nonaligned vertical grid. *J. Geophys. Res.* **108**(C3): DOI: 10.1029/2001/JC001047.

Shchepetkin AF, McWilliams JC. 2005. The Regional Oceanic Modeling System (ROMS): a split explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Model.* **9**: 347–404.

Tarantola A. 2005. *Inverse Problem Theory. Methods for Data Fitting and Model Parameter Estimation*. SIAM: Philadelphia, PA.

Trémolet Y. 2006. Accounting for an imperfect model in 4D-Var. *Q. J. R. Meteorol. Soc.* **132**: 2483–2504.

Tshimanga J, Gratton S, Weaver AT, Sartenaer A. 2008. Limited-memory preconditioners with application to incremental four-dimensional variational data assimilation. *Q. J. R. Meteorol. Soc.* **134**: 751–769.

Veneziani M, Edwards CA, Doyle JD, Foley D. 2009a. A central California coastal ocean modeling study: 1. Forward model and the influence of realistic versus climatological forcing. *J. Geophys. Res.* **114**(C04015): DOI: 10.1029/2008JC004774.

Veneziani M, Edwards CA, Moore AM. 2009b. A central California coastal ocean modeling study: 2. Adjoint sensitivities to local and remote forcing mechanisms. *J. Geophys. Res.* **114**(C04020): DOI: 10.1029/2008JC004775.

Wang X, Synder C, Hamill TM. 2007. On the theoretical equivalence of differently proposed ensemble–3DVAR hybrid analysis schemes. *Mon. Weather Rev.* **135**: 222–227.

Wang X, Barker DM, Synder C, Hamill TM. 2008. A hybrid ETKF–3DVAR data assimilation scheme for the WRF model. Part I: Observing system simulation experiment. *Mon. Weather Rev.* **136**: 5116–5131.

Weaver AT, Mirouze I. 2013. On the diffusion equation and its application to isotropic and anisotropic correlation modelling in variational assimilation. *Q. J. R. Meteorol. Soc.* **139**: 242–260.

Weaver AT, Deltel C, Machu E, Ricci S, Daget N. 2005. A multivariate balance operator for variational ocean data assimilation. *Q. J. R. Meteorol4. Soc.* **131**: 3605–3625.

Weaver AT, Vialard J, Anderson DLT. 2003. Three- and four-dimensional variational assimilation with a general circulation model of the tropical Pacific ocean. Part 1: Formulation, internal diagnostics and consistency checks. *Mon. Weather Rev.* **131**: 1360–1378.

Wunsch C, Heimbach P. 2007. Practical global ocean state estimation. *Physica D* **230**: 197–208.

 

# CHAPTER 8

## Conclusions and Future Directions

The thesis concentrates on the solution of the nonlinear least squares problem, in which there are many fewer observations than variables to be estimated, and where a quadratic regularization term has therefore to be introduced in order to guarantee uniqueness of the solution. Within this study, a solution method based on a Gauss-Newton technique, made globally convergent with a trust-region strategy is considered. Each member of the sequence of linear least-squares problems involved in the method is solved iteratively by the conjugate gradient method (or the Lanczos method), appropriately truncated by the Steihaug-Toint strategy, and which is accelerated by limited memory preconditioners (Gratton, Sartenaer and Tshimanga, 2011).

It has been recently shown that it is possible to use duality theory and rewrite the linear least-squares solver in an equivalent algorithm (in exact arithmetic) in which all vectors of the short-term recurrences are represented by vectors of dimension $m$, $m$ being the number of physical observations. Two proposed dual approaches, called PSAS and RPCG, are shown to differ in the way they define the scalar product in dual space (Gratton, Gürol and Toint, 2013). It is argued that RPCG is preferable to PSAS because it maintains the convergence properties of the initial Gauss-Newton process (Gratton and Tshimanga, 2009).

