THE MERITS OF KEEPING IT SMOOTH: ITERATIVE LINEAR SOLVERS AND A SMOOTH EXACT PENALTY FUNCTION FOR CONSTRAINED NONLINEAR OPTIMIZATION

A DISSERTATION
SUBMITTED TO THE INSTITUTE FOR
COMPUTATIONAL AND MATHEMATICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ron Estrin
May 2019

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Yinyu Ye, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Michael Saunders, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Margot Gerritsen

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Lexing Ying

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Michael Friedlander

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

Abstract

Part I involves iterative methods for solving linear systems, least-squares problems, and least-norm problems. We show that solution error bounds at each iteration of these methods can be computed efficiently provided certain additional spectral information of the linear operators involved. Part II develops a smooth exact penalty method for constrained nonlinear optimization based on the work of Fletcher (1970, 1973b), where the methods of Part I play a central role in evaluating the penalty function and its gradients.

Often the most computationally intensive operation in numerical methods is solving linear systems, least-squares, and least-norm problems. Further, these linear systems often do not need to be solved to high accuracy—many methods can accept solutions solved to a prescribed accuracy. For positive definite systems, Part I develops Euclidean-norm error bounds for the Krylov methods SYMMLQ and CG using Gauss-Radau quadrature, when provided an underestimate of the smallest eigenvalue. For least-squares and least-norm problems, we develop solvers LSLQ and LNLQ (equivalent to SYMMLQ applied to the associated normal equations) and extend the error-bounding procedure for SYMMLQ to LSLQ and LNLQ. Similarly, the error-bounding procedure for CG is extended to LSQR and CRAIG. We compare with existing approaches for bounding errors, using linear systems from a standard test set and from the penalty method of Part II. Our approach is remarkably tight for the LQ methods (when good estimates of the spectrum are available), and gives reliable bounds for the CG-based methods.

In Part II, we develop a general constrained nonlinear optimization algorithm based on a smooth penalty function proposed by Fletcher (1970, 1973b). We first present the penalty function for equality-constrained problems, then provide a new smooth extension to inequality constrained problems. Although it was historically considered to be computationally prohibitive in practice, we demonstrate that the computational kernels required are no more expensive than other widely accepted methods for nonlinear optimization. The main computational kernel required to evaluate the penalty function and its derivatives is solving a structured linear system. This system can be solved efficiently by storing a single factorization per iteration. Alternatively, we can adapt the penalty function to the class of factorization-free algorithms by solving the linear system iteratively, using for example the methods described in Part I. The penalty function shows particular promise in cases where such linear system can be solved efficiently, e.g., for PDE-constrained optimization problems where efficient preconditioners exist, and opens the door to optimization solvers that accept inexact evaluations and derivatives. We discuss extensions including handling simple constraints explicitly, regularizing the penalty function, and demonstrate the merits of this approach on nonlinear optimization problems with PDE-constraints, and those from a standard test set.

Acknowledgements

My academic journey would not have been possible without the help of innumerable people with whom I've had the privilege to interact with over the last several years. I'm afraid that nothing that I could write below would truly do them justice for their support, but I will try.

It is difficult to imagine completing this dissertation without the constant support and kindness of Michael Saunders. He has been an amazing thesis advisor, mentor, friend, and tennis partner. His truly encyclopedic knowledge of all things numerical has often left me in awe. Michael is further unsurpassed as a technical writer and copyeditor; every paper written with him becomes a work of art. I have learned a great deal about writing from our long editing sessions for each paper, and in the process developed my own idiosyncrasies when writing. Michael is a giant to aspire to for both his unparalleled technical skill, but also for his constant gentle and cheerful nature.

I was also incredibly fortunate to have Michael Friedlander take on the role of an advisor. Coincidently, Michael F was Michael S's first PhD student, whereas I am his last (it's come full circle!). The second part of this thesis would not be possible without Michael: he let me to take over the Fletcher penalty project that he previously started with Dominique Orban, and hosted me for multiple quarters at UBC to work on it together. Michael is incredible at finding interesting problems that he gladly shares—several times I've found myself captivated by side projects rather than the original purpose of my visits.

Dominique Orban is the other constant (beside Michael Saunders) of every project in this thesis. He's a true role model of an academic that can stand with both feet solidly planted in both numerical linear algebra and optimization, and back up all the theory with professional, efficient, and beautifully written code. His excitement for working on new projects that blend these areas is always infectious. I would also like to thank him for supporting my conference travel on several occasions.

Chen Greif started me on my path in numerical analysis by instructing my first course in numerical linear algebra, and then supervising my undergraduate thesis on iterative linear solvers. His exceptional teaching during the course hooked me on the beauty of scientific computing. Beyond those beginnings, he's been an amazing collaborator and mentor (academic and otherwise) throughout my PhD; I also greatly appreciate the several excuses he's provided me to come home by hosting me at UBC.

Next, I would like to thank the rest of my thesis committee: Yinyu Ye, Margot Gerritsen, Lexing Ying, and Juan Alonso. This work benefited substantially from their questions and suggestions. Yinyu also merits special mention for agreeing to be my official principal supervisor upon Michael Saunders's retirement.

It's well known that the ICME staff are the ones that truly run the department, and I'd like to thank Indira, Matt, Brian, Claudine, Antoinette, and Amanda for making sure that the ICME students have everything they need (and making sure we've submitted all the correct forms on time). ICME has had two directors during my time here, and I'd like to thank both Margot Gerritsen and Gianluca Iaccarino for shaping ICME into the wonderful research institution it is today.

I'm grateful for the wonderful friends that kept PhD life interesting outside of academics; those back home, abroad, and at Stanford. I've met many of these friends during my time at Stanford, and what makes ICME such a special place is the many interactions with them over research, lunch, beers, and board games. A big thanks to all of the students at ICME, and especially to Nolan, Lan, Anjan, Brad, Arun, Victor, Anil, Austin, Xiaotong, Sven, Ollie, Ryan, Fred, Dan, Steven, Casey, Jordi, Nimit, Julia, Halwest, Nurbek, and doubly-especially to Allison (for her unbounded support and unrestrained proofreading).

Stanford Taekwondo, and in particular the team, has been a large part of my life at Stanford since my first year. They gave me goals to strive for beyond academics, and I am glad to be part of such an incredible and supportive community. Traveling with the team for competition or training trips was always a blast.

Above all, I want to thank my parents, Boris and Ella, and younger brother, Dan, for their unconditional love and support throughout my entire life. Nothing that I could write here could express how lucky and grateful I feel to have you in my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Numerical linear algebra and optimization are two core pillars of the computational sciences. Two subproblems that appear repeatedly in numerical methods are the solution of nonlinearly constrained optimization problems and the solution of structured linear systems (the former often depending on the latter). As these are often the most computationally intensive steps of numerical methods, developing efficient methods for these subproblems has wide-reaching impact across several application domains.

Ever-increasing problem scales result in larger constrained optimization problems and linear systems. To accommodate these large-scale problems, *factorization-free* approaches have gained popularity: those that avoid matrix-factorization and instead rely on matrix-vector products. Such approaches already have a rich history in the linear algebra community in the form of iterative methods, particularly Krylov subspace methods. The maturation of iterative linear solvers in turn helped increase the popularity of factorization-free optimization solvers. An early example is the Newton-CG trust-region solver of Steihaug (1983).

Key to the success of many large-scale numerical solvers is the use of *inexactness*: the idea that subproblems need not always be solved exactly. One of the earliest applications of this idea to optimization is the Inexact Newton method of Dembo, Eisenstat, and Steihaug (1982), which specifies how accurately the Newton linear system needed to be solved at every iteration to retain fast asymptotic convergence. In such algorithms, although subproblems need not be solved exactly, they must still be solved to a prescribed accuracy; these approximate subproblem solutions must be within a prescribed distance of the true solution. It is therefore of interest to design solvers that provide error bounds on intermediate approximate solutions to allow methods to stop early while guaranteeing that a prescribed accuracy has been attained. This is particularly true in the case of solving linear systems, the most common expensive subproblem encountered, which is the subject of Part I. The iterative methods of Part I are then used to develop a factorization-free smooth exact penalty method for nonlinearly constrained optimization in Part II.

1.1 Thesis overview

This thesis is organized in two parts: Part I develops iterative methods for linear systems, least-squares, and least-norm problems that provide error bounds on iterates, given additional spectral information on the necessary linear operators. Part II develops a smooth exact penalty method for nonlinearly constrained optimization based on the work of Fletcher (1970). These projects are connected by the dependency of Part II on Part I: evaluating our penalty function requires the solution of structured linear systems that are the subject of Part I—in particular, when the penalty function is evaluated approximately.

In Chapter 2 we introduce preliminaries for the iterative solution of linear systems, least-squares, and least-norm problems. Chapter 3 derives a method for computing error upper bounds on iterates of methods SYMMLQ and CG for positive definite linear systems. Chapter 4 and Chapter 5 develop methods LSLQ and LNLQ for least-squares and least-norm problems respectively; these methods are based on SYMMLQ applied to the corresponding normal equations, thus allowing us to develop error upper bounds based on the approach in Chapter 3. Extensions to symmetric quasidefinite systems are discussed in Chapter 6. We summarize the contributions and discuss future directions in Chapter 7.

Chapter 8 introduces the proposed penalty function for constrained nonlinear programming. Chapter 9 develops the theory for equality-constrained problems including: efficient evaluation of the penalty function and its derivatives; maintaining explicit linear constraints; regularizing the penalty function under some forms of constraint degeneracy; and inexact evaluation of the penalty function. Chapter 10 derives a new smooth extension of the penalty function for handling inequality constraints. We apply the penalty function to solve several PDE-constrained optimization problems and problems from a standard test set in Chapter 11. Contributions and future directions for this part are discussed in Chapter 12.

1.2 Code

Most of the methods presented are implemented in various libraries. These are the ones used for numerical experiments throughout. The iterative methods of Part I (SYMMLQ, LSLQ, and LNLQ) are available in Matlab[1] and Julia[2]. Fletcher's penalty function (Part II) is implemented in Matlab[3] with all of the discussed features. It is also implemented in C++ within Sandia National Labs' Rapid Optimization Library in Trilinos[4] (Heroux et. al., 2003); this implementation does not include some features such as explicit linear constraints.

1.3 Notation

A word of caution: the scope of notational consistency is mostly limited to individual chapters. I would like to take this moment to apologize for the inconsistency. This is unfortunately necessary because our work spans the subject matter of multiple communities, each having their own standardized notation.However, the following notation remains consistent throughout.

We use Householder notation. Matrices are denoted by capital letters A, B, ..., vectors by lowercase letters v, w, ..., and scalars by Greek letters α, β, γ, ..., with exceptions for c and s, which may be used for plane reflections with $c^2 + s^2 = 1$. All vectors are columns, but the slightly abusive notation $(\xi_1, \dots, \xi_k)$ may be used to enumerate their components. We use I for the identity matrix of the appropriate size, with e_k denoting the kth column. Define $\mathbb{1}$ as the vector of all ones. Denote $\|\cdot\|$ as the Euclidean-norm, and $\|\cdot\|_A$ as the energy norm defined by $\|u\|_A^2 := u^TAu$ for A symmetric positive definite (SPD). Define $[n] = \{1, 2, \dots, n\}$.

For square $A \in \mathbb{R}^{n\times n}$, we order its eigenvalues according to $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$. Similarly, for rectangular $A \in \mathbb{R}^{m\times n}$, we order the singular values according to $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_{\min\{m,n\}} \geqslant 0$, and let $\operatorname{cond}(A) = \sigma_1/\sigma_{\min\{m,n\}}$. It should be clear from context if λ and σ are referring to scalars that are not the eigenvalues or singular values of a linear operator.

[1]https://github.com/restrin/LinearSystemSolvers
[2]https://github.com/JuliaSmoothOptimizers/Krylov.jl
[3]https://github.com/optimizers/FletcherPenalty
[4]https://github.com/trilinos/Trilinos

Part I

Iterative Methods for Linear Systems with Error Bounding Properties

Chapter 2

Preliminaries

We consider solving three related problems

$$Ax = b, \qquad (A = A^T) \tag{L}$$

$$\min_x \|Ax - b\|, \tag{LS}$$

$$\min_x \|x\| \text{ subject to } Ax = b, \tag{LN}$$

where $A \in \mathbb{R}^{m \times n}$ is a linear operator with efficiently computable products. We assume that (L) and (LN) are both consistent. Denote $x^\star := A^\dagger b$ as the solution of each problem, where $A^\dagger$ is the Moore-Penrose pseudoinverse, which solves

$$x^\star := \min_x \|x\| \quad \text{subject to} \quad x \in \operatorname*{argmin}_{\bar{x}} \|A\bar{x} - b\|, \tag{MLS}$$

regardless of whether the system is full-rank or consistent.

For (LN), define $y^\star := (AA^T)^\dagger b$, which solves

$$y^\star := \min_{y \in \mathbb{R}^m} \|y\| \text{ subject to } AA^T y = b,$$

and satisfies $x^\star = A^T y^\star$.

Problems (LS) and (LN) are equivalent to solving positive-semidefinite linear systems (the *normal equations*), or solving a 2×2 block system (the *augmented system*). These are shown in the second and third columns of Table 2.1.

Table 2.1: Relationships between (L), (LS), and (LN). For each row, the first three columns describe equivalent problems. The last column gives common methods for each problem.

Problem	Normal Equations	Augmented System	Methods	
$Ax = b$			CG MINRES SYMMLQ	(HS) (PSa) (PSa)
$\min_x \|Ax - b\|$	$A^T Ax = A^T b$	$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$	LSQR LSMR LSLQ	(PSb) (FS) (EOSa)
$\min_x \|x\| : Ax = b$	$AA^T y = b, \;\; x = A^T y$	$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ -y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}$	CRAIG LSQR LNLQ	(C) (PSb) (EOSb)

HS (Hestenes and Stiefel, 1952), FS (Fong and Saunders, 2011), C (Craig, 1955), PSa (Paige and Saunders, 1975), PSb (Paige and Saunders, 1982a), EOSa (Estrin, Orban, and Saunders, 2019c), EOSb (Estrin, Orban, and Saunders, 2019e)

2.1 Methods for symmetric linear systems

2.1.1 The Lanczos process

The Lanczos (1950) process, described in Algorithm 1, is the basis for most Krylov subspace methods for symmetric linear systems. In line 1, $\beta_1 v_1 = b$ is short for "$\beta_1 = \|b\|$; if $\beta_1 = 0$ then exit; else $v_1 = b/\beta_1$"; similarly for line 5.

Algorithm 1 Lanczos Tridiagonalization Process

```
Require: $A$, $b$
1: $\beta_1 v_1 = b$
2: for $k = 1, 2, \dots$ do
3:     $w = Av_k$
4:     $\alpha_k = v_k^T w$
5:     $\beta_{k+1} v_{k+1} = w - \alpha_k v_k - \beta_k v_{k-1}$
6: end for
```

After k steps, the Lanczos process can be summarized as

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T = V_{k+1} H_k, \tag{2.2}$$

where $V_k = [v_1 \ \dots \ v_k]$ has orthonormal columns in exact arithmetic, and

$$T_k := \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \beta_k & \alpha_k \end{bmatrix} = \begin{bmatrix} T_{k-1} & \beta_k e_{k-1} \\ \beta_k e_{k-1}^T & \alpha_k \end{bmatrix}, \qquad H_k := \begin{bmatrix} T_k \\ \beta_{k+1} e_k^T \end{bmatrix}. \tag{2.3}$$

Note that (2.2) holds up to machine precision under floating-point arithmetic, but V_k quickly loses orthogonality. For each k, V_k forms a (theoretically) orthonormal basis for the Krylov subspace $\mathcal{K}_k = \mathcal{K}_k(A, b) := \{\, b, Ab, \dots, A^{k-1} b \,\}$.

Krylov subspace methods proceed by defining iterates $x_k \in \mathcal{K}_k$ as the "best" approximation of the solution $x^\star$:

$$x_k = \underset{x \in \mathcal{K}_k}{\operatorname{argmin}} \ \rho(x), \tag{2.4}$$

where $\rho : \mathbb{R}^n \to \mathbb{R}$ is some measure of how close x is to $x^\star$. By choosing different functions ρ we can derive the iterative methods that appear in the following section. In practice, we compute the iterates by expressing $x_k = V_k \bar{x}_k \in \mathcal{K}_k$ and defining an equivalent problem to (2.4) by seeking $H_k \bar{x}_k \approx \beta_1 e_1$ (where the meaning of "$\approx$" depends on ρ).

When A is singular, (L) does not have a unique solution. The following proposition shows that all Krylov methods converge to the same canonical solution when (L) is consistent.

Proposition 2.1 *Assume symmetric A is singular but $Ax = b$ is consistent. Let $x^\star$ be the solution produced by a Krylov subspace method for solving $Ax^\star = b$; that is, $x^\star \in \mathcal{K}_\ell$ for some ℓ. Then $x^\star$ is the unique solution to*

$$\min_{x \in \mathbb{R}^n} \|x\| \quad \textit{subject to} \quad Ax = b. \tag{2.5}$$

Proof. The necessary and sufficient conditions for $x^\star$ to solve (2.5) are that $Ax^\star = b$ and $x^\star \in \text{range}(A)$. Because $Ax = b$ is consistent, $b \in \text{range}(A)$, and so the Krylov subspace is contained in $\text{range}(A)$, implying that $x^\star \in \mathcal{K}_k \subseteq \text{range}(A)$. Because $Ax^\star = b$ and $x^\star \in \text{range}(A)$, it must be the solution to (2.5). □

The following sections briefly describe three popular iterative methods for symmetric linear systems based on the Lanczos process.

2.1.2 CG

The Conjugate Gradient method (CG) (Hestenes and Stiefel, 1952) is arguably the most popular method for SPD systems. The iterates are defined by

$$x_k^C := \operatorname*{argmin}_{x \in \mathcal{K}_k} \|x^\star - x\|_A^2. \tag{2.6}$$

An equivalent formulation of CG iterates comes from the Lanczos process, where

$$x_k^C = V_k \bar{x}_k^C, \qquad T_k \bar{x}_k^C := \beta_1 e_1. \tag{2.7}$$

Further details on how CG iterates are computed can be found in (Saunders, 2019; Demmel, 1997, §6.6.3). When A is not SPD, CG is not advised because A no longer defines a norm in (2.6), and some iterates may be undefined because T_k could become indefinite or singular.

2.1.3 MINRES

The Minimum Residual method (MINRES) (Paige and Saunders, 1975) handles the case where A is symmetric but possibly indefinite. Its iterates are defined by

$$x_k^M := \operatorname*{argmin}_{x \in \mathcal{K}_k} \|Ax - b\|, \tag{2.8}$$

which via the Lanczos process is equivalent to

$$x_k^M = V_k x_k^M, \qquad \bar{x}_k^M := \operatorname*{argmin}_{\bar{x} \in \mathbb{R}^k} \|H_k \bar{x} - \beta_1 e_1\|.$$

Hestenes and Stiefel's Conjugate Residual method (CR) Hestenes and Stiefel (1952) is equivalent to MINRES when A is SPD. Note however that MINRES is stable for any symmetric A, while CR can break down when A is indefinite. Fong and Saunders (2011) derive several relationships between CR and CG for positive definite systems.

2.1.4 SYMMLQ

The last common method for symmetric linear systems is SYMMLQ (Paige and Saunders, 1975), defined by two equivalent subproblems (Fischer, 1996; Saunders, 2019):

$$\begin{aligned} x_k^L &:= \operatorname*{argmin}_{x \in \mathcal{K}_k} \|x\| \qquad \text{subject to} \qquad b - Ax \perp \mathcal{K}_{k-1} \\ &= \operatorname*{argmin}_{x \in \mathbb{R}^n} \|x^\star - x\|^2, \;\; \text{with} \;\; x \in A\mathcal{K}_{k-1} := \text{span}\{\, Ab, A^2 b, \ldots, A^{k-1} b \,\}. \end{aligned}$$

The equivalent definition via the Lanczos process is

$$x_k^L = V_k \bar{x}_k^L, \qquad \bar{x}_k^L := \underset{\bar{x}\in\mathbb{R}^k}{\operatorname{argmin}} \ \|\bar{x}\| \text{ subject to } H_{k-1}^T \bar{x} = \beta_1 e_1. \tag{2.9}$$

SYMMLQ and related methods feature prominently in Part I. The details for its implementation are given in Chapter 3.

We also give a new, simpler, but totally unstable CG-like implementation in Appendix A.

2.2 Methods for rectangular linear systems

2.2.1 The Golub-Kahan process

The Golub and Kahan (1965) process, described in Algorithm 2, is the basis of many Krylov subspace methods for nonsymmetric and rectangular linear systems.

Algorithm 2 Golub-Kahan Bidiagonalization Process

```
Require: $A$, $b$
1: $\beta_1 u_1 = b$
2: $\alpha_1 v_1 = A^T u_1$
3: for $k = 1, 2, \dots$ do
4:     $\beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k$
5:     $\alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k$
6: end for
```

Define $U_k := \begin{bmatrix} u_1 & \cdots & u_k \end{bmatrix}$, $V_k := \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}$, and

$$L_k := \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{bmatrix}, \quad B_k := \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix} = \begin{bmatrix} L_k \\ \beta_{k+1} e_k^T \end{bmatrix}. \tag{2.10}$$

The matrices U_k and V_k produced by Algorithm 2 satisfy

$$\begin{aligned} AV_k &= U_{k+1} B_k, \\ A^T U_{k+1} &= V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T = V_{k+1} L_{k+1}^T, \end{aligned} \tag{2.11}$$

and in exact arithmetic, the identities $U_k^T U_k = I_k$ and $V_k^T V_k = I_k$ hold as well. Again, (2.11) holds to machine precision under floating-point arithmetic, but U_k and V_k lose orthogonality.

Algorithm 2 can be interpreted as a more accurate form of Lanczos process (Algorithm 1) for the Gramian matrices $A^T A$ and AA^T with starting vectors $A^T b$ and b:

$$A^T A V_k = V_{k+1} L_{k+1}^T B_k, \tag{2.12a}$$

$$AA^T U_k = U_{k+1} B_k L_k^T. \tag{2.12b}$$

Thus V_k and U_k are orthonormal bases for $\mathcal{K}_k(A^T A, A^T b)$, and $\mathcal{K}_k(AA^T, b)$ respectively. The relationship between the Golub-Kahan and Lanczos processes leads to a close relationship between the methods discussed in Section 2.1, and the methods below for problems (LS) and (LN). This relationship is further explored in Section 2.2.5.

Below we describe three popular methods for least-squares and least-norm problems.

2.2.2 LSQR

LSQR (Paige and Saunders, 1975) is for least-squares problems (LS); it defines iterates according to the following equivalent definitions:

$$x_k^C := \underset{x \in \mathcal{K}_k(A^TA, A^Tb)}{\operatorname{argmin}} \|Ax - b\|, \tag{2.13}$$

$$:= V_k \bar{x}_k^C, \qquad \bar{x}_k^C := \underset{\bar{x} \in \mathbb{R}^k}{\operatorname{argmin}} \|B_k \bar{x} - \beta_1 e_1\|. \tag{2.14}$$

We intentionally use the same notation x_k^C because of the equivalence between LSQR and CG (see Table 2.2).

2.2.3 LSMR

LSMR (Fong and Saunders, 2011) is also for least-squares (LS); it is defined by

$$x_k^M := \underset{x \in \mathcal{K}_k(A^TA, A^Tb)}{\operatorname{argmin}} \|A^T(Ax - b)\|, \quad := V_k \bar{x}_k^M, \qquad \bar{x}_k^M := \underset{\bar{x} \in \mathbb{R}^k}{\operatorname{argmin}} \|L_{k+1}^T B_k \bar{x} - \alpha_1 \beta_1 e_1\|. \tag{2.15}$$

LSMR shares a similar equivalence with MINRES (Table 2.2).

2.2.4 CRAIG

CRAIG (Craig, 1955) is for consisten least-norm problems (LN) and is defined equivalently in two ways:

$$x_k^C := \underset{x \in \mathcal{K}_k(A^TA, A^Tb)}{\operatorname{argmin}} \|x^\star - x\|, \tag{2.16a}$$

$$= A^T y_k^C, \qquad y_k^C := \underset{y \in \mathcal{K}_k(AA^T, b)}{\operatorname{argmin}} \|y^\star - y\|_{AA^T}. \tag{2.16b}$$

From (2.16b), it is clear that CRAIG is equivalent to CG on $AA^Ty = b$ (and we denote its iterates by x_k^C as well; see Table 2.2). Further, from (2.16a), we see that CRAIG is the error-minimizing method among all methods producing iterates in $\mathcal{K}_k(A^TA, A^Tb)$.

2.2.5 Equivalence of methods

Each of the methods described in Section 2.2 can be interpreted as a stable implementation of the methods in Section 2.1 applied to the corresponding normal equations, given the connection between the Golub-Kahan and Lanczos processes via (2.12). Table 2.2 summarizes the relationship between these methods. In particular, we complete the table by introducing LSLQ (Chapter 4) and LNLQ (Chapter 5). We also prove the relationship between LSQR applied to least-norm problems and MINRES.

Table 2.2: Methods for least-squares or least-norm problems and their corresponding equivalent method on the normal equations. Assume that the least-norm problem is consistent.

Method	Problem	Equivalent to	Applied to
LSQR, LSMR, LSLQ for x	$\min_x \lVert Ax-b\rVert_2$	CG, MINRES, SYMMLQ for x	$A^TAx = A^Tb$
CRAIG, LSQR, LNLQ for x	$\min_x \lVert x\rVert_2 : Ax = b$	CG, MINRES, SYMMLQ for y	$AA^Ty = b$, $x = A^Ty$

Proposition 2.2 *LSQR applied to $Ax = b$ in (LN) is equivalent to MINRES applied to $AA^Ty = b$. The iterates satisfy $x_k = A^Ty_k$, where x_k are iterates from LSQR and y_k are from MINRES.*

Proof. First note that $x \in \mathcal{K}_k(A^TA, A^Tb) = A^T\mathcal{K}_k(AA^T, b)$ is equivalent to there existing $y \in \mathcal{K}(AA^T, b)$ such that $x = A^Ty$. Observe then that (2.13) is equal to

$$x_k = \underset{x\in\mathcal{K}_k(A^TA,A^Tb)}{\operatorname{argmin}} \lVert Ax-b\rVert = \underset{\substack{y\in\mathcal{K}_k(AA^T,b)\\ x=A^Ty}}{\operatorname{argmin}} \lVert AA^Ty-b\rVert,$$

which defines the same subproblem as (2.8) for y_k. ∎

2.3 Termination criteria

Typically Krylov subspace methods are terminated according to the residual norm $\lVert r\rVert = \lVert b - Ax\rVert$ (which is zero at the solution for consistent systems), or the optimality residual norm $\lVert A^Tr\rVert$ (which is always zero at the solution). However, a small residual can still lead to loose error bounds that depend on the condition number of A, because

$$\lVert x^\star - x\rVert \leqslant \lVert r\rVert\lVert A^{-1}\rVert \quad \text{and} \quad \frac{\lVert x^\star - x\rVert}{\lVert x^\star\rVert} \leqslant \frac{\lVert r\rVert}{\lVert b\rVert}\lVert A\rVert\lVert A^{-1}\rVert.$$

It is therefore of interest to design iterative methods capable of estimating and bounding the error norm directly. The focus of the upcoming chapters is thus to: develop approaches for computing cheap error bounds at every iteration of CG and SYMMLQ, and then design iterative methods for (LS) and (LN) for which it is possible to compute such error bounds.

2.3.1 Matrices, moments, and quadrature

We give a brief overview of the seminal work of Golub and Meurant (1994, 1997) that relates the evaluation of quadratic forms with Gauss quadrature and the Lanczos (1950) process. These results are vital to developing the error bounds in the upcoming chapters. A more detailed treatment can be found in Golub and Meurant (2010). Assume for now that A is an SPD linear operator.

We are interested in evaluating quanties of the form

$$b^T f(A) b = b^T \left(\sum_{i=1}^{n} f(\lambda_i) p_i p_i^T \right) b = \sum_{i=1}^{n} f(\lambda_i) \mu_i^2, \qquad \mu_i := p_i^T b, \;\; i = 1, \ldots, n, \tag{2.17}$$

where f is an analytic function, $A \in \mathbb{R}^{n \times n}$ has eigenvalues $\lambda_1 \geqslant \cdots \geqslant \lambda_n > 0$, and corresponding eigenvectors $\{p_1, \ldots, p_n\}$.

Golub and Meurant (1994) explain that the main insight for evaluating (2.17) comes from viewing it as a Riemann-Stieltjes integral with piecewise constant Stieltjes measure:

$$\sum_{i=1}^{n} f(\lambda_i) \mu_i^2 = \int_{\sigma_n}^{\sigma_1} f(\lambda) d\mu(\lambda), \qquad \mu(\lambda) := \begin{cases} 0 & \text{if } \lambda \; < \lambda_n \\ \sum_{j=i}^{n} \mu_j^2 & \text{if } \lambda_i \leqslant \lambda < \lambda_{i-1} \\ \sum_{j=1}^{n} \mu_j^2 & \text{if } \lambda_1 \leqslant \lambda, \end{cases}$$

allowing (2.17) to be evaluated via Gauss quadrature.

The second main insight is the close relationship between Gauss quadrature and the Lanczos (1950) process. In particular, the nodes and weights of the Gauss quadrature can be obtained directly from the eigenvalue decomposition of T_k from Algorithm 1 (see Golub and Meurant (1994, §3)). If upper or lower bounds on the spectrum of A are available, we can perform Gauss-Radau or Gauss-Lobato quadrature to obtain upper or lower bounds on (2.17). The connection between such quadratic forms and their approximation via Gaussian quadrature is most notably studied by Dahlquist, Eisenstat, and Golub (1972), Dahlquist, Golub, and Nash (1979), and Golub and Meurant (1994, 1997).

In our case, we are interested in upper bounds on (2.17), meaning that we perform Gauss-Radau quadrature and require an underestimate of the smallest eigenvalue of A, namely $\lambda_{\text{est}} < \lambda_n$[1]. The tightness of the Gauss-Radau quadrature therefore depends on the tightness of the eigenvalue estimate. If A is semidefinite (with rank r), the sum in (2.17) is evaluated over the first r eigenvalues. In this case, we require that $\lambda_{\text{est}} < \lambda_r$.

The following theorem summarizes these results.

Theorem 2.3 *Let A be positive semidefinite with rank r and $Ax = b$ consistent, and let $f : (0, \infty) \to \mathbb{R}$ be such that its derivatives satisfy $f^{(2m+1)}(\xi) < 0$ for all $\xi \in (\lambda_r, \lambda_{\max}(A))$ and all integers $m \geqslant 0$. Fix $\lambda_{\text{est}} \in (0, \lambda_r)$. Let T_k be generated by k steps of the Lanczos process (Algorithm 1) on (A, b) and let*

$$\widetilde{T}_k := \begin{bmatrix} T_{k-1} & \beta_k e_{k-1} \\ \beta_k e_{k-1}^T & \omega_k \end{bmatrix},$$

where ω_k is chosen such that $\lambda_{\min}(\widetilde{T}_k) = \lambda_{\text{est}}$. Then

$$b^T f(A) b \leqslant \|b\|^2 e_1^T f(\widetilde{T}_k) e_1.$$

Proof. The result follows from (Golub and Meurant, 1994, Theorem 3.2) and the section preceding it, as well as (Golub and Meurant, 1994, Theorem 3.4), although those results only consider the case where A is SPD. ∎

[1]In general, if there exists ℓ such that $\mu_i = 0$ for all $i > \ell$, it is sufficient for $\lambda_{\text{est}} < \lambda_\ell$.

Chapter 3

Euclidean-norm error bounds for CG and SYMMLQ

We derive error bounds for SYMMLQ and CG iterates using Gauss-Radau quadrature. The contributions of this paper are predominantly from Estrin, Orban, and Saunders (2019d).

3.1 Computing SYMMLQ iterates

We begin with a brief overview of how SYMMLQ iterates are computed, as well as some of its key properties. A more detailed treatment is given by Paige and Saunders (1975), from which we derive most of the notation with minor differences. In particular, we set indices such that $x_k \in \mathcal{K}_k$ for all Krylov subspace methods. This section further serves to set the notation used for this chapter.

To obtain x_k^L (defined in (2.9)), we compute the LQ factorization $T_{k-1}Q_{k-1}^T = \bar{L}_{k-1}$, where Q_{k-1} is orthogonal and

$$\bar{L}_{k-1} := \begin{bmatrix} \gamma_1 & & & & \\ \delta_2 & \gamma_2 & & & \\ \varepsilon_3 & \delta_3 & \gamma_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \varepsilon_{k-1} & \delta_{k-1} & \bar{\gamma}_{k-1} \end{bmatrix}.$$

Note that the diagonal entries of $\bar{L}_{k-1}$ are γ_j for $j = 1, \ldots, k-2$, and the last entry is $\bar{\gamma}_{k-1}$. A single $2{\times}2$ reflection is applied on the right to obtain $H_{k-1}^T Q_k^T = [L_{k-1}\ 0]$, so that L_{k-1} differs from $\bar{L}_{k-1}$ only in the last diagonal entry, which becomes γ_{k-1}. The reflection is constructed so that

$$\begin{array}{c|cc} & k-1 & k \\ \hline k-1 & \bar{\gamma}_{k-1} & \beta_k \\ k & \bar{\delta}_k & \alpha_k \\ k+1 & 0 & \beta_{k+1} \end{array} \quad \begin{array}{c|cc} & k-1 & k \\ \hline & c_k & s_k \\ & s_k & -c_k \end{array} = \begin{array}{c|cc} & k-1 & k \\ \hline & \gamma_{k-1} & 0 \\ & \delta_k & \bar{\gamma}_k \\ & \varepsilon_{k+1} & \bar{\delta}_{k+1} \end{array}.$$

The first iteration begins with $k = 2$ (because SYMMLQ iterates are defined only for $k \geqslant 2$), and $\bar{\gamma}_1 = \alpha_1$ and $\bar{\delta}_2 = \beta_2$. For $k \geqslant 2$, define $z_{k-1} = (\zeta_1, \ldots, \zeta_{k-1})$ as the solution to $L_{k-1}z_{k-1} = \beta_1 e_1$. Then $\bar{x}_k^L = Q_k^T \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix}$ solves (2.9), so that

$$x_k^L = V_k \bar{x}_k^L = V_k Q_k^T \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} = \overline{W}_k \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} = W_{k-1} z_{k-1} \tag{3.1}$$

with the orthogonal matrix $\overline{W}_k = V_k Q_k^T = \begin{bmatrix} w_1 & \dots & w_{k-1} & \bar{w}_k \end{bmatrix} = \begin{bmatrix} W_{k-1} & \bar{w}_k \end{bmatrix}$.

We can readily obtain the kth CG iterate as part of the SYMMLQ iteration. We define $\bar{z}_k = (z_{k-1}, \bar{\zeta}_k)$ as the solution to $\bar{L}_k \bar{z} = \beta_1 e_1$, so that we solve (2.7) using $T_k = \bar{L}_k Q_k$ and

$$x_k^C = V_k \bar{x}_k^C = V_k Q_k^T \bar{z}_k = \overline{W}_k \bar{z}_k = x_k^L + \bar{\zeta}_k \bar{w}_k. \tag{3.2}$$

It can further be shown that $\bar{\zeta}_k = \zeta_k / c_{k+1}$, and (3.2) implies that

$$\|x_k^C\|^2 = \|x_k^L\|^2 + \bar{\zeta}_k^2 \tag{3.3}$$

Paige and Saunders (1975) establish the following results.

Lemma 3.1 *The SYMMLQ iterates x_k^L satisfy the following properties:*

1. *$x_k^L = x_{k-1}^L + \zeta_{k-1} w_{k-1} \in \mathcal{K}_k$, with $w_{k-1} \perp x_{k-1}^L$. Furthermore, $\|x_k^L\| = \|z_{k-1}\|$ and is monotonically increasing.*

2. *Since x_k^L is updated along orthogonal directions, $\|x^\star - x_k^L\|^2 = \|x^\star\|^2 - \|x_k^L\|^2$ is monotonically decreasing.*

3.2 Upper bounds when A is semidefinite

In this section, we derive an upper bound on the error in SYMMLQ and build upon it to derive an upper bound for CG. As with other Gauss-Radau based approaches, we assume the availability of a non-zero underestimate to the smallest non-zero eigenvalue of A.

We assume that A is positive semidefinite with rank $r \leqslant n$, but that $Ax = b$ is consistent. The situation where A is SPD is simply a special case. By Proposition 2.1, SYMMLQ and CG identify the pseudoinverse solution $x^\star = A^\dagger b = \operatorname{argmin}_x\{\|x\| \mid Ax = b\}$. The Rayleigh-Ritz theorem states that

$$\lambda_r = \min\{v^T A v \mid v \in \operatorname{range}(A),\ \|v\| = 1\}.$$

In addition, for any $u \in \mathbb{R}^k$ with $\|u\| = 1$, $V_k u \in \operatorname{range}(A)$ because each $v_i \in \operatorname{range}(A)$, and $\|V_k u\| = 1$. Then, each T_k is positive definite because $u^T T_k u = (V_k u)^T A (V_k u) \geqslant \lambda_r > 0$. Because each x_k^L and x_k^C lies in range(A) by definition, the SYMMLQ and CG iterations occur as if they were applied to the symmetric and positive definite system consisting in the restriction of $Ax = b$ to range(A).

3.2.1 Existing error estimates for Krylov subspace methods

There has been significant interest in estimating the A-norm of the CG error, the history of which is detailed by Strakoš and Tichý (2002). The Euclidean-norm has received less attention as it is more difficult to estimate for CG, although it has been studied by Strakoš and Tichý (2002), Golub and Meurant (1997), Meurant (1997, 2005), and Frommer, Kahl, Lippert, and Rittich (2013). Although estimates for the CG error are derived by Meurant (2005), they are not proved to be upper bounds, while those of Frommer et al. (2013) are upper bounds but can be more expensive in ill-conditioned cases in order to achieve improved accuracy (by increasing d in Section 3.6). The only Euclidean-norm SYMMLQ error upper bounds we are aware of are those of Szyld and Widlund (1993), who provide a pessimistic geometric error decay rate.

The strategy behind estimating error norms (including our own) is to recognize the error and related quantities as quadratic forms $r^T f(A) r$ evaluated at A for a certain function f (e.g., $f(\xi) = \xi^{-2}$ and $r = b - Ax$) and seek estimates of this quadratic form using the techniques in Section 2.3.1.

3.2.2 Upper bounds on the SYMMLQ error

According to (3.1) and result 2 of Lemma 3.1, we have

$$\|x^\star - x_k^L\|^2 = \|x^\star\|^2 - \|x_k^L\|^2 = \|x^\star\|^2 - \|z_{k-1}\|^2. \tag{3.4}$$

Thus it is sufficient to find an upper bound on $\|x^\star\|^2 = b^T A^{-2} b$, assuming temporarily for the clarity of exposition that A is SPD. In this section, we show how to obtain such a bound at the cost of a few scalar operations per iteration.

We are interested in the choices $f(\xi) = \xi^{-2}$ (with $\xi = A$) as well as $f(\xi) = \xi^{-1}$ (with $\xi = A^2$). Although these appear to be exactly the same, the estimation procedure and convergence properties of the estimates are different when A is indefinite, since A^2 is guaranteed to be positive semidefinite.

When A is only semidefinite, we need to estimate $\|x^\star\|^2 = b^T \left(A^\dagger\right)^2 b = b^T f(A) b$, where

$$f(\xi) = \begin{cases} \xi^{-2} & \xi > 0, \\ 0 & \xi = 0. \end{cases} \tag{3.5}$$

We therefore obtain upper bounds of $b^T f(A) b$ using Gauss-Radau quadrature by invoking Theorem 2.3. We therefore need to compute ω_k in $\widetilde{T}_k$ from Theorem 2.3, then efficiently evaluate the quadratic form using $\widetilde{T}_k$. In Golub and Meurant (1997) it is shown that $\omega_k = \lambda_{\text{est}} + \eta_{k-1}$, where η_{k-1} is obtained from the last entry of the solution of the system

$$(T_{k-1} - \lambda_{\text{est}} I)\, u_{k-1} = \beta_k^2 e_{k-1}. \tag{3.6}$$

To compute u_{k-1}, we take the QR factorization of $T_{k-1} - \lambda_{\text{est}} I$ analogous to the LQ factorization of H_{k-1}^T in SYMMLQ. This differs from (Orban and Arioli, 2017), where a Cholesky factorization is used, but QR factorization allows us to solve an indefinite system using a stable factorization. It begins with the 2×2 reflection

$$\begin{bmatrix} c_1^{(\omega)} & s_1^{(\omega)} \\ s_1^{(\omega)} & -c_1^{(\omega)} \end{bmatrix} \begin{bmatrix} \alpha_1 - \lambda_{\text{est}} & \beta_2 & \\ \beta_2 & \alpha_2 - \lambda_{\text{est}} & \beta_3 \end{bmatrix} = \begin{bmatrix} \rho_1 & \sigma_2 & \tau_3 \\ & \bar\rho_2 & \bar\sigma_3 \end{bmatrix},$$

and proceeds with reflections defined by

$$\begin{array}{c} \\ j \\ j+1 \end{array} \begin{array}{cc} j & j+1 \\ \left[\begin{array}{c} c_j^{(\omega)} \\ s_j^{(\omega)} \end{array}\right. & \left.\begin{array}{c} s_j^{(\omega)} \\ -c_j^{(\omega)} \end{array}\right] \end{array} \begin{array}{ccc} j & j+1 & j+2 \\ \left[\begin{array}{c} \bar\rho_j \\ \beta_{j+1} \end{array}\right. & \begin{array}{c} \bar\sigma_{j+1} \\ \alpha_{j+1} - \lambda_{\text{est}} \end{array} & \left.\begin{array}{c} \\ \beta_{j+2} \end{array}\right] \end{array} = \begin{array}{ccc} j & j+1 & j+2 \\ \left[\begin{array}{c} \rho_j \\ \\ \end{array}\right. & \begin{array}{c} \sigma_{j+1} \\ \bar\rho_{j+1} \end{array} & \left.\begin{array}{c} \tau_{j+2} \\ \bar\sigma_{j+2} \end{array}\right] \end{array}.$$

Putting the QR factorization together, we have

$$T_{k-1} - \lambda_{\text{est}} I = \begin{bmatrix} \times & \times & \cdots & \times \\ \times & \times & & \times \\ & \ddots & \ddots & \vdots \\ & & s_{k-2}^{(\omega)} & -c_{k-2}^{(\omega)} \end{bmatrix} \begin{bmatrix} \rho_1 & \sigma_2 & \tau_3 & & \\ & \rho_2 & \sigma_3 & \ddots & \\ & & \rho_3 & \ddots & \tau_{k-1} \\ & & & \ddots & \sigma_{k-1} \\ & & & & \bar\rho_{k-1} \end{bmatrix},$$

where $\times$ is a placeholder for entries we are not interested in. We do not need to compute the QR factorization fully as we require only the scalars $s_{k-2}^{(\omega)}$, $c_{k-2}^{(\omega)}$, and $\bar\rho_{k-1}$ at the kth iteration. The relevant recurrence relations are

$$\begin{aligned} \bar\rho_1 &= \alpha_1 - \lambda_{\text{est}}, \\ \bar\sigma_2 &= \beta_2, \qquad c_0^{(\omega)} = -1, \\ \rho_1 &= \sqrt{\bar\rho_1^2 + \beta_2^2}, \qquad c_1^{(\omega)} = \frac{\alpha_1 - \lambda_{\text{est}}}{\rho_1}, \qquad s_1^{(\omega)} = \frac{\beta_2}{\rho_1}; \end{aligned}$$

for $k \geqslant 2$:

$$\begin{aligned} \bar\rho_k &= s_{k-1}^{(\omega)} \bar\sigma_k - c_{k-1}^{(\omega)} (\alpha_k - \lambda_{\text{est}}), \\ \bar\sigma_{k+1} &= -c_{k-1}^{(\omega)} \beta_{k+1}, \qquad \tau_k = s_{k-2}^{(\omega)} \beta_k, \\ \rho_k &= \sqrt{\bar\rho_k^2 + \beta_{k+1}^2}, \qquad c_k^{(\omega)} = \frac{\bar\rho_k}{\rho_k}, \qquad s_k^{(\omega)} = \frac{\beta_{k+1}}{\rho_k}. \end{aligned}$$

From the QR factorization of (3.6), we see that

$$\begin{bmatrix} \rho_1 & \sigma_2 & \tau_3 & & \\ & \rho_2 & \sigma_3 & \ddots & \\ & & \rho_3 & \ddots & \tau_{k-1} \\ & & & \ddots & \sigma_{k-1} \\ & & & & \bar\rho_{k-1} \end{bmatrix} \begin{bmatrix} \times \\ \vdots \\ \times \\ \eta_{k-1} \end{bmatrix} = \begin{bmatrix} \times & \times & & \\ \times & \times & \ddots & \\ \vdots & & \ddots & s_{k-2}^{(\omega)} \\ \times & \cdots & \cdots & -c_{k-2}^{(\omega)} \end{bmatrix} \beta_k^2 e_{k-1} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_k^2 s_{k-2}^{(\omega)} \\ -\beta_k^2 c_{k-2}^{(\omega)} \end{bmatrix},$$

and therefore $\eta_{k-1} = -\beta_k^2 c_{k-2}^{(\omega)} / \bar\rho_{k-1}$, with $\omega_k = \lambda_{\text{est}} + \eta_{k-1}$.

We now describe how to compute $\beta_1^2 e_1^T \widetilde{T}_k^{-2} e_1$ efficiently. Note that if we take the LQ factorization of $\widetilde{T}_k = \widetilde{L}_k \widetilde{Q}_k$, then by symmetry of $\widetilde{T}_k$,

$$\begin{aligned} \beta_1^2 e_1^T \widetilde{T}_k^{-2} e_1 &= \beta_1^2 e_1^T \big(\widetilde{L}_k \widetilde{Q}_k\big)^{-T} \big(\widetilde{L}_k \widetilde{Q}_k\big)^{-1} e_1 \\ &= \beta_1^2 e_1^T \widetilde{L}_k^{-T} \widetilde{L}_k^{-1} e_1 = \|\beta_1 \widetilde{L}_k^{-1} e_1\|^2 \\ &= \|\widetilde{z}_k\|^2, \end{aligned} \tag{3.7}$$

where $\widetilde{L}_k \widetilde{z}_k = \beta_1 e_1$. Because $\widetilde{T}_k$ differs from T_k only in the (k,k) entry, we have

$$\widetilde{L}_k = \begin{bmatrix} L_{k-1} & 0 \\ \varepsilon_k e_{k-2}^T + \psi_k e_{k-1}^T & \bar\omega_k \end{bmatrix}, \quad \text{where} \quad \begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix} \begin{bmatrix} \bar\delta_k \\ \omega_k \end{bmatrix} = \begin{bmatrix} \psi_k \\ \bar\omega_k \end{bmatrix},$$

where ε_k comes from the LQ factorization of T_k. The vector $\widetilde{z}_k$ is closely related to z_k. Indeed $L_{k-1}z_{k-1} = \beta_1 e_1$, and therefore

$$\widetilde{z}_k = \begin{bmatrix} z_{k-1} \\ \widetilde{\zeta}_k \end{bmatrix}, \qquad \widetilde{\zeta}_k = -\frac{1}{\bar{\omega}_k}\left(\varepsilon_k \zeta_{k-2} + \psi_k \zeta_{k-1}\right). \tag{3.8}$$

Theorem 2.3 (with f defined in (3.5)) and (3.7) imply that $\|x^\star\|^2 \leqslant \|\widetilde{z}_k\|^2$ so that (3.4) yields

$$\|x^\star - x_k^L\|^2 = \|x^\star\|^2 - \|x_k^L\|^2 \leqslant \|\widetilde{z}_k\|^2 - \|z_{k-1}\|^2 = (\epsilon_k^L)^2, \tag{3.9}$$

where we define

$$\epsilon_k^L := |\widetilde{\zeta}_k|. \tag{3.10}$$

Thus, with only a few extra floating-point operations per iteration we can compute an upper bound ϵ_k^L on the SYMMLQ error in the Euclidean-norm.

Note that this approach can be applied when a positive definite preconditioner $M \approx A$ is used. The preconditioner changes the Lanczos decomposition, but all remaining computations carry through as above. We obtain an estimate of the error in the norm defined by the preconditioner, namely $\|x^\star - x_k\|_M$.