In this thesis, a further step is taken in making the RPCG dual solver relevant to practice for large scale, nonlinear problems. A general LMP is first introduced for the case when the system matrix of a sequence of linear systems (arising from solving the nonlinear least squares problem with a Gauss-Newton method) is assumed to be constant. This leads to solving a convergent sequence of linear systems with multiple right hand sides. Later particular LMPs in dual space such as the Ritz-LMP, the spectral LMP and the quasi-Newton LMP are introduced. All these techniques are implemented in such a way that RPCG and the primal approach generate the same sequence of iterates. A further advantage of the proposed dual approach in the common situation where $m \ll n$, is that storing vectors for the preconditioner or performing re-orthogonalization is computationally much cheaper than with standard

primal algorithms, making both of these techniques applicable in realistic cases.

These preconditioners are later adapted to the case where a sequence of slowly varying linear systems has to be solved as in the Gauss-Newton process by introducing an efficient implementation of the Steihaug-Toint truncation of CG in dual space. Numerical experiments are presented on a test problem based on the heat equation which can be considered as an idealized data assimilation system. With the numerical experiments, it can be seen that the dual solver performs well on mildly nonlinear systems and can also be safely used in severe cases using the proposed flexible algorithm. This algorithm enables a global convergence disregarding the starting point of the iterations, at the cost of one function evaluation per outer loop. However the robustness that is gained with this algorithm is especially important for operational systems.

In addition to designing memory efficient preconditioners in dual space, an alternative memory efficient preconditioning is proposed in primal space by using the PCGIF algorithm. This algorithm avoids expensive matrix vector products (especially for large-scale problems) with the inverse of the *a priori* error covariance matrix $B$ and does not require the factorization of the preconditioner. All the particular LMPs are adapted to be used in the PCGIF algorithm.

Knowing that Lanczos type algorithms are widely used in data assimilation, Lanczos versions of all CG variant algorithms are also derived both in primal and dual space. The implementation of the limited memory preconditioners are explained in detail for these Lanczos type algorithms.

Important applications of the problem stated in this thesis arise in environmental sciences like meteorology or oceanography, where the estimated vector is the initial state of a dynamical system which is then integrated forward in time to produce a forecast. In this thesis, RPCG and its Lanczos version RPLanczos were implemented in two different variational ocean data assimilation systems: NEMOVAR and ROMS, where the *a priori* error covariance matrix $B$ is used as a preconditioner and a single outer loop is considered. NEMOVAR is based on a global ocean model and captures the dominant large-scale features of the global circulation whereas ROMS is based on a regional model and is configured as an eddy-resolving model of the west coast of North America, and captures the energetic meso-scale circulation environment. ROMS accounts for model errors also. Therefore, RPCG and RPLanczos are validated in two ocean systems which represent the main approaches for such systems.

There are some issues worth further exploration. It would be interesting to further explore globalization strategies by developing algorithms that are similar in spirit to the Moré-Sorensen (Conn, Gould and Toint, 2000, Section 7.3) approach, or by considering techniques based on cubic regularization (Cartis, Gould and Toint, 2009). In both cases, as for the Steihaug-Toint truncated CG algorithm we expect that keeping the symmetry with respect to the inner product in dual space may be a key issue to obtain a robust algorithm. A further improvement of the algorithms presented in ths thesis is to design computationally efficient criteria to detect the loss of symmetry with respect to the inner product in dual space and thereby avoid the computation of a magical step that would not result in a decrease of the objective function.

It would be also interesting to further explore the adaptation of the proposed algorithms when the errors of assimilated data (observations and a priori background estimates) have non-Gaussian probability density functions (pdfs) (Bocquet, 2007).

In view of the importance of re-orthogonalization in ocean data assimilation systems the use of the Full Orthogonalization Method (FOM), which is more stable with respect to loss of orthogonality, can also be investigated.

The implementation of the particular LMPs such as the quasi-Newton LMP and the Ritz LMP in dual space with multiple outer loops in ocean data assimilation systems: NEMOVAR and ROMS, and atmospheric data assimilation system: OOPS (Object Oriented Prediction System) developed by ECMWF (European Center for Medium-Range Weather Forecast) are of interest for future studies.