3.2.3 Upper bounds on the CG error

We now use the error bound derived in the previous section to obtain an upper bound on the CG error in the Euclidean norm. We first establish that the CG error is always lower than that of SYMMLQ for A positive semidefinite and $Ax = b$ consistent. Although the result yields the trivial upper bound (3.10), it also allows us to identify an improved bound. Define the kth CG direction as p_k with step length $\alpha_k^C > 0$, so that $x_k^C = \sum_{j=1}^{k} \alpha_j^C p_j$.

Lemma 3.2 *The CG search directions satisfy $p_i^T p_j \geqslant 0$ for all i, j.*

The following lemma is also useful in our analysis.

Lemma 3.3 *For $1 \leqslant k \leqslant \ell$ and $0 \leqslant d_1 \leqslant d_2 \leqslant \ell - k$,*

$$(x_{k+d_2}^C)^T x_k^C \geqslant (x_{k+d_1}^C)^T x_k^C \geqslant \|x_k^C\|^2, \quad \textit{and in particular,} \quad (x^\star)^T x_k^C \geqslant \|x_k^C\|^2.$$

Proof. Because $\alpha_i^C > 0$, Lemma 3.2 yields

$$\begin{aligned} (x_{k+d_2})^T x_k^C = \left(x_k^C + \sum_{i=k+1}^{k+d_2} \alpha_i^C p_i \right)^T x_k^C &= \|x_k^C\|^2 + \sum_{i=k+1}^{k+d_2} \sum_{j=1}^{k} \alpha_i^C \alpha_j^C p_i^T p_j \\ &\geqslant \|x_k^C\|^2 + \sum_{i=k+1}^{k+d_1} \sum_{j=1}^{k} \alpha_i^C \alpha_j^C p_i^T p_j \\ &\geqslant \|x_k^C\|^2. \end{aligned}$$

□

We now relate the Euclidean-norm errors of SYMMLQ and CG.

Theorem 3.4 *Let A be positive semidefinite and $Ax = b$ be consistent and let $x^\star$ be the solution identified by both CG and SYMMLQ by virtue of Proposition 2.1. The following hold* *in exact arithmetic for all* $2 \leqslant k \leqslant \ell$*:*

$$\|x_k^L\| \leqslant \|x_k^C\|, \tag{3.11}$$
$$\|x^\star - x_k^C\| \leqslant \|x^\star - x_k^L\|. \tag{3.12}$$

Proof. Relation (3.11) follows from (3.3), and this with Lemma 3.3 implies that

$$\|x_k^L\|^2 + \|x_k^C\|^2 \leqslant 2\|x_k^C\|^2 \leqslant 2(x^\star)^T x_k^C.$$

Rearranging and adding $\|x^\star\|^2$ to both sides gives

$$\|x^\star\|^2 - 2(x^\star)^T x_k^C + \|x_k^C\|^2 \leqslant \|x^\star\|^2 - \|x_k^L\|^2.$$

By factoring the left and using result 2 of Lemma 3.1 on the right, we obtain (3.12). ☐

Although the proof of Theorem 3.4 assumes exact arithmetic, we have observed empirically that the result holds until the error in x_k^L plateaus at convergence.

Theorem 3.4 immediately establishes the trivial bound

$$\|x^\star - x_k^C\| \leqslant \|x^\star - x_k^L\| \leqslant \epsilon_k^L, \tag{3.13}$$

which provides an upper bound on the Euclidean-norm CG error, in contrast to the estimates of Meurant (2005). We can improve bound (3.13) using a few observations.

From Lemma 3.3,

$$\theta_k := (x^\star)^T x_k^C - \|x_k^C\|^2 \geqslant 0. \tag{3.14}$$

Hence from part (3.3)

$$\begin{aligned}\|x^\star - x_k^C\|^2 &= \|x^\star\|^2 - 2(x^\star)^T x_k^C + \|x_k^C\|^2 \\ &= \|x^\star\|^2 - 2\theta_k - \|x_k^C\|^2 \\ &= \|x^\star\|^2 - 2\theta_k - \|x_k^L\|^2 - \bar{\zeta}_k^2,\end{aligned}$$

and since $\|x^\star - x_k^L\| \leqslant \epsilon_k^L = |\widetilde{\zeta}_k|$ it follows that

$$\begin{aligned}\|x^\star - x_k^C\|^2 &= \|x^\star - x_k^L\|^2 - \bar{\zeta}_k^2 - 2\theta_k \\ &\leqslant \widetilde{\zeta}_k^2 - \bar{\zeta}_k^2 - 2\theta_k \qquad (3.15)\\ &\leqslant \widetilde{\zeta}_k^2 - \bar{\zeta}_k^2. \qquad (3.16)\end{aligned}$$

Since $\bar{\zeta}_k$ is readily available as part of the SYMMLQ iteration, (3.16) is an improvement upon the bound (3.13). Unfortunately, bound (3.15) is not computable because $x^\star$ is unavailable. We define

$$\epsilon_k^C := \sqrt{\widetilde{\zeta}_k^2 - \bar{\zeta}_k^2} \leqslant |\widetilde{\zeta}_k| = \epsilon_k^L \tag{3.17}$$

as an upper bound on the error of the kth CG iterate.

Using Lemma 3.3, we could further improve the error estimate by approximating θ_k from below by introducing a delay, implemented using the sliding-window approach originally appearing in Golub and Strakŏs (1994) (stabilized by Golub and Meurant (1997) and used by

Meurant (2005) and Orban and Arioli (2017)). Given Lemma 3.3, we define an approximation of (3.14) as

$$\theta_k^{(d)} := (x_{k+d}^C)^T x_k^C - \|x_k^C\|^2 \leqslant \theta_k \qquad (d > 0),$$

noting that $0 \leqslant \theta_k^{(1)} \leqslant \cdots \leqslant \theta_k^{(\ell-k)} = \theta_k$.

We now describe how to compute $\theta_k^{(d)}$ without storing the iterates $x_k^C, \dots, x_{k+d}^C$ explicitly. Recalling that $x_k^C = x_k^L + \bar\zeta_k \bar w_k = \sum_{i=1}^{k-1} \zeta_i w_i + \bar\zeta_k \bar w_k$, we have

$$\begin{aligned}
\theta_k^{(d)} &= \left(x_k^L + \bar\zeta_k \bar w_k\right)^T \left(x_{k+d}^L + \bar\zeta_{k+d} \bar w_{k+d}\right) - \left(\|x_k^L\|^2 + \bar\zeta_k^2\right) \\
&= \|x_k^L\|^2 + \bar\zeta_k \bar w_k^T x_{k+d}^L + \bar\zeta_k \bar\zeta_{k+d} \bar w_k^T \bar w_{k+d} - \left(\|x_k^L\|^2 + \bar\zeta_k^2\right) \\
&= \bar\zeta_k \sum_{i=k}^{k+d-1} \zeta_i \bar w_k^T w_i + \bar\zeta_k \bar\zeta_{k+d} \bar w_k^T \bar w_{k+d} - \bar\zeta_k^2,
\end{aligned}$$

where we use the fact that $w_i^T w_j = 0$ for $i \neq j$ and $\bar w_i^T w_j = 0$ for $j < i$. Note that

$$\bar w_k^T w_i = c_{i+1} \prod_{j=k+1}^{i} s_j \quad \text{and} \quad \bar w_k^T \bar w_i = \prod_{j=k+1}^{i} s_j \quad \text{for } i \geqslant k,$$

so that

$$\theta_k^{(d)} = \bar\zeta_k \sum_{i=k}^{k+d-1} \left(\zeta_i c_{i+1} \prod_{j=k+1}^{i} s_j \right) + \bar\zeta_k \bar\zeta_{k+d} \prod_{j=k+1}^{k+d} s_j - \bar\zeta_k^2.$$

We can compute $\theta_k^{(d)}$ in $O(d)$ flops and $O(d)$ storage by maintaining d partial products of the form $\prod_{j=k+1}^{i} s_j$ for $k+1 \leqslant i \leqslant k+d$. At the next iteration we can divide each partial product by s_{k+1} and multiply the last one by s_{k+d} to obtain the necessary partial products for iteration $k+1$.

With the above expression we can improve (3.16) to

$$\|x^\star - x_k^C\|^2 \leqslant \left(\epsilon_k^C\right)^2 - 2\theta_k^{(d)}. \tag{3.18}$$

This improved bound is only noticeable when λ_{est} is a close estimate to $\lambda_{\min}$. Otherwise, the difference between the ϵ_k^C and $\|x^\star - x_k^C\|$ is dominated by the error in the Gauss-Radau quadrature (the difference between ϵ_k^L and $\|x^\star - x_k^L\|$).

It is not necessary to implement CG via the transfer point from SYMMLQ in order to compute these error bounds because only $\{\alpha_k, \beta_k\}$ from the Lanczos process are required. These can be recovered from the classic Hestenes and Stiefel (1952) implementation of CG using equations provided by Meurant (2005).

For positive semidefinite A, we have derived upper bounds on the SYMMLQ and CG errors when $Ax = b$ is consistent. Only a few extra scalar operations are needed per iteration, and $O(1)$ extra memory.

3.3 Complete algorithm

Algorithm 3 provides the complete algorithm to compute the error bounds ϵ_k^L and ϵ_k^C, given $\{\alpha_k, \beta_k\}$ from the Lanczos process. Although it did not make a difference in our numerical

Algorithm 3 SYMMLQ with CG error estimation

```
Require: $A$, $b$, and $\lambda_{\text{est}}$ such that $\lambda_{\text{est}} < \lambda_{\min}(A)$.
 1: Obtain $\alpha_1, \beta_1, \beta_2$ of Lanczos process on $(A, b)$
 2: $\bar{\gamma}_1 = \alpha_1$, $\bar{\delta}_2 = \beta_2$, $\varepsilon_1 = \varepsilon_2 = 0$        ▷ begin QR of $\bar{L}_k$
 3: $\bar{\rho}_1 = \alpha_1 - \lambda_{\text{est}}$, $\bar{\sigma}_2 = \beta_2$, $\rho_1 = \sqrt{\bar{\rho}_1^2 + \beta_2^2}$        ▷ begin QR of (3.6)
 4: $c_0^{(\omega)} = -1$, $c_1^{(\omega)} = (\alpha_1 - \lambda_{\text{est}})/\rho_1$, $s_1^{(\omega)} = \beta_2/\rho_1$
 5: $\zeta_0 = 0$, $\bar{\zeta}_1 = \beta_1/\bar{\gamma}_1$        ▷ initialize remaining variables
 6: for $k = 2, 3, \dots$ do
 7:     $\gamma_{k-1} = \sqrt{\bar{\gamma}_{k-1}^2 + \beta_k^2}$
 8:     $c_k = \bar{\gamma}_{k-1}/\gamma_{k-1}$, $s_k = \beta_k/\gamma_{k-1}$
 9:     Obtain $\alpha_k, \beta_{k+1}$ from Lanczos process on $(A, b)$
10:     $\delta_k = \bar{\delta}_k c_k + \alpha_k s_k$, $\bar{\gamma}_k = \bar{\delta}_k s_k - \alpha_k c_k$        ▷ continue QR of $\bar{L}_k$
11:     $\varepsilon_{k+1} = \beta_{k+1} s_k$, $\bar{\delta}_{k+1} = -\beta_{k+1} c_k$
12:     $\zeta_{k-1} = \bar{\zeta}_{k-1} c_k$        ▷ forward substitution
13:     $\bar{\zeta}_k = -(\varepsilon_k \zeta_{k-2} + \delta_k \zeta_{k-1})/\bar{\gamma}_k$
14:     $\eta_{k-1} = -\beta_k^2 c_{k-2}^{(\omega)}/\bar{\rho}_{k-1}$        ▷ forward substitution on (3.6)
15:     $\omega_k = \lambda_{\text{est}} + \eta_{k-1}$
16:     $\psi_k = c_k \bar{\delta}_k + s_k \omega_k$, $\bar{\omega}_k = s_k \bar{\delta}_k - c_k \omega_k$
17:     $\epsilon_k^L = |(\varepsilon_k \zeta_{k-2} + \psi_k \zeta_{k-1})/\bar{\omega}_k|$        ▷ compute error bounds
18:     $\epsilon_k^C = \left((\epsilon_k^L)^2 - \bar{\zeta}_k^2\right)^{\frac{1}{2}}$
19:     $\bar{\rho}_k = s_{k-1}^{(\omega)} \bar{\sigma}_k - c_{k-1}^{(\omega)}(\alpha_k - \lambda_{\text{est}})$        ▷ continue QR of (3.6)
20:     $\bar{\sigma}_{k+1} = -c_{k-1}^{(\omega)} \beta_{k+1}$, $\rho_k = \sqrt{\bar{\rho}_k^2 + \beta_{k+1}^2}$
21:     $c_k^{(\omega)} = \bar{\rho}_k/\rho_k$, $s_k^{(\omega)} = \beta_{k+1}/\rho_k$
22: end for
```

experiments, it may be safer in practice to compute reflections $c_k^{(\omega)}, s_k^{(\omega)}, c_k, s_k$ using a variant of (Golub and Van Loan, 2013, §5.1.8).

3.4 Error estimation with A indefinite

We now focus on the SYMMLQ error when A is indefinite. Theorem 2.3 no longer applies, and so $\beta_1^2 e_1^T \widetilde{T}_k^{-2} e_1$ is only an estimate of $\|x^\star\|$ rather than an upper bound.

There are two approaches. The first is to continue as in Section 3.2.2 and accept ϵ_k^L as an estimate of the error rather than an upper bound. Alternatively we can treat $\|x^\star\|^2 = b^T(A^2)^\dagger b$ as a quadratic form in A^2 rather than A. (Recall that for real symmetric A, $(A^2)^\dagger = (A^\dagger)^2$.) We formulate the problem as upper bounding the energy norm $\|x^\star\| = \|b\|_{B^\dagger}$ with $B = A^2$. Such computation is akin to computing the energy norm error for CG using Gauss-Radau quadrature, which has been studied by Golub and Meurant (1997) and others. The main difficulty is that it requires applying the Lanczos process to A^2 and b, which means two applications of A per iteration of SYMMLQ (an impractical requirement in general). Although this theoretically guarantees that we obtain an upper bound on $\|x^\star\|$ (and therefore an upper bound on the error), roundoff error can diminish the quality of the estimation.

With these ideas in mind, we consider the procedure outlined in Section 3.2.2, treating $b^T(A^2)^\dagger b$ as a quadratic form in A to estimate the error. In numerical experiments we observe that the estimate often remains an upper bound, even as the iterates converge to the solution.

It is possible to loosen the error estimate by choosing a smaller value for λ_{est} to encourage the estimate to remain an upper bound; however, without knowing $\lambda_{|\min|}$, this may not be a practical solution. This is also illustrated in the numerical experiments.

Note that with A indefinite, λ_{est} should be chosen between zero and the eigenvalue closest to zero (keeping the sign of that eigenvalue). This is the only difference in the computation of ϵ_k^L. There may be iterations where $T_{k-1} - \lambda_{\text{est}} I$ becomes singular, and it may not be possible to compute ϵ_k^L for that iteration, but the QR factorization of $T_k - \lambda_{\text{est}} I$ could still remain computable at future iterations.

3.5 The choice of λ_{est}

A reasonably tight underestimate of λ_{est} is required for approaches using Gauss-Radau quadrature, such as for other error estimates proposed by Meurant (1997) and Frommer et al. (2013). The quality of our error bounds is directly dependent on the quality of the Gauss-Radau quadrature, which in turn depends on the quality of the eigenvalue estimate. Meurant and Tichý (2015) investigated the effect of λ_{est} on the quality of Gauss-Radau quadrature for the CG A-norm error.

If $\lambda_{|\min|} := \operatorname{argmin}_{\lambda \in \Lambda(A)} |\lambda|$ is known, one should choose $\lambda_{\text{est}} = (1-\epsilon)\lambda_{|\min|}$ with $\epsilon \ll 1$. In the experiments below, we usually use $\epsilon = 10^{-10}$. Choosing λ_{est} slightly closer to zero alleviates numerical stability issues in computing ω_k with a near-singular $T_k - \lambda_{\text{est}} I$. This also applies when A is indefinite.

One example where it is trivial to obtain an underestimate of the smallest eigenvalue is for shifted linear systems $(A + \delta I)x = b$ with A SPD and $\delta > 0$, where the choice $\lambda_{\text{est}} = \delta$ may give good error estimates if A is close to singularity. This is of interest for regularized least-squares problems $(A^T A + \delta^2 I)x = A^T b$ (this is further explored in Chapter 4).

When $\lambda_{|\min|}$ is not known, the choice of λ_{est} becomes application-specific. It may be possible to estimate the smallest eigenvalue as the iterations progress, similar to Frommer et al. (2013), although this is the subject of ongoing research. If no information is known about the spectrum of A, Gauss-Radau quadrature approaches such as the one presented in this chapter may not be practical.

3.6 Previous error estimates

As discussed in Section 3.2.1, there are other approaches to estimating the error in the iterates of Krylov subspace methods, particularly for CG. In this section we provide a brief overview of the approaches taken by Brezinski (1999), Meurant (2005), and Frommer et al. (2013) as applied to CG, followed by some numerical experiments comparing the approaches. Only the error estimate by Brezinski (1999) applies to SYMMLQ as well. We include this in the numerical experiments.

Brezinski (1999) describes several error estimates for nonsingular square systems, including

$$\|x^\star - x_k\| \approx \frac{\|r_k\|^2}{\|Ar_k\|}, \qquad r_k = b - Ax_k \tag{3.19}$$

Table 3.1: Cost of computing an error estimate for CG using various methods, where d is the window size for methods using a delay (denoted by *). The right column refers to whether the method guarantees an upper bound in exact arithmetic.

	Cost per iteration	Storage	Upper bound
Brezinski (1999)	$O(n + nnz(A))$	$O(1)$	Yes, if scaled by $\kappa(A)$
Meurant (2005)*	$O(1)$	$O(d)$	No
Frommer et al. (2013)*	$O(d^2)$	$O(d)$	Yes
Bound (3.17)	$O(1)$	$O(1)$	Yes
Bound (3.18)*	$O(d)$	$O(d)$	Yes

(see also Auchmuty (1992)). This estimate is simple to implement, but requires an extra product Ar_k each iteration (note that $\|r_k\|$ is cheap to compute as part of the CG and SYMMLQ iterations). The estimate can be made into an upper bound by multiplying it by the condition number of A, or an upper bound thereof, assuming the latter is known ahead of time, although this considerably loosens the estimate. Thus, such conversion to an upper bound is only possible when A is nonsingular.

Meurant (2005) uses the relation

$$\|x^\star - x_k^C\|^2 = \|b\|^2 \left(e_1^T T_n^{-2} e_1 - e_1^T T_k^{-2} e_1\right) + (-1)^k \beta_{k+1} \|x^\star - x_k^C\|_A^2 \frac{\|b\|}{\|r_k^C\|} e_k^T T_k^{-2} e_1 \tag{3.20}$$

to relate the A-norm error to that of the Euclidean error for CG iterates. The first term can be approximated by introducing a delay d and replacing $e_1^T T_n^{-2} e_1$ by $e_1^T T_{k+d}^{-2} e_1$. The A-norm error can be estimated via Gauss quadrature as described by Golub and Meurant (1997), and the remaining terms by updating a QR factorization of T_k, so that the total cost is only $O(1)$ flops per iteration.

Frommer et al. (2013) use the fact that $r_k^C = \|r_k^C\| v_{k+1}$, where v_{k+1} is the $(k+1)$th Lanczos vector, and so

$$\|x^\star - x_k^C\|^2 = \|r_k^C\|^2 v_{k+1}^T A^{-2} v_{k+1}. \tag{3.21}$$

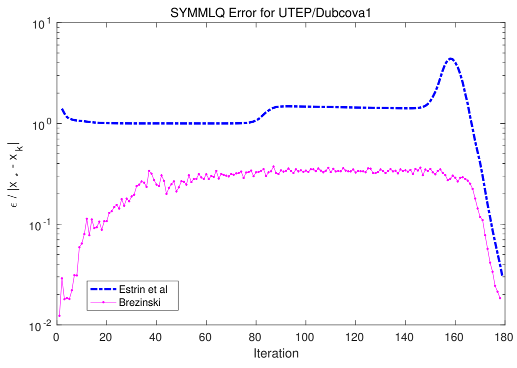
The right-hand side of (3.21) is upper-bounded using Gauss-Radau quadrature. Rather than restarting the Lanczos process on A using v_{k+1} as the initial vector at each CG iteration, they cleverly perform the Lanczos process on the lower $2d \times 2d$ submatrix of T_{k+d+1} using e_{d+1} as the starting vector, thus recovering the same estimate. The restarted Lanczos factorization requires $O(d^2)$ flops at each iteration.

In Table 3.1 we summarize the costs of the various error estimates for CG and say whether the estimate can be shown to be an upper bound in exact arithmetic.

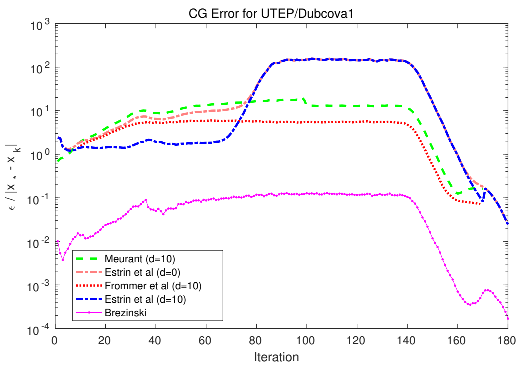
3.7 Numerical experiments

3.7.1 Comparison with previous estimates

We give some numerical examples comparing the various error estimation procedures for CG and SYMMLQ, using SPD matrices from the SuiteSparse Matrix Collection (Davis and Hu, 2011) and Matlab implementations of all error estimates described in Section 3.6. In each

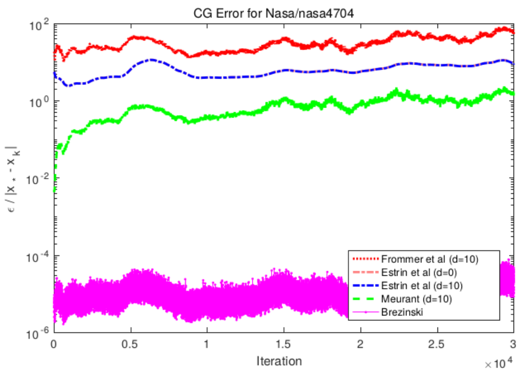


(a) SYMMLQ

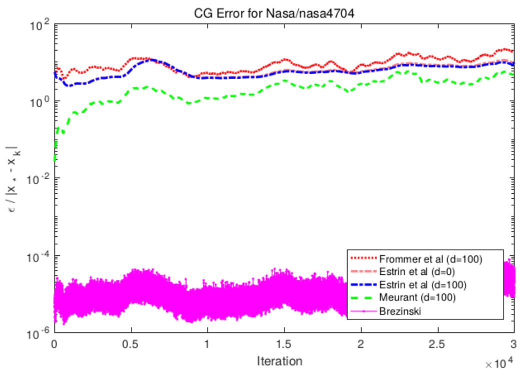


(b) CG. Window size $d = 10$ used.

Figure 3.1: $\epsilon_k/\|x^\star - x_k\|$ for SPD system UTEP/Dubcova1 using SYMMLQ and CG, where ϵ_k is the error bound for either SYMMLQ or CG.



(a) $d = 10$



(b) $d = 100$

Figure 3.2: $\epsilon_k^C/\|x^\star - x_k^C\|$ for SPD system Nasa/nasa4704. Delays $d = 10$ and 100 are used for estimates that take advantage of them.

experiment, we use $b = \mathbb{1}/\sqrt{n}$ and compute $x^\star = A\backslash b$ via Matlab. The solvers terminate when $\|r_k\|/\|b\| \leqslant 10^{-10}$. For estimates using a delay d, we report the estimated error at iteration k using information obtained during iterations $k, k+1, \dots, k+d$. Estimates requiring bounds on eigenvalues use $(1 - 10^{-10})\lambda_{\min}(A)$ for the lower bound and $(1 + 10^{-10})\lambda_{\max}(A)$ for the upper bound. (Further experiments in Section 3.7.2 use a less accurate estimate of $\lambda_{\min}(A)$.) For each approach to estimating the error, we plot $\epsilon/\|x^\star - x_k\|$, that is, the ratio of the estimate, ϵ, to the true error.

First we compare our SYMMLQ error estimate with that of Brezinski (1999). We use the matrix UTEP/Dubcova1 ($n = 16,129$ and $\kappa(A) \approx 10^3$). The ratio of the true error to the corresponding bounds are plotted in Fig. 3.1a. We see that our bound is close to the true error until x_k^L attains its maximum accuracy, whereas the Brezinski (1999) estimate is a lower bound on the error for the examples in this section; however if it is scaled by $\kappa(A)$ then it becomes a loose upper bound.

We now compare the estimates for CG from (3.17) and (3.18) using a well-conditioned system (UTEP/Dubcova1) and an ill-conditioned system (Nasa/nasa4704, $n = 4704$ and

$\kappa(A) \approx 10^7$). In Fig. 3.1b, we see that all estimates do fairly well, as they are off by at most one or two orders of magnitude. Estimate (3.17) performs nearly as well as those of Meurant (2005) and Frommer et al. (2013) when $d = 10$, until a divergence occurs near iteration 70. The improved estimate (3.18) appears tightest until that same divergence occurs.

Next, we compare against the estimates of Meurant (2005) and Frommer et al. (2013) on Nasa/nasa4704 using $d = 10$ in Fig. 3.2a and $d = 100$ in Fig. 3.2b. We see that for $d = 10$, the (Meurant, 2005) estimate is not an upper bound, while that of Frommer et al. (2013) is looser than ours. The situation is improved for the other estimates with $d = 100$, where (3.17) and those of (Meurant, 2005; Frommer et al., 2013) are fairly similar, but the Meurant (2005) estimate is still not an upper bound, and the estimate of Frommer et al. (2013) is more costly for such d. We also note that in this case, increasing d does not noticeably improve (3.18) compared to (3.17).

For CG, (3.17) is the cheapest and in exact arithmetic is guaranteed to be an upper bound. At the same time, it is not necessarily the tightest estimate, and the estimate of Frommer et al. (2013) has the advantage of improved accuracy of the error estimate with increased window size d (moreso than (3.18)), although at a higher computational cost and it requires computing d iterations into the future. In some cases, such as Fig. 3.2a, a good estimate that is not guaranteed to be a bound may more useful, but without accuracy guarantees it may be difficult to use such estimates within termination criteria.

3.7.2 Additional SPD experiments

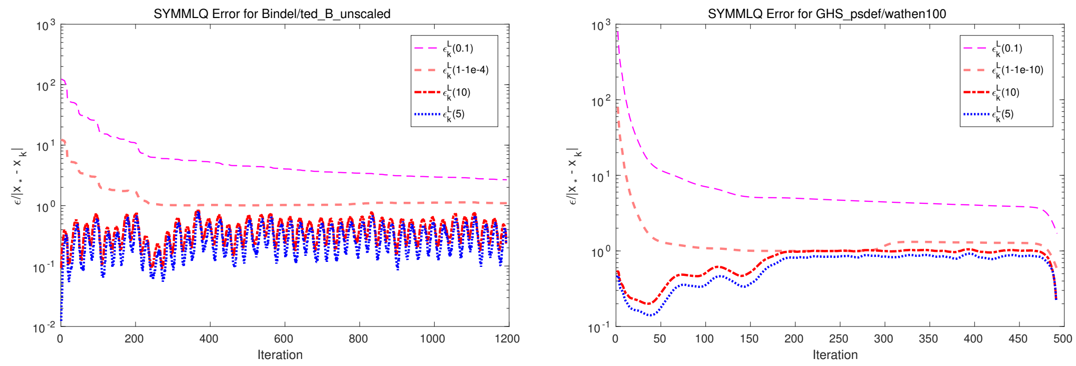
We evaluate the quality of our error bounds (3.10), (3.17) and (3.18) on further SPD examples from the SuiteSparse collection. Again we solve $Ax = b$ with $b = \mathbb{1}/\sqrt{n}$, taking $x^\star = A\backslash b$ from Matlab and terminating when $\|r_k\|/\|b\| \leqslant 10^{-10}$. We compute $\lambda_{|\min|}(A)$, the eigenvalue closest to zero, and obtain the error bounds using $\lambda_{\text{est}} = \mu\lambda_{|\min|}(A)$, typically with $\mu = 1 - 10^{-10}$ or 0.1. We also include a lower-bound error estimate using a delay (Golub and Strakŏs, 1994; Hestenes and Stiefel, 1952). Because SYMMLQ takes orthogonal steps,

$$\|x_{k+d}^L - x_k^L\|^2 = \sum_{i=k}^{k+d-1} \zeta_i^2 \leqslant \sum_{i=k}^{\ell} \zeta_i^2 = \|x^\star - x_k^L\|^2 \tag{3.22}$$

for any $d \geqslant 1$. By choosing a modest value $d = 5$ or 10 and storing the last d steplengths ζ_i, we can compute a lower bound on the error. Note that we can compute a lower bound via Gauss and Gauss-Radau quadrature with $\lambda_{\text{est}} \geqslant \|A\|_2$. Such techniques were used by Arioli (2013), and provide lower bounds comparable to those using a delay. We plot $\epsilon/\|x^\star - x_k\|$ to investigate the tightness of the bounds.

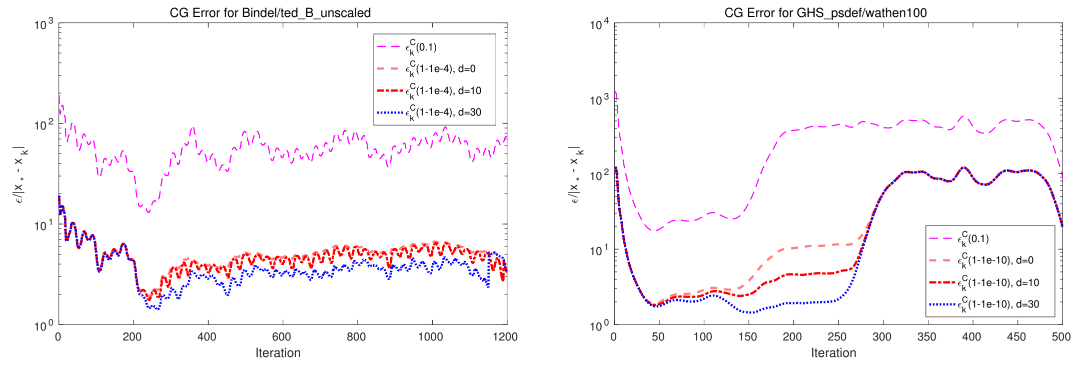
In the figure legends, $\epsilon_k^L(\mu)$ and $\epsilon_k^C(\mu)$ denote error bounds for SYMMLQ and CG obtained from Gauss-Radau quadrature when $\lambda_{\text{est}} = \mu\lambda_{|\min|}(A)$, where $0 < \mu < 1$. For SYMMLQ we include the lower-bound error obtained using a delay with $d > 1$, denoted by $\epsilon_k^L(d)$.

For SYMMLQ on Bindel/ted_B_unscaled ($n = 10605$ and $\kappa(A) \approx 10^{11}$), the bound to error ratios are shown in Figure 3.3a. For GHS_psdef/wathen100 ($n = 30401$ and $\kappa(A) \approx 10^3$), they are in Fig. 3.3b. When λ_{est} approximates $\lambda_{|\min|} = \lambda_r$ well, the bound ϵ_k^L is remarkably tight after an initial lag. We used $\mu = 1 - 10^{-6}$ for the first problem due to A being ill-conditioned ($\lambda_{|\min|} \approx 10^{-11}$), and $\mu = 1 - 10^{-10}$ for the second problem. Even when λ_{est} is a tenth of the true eigenvalue, it appears that the bound is at most an order of magnitude



(a) Bindel/ted_B_unscaled (b) GHS_psdef/wathen100

Figure 3.3: $\epsilon_k^L(\cdot)/\|x^\star - x_k^L\|$ for two SPD systems. The Gauss-Radau approach gives upper bounds, while the delay gives lower bounds.

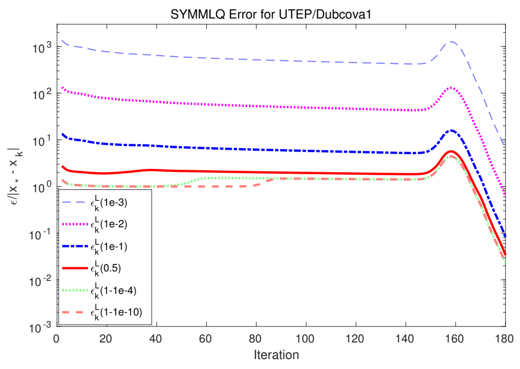


(a) Bindel/ted_B_unscaled (b) GHS_psdef/wathen100

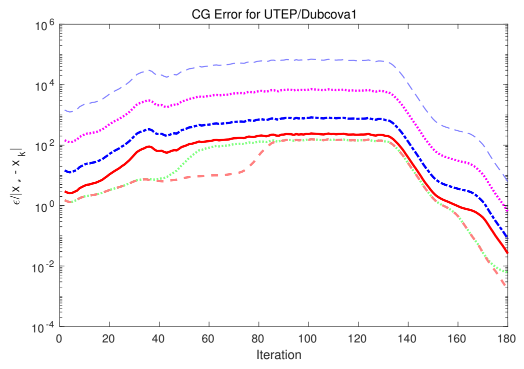
Figure 3.4: $\epsilon_k^C(\mu)/\|x^\star - x_k^C\|$ for two SPD systems.

larger, still outlining the true error from above. Only near convergence, ϵ_k^L may no longer be a bound when the true error plateaus. Having the computed bound continue to decrease after convergence is a desirable property for termination criteria. The lower bounds $\epsilon_k^L(d)$ oscillate an order of magnitude below the true error in Fig. 3.3a, but in Fig. 3.3b, both upper and lower bounds soon approximate the true error to within a couple orders of magnitude.

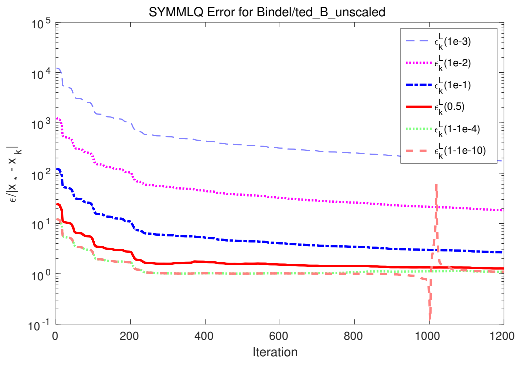
We now solve the same problems using CG. Fig. 3.4 shows that ϵ_k^C is a considerably looser bound on the CG error than ϵ_k^L is on the SYMMLQ error, although both remain true upper bounds until convergence. As with SYMMLQ, if the error stagnates at convergence, the "bound" may continue to decrease. We see that increasing d in (3.18) (when using an accurate estimate of the smallest eigenvalue) improves the bound when A is reasonably conditioned, but does not have a large impact for ill-conditioned problems. Also, ϵ_k^C diverges slightly from the true CG error when the error is roughly the square-root of the maximum attainable accuracy; in particular, d has nearly no noticeable effect past that point. This is probably due to $\bar{\zeta}_k$ becoming an order of magnitude smaller than ϵ_k^L.



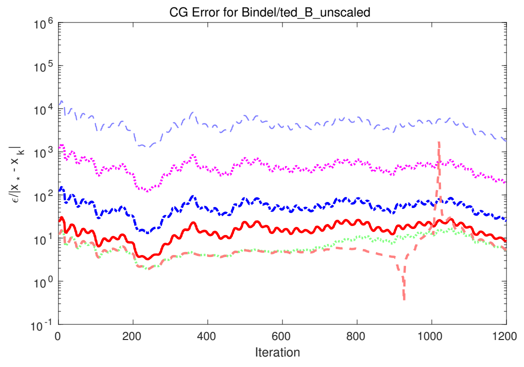
(a) SYMMLQ on UTEP/Dubcova1



(b) CG on UTEP/Dubcova1



(c) SYMMLQ on Bindel/ted_B_unscaled



(d) CG on Bindel/ted_B_unscaled

Figure 3.5: $\epsilon_k(\mu)/\|x_k - x^\star\|$ when running SYMMLQ and CG on two SPD problems for using various values of $\lambda_{\text{est}} = \mu\lambda_{|\min|}$.

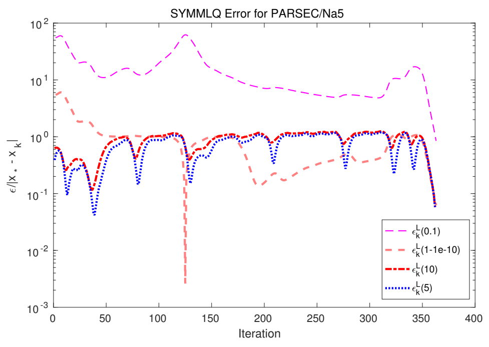
3.7.3 Empirical check

To check whether the error bounds behave as upper bounds numerically, we ran SYMMLQ and CG on all SuiteSparse matrices of size $n \leqslant 25000$ with $\kappa(A) < 10^{16}$, resulting in 140 problems. The results are in Appendix B. We used $b = \mathbb{1}/\sqrt{n}$ and $\lambda_{\text{est}} = (1 - 10^{-10})\lambda_{\min}$ or $0.1\lambda_{\min}$, and terminated when the estimate $\epsilon_k^L, \epsilon_k^C \leqslant 10^{-10}$. We then counted the number of iterations where $\epsilon_k^L \geqslant \|x^\star - x_k^L\|$ and $\epsilon_k^C \geqslant \|x^\star - x_k^C\|$ were satisfied. For $\lambda_{\text{est}} = (1-10^{-10})\lambda_{\min}$ $(0.1\lambda_{\min})$, 121 (129) problems had ϵ_k^L and ϵ_k^C behave as upper bounds for all iterations, while for the remaining 19 (11) problems we saw a cross-over at convergence similar to Fig. 3.3b, with ϵ_k^L and ϵ_k^C continuing to decrease once the true error plateaued. Thus empirically our bounds do behave as upper bounds until convergence.

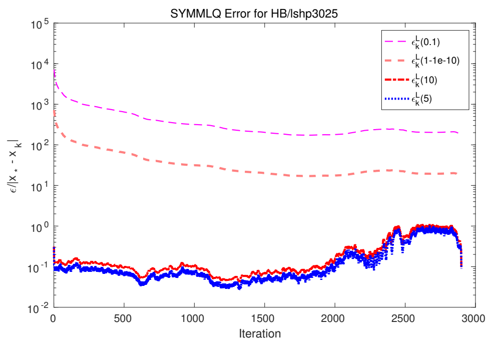
3.7.4 Effect of λ_{est}

We briefly investigate the effect of λ_{est} on the tightness of the error bounds (3.10) and (3.17). We use problems UTEP/Dubcova1 and Bindel/ted_B_unscaled again as examples of well- and ill-conditioned systems.

We observe in Figs. 3.5a and 3.5c that for SYMMLQ, $\epsilon_k^L(\mu)/\|x^\star - x_k^L\| \approx \mu^{-1}$ after an initial lag. In the case of Bindel/ted_B_unscaled, an instability occurs for $\mu = 1 - 10^{-10}$ because the smallest eigenvalue is $\lambda_{|\min|} \approx 10^{-11}$. The instability is remedied by using a



(a) PARSEC/Na5



(b) HB/lshp3025

Figure 3.6: $\epsilon_k^L(\mu)/\|x^\star - x_k^L\|$ for two indefinite systems. The Gauss-Radau approach no longer guarantees an upper bound, but works in some problems. The delay continues to provide a lower bound.

slightly larger $\mu = 1 - 10^{-4}$, resulting in an almost identical bound without the instability.

For CG in Figs. 3.5b and 3.5d, we also notice that for $\mu \leqslant 0.1$, the bound loosens by a factor of μ but keeps the same shape. The exception is when $\mu \approx 1$, where the bound is fairly tight until a divergence occurs and the bound nearly overlaps with the curve for $\mu = 0.1$. The closer μ is to 1, the later this divergence occurs; however when $\lambda_{|\min|}$ is very small (as in Fig. 3.5d), this may result in numerically unstable computations. This is because we are implicitly solving against the shifted system $T_k - \lambda_{\text{est}} I$ to compute the bound, which becomes singular as λ_{est} approaches $\lambda_{|min|}$. Similar instabilities for CG A-norm error bounds were observed in Meurant and Tichý (2015) when the true error approaches the square root of machine precision.

3.7.5 Indefinite A

We now consider indefinite examples PARSEC/Na5 and HB/lshp3025 ($n = 5822$ and 3025, $\kappa(A) \approx 10^3$ and 10^4). The former contains few negative eigenvalues, while for the latter, nearly half of its spectrum is negative. Fig. 3.6a shows that with the negative eigenvalue, (3.10) is no longer a bound for all iterations, and behaves only as an estimate which often dips below the true error. However, for many problems, such as for HB/lshp3025 in Fig. 3.6b, we see that the error estimate using $\lambda_{|\min|}$ remains an upper bound (until convergence) and tracks the true error to nearly an order of magnitude. Underestimation of $\lambda_{|\min|}$ loosens the bound, but in the case of both problems here, keeps (3.10) an upper bound to the true error, although this is again heuristic.

3.8 Finite-precision and termination considerations

We must remember that the previous sections assumed exact arithmetic, including global preservation of orthogonality of the columns of V_k. The question arises whether ϵ_k^L (3.13) and ϵ_k^C (3.17) remain upper bounds in finite precision. A rounding-error analysis is needed, similar to that of Strakoš and Tichý (2002) for CG A-norm error lower bounds, but this remains for future work. The rigorous analysis of Golub and Strakŏs (1994) shows that

Gauss-Radau quadrature may not yield upper bounds in finite precision, yet its use in finite-precision computation remains justified. In all of our numerical experiments with positive semidefinite A, we have observed that the computed ϵ_k^L and ϵ_k^C are indeed upper bounds on the errors in x_k^L and x_k^C until convergence. It may therefore be possible to derive the error bounds in this paper only using assumptions of local orthogonality in the CG and Lanczos algorithms.

For positive semidefinite A, we have seen in practice that if λ_{est} is close to λ_r, the error bounds are remarkably tight. Heuristically, we observe that when λ_{est} is loose, $|\lambda_r|/|\lambda_{\text{est}}| \approx \epsilon_k^L/\|x^\star - x_k^L\|$. It was shown in Sections 3.7.2–3.7.3 that the error estimate is an upper bound until convergence, after which the true error may plateau but ϵ_k^C and ϵ_k^L continue to decrease. This property makes it possible to terminate the iterations as soon as ϵ_k^L or ϵ_k^C drops below a prescribed level.

For CG with positive semidefinite A, we have seen that ϵ_k^C is typically one or two orders of magnitude larger than the true error for reasonable choices of λ_{est}. Using the ϵ_k^C termination criterion will ensure that the error satisfies some tolerance, but CG may take a few more iterations than necessary to achieve that tolerance.

For SYMMLQ with indefinite A, although ϵ_k^L is not guaranteed to upper bound the error, it still acts as a useful estimate of the error. Since ϵ_k^L may diverge from the exact values, if one additionally monitors the residual it would not be difficult to tell if ϵ_k^L is erroneously approaching zero. Since ϵ_k^L tends to upper bound the error near convergence, it can still be used in conjunction with other termination criteria involving the residual and related quantities, to obtain solutions that probably satisfy a given error tolerance.

Chapter 4

LSLQ: An iterative method for linear least-squares problems

We propose the iterative method LSLQ for solving least-squares problems (LS):

$$\min_{x \in \mathbb{R}^n} \ \tfrac{1}{2}\|Ax - b\|^2.$$

LSLQ can also handle linear systems (L) and least-norm problems (LN) as special cases. We often refer to the optimality conditions of (LS), namely the normal equations

$$A^T Ax = A^T b. \tag{NE-LS}$$

When $Ax = b$ is consistent, LSLQ identifies a solution of (LN). If $\operatorname{rank}(A) < n$, LSLQ finds the minimum-length solution (MLS) $x^\star = A^\dagger b$, where $A^\dagger$ is the pseudoinverse.

This chapter is based on (Estrin et al., 2019c).

4.1 Motivation

van Leeuwen and Herrmann (2016) describe a penalty method for PDE-constrained optimization in the context of a seismic inverse problem. The penalty objective $\phi_\rho(z, u)$ depends on control variable z and wavefields u, where $\rho > 0$ is a penalty parameter. For fixed values of ρ and z, the wavefields $u(z)$ satisfying $\nabla_u \phi_\rho(z, u(z)) = 0$ can be found as the solution of a linear least-squares (LS) problem in u. The gradient of ϕ with respect to z is subsequently expressed as a linear function of $u(z)$, say

$$\nabla_z \phi_\rho(z, u(z)) = G\, u(z) - g$$

for a certain matrix G and vector g. Assume now that an inexact solution $\widetilde{u}$ of the LS problem for $u(z)$ is determined. The error in u translates directly into an error in the gradient of the penalty function, for

$$\|\nabla_z \phi_\rho(z, u) - \nabla_z \phi_\rho(z, \widetilde{u})\| \leqslant \alpha\, \|u - \widetilde{u}\| + \beta\, \|u - \widetilde{u}\|^2, \qquad u \equiv u(z), \tag{4.1}$$

for certain positive constants α and β. If a derivative-based optimization method is to be used to minimize the penalty function, there is interest in a method to approximate u in which the error is monotonically decreasing. Indeed, the convergence properties of derivative-based optimization methods are not altered provided the gradient is computed sufficiently accurately in the sense that the left-hand side of (4.1) is sufficiently small compared to $\|\nabla_z \phi_\rho(z, u)\|$ (Conn, Gould, and Toint, 2000, §8.4.1.1).

We comment on the necessity for LSLQ in order to monitor the error reliably. It is

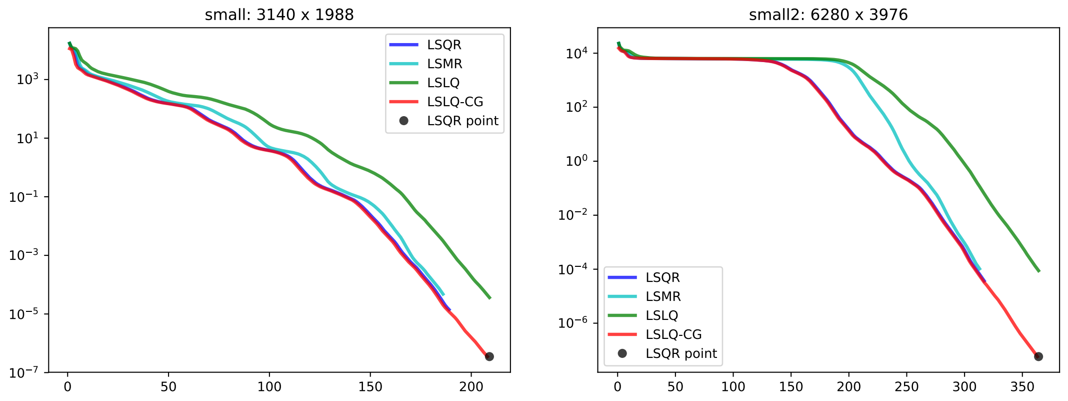


Figure 4.1: Error along the LSQR, LSMR and LSLQ iterations on problems small and small2 from the animal breeding set. The red curve corresponds to the LSQR iterates generated as a by-product during the LSLQ iterations. The horizontal axis represents the number of iterations (each involving a product with A and a product with A^T).

sufficient to say that LSLQ applied to problem (LS) is equivalent to SYMMLQ (Paige and Saunders, 1975) applied to (NE-LS). The key advantage that LSLQ inherits from SYMMLQ is that the solution estimate is updated along orthogonal directions. As a consequence, the solution norm increases and the error decreases along the iterations. It happens that both LSQR and LSMR share those properties (Fong and Saunders, 2011, Table 5.2) but with important differences. First, LSLQ's orthogonal updates suggest error lower and upper bounds initially developed for SYMMLQ in Chapter 3, and which are the subject of Section 4.4. Second, the error is *minimized* in LSLQ, while it is only monotonic in LSQR and LSMR. In spite of the latter observation, the error along the LSQR and LSMR iterations is typically smaller than for the LSLQ iterations—see Proposition 4.1. This is not a contradiction because LSLQ minimizes the error in a transformation of the Krylov subspace. Figure 4.1 illustrates a typical scenario, where the error is represented along the LSQR, LSMR, and LSLQ iterations on two over-determined problems arising from an animal breeding application (Hegland, 1990, 1993), and where we consider that the solution obtained with a complete orthogonal decomposition is the exact solution.

Our main objective is to exploit the reliable lower and upper bounds on the LSLQ error based on those developed for SYMMLQ in Chapter 3. The upper bound on the LSLQ errors combined with the tight relationship between LSLQ and LSQR leads to an upper bound on the LSQR error. Thus it becomes possible to end the LSLQ iterations as soon as it becomes apparent that the upper bound on the LSQR error is below a prescribed tolerance.

Both problems used in Figure 4.1 are rank-deficient and the curves indicate that all methods tested identify the MLS solution. Problem small2 is included in the illustration because it is an example where the error plateaus. We return to this point in section 4.4.

We do not consider LSMR further for two reasons. First, it is a consequence of (Hestenes and Stiefel, 1952, Theorem 7:5) that the LSMR error is monotonic and at least as large as that of LSQR—see also (Fong and Saunders, 2011, Theorem 2.4). Second, LSMR is a variant of MINRES (Paige and Saunders, 1975) and we know of no result relating the errors along the MINRES iterations on an SPD system to those along the SYMMLQ iterations.

4.2 Derivation

LSLQ is based on the Golub and Kahan (1965) process (Algorithm 2) from which we derive our notation. By definition, LSLQ applied to (LS) is equivalent to SYMMLQ (Paige and Saunders, 1975) applied to (NE-LS). From (2.12a) we have

$$A^TAV_k = V_{k+1}L_{k+1}^TB_k = V_{k+1}H_k, \tag{4.2}$$

where

$$H_k := \begin{bmatrix} B_k^TB_k \\ \alpha_{k+1}\beta_{k+1}e_k^T \end{bmatrix}, \tag{4.3}$$

while lines 1 and 2 of Algorithm 2 yield $A^Tb = \alpha_1\beta_1v_1$. From here on in this chapter, we use the shorthand

$$\bar{\alpha}_k := \alpha_k^2 + \beta_{k+1}^2, \quad \text{and} \quad \bar{\beta}_k := \alpha_k\beta_k, \quad k = 1, 2, \dots \tag{4.4}$$

As noted by Fong and Saunders (2011) and Section 2.2, the above characterizes the situation after $k+1$ steps of the Lanczos (1950) process applied to A^TA with initial vector A^Tb. For all $k \geqslant 1$, we denote

$$T_k := B_k^TB_k = \begin{bmatrix} \bar{\alpha}_1 & \bar{\beta}_2 & & \\ \bar{\beta}_2 & \bar{\alpha}_2 & \ddots & \\ & \ddots & \ddots & \bar{\beta}_k \\ & & \bar{\beta}_k & \bar{\alpha}_k \end{bmatrix}, \qquad H_k = \begin{bmatrix} T_k \\ \bar{\beta}_{k+1}e_k^T \end{bmatrix}. \tag{4.5}$$

The k-th iteration of CG applied to (NE-LS) computes $x_k^C = V_k\bar{x}_k^C$, where $\bar{x}_k^C$ is the solution of the subproblem

$$T_k\bar{x}_k^C = \bar{\beta}_1e_1. \tag{4.6}$$

The resulting x_k^C can be shown to solve the subproblem

$$\underset{x\in\mathcal{K}_k}{\text{minimize}} \; \|x^\star - x\|_{A^TA}, \tag{4.7}$$

where $\mathcal{K}_k := \text{span}\{A^Tb,\, (A^TA)A^Tb,\, \dots,\, (A^TA)^kA^Tb\}$ is the k-th Krylov subspace associated with A^TA and A^Tb. LSQR (Paige and Saunders, 1982a,b) is equivalent in exact arithmetic. The k-th iteration of SYMMLQ applied to (NE-LS) computes y_k^L as the solution of

$$\text{minimize } \tfrac{1}{2}\|y_k^L\|^2 \quad \text{subject to} \quad H_{k-1}^T\bar{x}_k^L = \bar{\beta}_1e_1, \tag{4.8}$$

and sets $x_k^L := V_k\bar{x}_k^L$. Note that H_{k-1}^T is the first $k-1$ rows of T_k and may be written as $H_{k-1}^T = B_{k-1}^TL_k$. Comparing to (2.9), we see that x_k^L solves the subproblem

$$\underset{x\in A^TA\mathcal{K}_{k-1}}{\text{minimize}} \; \|x^\star - x\|. \tag{4.9}$$

One important distinction between (4.7) and (4.9) is that $x_k^C \in \mathcal{K}_k$ while $x_k^L \in (A^TA)\mathcal{K}_{k-1}$, a subset of $\mathcal{K}_k$. By construction, $\|x^\star - x_k\|$ is monotonic along the LSLQ iterates, but as mentioned earlier, it also happens to be monotonic along the LSQR iterates. A corollary of

Theorem 3.4 is that the LSQR error is always smaller than the LSLQ error.

Proposition 4.1 *Let $x_k^C = V_k \bar{x}_k^C$ and $x_k^L = V_k \bar{x}_k^L$ with $\bar{x}_k^C$ and $\bar{x}_k^L$ defined as in (4.6) and (4.8). Then, for all k,*

$$\begin{aligned} \|x_k^L\| &\leqslant \|x_k^C\|, \\ \|x^\star - x_k^C\| &\leqslant \|x^\star - x_k^L\|. \end{aligned}$$

Note first that Proposition 4.1 holds whether A has full column rank or not. Note also that Proposition 4.1 does not contradict the definition of LSLQ as minimizing the error because the latter is not minimized over the same subspace as that used during the k-th iteration of LSQR.

In the next section we describe the implementation of LSLQ, and we return to the two errors in section 4.4.

4.2.1 LSLQ: implementation

We identify y_k^L by way of an LQ factorization of H_{k-1}^T (as in Section 3.1), which we compute via an implicit LQ factorization of $T_k = B_k^T B_k$. As in LSQR and LSMR we begin with the QR factorization

$$P_k^T \begin{bmatrix} B_k & \beta_1 e_1 \end{bmatrix} = \begin{bmatrix} R_k & g_k \\ 0 & \psi'_{k+1} \end{bmatrix}, \quad R_k := \begin{bmatrix} \gamma_1 & \delta_2 & & \\ & \gamma_2 & \ddots & \\ & & \ddots & \delta_k \\ & & & \gamma_k \end{bmatrix}, \quad g_k = \begin{bmatrix} \psi_1 \\ \vdots \\ \psi_k \end{bmatrix}, \tag{4.10}$$

where $P_k^T = P_{k,k+1} \dots P_{2,3} P_{1,2}$ is a product of orthogonal reflections. The j-th reflection $P_{j,j+1}$ is designed to zero out the sub-diagonal element β_{j+1} in B_k. With $\bar{\gamma}_1 := \alpha_1$ it may be represented as

$$\begin{array}{c|cc} & j & j+1 \\ \hline j & c'_j & s'_j \\ j+1 & s'_j & -c'_j \end{array} \; \begin{array}{cc} j & j+1 \\ \hline \bar{\gamma}_j & \\ \beta_{j+1} & \alpha_{j+1} \end{array} = \begin{array}{cc} j & j+1 \\ \hline \gamma_j & \delta_{j+1} \\ & \bar{\gamma}_{j+1} \end{array}, \tag{4.11}$$

where $\gamma_j = (\bar{\gamma}_j^2 + \beta_{j+1}^2)^{\frac{1}{2}}$, $c'_j = \bar{\gamma}_j / \gamma_j$, $s'_j = \beta_{j+1}/\gamma_j$, and

$$\begin{aligned} \delta_{j+1} &= s'_j \alpha_{j+1}, \\ \bar{\gamma}_{j+1} &= -c'_j \alpha_{j+1}. \end{aligned} \tag{4.12}$$

The rotations apply to the right-hand side $\beta_1 e_1$ to produce g_k defined by the recurrence

$$\psi'_1 = \beta_1, \quad \psi_k = c'_k \psi'_k, \quad \psi'_{k+1} = s'_k \psi'_k, \quad k = 1, 2, \dots \tag{4.13}$$

It will be convenient to use the notation $g'_{k+1} = (g_k, \psi'_{k+1})$.

The QR factors of B_k give the Cholesky factorization $T_k = R_k^T R_k$. To form LQ factors of T_k we take the LQ factorization

$$R_k = \overline{M}_k Q_k, \qquad \overline{M}_k := \begin{bmatrix} \varepsilon_1 & & & \\ \eta_2 & \varepsilon_2 & & \\ & \ddots & \ddots & \\ & & \eta_k & \bar{\varepsilon}_k \end{bmatrix}. \tag{4.14}$$

Initially, $\bar{\varepsilon}_1 = \gamma_1$ so that $R_1 = \overline{M}_1$. We use the notation of Paige and Saunders (1975) to indicate that $\overline{M}_k$ differs from the leading k-by-k submatrix M_k of $\overline{M}_{k+1}$ in the (k,k)-th element only, which is updated to ε_k once $\delta_{k+1} = \alpha_{k+1}\beta_{k+1}/\gamma_k$ is computed. This results in the plane reflection $Q_{k,k+1}$ defined by

$$\begin{array}{c} \\ {\scriptstyle k} \\ {\scriptstyle k+1} \end{array} \begin{array}{c} \begin{array}{cc} {\scriptstyle k} & {\scriptstyle k+1} \end{array} \\ \begin{bmatrix} \bar{\varepsilon}_k & \delta_{k+1} \\ & \gamma_{k+1} \end{bmatrix} \end{array} \begin{array}{c} \begin{array}{cc} {\scriptstyle k} & {\scriptstyle k+1} \end{array} \\ \begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix} \end{array} = \begin{array}{c} \begin{array}{cc} {\scriptstyle k} & {\scriptstyle k+1} \end{array} \\ \begin{bmatrix} \varepsilon_k & \\ \eta_{k+1} & \bar{\varepsilon}_{k+1} \end{bmatrix} \end{array}, \tag{4.15}$$

where $\varepsilon_k = (\bar{\varepsilon}_k^2 + \delta_{k+1}^2)^{\frac{1}{2}}$, $c_k = \bar{\varepsilon}_k/\varepsilon_k$, $s_k = \delta_{k+1}/\varepsilon_k$, and

$$\begin{aligned} \eta_{k+1} &= \gamma_{k+1} s_k, \\ \bar{\varepsilon}_{k+1} &= -\gamma_{k+1} c_k. \end{aligned} \tag{4.16}$$

Combining (4.10) and (4.14) gives

$$H_{k-1}^T = B_{k-1}^T L_k = \begin{bmatrix} B_{k-1}^T B_{k-1} & \alpha_k\beta_k e_{k-1} \end{bmatrix} = R_{k-1}^T \begin{bmatrix} R_{k-1} & \delta_k e_{k-1} \end{bmatrix}.$$

By construction,

$$R_k = \begin{bmatrix} R_{k-1} & \delta_k e_{k-1} \\ & \gamma_k \end{bmatrix} = \overline{M}_k Q_k = \begin{bmatrix} M_{k-1} & 0 \\ \eta_k e_{k-1}^T & \bar{\varepsilon}_k \end{bmatrix} Q_k$$

and we obtain the LQ factorization

$$H_{k-1}^T = R_{k-1}^T \begin{bmatrix} M_{k-1} & 0 \end{bmatrix} Q_k = \begin{bmatrix} R_{k-1}^T M_{k-1} & 0 \end{bmatrix} Q_k.$$

With the solution of $H_{k-1}^T \bar{x}_k^L = \bar{\beta}_1 e_1$ in mind, we consider the system $R_k^T t_k = \alpha_1\beta_1 e_1$ and obtain $t_k := (\tau_1, \ldots, \tau_k)$ by the recursion

$$\begin{aligned} \tau_1 &:= \alpha_1\beta_1/\gamma_1, \\ \tau_j &:= -\tau_{j-1}\delta_j/\gamma_j, \quad j = 2, \ldots, k. \end{aligned} \tag{4.17}$$

We also consider the systems $M_{k-1} z_{k-1} = t_{k-1}$ and $\overline{M}_k \bar{z}_k := t_k$ and obtain $z_{k-1} := (\zeta_1, \ldots, \zeta_{k-1})$ and $\bar{z}_k = (z_{k-1}, \bar{\zeta}_k)$ by the recursion

$$\begin{aligned} \zeta_1 &= \tau_1/\varepsilon_1, \\ \zeta_j &= (\tau_j - \zeta_{j-1}\eta_j)/\varepsilon_j, \quad j = 2, \ldots, k-1, \\ \bar{\zeta}_k &= (\tau_j - \zeta_{k-1}\eta_k)/\bar{\varepsilon}_k = \zeta_k/c_k. \end{aligned} \tag{4.18}$$

Then $\bar{x}_k^L = Q_k^T \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix}$ solves (4.8), while $\bar{x}_k^C = Q_k^T \bar{z}_k$ solves (4.6).

Now let $\overline{W}_k := V_k Q_k^T = \begin{bmatrix} w_1 & \dots & w_{k-1} & \bar{w}_k \end{bmatrix} = \begin{bmatrix} W_{k-1} & \bar{w}_k \end{bmatrix}$. Starting with $x_1^L := 0$ and $x_0^C := 0$ we obtain

$$x_k^L = V_k y_k^L = V_k Q_k^T \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} = \overline{W}_k \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} = W_{k-1} z_{k-1} = x_{k-1}^L + \zeta_{k-1} w_{k-1}, \tag{4.19}$$

$$x_k^C = V_k Q_k^T \bar{z}_k = \overline{W}_k \bar{z}_k = W_{k-1} z_{k-1} + \bar{\zeta}_k \bar{w}_k = x_k^L + \bar{\zeta}_k \bar{w}_k. \tag{4.20}$$

Thus, as in SYMMLQ it is always possible to transfer to the LSQR point. In terms of error, Proposition 4.1 indicates that transferring is always desirable.

At the next iteration we have $\overline{W}_{k+1} = V_{k+1} Q_{k+1}^T$, where

$$\begin{bmatrix} \bar{w}_k & v_{k+1} \end{bmatrix} \begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix} = \begin{bmatrix} w_k & \bar{w}_{k+1} \end{bmatrix}.$$

With $\bar{w}_1 := v_1$ this gives

$$w_k = c_k \bar{w}_k + s_k v_{k+1}, \tag{4.21a}$$

$$\bar{w}_{k+1} = s_k \bar{w}_k - c_k v_{k+1}. \tag{4.21b}$$

Because the columns of W_{k-1} and $\overline{W}_k$ are orthonormal in exact arithmetic, we have

$$\|x_k^L\|^2 = \|W_{k-1} z_{k-1}\|^2 = \|z_{k-1}\|^2 = \sum_{j=1}^{k-1} \zeta_j^2 = \|x_{k-1}^L\|^2 + \zeta_{k-1}^2, \tag{4.22}$$

$$\|x_k^C\|^2 = \|x_k^L\|^2 + \bar{\zeta}_k^2. \tag{4.23}$$

4.2.2 Residual estimates

The k-th LSLQ residual is defined as $r_k^L := b - A x_k^L$. We use the definition of $x_k^L = V_k \bar{x}_k^L$, (2.11), (4.10) and (4.14) to express it as

$$\begin{aligned} r_k^L = b - A V_k \bar{x}_k^L &= U_{k+1} \left(\beta_1 e_1 - B_k \bar{x}_k^L \right) \\ &= U_{k+1} P_k \left(\beta_1 P_k^T e_1 - \begin{bmatrix} R_k \\ 0 \end{bmatrix} \bar{x}_k^L \right) \\ &= U_{k+1} P_k \left(g'_{k+1} - \begin{bmatrix} \overline{M}_k Q_k \\ 0 \end{bmatrix} \bar{x}_k^L \right) \\ &= U_{k+1} P_k \left(g'_{k+1} - \begin{bmatrix} \overline{M}_k \\ 0 \end{bmatrix} \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} \right) \\ &= U_{k+1} P_k \left(g'_{k+1} - \begin{bmatrix} M_{k-1} z_{k-1} \\ \eta_k \zeta_{k-1} \\ 0 \end{bmatrix} \right) \end{aligned}$$

$$= U_{k+1} P_k \left(\begin{bmatrix} g_{k-1} \\ \psi_k \\ \psi'_{k+1} \end{bmatrix} - \begin{bmatrix} t_{k-1} \\ \eta_k \zeta_{k-1} \\ 0 \end{bmatrix} \right),$$

where g'_{k+1} is defined in (4.10) and (4.13). It is not immediately obvious that $g_{k-1} = t_{k-1}$, but note that (4.10) yields $\begin{bmatrix} R_{k-1}^T & 0 \end{bmatrix} P_{k-1}^T = B_{k-1}^T$, so that

$$R_{k-1}^T g_{k-1} = \begin{bmatrix} R_{k-1}^T & 0 \end{bmatrix} \begin{bmatrix} g_{k-1} \\ \psi'_k \end{bmatrix} = B_{k-1}^T \beta_1 e_1 = \alpha_1 \beta_1 e_1 = R_{k-1}^T t_{k-1}$$

as long as $\gamma_{k-1} \neq 0$. Therefore, if the process does not terminate, we have $g_{k-1} = t_{k-1}$ as announced. By orthogonality of U_{k+1} and P_k we have

$$\|r_k^L\|^2 = \left\| \begin{bmatrix} 0 \\ \psi_k - \eta_k \zeta_{k-1} \\ \psi'_{k+1} \end{bmatrix} \right\|^2 = (\psi_k - \eta_k \zeta_{k-1})^2 + (\psi'_{k+1})^2. \tag{4.24}$$

The residual norm for the CG-point can also be computed as

$$r_k^C := b - A x_k^C = U_{k+1} P_k \left(P_k^T \beta_1 e_1 - \begin{bmatrix} R_k \\ 0 \end{bmatrix} \bar{x}_k^C \right) = U_{k+1} P_k \left(\begin{bmatrix} g_k \\ \psi'_{k+1} \end{bmatrix} - \begin{bmatrix} R_k \\ 0 \end{bmatrix} \bar{x}_k^C \right).$$

The top k rows of the parenthesized expression vanish by definition of $\bar{x}_k^C$, and there remains

$$\|r_k^C\| = (\beta_1 P_k^T e_1)_{k+1} = |\psi'_{k+1}|.$$

To derive recurrences for the residual norm for (NE-LS), we can use the recurrences derived in (Paige and Saunders, 1975) for SYMMLQ and CG, which become

$$\begin{aligned} \|A^T r_k^L\|^2 &= (\gamma_k \epsilon_k)^2 \zeta_k^2 + (\delta_k \eta_{k-1})^2 \zeta_{k-1}^2, \\ \|A^T r_k^C\| &= \alpha_1 \beta_1 s_1 \cdots s_{k-1} s_k / c_k. \end{aligned}$$

4.2.3 Norm and condition number estimates

Assuming orthonormality of V_k, (4.2) yields $V_k^T A^T A V_k = B_k^T B_k$, so that the Poincaré separation theorem ensures $\sigma_{\min}(A) \leqslant \sigma_{\min}(B_k) \leqslant \sigma_{\max}(B_k) \leqslant \sigma_{\max}(A)$ for all k. Therefore we may use $\|B_k\|$ as an estimate of $\|A\|$ and $\text{cond}(B_k)$ as an estimate of $\text{cond}(A)$ in both the Euclidean and Frobenius norms. In particular, $\|B_{k+1}\|_F^2 = \|B_k\|_F^2 + \alpha_k^2 + \beta_{k+1}^2$.

As in (Fong and Saunders, 2011, Section 3.4), our approximation of $\text{cond}(A)$ rests on the QLP factorization

$$P_k^T B_k Q_k^T = \begin{bmatrix} M_{k-1} & 0 \\ \eta_k e_{k-1}^T & \bar{\epsilon}_k \end{bmatrix}.$$

According to Stewart (1999), the absolute values of the diagonals of the bidiagonal matrix above are tight approximations to the singular values of B_k. Thus we estimate

$$\sigma_{\min}(B_k) \approx \min(\epsilon_1, \ldots, \epsilon_{k-1}, |\bar{\epsilon}_k|), \quad \sigma_{\max}(B_k) \approx \max(\epsilon_1, \ldots, \epsilon_{k-1}, |\bar{\epsilon}_k|),$$

Algorithm 4 LSLQ

```
1: $\beta_1 u_1 = b,\ \ \alpha_1 v_1 = A^T u_1$                                  ▷ begin Golub-Kahan process
2: $\delta_1 = -1,\ \ \psi_1 = \beta_1$                                          ▷ initialize variables
3: $\tau_0 = \alpha_1\beta_1,\ \ \zeta_0 = 0$
4: $c_0 = 1,\ \ s_0 = 0$
5: $\|A^T r_0^C\| = \alpha_1\beta_1$
6: $\bar{w}_1 = v_1,\ \ x_1^L = 0$
7: for $k = 1, 2, \dots$ do
8:     $\beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k$                              ▷ continue Golub-Kahan process
9:     $\alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k$
10:    $\gamma_k = (\bar{\gamma}_k^2 + \beta_{k+1}^2)^{\frac{1}{2}},\ \ c_k' = \bar{\gamma}_k/\gamma_k,\ \ s_k' = \bar{\beta}_{k+1}/\gamma_k$    ▷ continue QR factorization
11:    $\delta_{k+1} = s_k' \alpha_{k+1}$
12:    $\bar{\gamma}_{k+1} = -c_k' \alpha_{k+1}$
13:    $\tau_k = -\tau_{k-1}\delta_k/\gamma_k$
14:    $\bar{\varepsilon}_k = -\gamma_k c_{k-1}$                                  ▷ continue LQ factorization
15:    $\eta_k = \gamma_k s_{k-1}$
16:    $\varepsilon_k = (\bar{\varepsilon}_k^2 + \delta_{k+1}^2)^{\frac{1}{2}},\ \ c_k = \bar{\varepsilon}_k/\varepsilon_k,\ \ s_k = \delta_{k+1}/\varepsilon_k$
17:    $\|r_{k-1}^L\| = ((\psi_{k-1}c_k' - \zeta_{k-1}\eta_k)^2 + (\psi_{k-1}s_k')^2)^{\frac{1}{2}}$
18:    $\psi_k = \psi_{k-1} s_k'$
19:    $\|r_k^C\| = \psi_k$
20:    $\zeta_k = (\tau_k - \zeta_{k-1}\eta_k)/\varepsilon_k$                      ▷ optional: $\bar{\zeta}_k = \zeta_k/c_k$
21:    $\|A^T r_k^L\| = (\gamma_k^2 \epsilon_k^2 \zeta_k^2 + \delta_k^2 \eta_k^2 \zeta_{k-1}^2)^{\frac{1}{2}}$    ▷ optional: $\|A^T r_k^C\| = \|A^T r_{k-1}^C\| s_k c_{k-1}/c_k$
22:    $w_k = c_k \bar{w}_k + s_k v_{k+1}$
23:    $\bar{w}_{k+1} = s_k \bar{w}_k - c_k v_{k+1}$
24:    $x_{k+1}^L = x_k^L + \zeta_k w_k$                                          ▷ optional: $x_k^C = x_k^L + \bar{\zeta}_k \bar{w}_k$
25:    $\|x_{k+1}^L\|^2 = \|x_k^L\|^2 + \zeta_k^2$                                ▷ optional: $\|x_{k+1}^C\|^2 = \|x_k^C\|^2 + \bar{\zeta}_k^2$
26: end for
```

and $\operatorname{cond}(A) \approx \sigma_{\max}(B_k)/\sigma_{\min}(B_k)$, which turns out to be reasonably accurate in practice. If $A^T b$ lies in a subspace spanned by few singular vectors of A, iterations will terminate early and $\operatorname{cond}(B_k)$ will be an improving estimate of $\operatorname{cond}(AV_\ell)$, where ℓ is the last iteration.

4.3 Complete algorithm

The complete procedure is summarized as Algorithm 4. As in (Fong and Saunders, 2011, Theorem 4.2), we can prove the following result using Proposition 2.1 applied to (NE-LS).

Theorem 4.2 *LSLQ returns the MLS solution, i.e., it solves*

$$\min_{x\in\mathbb{R}^n} \|x\| \quad \text{subject to} \quad x \in \operatorname*{argmin}_{\bar{x}} \|A\bar{x} - b\|.$$

4.4 Error estimates

In exact arithmetic, a least-squares solution $x^\star$ is identified after at most $\ell \leqslant \min(m, n)$ iterations, so that $x^\star = x_{\ell+1}^L = \sum_{j=1}^{\ell} \zeta_j w_j$. Because $x_k^L = \sum_{j=1}^{k-1} \zeta_j w_j$, the error may be written as $e_k^L = x_{\ell+1}^L - x_k^L = \sum_{j=k}^{\ell} \zeta_j w_j$. By orthogonality, $\|e_k^L\|^2 = \sum_{j=k}^{\ell} \zeta_j^2$. A possible stopping condition is

$$\|x_{k+1}^L - x_{k-d}^L\|^2 = \left(\sum_{j=k-d}^{k} \zeta_j^2 \right)^{\frac{1}{2}} \leqslant \varepsilon \|x_{k+1}^L\| \quad (k > d), \tag{4.25}$$

where $d \in \mathbb{N}$ is a delay and $0 < \varepsilon < 1$ is a tolerance. The left-hand side of (4.25) is a lower bound on the error $\|e_{k-d}^L\|$.

As we illustrate in section 4.6, (4.25) is not a robust stopping criterion because the lower bound may sometimes underestimate the actual error by several orders of magnitude. In the following sections, we develop a more robust estimate defined by an upper bound.

4.4.1 Upper bound on the LSLQ error

Chapter 3 develops an upper bound on the Euclidean error along SYMMLQ iterations for a symmetric positive semidefinite system. The bound leads to an upper bound on the error along CG iterations. We now translate those estimates to the present scenario and obtain upper bounds on the error along LSLQ and LSQR iterations for (LS). We begin with an upper bound on the LSLQ error.

By orthogonality, $\|x^\star - x_k^L\|^2 = \|x^\star\|^2 - \|x_k^L\|^2$, and because $\|x_k^L\|^2$ can be computed, an upper bound on the error will follow from an upper bound on $\|x^\star\|^2$. Assume temporarily that $m \geqslant n$ and that A has full column rank, so that A^TA is nonsingular. We may express

$$\|x^\star\|^2 = b^T A (A^T A)^{-2} A^T b = b^T A f(A^T A) A^T b,$$

where $f(\xi) := \xi^{-2}$ is defined for all $\xi \in (0, \sigma_1^2]$, and where we define $f(A^TA) := P f(\Sigma^T\Sigma) P^T$ with $A = Q\Sigma P^T$ the SVD of A. In other words, if p_i is the i-th column of P and σ_i is the i-th largest singular value of A,

$$f(A^TA) = \sum_{i=1}^{n} f(\sigma_i^2) p_i p_i^T.$$

We have from line 2 of Algorithm 2 and (4.4) that $A^Tb = \bar{\beta}_1 v_1$ and therefore

$$\|x^\star\|^2 = \bar{\beta}_1^2 \sum_{i=1}^{n} f(\sigma_i^2)\mu_i^2, \qquad \mu_i := p_i^T v_1, \ i = 1, \dots, n.$$

When A is rank-deficient, A^TA is positive semidefinite and singular, but (NE-LS) remains consistent. In addition, the MLS solution of (LS) lies in range(A^T). Let r be the smallest integer in $\{1, \dots, n\}$ such that $\sigma_{r+1} = \dots = \sigma_n = 0$ and $\sigma_r > 0$. Then $\operatorname{rank}(A) = r = \dim \operatorname{range}(A^T)$ and the smallest nonzero eigenvalue of A^TA is σ_r^2. By the Rayleigh-Ritz theorem,

$$\sigma_r^2 = \min \left\{ \|Av\|^2 \mid v \in \operatorname{range}(A^T), \ \|v\| = 1 \right\}.$$

Note that each $v_i \in \operatorname{range}(A^T)$ and that (4.2) implies $T_k = V_k^T A^T A V_k$ in exact arithmetic. Hence, for all $u \in \mathbb{R}^k$ with $\|u\| = 1$, we have $\|V_k u\| = 1$ and $u^T T_k u = \|AV_k u\|^2 \geqslant \sigma_r^2 > 0$, and each T_k is uniformly positive definite, despite the fact that A^TA is singular.

Thus, in the rank-deficient case, $A^TA = \sum_{i=1}^{r} \sigma_i^2 p_i p_i^T$. The only difference with the full-rank case is that the sum occurs over all nonzero singular values of A. Therefore, we need only redefine

$$f(\xi) := \begin{cases} \xi^{-2} & \text{if } x > 0 \\ 0 & \text{if } x = 0. \end{cases} \tag{4.26}$$

Because each x_k^L and each $x_k^C \in \operatorname{range}(A^T)$, the LSLQ and LSQR iterations occur in $\operatorname{range}(A^T)$ exactly as if they were applied to the r-by-r positive-definite system

$$P_r^T A^T A P_r \bar{x} = P_r^T A^T b,$$

where $P_r = \begin{bmatrix} p_1 & \dots & p_r \end{bmatrix}$ and $x^\star = P_r \bar{x}$. A consequence of the above discussion is that

$$\|x^\star\|^2 = \bar{\beta}_1^2 \sum_{i=1}^{r} f(\sigma_i^2)\mu_i^2, \qquad \mu_i := p_i^T v_1,\ i = 1, \dots, n.$$

The problem is thus reduced to upper bounding a quadratic form as described in Section 2.3.1 using Gauss-Radau quadrature. We begin with a paraphrase of Theorem 2.3.

Proposition 4.3 *Suppose $f : \mathbb{R} \to \mathbb{R}$ is such that $f^{(2j+1)}(\xi) < 0$ for all $\xi \in (\sigma_r^2, \sigma_1^2)$ and all $j \geqslant 0$. Fix $\sigma_{\text{est}} \in (-\sigma_r, \sigma_r)$, $\sigma_{\text{est}} \neq 0$. Let $T_k = B_k^T B_k$ be the tridiagonal generated after k steps of Algorithm 2 and $\varpi_k \in \mathbb{C}$ be chosen so that the smallest eigenvalue of*

$$\widetilde{T}_k := \begin{bmatrix} T_{k-1} & \bar{\beta}_k e_{k-1} \\ \bar{\beta}_k e_{k-1}^T & \alpha_k^2 + \varpi_k^2 \end{bmatrix}$$

is precisely σ_{est}^2. Then,

$$b^T A f(A^T A) A^T b \leqslant \bar{\beta}_1^2 e_1^T f(\widetilde{T}_k) e_1.$$

Thus Proposition 4.3 applied to f defined in (4.26) provides an upper bound on $\|x^\star\|^2$.

Note that the Poincaré separation theorem ensures that the smallest eigenvalue of each T_{k-1} is at least σ_r^2 and that the Cauchy interlace theorem guarantees that the smallest eigenvalue of $\widetilde{T}_k$ is smaller than or equal to that of T_{k-1}. Thus it is possible to choose ϖ_k satisfying the requirements of Proposition 4.3.

We now comment on the surprising fact that $\varpi_k \in \mathbb{C}$ in Proposition 4.3. To avoid forming T_k and $\widetilde{T}_k$ explicitly, we would prefer to pick a nonzero $\sigma_{\text{est}} \in (0, \sigma_r)$ and seek ϖ_k such that σ_{est} is the smallest singular value of

$$\widetilde{B}_k = \begin{bmatrix} L_k \\ \varpi_k e_k^T \end{bmatrix}. \tag{4.27}$$

The fact that $\varpi_k \in \mathbb{C}$ is a departure from the computations of Chapter 3, which established that the last diagonal of $\widetilde{T}_k$ is real: $\alpha_k^2 + \varpi_k^2 \in \mathbb{R}$. In order for ϖ_k^2 to be real, ϖ_k must be either real or purely imaginary. In a numerical implementation of (4.27), although it is possible to avoid computations in complex arithmetic, we do observe corrections ϖ_k such that the last diagonal is strictly less than α_k^2, i.e., such that ϖ_k is purely imaginary.

An alternative strategy that avoids complex numbers altogether is to pick a nonzero

$\sigma_{est} \in (0, \sigma_r)$ and seek ω_k such that σ_{est} is the smallest singular value of

$$\widetilde{R}_k = \begin{bmatrix} R_{k-1} & \delta_k e_{k-1} \\ & \omega_k \end{bmatrix}. \tag{4.28}$$

Note that $\widetilde{R}_k$ differs from R_k, the R factor in the QR factors of B_k (4.10), in the (k,k)-th entry only. In addition, if $\widetilde{R}_k$ is the Cholesky factor of $\widetilde{T}_k$, its diagonals are guaranteed to be real and positive and the smallest eigenvalue of $\widetilde{T}_k$ will be σ_{est}^2.

As earlier, the Poincaré separation theorem guarantees that the singular values of each R_{k-1}, which are the same as those of B_{k-1}, lie between σ_r and σ_1, and the Cauchy interlace theorem for singular values guarantees that it is indeed possible to choose ω_k so that the smallest singular value (4.28) is σ_{est}. We restate Proposition 4.3 with the above in mind.

Theorem 4.4 *Suppose $f : \mathbb{R} \to \mathbb{R}$ is such that $f^{(2j+1)}(\xi) < 0$ for all $\xi \in (\sigma_r^2, \sigma_1^2)$ and all $j \geqslant 0$. Fix $\sigma_{\text{est}} \in (0, \sigma_r)$. Let B_k be the bidiagonal generated after k steps of Algorithm 2 and $\omega_k > 0$ be chosen so that the smallest singular value of (4.28) is precisely σ_{est}. Then,*

$$b^T A f(A^T A) A^T b \leqslant \beta_1^2 e_1^T f(\widetilde{R}_k^T \widetilde{R}_k) e_1.$$

In order to determine ω_k, we follow Golub and Kahan (1965) and embed $\widetilde{R}_k$ into a larger symmetric matrix to change the singular value problem into an eigenvalue problem. Indeed,

$$\begin{bmatrix} 0 & \widetilde{R}_k \\ \widetilde{R}_k^T & 0 \end{bmatrix} \tag{4.29}$$

has eigenvalues $\pm\sigma_i(\widetilde{R}_k)$. Define

$$Y_{2k-2} := \begin{bmatrix} 0 & \gamma_1 & & & & & & \\ \gamma_1 & 0 & \delta_2 & & & & & \\ & \delta_2 & 0 & \gamma_2 & & & & \\ & & \gamma_2 & 0 & \delta_3 & & & \\ & & & \delta_3 & 0 & \ddots & & \\ & & & & \ddots & \ddots & \gamma_{k-1} & \\ & & & & & \gamma_{k-1} & 0 & \end{bmatrix}, \quad \widetilde{Y}_{2k} := \begin{bmatrix} Y_{2k-2} & \delta_k e_{2k-2} & \\ \delta_k e_{2k-2}^T & 0 & \omega_k \\ & \omega_k & 0 \end{bmatrix}.$$

Note that $\widetilde{Y}_{2k}$ is a symmetric permutation of (4.29) and therefore shares the same eigenvalues. If σ_{est} is an eigenvalue of $\widetilde{Y}_{2k}$ and $h^{(2k)} = (\chi_1, \ldots, \chi_{2k})$ is a corresponding eigenvector, then $(\widetilde{Y}_{2k} - \sigma_{\text{est}} I) h^{(2k)} = 0$; that is,

$$\begin{bmatrix} Y_{2k-2} - \sigma_{\text{est}} I & \delta_k e_{2k-2} & \\ \delta_k e_{2k-2}^T & -\sigma_{\text{est}} & \omega_k \\ & \omega_k & -\sigma_{\text{est}} \end{bmatrix} \begin{bmatrix} h_{2k-2}^{(2k)} \\ \chi_{2k-1} \\ \chi_{2k} \end{bmatrix} = 0.$$

Necessarily, $\chi_{2k-1} \neq 0$ because otherwise $h^{(2k)} = 0$ entirely. Thus we may fix $\chi_{2k-1} = 1$. The first block equation reads $(Y_{2k-2} - \sigma_{\text{est}} I) h_{2k-2}^{(2k)} = -\delta_k e_{2k-2}$. Let χ_{2k-2} be the last entry of $h_{2k-2}^{(2k)}$, which can be computed by updating the QR factors of Y_{2k-2} as in Section 3.2.2.

In order to compute ω_k, note that the last two equations,

$$\begin{bmatrix} \delta_k & -\sigma_{\text{est}} & \omega_k \\ & \omega_k & -\sigma_{\text{est}} \end{bmatrix} \begin{bmatrix} \chi_{2k-2} \\ 1 \\ \chi_{2k} \end{bmatrix} = 0,$$

imply that $\omega_k = \sqrt{\sigma_{\text{est}}^2 - \sigma_{\text{est}} \delta_k \chi_{2k-2}}$.

With ω_k computed, we have $\widetilde{R}_k^T \widetilde{R}_k = \widetilde{T}_k$. We are now interested in efficiently computing the upper bound

$$\|x^\star\|^2 \leqslant \bar{\beta}_1^2 e_1^T f(\widetilde{R}_k^T \widetilde{R}_k) e_1 = \bar{\beta}_1^2 e_1^T (\widetilde{R}_k^T \widetilde{R}_k)^{-2} e_1. \tag{4.30}$$

The LQ factorization $\widetilde{R}_k = \widetilde{M}_k \widetilde{Q}_k$ provides the LQ factorization $\widetilde{T}_k = \widetilde{R}_k^T \widetilde{M}_k \widetilde{Q}_k$, which in turn yields

$$\|x^\star\|^2 \leqslant \left\| \bar{\beta}_1 \widetilde{M}_k^{-1} \widetilde{R}_k^{-T} e_1 \right\|^2 = \|\widetilde{M}_k^{-1} \tilde{t}_k\|^2 = \|\tilde{z}_k\|^2,$$

where we define $\tilde{t}_k$ and $\tilde{z}_k$ from $\widetilde{R}_k^T \tilde{t}_k = \bar{\beta}_1 e_1$ and $\widetilde{M}_k \tilde{z}_k = \tilde{t}_k$ as in Section 3.2.2. We determine the LQ factorization $\widetilde{R}_k = \widetilde{M}_k \widetilde{Q}_k$ from

$$\widetilde{R}_k = \begin{bmatrix} R_{k-1} & \delta_k e_{k-1} \\ & \omega_k \end{bmatrix} = \begin{bmatrix} M_{k-1} & \\ \tilde{\eta}_k e_{k-1}^T & \tilde{\varepsilon}_k \end{bmatrix} \begin{bmatrix} Q_{k-1} & \\ & 1 \end{bmatrix}.$$

Thus $\widetilde{Q}_k = Q_k$ and $\widetilde{M}_k$ differs from M_k in the $(k, k-1)$-th and (k, k)-th entries only, which become

$$\tilde{\eta}_k = \omega_k s_{k-1}, \qquad \tilde{\varepsilon}_k = -\omega_k c_{k-1}.$$

Recalling the definition of t_k in (4.17) and z_{k-1} in (4.18) we observe that

$$\tilde{t}_k = \begin{bmatrix} t_{k-1} \\ \tilde{\tau}_k \end{bmatrix} \quad \text{and} \quad \tilde{z}_k = \begin{bmatrix} z_{k-1} \\ \tilde{\zeta}_k \end{bmatrix}, \tag{4.31}$$

where

$$\tilde{\tau}_k = -\tau_{k-1} \delta_k / \omega_k = \tau_k \gamma_k / \omega_k \quad \text{and} \quad \tilde{\zeta}_k = (\tilde{\tau}_k - \tilde{\eta}_k \zeta_{k-1}) / \tilde{\varepsilon}_k. \tag{4.32}$$

From (4.22) and orthogonality of W_k we now have

$$\|x^\star - x_k^L\|^2 = \|x^\star\|^2 - \|x_k^L\|^2 \leqslant \|z_{k-1}\|^2 + \tilde{\zeta}_k^2 - \|z_{k-1}\|^2 = \tilde{\zeta}_k^2. \tag{4.33}$$

4.4.2 Upper bound on the LSQR error

Obtaining an upper bound on the LSQR error is of interest for two reasons. First, LSLQ may transfer to the LSQR point at any iteration using a simple vector operation—see (4.20). Second, LSQR always produces a smaller error, as formalized by Proposition 4.1.

Based on Proposition 4.1, we wish to use the upper bound (4.33) and the transition (4.20) to the LSQR point to terminate LSLQ early and obtain an iterate with an error below a prescribed level. Evidently the same upper bound (4.33) could be used, but (3.17) provides the improved bound

$$\|x^\star - x_k^C\|^2 \leqslant \tilde{\zeta}_k^2 - \bar{\zeta}_k^2, \tag{4.34}$$

where $\bar{\zeta}_k$ is defined in (4.18) and $\widetilde{\zeta}_k$ is in (4.32). The bound can further be improved using (3.18) using an additional $O(d)$ flops and storage by computing $\theta_k^{(d)} \geqslant 0$ such that

$$\|x^\star - x_k^C\|^2 \leqslant \widetilde{\zeta}_k^2 - \bar{\zeta}_k^2 - 2\theta_k^{(d)}.$$

4.5 Regularization

LSLQ may be adapted to solve the regularized least-squares problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \tfrac{1}{2}\left\|\begin{bmatrix}A\\ \lambda I\end{bmatrix}x - \begin{bmatrix}b\\ 0\end{bmatrix}\right\|^2, \tag{4.35}$$

where $\lambda \geqslant 0$ is a given regularization parameter. The optimality conditions (NE-LS) become

$$(A^TA + \lambda^2 I)x = A^Tb. \tag{4.36}$$

If we run Algorithm 2 on A only, we will produce the factorization

$$\begin{bmatrix}A\\ \lambda I\end{bmatrix}V_k = \begin{bmatrix}U_{k+1} & \\ & V_k\end{bmatrix}\begin{bmatrix}B_k\\ \lambda I\end{bmatrix}, \tag{4.37}$$

which we can compare to the factorization achieved when running Algorithm 2 on the entire regularized system,

$$\begin{bmatrix}A\\ \lambda I\end{bmatrix}V_k = \hat{U}_{k+1}\hat{B}_k = \hat{U}_{k+1}\begin{bmatrix}\hat{\alpha}_1 & & \\ \hat{\beta}_2 & \ddots & \\ & \ddots & \hat{\alpha}_k\\ & & \hat{\beta}_{k+1}\end{bmatrix}. \tag{4.38}$$

Note that V_k will remain unchanged, as can be seen from the equivalence between the Golub-Kahan process and the Lanczos process on the normal equations (Saunders, 1995). Given $\hat{B}_k$, we could run the non-regularized LSLQ algorithm (using $\hat{\alpha}$ and $\hat{\beta}$ instead of α and β) to obtain all of the desired iterates and estimates. The idea is then to compute B_k via Golub-Kahan on (A, b), cheaply compute each $\hat{\alpha}_k$ and $\hat{\beta}_k$ and use them in place of α_k and β_k in the rest of the algorithm. For $k = 3$, the factorization proceeds according to Fig. 4.2.

We use β_{k+1} to zero out λ_k, which transforms α_{k+1} into $\hat{\alpha}_{k+1}$ and introduces a nonzero $\hat{\lambda}_{k+1}$ above λ in the next column. We then use a second reflection to zero out $\hat{\lambda}_{k+1}$ using λ, which produces λ_{k+1}. With $\lambda_1 = \lambda$, the recurrences for $k \geqslant 2$ are

$$\begin{aligned}\hat{\beta}_{k+1} &= (\beta_{k+1}^2 + \lambda_k^2)^{\frac{1}{2}},\\ c_k^L &= \beta_{k+1}/\hat{\beta}_{k+1},\\ s_k^L &= \lambda_k/\hat{\beta}_{k+1},\\ \hat{\alpha}_{k+1} &= c_k^L\alpha_{k+1},\\ \hat{\lambda}_{k+1} &= s_k^L\alpha_{k+1},\\ \lambda_{k+1} &= (\lambda^2 + \hat{\lambda}_{k+1}^2)^{\frac{1}{2}}.\end{aligned} \tag{4.39}$$

$$
\begin{bmatrix} \alpha_1 & & \\ \beta_2 & \alpha_2 & \\ & \beta_3 & \alpha_3 \\ & & \beta_4 \\ \lambda & & \\ & \lambda & \\ & & \lambda \end{bmatrix}
\rightarrow
\begin{bmatrix} \alpha_1 & & \\ \hat\beta_2 & \hat\alpha_2 & \\ & \beta_3 & \alpha_3 \\ & & \beta_4 \\ & \hat\lambda_2 & \\ & \lambda & \\ & & \lambda \end{bmatrix}
\rightarrow
\begin{bmatrix} \alpha_1 & & \\ \hat\beta_2 & \hat\alpha_2 & \\ & \beta_3 & \alpha_3 \\ & & \beta_4 \\ & & \\ & \lambda_2 & \\ & & \lambda \end{bmatrix}
$$

$$
\rightarrow
\begin{bmatrix} \alpha_1 & & \\ \hat\beta_2 & \hat\alpha_2 & \\ & \hat\beta_3 & \hat\alpha_3 \\ & & \beta_4 \\ & & \\ & & \hat\lambda_3 \\ & & \lambda \end{bmatrix}
\rightarrow
\begin{bmatrix} \alpha_1 & & \\ \hat\beta_2 & \hat\alpha_2 & \\ & \hat\beta_3 & \hat\alpha_3 \\ & & \beta_4 \\ & & \\ & & \\ & & \lambda_3 \end{bmatrix}
\rightarrow
\begin{bmatrix} \alpha_1 & & \\ \hat\beta_2 & \hat\alpha_2 & \\ & \hat\beta_3 & \hat\alpha_3 \\ & & \hat\beta_4 \\ & & \\ & & \\ & & \end{bmatrix}
$$

Figure 4.2: Reducing $\begin{bmatrix} B_k^T & \lambda I \end{bmatrix}^T$ to $\hat{B}_k$ for $k = 3$ via Givens rotations.

With $\lambda > 0$, the operator of (4.35) has full column rank, i.e., $r = n$, and satisfies $\sigma_n \geqslant \lambda$. Theorem 4.4 then states that we should select $\sigma_{\text{est}} \in [\lambda, \sigma_n)$ if A is nearly rank-deficient (choosing $\sigma_{\text{est}} = \lambda$ works well if A is nearly rank-deficient).

4.6 Numerical experiments

We use the Julia LSLQ implementation[1] with the relevant error bounds for the following experiments. The exact solution of (LS) was computed using a complete orthogonal decomposition of A via the `Factorize` package (Davis, 2013).

4.6.1 Problems from the animal breeding test set

In this section, we use test problems from the animal breeding collection of Hegland (1990, 1993). These over-determined problems have rank-deficiency 1, come in two flavors and sizes, and have accompanying right-hand sides. In the first flavor, a single parameter is fitted per animal, while in the second flavor, two parameters are fitted per animal and A has twice as many rows and columns. The nonzero columns of A are scaled to have unit Euclidean norm.

We begin with an illustration of the non-robust lower bound (4.25) based on a delay d. Figure 4.3 plots the actual LSLQ error along with the lower bound with delay (window size) $d = 5$ and 10 iterations for problems large and large2. The behavior seen is typical. As in the left-hand plot, the lower bound tends to follow the exact error curve tightly when the latter is strictly decreasing. But as the right-hand plot shows, it tends to underestimate the actual error by several orders of magnitude when the latter plateaus, and requires a fair number of iterations to recover, rendering the (4.25) unreliable by itself. In both plots, the stopping test used is (4.25) with $\varepsilon = 10^{-10}$. The curves for $d = 5$ and 10 are almost the same.

Figure 4.4 illustrates the behavior of our upper bound (4.33) on problems large and large2 with regularization: a typical scenario for rank-deficient problems whose smallest nonzero singular value is unknown. For a given value $\lambda \neq 0$, the smallest singular value of the regularized A is $\sigma_n = |\lambda|$. Experiments in Section 3.7.2 show that the upper bound is tighter when $|\sigma_{\text{est}}|$ is closer to $|\sigma_n|$, but they do not consider the effect of regularization. For

[1] https://github.com/JuliaSmoothOptimizers/Krylov.jl

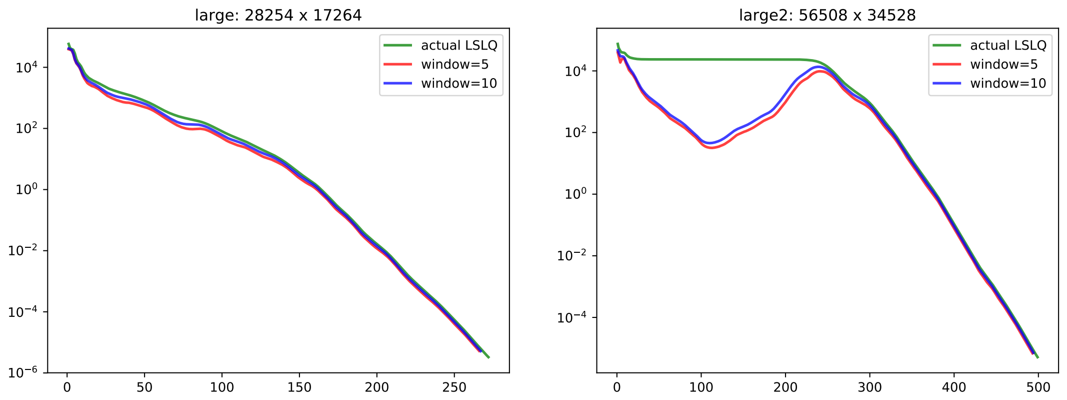


Figure 4.3: Error along the LSLQ iterations on problems large and large2 from the animal breeding set. The red and blue curves show the lower bounds with $d = 5$ and $d = 10$.