# Bibliography


Akkraoui AE, Gauthier P, Pellerin S, Buis S. 2008. Intercomparison of the primal and dual formulations of variational data assimilation. *Quarterly Journal of the Royal Meteorological Society* **134**: 1015–1025.

Arioli M, Orban D. 2012. Iterative methods for symmetric quasi-definite linear systems. Cahier du GERAD G-2012-xx, GERAD, Montréal, Québec, Canada. In preparation.

Axelsson O. 1996. *Iterative solution methods*. Cambridge University Press: Cambridge, England.

Barrett R, Berry M, Chan TF, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C, van der Vorst HA. 1994. *Templates for the solution of linear systems: Building blocks for iterative methods, 2nd edition*. SIAM: Philadelphia, PA.

Benzi M. 2002. Preconditioning techniques for large linear systems: A survey. *J. Comput. Phys.* **182**: 418–477.

Bertsekas DP. 1995. *Nonlinear programming*. Athena Scientific: Belmont, Massachussetts, USA.

Björck Å. 1996. *Numerical methods for least squares problems*. SIAM: Philadelphia, USA.

Bocquet M. 18 December 2007. Modélisation inverse et assimilation de données non-gaussiennes pour les traceurs atmosphériques. application a etex, algésiras et tchernobyl. Technical report, Universit Paris VI. Habilitation  Diriger des Recherches, granted by Universit Paris VI, Pierre et Marie Curie.

Boyd S, Vandenberghe L. 2004. *Convex optimization*. Cambridge University Press: Cambridge, England.

Cartis C, Gould NIM, Toint PL. 2009. Adaptive cubic overestimation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Mathematical Programming, Series A* DOI: 10.1007/s10107-009-0286-5, 51 pages.

Chan TF, Chow E, Saad Y, Yeung MC. 1999. Preserving symmetry in preconditioned krylov subspace methods. *SIAM J. Sci. Comput* **20**: 568–581.

Cohn S, Da Silva A, Guo J, Sienkiewicz M, Lamich D. 1998. Assessing the effects of data selection with the DAO physical-space statistical analysis system. *Mon. Weather. Rev.* **126**: 2913–2926.

Conn AR, Gould NIM, Toint PL. 2000. *Trust-region methods*. No. 01 in: MPS-SIAM Series on Optimization, SIAM: Philadelphia, USA.

Courtier P. 1997. Dual formulation of four-dimensional variational assimilation. *Quarterly Journal of the Royal Meteorological Society* **123**: 2449–2461.

Courtier P, Thépaut JN, Hollingsworth A. 1994. A strategy for operational implementation of 4D-Var using an incremental approach. *Quarterly Journal of the Royal Meteorological Society* **120**: 1367–1388.

Craig JE. 1955. The N-step iteration procedures. *J. Math and Phys.* **34**(1): 64–73.

Cullum J, Willoughby RA. 1980. The Lanczos phenomenon—an interpretation based upon conjugate gradient optimization. *Linear Algebra and its Applications* **29**: 63–90.

Da Silva A, Pfaendtner J, Guo J, Sienkiewicz M, Cohn S. 1995. Assessing the effects of data selection with the DAO physical-space statistical analysis system. In: *Proceedings of the 2nd WMO Symposium on assimilation of observations in meteorology and oceanography*. World Meteorological Organization: Geneva, pp. 273–278.

Daley R. 1991. *Atmospheric data analysis*. Cambridge University Press.

Dennis JE, Schnabel RB. 1983. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall: Englewood Cliffs, NJ, USA. Reprinted as *Classics in Applied Mathematics 16*, SIAM, Philadelphia, USA, 1996.

Derber J, Rosati A. 1989. A global oceanic data assimilation system. *Journal of physical oceanography* **19**: 1333–1347.

Egbert GD, Bennett AF, Foreman MCG. 1994. Topex/poseidon tides estimated using a global inverse method. *J. Geophys. Res.* **99**(24): 821–24,852.