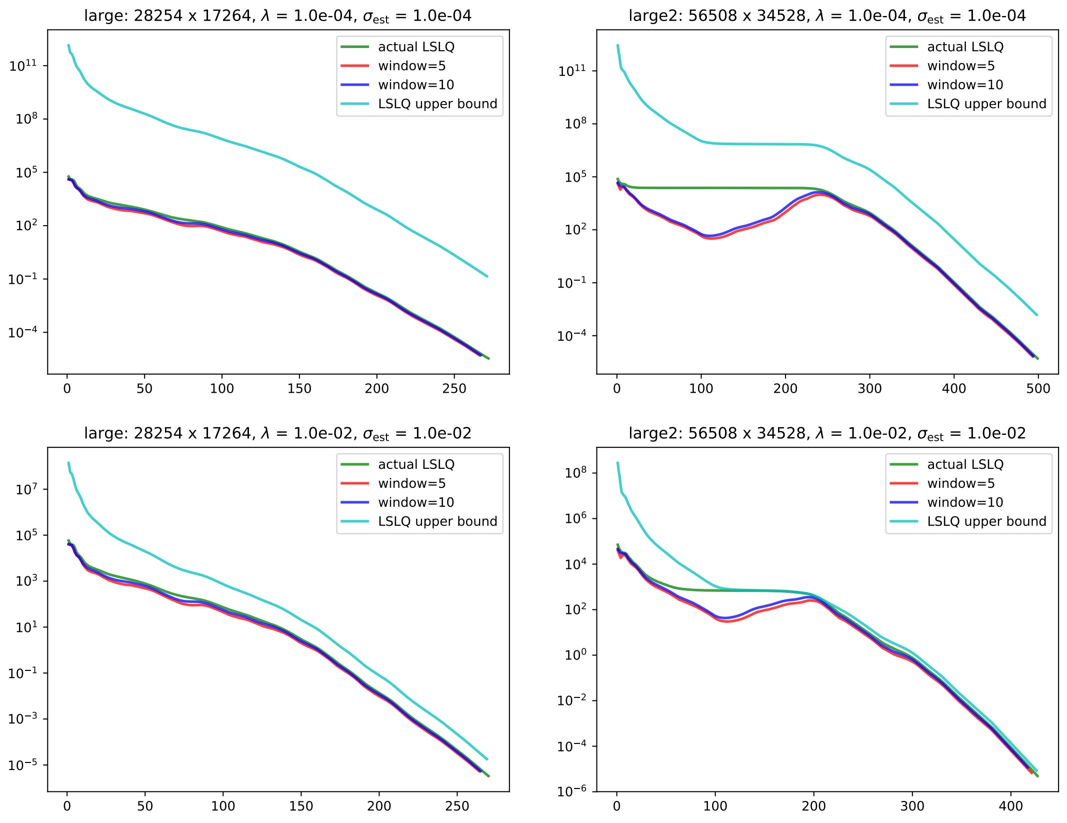


Figure 4.4: Error along the LSLQ iterations on problems large and large2 with regularization. The red and blue curves show the lower bounds with $d = 5$ and $d = 10$. The cyan curve shows the upper bounds for $\lambda = 10^{-4}$ (top) and $\lambda = 10^{-2}$ (bottom).

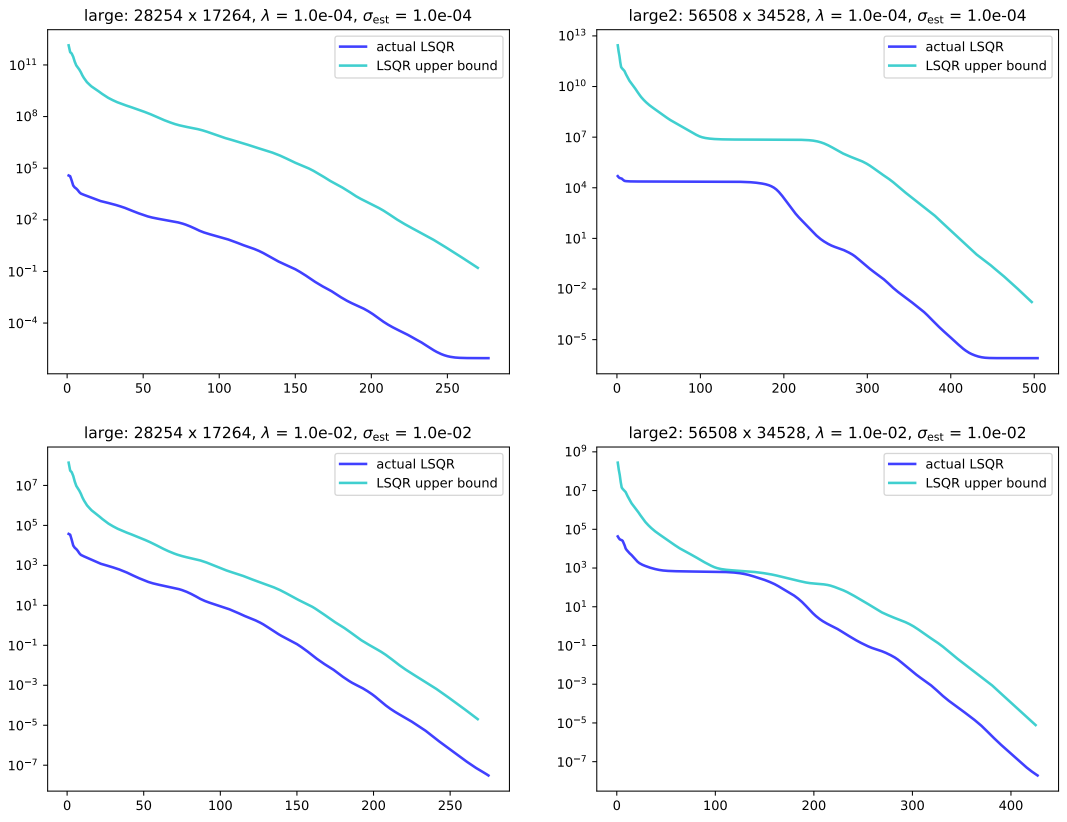


Figure 4.5: Error along the LSQR iterations on problems large and large2 with regularization. The cyan curve shows the upper bounds for $\sigma_{est} = 10^{-4}$ (top) and $\sigma_{est} = 10^{-2}$ (bottom).

each value of $\lambda > 0$, we set $\sigma_{est} := (1 - 10^{-10})\,\lambda$ and measure the error with respect to the solution of the regularized problem.

We observe from Figure 4.4 that increasing λ (and hence σ_{est}) substantially improves the quality of the upper bound. The reason may be that $\widetilde{T}_k$ is moved further away from singularity. In the case of large2 with $\lambda = 10^{-2}$, the upper bound is exceptionally tight after about 100 iterations. As λ decreases, the upper bound deteriorates, although it remains a potentially useful bound as long as $\lambda \neq 0$.

In Figure 4.5, we compute the bound (4.34) on the error along the LSQR iterates or, equivalently, along the LSQR points obtained by transitioning from a corresponding LSLQ point. As with LSLQ, the quality of the LSQR upper bound deteriorates when A, or its regularization, approaches rank-deficiency. The LSQR bound appears somewhat looser than the LSLQ bound, although it could be tightened using (4.34).

The next experiment illustrates the upper bounds for rank-deficient problems when we have knowledge of σ_r. A sparse SVD reveals that the smallest nonzero singular value after scaling is approximately $\sigma_r = \sigma_{n-1} \approx 0.0498733$ for problem small and $\sigma_r = \sigma_{n-1} \approx 0.00499044$ for small2. In each case, we set $\sigma_{est} = (1 - 10^{-10})\,\sigma_{n-1}$. In practice, one may need to underestimate further in order to account for inaccurate σ_r.

As the error bounds in Figure 4.6 are quite tight, it seems important to supply an estimate

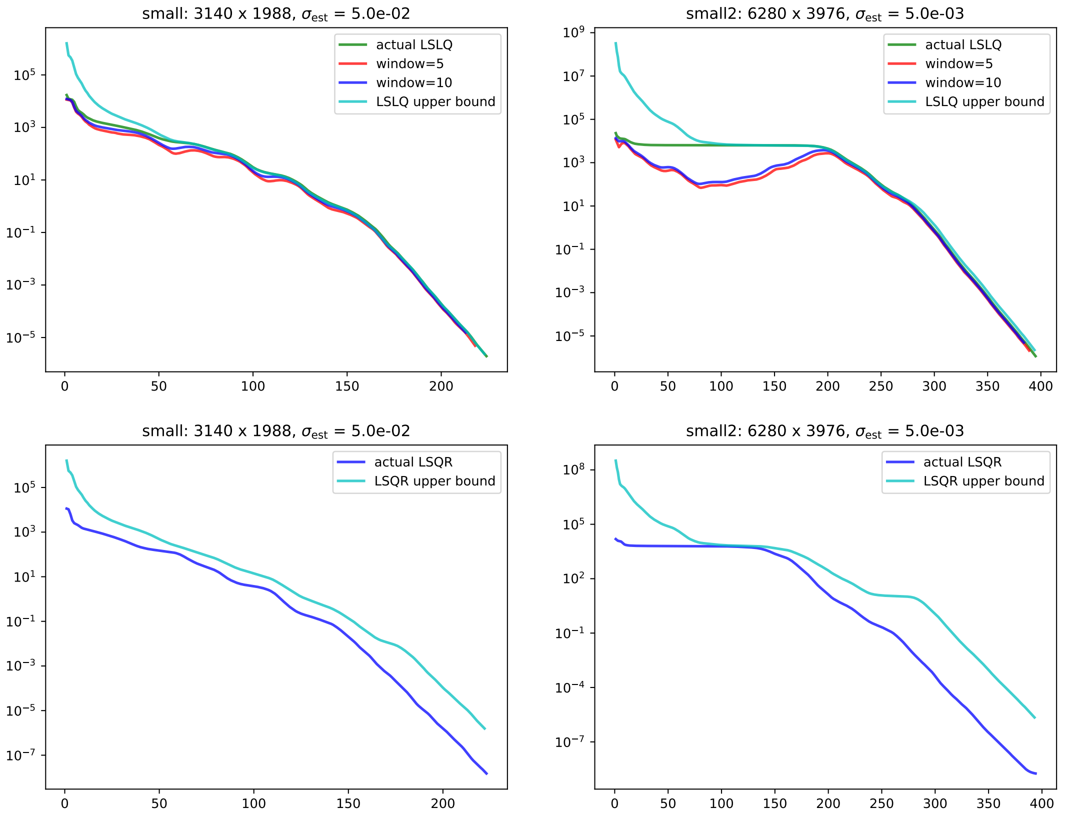


Figure 4.6: Error along the LSLQ and LSQR iterations on problems small and small2 without regularization. Both problems have rank-deficiency 1.

of σ_r in rank-deficient problems if such knowledge is available. In Figure 4.6, LSLQ stops as soon as the upper bound on the LSQR error falls below $10^{-10}\|x_k^C\|$.

4.6.2 The seismic inverse problem

The least-squares problem arising from the PDE-constrained optimization problem described in Section 4.1 has the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \frac{1}{2}\left\|\begin{bmatrix}\rho A\\ P\end{bmatrix}x-\begin{bmatrix}\rho q\\ d\end{bmatrix}\right\|^2, \tag{4.40}$$

where $\rho = 0.1$ is fixed, A is a square 5-point stencil discretization of a Helmholtz operator, P is a sampling operator (some rows of the identity), and q and d are fixed vectors. We experimented with a case in which $n = 83,600$ and P has 248 rows. The columns of the operator were not scaled as in the previous section, as that reduced the performance of LSLQ. A complete orthogonal decomposition, used to compute the exact solution, reveals that the operator of (4.40) has full rank but its smallest nonzero singular value is $O(10^{-6})$. A partial sparse SVD suggests that there are several small singular values. To obtain upper error bounds, it was necessary to set $\sigma_{est} = 10^{-7}$ to avoid domain errors in computing the square

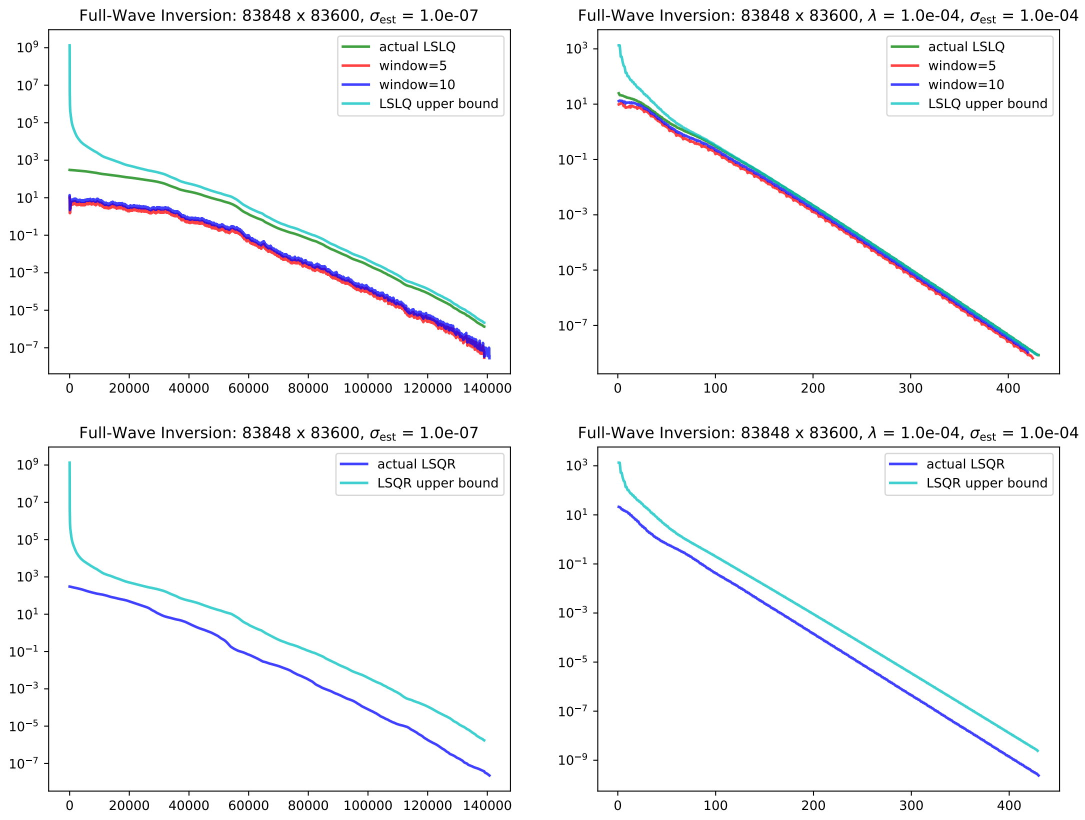


Figure 4.7: Error along the LSLQ and LSQR iterations on the seismic inverse problem without regularization (left) and with regularization (right).

root in the expression for ω_k preceding (4.30).

The left plots of Figure 4.7 illustrate the upper and lower bounds on the error and the large number of iterations needed to decrease the error by a factor of 10^{10}. The bounds on the LSLQ and LSQR errors nonetheless track the exact errors quite accurately, with the upper bound on the LSQR error overestimating by one or two orders of magnitude. Though the factor 10^{10} is far too demanding in practice, it illustrates that many iterations are likely when there are many tiny singular values. The situation is similar when the problem is regularized and the error is measured with respect to the exact solution of the original, unregularized, problem. The right plots of Figure 4.7 show the bounds in the presence of modest regularization λ when the error is computed with respect to the exact solution of the regularized problem. Dramatically fewer iterations are needed to achieve a corresponding decrease in the error. Note the remarkable tightness of the LSLQ and LSQR bounds, with the LSQR upper bound consistently overestimating by about one order of magnitude. The improved performance on the regularized problem suggests that a regularized optimization approach, such as that of Arreckx and Orban (2018), could be appropriate.

Chapter 5

LNLQ: An iterative method for linear least-norm problems

We complete the trifecta of LQ methods by introducing LNLQ for solving consistent least-norm problems (LN):

$$x^\star := \min_{x\in\mathbb{R}^n} \|x\| \text{ subject to } Ax = b.$$

A unique $y^\star$ solves the problem

$$\min_{y\in\mathbb{R}^m} \|y\| \text{ subject to } AA^T y = b, \tag{5.1}$$

and $(x^\star, y^\star)$ is the least-norm solution of the normal equations of the second kind:

$$AA^T y = b, \quad x = A^T y \qquad \Longleftrightarrow \qquad \begin{bmatrix} I & A^T \\ A & \end{bmatrix} \begin{bmatrix} x \\ -y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}. \tag{NE-LN}$$

LNLQ can also handle linear systems (L) as a special case.

This chapter is based on (Estrin et al., 2019e).

5.1 Motivation

Block linear systems of the form (NE-LN) occur during evaluation of the value and gradient of the penalty function for constrained optimization of Part II. Our main motivation is to devise reliable termination criteria that allow control of the error in the solution of (NE-LN), thus allowing us to evaluate inexact gradients cheaply while maintaining global convergence properties of the underlying optimization method. Our approach follows the philosophy of Chapters 3 and 4 and requires an estimate of the smallest singular value of A. Although such an estimate may not always be available in practice, good underestimates are often available in optimization problems, including PDE-constrained problems—see Section 5.7.

Arioli (2013) develops an upper bound on the error in x_k along the CRAIG (Section 2.2.4) iterations based on an appropriate Gauss-Radau quadrature (see Section 2.3.1), and suggests the seemingly simplistic upper bound $\|y_k - y^\star\| \leqslant \|x_k - x^\star\|/\sigma_r$, where σ_r is the smallest nonzero singular value of A. Although his bound is often effective, we derive improved bounds for CRAIG using LNLQ by introducing a delay d as in (Golub and Strakŏs, 1994).

5.2 Derivation

LNLQ is based on the Golub and Kahan process (Algorithm 2); LNLQ applied to (LN) is equivalent to SYMMLQ (Paige and Saunders, 1975) applied to (NE-LN). Identity (2.12b) yields

$$AA^TU_k = U_{k+1}B_{k+1}L_k^T = U_{k+1}H_k, \qquad \text{where } H_k := \begin{bmatrix} L_kL_k^T \\ \alpha_k\beta_{k+1}e_k^T \end{bmatrix}, \tag{5.2}$$

while line 1 of Algorithm 2 yields $b = \beta_1 u_1$. For this chapter, we use the shorthand

$$\bar{\alpha}_1 := \alpha_1^2, \quad \bar{\alpha}_k := \alpha_k^2 + \beta_k^2, \quad \text{and} \quad \bar{\beta}_k := \alpha_k\beta_{k+1}, \quad k = 2, 3, \dots \tag{5.3}$$

As noted by Fong and Saunders (2011), the above characterizes the situation after $k+1$ steps of the Lanczos (1950) process applied to AA^T with initial vector b. For $k \geqslant 1$, we denote

$$T_k := L_k^TL_k = \begin{bmatrix} \bar{\alpha}_1 & \bar{\beta}_2 & & \\ \bar{\beta}_2 & \bar{\alpha}_2 & \ddots & \\ & \ddots & \ddots & \bar{\beta}_k \\ & & \bar{\beta}_k & \bar{\alpha}_k \end{bmatrix}, \qquad H_k = \begin{bmatrix} T_k \\ \bar{\beta}_{k+1}e_k^T \end{bmatrix}. \tag{5.4}$$

Note that T_k is k-by-k and tridiagonal, and H_k is $(k+1)$-by-k.

5.2.1 CRAIG

The kth iteration of CG applied to (NE-LN) computes $y_k^C = U_k\bar{y}_k^C$, where $T_k\bar{y}_k^C = \beta_1e_1$. In exact arithmetic, $x_k^C = A^Ty_k^C$ is equivalent to the CRAIG iterate. Paige (1974) also provided an equivalent description based on Algorithm 2:

$$L_kt_k = \beta_1e_1, \qquad x_k^C := V_kt_k = x_{k-1}^C + \tau_kv_k, \tag{5.5}$$

where $t_k := (\tau_1, \dots, \tau_k)$, and the components of t_k can be found recursively from $\tau_1 = \beta_1/\alpha_1$, $\tau_j = -\beta_j\tau_{j-1}/\alpha_j$ $(j \geqslant 2)$. The residual for CRAIG is

$$r_k^C := b - Ax_k^C = \beta_1u_1 - AV_kt_k = U_{k+1}(\beta_1e_1 - B_kt_k) = -\beta_{k+1}\tau_ku_{k+1}. \tag{5.6}$$

Many of the following results can be found scattered in the literature. For completeness, we gather them here and provide proofs.

Proposition 5.1 *Let $x_\star$ be the solution of* (LN) *and $y_\star$ the associated Lagrange multiplier with minimum norm, i.e., the solution of* (5.1). *The kth CRAIG iterates x_k^C and y_k^C solve*

$$x_k^C = \underset{x}{\operatorname{argmin}} \quad \tfrac{1}{2}\|x - x_\star\|^2 \text{ subject to } x \in \operatorname{range}(V_k), \tag{5.7}$$

$$= \underset{x}{\operatorname{argmin}} \quad \tfrac{1}{2}\|x\|^2 \text{ subject to } x \in \operatorname{range}(V_k),\ b - Ax \perp \operatorname{range}(U_k), \tag{5.8}$$

$$y_k^C = \underset{y}{\operatorname{argmin}} \quad \tfrac{1}{2}\|y - y_\star\|_{AA^T}^2 \text{ subject to } y \in \operatorname{range}(U_k), \tag{5.9}$$

$$= \underset{y}{\operatorname{argmin}} \quad \tfrac{1}{2}\|y\|_{AA^T}^2 \text{ subject to } y \in \operatorname{range}(U_k),\ b - AA^Ty \perp \operatorname{range}(U_k). \tag{5.10}$$

When A is row-rank-deficient, the (AA^T)-norm should be interpreted as a norm when restricted to $\operatorname{range}(A)$.

Proof. Assume temporarily that A has full row rank, so that AA^T is symmetric positive definite. Then there exists a unique $y_\star$ such that $x_\star = A^T y_\star$ and

$$\|x_k^C - x_\star\| = \|A^T(y_k^C - y_\star)\| = \|y_k^C - y_\star\|_{AA^T}.$$

In words, the Euclidean norm of the error in x_k is the energy norm of the error in y_k. Theorem 6:1 of Hestenes and Stiefel (1952) ensures that y_k^C is chosen to minimize the energy norm of the error over all $y \in \operatorname{range}(U_k)$, i.e., y_k^C solves (5.9).

To $y \in \operatorname{range}(U_k)$, there corresponds $x = A^T y \in \operatorname{range}(A^T U_k) = \operatorname{range}(V_k L_k^T) = \operatorname{range}(V_k)$ by (2.11) because L_k is nonsingular. Consequently, CRAIG generates x_k^C as a solution of (5.7).

When A is rank-deficient, our assumption that $Ax = b$ is consistent ensures that $AA^T y = b$ is also consistent because if there exists a subpace of solutions x, it is possible to pick the one that solves (NE-LN), and therefore $b \in \operatorname{range}(AA^T)$. Kammerer and Nashed (1972) show that in the consistent singular case, CG converges to the solution $y_\star$ of (5.1). Let $r < \min(m, n)$ be such that $\sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_{\min(m,n)} = 0$. Then $\operatorname{rank}(A) = r = \dim \operatorname{range}(A)$ and the smallest nonzero eigenvalue of AA^T is σ_r^2. The Rayleigh-Ritz theorem states that

$$\sigma_r^2 = \min \{\|A^T w\|^2 \mid w \in \operatorname{range}(A),\ \|w\| = 1\}.$$

By (2.11), each $u_k \in \operatorname{range}(A)$, and (5.2) and (5.4) imply that $U_k^T AA^T U_k = T_k$ in exact arithmetic. Thus for any $t \in \mathbb{R}^k$ such that $\|t\| = 1$, we have $\|U_k t\| = 1$ and

$$t^T U_k^T AA^T U_k t = t^T T_k t \geqslant \sigma_r^2,$$

so that the T_k are uniformly positive definite and CG iterations occur as if CG were applied to the positive-definite reduced system $P_r^T AA^T P_r \tilde{y} = P_r^T b$, where P_r is the $m \times r$ matrix of orthogonal eigenvectors of AA^T corresponding to nonzero eigenvalues. Thus in the rank-deficient case, y_k^C also solves (5.9) except that the energy norm" is only a norm when restricted to $\operatorname{range}(A)$, and x_k^C also solves (5.7).

To establish (5.8), note that (5.5) and (5.6) imply that x_k^C is primal feasible for (5.8). Dual feasibility requires that there exist vectors $\bar{x}$, $\bar{y}$ and $\bar{z}$ such that $x = \bar{z} + A^T U_k \bar{y}$, $V_k^T \bar{z} = 0$ and $x = V_k \bar{x}$. The first two conditions are equivalent to $V_k^T x = 0 + V_k^T A^T U_k \bar{y} = B_k^T U_{k+1}^T U_k \bar{y} = L_k^T \bar{y}$. Because $x = V_k \bar{x}$, this amounts to $\bar{x} = L_k^T \bar{y}$. Thus dual feasibility is satisfied with $\bar{x} := \bar{x}_k^C$, $\bar{y} := \bar{y}_k^C$ and $\bar{z} := 0$. The proof of (5.10) is similar. □

5.2.2 LNLQ: implementation

By contrast, LNLQ iterates are defined by $y_k^L = U_k \bar{y}_k^L$, where $\bar{y}_k^L$ is the solution to

$$\underset{\bar{y}}{\text{minimize}}\ \tfrac{1}{2}\|\bar{y}\|^2 \quad \text{subject to} \quad H_{k-1}^T \bar{y} = \beta_1 e_1. \tag{5.11}$$

As in SYMMLQ, the computation of $\bar{y}_k^L$ follows from the LQ factorization of H_{k-1}^T, which can be derived implicitly via the LQ factorization of $T_k = L_k L_k^T$. As L_k is already lower triangular, we only need the factorization

$$L_k^T = \overline{M}_k Q_k, \qquad \overline{M}_k := \begin{bmatrix} \varepsilon_1 & & & \\ \eta_2 & \varepsilon_2 & & \\ & \ddots & \ddots & \\ & & \eta_k & \bar{\varepsilon}_k \end{bmatrix} = \begin{bmatrix} M_{k-1} & \\ \eta_k e_{k-1}^T & \bar{\varepsilon}_k \end{bmatrix}, \tag{5.12}$$

where $Q_k^T = Q_{1,2}Q_{2,3}\dots Q_{k-1,k}$ is orthogonal and defined as a product of plane reflections, where $Q_{j-1,j}$ is the identity except for elements at the intersection of rows and columns $j-1$ and j. Initially, $\bar{\varepsilon}_1 = \alpha_1$ and $Q_1 = I$. Subsequent factorization steps are represented as

$$\begin{array}{c|ccc} & {\scriptstyle j-2} & {\scriptstyle j-1} & {\scriptstyle j} \\ \hline {\scriptstyle j-1} & \eta_{j-1} & \bar{\varepsilon}_{j-1} & \beta_j \\ {\scriptstyle j} & & & \alpha_j \end{array} \begin{array}{ccc} {\scriptstyle j-2} & {\scriptstyle j-1} & {\scriptstyle j} \\ \hline 1 & & \\ & c_j & s_j \\ & s_j & -c_j \end{array} = \begin{array}{ccc} {\scriptstyle j-2} & {\scriptstyle j-1} & {\scriptstyle j} \\ \hline \eta_{j-1} & \varepsilon_{j-1} & \\ & \eta_j & \bar{\varepsilon}_j \end{array},$$

where the border indices indicate row and column numbers, with the understanding that η_{j-1} is absent when $j = 2$. For $j \geqslant 2$, $Q_{j-1,j}$ is defined by

$$\varepsilon_{j-1} = \sqrt{\bar{\varepsilon}_{j-1}^2 + \beta_j^2}, \quad c_j = \bar{\varepsilon}_{j-1}/\varepsilon_{j-1}, \quad s_j = \beta_j/\varepsilon_{j-1},$$

and the application of $Q_{j-1,j}$ results in

$$\eta_j = \alpha_j s_j, \quad \bar{\varepsilon}_j = -\alpha_j c_j. \tag{5.13}$$

We may write $H_{k-1}^T = \begin{bmatrix} L_{k-1}L_{k-1}^T & \alpha_{k-1}\beta_k e_{k-1} \end{bmatrix} = L_{k-1} \begin{bmatrix} L_{k-1}^T & \beta_k e_{k-1} \end{bmatrix}$. From (5.12),

$$L_k^T = \begin{bmatrix} L_{k-1}^T & \beta_k e_{k-1} \\ & \alpha_k \end{bmatrix} = \begin{bmatrix} M_{k-1} & \\ \eta_k e_{k-1}^T & \bar{\varepsilon}_k \end{bmatrix} Q_k \quad \Rightarrow \quad \begin{bmatrix} L_{k-1}^T & \beta_k e_{k-1} \end{bmatrix} = \begin{bmatrix} M_{k-1} & 0 \end{bmatrix} Q_k.$$

Finally, we obtain the LQ factorization

$$H_{k-1}^T = \begin{bmatrix} L_{k-1}M_{k-1} & 0 \end{bmatrix} Q_k. \tag{5.14}$$

5.2.3 Definition and update of LNLQ iterates

To solve $H_{k-1}^T \bar{y}_k^L = \beta_1 e_1$ using (5.14), note that we already have $L_{k-1}t_{k-1} = \beta_1 e_1$ (5.5), with the next iteration giving $\tau_k = -\beta_k \tau_{k-1}/\alpha_k$. Next, we consider $M_{k-1}z_{k-1} = t_{k-1}$ and find the components of $z_{k-1} = (\zeta_1, \dots, \zeta_{k-1})$ recursively as $\zeta_1 = \tau_1/\varepsilon_1$, $\zeta_j = (\tau_j - \eta_j\zeta_{j-1})/\varepsilon_j$ $(j \geqslant 2)$. This time, the next iteration yields $\bar{\zeta}_k = (\tau_k - \eta_k\zeta_{k-1})/\bar{\varepsilon}_k$ and $\zeta_k = \bar{\zeta}_k\bar{\varepsilon}_k/\varepsilon_k = c_{k+1}\bar{\zeta}_k$. Thus,

$$\bar{y}_k^L = Q_k^T \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} \qquad \text{and} \qquad \bar{y}_k^C = Q_k^T \begin{bmatrix} z_{k-1} \\ \bar{\zeta}_k \end{bmatrix} = Q_k^T \bar{z}_k \tag{5.15}$$

solve (5.11) and $T_k \bar{y}_k^C = \beta_1 e_1$ respectively, matching the definition of the CRAIG iterate.

By construction, $y_k^L = U_k \bar{y}_k^L$ and $y_k^C = U_k \bar{y}_k^C$. We define the orthogonal matrix

$$\overline{W}_k = U_k Q_k^T = \begin{bmatrix} w_1 & \cdots & w_{k-1} & \bar{w}_k \end{bmatrix} = \begin{bmatrix} W_{k-1} & \bar{w}_k \end{bmatrix}, \quad \bar{w}_1 := u_1,$$

so that (5.15) with z_{k-1} and $\bar{z}_k := (z_{k-1}, \bar{\zeta}_k)$ yields the orthogonal updates

$$y_k^L = \overline{W}_k \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} = W_{k-1} z_{k-1} = y_{k-1}^L + \zeta_{k-1} w_{k-1}, \tag{5.16}$$

$$y_k^C = \overline{W}_k \bar{z}_k = W_{k-1} z_{k-1} + \bar{\zeta}_k \bar{w}_k = y_k^L + \bar{\zeta}_k \bar{w}_k. \tag{5.17}$$

Because $\overline{W}_k$ is orthogonal, we have

$$\|y_k^L\|^2 = \|z_{k-1}\|^2 = \sum_{j=1}^{k-1} \zeta_j^2 \quad \text{and} \quad \|y_k^C\|^2 = \|y_k^L\|^2 + \bar{\zeta}_k^2. \tag{5.18}$$

Thus $\|y_k^C\| \geqslant \|y_k^L\|$, $\|y_k^L\|$ is monotonically increasing, $\|y_\star - y_k^L\|$ is monotonically decreasing, and $\|y_\star - y_k^L\| \geqslant \|y_\star - y_k^C\|$, consistent with Theorem 3.4.

Contrary to the update of y_k^C in CRAIG, y_k^L is updated along orthogonal directions and y_k^C is found as an orthogonal update of y_k^L. The latter follows from the transfer procedure of SYMMLQ to the CG point described by Paige and Saunders (1975).

At the next iteration,

$$\begin{bmatrix} w_k & \bar{w}_{k+1} \end{bmatrix} = \begin{bmatrix} \bar{w}_k & u_{k+1} \end{bmatrix} \begin{array}{c} \begin{array}{cc} \scriptstyle k & \scriptstyle k+1 \end{array} \\ \begin{bmatrix} c_{k+1} & s_{k+1} \\ s_{k+1} & -c_{k+1} \end{bmatrix} \end{array}$$

$$\begin{aligned} \Rightarrow \quad w_k &= c_{k+1} \bar{w}_k + s_{k+1} u_{k+1}, \\ \bar{w}_{k+1} &= s_{k+1} \bar{w}_k - c_{k+1} u_{k+1}. \end{aligned}$$

5.2.4 Residual estimates

We define the residual

$$r_k := b - Ax_k = b - AA^T U_k \bar{y}_k = U_{k+1}(\beta_1 e_1 - H_k \bar{y}_k)$$

using line 1 of Algorithm 2 and (5.2), where $\bar{y}_k$ is either $\bar{y}_k^L$ or $\bar{y}_k^C$. Then for $k > 1$,

$$\begin{aligned} T_k \bar{y}_k^L = L_k L_k^T \bar{y}_k^L = L_k \overline{M}_k Q_k Q_k^T \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} \\ = \begin{bmatrix} L_{k-1} & \\ \beta_k e_{k-1}^T & \alpha_k \end{bmatrix} \begin{bmatrix} M_{k-1} & \\ \eta_k e_{k-1}^T & \bar{\varepsilon}_k \end{bmatrix} \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} \\ = \begin{bmatrix} L_{k-1} & \\ \beta_k e_{k-1}^T & \alpha_k \end{bmatrix} \begin{bmatrix} t_{k-1} \\ \eta_k \zeta_{k-1} \end{bmatrix} = \begin{bmatrix} \beta_1 e_1 \\ \beta_k \tau_{k-1} + \alpha_k \eta_k \zeta_{k-1} \end{bmatrix}, \end{aligned}$$

where we use (5.12), the definition of t_{k-1} and z_{k-1}, and (5.15). Note also that the identity $Q_k e_k = s_k e_{k-1} - c_k e_k$ yields

$$e_k^T \bar{y}_k^L = e_k^T Q_k^T \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} = s_k \zeta_{k-1}.$$

The above and (5.2) combine to give

$$\begin{aligned} r_k^L &= U_{k+1} \left(\begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} - \begin{bmatrix} L_k L_k^T \\ \bar{\beta}_{k+1} e_k^T \end{bmatrix} \bar{y}_k^L \right) = -\, U_{k+1} \begin{bmatrix} 0 \\ \beta_k \tau_{k-1} + \alpha_k \eta_k \zeta_{k-1} \\ \bar{\beta}_{k+1} s_k \zeta_{k-1} \end{bmatrix} \\ &= -\, (\beta_k \tau_{k-1} + \alpha_k \eta_k \zeta_{k-1}) u_k - \bar{\beta}_{k+1} s_k \zeta_{k-1} u_{k+1}. \end{aligned} \tag{5.19}$$

By orthogonality, the residual norm is cheaply computable as

$$\|r_k^L\|^2 = (\beta_k \tau_{k-1} + \alpha_k \eta_k \zeta_{k-1})^2 + (\bar{\beta}_{k+1} s_k \zeta_{k-1})^2.$$

Similarly,

$$\begin{aligned} r_k^C &= U_{k+1} \left(\begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} - \begin{bmatrix} T_k \\ \bar{\beta}_{k+1} e_k^T \end{bmatrix} \bar{y}_k^C \right) = -\, U_{k+1} \begin{bmatrix} 0 \\ \bar{\beta}_{k+1} e_k^T \end{bmatrix} Q_k^T \bar{z}_k \\ &= -\, \bar{\beta}_{k+1} U_{k+1} \begin{bmatrix} 0 \\ s_k e_{k-1}^T - c_k e_k^T \end{bmatrix} \begin{bmatrix} z_{k-1} \\ \bar{\zeta}_k \end{bmatrix} \\ &= -\bar{\beta}_{k+1} (s_k \zeta_{k-1} - c_k \bar{\zeta}_k) u_{k+1}, \end{aligned} \tag{5.20}$$

where we use $T_k \bar{y}_k^C = \beta_1 e_1$ (by definition) and (5.15). Orthogonality of the u_j yields orthogonality of the CRAIG residuals, a property of CG (Hestenes and Stiefel, 1952, Theorem 5:1). The CRAIG residual norm is simply

$$\|r_k^C\| = \bar{\beta}_{k+1} \, |s_k \zeta_{k-1} - c_k \bar{\zeta}_k|.$$

In the next section, alternative expressions of $\|r_k^L\|$ and $\|r_k^C\|$ emerge.

5.2.5 Updating $x = A^T y$

The definition $y_k = U_k \bar{y}_k$ and (2.11) yield $x_k = A^T y_k = A^T U_k \bar{y}_k = V_k L_k^T \bar{y}_k$. The LNLQ and CRAIG iterates may then be updated as

$$\begin{aligned} x_k^L = V_k L_k^T \bar{y}_k^L &= V_k L_k^T Q_k \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} \\ &= V_k \overline{M}_k \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} = V_k \begin{bmatrix} M_{k-1} & \\ \eta_k e_{k-1}^T & \bar{\varepsilon}_k \end{bmatrix} \begin{bmatrix} z_{k-1} \\ 0 \end{bmatrix} \\ &= V_{k-1} M_{k-1} z_{k-1} + \eta_k \zeta_{k-1} v_k \\ &= V_{k-1} t_{k-1} + \eta_k \zeta_{k-1} v_k, \end{aligned} \tag{5.21}$$

and similarly,

$$x_k^C = V_k \begin{bmatrix} M_{k-1} & \\ \eta_k e_{k-1}^T & \bar{\varepsilon}_k \end{bmatrix} \begin{bmatrix} z_{k-1} \\ \bar{\zeta}_k \end{bmatrix} = x_k^L + \bar{\varepsilon}_k \bar{\zeta}_k v_k. \tag{5.22}$$

Because V_k is orthogonal, we have

$$\|x_k^L\|^2 = \sum_{j=1}^{k-1} \tau_j^2 + (\eta_k \zeta_{k-1})^2 \quad \text{and} \quad \|x_k^C\|^2 = \sum_{j=1}^{k-1} \tau_j^2 + (\eta_k \zeta_{k-1} + \bar{\varepsilon}_k \bar{\zeta}_k)^2. \tag{5.23}$$

Both x_k^L and x_k^C may be found conveniently if we maintain the delayed iterate $\tilde{x}_{k-1} := \tau_1 v_1 + \cdots + \tau_{k-1} v_{k-1} = \tilde{x}_{k-2} + \tau_{k-1} v_{k-1}$, for then we have the orthogonal updates

$$x_k^L = \tilde{x}_{k-1} + \eta_k \zeta_{k-1} v_k \quad \text{and} \quad x_k^C = \tilde{x}_{k-1} + (\eta_k \zeta_{k-1} + \bar{\varepsilon}_k \bar{\zeta}_k) v_k. \tag{5.24}$$

Proposition 5.2 *We have $\bar{\varepsilon}_1 \bar{\zeta}_1 = \tau_1$ and for $k > 1$, $\eta_k \zeta_{k-1} + \bar{\varepsilon}_k \bar{\zeta}_k = \tau_k$. This gives the same expressions as for standard CRAIG:*

$$x_k^C = \sum_{j=1}^{k} \tau_k v_k \quad \textit{and} \quad r_k^C = -\beta_{k+1} \tau_k u_{k+1}.$$

Proof. The identity for $k = 1$ follows from the definitions of $\bar{\varepsilon}_1$, $\bar{\zeta}_1$, and τ_1. By definition of $\bar{\zeta}_k$, we have $\bar{\varepsilon}_k \bar{\zeta}_k = \tau_k - \eta_k \zeta_{k-1}$, i.e., $\eta_k \zeta_{k-1} + \bar{\varepsilon}_k \bar{\zeta}_k = \tau_k$. The expressions for x_k^C and r_k^C follow from (5.24) and from (5.20), the definition of $\bar{\beta}_{k+1}$, and (5.13). ∎

Proposition 5.2 shows that x_k^C is updated along orthogonal directions, so that $\|x_k^C\|$ is monotonically increasing and $\|x_\star - x_k^C\|$ is monotonically decreasing, as stated by Paige (1974). Finally, (5.21) and Proposition 5.2 give $x_k^L = x_{k-1}^C + \eta_k \zeta_{k-1} v_k$.

Proposition 5.2 allows us to write $\tau_k - \eta_k \zeta_{k-1} = \bar{\epsilon}_k \bar{\zeta}_k$. Because $\beta_k \tau_{k-1} = -\alpha_k \tau_k$, the LNLQ residual may be rewritten

$$\begin{aligned} r_k^L &= \alpha_k (\tau_k - \eta_k \zeta_{k-1}) u_k - \bar{\beta}_{k+1} s_k \zeta_{k-1} u_{k+1} \\ &= \alpha_k \bar{\epsilon}_k \bar{\zeta}_k u_k - \alpha_k \beta_{k+1} s_k \zeta_{k-1} u_{k+1}, \end{aligned}$$

and correspondingly, $\|r_k^L\|^2 = \alpha_k^2 ((\bar{\epsilon}_k \bar{\zeta}_k)^2 + (\beta_{k+1} s_k \zeta_{k-1})^2)$.

Proposition 5.3 *Let $x_\star$ and $y_\star$ solve (5.1) and (NE-LN). The kth LNLQ iterates y_k^L and x_k^L solve*

$$\begin{aligned} x_k^L &= \underset{x}{\operatorname{argmin}} \quad \tfrac{1}{2}\|x - x_\star\|^2_{(AA^T)^\dagger} \text{ subject to } x \in \operatorname{range}(V_{k-1}), && (5.25) \\ &= \underset{x}{\operatorname{argmin}} \quad \tfrac{1}{2}\|x\|^2_{(AA^T)^\dagger} \text{ subject to } x \in \operatorname{range}(V_k),\ b - Ax \perp \operatorname{range}(U_{k-1}), && (5.26) \\ y_k^L &= \underset{y}{\operatorname{argmin}} \quad \tfrac{1}{2}\|y - y_\star\|^2 \text{ subject to } y \in \operatorname{range}(AA^T U_{k-1}), && (5.27) \\ &= \underset{y}{\operatorname{argmin}} \quad \tfrac{1}{2}\|y\|^2 \text{ subject to } y \in \operatorname{range}(U_k),\ b - AA^T y \perp \operatorname{range}(U_{k-1}). && (5.28) \end{aligned}$$

When A is row-rank-deficient, the (AA^T)-norm should be interpreted as a norm when restricted to range(A).

Algorithm 5 LNLQ

```
1: $\beta_1 u_1 = b,\ \ \alpha_1 v_1 = A^T u_1$    ▷ begin Golub-Kahan process
2: $\bar\varepsilon_1 = \alpha_1,\ \tau_1 = \beta_1/\alpha_1,\ \bar\zeta_1 = \tau_1/\bar\varepsilon_1$    ▷ begin LQ factorization
3: $w_1 = 0,\ \bar w_1 = u_1$
4: $y_1^L = 0,\ y_1^C = \bar\zeta_1 \bar w_1$
5: $x_1^L = 0,\ x_1^C = \tau_1 v_1$
6: for $k = 1, 2, \dots$ do
7:     $\beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k$    ▷ continue Golub-Kahan process
8:     $\alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k$
9:     $\varepsilon_k = (\bar\varepsilon_k^2 + \beta_{k+1}^2)^{\frac{1}{2}}$    ▷ continue LQ factorization
10:    $c_{k+1} = \bar\varepsilon_k/\varepsilon_k,\ s_{k+1} = \beta_{k+1}/\varepsilon_k$
11:    $\eta_{k+1} = \alpha_{k+1} s_{k+1},\ \bar\varepsilon_{k+1} = -\alpha_{k+1} c_{k+1}$
12:    $\zeta_k = c_{k+1}\bar\zeta_k,\ \bar\zeta_{k+1} = (\tau_{k+1} - \eta_{k+1}\zeta_k)/\bar\varepsilon_{k+1}$    ▷ prepare to update $y$
13:    $w_k = c_{k+1}\bar w_k + s_{k+1} u_{k+1},\ \bar w_{k+1} = s_{k+1}\bar w_k - c_{k+1} u_{k+1}$
14:    $y_{k+1}^L = y_k^L + \zeta_k w_k$    ▷ update $y$
15:    $y_{k+1}^C = y_{k+1}^L + \bar\zeta_{k+1}\bar w_{k+1}$
16:    $x_{k+1}^L = x_k^C + \eta_{k+1}\zeta_k v_{k+1}$    ▷ update $x$
17:    $\tau_{k+1} = -\beta_{k+1}\tau_k/\alpha_{k+1}$
18:    $x_{k+1}^C = x_k^C + \tau_{k+1} v_{k+1}$
19: end for
```

Proof. By definition, $\bar y_k^L$ solves (5.11). Hence there must exist $\bar t$ such that $\bar y_k^L = H_{k-1}\bar t$ and $H_{k-1}^T \bar y_k^L = \beta_1 e_1$. By definition of H_{k-1} and (2.11), we have $y_k^L = U_k \bar y_k^L = U_k B_{k-1} L_{k-1}^T \bar t = AV_{k-1} L_{k-1}^T \bar t = AA^T U_{k-1}\bar t$.

The above implies that y_k^L is primal feasible for (5.27). Dual feasibility requires that $U_{k-1}^T AA^T(y - y_\star) = 0$, which is equivalent to $U_{k-1}^T r_k^L = 0$ because $AA^T y_\star = b$. The expression (5.19) confirms dual feasibility.

With $y_k^L \in \operatorname{range}(A)$, we have $y_k^L = (A^\dagger)^T x_k^L$ and then (5.25) follows from (5.27).

Using (5.19), we see that y_k^L is primal feasible for (5.28). Dual feasibility requires that $y_k^L = p + AA^T U_{k-1} q$ and $U_k^T p = 0$ for certain vectors p and q, but those conditions are satisfied for $p := 0$ and $q := \bar t$. Since $y_k^L = (A^\dagger)^T x_k^L$, we obtain (5.26) from (5.28). ∎

Corollary 5.4 *For each k, $\|x_k^C - x_\star\| \leqslant \|x_k^L - x_\star\|$.*

Proof. By comparing (5.7) with (5.25), we see that $\|x_k^C - x_\star\| \leqslant \|x_k^L - x_\star\|$ because $\operatorname{range}(V_{k-1}) \subset \operatorname{range}(V_k)$. ∎

5.3 Complete algorithm

Algorithm 5 summarizes LNLQ. Note that if only the x part of the solution is desired, there is no need to initialize and update the vectors w_k, $\bar w_k$, y_k^L and y_k^C unless one wants to retrieve x as $A^T y$ at the end of the procedure. Similarly, if only the y part of the solution is desired, there is no need to initialize and update the vectors x_k^L and x_k^C. The update for x_{k+1}^C in line 18 of Algorithm 5 can be used even if the user wishes to dispense with updating x_k^L.

5.4 Error estimates

5.4.1 Upper bound on $\|y_\star - y_k^L\|$

By orthogonality, $\|y^\star - y_k^L\|^2 = \|y^\star\|^2 - \|y_k^L\|^2$, so as before we want to obtain upper bounds of $\|y_k^L\|^2 = b^T f(AA^T)b$ for f defined in (4.26). We accomplish this using Gauss-Radau quadrature as in Section 4.4. The fixed Gauss-Radau quadrature node is set to a prescribed $\sigma_{\text{est}} \in (0, \sigma_r)$. We follow Section 4.4.1 and modify L_k rather than T_k. Let

$$\widetilde{L}_k := \begin{bmatrix} L_{k-1} & 0 \\ \beta_k e_{k-1}^T & \omega_k \end{bmatrix}, \tag{5.29}$$

which differs from L_k in its (k,k)th element only, and

$$\widetilde{T}_k := \widetilde{L}_k \widetilde{L}_k^T = \begin{bmatrix} T_{k-1} & \bar{\beta}_{k-1} e_{k-1} \\ \bar{\beta}_{k-1} e_{k-1}^T & \beta_k^2 + \omega_k^2 \end{bmatrix}$$

(with $\bar{\beta}_{k-1}$ defined in (5.3)), which differs from T_k in its (k,k)th element only. The Poincaré separation theorem ensures that the singular values of L_k lie in (σ_r, σ_1). The Cauchy interlace theorem for singular values ensures that it is possible to select ω_k so that the smallest singular value of (5.29) is σ_{est}. The next result is a paraphrase of Theorem 2.3.

Theorem 5.5 *Let $f : [0, \infty) \to \mathbb{R}$ be such that $f^{(2j+1)}(\xi) < 0$ for all $\xi \in (\sigma_r^2, \sigma_1^2)$ and all $j \geqslant 0$. Fix $\sigma_{\text{est}} \in (0, \sigma_r)$. Let L_k be the bidiagonal generated after k steps of Algorithm 2, and $\omega_k > 0$ be chosen so that the smallest singular value of (5.29) is σ_{est}. Then,*

$$b^T f(AA^T) b \leqslant \beta_1^2 e_1^T f(\widetilde{L}_k \widetilde{L}_k^T) e_1.$$

The procedure to compute $\omega_k = \sqrt{\sigma_{\text{est}}^2 - \sigma_{\text{est}} \beta_k \theta_{2k-2}}$ is identical to that of Section 4.4.1, where θ_{2k-2} is an element of a related eigenvector. Application of Theorem 5.5 to $f(\xi) := \xi^{-2}$ with the convention that $f(0) := 0$ provides an upper bound on $\|y^\star\|^2$.

Corollary 5.6 *Fix $\sigma_{\text{est}} \in (0, \sigma_r)$. Let L_k be the bidiagonal generated after k steps of Algorithm 2, and $\omega_k > 0$ be chosen so that the smallest singular value of (5.29) is σ_{est}. Then*

$$\|y^\star\|^2 \leqslant \beta_1^2 e_1^T \big(\widetilde{L}_k \widetilde{L}_k^T\big)^{-2} e_1.$$

To evaluate the bound in Corollary 5.6, we modify the LQ factorization (5.12) to

$$\widetilde{L}_k^T = \begin{bmatrix} L_{k-1}^T & \beta_k e_{k-1} \\ 0 & \omega_k \end{bmatrix} = \begin{bmatrix} M_{k-1} & \\ \widetilde{\eta}_k e_{k-1}^T & \widetilde{\varepsilon}_k \end{bmatrix} \begin{bmatrix} Q_{k-1} & \\ & 1 \end{bmatrix} = \widetilde{M}_k Q_k,$$

where $\widetilde{\eta}_k = \omega_k s_k$ and $\widetilde{\varepsilon}_k = -\omega_k c_k$. Define $\widetilde{t}_k$ and $\widetilde{z}_k$ from

$$\widetilde{L}_k \widetilde{t}_k = \beta_1 e_1 \quad \text{and} \quad \widetilde{M}_k \widetilde{z}_k = \widetilde{t}_k. \tag{5.30}$$

The updated factorization and the definition of f yield

$$\|y^\star\|^2 \leqslant \beta_1^2 \|(\widetilde{L}_k \widetilde{M}_k Q_k)^{-1} e_1\|^2 = \beta_1^2 \|\widetilde{M}_k^{-1} \widetilde{L}_k^{-1} e_1\|^2 = \|\widetilde{M}_k^{-1} \widetilde{t}_k\|^2 = \|\widetilde{z}_k\|^2.$$

Comparing with the definition of t_k and z_k in Section 5.2.3 reveals that $\widetilde{t}_k = (t_{k-1}, \widetilde{\tau}_k)$ and $\widetilde{z}_k = (z_{k-1}, \widetilde{\zeta}_k)$, with $\widetilde{\tau}_k = -\beta_k \tau_{k-1}/\omega_k$ and $\widetilde{\zeta}_k = (\widetilde{\tau}_k - \widetilde{\eta}_k \zeta_{k-1})/\widetilde{\varepsilon}_k$. Combining with (5.18) yields the bound

$$\|y^\star - y_k^L\|^2 = \|y^\star\|^2 - \|z_{k-1}\|^2 \leqslant \|z_{k-1}\|^2 + \widetilde{\zeta}_k^2 - \|z_{k-1}\|^2 = \widetilde{\zeta}_k^2. \tag{5.31}$$

5.4.2 Upper bound on $\|y^\star - y_k^C\|$

Theorem 3.4 establishes that $\|y^\star - y_k^C\| \leqslant \|y^\star - y_k^L\|$, so that the bound from the previous section applies. With $\bar{\zeta}_k$ defined in Section 5.2.3, we can improve the bound as in (3.17) to

$$\|y^\star - y_k^C\|^2 \leqslant \widetilde{\zeta}_k^2 - \bar{\zeta}_k^2. \tag{5.32}$$

They provide further refinement of this bound by using the sliding window approach of Golub and Strakǒs (1994). For a chosen delay d, $O(d)$ scalars can be stored at each iteration, and for $O(d)$ additional work, we can compute $\theta_k^{(d)} \geqslant 0$ (see (3.18)) so that

$$\|y^\star - y_k^C\|^2 \leqslant \widetilde{\zeta}_k^2 - \bar{\zeta}_k^2 - 2\theta_k^{(d)}. \tag{5.33}$$

5.4.3 Upper bound on $\|x_\star - x_k^C\|$

Assume temporarily that A has full row rank. By orthogonality in (5.21), $\|x_\star - x_k^C\|^2 = \|x_\star\|^2 - \|x_k^C\|^2$. We may then use

$$\|x_\star\|^2 = \|A^T y^\star\|^2 = \|y^\star\|^2_{AA^T} = \|b\|^2_{(AA^T)^{-1}}.$$

Applying Theorem 5.5 to $f(\xi) := \xi^{-1}$ with $f(0) := 0$ provides an upper bound on $\|x_\star\|^2$ in the vein of Golub and Meurant (1994, Theorems 3.2 and 3.4).

Corollary 5.7 *Fix $\sigma_{\text{est}} \in (0, \sigma_r)$. Let L_k be the bidiagonal generated after k steps of Algorithm 2, and $\omega_k > 0$ be chosen so that the smallest singular value of (5.29) is σ_{est}. Then*

$$\|x_\star\|^2 \leqslant \beta_1^2 e_1^T \big(\widetilde{L}_k \widetilde{L}_k^T\big)^{-1} e_1.$$

We use (5.30) to evaluate the bound of Corollary 5.7 as

$$\beta_1^2 e_1^T \big(\widetilde{L}_k \widetilde{L}_k^T\big)^{-1} e_1 = \|\beta_1 \widetilde{L}_k^{-1} e_1\|^2 = \|\widetilde{t}_k\|^2,$$

which leads to the bound

$$\|x_\star - x_k^C\|^2 \leqslant \|\widetilde{t}_k\|^2 - \|t_k\|^2 = \widetilde{\tau}_k^2 - \tau_k^2. \tag{5.34}$$

This equals the bound of Arioli (2013), who derived it using the Cholesky factorization of T_k.

Note that Arioli (2013, Equation (4.4)) proposes the error bound

$$\|y^\star - y_k^C\| = \|L_n^{-1}(x_\star - x_k^C)\| \leqslant \sigma_{\min}(L_k)^{-1}\,\|x_\star - x_k^C\| \leqslant \sigma_r^{-1}\,\|x_\star - x_k^C\|. \tag{5.35}$$

It may be possible to improve on (5.35) by maintaining a running estimate of $\sigma_{\min}(L_k)$, such as the estimate $\min(\varepsilon_1, \dots, \varepsilon_{k-1}, \bar{\varepsilon}_k)$ discussed by Stewart (1999).

5.4.4 Upper bound on $\|x_\star - x_k^L\|$

Using $x_k^L = x_{k-1}^C + \eta_k \zeta_{k-1} v_k$, we have

$$\|x_\star - x_k^L\|^2 = \left\| V_n \left(t_n - \begin{bmatrix} t_{k-1} \\ \eta_k \zeta_{k-1} \\ 0 \end{bmatrix} \right) \right\|^2 = \|x_\star - x_k^C\|^2 + (\tau_k - \eta_k \zeta_{k-1})^2.$$

Thus, using the error bound in (5.34) we obtain

$$\|x_\star - x_k^L\|^2 \leqslant \tilde{\tau}_k^2 - \tau_k^2 + (\tau_k - \eta_k \zeta_{k-1})^2. \tag{5.36}$$

5.5 Regularization

The regularized least-norm problem is

$$\underset{x \in \mathbb{R}^n,\, s \in \mathbb{R}^m}{\text{minimize}} \ \tfrac{1}{2}(\|x\|^2 + \|s\|^2) \quad \text{subject to } Ax + \lambda s = b, \tag{5.37}$$

which is compatible for any $\lambda \neq 0$. Saunders (1995, Result 7) states that applying Algorithm 2 to $\hat{A} := \begin{bmatrix} A & \lambda I \end{bmatrix}$ with initial vector b preserves U_k. We find corresponding $\widehat{V}_k$ and lower bidiagonal $\hat{L}_k$ by comparing the identities

$$\begin{bmatrix} A^T \\ \lambda I \end{bmatrix} U_k = \begin{bmatrix} V_k & \\ & U_k \end{bmatrix} \begin{bmatrix} L_k^T \\ \lambda I \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} A^T \\ \lambda I \end{bmatrix} U_k = \widehat{V}_k \hat{L}_k^T, \tag{5.38}$$

the first of which results from (2.11) and the second from Algorithm 2 applied to $\hat{A}$. At iteration k, we apply reflections $\hat{Q}_k$ designed to zero out the λI block, resulting in

$$\begin{bmatrix} V_k & \\ & U_k \end{bmatrix} \begin{bmatrix} L_k^T \\ \lambda I \end{bmatrix} = \begin{bmatrix} V_k & \\ & U_k \end{bmatrix} \hat{Q}_k^T \hat{Q}_k \begin{bmatrix} L_k^T \\ \lambda I \end{bmatrix} = \begin{bmatrix} \widehat{V}_k & \hat{Y}_k \end{bmatrix} \begin{bmatrix} \hat{L}_k^T \\ 0 \end{bmatrix} = \widehat{V}_k \hat{L}_k^T.$$

Saunders (1995) uses $\hat{Q}_k$ to describe CRAIG with regularization under the name *extended CRAIG*. If we initialize $\lambda_1 := \lambda$, the first few reflections are illustrated as in Fig. 5.1, where shaded elements are those participating in the current reflection and grayed out elements have not yet been used. Two reflections per iteration are necessary, and the situation at iteration k may be described as

$$\begin{array}{c|ccc} & {\scriptstyle k} & {\scriptstyle 2k} & {\scriptstyle 2k+1} \\ {\scriptstyle k} & \alpha_k & \lambda_k & \\ {\scriptstyle k+1} & \beta_{k+1} & & \lambda \end{array} \begin{array}{cc} {\scriptstyle k} & {\scriptstyle 2k} \\ \hat{c}_k & \hat{s}_k \\ \hat{s}_k & -\hat{c}_k \end{array} \begin{array}{cc} {\scriptstyle 2k} & {\scriptstyle 2k+1} \\ \tilde{c}_k & \tilde{s}_k \\ \tilde{s}_k & -\tilde{c}_k \end{array} = \begin{array}{ccc} {\scriptstyle k} & {\scriptstyle 2k} & {\scriptstyle 2k+1} \\ \hat{\alpha}_k & 0 & \\ \hat{\beta}_{k+1} & \hat{\lambda}_{k+1} & \lambda \end{array} \begin{array}{cc} {\scriptstyle 2k} & {\scriptstyle 2k+1} \\ \tilde{c}_k & \tilde{s}_k \\ \tilde{s}_k & -\tilde{c}_k \end{array}$$

$$= \begin{array}{ccc} {\scriptstyle k} & {\scriptstyle 2k} & {\scriptstyle 2k+1} \\ \hat{\alpha}_k & 0 & \\ \hat{\beta}_{k+1} & 0 & \lambda_{k+1} \end{array}.$$

The first reflection is defined by $\hat{\alpha}_k := \sqrt{\alpha_k^2 + \lambda_k^2}$, $\hat{c}_k := \alpha_k / \hat{\alpha}_k$, $\hat{s}_k := \lambda_k / \hat{\alpha}_k$, and results in $\hat{\beta}_{k+1} = \hat{c}_k \beta_{k+1}$ and $\hat{\lambda}_{k+1} = \hat{s}_k \beta_{k+1}$. The second reflection defines $\lambda_{k+1} := \sqrt{\hat{\lambda}_{k+1}^2 + \lambda^2}$, $\tilde{c}_k :=$

$$\begin{bmatrix} \alpha_1 & & & & \lambda_1 & & & \\ \beta_2 & \alpha_2 & & & & \lambda & & \\ & \beta_3 & \alpha_3 & & & & \lambda & \\ & & \beta_4 & \alpha_4 & & & & \lambda \end{bmatrix} \rightarrow \begin{bmatrix} \hat\alpha_1 & & & & 0 & & & \\ \hat\beta_2 & \alpha_2 & & & \hat\lambda_2 & \lambda & & \\ & \beta_3 & \alpha_3 & & & & \lambda & \\ & & \beta_4 & \alpha_4 & & & & \lambda \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} \hat\alpha_1 & & & & 0 & & & \\ \hat\beta_2 & \alpha_2 & & & 0 & \lambda_2 & & \\ & \beta_3 & \alpha_3 & & & & \lambda & \\ & & \beta_4 & \alpha_4 & & & & \lambda \end{bmatrix} \rightarrow \begin{bmatrix} \hat\alpha_1 & & & & 0 & & & \\ \hat\beta_2 & \hat\alpha_2 & & & 0 & 0 & & \\ & \hat\beta_3 & \alpha_3 & & & \hat\lambda_3 & \lambda & \\ & & \beta_4 & \alpha_4 & & & & \lambda \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} \hat\alpha_1 & & & & 0 & & & \\ \hat\beta_2 & \hat\alpha_2 & & & 0 & 0 & & \\ & \hat\beta_3 & \alpha_3 & & & 0 & \lambda_3 & \\ & & \beta_4 & \alpha_4 & & & & \lambda \end{bmatrix} \rightarrow \begin{bmatrix} \hat\alpha_1 & & & & 0 & & & \\ \hat\beta_2 & \hat\alpha_2 & & & 0 & 0 & & \\ & \hat\beta_3 & \hat\alpha_3 & & & 0 & 0 & \\ & & \hat\beta_4 & \alpha_4 & & & \hat\lambda_4 & \lambda \end{bmatrix},$$

Figure 5.1: Illustration of a few steps of the factorization in the presence of regularization.

$\hat\lambda_{k+1}/\lambda_{k+1}$, $\tilde s_k := \lambda/\lambda_{k+1}$, and does not create a new nonzero. Only the first reflection contributes to the kth column of $\widehat V_k$:

$$\begin{array}{c} \begin{array}{cc} {\scriptstyle k} & {\scriptstyle 2k} \end{array} \\ \begin{bmatrix} v_k & 0 \\ 0 & u_k \end{bmatrix} \end{array} \begin{array}{c} \begin{array}{cc} {\scriptstyle k} & {\scriptstyle 2k} \end{array} \\ \begin{bmatrix} \hat c_k & \hat s_k \\ \hat s_k & -\hat c_k \end{bmatrix} \end{array} = \begin{array}{c} \begin{array}{cc} {\scriptstyle k} & {\scriptstyle 2k} \end{array} \\ \begin{bmatrix} \hat c_k v_k & \hat s_k v_k \\ \hat s_k u_k & -\hat c_k u_k \end{bmatrix} \end{array}. \tag{5.39}$$

Iteration k of LNLQ with regularization solves (5.11), but H_{k-1}^T is then the top $(k-1)\times k$ submatrix of

$$\begin{bmatrix} L_k & \lambda I \end{bmatrix} \begin{bmatrix} L_k^T \\ \lambda I \end{bmatrix} = L_k L_k^T + \lambda^2 I = T_k + \lambda^2 I.$$

In (5.12), we compute the LQ factorization of $\hat L_k^T$ instead of L_k^T, but the details are identical, as are the updates of y_k^L in (5.16) and y_k^C in (5.17). Because U_k is unchanged by regularization, the residual expressions (5.19) and (5.20) remain valid. Subsequently,

$$\begin{bmatrix} x_k^L \\ s_k^L \end{bmatrix} = \begin{bmatrix} A^T \\ \lambda I \end{bmatrix} U_k \bar y_k = \widehat V_k \hat L_k^T \bar y_k,$$

but we are only interested in the top half of x_k^L. Let the top $n \times k$ submatrix of $\widehat V_k$ be

$$\widehat W_k := \begin{bmatrix} \widehat w_1 & \cdots & \widehat w_k \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix} \widehat V_k = \begin{bmatrix} V_k & 0 \end{bmatrix} \hat Q_k^T.$$

We conclude from (5.39) that $\widehat w_j = \hat c_j v_j$ for $j = 1, \dots, k$. The update (5.22) remains valid with v_k replaced by $\widehat w_k$.

5.6 Preconditioning

As with other Golub-Kahan-based methods, convergence depends on the distribution of $\{\sigma_i(A)\}$. Therefore we consider an equivalent system $N^{-\frac{1}{2}}AA^TN^{-\frac{1}{2}}N^{\frac{1}{2}}y = N^{-\frac{1}{2}}b$, where $N^{-\frac{1}{2}}A$ has clustered singular values.

For the unregularized problem (NE-LN), to run preconditioned LNLQ efficiently we replace Algorithm 2 by the generalized Golub-Kahan process (Arioli, 2013, Algorithm 3.1). We seek a preconditioner $N \succ 0$ such that $N \approx AA^T$, and require no changes to the algorithm except in how we generate vectors u_k and v_k. This is equivalent to applying a block-diagonal preconditioner to the saddle-point system

$$\begin{bmatrix} I & \\ & N^{-1} \end{bmatrix} \begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ -y \end{bmatrix} = \begin{bmatrix} I & \\ & N^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ b \end{bmatrix}.$$

For a regularized system with $\lambda \neq 0$, we need to solve a 2×2 quasi-definite system

$$\begin{bmatrix} I & A^T \\ A & -\lambda^2 I \end{bmatrix} \begin{bmatrix} x \\ -y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}. \tag{5.40}$$

We cannot directly precondition with generalized Golub-Kahan as before, because properties analogous to (5.38) do not hold for $N \neq I$. Instead we must precondition the equivalent 3×3 block system

$$\begin{bmatrix} I & & \\ & I & \\ & & N^{-1} \end{bmatrix} \begin{bmatrix} I & & A^T \\ & I & \lambda I \\ A & \lambda I & \end{bmatrix} \begin{bmatrix} x \\ s \\ -y \end{bmatrix} = \begin{bmatrix} I & & \\ & I & \\ & & N^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ b \end{bmatrix},$$

where $N \approx AA^T + \lambda^2 I$ is a symmetric positive definite preconditioner. In effect, we must run preconditioned LNLQ directly on $\hat{A} = \begin{bmatrix} A & \lambda I \end{bmatrix}$.

5.7 Implementation and numerical experiments

We use the Matlab implementation of LNLQ[1], including the relevant error bounds. The exact solution for each experiment is computed using Matlab's backslash operator on the augmented system (NE-LN). Mentions of CRAIG below refer to transferring from the LNLQ point to the CRAIG point.

5.7.1 UFL problems

Matrix Meszaros/scagr7-2c from the UFL collection (Davis and Hu, 2011) has size 2447×3479. We set $b = \mathbb{1}/\sqrt{m}$. For LNLQ and CRAIG we record the error in x_k and y_k at each iteration using the exact solution, and the error bounds discussed above using $\sigma_{\text{est}} = (1 - 10^{-10})\,\sigma_{\min}(A)$, where $\sigma_{\min}(A)$ was provided from the UFL collection. The same σ_{est} is used to evaluate the bound (5.35). Fig. 5.2 records the results.

[1] `github.com/restrin/LinearSystemSolvers`

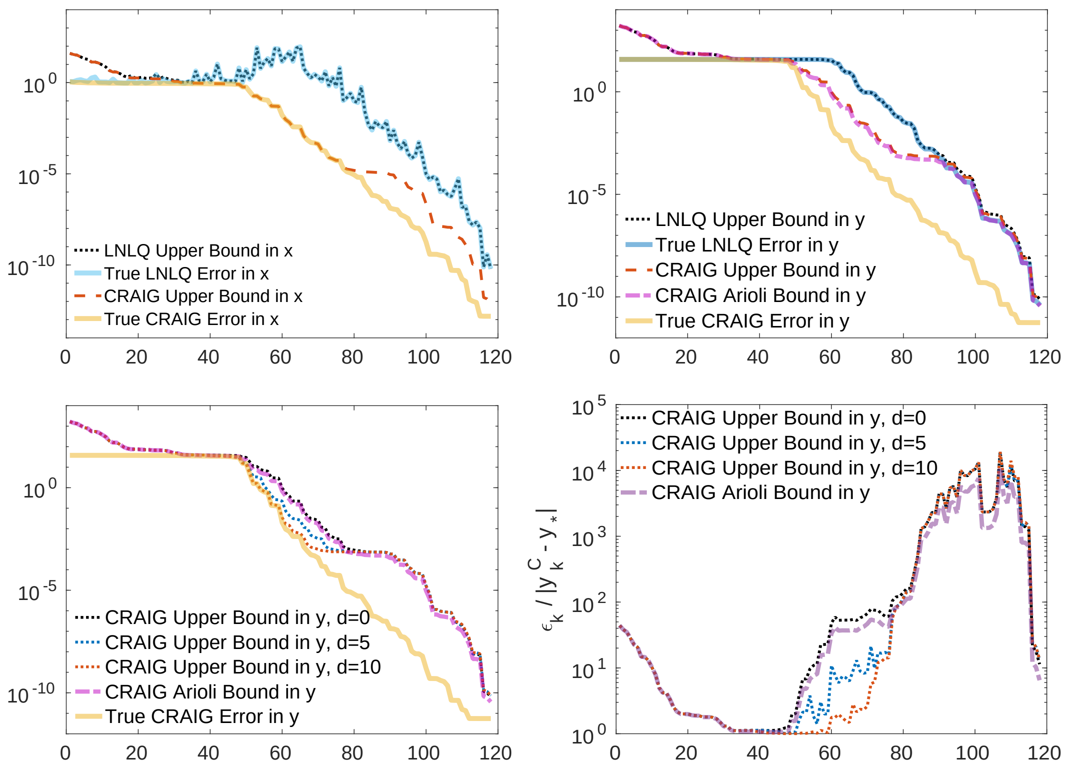


Figure 5.2: Error in x_k (top left) and y_k (top right) along the LNLQ and CRAIG iterations for Meszaros/scagr7-2c. The solid blue (yellow) line is the exact error for LNLQ (CRAIG), and the remaining lines show the various error bounds. The bottom left plot shows the improved bounds (5.33) and bounds from Arioli (2013) for the error in y_k for CRAIG with $d = 5$ and 10. The bottom right plot shows the same bounds divided by the true error.

We see that the LNLQ error bounds are tight, even though the error in x_k is not monotonic. In accordance with Proposition 5.1, the CRAIG error in x_k is lower than the LNLQ error. The same for the error in y_k. The CRAIG error in x_k is tight until the Gauss-Radau quadrature becomes inaccurate—a phenomenon also observed by Meurant and Tichý (2014); Meurant and Tichỳ (2018).

Regarding the CRAIG error in y_k, we see that the error bounds from (5.32) and (5.35) are similar, with (5.35) being slightly tighter. We observed that the simpler bound (5.35) nearly overlaps with the bound (5.32) on other problems. However, (5.33) provides the ability to tighten (5.32), and even small delays such as $d = 5$ or 10 can improve the bound significantly until the Gauss-Radau quadrature becomes inaccurate. Thus, the sliding window approach can be useful when an accurate estimate of $\sigma_{\min}(A)$ is available and early termination is relevant, for example when only a crude approximation of $x_\star$ and $y_\star$ is required.

In Fig. 5.3 we repeat the experiment with UFL problem LPnetlib/lp_kb2, which has size 43×68. Because LNLQ and CRAIG take more than 250 iterations, it is clear that global orthogonality is violated, yet the upper bounds remain faithful. Hence, it may be possible to derive these bounds by assuming only local orthogonality in the Golub-Kahan process. This is a direction for future research.

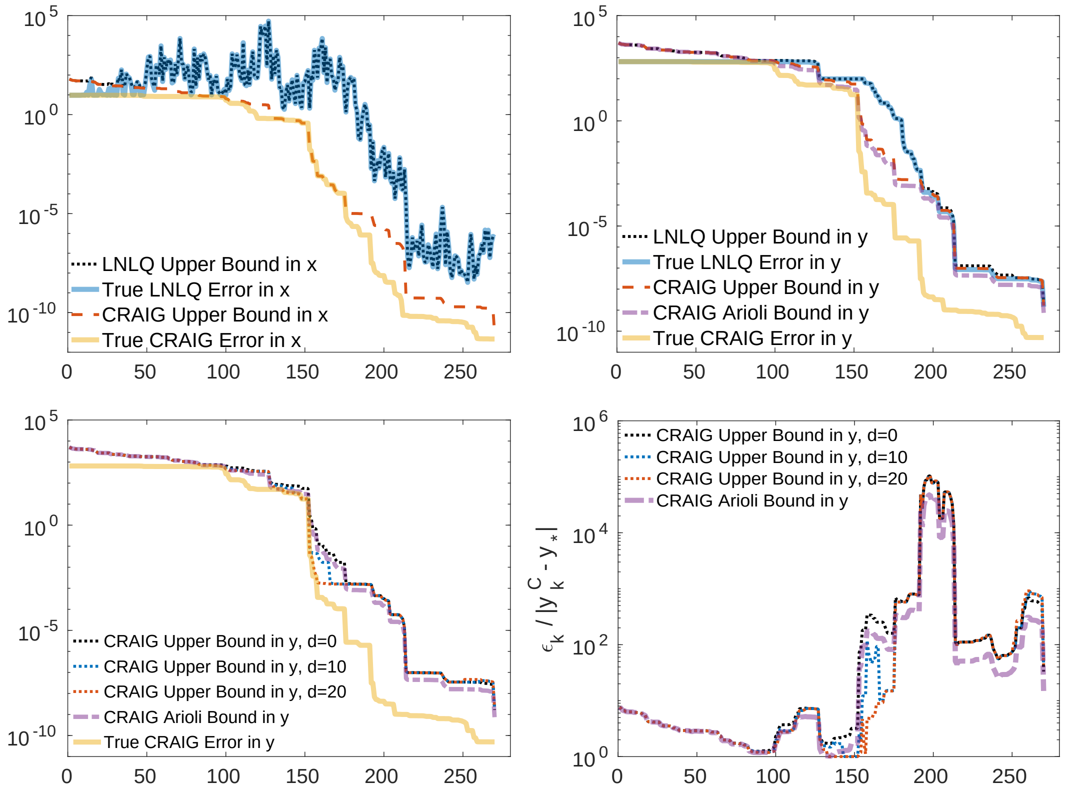


Figure 5.3: Error in x_k (top left) and y_k (top right) along the LNLQ and CRAIG iterations for LPnetlib/lp_kb2. The solid blue (yellow) line is the exact error for LNLQ (CRAIG), and the remaining lines show the various error bounds. The bottom left plot shows the improved bounds (5.33) and bounds from Arioli (2013) for the error in y_k for CRAIG with $d = 5$ and 10. The bottom right plot shows the same bounds divided by the true error.

5.7.2 Fletcher's penalty function

We now apply LNLQ to least-norm problems arising from using Fletcher's exact penalty function (see (Fletcher, 1973a) and Part II) to solve PDE-constrained control problems. We consider the problem

$$\begin{aligned} \underset{u,\,z}{\text{minimize}} \quad & \tfrac{1}{2}\int_\Omega \|u - u_d\|^2\,dx + \tfrac{1}{2}\alpha\int_\Omega z^2\,dx \\ \text{subject to} \quad & \nabla\cdot(z\nabla u) = -\sin(\omega x_1)\sin(\omega x_2) \quad \text{in } \Omega, \\ & u = 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{5.41}$$

where $\omega = \pi - \frac{1}{8}$, $\Omega = [-1,1]^2$, and $\alpha \geqslant 0$ is a small regularization parameter. Here, u might represent the temperature distribution on a square metal plate, u_d is the observed temperature, and we aim to determine the diffusion coefficients z so that u matches the observations in a least-squares sense. We discretize (5.41) using finite elements with triangular

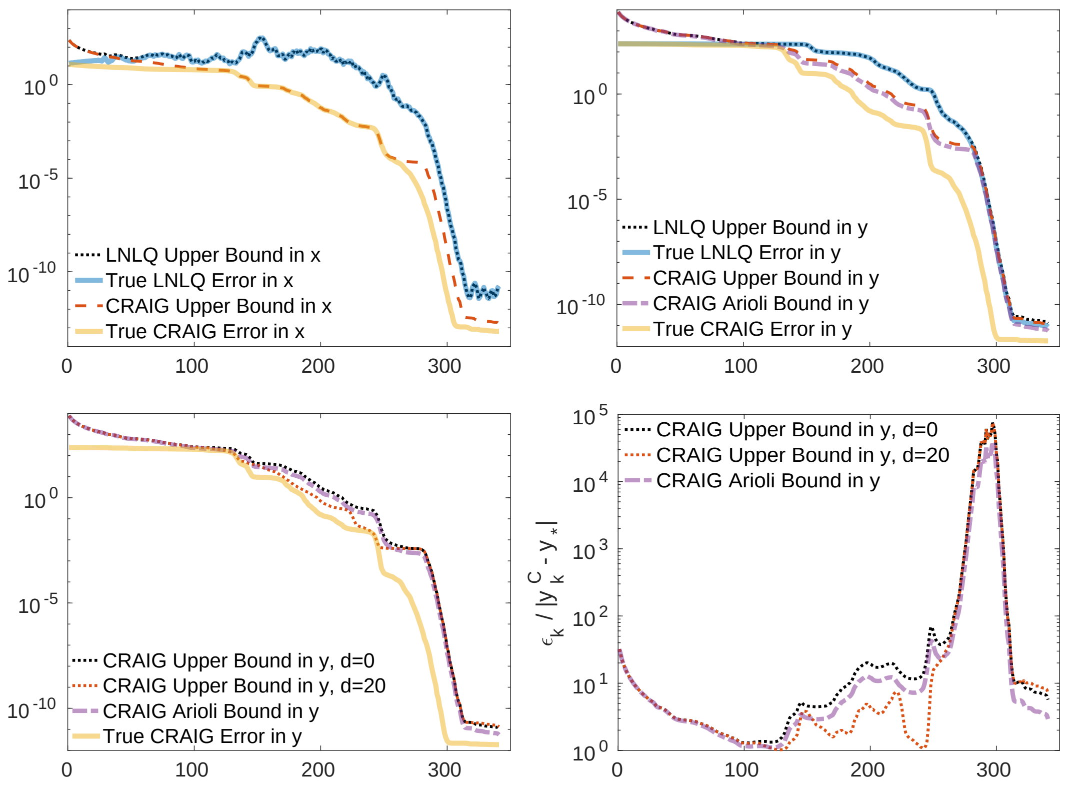


Figure 5.4: Error in x_k (top left) and y_k (top right) along the LNLQ and CRAIG iterations. The solid blue (yellow) line is the exact error for LNLQ (CRAIG), and the remaining lines show the various error bounds. The bottom left plot shows the improved bounds (5.33) and bounds from Arioli (2013) for the error in y_k for CRAIG with $d = 20$. The bottom right plot shows the same bounds divided by the true error.

cells, and obtain the equality-constrained problem

$$\underset{\bar{u}}{\text{minimize}}\ f(\bar{u}) \ \text{ subject to } \ c(\bar{u}) = 0.$$

Let p be the number of cells along one dimension, so that $u \in \mathbb{R}^{p^2}$ and $z \in \mathbb{R}^{(p+2)^2}$ are the discretizations of u and z, $\bar{u} := (u, z)$, and $c(\bar{u}) \in \mathbb{R}^{p^2}$. We use $p = 31$ in the experiments below. Let $A(\bar{u}) := \begin{bmatrix} A_u & A_z \end{bmatrix}$ be the Jacobian of $c(\bar{u})$.

For a given penalty parameter $\sigma > 0$, Fletcher's exact penalty approach is to

$$\begin{aligned} &\underset{\bar{u}}{\text{minimize}}\ \phi_\sigma(\bar{u}) := f(\bar{u}) - c(\bar{u})^T y_\sigma(\bar{u}) \\ &\quad \text{where } \ y_\sigma(\bar{u}) \in \operatorname{argmin} y \tfrac{1}{2} \left\| \nabla f(\bar{u}) - A(\bar{u})^T y \right\|^2 + \sigma c(\bar{u})^T y. \end{aligned}$$

In order to evaluate $\phi_\sigma(\bar{u})$ and $\nabla \phi_\sigma(\bar{u})$, we must solve systems of the form (NE-LN). For these experiments, we use $b = -c(\bar{u})$ and $A = A(\bar{u})$. Note that by controlling the error in the solution of (NE-LN), we control the inexactness in the computation of the penalty function value and gradient. In our experiments, we evaluate b and A at $\bar{u} = \mathbb{1}$. We first apply LNLQ

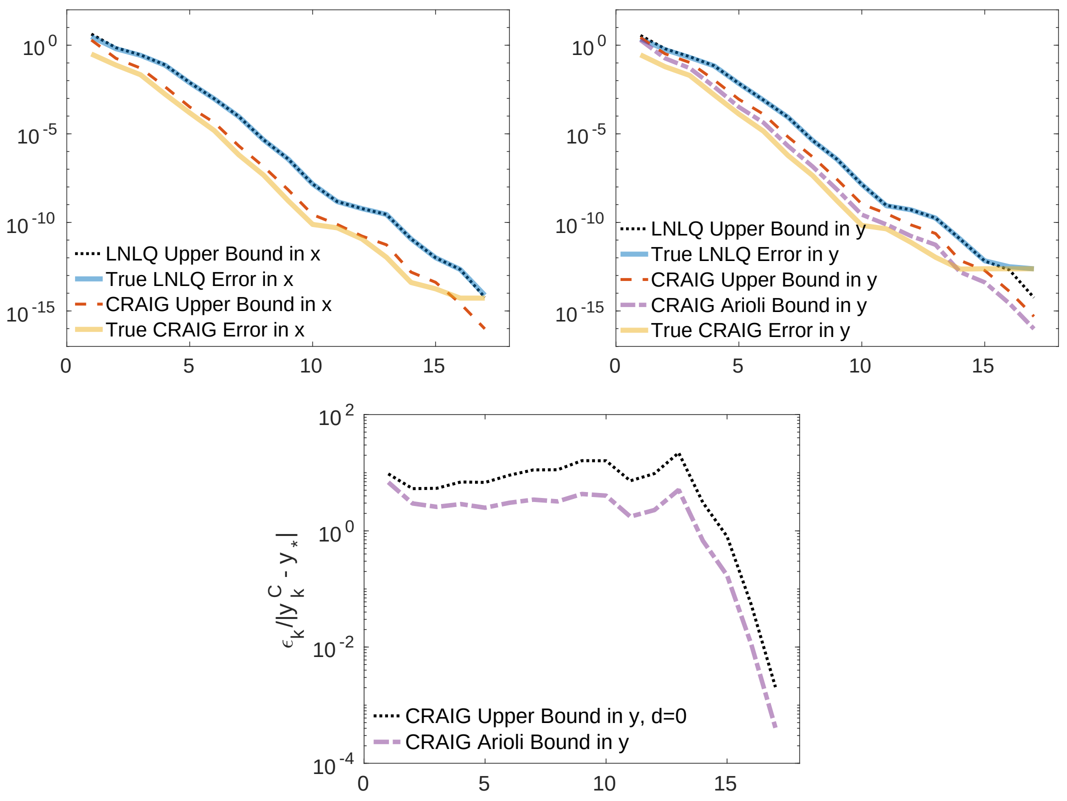


Figure 5.5: Error in x_k (left) and y_k (right) along the LNLQ and CRAIG iterations. The solid blue (yellow) line is the exact error for LNLQ (CRAIG), and the remaining lines show the various error bounds. The bottom plot shows the same bounds for CRAIG for the error in y_k, but divided by the true error.

and CRAIG without preconditioning. The results are summarized in Fig. 5.4.

We observe trends like those in the previous section. The LNLQ bounds are quite accurate because of our good estimate of the smallest singular value, even though the LNLQ error in x_k is not monotonic. The CRAIG error bound for x_k is tight until the Gauss-Radau quadrature becomes inaccurate, which results in a looser bound. The latter impacts the CRAIG error bound for y in the form of the plateau after iteration 250. The error bound (5.35) is slightly tighter than (5.32), while if we use (5.33) with $d = 20$, we achieve a tighter bound until the plateau occurs.

We now use the preconditioner $N = A_u A_u^T$, which corresponds to two solves of Poisson's equation with fixed diffusion coefficients. Because $\sigma_{\min}((A_u A_u)^{-1} A A^T) = \sigma_{\min}(I + (A_u A_u^T)^{-1} A_z A_z^T) \geqslant 1$, we choose $\sigma_{\text{est}} = 1$. Recall that the y-error is now measured in the N-energy norm. The results appear in Fig. 5.5.

We see that the preconditioner is effective, and that $\sigma_{\text{est}} = 1$ is an accurate approximation as the LNLQ error bounds are extremely tight. The CRAIG error bounds are tight as well, although the error bounds" for y_k go below the true error in the last few iterations, which is expected and observed in Chapter 3.

Chapter 6

Extension to SQD systems

We can generalize LSLQ and LNLQ to the solution of symmetric quasi-definite systems (Vanderbei, 1995) of the form

$$\begin{bmatrix} M & A^T \\ A & -N \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}, \tag{6.1}$$

where $M = M^T$ and $N = N^T$ are positive definite and linear systems with these matrices can be efficiently solved. This represents the optimality conditions for problems

$$\underset{x}{\text{minimize}} \ \tfrac{1}{2}\|Ax - b\|^2_{N^{-1}} + \tfrac{1}{2}\|x\|^2_M, \tag{6.2}$$

$$\underset{x,\, y}{\text{minimize}} \ \tfrac{1}{2}\|x\|^2_M + \tfrac{1}{2}\|y\|^2_N \quad \text{subject to } A^T x - Ny = b. \tag{6.3}$$

The only change required are to replace Algorithm 2 by the generalized Golub-Kahan process (Orban and Arioli, 2017, Algorithm 4.2). The latter requires one system solve with M and one system solve with N per iteration. Thus the generalized LSLQ applies to (6.2), while generalized LNLQ applies to (6.3). They are equivalent to applying SYMMLQ to the respective systems

$$(A^T N^{-1} A + M)x = A^T N^{-1} b, \tag{6.4}$$

$$(AM^{-1}A^T + N)y = -b, \qquad Mx = -A^T y. \tag{6.5}$$

In lieu of (2.11), the generalized Golub-Kahan process can be summarized as

$$AV_k = MU_{k+1}B_k, \tag{6.6a}$$

$$A^T U_{k+1} = NV_k B_k^T + \alpha_{k+1} N v_{k+1} e_{k+1}^T = NV_{k+1} L_{k+1}^T, \tag{6.6b}$$

where now $U_k^T M U_k = I$ and $V_k^T N V_k = I$ in exact arithmetic. Pasting (6.6) together yields

$$\begin{bmatrix} M & A^T \\ A & -N \end{bmatrix} \begin{bmatrix} V_k & \\ & U_k \end{bmatrix} = \begin{bmatrix} M & \\ & N \end{bmatrix} \begin{bmatrix} V_k & \\ & U_k \end{bmatrix} \begin{bmatrix} I & L_k^T \\ L_k & -I \end{bmatrix} + \begin{bmatrix} 0 \\ \beta_{k+1} N u_{k+1} \end{bmatrix} e_{2k}^T,$$

$$\begin{bmatrix} M & A^T \\ A & -N \end{bmatrix} \begin{bmatrix} V_k & \\ & U_{k+1} \end{bmatrix} = \begin{bmatrix} M & \\ & N \end{bmatrix} \begin{bmatrix} V_k & \\ & U_{k+1} \end{bmatrix} \begin{bmatrix} I & B_k^T \\ B_k & -I \end{bmatrix} + \begin{bmatrix} \alpha_{k+1} M v_{k+1} \\ 0 \end{bmatrix} e_{2k+1}^T.$$

These relations correspond to a Lanczos process applied to (6.1) with preconditioner blkdiag(M, N). The small symmetric quasi-definite matrix on the right-hand side of the previous identities is a symmetric permutation of the Lanczos tridiagonal, which is found by restoring the order in which the Lanczos vectors $(v_k,\, 0)$ and $(0,\, u_k)$ are generated:

$$T_{2k+1} = \begin{bmatrix} 1 & \alpha_1 & & & & \\ \alpha_1 & -1 & \beta_2 & & & \\ & \beta_2 & 1 & \ddots & & \\ & & \ddots & \ddots & \alpha_k & \\ & & & \alpha_k & -1 & \beta_{k+1} \\ & & & & \beta_{k+1} & 1 \end{bmatrix} = \begin{bmatrix} T_{2k} & \beta_{k+1} e_{2k} \\ \beta_{k+1} e_{2k}^T & 1 \end{bmatrix}.$$

Saunders (1995) and Orban and Arioli (2017) show that the CG iterates are well-defined for (6.1) even though it is indefinite. In a similar vein, Orban and Arioli (2017) establish that applying MINRES to (6.1) with the block-diagonal preconditioner produces alternating preconditioned LSMR and LSQR iterations, where LSMR is applied to (6.4) and LSQR is applied to (6.5).

It turns out that SYMMLQ applied directly to (6.1) with this preconditioner satisfies the following property: even iterations are CG iterations, while odd iterations take a zero step and make no progress. Thus every other iteration is wasted. Therefore, the generalized versions of iterative methods (Orban and Arioli, 2017), such as generalized LSLQ or LNLQ, should be used instead. The property is formalized in the following result.

Theorem 6.1 *Let x_k^{LQ} and x_k^{CG} be the iterates generated at iteration k of SYMMLQ and CG applied to (6.1) preconditioned by* $\operatorname{blkdiag}(M,\, N)$*, and x_k^C be the iterate defined in (5.5) (using the generalized Golub-Kahan process). Then for $k \geqslant 1$, $x_{2k-1}^{LQ} = x_{2k}^{LQ} = x_{2k}^{CG} = x_k^C$.*

Proof. We use the notation from Section 3.1 to describe the Lanczos process and how to construct the CG and SYMMLQ iterates. By (6.6), $\underline{T}_k$ and the L factor of the LQ factorization of $\underline{T}_{k-1}^T$ have the form

$$\underline{T}_k = \begin{bmatrix} 1 & t_2 & & & \\ t_2 & -1 & t_3 & & \\ & t_3 & 1 & \ddots & \\ & & \ddots & \ddots & t_k \\ & & & t_k & (-1)^{k-1} \\ & & & & t_{k+1} \end{bmatrix} \qquad L_k = \begin{bmatrix} \gamma_1 & & & \\ \delta_2 & \gamma_2 & & \\ \varepsilon_3 & \delta_3 & \gamma_3 & \\ & \ddots & \ddots & \ddots & \\ & & \varepsilon_{k-1} & \delta_{k-1} & \gamma_{k-1} \end{bmatrix}$$

where each t_i is a scalar. For $k \geqslant 2$, the LQ factorization is accomplished using reflections defined by

$$\begin{bmatrix} \bar{\gamma}_{k-1} & t_k \\ \bar{\delta}_k & (-1)^{k-1} \\ 0 & t_{k+1} \end{bmatrix} \begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix} = \begin{bmatrix} \gamma_{k-1} & 0 \\ \delta_k & \bar{\gamma}_k \\ \varepsilon_{k+1} & \bar{\delta}_{k+1} \end{bmatrix},$$

with $\bar{\gamma}_1 = 1$, $\bar{\delta}_2 = t_2$, $c_k = \frac{\bar{\gamma}_{k-1}}{\gamma_{k-1}}$, and $s_k = \frac{t_k}{\gamma_{k-1}}$.

We show that $\delta_j = 0$ for all j by showing that $\bar{\gamma}_k = \frac{(-1)^k}{c_k}$ for $k \geqslant 2$, because in that case

$$\begin{aligned} \delta_k = \bar{\delta}_k c_k - (-1)^{k-1} s_k &= (t_k c_{k-1}) \frac{\bar{\gamma}_{k-1}}{\gamma_{k-1}} - (-1)^{k-1} \frac{t_k}{\gamma_{k-1}} \\ &= \frac{t_k}{\gamma_{k-1}} \left((-1)^{k-1} - (-1)^{k-1} \right) = 0. \end{aligned}$$

For $k = 2$ we have $\gamma_2^2 = 1 + t_2^2$ and $c_2 = \frac{1}{\gamma_2}$, so that $\bar{\gamma}_2 = \bar{\delta}_2 s_2 + c_2 = \frac{t_2^2}{\gamma_2} + \frac{1}{\gamma_2} = \gamma_2 = \frac{1}{c_2}$.

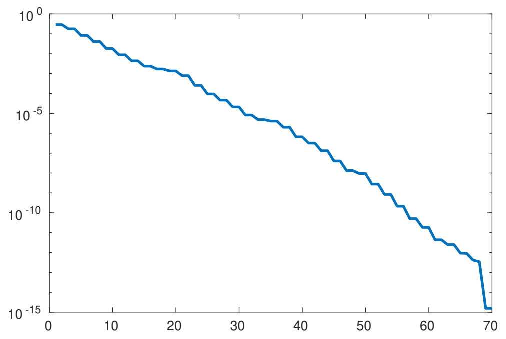


Figure 6.1: Error $\|x_k - x_\star\|$ generated by SYMMLQ applied to (6.1). Note that every odd iteration makes no progress, resulting in a convergence plot resembling a step function.

Proceeding by induction, assume $c_{k-1} = \frac{(-1)^{k-1}}{\bar{\gamma}_{k-1}}$. Then

$$\begin{aligned}\bar{\gamma}_k = \bar{\delta}_k s_k - (-1)^{k-1} c_k &= \tfrac{1}{c_k}\left(-t_k c_{k-1} s_k c_k - (-1)^{k-1} c_k^2\right) \\ &= -\tfrac{1}{c_k}\left((-1)^{k-1} \tfrac{t_k}{\bar{\gamma}_{k-1}} s_k c_k + (-1)^{k-1} c_k^2\right) \\ &= \tfrac{(-1)^k}{c_k}\left(\frac{s_k}{c_k} s_k c_k + c_k^2\right) = \tfrac{(-1)^k}{c_k}.\end{aligned}$$

For all k, since $\delta_k = 0$ and $x_k^{LQ} = W_{k-1} z_{k-1}$ with W_{k-1} having orthonormal columns, and since $(z_{k-1})_j = \zeta_j$ is defined by $L_{k-1} z_{k-1} = \|b\| e_1$, we have $\zeta_k = 0$ for k even. Therefore $x_{2k}^{LQ} = x_{2k-1}^{LQ}$. Further, since $\zeta_k = c_k \bar{\zeta}_k$ and $x_k^{CG} = x_k^{LQ} + \bar{\zeta}_k \bar{w}_k$ for some $\bar{w}_k \perp W_k$, we have $\zeta_{2k} = 0$ and $x_{2k}^{CG} = x_{2k}^{LQ}$. The identity $x_{2k}^{CG} = x_k^{C}$ follows from (Saunders, 1995, Result 11).□

We illustrate Theorem 6.1 using a small numerical example. We randomly generate A and b with $m = 50$, $n = 30$, $M = I$, and $N = I$ and run SYMMLQ directly on (6.1). We compute $x_\star$ via Matlab's backslash operator, and compute $\|x_k - x_\star\|$ at each iteration to produce Fig. 6.1. The resulting convergence plot resembles a staircase because every odd iteration produces a zero step.

Chapter 7

Contributions and future directions

7.1 Contributions

The main contributions of Part I are threefold: developing an error bounding procedure in the Euclidean norm for SYMMLQ and CG; introducing the methods LSLQ and LNLQ for least-squares and least-norm problems respectively; and furthering our understanding of Krylov subspace methods, particularly methods based on SYMMLQ and CG. These results are summarized below.