El Akkroui A, Gauthier P, Pellerin S, Buis S. 2008. Intercomparison of the primal and dual formulations of variational data assimilation. *Quarterly Journal of the Royal Meteorological Society* **134**(633): 1015–1025.

Engeli M, Ginsburg T, Rutishauser H, Stiefel E. 1960. Refined iterative methods for computation of the solution and the eigenvalues of self-adjoint bondary value problems. (mitteilungen aus dem institut für angewandte mathematik der eth zürich, nr. 8.) 107 s. basel/stuttgart 1959. birkhäuser verlag. preis brosch. dm 17,. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* **40**(10-11): 525–525, doi:10.1002/zamm.19600401041, URL http://dx.doi.org/10.1002/zamm.19600401041.

Eriksson J, Wedin PA, Gulliksson M. 1998. Regularization methods for nonlinear least squares problems. part i: Exactly rank-deficiency. Technical report.

Fisher M. 1998. Minimization algorithms for variational data assimilation. In: *Recent Developments in Numerical Methods for Atmospheric Modelling*. European Center for Medium-Range Weather Forecasts: Reading, UK, pp. 364–385.

Golub GH, Van Loan CF. 1989. *Matrix computations*. Johns Hopkins University Press: Baltimore, second edn.

Golub GH, Van Loan CF. 1996. *Matrix computations*. Johns Hopkins University Press: Baltimore, third edn.

Gould NIM, Lucidi S, Roma M, Toint PL. 1999. Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization* **9**(2): 504–525.

Gratton S, Gürol S, Toint PL. 2013. Preconditioning and globalizing conjugate gradients in dual space for quadratically penalized nonlinear-least squares problems. *Computational Optimization and Applications* **54**: 1–25, doi:10.1007/s10589-012-9478-7.

Gratton S, Lawless A, Nichols NK. 2007. Approximate Gauss-Newton methods for nonlinear least-squares problems. *SIAM Journal on Optimization* **18**: 106–132.

Gratton S, Sartenaer A, Tshimanga J. 2011. On a class of limited memory preconditioners for large scale linear systems with multiple right-hand sides. *SIAM Journal on Optimization* **21**(3): 912–935.

Gratton S, Toint PL, Tshimanga J. 2009. Inexact range-space Krylov solvers for linear systems arising from inverse problems. Technical Report 09/20, Department of Mathematics, FUNDP - University of Namur, Namur, Belgium.

Gratton S, Tshimanga J. 2009. An observation-space formulation of variational assimilation using a restricted preconditioned conjugate-gradient algorithm. *Quarterly Journal of the Royal Meteorological Society* **135**: 1573–1585.

Hestenes MR, Stiefel E. 1952. Methods of conjugate gradients for solving linear systems. *Journal of the National Bureau of Standards* **49**: 409–436.

Kalnay E. 2003. *Atmospheric modeling, data assimilation and predictability*. Cambridge University Press.

Kaniel S. 1966. Estimates for some computational techniques in linear algebra. *Mathematics of Computation* **20**(95): 369–378.

Kelley CT. 1999. *Iterative methods for optimization*. Frontiers in Applied Mathematics, SIAM: Philadelphia, USA.

Lampe J, Rojas M, Sorensen DC, Voss H. 2011. Accelerating the LSTRS algorithm. *SIAM Journal on Scientific Computing* **33**(1): 175–194.

Lanczos C. 1950. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of research of the National Bureau of Standards B* **45**: 225–280.

Levenberg K. 1944. A method for the solution of certain problems in least squares. *Quarterly Journal on Applied Mathematics* **2**: 164–168.

Mangasarian OL. 1969. *Nonlinear programming*. McGraw-Hill: New York, USA. Reprinted as *Classics in Applied Mathematics 10*, SIAM, Philadelphia, USA, 1994.

Marquardt D. 1963. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics* **11**: 431–441.

Meurant G. 2006. *The Lanczos and conjugate-gradient algorithms*. SIAM: Philadelphia, USA.