Euclidean-norm error bounds

We developed cheap estimates for the error in SYMMLQ and CG iterates, and explored the relationship between those errors. The approach to computing these error bounds is the basis for computing error bounds for LSLQ, LNLQ, LSQR, and CRAIG iterates. These estimates are based on Gauss-Radau quadrature, and require the availability of an underestimate λ_{est} of the smallest singular value of A. The main results for SYMMLQ and CG are in (3.10), (3.12), and (3.17)–(3.18).

When A is positive semidefinite with rank r, our error estimates are upper bounds up to convergence (under exact arithmetic). For CG, the estimate can be made tighter when $\lambda_{\text{est}} \approx \lambda_r$ by utilizing a delay d as described in (3.18), for an additional $O(d)$ flops and storage. When A is indefinite, the SYMMLQ estimate is not guaranteed to be an upper bound, but often tracks the error closely after an initial lag.

LSLQ and LNLQ

We introduced LSLQ, an iterative method for the least-squares and least-norm problems (LS) and (LN) with the attractive property that it ensures monotonic reduction in the Euclidean error $\|x^\star - x_k\|$. It completes the triad of solvers LSQR, LSMR, LSLQ for problem (LS) based on the Golub and Kahan (1965) process. LNLQ fills a gap in the family of iterative methods for (LN) based on the Golub and Kahan (1965) process, which ensures monotonic reduction in the Euclidean error $\|y^\star - y_k\|$. Although LSLQ and LNLQ are numerically equivalent to SYMMLQ on the appropriate normal equations, they are more numerically more reliable when A is ill-conditioned.

Further, we developed cheaply computable lower and upper bounds on the error for LSLQ, and therefore for LSQR, using the intimate relationship between the methods and the techniques of Chapter 3. Such an upper bound was not previously available for LSQR. Similar upper bounds were developed for LNLQ and CRAIG.

Relationship between Krylov subspace methods

We discovered new connections between existing Krylov subspace methods and the new methods LSLQ and LNLQ; for example, the equivalence between MINRES and LSQR for (LN)

Table 7.1: Comparison of CG-type and LQ-type method properties for problems (L), (LN), and (LN) respectively. In the first table, italicized results hold for indefinite systems as well.

	CG	SYMMLQ
$\lVert x_k \rVert$	$\nearrow$ (S, 1983, Theorem 2.1)	$\nearrow$ *(PS, 1975)*, $\leqslant$ CG (Theorem 3.4)
$\lVert x^\star - x_k \rVert$	$\searrow$ (HS, 1952, Theorem 6:3)	$\searrow$ *(PS, 1975)*, $\geqslant$ CG (Theorem 3.4)
$\lVert x^\star - x_k \rVert_A$	$\searrow$ (HS, 1952, Theorem 4:3)	not-monotonic
$\lVert r_k \rVert$	not-monotonic	not-monotonic
$\lVert r_k \rVert / \lVert x_k \rVert$	not-monotonic	not-monotonic

S (Steihaug, 1983), HS (Hestenes and Stiefel, 1952), PS (Paige and Saunders, 1975)

	LSQR	LSLQ
$\lVert x_k \rVert$	$\nearrow$ (F, 2011, Theorem 3.3.1)	$\nearrow$ (PS, 1975), $\leqslant$ LSQR (Proposition 4.1)
$\lVert x^\star - x_k \rVert$	$\searrow$ (F, 2011, Theorem 3.3.2)	$\searrow$ (PS, 1975), $\geqslant$ LSQR (Proposition 4.1)
$\lVert r_\star - r_k \rVert$	$\searrow$ (F, 2011, Theorem 3.3.3)	not-monotonic
$\lVert r_k \rVert$	$\searrow$	not-monotonic
$\lVert A^T r_k \rVert$	not-monotonic	not-monotonic

F (Fong, 2011), PS (Paige and Saunders, 1975)

	CRAIG	LNLQ
$\lVert x_k \rVert$	$\nearrow$ (5.8) and (P, 1974)	not-monotonic
$\lVert x_\star - x_k \rVert$	$\searrow$ (5.7) and (P, 1974)	not-monotonic, $\geqslant$ CRAIG (Corollary 5.4)
$\lVert y_k \rVert$	$\nearrow$ (5.18) and (HS, 1952)	$\nearrow$ (5.18) and (PS, 1975), $\leqslant$ CRAIG (Theorem 3.4)
$\lVert y_\star - y_k \rVert$	$\searrow$ (5.18) and (HS, 1952)	$\searrow$ (5.18) and (PS, 1975), $\geqslant$ CRAIG (Theorem 3.4)
$\lVert r_k \rVert$	not-monotonic	not-monotonic

HS (Hestenes and Stiefel, 1952), P (Paige, 1974), PS (Paige and Saunders, 1975)
$\nearrow$ monotonically increasing $\searrow$ monotonically decreasing

(Proposition 2.2). Many important properties about Krylov subspace methods for least-squares and least-norm problems can be understood through their equivalences with CG, MINRES, and SYMMLQ. Fong (2011, Tables 7.1, 7.2) summarize the monotonicity of various quantities related to the CG and MINRES, and LSQR and LSMR iterations. Table 7.1 is similar but focuses on the CG-type and LQ-type methods presented in Chapters 3 to 5.

7.2 Future directions

Some practical matters related to the error bounding procedure are yet to be addressed, and are left as future research directions.

Finite-precision error analysis

The error bounding methods presented in this section have assumed exact arithmetic, and in particular, global orthogonality of the vectors generated by the Lanczos and Golub and Kahan processes. These assumptions are clearly violated in practice, but we observed that the error bounds did indeed remain bounds until convergence (see, e.g., Appendix B). It is therefore likely that we can derive our error bounding procedure assuming only local orthogonality of the basis vectors (to machine precision). To confirm, finite-precision analysis similar to that of Strakoš and Tichý (2002) for CG error estimation in the A-norm is required to ensure that these techniques are safe in practice.

Adaptive eigenvalue estimates

The error bounding procedure would be more practical if the dependence on the eigenvalue underestimate of the smallest nonzero eigenvalue of A (or singular value for (LS) and (LN)) is reduced. An underestimate may be readily available in some application, but this is not true in general, and having the ability to refine the eigenvalue estimate throughout the iterative method may improve the tightness of the error bounds. It may be possible to employ techniques similar to Meurant and Tichỳ (2018), who use the smallest Ritz value from the Lanczos process (obtained as part of the iterative method) as the eigenvalue estimate; the main challenge is to avoid recomputation of implicit factorizations performed during the error bounding procedure.

Part II

A Smooth Exact Penalty Method for Nonlinearly Constrained Optimization

Chapter 8

Introduction

We consider a penalty-function approach to solve constrained nonlinear optimization problems

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) & \\ \text{subject to} & c(x) = 0 & : \ y \\ & \ell \leqslant x \leqslant u & : \ z, \end{array} \tag{NP}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are smooth functions ($m \leqslant n$), and the n-vectors ℓ and u provide lower and upper bounds (possibly infinite) on x. Additionally, $y \in \mathbb{R}^m$ is the dual variable for the equality constraints, and z is the dual variable for the bound constraints. A smooth exact penalty function ϕ_σ is used to eliminate the constraints $c(x) = 0$. The penalty function is the Lagrangian $L(x, y) = f(x) - c(x)^T y$ with the vector $y = y_\sigma(x)$ treated as a function of x depending on a parameter $\sigma > 0$. The penalty function depends only on the primal variables x, so that we instead solve the problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \phi_\sigma(x) \quad \text{subject to} \quad \ell \leqslant x \leqslant u : \ z. \tag{PP}$$

We purposely set z to be the Lagrange multiplier for the bound constraints for both (NP) and (PP) because, as we show, they equal each other at a solution. If ℓ and u are both infinite, then (PP) is unconstrained.

The penalty function for equality-constrained problems was proposed by Fletcher (1970). A long-held view is that Fletcher's penalty function is not practical because it is costly to compute (Bertsekas, 1975; Conn et al., 2000; Nocedal and Wright, 2006). In particular, Nocedal and Wright (2006, p.436) warn that "although this merit function has some interesting theoretical properties, it has practical limitations. . . ". Our aim is to challenge that notion by demonstrating that the computational kernels are no more expensive than other widely accepted methods for nonlinear optimization, such as sequential quadratic programming. We further derive a smooth extension of the penalty function to handle bound constraints.

The penalty function is *exact* because local minimizers of (NP) are minimizers of (PP) for all values of σ larger than a finite threshold $\sigma^\star$. The main computational kernel for evaluating the penalty function and its derivatives is the solution of certain structured linear systems. If the system matrix is available explicitly, we show how to factorize it once and re-use the factors to evaluate the penalty function and its derivatives. We also adapt the penalty function for *factorization-free* optimization by solving the linear system iteratively. This makes the penalty function particularly promising for certain problem classes, such as PDE-constrained optimization problems when good preconditioners exist.

Part II is based on Estrin, Friedlander, Orban, and Saunders (2019a,b).

8.1 The proposed penalty function

For (NP), we propose the penalty function

$$\phi_\sigma(x) := f(x) - c(x)^T y_\sigma(x), \tag{8.1}$$

where $y_\sigma(x)$ are Lagrange multiplier estimates defined with other items as

$$y_\sigma(x) := \operatorname{argmin}_y \tfrac{1}{2}\|A(x)y - g(x)\|^2_{Q(x)} + \sigma c(x)^T y, \qquad g(x) := \nabla f(x), \tag{8.2}$$

$$A(x) := \nabla c(x) = \begin{bmatrix} g_1(x) & \cdots & g_m(x) \end{bmatrix}, \qquad g_i(x) := \nabla c_i(x), \tag{8.3}$$

$$Y_\sigma(x) := \nabla y_\sigma(x). \tag{8.4}$$

Note that A and Y_σ are n-by-m matrices. The n-by-n diagonal matrix $Q(x)$ has entries $Q_{i,i}(x) := q_i(x_i)$, where

$$q_i(x_i) := \begin{cases} 1 & \text{if } \ell_i = -\infty \text{ and } u_i = \infty, \\ x_i - \ell_i - \frac{1}{2\omega_i}\left(2x_i + u_i - \ell_i - \frac{\omega_i}{2}\right)^2 & \text{if } |u_i + \ell_i - 2x_i| \leqslant \frac{\omega_i}{2}, \\ \min\{x_i - \ell_i, u_i - x_i\} & \text{otherwise.} \end{cases} \tag{8.5}$$

The diagonal of $Q(x)$ is a smooth approximation of $\min\{x-\ell, u-x\}$, where $\omega \in \mathbb{R}^n$ controls the smoothness of the approximation (we use $\omega_i = \min\{1, \frac{1}{2}(u_i - \ell_i)\}$). Note that $Q(x)$ is nonnegative on $[\ell, u]$. When all of ℓ and u are infinite, $Q(x) = I$ and we recover the penalty function proposed by Fletcher (1970). We discuss $Q(x)$ in more detail in Chapter 10.

We assume that (NP) satisfies some variation of the following assumptions:

(A1) f and c are twice continuously differentiable and have Lipschitz second-derivatives (A1a), or are three-times continuously differentiable (A1b).

(A2) The linear independence constraint qualification (LICQ) is satisfied for stationary points (A2a), or additionally for all $\ell < x < u$ (A2b). LICQ is satisfied at x if

$$\{\nabla c_i(x), e_j \mid x_j \in \{\ell_j, u_j\},\ i \in [m],\ j \in [n]\}$$

is linearly independent, where e_j is the jth column of the identity matrix.

(A3) Stationary points satisfy strict complementarity. If $(x^\star, y^\star, z^\star)$ is a primal-dual solution, exactly one of $z_j^\star$ and $\min\{x_j^\star - \ell_j, u_j - x_j^\star\}$ is zero $\forall j \in [n]$.

(A4) The problem is feasible. That is, there exists x such that $\ell \leqslant x \leqslant u$ and $c(x) = 0$, with $\ell_j < u_j$ for all $j \in [n]$. We further assume that fixed variables have been eliminated from the problem.

Assumption (A1b) ensures that ϕ_σ has two continuous derivatives and is typical for smooth exact penalty functions (Bertsekas, 1982, Proposition 4.16). However, at most two derivatives of f and c are required to implement this penalty function in practice. We typically assume (A1b) to simplify the discussion, but this assumption can often be weakened to (A1a). Assumption (A2b) guarantees that $Y_\sigma(x)$ and $y_\sigma(x)$ are uniquely defined; (A3) provides additional regularity to ensure that the threshold penalty parameter $\sigma^\star$ is well defined.

8.2 Notation

Denote $x^\star$ as a local stationary point of (NP), with corresponding dual solutions $y^\star, z^\star$. Define the set of active bounds at $x^\star$ as

$$\mathcal{A}(x^\star) := \{j \mid x_j \in \{\ell_j, u_j\}\}, \tag{8.6}$$

and define the critical cones $C_\phi(x^\star, z^\star)$ and $C(x^\star, z^\star)$ as

$$\mathcal{C}_\phi(x^\star, z^\star) := \left\{ p \,\middle|\, \begin{array}{ll} p_j = 0 & \text{if } z_j^\star \neq 0 \\ p_j \geqslant 0 & \text{if } x_j^\star = \ell_j \\ p_j \leqslant 0 & \text{if } x_j^\star = u_j \end{array} \right\}, \tag{8.7a}$$

$$\mathcal{C}(x^\star, z^\star) := \left\{ p \in \mathcal{C}_\phi(x^\star, z^\star) \,\middle|\, A(x^\star)^T p = 0 \right\}. \tag{8.7b}$$

Observe that by (A3), $\mathcal{C}_\phi(x^\star, z^\star) = \left\{p \mid p_j = 0 \text{ if } z_j^\star \neq 0\right\}$, so $p \in \mathcal{C}_\phi(x^\star, z^\star)$ if and only if $p = Q(x^\star)^{1/2}\bar{p}$ for some $\bar{p} \in \mathbb{R}^n$.

Let $H(x) = \nabla^2 f(x)$, $H_i(x) = \nabla^2 c_i(x)$, and define

$$g_L(x,y) := g(x) - A(x)y, \qquad g_\sigma(x) := g_L(x, y_\sigma(x)), \tag{8.8a}$$

$$H_L(x,y) := H(x) - \textstyle\sum_{i=1}^m y_i H_i(x), \qquad H_\sigma(x) := H_L(x, y_\sigma(x)) \tag{8.8b}$$

as the gradient and Hessian of $L(x, y)$ evaluated at x and y or $y_\sigma(x)$. We also define the matrix operators

$$R(x,v) := \nabla_x[Q(x)v] = \nabla_x \begin{bmatrix} q_i(x_1)v_1 \\ \vdots \\ q_i(x_n)v_n \end{bmatrix} = \operatorname{diag}\left(\begin{bmatrix} q_i'(x_1)v_1 \\ \vdots \\ q_i'(x_n)v_n \end{bmatrix} \right),$$

$$S(x,v) := \nabla_x[A(x)^T v] = \nabla_x \begin{bmatrix} g_1(x)^T v \\ \vdots \\ g_m(x)^T v \end{bmatrix} = \begin{bmatrix} v^T H_1(x) \\ \vdots \\ v^T H_m(x) \end{bmatrix},$$

$$T(x,w) := \nabla_x[A(x)w] = \nabla_x \left[\sum_{i=1}^m w_i g_i(x) \right] = \sum_{i=1}^m w_i H_i(x),$$

where $v \in \mathbb{R}^n$, $w \in \mathbb{R}^m$, and $T(x,w)$ is a symmetric matrix. (We do not use R until Chapter 10.) The operation of multiplying the adjoint of S with a vector w is described by

$$S(x,v)^T w = \left[\textstyle\sum_{i=1}^m w_i H_i(x)\right] v = T(x,w)v = T(x,w)^T v\,.$$

If M has full rank, the operators

$$P_M := M\left(M^T M\right)^{-1} M^T \qquad \text{and} \qquad \bar{P}_M := I - P$$

define, respectively, orthogonal projectors onto range(M) and its complement. We define $M^\dagger$ as the Moore-Penrose pseudoinverse, where $M^\dagger = (M^T M)^{-1} M^T$ if M has full column-rank.

8.3 Related work on penalty functions

Penalty functions have long been used to solve constrained problems by transforming them into problems with simpler constraints that penalize violations of feasibility. We provide a brief overview of common penalty methods and their relation to Fletcher's penalty $\phi_\sigma(x)$. A more detailed treatment is given by Pillo and Grippo (1984), Conn et al. (2000), and Nocedal and Wright (2006).

The simplest example is the quadratic penalty function (Courant, 1943), which removes the nonlinear constraints by adding $\frac{1}{2}\rho\|c(x)\|^2$ to the objective (for some $\rho > 0$). Similarly, bounds are replaced by $\frac{1}{2}\rho\|(x-\ell)_+\|^2 + \frac{1}{2}\rho\|(u-x)_+\|^2$. There are two main drawbacks: a sequence of optimization subproblems must be solved with increasing ρ, and a feasible solution is obtained only when $\rho \to \infty$. Further, these subproblems become increasingly ill-conditioned and difficult to solve.

An alternative to smooth non-exact penalty functions is an exact non-smooth function such as the 1-norm penalty $\rho\|c(x)\|_1 + \rho\max\{0, \ell - x, x - u\}$ (Nocedal and Wright, 2006, §17.2). However, only non-smooth optimization methods apply, which typically exhibit slower convergence. Maratos (1978) further noted that non-smooth merit functions may reject steps and prevent quadratic convergence.

Another distinct approach is the class of augmented Lagrangian methods, independently introduced by Hestenes (1969) and Powell (1969). For (NP), solvers such as LANCELOT (Conn, Gould, and Toint, 1991, 1992) minimize a sequence of augmented Lagrangians $L_{\rho_k}(x, y_k) = L(x, y_k) + \frac{1}{2}\rho_k\|c(x)\|^2$ over the bounds. When y_k is optimal $L_{\rho_k}(x, y_k)$ is exact for sufficiently large ρ_k, thus avoiding the stability issues of the quadratic penalty. However, a sequence of subproblems must be solved to drive y_k to optimality.

Although these penalty functions have often been successful in practice, in light of their drawbacks, a class of smooth exact penalty functions has been explored (Fletcher, 1970; Pillo and Grippo, 1984; Zavala and Anitescu, 2014). With smooth exact penalty functions, constrained optimization problems such as (NP) can be replaced by a single smooth bound-constrained optimization problem (or unconstrained if (NP) is equality-constrained only); provided the penalty parameter is sufficiently large. Approximate second-order methods can be applied to obtain at least superlinear local convergence. The price for smoothness is that a derivative of the penalty function requires a higher-order derivative from the original problem data. That is, evaluating ϕ_σ requires ∇f and ∇c, $\nabla\phi_\sigma$ requires $\nabla^2 f$ and $\nabla^2 c_i$; and so on. In practice, the third derivative terms are typically discarded, but it can be shown that superlinear convergence is retained.

Fletcher (1970) originally introduced a class of smooth exact penalty functions for equality-constrained problems with 3 papers (Fletcher, 1970; Fletcher and Lill, 1971; Fletcher, 1973a). In the equality-constrained case, ϕ_σ reduces to one of the penalty functions introduced by Fletcher. For inequality constrained problems with $\ell = 0$ and $u = \infty$, Fletcher (1973b) proposed the penalty function

$$\begin{aligned} \psi_\sigma(x) &:= f(x) - c(x)^T y_\sigma(x) - z_\sigma(x)^T x \\ y_\sigma(x), z_\sigma(x) &:= \operatorname*{argmin}_{y\in\mathbb{R}^m,\, z\geqslant 0} \tfrac{1}{2}\|A(x)y + z - g(x)\|_2^2 + \sigma c(x)^T y, \end{aligned} \tag{8.9}$$

to be minimized without any constraints. Although it can be shown that this penalty is exact, it is nonsmooth because of the bound constraints on the multiplier estimates: active-set changes on the bounds of the multiplier estimates z correspond to non-differentiable points for the penalty function. The nonsmooth penalty then requires a minimization method for nonsmooth problems, which often results in slower convergence.

Fletcher (1970) did not envision his method applied to large-scale problems and assumed "the matrices in the problem are non-sparse". Further, most developments surrounding this method focused on linesearch schemes that require computing an explicit Hessian approximation and using it to compute a Newton direction. One of our goals is to show how to adapt the method to large-scale problems by taking advantage of computational advances made since Fletcher's proposal. Improved sparse matrix factorizations and iterative methods for solving linear systems, and modern Newton-CG trust-region methods (Ph. L. Toint, 1981; Steihaug, 1983), play a key role in the efficient implementation of his penalty function.

Since Fletcher (1970), there has been significant work on smooth exact penalty methods that handle inequality constraints—see for example Pillo and Grippo (1984); Di Pillo and Grippo (1985); Boggs, Tolle, and Kearsley (1992); Zavala and Anitescu (2014). Many approaches replace the inequality constraints with equalities using squared slacks (Bertsekas, 1982), at which point the equality constrained problem is solved via a smooth exact penalty approach. (This is one approach for deriving ϕ_σ and (8.2); however, it is also possible to derive it directly from the first-order KKT conditions.) The penalty function in these cases is the augmented Lagrangian with the dual variables explicit and penalizes the gradient of the Lagrangian (Zavala and Anitescu, 2014), or withthe dual variables expressed as a function of the primals (Fletcher, 1970; Pillo and Grippo, 1984). Our penalty function (8.1) takes the latter approach, but defines this parametrization differently from previous approaches; rather than introducing additional dual variables for the bounds in (8.2), we change the norm of the least-squares problem according to the distance from the bounds, to approximate the complementarity conditions of first-order KKT points.

Chapter 9

The equality-constrained case

We first focus our attention on the equality-constrained case:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \ f(x)\ \ \text{subject to}\ \ c(x) = 0\ :\ y, \tag{NP-Eq}$$

that is, we let the bounds ℓ and u be infinite. For (NP-Eq), Fletcher's penalty function is

$$\phi_\sigma(x) := f(x) - c(x)^T y_\sigma(x), \tag{9.1a}$$

$$y_\sigma(x) := \underset{y}{\text{argmin}}\ \tfrac{1}{2}\|A(x)y - g(x)\|^2 + \sigma c(x)^T y. \tag{9.1b}$$

The form of $y_\sigma(x)$ is reminiscient of the variable-projection algorithm of Golub and Pereyra (1973) for separable nonlinear least-squares problems.

Instead of working directly with (NP-Eq), we solve the unconstrained problem (PP). We initially assume that (NP-Eq) satisfies (A2b) so that $y_\sigma(x)$ and $Y_\sigma(x)$ are uniquely defined. We relax this assumption to (A2a) in Section 9.4.

9.1 Properties of the penalty function

We show that $\phi_\sigma(x)$ naturally expresses the optimality conditions of (NP-Eq). This leads to an explicit expressions for the threshold value of the penalty parameter σ.

9.1.1 Derivatives of the penalty function

The gradient and Hessian of ϕ_σ may be written as

$$\nabla\phi_\sigma(x) = g_\sigma(x) - Y_\sigma(x)c(x), \tag{9.2a}$$

$$\nabla^2\phi_\sigma(x) = H_\sigma(x) - A(x)Y_\sigma(x)^T - Y_\sigma(x)A(x)^T - \nabla_x\left[Y_\sigma(x)c\right], \tag{9.2b}$$

where the last term $\nabla_x[Y_\sigma(x)c]$ purposely drops the argument on c to emphasize that this gradient is made on the product $Y_\sigma(x)c$ with $c := c(x)$ held fixed. This term involves third derivatives of f and c, and as we shall see, it is convenient and computationally efficient to ignore it. We leave it unexpanded.

9.1.2 Optimality conditions

The penalty function ϕ_σ is closely related to the Lagrangian $L(x, y)$ associated with (NP-Eq). To make this connection clear, we define the Karush-Kuhn-Tucker (KKT) optimality conditions for (NP-Eq) in terms of formulas related to ϕ_σ. From the definition of ϕ_σ and y_σ and the derivatives (9.2), the following definitions are equivalent to the KKT conditions.

Definition 9.1 (First-order KKT point) *The point $x^\star$ is a first-order KKT point of (NP-Eq) if for any $\sigma \geqslant 0$ the following hold:*

$$c(x^\star) = 0, \tag{9.3a}$$
$$\nabla \phi_\sigma(x^\star) = 0. \tag{9.3b}$$

The Lagrange multipliers associated with $x^\star$ are $y^\star := y_\sigma(x^\star)$.

Definition 9.2 (Second-order KKT point) *The first-order KKT point $x^\star$ satisfies the second-order necessary KKT condition for (NP-Eq) if for any $\sigma \geqslant 0$,*

$$p^T \nabla^2 \phi_\sigma(x^\star) p \geqslant 0 \quad \textit{for all } p \textit{ such that } A(x^\star)^T p = 0,$$

i.e., $\bar{P}(x^\star)\nabla^2\phi_\sigma(x^\star)\bar{P}(x^\star) \succeq 0$. The condition is sufficient if the inequality is strict.

The second-order KKT condition says that at $x^\star$, ϕ_σ has nonnegative curvature along directions in the tangent space of the constraints. However, at $x^\star$, increasing σ will increase curvature along the normal cone of the feasible set. We derive a threshold value for σ that causes ϕ_σ to have nonnegative curvature at a second-order KKT point $x^\star$, as well as a condition on σ that ensures stationary points of ϕ_σ are primal feasible. For a given first- or second-order KKT pair $(x^\star, y^\star)$ of (NP-Eq), we define $P(x^\star) := P_{A(x^\star)}$ and

$$\sigma^\star := \tfrac{1}{2}\lambda^+_{\max}\left(P(x^\star)H_L(x^\star, y^\star)P(x^\star)\right), \tag{9.4}$$

where $\lambda^+_{\max}(\cdot)$ is the maximum of the largest eigenvalue and zero.

Lemma 9.3 *If $c(x) \in \operatorname{range}(A(x)^T)$, then $y_\sigma(x)$ satisfies*

$$A(x)^T A(x) y_\sigma(x) = A(x)^T g(x) - \sigma c(x). \tag{9.5}$$

Further, if $A(x)$ has full rank, then

$$A(x)^T A(x) Y_\sigma(x)^T = A(x)^T[H_\sigma(x) - \sigma I] + S(x, g_\sigma(x)). \tag{9.6}$$

Proof. For any x, the necessary and sufficient optimality conditions for (9.1b) give (9.5). By differentiating both sides of (9.5), we obtain

$$S(x, A(x)y_\sigma(x)) + A(x)^T\left[T(x, y_\sigma(x)) + A(x)Y_\sigma(x)^T\right] = S(x, g(x)) + A(x)^T[H(x) - \sigma I].$$

From definitions (8.8), we obtain (9.6). □

Theorem 9.4 *Suppose $\nabla\phi_\sigma(\bar{x}) = 0$ for some $\bar{x}$, and let $x_1^\star$ and $x_2^\star$ be first- and second-order necessary KKT points, respectively, for (NP-Eq). Let $\sigma^\star$ be defined as in (9.4). Then*

$$\sigma > \|A(\bar{x})^T Y_\sigma(\bar{x})\| \quad \Longrightarrow \quad g(\bar{x}) = A(\bar{x})y_\sigma(\bar{x}), \quad c(\bar{x}) = 0; \tag{9.7a}$$
$$\sigma \geqslant \|A(x_1^\star)Y_\sigma(x_1^\star)^T\| \quad \Longrightarrow \quad \sigma \geqslant \sigma^\star; \tag{9.7b}$$
$$\nabla^2\phi_\sigma(x_2^\star) \succeq 0 \quad \Longleftrightarrow \quad \sigma \geqslant \sigma^\star. \tag{9.7c}$$

If $x_2^\star$ is second-order sufficient, then the inequalities in (9.7c) hold strictly.

Proof. We prove (9.7a), (9.7c), and (9.7b) in order.

Proof of (9.7a): The condition $\nabla\phi_\sigma(\bar{x}) = 0$ implies that

$$g(\bar{x}) = A(\bar{x})y_\sigma(\bar{x}) + Y_\sigma(\bar{x})c(\bar{x}).$$

Substituting (9.5) evaluated at $\bar{x}$ into this equation yields, after simplifying,

$$A(\bar{x})^T Y_\sigma(\bar{x})c(\bar{x}) = \sigma c(\bar{x}).$$

Taking norms of both sides and using the triangle inequality gives the inequality $\sigma\|c(\bar{x})\| \leqslant \|A(\bar{x})^T Y_\sigma(\bar{x})\|\,\|c(\bar{x})\|$, which immediately implies that $c(\bar{x}) = 0$. The condition $\nabla\phi_\sigma(\bar{x}) = 0$ then becomes $g_\sigma(\bar{x}) = 0$.

Proof of (9.7c): Because $x_2^\star$ satisfies (9.3), we have $g_\sigma(x_2^\star) = 0$ and $y^* = y_\sigma(x)$, independently of σ. It follows from (9.6), $H_L(x_2^\star, y^\star) = H_\sigma(x_2^\star)$, and the definition of the projector $P := P_{A(x_2^\star)}$ that

$$A(x_2^\star)Y_\sigma(x_2^\star)^T = P(H_L(x_2^\star, y^\star) - \sigma I). \tag{9.8}$$

We substitute this equation into (9.2b) and use the relation $P + \bar{P} = I$ to obtain

$$\nabla^2\phi_\sigma(x_2^\star) = \bar{P}H_L(x_2^\star, y^\star)\bar{P} - PH_L(x_2^\star, y^\star)P + 2\sigma P.$$

Note that $\bar{P}H_L(x_2^\star, y^\star)\bar{P} \succeq 0$ because $x_2^\star$ is a second-order KKT point, so σ needs to be sufficiently large that $2\sigma P - PH_L(x_2^\star, y^\star)P \succeq 0$, which is equivalent to $\sigma \geqslant \sigma^\star$.

Proof of (9.7b): With $x_1^\star$ in (9.8), $y^* = y_\sigma(x_1^\star)$ and properties of $P := P_{A(x_1^\star)}$, we have

$$\begin{aligned}\sigma \geqslant \|A(x_1^\star)Y_\sigma(x_1^\star)^T\| &= \|P(H_L(x_1^\star, y^\star) - \sigma I)\| \\ &\geqslant \|P(H_L(x_1^\star, y^\star) - \sigma I)P\| \\ &\geqslant \|PH_L(x_1^\star, y^\star)P\| - \sigma\|P\| \geqslant 2\sigma^\star - \sigma.\end{aligned}$$

Thus, $\sigma \geqslant \sigma^\star$ as required. □

According to (9.7c), if $x^\star$ is a second-order KKT point, there exists a threshold value $\sigma^\star$ such that ϕ_σ has nonnegative curvature for $\sigma \geqslant \sigma^\star$. Unfortunately, as for many exact penalty functions, Theorem 9.4 does not discount the possibility of stationary points of $\phi_\sigma(x)$ that are not feasible points of (NP-Eq).

Consider, for example, the feasibility problem with $f(x) = 0$ and $c(x) = x^3 + x - 2$. The only minimizer is $x^\star = 1$. The penalty function

$$\phi_\sigma(x) = \sigma\frac{(x^3 + x - 2)^2}{(3x^2 + 1)^2}$$

is defined everywhere and has local minimizers at $x_1 = 1$ (the solution) and $x_2 \approx -1.56$. Because the stationary points are independent of σ in this case, ϕ_σ always has the spurious local minimizer x_2.

Note that we rarely encounter spuri and ous stationary points in practice, and usually minimizers of $\phi_\sigma(x)$ correspond to feasible (and therefore optimal) points of (NP-Eq).

9.1.3 Additional quadratic penalty

In light of Theorem 9.4, it is somewhat unsatisfying that local minimizers of $\phi_\sigma(x)$ might not be local minimizers of (NP-Eq). We may add a quadratic penalty term to promote feasibility, and under mild conditions ensure that minimizers of ϕ_σ are KKT points of (NP-Eq). Like Fletcher (1970), we define

$$\phi_{\sigma,\rho}(x) := \phi_\sigma(x) + \tfrac{1}{2}\rho\|c(x)\|^2 = f(x) - [y_\sigma(x) - \tfrac{1}{2}\rho c(x)]^T c(x). \tag{9.9}$$

The multiplier estimates are now shifted by the constraint violation, similar to an augmented Lagrangian. All expressions for the derivatives follow as before with an additional term from the quadratic penalty.

Theorem 9.5 *Let $\mathcal{S} \subset \mathbb{R}^n$ be a compact set, and suppose that $\sigma_{\min}(A(x)) \geqslant \lambda > 0$ for all $x \in \mathcal{S}$. Then for any $\sigma \geqslant 0$ there exists $\rho^\star(\sigma) > 0$ such that for all $\rho > \rho^\star(\sigma)$, if $\nabla\phi_{\sigma,\rho}(\bar{x}) = 0$ and $\bar{x} \in \mathcal{S}$, then $\bar{x}$ is a first-order KKT point for* (NP-Eq).

Proof. The condition $\nabla\phi_{\sigma,\rho}(\bar{x}) = 0$ implies that

$$g(\bar{x}) - A(\bar{x})y_\sigma(\bar{x}) - Y_\sigma(\bar{x})c(\bar{x}) = \rho A(\bar{x})c(\bar{x}).$$

We premultiply with $A(\bar{x})^T$ and use (9.5) to obtain

$$\left(\sigma I - A(\bar{x})^T Y_\sigma(\bar{x})\right) c(\bar{x}) = \rho A(\bar{x})^T A(\bar{x}) c(\bar{x}). \tag{9.10}$$

The left-hand side of (9.10) is a continuous matrix function with finite supremum $R(\sigma) := \sup_{x\in\mathcal{S}} \|\sigma I - A(x)^T Y(x)\|$ defined over the compact set $\mathcal{S}$. We now define $\rho^\star(\sigma) := R(\sigma)/\lambda^2$, so that for $\rho > \rho^\star(\sigma)$, if $c(\bar{x}) \neq 0$,

$$\begin{aligned} R(\sigma)\|c(\bar{x})\| &\geqslant \|\sigma I - A(\bar{x})^T Y_\sigma(\bar{x})\| \cdot \|c(\bar{x})\| \\ &\geqslant \|\left(\sigma I - A(\bar{x})^T Y_\sigma(\bar{x})\right) c(\bar{x})\| \\ &= \rho\|A(\bar{x})^T A(\bar{x}) c(\bar{x})\| \\ &\geqslant \rho\lambda^2\|c(\bar{x})\| > R(\sigma)\|c(\bar{x})\|, \end{aligned}$$

which is a contradiction, implying $\|c(\bar{x})\| = 0$, so that $\bar{x}$ is feasible for (NP-Eq). Because $c(\bar{x}) = 0$ and $\nabla\phi_\sigma(x) = \nabla\phi_{\sigma,\rho}(\bar{x}) = 0$, $\bar{x}$ is a first-order KKT point. ∎

We briefly consider the case $\sigma = 0$ and $\rho > 0$. The threshold value to ensure positive semidefiniteness of $\nabla^2\phi_{\sigma,\rho}$ at a second-order KKT pair $(x^\star, y^\star)$ of (NP-Eq) is

$$\rho^\star = \lambda_{\max}^{+}\left(A(x^\star)^\dagger H_L(x^\star, y^\star) A(x^\star)^\dagger\right).$$

This threshold parameter is more difficult to interpret in terms of the original problem data compared to $\sigma^\star$ because of the pseudoinverse. The following theorem is analogous to Theorem 9.4, but we omit the proof as it is nearly identical.

Theorem 9.6 *Suppose $\sigma = 0$ and $\rho \geqslant 0$. Let $\nabla\phi_{\sigma,\rho}(\bar{x}) = 0$ for some $\bar{x}$, and let $x^\star$ be a* *second-order necessary KKT point for* (NP-Eq). *Then*

$$\rho > \|A(\bar{x})^\dagger Y_\sigma(\bar{x})\| \quad \Longrightarrow \quad g(\bar{x}) = A(\bar{x})y_\sigma(\bar{x}), \quad c(\bar{x}) = 0; \tag{9.11a}$$

$$\nabla^2 \phi_\sigma(x^\star) \succeq 0 \quad \Longleftrightarrow \quad \rho \geqslant \rho^\star. \tag{9.11b}$$

If $x^\star$ is second-order sufficient, the inequalities in (9.11b) *hold strictly.*

Using $\rho > 0$ can help cases where attaining feasibility is problematic for moderate values of σ. For simplicity we let $\rho = 0$ from now on, because it is trivial to evaluate $\phi_{\sigma,\rho}$ and its derivatives if one can compute ϕ_σ.

9.1.4 Scale invariance

Note that ϕ_σ is invariant under diagonal scaling of the constraints, i.e., if $c(x)$ is replaced by $Dc(x)$ for some diagonal matrix D, then ϕ_σ is unchanged. It is an attractive property that ϕ_σ and $\sigma^\star$ are independent of some model formulation choices, like the Lagrangian. However, like the augmented Lagrangian, $\phi_{\sigma,\rho}$ with $\rho > 0$ is not scale invariant because of the quadratic term; thus, constraint scaling is an important consideration if $\phi_{\sigma,\rho}$ is used.

9.2 Evaluating the penalty function

The main challenge in evaluating ϕ_σ and its gradient is solving the shifted least-squares problem (9.1b) in order to compute $y_\sigma(x)$, and computing the gradient $Y_\sigma(x)$. Below we show it is possible to compute products $Y_\sigma(x)v$ and $Y_\sigma(x)^T u$ by solving structured linear systems involving the matrix used to compute $y_\sigma(x)$. If direct methods are used, a single factorization that gives the solution (9.1b) is sufficient for all products.

For this section, it is convenient to drop the arguments on the various functions and assume they are all evaluated at a point x for some parameter σ. For example, $y_\sigma = y_\sigma(x)$, $A = A(x)$, $Y_\sigma = Y_\sigma(x)$, $H_\sigma = H_\sigma(x)$, $S_\sigma = S(x, g_\sigma(x))$, etc. We also express (9.6) using the shorthand notation

$$A^T A Y_\sigma^T = A^T[H_\sigma - \sigma I] + S_\sigma. \tag{9.12}$$

We first describe how to compute products $Y_\sigma u$ and $Y_\sigma^T v$, then describe how to put those pieces together to evaluate the penalty function and its derivatives.

9.2.1 Computing the product $\mathbf{Y_\sigma u}$

For a given $u \in \mathbb{R}^m$, we premultiply (9.12) by $u^T(A^TA)^{-1}$ to obtain

$$\begin{aligned} Y_\sigma u &= [H_\sigma - \sigma I]A(A^TA)^{-1}u + S_\sigma^T(A^TA)^{-1}u \\ &= [H_\sigma - \sigma I]v - S_\sigma^T w, \end{aligned}$$

where we define $w = -(A^TA)^{-1}u$ and $v = -Aw$. Observe that w and v solve the system

$$\begin{bmatrix} I & A \\ A^T & \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ u \end{bmatrix}. \tag{9.13}$$

Algorithm 6 formalizes the process.

Algorithm 6 Computing the matrix-vector product $Y_\sigma u$

```
1: $(v, w) \leftarrow$ solution of (9.13)
2: return $[H_\sigma - \sigma I]v - S_\sigma^T w$
```

9.2.2 Computing the product $\mathbf{Y_\sigma^T v}$

Multiplying both sides of (9.12) on the right by v gives

$$A^T A(Y_\sigma^T v) = A^T([H_\sigma - \sigma I]v) + (S_\sigma v).$$

The required product $u = Y_\sigma^T v$ is in the solution of the system

$$\begin{bmatrix} I & A \\ A^T & \end{bmatrix} \begin{bmatrix} r \\ u \end{bmatrix} = \begin{bmatrix} [H_\sigma - \sigma I]v \\ -S_\sigma v \end{bmatrix}. \tag{9.14}$$

Algorithm 7 formalizes the process.

Algorithm 7 Computing the matrix-vector product $Y_\sigma^T v$

```
1: Evaluate $[H_\sigma - \sigma I]v$ and $S_\sigma v$
2: $(r, u) \leftarrow$ solution of (9.14)
3: return $u$
```

9.2.3 Computing multipliers and first derivatives

The multiplier estimates y_σ can be obtained from the optimality conditions for (9.1b):

$$\begin{bmatrix} I & A \\ A^T & \end{bmatrix} \begin{bmatrix} g_\sigma \\ y_\sigma \end{bmatrix} = \begin{bmatrix} g \\ \sigma c \end{bmatrix}, \tag{9.15}$$

which also gives g_σ. Algorithm 6 then gives $Y_\sigma c$ and hence $\nabla\phi_\sigma$ in (9.2a).

Observe that we can re-order operations to take advantage of specialized solvers. Consider the pair of systems

$$\begin{bmatrix} I & A \\ A^T & \end{bmatrix} \begin{bmatrix} d \\ y \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} I & A \\ A^T & \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ c \end{bmatrix}. \tag{9.16}$$

We have $g_\sigma = d + \sigma v$ and $y_\sigma = y + \sigma w$, while the computation of $Y_\sigma c$ is unchanged. The systems in (9.16) correspond to pure least-squares and least-norm problems respectively. Specially tailored solvers may be used to improve efficiency or accuracy. This is further explored in Section 9.2.5.

9.2.4 Computing second derivatives

We can approximate $\nabla^2\phi_\sigma$ using (9.2b) and (9.6) in two ways according to

$$\begin{aligned}\nabla^2\phi_\sigma \approx B_1 &:= H_\sigma - AY_\sigma^T - Y_\sigma A^T \qquad (9.17a)\\ &= H_\sigma - \widetilde{P}H_\sigma - H_\sigma\widetilde{P} + 2\sigma\widetilde{P} - A(A^TA)^{-1}S_\sigma - S_\sigma^T(A^TA)^{-1}A\\ \nabla^2\phi_\sigma \approx B_2 &:= H_\sigma - \widetilde{P}H_\sigma - H_\sigma\widetilde{P} + 2\sigma\widetilde{P}, \qquad (9.17b)\end{aligned}$$

where $\widetilde{P} = A(A^TA)^{-1}A^T$. Note that $\widetilde{P} = P_A$ here, but this changes when regularization is used; see Section 9.4. The first approximation ignores $\nabla[Y_\sigma(x)c]$ in (9.2b), while the second ignores $S_\sigma = S(x, g_\sigma(x))$. Because we expect $c(x) \to 0$ and $g_\sigma(x) \to 0$, B_1 and B_2 are similar to Gauss-Newton approximations to $\nabla^2\phi_\sigma(x)$, and as Fletcher (1973a, Theorem 2) shows, using them in a Newton-like scheme is sufficient for quadratic convergence if (A1a) is satisfied.

Because $\widetilde{P}$ is a projection on range(A), we can compute products $\widetilde{P}u$ by solving

$$\begin{bmatrix} I & A \\ A^T & \end{bmatrix}\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} u \\ 0 \end{bmatrix} \qquad (9.18)$$

and setting $\widetilde{P}u \leftarrow u - p$. Note that with regularization, the $(2, 2)$ block of this system is modified and $\widetilde{P}$ is no longer a projection; see Section 9.4.

The approximations (9.17a) and (9.17b) trade Hessian accuracy for computational efficiency. If the operator $S(x, v)$ is not immediately available (or not efficiently implemented), it may be avoided. Using B_2 requires only least-square solves, which allows us to apply specialized solvers (e.g., LSQR (Paige and Saunders, 1982a)), which cannot be done when products with Y_σ^T are required.

9.2.5 Solving the augmented linear system

We discuss some approaches to solving linear systems of the form

$$\mathcal{K}\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} w \\ z \end{bmatrix}, \qquad \mathcal{K} := \begin{bmatrix} I & A \\ A^T & -\delta^2 I \end{bmatrix}, \qquad (9.19)$$

which have appeared repeatedly in this section. Although $\delta = 0$ so far, we now look ahead to regularized systems because they require only minor modification. Let $(p^\star, q^\star)$ solve (9.19). Define $A_\delta := \begin{bmatrix} A^T & \delta I \end{bmatrix}^T$ when $\delta > 0$; otherwise $A_\delta := A$.

Conceptually it is not important how this system is solved as long as it is with sufficient accuracy. However, from a practical point of view, this is the most computationally intensive part of using ϕ_σ. Different solution methods have different advantages and limitations, depending on the size and sparsity of A, whether A is available explicitly, and the prescribed solution accuracy. One option is to use direct methods: factorize $\mathcal{K}$ once per iteration and use the factors to solve with each right-hand side. Several factorization-based approaches can be employed with various advantages and drawbacks—see Appendix C for details.

In line with the goal of creating a factorization-free solver for minimizing ϕ_σ, we discuss iterative methods for solving (9.19), particularly Krylov subspace solvers. This approach has two potential advantages: if a good preconditioner $\mathcal{P} \approx A^T A$ is available, solving (9.19) could be much more efficient than with direct methods, and we can take advantage of solvers using inexact function values, gradients, or Hessian products by solving (9.19) approximately.

When $z = 0$, (9.19) is a (regularized) least-squares problem: $\min_q \|A_\delta q - w_\delta\|$. We advocate for LSQR (Paige and Saunders, 1982a), which ensures that the error in iterates p_k and q_k decreases monotonically at every iteration. Furthermore, we can obtain upper bounds on $\|p^\star - p_k\|$ and $\|q^\star - q_k\|$ using LSLQ (Chapter 4) when an understimate of $\sigma_{\min}(A\mathcal{P}^{-1/2})$ is available. (Note that the error norm for q depends on the preconditioner.)

When $w = 0$, (9.19) is a least-norm problem: $\min_p \|p\|$ s.t. $A_\delta^T p = z$. We then advocate for CRAIG (Craig, 1955) because it minimizes the error in the Krylov subspace. Given the same underestimate of $\sigma_{\min}(A\mathcal{P}^{-1/2})$, we use LNLQ (Chapter 5) to bound the error norms for p and q.

Recall that ϕ_σ and $\nabla\phi_\sigma$ can be computed by solving only least-squares and least-norm problems (only one of w and z is nonzero at a time). Furthermore, if (9.17b) is used, the remaining solves with $\mathcal{K}$ are equivalent to least-squares solves. If both w and z are nonzero (for products with Y_σ^T), we can shift the right-hand side of (9.19) and solve the system

$$\mathcal{K}\begin{bmatrix}\bar{p}\\ q\end{bmatrix} = \begin{bmatrix}0\\ z - A^T w\end{bmatrix}, \qquad p = \bar{p} + w.$$

Thus, (9.19) can in general be solved by CRAIG or LNLQ.

Although $\mathcal{K}$ is symmetric indefinite, we do not recommend methods such as MINRES or SYMMLQ (Paige and Saunders, 1975). Orban and Arioli (2017) show that if full-space methods are applied directly to $\mathcal{K}$ then every other iteration of the solver makes little progress. If solves with $\mathcal{P}$ can only be performed approximately, it may be necessary to apply flexible variants of nonsymmetric full-space methods to $\mathcal{K}$, such as GMRES (Saad and Schultz, 1986).

9.3 Maintaining explicit constraints

We consider a variation of (NP) where some of the constraints $c(x)$ are easy to maintain explicitly; for example, some are linear. We can then maintain feasibility for a subset of the constraints, the contours of the ϕ_σ are simplified, and as we show soon, the threshold penalty parameter $\sigma^\star$ is decreased. We discuss the case where some of the constraints are linear, but it is possible to extend the theory to any type of constraint.

Consider the problem

$$\begin{aligned}\underset{x\in\mathbb{R}^n}{\text{minimize}}\quad & f(x),\\ \text{subject to}\quad & c(x) = 0 \ :\ y,\\ & B^T x = d \ :\ w,\end{aligned} \tag{NP-EXP}$$

where we have nonlinear constraints $c(x) \in \mathbb{R}^{m_1}$ and linear constraints $B^T x = d$ with $B \in \mathbb{R}^{n\times m_2}$, so that $m_1 + m_2 = m$. The respective dual variables for the constraints are $y \in \mathbb{R}^{m_1}$ and $w \in \mathbb{R}^{m_2}$. We assume that (NP-EXP) at least satisfies (A2a), so that B has full column rank. We define the penalty function problem to be

$$\begin{aligned}&\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \phi_\sigma(x) := f(x) - c(x)^T y_\sigma(x) \quad \text{subject to} \quad B^T x = d,\\ &\begin{bmatrix} y_\sigma(x) \\ w_\sigma(x) \end{bmatrix} := \underset{y,w}{\operatorname{argmin}}\ \tfrac{1}{2}\|A(x)y + Bw - g(x)\|^2 + \sigma \begin{bmatrix} c(x) \\ B^T x - d \end{bmatrix}^T \begin{bmatrix} y \\ w \end{bmatrix},\end{aligned} \tag{9.20}$$

which is similar to (PP) except the linear constraints are not penalized in $\phi_\sigma(x)$, and the penalty function is minimized subject to the linear constraints. A possibility is to penalize the linear constraints as well, while keeping the linear constraints explicit; however, penalizing the linear constraints in $\phi_\sigma(x)$ introduces additional nonlinearity, and if all constraints are linear, it makes sense that the penalty function reduces to (NP-EXP).

9.3.1 Properties of the modified penalty function

Define $W_\sigma(x) := \nabla w_\sigma(x) \in \mathbb{R}^{n\times m_2}$, and $C(x) := \begin{bmatrix} A(x) & B \end{bmatrix}$ as the Jacobian of all constraints. The operators $g_\sigma(x)$, $H_\sigma(x)$, $S(x,v)$ and $T(x,w)$ are still defined over all constraints, not just the nonlinear ones, and so they act on $C(x)$ instead of $A(x)$. Define

$$g_\sigma^y(x) = g(x) - A(x)y_\sigma(x) \tag{9.21}$$

as the gradient of the partial Lagrangian with respect to the nonlinear constraints $c(x)$ only (note that the linear constraints do not affect H_σ). The gradient and Hessian of the penalty function are the same as (9.2), except that $g_\sigma(x)$ is replaced by $g_\sigma^y(x)$ in (9.2a).

We restate the optimality conditions for (NP-EXP) in terms of the penalty function.

Definition 9.7 (First-order KKT point) *The point $x^\star$ is a first-order KKT point of (NP-EXP) if there exists a Lagrange multiplier $w^\star \in \mathbb{R}^{m_2}$ for the linear constraints such that for any $\sigma \geqslant 0$ the following hold:*

$$c(x^\star) = 0, \tag{9.22a}$$

$$B^T x^\star = d, \tag{9.22b}$$

$$\nabla\phi_\sigma(x^\star) = Bw^\star. \tag{9.22c}$$

The elements of $y^\star := y_\sigma(x^\star)$ and $w^\star := w_\sigma(x^\star)$ are the Lagrange multipliers of (NP-EXP) associated with $x^\star$.

Definition 9.8 (Second-order KKT point) *The first-order KKT point $x^\star$ satisfies the second-order necessary KKT condition for (NP-EXP) if for any $\sigma \geqslant 0$,*

$$p^T \nabla^2\phi_\sigma(x^\star)p \geqslant 0 \text{ for all } p \text{ such that } C(x^\star)^T p = 0. \tag{9.23}$$

The condition is sufficient if the above inequality is strict.

For a given first- or second-order KKT solution $(x^\star, y^\star)$, the threshold penalty parameter becomes

$$\sigma^\star := \tfrac{1}{2}\lambda^+_{\max}\left(\bar{P}_B P_C H_L(x^\star, y^\star) P_C \bar{P}_B\right) \leqslant \tfrac{1}{2}\lambda^+_{\max}\left(P_C H_L(x^\star, y^\star) P_C\right). \tag{9.24}$$

Inequality (9.24) holds because $\bar{P}_B$ is an orthogonal projector. If the linear constraints were not explicit, the threshold value would be the right-most term in (9.24). Intuitively, the threshold penalty value decreases by the amount of the top eigenspace of the Lagrangian Hessian that lies in the range of B^T, because positive semidefiniteness of $\nabla^2\phi_\sigma(x^\star)$ along that space is guaranteed by Definition 9.8.

The following result is analogous to Theorem 9.4 with the smaller threshold value.

Theorem 9.9 *Suppose $\bar{x}$ is a first-order necessary KKT point for (9.20):*

$$B^T\bar{x} = d,$$
$$\nabla\phi_\sigma(\bar{x}) = Bw^\star,$$

and let $x_1^\star$ and $x_2^\star$ be first- and second-order necessary KKT points respectively for (NP-EXP). Then for all $p \neq 0$ such that $B^Tp = 0$,

$$\sigma > \|A(\bar{x})^T\bar{P}_BY_\sigma(\bar{x})\| \quad\Longrightarrow\quad g(\bar{x}) = A(\bar{x})y_\sigma(\bar{x}) + Bw_\sigma(\bar{x}), \quad c(\bar{x}) = 0; \tag{9.25a}$$
$$\sigma \geqslant \|\bar{P}_BA(x_1^\star)Y_\sigma(x_1^\star)^T\| \quad\Longrightarrow\quad \sigma \geqslant \sigma^\star; \tag{9.25b}$$
$$p^T\nabla^2\phi_\sigma(x_2^\star)p \geq 0 \quad\Longleftrightarrow\quad \sigma \geqslant \sigma^\star. \tag{9.25c}$$

If $x_2^\star$ is second-order sufficient, the inequalities in (9.25c) hold strictly.

Proof. We prove (9.25a), (9.25c) and (9.25b) in order. Observe that the multiplier estimates $y_\sigma(x)$ and $w_\sigma(x)$ satisfy

$$C(x)^TC(x)\begin{bmatrix} y_\sigma(x) \\ w_\sigma(x) \end{bmatrix} = C(x)^Tg(x) - \sigma\begin{bmatrix} c(x) \\ B^Tx - d \end{bmatrix}. \tag{9.26}$$

Proof of (9.25a): If $\bar{x}$ is a first-order KKT point for (9.20), then

$$Bw^\star = g(\bar{x}) - A(\bar{x})y_\sigma(\bar{x}) - Y_\sigma(\bar{x})c(\bar{x}),$$

and by multiplying both sides by $C(\bar{x})^T$ and using (9.26) we have

$$\begin{bmatrix} A(\bar{x})^TBw^\star \\ B^TBw^\star \end{bmatrix} = \sigma\begin{bmatrix} c(\bar{x}) \\ 0 \end{bmatrix} + \begin{bmatrix} A(\bar{x})^TBw_\sigma(\bar{x}) \\ B^TBw_\sigma(\bar{x}) \end{bmatrix} - \begin{bmatrix} A(\bar{x})^TY_\sigma(\bar{x})c(\bar{x}) \\ B^TY_\sigma(\bar{x})c(\bar{x}) \end{bmatrix},$$

so that $w_\sigma(\bar{x}) = w^\star + (B^TB)^{-1}B^TY_\sigma(\bar{x})c(\bar{x})$. Substituting $w_\sigma(\bar{x})$ into the first block of equations and rearranging gives

$$A(\bar{x})\bar{P}_BY_\sigma(\bar{x})c(\bar{x}) = \sigma c(\bar{x}).$$

The triangle inequality gives $\sigma\|c(\bar{x})\| \leqslant \|A(\bar{x})^T\bar{P}_BY_\sigma(\bar{x})\|\|c(\bar{x})\|$, implying $c(\bar{x}) = 0$. Then $w_\sigma(\bar{x}) = w^\star$ and $g_\sigma(\bar{x}) = 0$, so $\bar{x}$ is a first-order KKT point for (NP-EXP).

Proof of (9.25c): As in the proof of (9.7c), we differentiate (9.26) to obtain

$$C(x)^TC(x)\begin{bmatrix} Y_\sigma(x)^T \\ W_\sigma(x)^T \end{bmatrix} = C(x)^T[H_\sigma(x) - \sigma I] + S(x, g_\sigma(x)). \tag{9.27}$$

Because $x_2^\star$ satisfies first-order conditions (9.22), $g_\sigma(x) = 0$ and so for $P_C = P_C(x)$,

$$A(x_2^\star)Y_\sigma(x_2^\star)^T + BW_\sigma(x_2^\star)^T = P_C[H_\sigma(x_2^\star) - \sigma I]. \tag{9.28}$$

Substituting into (9.2b) and using $H_\sigma(x_2^\star) = H_L(x_2^\star, y^\star)$ and $P_C + \bar{P}_C = I$ gives

$$\nabla^2\phi_\sigma(x_2^\star) = \bar{P}_C H_L(x_2^\star, y^\star)\bar{P}_C - P_C H_L(x_2^\star, y^\star)P_C + 2\sigma P_C - BW_\sigma(x)^T - W_\sigma(x)B^T.$$

Because $B^T p = 0$, we can write $p = \bar{P}_B \bar{p}$ and hence

$$\begin{aligned} &0 \leqslant p^T\nabla^2\phi_\sigma(x_2^\star)p \\ \Longleftrightarrow\ &0 \preceq \bar{P}_B\bar{P}_C H_L(x_2^\star, y^\star)\bar{P}_C\bar{P}_B - \bar{P}_B P_C H_L(x_2^\star, y^\star)P_C\bar{P}_B + 2\sigma\bar{P}_B P_C\bar{P}_B, \end{aligned}$$

which is equivalent to $\sigma \geqslant \sigma^\star$.

Proof of (9.25b). With $x_1^\star$ in (9.28) and again using properties of P_C and $\bar{P}_B$,

$$\begin{aligned} \sigma \geqslant \|\bar{P}_B A(x_1^\star)Y_\sigma(\bar{x}_1)\| &= \|\bar{P}_B P_C(H_L(x_1^\star, y^\star) - \sigma I)\| \\ &\geqslant \|\bar{P}_B P_C(H_L(x_1^\star, y^\star) - \sigma I)P_C\bar{P}_B\| \\ &\geqslant \|\bar{P}_B P_C H_L(x_1^\star, y^\star)P_C\bar{P}_B\| - \|\sigma P_C\bar{P}_B\| \\ &\geqslant 2\sigma^\star - \sigma. \end{aligned}$$

Thus $\sigma \geqslant \sigma^\star$ as required. ◻

9.3.2 Evaluating the penalty function and derivatives

We again drop the arguments on functions and assume they are evaluated at x for some σ. The multipliers for evaluating the penalty function are obtained by solving

$$\begin{bmatrix} I & A & B \\ A^T & & \\ B^T & & \end{bmatrix}\begin{bmatrix} g_\sigma \\ y_\sigma \\ w_\sigma \end{bmatrix} = \begin{bmatrix} g \\ \sigma c \\ \sigma(Bx - d) \end{bmatrix}. \tag{9.29}$$

To compute the gradient and Hessian products, we use the identity

$$C^T C\begin{bmatrix} Y_\sigma^T \\ W_\sigma^T \end{bmatrix} = C^T[H_\sigma - \sigma I] + S_\sigma \tag{9.30}$$

to obtain the necessary products with Y_σ and Y_σ^T. Observe that

$$Y_\sigma u = \begin{bmatrix} Y_\sigma & W_\sigma \end{bmatrix}\begin{bmatrix} u \\ 0 \end{bmatrix}, \qquad Y_\sigma^T v = \begin{bmatrix} I & 0 \end{bmatrix}\begin{bmatrix} Y_\sigma^T \\ W_\sigma^T \end{bmatrix} v,$$

so that Algorithm 6 and Algorithm 7 can be applied.

Note that to compute $\nabla\phi_\sigma(x)$, g_σ^y is not available directly from the solution to (9.29) and must be computed explicitly using (9.21).

Approximate products with $\nabla^2\phi_\sigma$ can be computed via

$$\begin{aligned}\nabla^2\phi_\sigma \approx B_1 &:= H_\sigma - AY^T - YA^T \qquad (9.31)\\ &= H_\sigma - \begin{bmatrix} A & 0\end{bmatrix}(C^TC)^{-1}C^T(H_\sigma - \sigma I) - \begin{bmatrix} A & 0\end{bmatrix}(C^TC)^{-1}S_\sigma \\ &\qquad - (H_\sigma - \sigma I)C(C^TC)^{-1}\begin{bmatrix} A^T \\ 0\end{bmatrix} - S_\sigma(C^TC)^{-1}\begin{bmatrix} A^T \\ 0\end{bmatrix}\\ \approx B_2 &:= H_\sigma - \begin{bmatrix} A & 0\end{bmatrix}C^\dagger(H_\sigma - \sigma I) - (H_\sigma - \sigma I)\left(C^\dagger\right)^T\begin{bmatrix} A^T \\ 0\end{bmatrix}. \qquad (9.32)\end{aligned}$$

For products with the pseudoinverse and its transpose, we can compute

$$\begin{bmatrix} u_1 \\ u_2\end{bmatrix} = (C^TC)^{-1}C^Tv, \qquad v = C(C^TC)^{-1}\begin{bmatrix} u_1 \\ u_2\end{bmatrix}$$

by solving the respective block systems

$$\begin{bmatrix} I & A & B \\ A^T & & \\ B^T & & \end{bmatrix}\begin{bmatrix} t \\ u_1 \\ u_2\end{bmatrix} = \begin{bmatrix} v \\ 0 \\ 0\end{bmatrix}, \qquad \begin{bmatrix} I & A & B \\ A^T & & \\ B^T & & \end{bmatrix}\begin{bmatrix} v \\ t_1 \\ t_2\end{bmatrix} = \begin{bmatrix} 0 \\ -u_1 \\ -u_2\end{bmatrix}. \qquad (9.33)$$

Thus we can obtain the same types of Hessian approximations with two augmented system solves, except in B_2 they now correspond to both least-squares and least-norm solves instead of only projections.

9.4 Regularization

Even if $A(x^\star)$ has full column rank, $A(x)$ might have low column rank or small singular values away from the solution. If $A(x)$ is rank-deficient and $c(x)$ is not in the range of $A(x)^T$, then $y_\sigma(x)$ and $\phi_\sigma(x)$ are undefined. Even if $A(x)$ has full column rank but is close to rank-deficiency, the linear systems (9.13)–(9.15) and (9.18) are ill-conditioned, threatening inaccurate solutions and impeded convergence.

We modify ϕ_σ by changing the definition of the multiplier estimates in (9.1b) to solve a regularized shifted least-squares problem with regularization parameter $\delta > 0$:

$$\begin{aligned}\phi_\sigma(x;\delta) &:= f(x) - c(x)^Ty_\sigma(x;\delta) \qquad (9.34a)\\ y_\sigma(x;\delta) &:= \underset{y}{\operatorname{argmin}}\ \tfrac{1}{2}\|A(x)y - g(x)\|_2^2 + \sigma c(x)^Ty + \tfrac{1}{2}\delta^2\|y\|_2^2. \qquad (9.34b)\end{aligned}$$

This modification is similar to the exact penalty function of Di Pillo and Grippo (1986). The regularization term $\frac{1}{2}\delta^2\|y\|_2^2$ ensures that the multiplier estimate $y_\sigma(x;\delta)$ is always defined even when $A(x)$ is rank-deficient. The only computational change is that the $(2,2)$ block of the matrices in (9.13)–(9.15) and (9.18) is now $-\delta^2 I$.

Besides improving $\operatorname{cond}(\mathcal{K})$, $\delta > 0$ has the advantage of making $\mathcal{K}$ symmetric quasi-definite. Vanderbei (1995) shows that any symmetric permutation of such a matrix possesses an LDL^T factorization with L unit lower triangular and D diagonal indefinite. Result 2 of Gill, Saunders, and Shinnerl (1996) implies that the factorization is stable as long as δ is sufficiently far from zero. Various authors propose regularized matrices of this type to stabilize optimization methods in the presence of degeneracy. In particular, Wright (1998) accompanies his discussion with an update scheme for δ that guarantees fast asymptotic convergence.

We continue to assume that (NP) satisfies (A1b), but we now replace (A2b) by (A2a). For a given isolated local minimum $x^\star$ of (NP), σ sufficiently large, and δ sufficiently small, we define

$$x(\delta) := \arg\min_x \|x - x^\star\| \;\text{ such that } x \text{ is a local-min of } \phi_\sigma(x;\delta)$$

for use as an analytical tool in the upcoming discussion.

Note that for $\delta > 0$, we would not expect that $x(\delta) = x^\star$, but we want to ensure that $x(\delta) \to x^\star$ as $\delta \to 0$. Note that for x such that $y_\sigma(x)$ is defined,

$$\begin{aligned} y_\sigma(x;\delta) &= (A(x)^T A(x) + \delta^2 I)^{-1} A(x)^T A(x) y_\sigma(x) \\ &= y_\sigma(x) - \delta^2 (A(x)^T A(x) + \delta^2 I)^{-1} y_\sigma(x). \end{aligned}$$

Therefore for x such that $\phi_\sigma(x)$ is defined, we can write the regularized penalty function as a perturbation of the unregularized one:

$$\begin{aligned} \phi_\sigma(x;\delta) &= f(x) - c(x)^T y_\sigma(x;\delta) \\ &= f(x) - c(x)^T y_\sigma(x) + \delta^2 c(x)^T (A(x)^T A(x) + \delta^2 I)^{-1} y_\sigma(x) \\ &= \phi_\sigma(x) + \delta^2 P_\delta(x), \end{aligned} \tag{9.35}$$

where $P_\delta(x) = c(x)^T (A(x)^T A(x) + \delta^2 I)^{-1} y_\sigma(x)$. By (A1b), P_δ is bounded and has at least two continuous derivatives in a neighbourhood of $x^\star$.

Theorem 9.10 *Suppose (A1b) and (A2a) are satisfied, $x^\star$ is a second-order KKT point for (NP), and $\nabla^2 \phi_\sigma(x^\star) \succ 0$. Then there exists $\bar{\delta} > 0$ such that $x(\delta)$ is a $\mathcal{C}_1$ function for $0 \leqslant \delta < \bar{\delta}$. In particular, $\|x(\delta) - x^\star\| = O(\delta)$.*

Proof. The theorem follows from the Implicit Function Theorem (Ortega and Rheinboldt, 2000, Theorem 5.2.4) applied to $\nabla \phi_\sigma(x;\delta) = 0$. ☐

An option to recover $x^\star$ using $\phi_\sigma(x;\delta)$ is to minimize a sequence of problems defined by $x_{k+1} = \operatorname{argmin}_x \phi_\sigma(x;\delta_k)$ with $\delta_k \to 0$, using x_k to warm-start the next subproblem. However, we show that it is possible to decrease δ within a single problem, while retaining fast local convergence.

To keep results independent of the minimization algorithm being used, for a family of functions $\mathcal{F}$ we define $G : \mathcal{F} \times \mathbb{R}^n \to \mathbb{R}^n$ such that for $f \in \mathcal{F}$ and an iterate x, $G(f,x)$ computes an update direction. For example, if $\mathcal{F} = \mathcal{C}_2$, we can represent Newton's method with $G(f,x) = -H(x)^{-1} g(x)$, where $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$. Define $\nu(\delta)$ as a function such that for repeated applications, $\nu^k(\delta) \to 0$ as $k \to \infty$ at a chosen rate; for example, for a quadratic rate, we let $\nu(\delta) = \delta^2$.

Algorithm 8 describes how to adaptively update δ each iteration. In order to analyze it, we formalize the notions of rates of convergence using definitions equivalent to those of Ortega and Rheinboldt (2000, §9).

Algorithm 8 Minimization of the regularized penalty function $\phi_\sigma(x, \delta)$ with $\delta \to 0$

```
1: Choose $x_1$, $\delta_0$
2: for $k = 1, 2, \ldots$ do
3:     Set $\delta_k \leftarrow \max\{\min\{\|\nabla\phi_\sigma(x_k; \delta_{k-1})\|, \delta_{k-1}\}, \nu(\delta_{k-1})\}$        (9.36)
4:     $p_k \leftarrow G(\phi_\sigma(\cdot, \delta_k), x_k)$
5:     $x_{k+1} \leftarrow x_k + p_k$
6: end for
```

Definition 9.11 *We say that $x_k \to x^\star$ with order at least $\tau > 1$ if there exists $M > 0$ such that, for all sufficiently large k, $\|x_{k+1} - x^\star\| \leqslant M\|x_k - x^\star\|^\tau$. We say that $x_k \to x^\star$ with R-order at least $\tau > 1$ if there exists a sequence α_k such that, for all sufficiently large k,*

$$\|x_k - x^\star\| \leqslant \alpha_k, \qquad \alpha_k \to 0 \textit{ with order at least } \tau.$$

We first show that any minimization algorithm achieving a certain local rate of convergence can be regarded as *inexact Newton* (Dembo et al., 1982).

Lemma 9.12 *Let $f(x)$ be a $\mathcal{C}_2$ function with local minimum $x^\star$ and $H(x^\star) \succ 0$. Suppose we minimize f according to*

$$x_{k+1} = x_k + p_k, \qquad p_k = G(f, x_k), \tag{9.37}$$

such that $x_k \to x^\star$ with order at least $\tau \in (1, 2]$. Then in some neighborhood of $x^\star$, the update procedure $G(f, x)$ is equivalent to the inexact-Newton iteration

$$x_{k+1} \leftarrow x_k + p_k, \qquad H(x_k)p_k = -g(x_k) + r_k, \qquad \|r_k\| = O(\|g(x_k)\|^\tau). \tag{9.38}$$

Proof. There exists a neighborhood $N_N(x^\star)$ such that for any $x_0^N \in N_N(x^\star)$, the Newton update $x_{k+1}^N = x_k^N + p_k^N$ with $H(x_k^N)p_k^N = -g(x_k^N)$ is quadratically convergent:

$$\|x^\star - x_{k+1}^N\| \leqslant M_1\|x^\star - x_k^N\|^2, \qquad x_k \in N_N(x^\star).$$

Similarly let $N_G(x^\star)$ be the neighborhood where order τ convergence is obtained for (9.37) with constant M_2. Let $B_\epsilon(x^\star) = \{x \mid \|x^\star - x\| \leqslant \epsilon\}$. Choose $\epsilon \leqslant \min\{M_2^{-1}, 1\}$ such that $B_\epsilon(x^\star) \subseteq N_N \cap N_G$, and observe that if $x_0 \in B_\epsilon(x^\star)$, then $x_k \in B_\epsilon(x^\star)$ for all k because $\|x^\star - x_k\|$ is monotonically decreasing. By continuity of $H(x)$, there exists $M_3 > 0$ such that $\|H(x)\| \leqslant M_3$ for all $B_\epsilon(x^\star)$. Then for $x_k \in B_\epsilon(x^\star)$,

$$\begin{aligned}
\|r_k\| = \|H(x_k)p_k + g(x_k)\| &= \|H(x_k)(x_{k+1} - x_k - p_k^N)\| \\
&\leqslant \|H(x_k)\|\|x_{k+1} - x^\star + x^\star - x_{k+1}^N\| \\
&\leqslant M_3(\|x_{k+1} - x^\star\| + \|x_{k+1}^N - x^\star\|) \\
&\leqslant M_3(M_1 + M_2)\|x_k - x^\star\|^\tau,
\end{aligned}$$

Because $f \in \mathcal{C}_2$, there exists a constant M_4 such that $\|x_k - x^\star\| \leqslant M_4\|g(x_k)\|$ for $x_k \in N_G(x^\star) \cap N_N(x^\star)$. Therefore $\|r_k\| \leqslant M_4 M_3(M_1 + M_2)\|g(x_k)\|^\tau$, which is the inexact-Newton method, convergent with order τ. ∎

Note that Lemma 9.12 can be modified to accommodate any form of superlinear convergence, as long as $\|r_k\|$ converges at the same rate as $x_k \to x^\star$.

Theorem 9.13 *Suppose that (A1b) and (A2a) are satisfied, $x^\star$ is a second-order KKT point for (NP), $\nabla^2\phi_\sigma(x^\star) \succ 0$, and there exists $\bar{\delta}$ and an open set $B(x^\star)$ containing $x^\star$ such that for $\widetilde{x}_0 \in B(x^\star)$ and $\delta \leqslant \bar{\delta}$, the sequence defined by $\widetilde{x}_{k+1} = \widetilde{x}_k + G(\phi_\sigma(\cdot;\delta), \widetilde{x}_k)$ converges quadratically to $x(\delta)$:*

$$\|x(\delta) - \widetilde{x}_{k+1}\| \leqslant M_\delta \|x(\delta) - \widetilde{x}_k\|^2.$$

Further suppose that for $\delta \leqslant \bar{\delta}$, $M_\delta \leqslant M$ is uniformly bounded. Then there exists an open set, $B'(x^\star)$ that contains $x^\star$, and $\delta' > 0$ such that if $x \in B'(x^\star)$, $\delta \leqslant \delta'$ and $x_k \to x^\star$ for x_k defined by Algorithm 8 (with $\nu(\delta) = \delta^2$), then $x_k \to x^\star$ R-quadratically.

The proof is in Appendix D. Although there are many technical assumptions, the takeaway message is that we need only minimize $\phi_\sigma(\cdot;\delta_k)$ until $\|\nabla\phi_\sigma\| = O(\delta_k)$, because under typical smoothness assumptions we have that $\|x(\delta) - x^\star\| = O(\delta)$ for δ sufficiently small. Decreasing δ at the same rate as the local convergence rate of the method on a fixed problem should not perturb $\phi_\sigma(x;\delta)$ too much, therefore allowing for significant progress on the perturbed problem in few steps. Within the basin of convergence and for a fixed $\delta > 0$, an optimization method would achieve the same local convergence rate that it would have with $\delta = 0$ fixed.

Theorem 9.13 can be generalized to superlinear rates of convergence using a similar proof. As long as $\nu(\cdot)$ drives $\delta \to 0$ as fast as the underlying algorithm would locally converge for fixed δ, local convergence of the entire regularized algorithm is unchanged.

9.5 Inexact evaluation of the penalty function

We discuss the effects of solving (9.19) approximately, and thus evaluating ϕ_σ and its derivatives inexactly. Various optimization solvers can utilize inexact function values and derivatives while ensuring global convergence and certain local convergence rates, provided the user can compute relevant quantities to a prescribed accuracy. For example, Conn et al. (2000, §8–9) describe conditions on the inexactness of model and gradient evaluations to ensure convergence, and Heinkenschloss and Ridzal (2014) describe an inexact trust-region SQP solver for PDE-constrained optimization using inexact function values and gradients. We focus on inexactness within trust-region methods for optimizing ϕ_σ.

The accuracy and computational cost in the evaluation of ϕ_σ and its derivatives depends on the accuracy of the solves of (9.19). If the cost to solve (9.19) depends on solution accuracy (e.g., with iterative linear solvers), it is advantageous to consider optimization solvers that use inexact computations, especially for large-scale problems.

Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a compact set. In this section, we use $\widetilde{\phi}_\sigma(x)$, $\nabla\widetilde{\phi}_\sigma(x)$, etc. to distinguish the inexact quantities from their exact counterparts. We also drop the arguments from operators as in Section 9.2. We consider three quantities that are computed inexactly: g_σ, ϕ_σ and $\nabla\phi_\sigma$. For given error tolerances η_i, we are interested in exploring termination criteria for solving (9.19) to ensure that the following conditions hold for all $x \in \mathcal{S}$:

$$\left|\phi_\sigma - \widetilde{\phi}_\sigma\right| \leqslant M\eta_1, \tag{9.39a}$$

$$\|\nabla\phi_\sigma - \nabla\widetilde{\phi}_\sigma\| \leqslant M\eta_2, \tag{9.39b}$$

$$\|g_\sigma - \widetilde{g}_\sigma\| \leqslant M\eta_3, \tag{9.39c}$$

where $M > 0$ is some fixed constant (which may or may not be known). Kouri, Heinkenschloss, Ridzal, and van Bloemen Waanders (2014) give a trust-region method using inexact objective value and gradient information that guarantees global convergence provided (9.39a)–(9.39b) hold without requiring that M be known a priori. We may compare this to the conditions of Conn et al. (2000, §8.4, §10.6), which require more stringent conditions on (9.39a)–(9.39b). They require that $\eta_2 = \|\nabla\widetilde{\phi}_\sigma\|$ and that M be known and fixed according to parameters in the trust-region method.

This leads us to the following proposition, which allows us to bound the residuals of (9.13) and (9.15) to ensure (9.39).

Proposition 9.14 *Let $\mathcal{S}$ be a compact set, and suppose that $\sigma_{\min}(A(x)) \geqslant \lambda > 0$ for all $x \in \mathcal{S}$. Then for $x \in \mathcal{S}$, if*

$$\|r_1\| = \left\|\mathcal{K}\begin{bmatrix}\widetilde{g}_\sigma\\ \widetilde{y}_\sigma\end{bmatrix} - \begin{bmatrix} g\\ \sigma c\end{bmatrix}\right\| \leqslant \min\{1, \|c\|^{-1}\}\cdot\min\{\eta_1,\eta_3\}, \tag{9.40}$$

then (9.39a) and (9.39c) hold for some constant M. Also, if

$$\|r_1\| \leqslant \eta_2 \text{ and } \|r_2\| = \left\|\mathcal{K}\begin{bmatrix}\widetilde{v}\\ \widetilde{w}\end{bmatrix} - \begin{bmatrix}0\\ c\end{bmatrix}\right\| \leqslant \min\{1,\eta_2\}, \tag{9.41}$$

then (9.39b) holds for some (perhaps different) constant M.

Proof. Because $\mathcal{S}$ is compact and $\lambda > 0$, there exists $\bar{\lambda} > 0$ such that $\|\mathcal{K}\|$, $\|\mathcal{K}^{-1}\| \leqslant \bar{\lambda}$ for all $x \in \mathcal{S}$. Thus, (9.39c) follows directly from (9.15) and (9.40) with $M = \bar{\lambda}$. Similarly,

$$\left|\phi_\sigma - \widetilde{\phi}_\sigma\right| = \left|c^T\left(y_\sigma - \widetilde{y_\sigma}\right)\right| \leqslant \|c\|\,\|y_\sigma - \widetilde{y_\sigma}\| \leqslant \bar{\lambda}\eta_1,$$

and (9.39a) holds with $M = \bar{\lambda}$. We apply a similar analysis to ensure that (9.39b) holds. Define the vector $h \in \mathbb{R}^m$ such that $h_i = \|H_i\|$. Define v, w as the solutions to (9.41) for $r_2 = 0$, so that from (9.41) we have

$$\begin{aligned}
\|\nabla\phi_\sigma - \nabla\widetilde{\phi}_\sigma\| &\leqslant \|g_\sigma - \widetilde{g}_\sigma\| + \|Y_\sigma c - \widetilde{Y}_\sigma c\| \\
&\leqslant \bar{\lambda}\eta_2 + \|(H_\sigma - \sigma I)v - S_\sigma^T w - (\widetilde{H}_\sigma - \sigma I)\widetilde{v} + \widetilde{S}_\sigma^T\widetilde{w}\| \\
&\leqslant \bar{\lambda}\eta_2 + \sigma\|v - \widetilde{v}\| + \|H_\sigma v - \widetilde{H}_\sigma\widetilde{v}\| + \|S_\sigma^T w - \widetilde{S}_\sigma^T\widetilde{w}\| \\
&\leqslant \left(\bar{\lambda} + \sigma\bar{\lambda}\right)\eta_2 + \|H_\sigma(v - \widetilde{v}) + (H_\sigma - \widetilde{H}_\sigma)\widetilde{v}\| \\
&\quad + \|S_\sigma^T(w - \widetilde{w}) + (S_\sigma - \widetilde{S}_\sigma)^T\widetilde{w}\| \\
&\leqslant \left(\bar{\lambda} + \sigma\bar{\lambda}\right)\eta_2 + \|H_\sigma\|\|v - \widetilde{v}\| + \|\sum_{i=1}^m\left((y_\sigma)_i - (\widetilde{y}_\sigma)_i\right)H_i\|\|\widetilde{v}\| \\
&\quad + \|S_\sigma\|\|w - \widetilde{w}\| + \|\sum_{i=1}^m \widetilde{w}_i H_i\|\|g_\sigma - \widetilde{g}_\sigma\| \\
&\leqslant \left(\bar{\lambda} + \sigma\bar{\lambda} + \|H_\sigma\|\bar{\lambda} + \bar{\lambda}\|\widetilde{v}\|\|h\| + \|S_\sigma\|\bar{\lambda} + \|\widetilde{w}\|\|h\|\bar{\lambda}\right)\eta_2.
\end{aligned}$$

Note that $\|H_\sigma\|$, $\|h\|$, $\|S_\sigma\|$, $\|\widetilde{w}\|$ and $\|\widetilde{v}\|$ are bounded uniformly in $\mathcal{S}$. □

In the absence of additional information, using (9.39) with unknown M may be the only way to take advantage of inexact computations, because computing exact constants (such as the norms $\mathcal{K}$ or the various operators above) is not practical. In some cases the bounds (9.39) are relative, e.g., $\eta_2 = \min\{\|\nabla\widetilde{\phi}_\sigma\|, \Delta\}$ for a certain $\Delta > 0$. It may then be necessary to compute $\|\nabla\widetilde{\phi}_\sigma\|$ and refine the solutions of (9.15) and (9.13) until they satisfy (9.40)–(9.41). However, given the expense of applying these operators, it may be more practical to use a nominal relative tolerance, as in the numerical experiments of Chapter 11.

We include a (trivial) improvement to Proposition 9.14 that satisfies (9.39a) and (9.39c) with $M = 1$, given additional spectral information about A. If we solve (9.15) by solving

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}\begin{bmatrix} \Delta g_\sigma \\ y_\sigma \end{bmatrix} = \begin{bmatrix} 0 \\ \sigma c - A^T g \end{bmatrix}, \qquad g_\sigma = g + \Delta g_\sigma,$$

we can use LNLQ (Chapter 5), a Krylov subspace method for such systems, which ensures that $\|\Delta g_\sigma - \Delta\widetilde{g}_\sigma^{(j)}\|$ and $\|y_\sigma - \widetilde{y}_\sigma^{(j)}\|$ are monotonic, where $\Delta\widetilde{g}_\sigma^{(j)}$, $\widetilde{y}_\sigma^{(j)}$ are the jth LNLQ iterates. Given $\lambda > 0$ such that $\sigma_{\min}(A) \geqslant \lambda$, LNLQ can compute cheap upper-bounds on $\|\Delta g_\sigma - \Delta\widetilde{g}_\sigma^{(j)}\|$ and $\|y_\sigma - \widetilde{y}_\sigma^{(j)}\|$, allowing us to terminate the solve when $\|\Delta g_\sigma - \widetilde{\Delta g_\sigma}\| \leqslant \eta_2\|\widetilde{\Delta g_\sigma} + g\| = \eta_2\|\widetilde{g}_\sigma\|$ and $\|y_\sigma - \widetilde{y}_\sigma\| \leqslant \min\{1, \|c\|^{-1}\}\eta_1$. Typically, termination criteria for the optimization solver will include a condition that $\|g_\sigma\| \leqslant \epsilon_d$ to determine approximate local minimizers to (NP). For such cases, we can instead require that $\|\widetilde{g}_\sigma\| \leqslant \frac{1}{1+\eta_2}\epsilon_d$, because then

$$\|g_\sigma\| \leqslant \|g_\sigma - \widetilde{g}_\sigma\| + \|\widetilde{g}_\sigma\| \leqslant (1+\eta_2)\|\widetilde{g}_\sigma\| \leqslant \epsilon_d.$$

Similarly, we have

$$\left|\phi_\sigma - \widetilde{\phi}_\sigma\right| \leqslant \|c\|\|y_\sigma - \widetilde{y}_\sigma\| \leqslant \eta_1,$$

which now satisfies (9.39a) with $M = 1$.

Although finding suitable λ may be difficult in general, it is trivially available in some cases because of the way $\mathcal{K}$ is preconditioned (for an example, see Chapter 11). However, a complication is that if LNLQ is used with a right-preconditioner $\mathcal{P} \approx A^TA$, then $\|y_\sigma - \widetilde{y}_\sigma\|_\mathcal{P}$ is monotonic and LNLQ provides bounds on the preconditioned norm instead of the Euclidean norm. If $\|\mathcal{P}^{-1}\|$ can be bounded, then the bound $\|y_\sigma - \widetilde{y}_\sigma\| \leqslant \|y_\sigma - \widetilde{y}_\sigma\|_\mathcal{P}\|\mathcal{P}^{-1}\|$ can be used.

Chapter 10

The inequality-constrained case

We now consider the general problem (NP) under assumptions (A1b) and (A2b) in particular. Recall that the penalty function (defined in (8.1)) is

$$\begin{aligned}\phi_\sigma(x) &:= f(x) - c(x)^T y_\sigma(x)\\ y_\sigma(x) &:= \operatorname{argmin}_y \tfrac{1}{2}\|A(x)y - g(x)\|^2_{Q(x)} + \sigma c(x)^T y,\end{aligned}$$

Notice that the multiplier estimate (9.1b) for equality-constrained problems differs from the above by scaling the least-squares problem with $Q(x)$. The diagonal entries of the scaling matrix $Q(x)$ (defined in (8.5)) are smooth approximations of the complementarity function $q(x) \approx \min\{x-\ell, u-x\}$; see Chen (2000). Fig. 10.1 plots $q(x)$ with finite ℓ and u.

The definition of $y_\sigma(x)$ (8.2) can be interpreted as a smooth approximation of the complementarity conditions in the first-order KKT conditions (10.2d)–(10.2f) below. The role of $Q(x)$ is therefore to ensure that the gradient of the Lagrangian is zero only at indices corresponding to inactive bounds. Several researchers have provided approaches for dealing with complementarity constraints in a similar fashion by introducing nonlinear constraints—see for example Anitescu (2000) and Leyffer (2006).

For a scalar x, the derivative of $q(x)$ is

$$q'(x) = \begin{cases} 0 & \text{if } \ell = -\infty \text{ and } u = \infty,\\ 1 - \frac{2}{\omega}\left(2x + u - \ell - \frac{\omega}{2}\right) & \text{else if } |u + \ell - 2x| \leqslant \frac{\omega}{2},\\ 1 & \text{else if } x - \ell < u - x,\\ -1 & \text{else if } x - \ell > u - x.\end{cases} \tag{10.1}$$

The choice of $q(x)$ is not unique because any nonnegative smooth concave function that is zero at $x_j \in \{\ell_j, u_j\}$ works in our framework. For example, we could use a smooth approximation of $\min\{x_j - \ell_j, u_j - x_j, 1\}$ (this potentially can avoid numerical issues that can arise if x is far from its bounds).

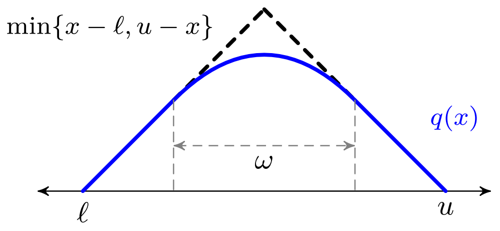


Figure 10.1: Plot of $q(x)$, a smooth approximation of $\min\{x-\ell, u-x\}$.

10.1 Properties of the penalty function

We show how the penalty function $\phi_\sigma(x)$ naturally expresses the optimality conditions of (NP). We also give explicit expressions for the threshold value of the penalty parameter σ. The gradient and Hessian of the penalty are given in (9.2).

10.1.1 Optimality conditions

The penalty function ϕ_σ is closely related to the (partial) Lagrangian $L(x, y)$ associated with (NP) that dualizes only the equality constraints. To make this connection clear, we define the Karush-Kuhn-Tucker (KKT) optimality conditions for (NP) in terms of the optimality conditions of (PP). From the definition of ϕ_σ and y_σ and the derivatives (9.2), the following definitions are equivalent to the KKT conditions.

Definition 10.1 (First-order KKT point) *A point $(x^\star, z^\star)$ is a first-order KKT point of* (NP) *if for any $\sigma \geqslant 0$ the following hold:*

$$\ell \leqslant x^\star \leqslant u \tag{10.2a}$$

$$c(x^\star) = 0, \tag{10.2b}$$

$$\nabla\phi_\sigma(x^\star) = z^\star, \tag{10.2c}$$

$$z_j^\star = 0, \qquad \text{if } j \notin \mathcal{A}(x^\star), \tag{10.2d}$$

$$z_j^\star \geqslant 0, \qquad \text{if } x_j^\star = \ell_j, \tag{10.2e}$$

$$z_j^\star \leqslant 0, \qquad \text{if } x_j^\star = u_j. \tag{10.2f}$$

Then $y^\star := y_\sigma(x^\star)$ is the Lagrange multiplier of (NP) *associated with $x^\star$. Note that by (A3), inequalities* (10.2e) *and* (10.2f) *are strict.*

Definition 10.2 (Second-order KKT point) *The first-order KKT point $(x^\star, z^\star)$ satisfies the second-order necessary KKT condition for* (NP) *if for any $\sigma \geqslant 0$,*

$$p^T\nabla^2\phi_\sigma(x^\star)p \geqslant 0 \quad \text{for all } p \in \mathcal{C}(x^\star, z^\star). \tag{10.3}$$

The condition is sufficient if the inequality is strict.

Remark 10.3 *If* (10.2b) *is omitted, Definition 10.1 corresponds to first-order KKT points of* (PP)*. Similarly, replacing $\mathcal{C}(x^\star, z^\star)$ by $\mathcal{C}_\phi(x^\star, z^\star)$ in Definition 10.2 corresponds to second-order KKT points of* (PP)*.*

The second-order KKT condition says that at a second-order KKT point of (PP), ϕ_σ has nonnegative curvature only along directions in the critical cone $\mathcal{C}_\phi(x^\star, z^\star)$. However, at $x^\star$, we show that increasing σ will increase curvature only along the normal cone of equality constraints. We derive a threshold value for σ so that ϕ_σ has nonnegative curvature even when $A(x^\star)^T p \neq 0$, as well as a condition on σ that ensures stationary points of (PP) are primal feasible. For a given first- or second-order KKT triple $(x^\star, y^\star, z^\star)$ of (NP), we define

$$\sigma^\star := \tfrac{1}{2}\lambda^+_{\max}\left(PQ(x^\star)^{1/2}H_L(x^\star, y^\star)Q(x^\star)^{1/2}P\right), \tag{10.4}$$

where $P := P_{Q^{1/2}(x^\star)A(x^\star)}$.

Lemma 10.4 *If $c(x) \in \operatorname{range}\big(A(x)^T Q(x)\big)$, then $y_\sigma(x)$ satisfies*

$$A(x)^T Q(x) A(x) y_\sigma(x) = A(x)^T Q(x) g(x) - \sigma c(x). \tag{10.5}$$

Furthermore, if $Q(x)A(x)$ has full rank, then

$$\begin{aligned} &A(x)^T Q(x) A(x) Y_\sigma(x)^T \\ &\qquad = A(x)^T [Q(x) H_\sigma(x) - \sigma I + R(x, g_\sigma(x))] + S(x, Q(x) g_\sigma(x)). \end{aligned} \tag{10.6}$$

Proof. For any x, the necessary and sufficient optimality conditions for (8.2) give (10.5). For brevity, let everything be evaluated at the same point x, and drop the argument x from all operators. By differentiating both sides of (10.5), we obtain

$$S(QAy_\sigma) + A^T\big[R(Ay_\sigma) + QT(y_\sigma) + QAY_\sigma^T\big] = S(Qg) + A^T[R(g) + QH - \sigma I].$$

By rearranging the above, and using definitions (8.8), we obtain (10.6). □

Theorem 10.5 *Suppose $(\bar{x}, \bar{z})$ is a first-order KKT point for (PP) with $Q(\bar{x})^{1/2} A(\bar{x})$ full-rank, and let $(x^\star, y^\star, z^\star)$ be a second-order necessary KKT point for (NP). Then*

$$\sigma > \|A(\bar{x})^T Q(\bar{x}) Y_\sigma(\bar{x})\| \quad \Longrightarrow \quad \bar{x} \text{ satisfies Definition 10.1}; \tag{10.7a}$$

$$p^T \nabla^2 \phi_\sigma(x^\star) p \geqslant 0 \quad \Longleftrightarrow \quad \sigma \geqslant \sigma^\star, \quad \forall p \in \mathcal{C}_\phi(x^\star, z^\star). \tag{10.7b}$$

If $x^\star$ is second-order sufficient, then the inequalities in (10.7b) hold strictly.

Proof. Proof of (10.7a): Because $\bar{x}$ is a first-order KKT point for (PP), we need only show that $c(\bar{x}) = 0$. By the complementarity conditions we have (10.2d)–(10.2f), $Q(\bar{x}) \nabla \phi_\sigma(\bar{x}) = 0$, so that

$$Q(\bar{x}) g(\bar{x}) = Q(\bar{x}) A(\bar{x}) y_\sigma(\bar{x}) + Q(\bar{x}) Y_\sigma(\bar{x}) c(\bar{x}).$$

Substituting (10.5) evaluated at $\bar{x}$ into this equation yields, after simplifying,

$$A(\bar{x})^T Q(\bar{x}) Y_\sigma(\bar{x}) c(\bar{x}) = \sigma c(\bar{x}).$$

Taking norms of both sides and using the triangle inequality gives the inequality $\sigma \|c(\bar{x})\| \leqslant \|A(\bar{x})^T Q(\bar{x}) Y_\sigma(\bar{x})\| \, \|c(\bar{x})\|$, which implies that $c(\bar{x}) = 0$.

Proof of (10.7b): Because $x^\star$ satisfies first-order conditions (10.2), we have $y^\star = y_\sigma(x)$ and $Q(x^\star) g_\sigma(x^\star) = 0$, independently of σ. We drop the arguments from operators that take x as input, and assume that they are all evaluated at $x^\star$. It follows from (10.6), $H_L(x^\star, y^\star) = H_\sigma$, and the definition of the projector $P := P_{Q^{1/2}(x^\star) A(x^\star)}$ that

$$Q^{1/2} A Y_\sigma^T Q^{1/2} = P(Q^{1/2} H_L(x^\star, y^\star) Q^{1/2} - \sigma I + R(g_\sigma) Q^{1/2}). \tag{10.8}$$

Observe that if $p \in \mathcal{C}_\phi(x^\star, z^\star)$, then $p = Q^{1/2} \bar{p}$ for some $\bar{p} \in \mathcal{C}_\phi(x^\star, z^\star)$. Because $Q^{1/2} g_\sigma = 0$, we have $R(g_\sigma) p = 0$. Therefore using (9.2b), (10.8), and the relation $P + \bar{P} = I$, we have

$$\begin{aligned} p^T \nabla^2 \phi_\sigma(x^\star) p \geqslant 0 &\iff \bar{p}^T Q^{1/2} \left(H_\sigma - A Y_\sigma^T - Y_\sigma A^T\right) Q^{1/2} \bar{p} \geqslant 0 \\ &\iff \bar{p}^T \left(\bar{P} Q^{1/2} H_\sigma Q^{1/2} \bar{P} - P Q^{1/2} H_\sigma Q^{1/2} P + 2\sigma P\right) \bar{p} \geqslant 0. \end{aligned}$$

Now, because $\bar{P}\bar{p} \in \ker(A^T Q^{1/2})$ implies that $Q^{1/2}\bar{P}\bar{p} \in \mathcal{C}(x^\star, z^\star)$, the first term above is nonnegative according to Definition 10.2. It follows that σ must be sufficiently large that $2\sigma P - PQ^{1/2}H_L(x^\star, y^\star)Q^{1/2}P \succeq 0$, which is equivalent to $\sigma \geqslant \sigma^\star$. ☐

As in Theorem 9.4, (10.7b) shows that if $x^\star$ is a second-order KKT point of (NP), there exists a threshold value $\sigma^\star$ such that $x^\star$ will also be a second-order KKT point of (PP). However, this does not preclude the possibility that there exist minimizers of the penalty function—for any value of σ—that are not minimizers of (NP). However, these are rarely encountered in practice. Further, we can add a quadratic penalty term which under certain conditions will ensure that KKT points of (PP) are feasible for (NP) as in Section 9.1.3

10.2 Evaluating the penalty function

As in Section 9.2, we show how to evaluate ϕ_σ ad its derivatives by solving structured linear systems. In this case, we show that this linear system may be either symmetric or unsymmetric, and discuss the tradeoffs between both approaches. In either case, if direct methods are to be used, only a single factorization that defines the solution (8.2) is required for all products.

We drop the arguments on various functions and assume they are all evaluated at a point x for some parameter σ. We express (10.6) using the shorthand notation

$$A^T QAY_\sigma^T = A^T(QH_\sigma - \sigma I + R_\sigma) + S_\sigma. \tag{10.9}$$

We first describe how to compute products $Y_\sigma u$ and $Y_\sigma^T v$, then how to put those pieces together to evaluate the penalty function and its derivatives.

Every quantity of interest can be computed by solving a symmetric or unsymmetric linear system and combining the solution with the derivatives of the problem data. Typically it is preferable to solve symmetric rather than unsymmetric linear systems; however, we find that additional Jacobian products are needed when the symmetric linear system is used. The additional cost may be negligible, but this matter becomes application dependent. We therefore present both options, beginning with the symmetric case.

There are many ways to construct the right-hand sides of the linear systems presented below. One consideration is that inversions with the diagonal matrix $Q^{1/2}$ should be avoided—even though the diagonal of Q will be assumed strictly positive because of the use of an interior method (see Chapter 11), numerical difficulties may arise near the boundary of the feasible set if $Q^{1/2}$ contains small entries and is inverted.