Moore AM, Arango HG, Broquet G, Powell BS, Zavala-Garay J, Weaver AT. 2011. The regional ocean modeling system (roms) 4-dimensional variational data assimilation systems. part i: System overview and formulation. *Progress in Oceanography* **91**: 34–49.

Morales JL, Nocedal J. 1999. Automatic preconditioning by limited memory quasi-newton updating. *SIAM Journal on Optimization* **10**: 1079–1096.

Morrison DD. 1960. Methods for nonlinear least squares problems and convergence proofs. In: *Proceedings of the Seminar on Tracking Programs and Orbit Determination*, Lorell J, Yagi F (eds). Jet Propulsion Laboratory: Pasadena, USA, pp. 1–9.

Nocedal J, Wright SJ. 2006. *Numerical optimization*. Series in Operations Research, Springer Verlag: Heidelberg, Berlin, New York.

Nour-Omid B, Parlett BN, Raefsky A. 1988. *Comparison of lanczos with conjugate gradient using element preconditioning*. SIAM.

Paige CC, Saunders MA. 1975. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis* **12**(4): 617–629.

Paige CC, Saunders MA. 1982. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software* **8**: 43–71.

Parlett BN. 1980. *The symmetric eigenvalue problem*. Prentice-Hall: Englewood Cliffs, NJ, USA. Reprinted as *Classics in Applied Mathematics 20*, SIAM, Philadelphia, USA, 1998.

Reid JK. 1971. On the method of conjugate gradients for the solution of large sparse linear equations. In: *Large sparse sets of linear equations*, Reid JK (ed). Academic Press: London, pp. 231–254.

Rey C, Risler F. 1998. A rayleigh–ritz preconditioner for the iterative solution to large scale nonlinear problems. *Numerical Algorithms* **17**: 279–311, URL http://dx.doi.org/10.1023/A:1016680306741.

Rockafellar RT. 1970. *Convex analyis*. Princeton University Press: Princeton, USA.

Rockafellar RT. 1976. Solving a nonlinear programming problem by way of a dual problem. Symposia Mathematica, Vol. 27.

Rojas M, Santos SA, Sorensen DC. 2008. Algorithm 873: LSTRS: MATLAB software for large-scale trust-region subproblems and regularization. *ACM Transactions on Mathematical Software* **34**(2): 11.

Roux FX. 1989. Acceleration of the outer conjugate gradient by reorthogonalization for a domain decomposition method for structural analysis problems. In: *Proceedings of the 3rd international conference on Supercomputing*, ICS '89. ACM: New York, NY, USA, ISBN 0-89791-309-4, pp. 471–476, doi:http://doi.acm.org/10.1145/318789.318895, URL http://doi.acm.org/10.1145/318789.318895.

Saad Y. 1996. *Iterative methods for sparse linear systems*. PWS Publishing Company: Boston, USA.

Saad Y, van der Vorst HA. 2000. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics* **123**: 1–33.

Saunders M. 1995. Solution of sparse rectangular systems using LSQR and CRAIG. *BIT* **35**: 588–604.

Tarantola A. 2005. *Inverse problem theory. methods for data fitting and model parameter estimation*. Elsevier: Amsterdam, The Netherlands.

Tshimanga J. 2007. On a class of limited memory preconditioners for large-scale nonlinear least-squares problems (with application to variational ocean data assimilation). PhD thesis, Department of Mathematics, FUNDP - University of Namur, Namur, Belgium.

Tshimanga J, Gratton S, Weaver A, Sartenaer A. 2008. Limited-memory preconditioners with application to incremental four-dimensional variational data assimilation. *Quarterly Journal of the Royal Meteorological Society* **134**: 751–769.

van der Vorst HA. 2003. *Iterative krylov methods for large linear systems*. Cambridge University Press.

Weaver AT, Vialard J, Anderson DLT. 2003. Three- and four-dimensional variational assimilation with a general circulation model of the tropical Pacific ocean, part 1 : formulation, internal diagnostics and consistency checks. *Monthly Weather Review* **131**: 1360–1378.