10.2.1 Computing the product $\mathbf{Y_\sigma u}$

It follows from (10.9) that for a given m-vector u,

$$Y_\sigma u = (H_\sigma Q - \sigma I + R_\sigma)A(A^T QA)^{-1}u + S_\sigma^T(A^T QA)^{-1}u.$$

Let $w = -(A^T QA)^{-1}u$ and $v = -Q^{1/2}Aw$, so that v and w are the solution of the symmetric linear system

$$\begin{bmatrix} I & Q^{1/2}A \\ A^T Q^{1/2} & \end{bmatrix}\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ u \end{bmatrix}. \tag{10.10}$$

Then $Y_\sigma u = H_\sigma Q^{1/2}v + (\sigma I - R_\sigma)Aw - S_\sigma^T w$. Algorithm 9 formalizes this process.

Algorithm 9 Computing the matrix-vector product $Y_\sigma u$

```
1: $(v, w) \leftarrow$ solution of (10.10)
2: return $H_\sigma Q^{1/2}v + (\sigma I - R_\sigma)Aw - S_\sigma^T w$
```

10.2.2 Computing the product $\mathbf{Y}_\sigma^\mathbf{T}\mathbf{v}$

Again from (10.9), multiplying both sides on the right by v gives

$$Y_\sigma^T v = (A^TQA)^{-1}A^T(QH_\sigma - \sigma I + R_\sigma)v + (A^TQA)^{-1}S_\sigma v.$$

The product $u = Y_\sigma^T v$ is the solution of the system

$$\begin{bmatrix} I & Q^{1/2}A \\ A^TQ^{1/2} & \end{bmatrix}\begin{bmatrix} r \\ u \end{bmatrix} = \begin{bmatrix} Q^{1/2}H_\sigma v \\ A^T(\sigma I - R_\sigma)v - S_\sigma v \end{bmatrix}. \tag{10.11}$$

Algorithm 10 formalizes the process.

Algorithm 10 Computing the matrix-vector product $Y_\sigma^T v$

```
1: Evaluate $Q^{1/2}H_\sigma v$ and $A^T(\sigma I - R_\sigma)v - S_\sigma v$
2: $(r, u) \leftarrow$ solution of (10.11)
3: return $u$
```

10.2.3 Unsymmetric linear system

We briefly comment on how to use unsymmetric systems in place of (10.10) and (10.11). We can compute products of the form $Y_\sigma u = (H_\sigma - \sigma I + R_\sigma)\bar{v} - S_\sigma^T w$ (where $w = -(A^TQA)^{-1}u$ and $\bar{v} = -Aw$), and products $u = Y_\sigma^T v$ by solving the respective linear systems:

$$\begin{bmatrix} I & A \\ A^TQ & \end{bmatrix}\begin{bmatrix} \bar{v} \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ u \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} I & QA \\ A^T & \end{bmatrix}\begin{bmatrix} \bar{r} \\ u \end{bmatrix} = \begin{bmatrix} (QH_\sigma - \sigma I + R_\sigma)v \\ -S_\sigma v \end{bmatrix}. \tag{10.12}$$

Algorithms 9 and 10 can then be appropriately modified to use the above linear systems.

10.2.4 Computing multipliers and first derivatives

The multiplier estimates y_σ and Lagrangian gradient can be obtained from one of the following linear systems:

$$\begin{bmatrix} I & Q^{1/2}A \\ A^TQ^{1/2} & \end{bmatrix}\begin{bmatrix} d \\ y_\sigma \end{bmatrix} = \begin{bmatrix} Q^{1/2}g \\ \sigma c \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} I & A \\ A^TQ & \end{bmatrix}\begin{bmatrix} g_\sigma \\ y_\sigma \end{bmatrix} = \begin{bmatrix} g \\ \sigma c \end{bmatrix}. \tag{10.13}$$

Observe that in the unsymmetric case we obtain g_σ immediately, and otherwise $d = Q^{1/2}g_\sigma$. As noted earlier, computing $g_\sigma \leftarrow Q^{-1/2}d$ may be inaccurate compared to $g_\sigma \leftarrow g - Ay_\sigma$, which requires an additional Jacobian product.

The penalty gradient $\nabla\phi_\sigma = g_\sigma - Y_\sigma c$ can then be computed using g_σ (as described above) and computing $Y_\sigma c$ via Algorithm 9 or its unsymmetric variant.

10.2.5 Computing second derivatives

We approximate $\nabla^2\phi_\sigma$ from (9.2b) using the same approaches as in (9.17):

$$\begin{aligned}
\nabla^2\phi_\sigma \approx B_1 &:= H_\sigma - AY_\sigma^T - Y_\sigma A^T && (10.14a)\\
&= H_\sigma - \widetilde{P}(QH_\sigma + R_\sigma - \sigma I) - (H_\sigma Q + R_\sigma - \sigma I)\widetilde{P}\\
&\quad - A(A^TQA)^{-1}S_\sigma - S_\sigma^T(A^TQA)^{-1}A\\
\approx B_2 &:= H_\sigma - \widetilde{P}(QH_\sigma + R_\sigma - \sigma I) - (H_\sigma Q + R_\sigma - \sigma I)\widetilde{P}, && (10.14b)
\end{aligned}$$

where $\widetilde{P} = A(A^TQA)^{-1}A$. The first approximation drops the third derivative term $\nabla[Y_\sigma c]$ in (9.2b), while the second approximation drops the term $S_\sigma(x, Qg_\sigma)$ because these terms are zero at the solution. Therefore, B_1 and B_2 can be interpreted as Gauss-Newton approximations of $\nabla^2\phi_\sigma$. Using similar arguments to those made by Fletcher (1973a, Theorem 2), we expect these approximations to result in quadratic convergence when $f, c \in \mathcal{C}_3$, and superlinear convergence when $f, c \in \mathcal{C}_2$.

Computing products with B_1 only requires products with Y_σ and Y_σ^T, which can be handled by Algorithms 9 and 10. To compute a product $\widetilde{P}u$, we can solve

$$\begin{bmatrix} I & Q^{1/2}A \\ A^TQ^{1/2} & \end{bmatrix}\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} 0 \\ A^Tu \end{bmatrix} \quad\text{or}\quad \begin{bmatrix} I & A \\ A^TQ & \end{bmatrix}\begin{bmatrix} \bar{p} \\ q \end{bmatrix} = \begin{bmatrix} u \\ 0 \end{bmatrix}, \qquad \widetilde{P}u = -Aq. \tag{10.15}$$

As before, using the unsymmetric system avoids an additional Jacobian product, which may be negligible compared to solving an unsymmetric system.

10.2.6 Solving the augmented linear system

We comment on various approaches for solving the necessary linear systems

$$\mathcal{K}\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} w \\ z \end{bmatrix}, \qquad\text{where}\qquad \mathcal{K} = \begin{bmatrix} I & Q^{1/2}A \\ A^TQ^{1/2} & \end{bmatrix} \quad\text{or}\quad \begin{bmatrix} I & A \\ A^TQ & \end{bmatrix}. \tag{10.16}$$

This is the most computationally intensive step in the penalty approach. Note that when direct methods are used, a single factorization is needed to evaluate ϕ_σ and its (approximate) derivatives.

Appendix C describe several approaches for solving the symmetric system (using both direct and iterative methods). For unsymmetric systems, any sparse factorization of $\mathcal{K}$ may be used; also, we could factorize $Q^{1/2}A$ with a Q-less QR factorization and use the (refined) semi-normal equations (Björck and Paige, 1994) as in the symmetric case (as long as multiplications with $Q^{-1/2}$ are avoided).

If iterative methods are used, the unsymmetric system requires unsymmetric iterative methods such as GMRES (Saad and Schultz, 1986), SPMR (Estrin and Greif, 2018), or QMR (Freund and Nachtigal, 1991), where the choice of method depends on considerations such as short- vs. long-recurrence, available preconditioners, or robustness. Note that preconditioners approximating $\mathcal{P} \approx A^T Q A$ apply to both the symmetric and unsymmetric systems; however, unsymmetric solvers may allow inexact preconditioner solves, while short-recurrence symmetric solvers may not.

If optimization solvers that accept inexact function and derivative evaluations are used (e.g., Conn et al. (2000, §8–9) or Heinkenschloss and Ridzal (2014)), the results of Section 9.5 apply here as well; that is, bounding the residual norm of the linear systems is sufficient to bound the function and derivative evaluation error up to a constant (under mild assumptions). This is useful for applications where solving the linear system exactly every iteration is prohibitively expensive. Further, when the symmetric system is used, it is possible to use methods that upper bound the solution error (e.g., CRAIG Arioli (2013) or LNLQ (Chapter 5)) when an underestimate of the smallest singular value of the preconditioned Jacobian is available.

Chapter 11

Practical considerations and numerical experiments

We discuss some matters related to the use of ϕ_σ in practice. In principle, nearly any smooth unconstrained solver can be used to find a local minimum of ϕ_σ because it has at least one continuous derivative, and a continuous Hessian approximation if (A1a) is satisfied. However, the structure of ϕ_σ lends itself more readily to certain optimization methods than to others, especially when the goal of creating a factorization-free solver is kept in mind.

Fletcher (1973a) originally envisioned a Newton-type procedure

$$x_{k+1} \leftarrow x_k - \alpha_k B_i^{-1}(x_k)\nabla\phi_\sigma(x_k), \qquad i = 1 \text{ or } 2,$$

where B_1, B_2 are the Hessian approximations from (9.17a)–(9.17b) and $\alpha_k > 0$ is a step size. Fletcher (1973a, Theorem 2) further proved that superlinear convergence is achieved, or quadratic convergence if the second derivatives of f and c are Lipschitz continuous. However, for large problems it is expensive to compute B_i explicitly and solve the dense system $B_i s_k = -\nabla\phi_\sigma(x_k)$.

We instead propose using a Steihaug (1983) Newton-CG type trust-region solver to minimize ϕ_σ. First, trust-region methods are preferable to linesearch methods (Nocedal and Wright, 2006, §3–4) for objectives with expensive evaluations; it is costly to evaluate ϕ_σ repeatedly to determine a step-size every iteration as this requires solving a linear system. Further, $\nabla^2\phi_\sigma$ is often indefinite and trust-region methods can take advantage of directions of negative curvature. Computing B_i explicitly is not practical, but products are reasonable as they only require solving two linear systems with the same matrix, thus motivating the use of a Newton-CG type trust-region solver. In particular, solvers such as TRON (Lin and Moré, 1999b) and KNITRO (Byrd, Nocedal, and Waltz, 2006) are suitable for minimizing ϕ_σ. KNITRO has the additional advantage of handling explicit linear constraints.

Further, we recommend interior solvers rather than exterior or active-set methods. For $\phi_\sigma(x)$ to be defined, we require that $Q(x) \succeq 0$ (thus disqualifying exterior point methods) and that $Q(x)^{1/2}A(x)$ has full column-rank (so that at most $n - m$ components of x can equal a bound). Even if (A2b) is satisfied, an active-set method may choose a poor active set that causes $\phi_\sigma(x)$ to be undefined (or it may have too many active bounds). On the other hand, interior methods ensure that $Q(x) \succ 0$ and avoid this issue (at least until x converges and approaches the bounds).

We have not yet addressed choosing σ. Although we can provide an a posteriori threshold value for $\sigma^\star$, it is difficult to know this threshold ahead of time. Mukai and Polak (1975) give a scheme for updating ρ with $\phi_{\sigma,\rho}$ and $\sigma = 0$; however, they were using a Newton-like scheme that required a solve with $B_1(x)$. Further, $\sigma^\star$ ensures only *local* convexity, and that a local minimizer of (NP) is a local minimizer of ϕ_σ—but as with other penalty functions,

ϕ_σ may be unbounded below in general for any σ. A heuristic that we employ is to ensure that the primal and dual feasibility, $\|c(x)\|$ and $\|g_L(x, y_\sigma(x))\|$, are within some factor of each other (e.g., 100) to encourage them to decrease at approximately the same rate. If primal feasibility decreases too quickly and small steps are taken, it is indicative of σ being too large, and similarly if primal feasibility is too large then σ should be increased; this can be done with a multiplicative factor or by estimating $\|P_A(x)H_\sigma(x)P_A(x)\|$ via the power method. Although this heuristic is often effective in our experience, in situations where the penalty function begins moving toward negative infinity, we require a different recovery strategy, which is the subject of future work.

In practice, regularization (Section 9.4) is used only if necessary. For well-behaved problems, using $\delta = 0$ typically requires fewer outer iterations than using $\delta > 0$. However, when convergence is slow and/or the Jacobians are ill-conditioned, initializing with $\delta > 0$ is often vital and can improve performance significantly.

11.1 Numerical experiments

We investigate the performance of Fletcher's penalty on several PDE-constrained optimization problems and some standard test problems. For each test we use the stopping criterion

$$\begin{matrix} \|c(x_k)\| \leqslant \epsilon_p \\ \|N(x)g_\sigma(x_k)\| \leqslant \epsilon_d \end{matrix} \qquad \text{or} \qquad \|N(x)\nabla\phi_\sigma(x_k)\| \leqslant \epsilon_d, \tag{11.1}$$

with $N(x) := \operatorname{diag}(\min\{x - \ell, u - x, \mathbb{1}\})$, $\epsilon_p := \epsilon\,(1 + \|x_k\|_\infty + \|c(x_0)\|_\infty)$, and $\epsilon_d := \epsilon\,(1 + \|y_k\|_\infty + \|g_\sigma(x_0)\|_\infty)$, where $\epsilon > 0$ is a tolerance, e.g., $\epsilon = 10^{-8}$. We also keep σ fixed for each experiment.

Depending on the problem, the augmented systems (10.16) are solved by either direct or iterative methods. For direct methods, we use the corrected semi-normal equations (Björck and Paige, 1994); see Appendix C. For iterative solves, we use CRAIG (Craig, 1955; Arioli, 2013) (computed via LNLQ) with preconditioner $\mathcal{P}$ and two possible termination criteria:

$$\left\| \begin{bmatrix} p^\star \\ q^\star \end{bmatrix} - \begin{bmatrix} p^{(k)} \\ q^{(k)} \end{bmatrix} \right\|_{\bar{\mathcal{P}}} \leqslant \eta \left\| \begin{bmatrix} p^{(k)} \\ q^{(k)} \end{bmatrix} \right\|_{\bar{\mathcal{P}}}, \qquad \bar{\mathcal{P}} := \begin{bmatrix} I & \\ & \mathcal{P} \end{bmatrix} \tag{11.2a}$$

$$\left\| \mathcal{K} \begin{bmatrix} p^{(k)} \\ q^{(k)} \end{bmatrix} - \begin{bmatrix} u \\ v \end{bmatrix} \right\|_{\bar{\mathcal{P}}^{-1}} \leqslant \eta \left\| \begin{bmatrix} u \\ v \end{bmatrix} \right\|_{\bar{\mathcal{P}}^{-1}}, \tag{11.2b}$$

which are respectively based on the relative error (obtained via LNLQ; see Chapter 5) and the relative residual. We can use (11.2a) when a lower bound on $\sigma_{\min}(A\mathcal{P}^{-1/2})$ is available (e.g., for the PDE-constrained optimization problems).

We use KNITRO (Byrd et al., 2006) and a Matlab implementation of TRON (Lin and Moré, 1999b). Our implementation of TRON[1] does not require explicit Hessians (only Hessian-vector products) and is unpreconditioned. We use $B_1(x)$ (10.14a) when efficient products with $S(u, x)$ are available, otherwise we use $B_2(x)$ (10.14b). When ϕ_σ is evaluated approximately (for coarse η), we use the solvers without modification, thus pretending that the function and gradient are evaluated exactly.

[1] https://github.com/optimizers/bcflash

Table 11.1: Results of solving (11.3) using TRON to minimize ϕ_σ, with various η in (11.2a) (left) and (11.2b) (right) to terminate the linear system solves. We record the number of function/gradient evaluations ($\#f, g$), Lagrangian Hessian ($\#Hv$), Jacobian ($\#Av$), and adjoint Jacobian ($\#A^Tv$) products.

	error-based termination					residual-based termination				
η	Iter.	$\#f, g$	$\#Hv$	$\#Av$	$\#A^Tv$	Iter.	$\#f, g$	$\#Hv$	$\#Av$	$\#A^Tv$
10^{-2}	37	37	19112	56797	50553	35	35	7275	29453	27148
10^{-4}	34	34	6758	35559	33423	35	35	7185	36757	34482
10^{-6}	35	35	7182	45893	43619	35	35	7194	47999	45721
10^{-8}	35	35	7176	53296	51204	35	35	7176	54025	51753
10^{-10}	35	35	7176	59802	57530	35	35	7176	59310	57038

11.2 1D Burger's equation

We solve the following one-dimensional ODE-constrained control problem:

$$\begin{aligned} \underset{u,z}{\text{minimize}} \quad & \tfrac{1}{2}\int_\Omega (u(s) - u_d(s))^2\, ds + \tfrac{1}{2}\alpha \int_\Omega z(s)^2 dx \\ \text{subject to} \quad & -\nu u_{ss} + u u_s = z + h \quad \text{in } \Omega, \\ & u(0) = 0, \;\; u(1) = -1, \end{aligned} \tag{11.3}$$

where the constraint is a 1D Burger's equation over $\Omega = (0, 1)$, with $h(s) = 2\left(\nu + s^3\right)$ and $\nu = 0.08$. The first objective term measures deviation from the data $u_d(s)$, while the second term regularizes the control with $\alpha = 10^{-2}$. We discretize (11.3) by segmenting Ω into $n_c = 512$ equal-sized cells, and approximate u and z with piecewise linear elements. This results in a nonlinearly constrained optimization problem with $n = 2n_c = 1024$ variables and $m = n_c - 1$ constraints.

We optimize $x = (u, z)$ by minimizing ϕ_σ with $\sigma = 10^3$, using $B_1(x)$ (9.17a) as Hessian approximation and $u_0 = \mathbb{1}$, $z_0 = \mathbb{1}$ as the initial point. We use TRON to optimize ϕ_σ and LNLQ to (approximately) solve (10.16). We partition the Jacobian of the discretized constraints into $A(x)^T = \begin{bmatrix} A_u(x)^T & A_z(x)^T \end{bmatrix}$, where $A_u(x) \in \mathbb{R}^{n\times n}$ and $A_z(x) \in \mathbb{R}^{m\times n}$ are the Jacobians for u and z. We use the preconditioner $\mathcal{P}(x) = A_u(x)^T A_u(x)$, which amounts to performing two solves of Burger's equation with a given source. For this preconditioner, $\sigma_{\min}(A\mathcal{P}^{-1/2}) \geqslant 1$, allowing us to bound the error via LNLQ and to use both (11.2a) and (11.2b) to terminate LNLQ. The maximum number of inner-CG iterations (for solving the trust-region subproblem) is n.

We choose $\epsilon = 10^{-8}$ in the stopping conditions (11.1). Table 11.1 records the number of Hessian- and Jacobian-vector products as we vary the accuracy of the linear system solves via η in (11.2). TRON required a moderate number of trust-region iterations. However, evaluating ϕ_σ and its derivatives can require many Jacobian and Hessian products, because for every product with the approximate Hessian we need to solve an augmented linear system. On the other hand, the linear systems did not have to be solved to full precision. As η increased from 10^{-10} to 10^{-2}, the number of Hessian-vector products stayed relatively constant, but the number of Jacobian-vector products dropped substantially, and the average

Table 11.2: Results of solving (11.4) using TRON to minimize ϕ_σ with various η in (11.2a) (left) and (11.2b) (right) to terminate the linear system solves. We record the number of function/gradient evaluations (#f, g), Lagrangian Hessian (#Hv), Jacobian (#Av), and adjoint Jacobian (#A^Tv) products.

η	Iter.	$\#f, g$	$\#Hv$	$\#Av$	$\#A^Tv$	Iter.	$\#f, g$	$\#Hv$	$\#Av$	$\#A^Tv$
10^{-2}	29	29	874	1794	2608	27	27	850	1772	2562
10^{-4}	27	27	830	1950	2728	25	25	668	1649	2265
10^{-6}	27	27	866	2317	3129	27	27	868	2356	3168
10^{-8}	27	27	866	2673	3485	27	27	866	2784	3596
10^{-10}	27	27	866	3145	3957	27	27	866	3251	4063
	error-based termination					residual-based termination				

number of LNLQ iterations required per solve dropped from about 9 to 5, except when $\eta = 10^{-2}$ in (11.2a) and the linear solves were too inaccurate so that the number of CG iterations per trust-region subproblem increased dramatically near the solution (requiring more linear solves). Using (11.2b) tended to perform more products with the Lagrangian Hessian and Jacobian, except when the linear solves were nearly exact, or extremely inexact.

11.3 2D Inverse Poisson problem

Let $\Omega = (0,1)^2$ denote the physical domain and $H^1(\Omega)$ denote the Sobolev space of functions in $L^2(\Omega)$, whose weak derivatives are also in $L^2(\Omega)$. Let $H_0^1(\Omega) \subset H^1(\Omega)$ be the Hilbert space of functions whose value on the boundary $\partial\Omega$ is zero. We solve the following 2D PDE-constrained control problem:

$$\begin{aligned} \underset{u \in H_0^1(\Omega),\, z \in L^2(\Omega)}{\text{minimize}} \quad & \tfrac{1}{2}\int_\Omega (u - u_d)^2\, ds + \tfrac{1}{2}\alpha \int_\Omega z^2 ds \\ \text{subject to} \quad & -\nabla \cdot (z \nabla u) = h \quad \text{in } \Omega, \\ & u = 0 \quad \text{in } \partial\Omega. \end{aligned} \tag{11.4}$$

Let $c = (0.2, 0.2)$ and define $S_1 = \{s \mid \|s - c\|_2 \leqslant 0.3\}$ and $S_2 = \{s \mid \|s - c\|_1 \leqslant 0.6\}$. The target state u_d is generated as the solution of the PDE with $z^\star(s) = 1 + 0.5 \cdot I_{S_1}(s) + 0.5 \cdot I_{S_2}(s)$, where for any set C, $I_C(s) = 1$ if $s \in C$ and 0 otherwise.

The force term is $h(s_1, s_2) = -\sin(\omega s_1)\sin(\omega s_2)$, with $\omega = \pi - \frac{1}{8}$. The control variable z represents the diffusion coefficients for the Poisson problem that we are trying to recover based on the observed state u_d. We set $\alpha = 10^{-4}$ as regularization parameter. We discretize (11.4) using P_1 finite elements on a uniform mesh of 1089 triangular elements and employ an identical discretization for the optimization variables $z \in L^2(\Omega)$, obtaining a problem with $n_u = 961$ states and $n_z = 1089$ controls, so that $n = n_u + n_z$. There are $m = n_u$ constraints, as we must solve the PDE on every interior grid point. The initial point is $u_0 = \mathbb{1}$, $z_0 = \mathbb{1}$.

We use $\sigma = 10^{-2}$ as penalty parameter and $B_2(x)$ as Hessian approximation. We again use LNLQ for the linear solves, with the same preconditioning strategy as in Section 11.2. The results are given in Table 11.2. We see a trend similar to that of Table 11.1, as larger η allows TRON to converge within nearly the same number of outer iterations and Lagrangian

Table 11.3: Results of solving (11.4) with $z \geqslant 0$ using KNITRO on (PP) with various η in (11.2a) (left) and (11.2b) (right) to terminate the linear system solves. The top (resp. bottom) table records results for the smaller problem with $n = 2050$, $m = 1089$ (resp. larger problem with $n = 20002$, $m = 10201$). We record the number of function/gradient evaluations ($\#f, g$), Lagrangian Hessian ($\#Hv$), Jacobian ($\#Av$), and adjoint Jacobian ($\#A^Tv$) products.

η	Iter.	$\#f,g$	$\#Hv$	$\#Av$	$\#A^Tv$	Iter.	$\#f,g$	$\#Hv$	$\#Av$	$\#A^Tv$
10^{-2}	46	64	2856	8436	8611	67	81	4374	12915	13145
10^{-4}	43	55	2168	6642	6796	36	51	1458	4642	4781
10^{-6}	35	46	2120	6876	7004	29	35	1194	4138	4238
10^{-8}	39	50	2322	7833	7973	47	71	7062	22150	22340
10^{-10}	37	47	2236	8110	8242	43	58	3170	11565	11725

10^{-2}	144	176	3662	12395	12892	100	126	3716	11702	12055
10^{-4}	131	177	4002	14470	14956	83	117	2752	9264	9582
10^{-6}	103	135	4386	15035	15409	88	132	4170	14421	14774
10^{-8}	73	103	3250	11960	12244	101	133	3726	13878	14246
10^{-10}	79	109	4088	15527	15825	104	139	5378	20291	20674

error-based termination residual-based termination

Hessian-vector products (even when $\eta = 10^{-2}$), while significantly decreasing the number of Jacobian-vector products. We see again that using (11.2a) to terminate LNLQ tends to need less work than with (11.2b). The exception is using (11.2b) with $\eta = 10^{-4}$. The solver terminates two iterations sooner, resulting in a sharp drop in Jacobian-vector products but equally good solution quality. Note that if $\epsilon = 10^{-9}$ were used for the experiment, the runs would appear more similar to one another.

We now solve (11.4) with the additional bound constraint $z \geqslant 0$ (keeping the diffusivity coefficient nonnegative). We solve the problem with the same discretization as before, as well as a second refined discretization with $n_z = 10201$, $n_u = 9801$.

We optimize $x = (u, z)$ by applying KNITRO to (PP) with the same initial conditions and penalty parameter. We again use the preconditioner $\mathcal{P}(x) = A_u(x)^T A_u(x)$, which ensures that $\sigma_{\min}(A(x)\mathcal{P}^{-1/2}) \geqslant 1$ because the bound constraints only apply to $z \geqslant 0$. Then $Q(x) = \operatorname{blkdiag}(I, Z)$ with $Z = \operatorname{diag}(z)$, and

$$\begin{aligned}\mathcal{P}^{-1}A(x)^T Q(x) A(x) &= \mathcal{P}^{-1}(A_u(x)^T A_u(x) + A_z(x) Z A_z(x)) \\ &= I + \mathcal{P}^{-1}A_z(x) Z A_z(x).\end{aligned}$$

Therefore we can use LNLQ with either (11.2b) or (11.2a) as termination criterion.

The results are recorded in Table 11.3. We observed that for the smaller problem, KNITRO converged in a moderate number of outer iterations in all cases. With (11.2a), we see that the number of Jacobian products tended to decrease, except when $\eta = 10^{-2}$, because the linear solves were too coarse. Using (11.2b) showed a less clear trend. In cases with comparable outer iteration numbers, larger η resulted in fewer Jacobian products. However, for moderate η the number of outer iterations proved to be significantly smaller, resulting in a more efficient solve than when η was too small or too large.

Table 11.4: Comparison of Fletcher (top) and composite step (bottom) on (11.4). Each table provides the corresponding iteration log. We record the objective value ($f(x_k)$), constraint violation ($\|c(x_k)\|$), Lagrangian gradient norm ($\|\nabla L(x_k, y_k)\|$), penalty function $\phi_\sigma(x_k)$, penalty gradient ($\|\phi_\sigma(x_k)\|$), CG iterations (CG), function and gradient evaluations ($\#f, g$), number of linear solves (Sys), and total GMRES iterations (Sys iter.).

Iter.	$f(x_k)$	$\|c(x_k)\|$	$\|\nabla L(x_k, y_k)\|$	$\phi_\sigma(x_k)$	$\|\nabla \phi_\sigma(x_k)\|$	CG	$\#f, g$
0	$6.6 \cdot 10^{-02}$	$1.1 \cdot 10^{-15}$	$6.4 \cdot 10^{-06}$	$6.6 \cdot 10^{-02}$	$6.4 \cdot 10^{-06}$		1
1	$6.6 \cdot 10^{-02}$	$5.5 \cdot 10^{-07}$	$3.0 \cdot 10^{-08}$	$6.6 \cdot 10^{-02}$	$3.5 \cdot 10^{-08}$	7	2
2	$6.6 \cdot 10^{-02}$	$7.6 \cdot 10^{-09}$	$3.2 \cdot 10^{-10}$	$6.6 \cdot 10^{-02}$	$3.4 \cdot 10^{-10}$	19	3
3	$6.6 \cdot 10^{-02}$	$8.1 \cdot 10^{-09}$	$3.7 \cdot 10^{-11}$	$6.6 \cdot 10^{-02}$	$7.3 \cdot 10^{-11}$	38	4
					Solve Time:		0.52s

Iter.	$f(x_k)$	$\|c(x_k)\|$	$\|\nabla L(x_k, y_k)\|$	Sys	Sys iter.	CG	$\#f, g$
0	$6.6 \cdot 10^{-02}$	$1.1 \cdot 10^{-15}$	$6.4 \cdot 10^{-06}$				
1	$6.6 \cdot 10^{-02}$	$1.7 \cdot 10^{-10}$	$2.8 \cdot 10^{-08}$	9	47	3	3
2	$6.6 \cdot 10^{-02}$	$3.3 \cdot 10^{-11}$	$2.3 \cdot 10^{-10}$	21	112	7	5
3	$6.6 \cdot 10^{-02}$	$6.1 \cdot 10^{-16}$	$1.2 \cdot 10^{-12}$	38	211	12	7
					Solve Time:		0.21s

For the larger problem with termination condition (11.2a), the number of outer iterations increased with η, the number of Lagrangian Hessian products fluctuated somewhat, and Jacobian products tended to decrease. The exception is $\eta = 10^{-8}$, which hits the sweet spot of solving the linear systems sufficiently accurately to avoid many additional outer iterations, but without performing too many iterations for each linear solve. Using residual-based termination (11.2b) shows a less clear trend; Jacobian products roughly decreased with increasing η while the Hessian products tended to oscillate. The sweet spot is hit with $\eta = 10^{-4}$, where the fewest outer iterations and operator products were performed. For this problem, it appears that the dependence of performance on the accuracy of the linear solves as measured by the residual (11.2b) is much more nonlinear than when the linear solves are terminated according to the error (11.2a).

11.3.1 Comparison with ROL

We compare our penalty method to the composite step trust-region method (Heinkenschloss and Ridzal, 2014) on (11.4). Both methods are implemented in C++ as part of the Rapid Optimization Library (ROL) in Trilinos (Heroux et al., 2003). Solving (11.4) on the unit cube discretized as an $8 \times 8 \times 8$ grid results in a problem with $n = 1458$ variables and $m = 729$ constraints. We run the composite step method (Table 11.4 bottom) and Fletcher (Table 11.4 top), and record their iteration logs. To solve the augmented systems, we use preconditioned GMRES (with the same preconditioner as before), terminated when the residual norm is below 10^{-12}. For Fletcher, we use $\sigma = 10^{-2}$ as the penalty parameter.

We see that the performance of Fletcher and the composite step method are comparable in terms of outer iterations and time. Fletcher solves about twice as many augmented systems (every CG iteration is two linear solves), and so it takes roughly twice the solve time of the composite step method.

Table 11.5: Results of solving (11.5) using TRON to minimize ϕ_σ with various η in (11.2a) (left) and (11.2b) (right) to terminate the linear system solves. We record the number of Lagrangian Hessian ($\#Hv$), Jacobian ($\#Av$), and adjoint Jacobian ($\#A^T v$) products.

η	Iter.	$\#f,g$	$\#Hv$	$\#Av$	$\#A^T v$	Iter.	$\#f,g$	$\#Hv$	$\#Av$	$\#A^T v$
10^{-2}	29	29	822	1582	2342	29	29	822	1636	2396
10^{-4}	29	29	816	1635	2389	29	29	816	1801	2555
10^{-6}	29	29	816	1800	2554	29	29	816	2029	2783
10^{-8}	29	29	816	2077	2831	29	29	816	2301	3055
10^{-10}	29	29	816	2351	3105	29	29	816	2637	3391
	error-based termination					residual-based termination				

11.4 2D Poisson-Boltzmann problem

We now solve a control problem where the constraint is a 2D Poisson-Boltzmann equation:

$$\begin{aligned} \underset{u\in H_0^1(\Omega),\, z\in L^2(\Omega)}{\text{minimize}} \quad & \tfrac{1}{2}\int_\Omega (u-u_d)^2\, ds + \tfrac{1}{2}\alpha\int_\Omega z^2 ds \\ \text{subject to} \quad & -\Delta u + \sinh(u) = h + z \quad \text{in } \Omega, \\ & \qquad\qquad\quad u = 0 \qquad\quad \text{in } \partial\Omega. \end{aligned} \tag{11.5}$$

We use the same notation and Ω as in Section 11.3, with the forcing term $h(s_1, s_2) = -\sin(\omega s_1)\sin(\omega s_2)$, $\omega = \pi - \frac{1}{8}$, and target state

$$u_d(s) = \begin{cases} 10 & \text{if } s \in [0.25, 0.75]^2 \\ 5 & \text{otherwise.} \end{cases}$$

We discretize (11.5) using P_1 finite elements on a uniform mesh with 1089 triangular elements, resulting in a problem with $n = 2050$ variables and $m = 961$ constraints. The initial point is $u_0 = \mathbb{1}$, $z_0 = \mathbb{1}$.

We perform the experiment described in Section 11.3 using $\sigma = 10^{-1}$, and record the results in Table 11.5. The results are similar to Table 11.2, where the number of Jacobian products decreases with η, while the number of outer iterations and Lagrangian-Hessian products stays fairly constant. We see that with stopping criterion (11.2b), the LNLQ iterations increase somewhat compared to (11.2a), as it is a tighter criterion.

We now repeat the experiment with the additional bound constraint $z \geqslant 0$, and solve the problem with the same discretization as before, as well as on a refined mesh where $n_z = 10201$ and $n_u = 9801$. The initial point and penalty parameter are unchanged, but we now use KNITRO as the optimization solver. The results are recorded in Table 11.6.

We see that the results for both problems are more robust to changes in the accuracy of the linear solves. In all cases, the number of outer iterations and function/gradient evaluations were the same, and the number of Lagrangian Hessian products changed little. The number of Jacobian products steadily decreased with increasing η, with a 20–30% drop in Jacobian products from $\eta = 10^{-10}$ to $\eta = 10^{-2}$.

Table 11.6: Results of solving (11.5) with $z \geqslant 0$ using KNITRO on (PP) with various η in (11.2a) (left) and (11.2b) (right) to terminate the linear system solves. The top (resp. bottom) table records results for the smaller problem with $n = 2050$, $m = 1089$ (resp. larger problem with $n = 20002$, $m = 10201$). We record the number of function/gradient evaluations ($\#f, g$), Lagrangian Hessian ($\#Hv$), Jacobian ($\#Av$), and adjoint Jacobian ($\#A^Tv$) products.

η	Its.	$\#f,g$	$\#Hv$	$\#Av$	$\#A^Tv$	Its.	$\#f,g$	$\#Hv$	$\#Av$	$\#A^Tv$
10^{-2}	19	20	1242	3648	3708	19	20	1242	3669	3729
10^{-4}	19	20	1252	3753	3813	19	20	1244	3762	3822
10^{-6}	19	20	1236	3868	3928	19	20	1234	3916	3976
10^{-8}	19	20	1244	4169	4229	19	20	1236	4286	4346
10^{-10}	19	20	1238	4725	4785	19	20	1250	4986	5046
10^{-2}	30	37	1524	4426	4531	30	37	1524	4468	4573
10^{-4}	30	37	1524	4574	4679	30	37	1524	4632	4737
10^{-6}	30	37	1524	4813	4918	30	37	1558	5033	5138
10^{-8}	30	37	1550	5396	5501	30	37	1550	5610	5715
10^{-10}	30	37	1550	6224	6329	30	37	1558	6582	6687

error-based termination residual-based termination

11.4.1 Comparison with ROL

We perform a comparison similar to Section 11.3.1 for (11.5), by comparing our method with the composite step trust-region method of Heinkenschloss and Ridzal (2014) and the augmented Lagrangian method (Hestenes, 1969; Powell, 1969; Conn et al., 1992). All method are implemented in C++ as part of ROL in Trilinos.

We first solve (11.5) without constraints on z. We discretize Ω on a 50×50 grid, producing a problem with $n = 5202$ variables and $m = 2601$ constraints. We then apply Fletcher's penalty function (with $\sigma = 10^{-1}$) and the composite step method to this problem, and record the results in Table 11.7. All augmented systems are solved using GMRES with the same preconditioner as before. The composite step method is significantly more efficient in this case, as our approach requires many CG iterations to converge near the solution.

We now solve the problem with bound constraints on the control variables: $0 \leqslant z \leqslant 10$. We discretize Ω into a 60×20 grid so that we have $n = 2562$ variables and $m = 1281$ constraints. We solve the problem using our approach and the augmented Lagrangian method, and record the results in Table 11.8. In this case, our penalty approach shows itself to be more efficient than using the augmented Lagrangian method. It should be noted that the Lagrangian Hessian and augmented Lagrangian Hessian are highly singular, resulting in slow convergence for both methods. In particular, this results in the high number of CG iterations for our approach and for the augmented Lagrangian method. A reduced-space method that always maintains feasibility with respect to the PDE constraint can be particularly advantageous here because the issue with the singular Hessian does not apply. However, it may not be as effective in large-scale cases because it becomes impractical to always solve accurately the differential equation, whereas full-space methods such as our approach can take advantage of inexact solves.

Table 11.7: Comparison of Fletcher (top) and composite step (bottom) on (11.5). Each table provides the corresponding iteration log. We record the objective value ($f(x_k)$), constraint violation ($\|c(x_k)\|$), Lagrangian gradient norm ($\|\nabla L(x_k, y_k)\|$), penalty function $\phi_\sigma(x_k)$, penalty gradient ($\|\phi_\sigma(x_k)\|$), CG iterations (CG), function and gradient evaluations ($\#f, g$), number of linear solves (Sys), and total GMRES iterations (Sys iter.).

Iter.	$f(x_k)$	$\|c(x_k)\|$	$\|\nabla L(x_k, y_k)\|$	$\phi_\sigma(x_k)$	$\|\nabla \phi_\sigma(x_k)\|$	CG	$\#f, g$
0	$3.1 \cdot 10^{-01}$	$1.3 \cdot 10^{-13}$	$9.3 \cdot 10^{-03}$	$3.1 \cdot 10^{-01}$	$9.3 \cdot 10^{-03}$		
1	$2.2 \cdot 10^{-01}$	$1.8 \cdot 10^{-04}$	$3.6 \cdot 10^{-02}$	$2.3 \cdot 10^{-01}$	$7.0 \cdot 10^{-02}$	1	2
2	$1.5 \cdot 10^{-01}$	$4.4 \cdot 10^{-02}$	$3.9 \cdot 10^{-02}$	$1.7 \cdot 10^{-01}$	$7.7 \cdot 10^{-02}$	2	3
3	$8.7 \cdot 10^{-02}$	$2.7 \cdot 10^{-01}$	$5.5 \cdot 10^{-02}$	$1.1 \cdot 10^{-01}$	$1.1 \cdot 10^{-01}$	4	4
4	$5.8 \cdot 10^{-02}$	$1.2 \cdot 10^{-01}$	$1.3 \cdot 10^{-02}$	$6.0 \cdot 10^{-02}$	$2.6 \cdot 10^{-02}$	5	5
5	$2.1 \cdot 10^{-02}$	$4.7 \cdot 10^{-01}$	$3.4 \cdot 10^{-02}$	$3.3 \cdot 10^{-02}$	$6.8 \cdot 10^{-02}$	4	6
6	$8.0 \cdot 10^{-03}$	$2.2 \cdot 10^{-04}$	$1.6 \cdot 10^{-02}$	$1.0 \cdot 10^{-02}$	$3.2 \cdot 10^{-02}$	6	7
7	$6.3 \cdot 10^{-03}$	$7.2 \cdot 10^{-04}$	$2.2 \cdot 10^{-03}$	$6.4 \cdot 10^{-03}$	$4.4 \cdot 10^{-03}$	5	8
8	$2.7 \cdot 10^{-03}$	$1.0 \cdot 10^{-04}$	$6.5 \cdot 10^{-03}$	$3.1 \cdot 10^{-03}$	$1.3 \cdot 10^{-02}$	8	9
9	$2.7 \cdot 10^{-03}$	$1.6 \cdot 10^{-05}$	$2.6 \cdot 10^{-05}$	$2.7 \cdot 10^{-03}$	$2.6 \cdot 10^{-05}$	1	10
10	$2.3 \cdot 10^{-03}$	$2.4 \cdot 10^{-06}$	$2.6 \cdot 10^{-04}$	$2.3 \cdot 10^{-03}$	$5.2 \cdot 10^{-04}$	99	11
11	$2.3 \cdot 10^{-03}$	$4.9 \cdot 10^{-07}$	$5.8 \cdot 10^{-07}$	$2.3 \cdot 10^{-03}$	$5.8 \cdot 10^{-07}$	1	12
12	$2.3 \cdot 10^{-03}$	$1.4 \cdot 10^{-06}$	$2.8 \cdot 10^{-06}$	$2.3 \cdot 10^{-03}$	$5.7 \cdot 10^{-06}$	500	13
13	$2.3 \cdot 10^{-03}$	$1.5 \cdot 10^{-08}$	$7.8 \cdot 10^{-08}$	$2.3 \cdot 10^{-03}$	$1.5 \cdot 10^{-07}$	3	14
14	$2.3 \cdot 10^{-03}$	$1.5 \cdot 10^{-07}$	$1.8 \cdot 10^{-07}$	$2.3 \cdot 10^{-03}$	$3.6 \cdot 10^{-07}$	500	15
15	$2.3 \cdot 10^{-03}$	$5.7 \cdot 10^{-10}$	$2.8 \cdot 10^{-09}$	$2.3 \cdot 10^{-03}$	$2.8 \cdot 10^{-09}$	1	16
					Solve time:	19.9817s	

Iter.	$f(x_k)$	$\|c(x_k)\|$	$\|\nabla L(x_k, y_k)\|$	Sys	Sys iter.	CG	$\#f, g$
0	$3.1 \cdot 10^{-01}$	$4.4 \cdot 10^{-13}$	$2.2 \cdot 10^{-02}$				
1	$5.4 \cdot 10^{-02}$	$2.4 \cdot 10^{-02}$	$1.2 \cdot 10^{-02}$	7	45	2	3
2	$6.4 \cdot 10^{-03}$	$7.8 \cdot 10^{-04}$	$4.2 \cdot 10^{-04}$	14	106	2	5
3	$2.4 \cdot 10^{-03}$	$1.0 \cdot 10^{-04}$	$5.0 \cdot 10^{-06}$	24	199	5	7
4	$2.3 \cdot 10^{-03}$	$2.4 \cdot 10^{-07}$	$2.1 \cdot 10^{-07}$	40	370	11	9
5	$2.3 \cdot 10^{-03}$	$5.6 \cdot 10^{-10}$	$2.4 \cdot 10^{-09}$	63	595	20	11
					Solve time:		0.28s

Table 11.8: Comparison of Fletcher (top) and augmented Lagrangian (bottom) on (11.5) with bound constraints. Each table provides the corresponding iteration log. We record the objective value ($f(x_k)$), constraint violation ($\|c(x_k)\|$), Lagrangian gradient norm ($\|\nabla L(x_k, y_k)\|$), penalty function $\phi_\sigma(x_k)$, penalty gradient ($\|\phi_\sigma(x_k)\|$), CG iterations (CG), function, gradient, and constraint evaluations ($\#f, g, c$), and augmented Lagrangian penalty parameter (ρ).

Iter.	$f(x_k)$	$\|c(x_k)\|$	$\|\nabla L(x_k, y_k)\|$	$\phi_\sigma(x_k)$	$\|\nabla\phi_\sigma(x_k)\|$	CG	$\#f, g$	$\#c$
0	$7.7 \cdot 10^{-04}$	$6.4 \cdot 10^{-09}$	$5.4 \cdot 10^{-02}$	$7.7 \cdot 10^{-04}$	$5.4 \cdot 10^{-02}$			1
1	$7.6 \cdot 10^{-05}$	$2.7 \cdot 10^{-06}$	$1.4 \cdot 10^{-01}$	$8.4 \cdot 10^{-05}$	$2.9 \cdot 10^{-01}$	3266	2	3
2	$5.5 \cdot 10^{-05}$	$9.1 \cdot 10^{-06}$	$1.6 \cdot 10^{-03}$	$5.5 \cdot 10^{-05}$	$2.9 \cdot 10^{-03}$	1882	3	5
3	$5.4 \cdot 10^{-05}$	$7.2 \cdot 10^{-06}$	$1.5 \cdot 10^{-03}$	$5.4 \cdot 10^{-05}$	$3.0 \cdot 10^{-03}$	2519	4	7
4	$5.4 \cdot 10^{-05}$	$6.9 \cdot 10^{-06}$	$6.5 \cdot 10^{-05}$	$5.4 \cdot 10^{-05}$	$3.0 \cdot 10^{-05}$	11	5	9
5	$5.3 \cdot 10^{-05}$	$2.3 \cdot 10^{-08}$	$7.1 \cdot 10^{-03}$	$5.3 \cdot 10^{-05}$	$1.4 \cdot 10^{-02}$	4476	6	11
6	$5.3 \cdot 10^{-05}$	$2.7 \cdot 10^{-09}$	$1.0 \cdot 10^{-05}$	$5.3 \cdot 10^{-05}$	$2.0 \cdot 10^{-05}$	4	7	13
7	$5.3 \cdot 10^{-05}$	$1.2 \cdot 10^{-08}$	$4.3 \cdot 10^{-07}$	$5.3 \cdot 10^{-05}$	$3.5 \cdot 10^{-07}$	1614	8	15
8	$5.3 \cdot 10^{-05}$	$5.4 \cdot 10^{-09}$	$8.6 \cdot 10^{-07}$	$5.3 \cdot 10^{-05}$	$1.4 \cdot 10^{-06}$	5000	9	17
9	$5.3 \cdot 10^{-05}$	$3.5 \cdot 10^{-09}$	$4.6 \cdot 10^{-08}$	$5.3 \cdot 10^{-05}$	$2.1 \cdot 10^{-08}$	3831	10	19
10	$5.3 \cdot 10^{-05}$	$1.4 \cdot 10^{-09}$	$1.5 \cdot 10^{-07}$	$5.3 \cdot 10^{-05}$	$2.6 \cdot 10^{-07}$	5000	11	21
11	$5.3 \cdot 10^{-05}$	$1.1 \cdot 10^{-10}$	$7.5 \cdot 10^{-09}$	$5.3 \cdot 10^{-05}$	$1.9 \cdot 10^{-08}$	5000	12	23
						Solve time:		664.1s

Iter.	$f(x_k)$	$\|c(x_k)\|$	$\|\nabla L(x_k, y_k)\|$	CG	ρ	$\#f$	$\#g$	$\#c$
0	$7.7 \cdot 10^{-04}$	$6.4 \cdot 10^{-09}$	$5.7 \cdot 10^{-02}$		10^{1}			
1	$3.1 \cdot 10^{-05}$	$1.0 \cdot 10^{-03}$	$2.9 \cdot 10^{-04}$	1	10^{1}	5	5	6
2	$6.8 \cdot 10^{-05}$	$4.2 \cdot 10^{-00}$	$9.5 \cdot 10^{-05}$	19	10^{2}	44	41	61
3	$8.6 \cdot 10^{-03}$	$4.3 \cdot 10^{-01}$	$1.7 \cdot 10^{-01}$	4	10^{3}	55	50	76
4	$8.2 \cdot 10^{-05}$	$4.2 \cdot 10^{-02}$	$1.7 \cdot 10^{-02}$	10	10^{4}	76	71	107
5	$5.7 \cdot 10^{-05}$	$4.2 \cdot 10^{-03}$	$1.5 \cdot 10^{-03}$	5	10^{5}	89	82	125
6	$5.4 \cdot 10^{-05}$	$4.2 \cdot 10^{-04}$	$1.9 \cdot 10^{-04}$	53	10^{5}	323	183	406
7	$5.3 \cdot 10^{-05}$	$1.5 \cdot 10^{-07}$	$1.7 \cdot 10^{-10}$	168	10^{5}	1128	509	1368
8	$5.3 \cdot 10^{-05}$	$2.6 \cdot 10^{-10}$	$1.5 \cdot 10^{-10}$	83	10^{5}	1512	671	1830
						Solve time:		1314.79s

Table 11.9: Results from solving (11.6) using KNITRO to optimize (PP) with various η in (11.2a) (left) and (11.2b) (right) to terminate the linear system solves. We record the number of function/gradient evaluations $\#f, g$), Lagrangian Hessian ($\#Hv$), Jacobian ($\#Av$), and adjoint Jacobian ($\#A^Tv$) products. The symbol $*$ indicates that the problem failed to converge to a feasible point after 500 iterations.

η	Its.	$\#f, g$	$\#Hv$	$\#Av$	$\#A^Tv$	Its.	$\#f, g$	$\#Hv$	$\#Av$	$\#A^Tv$
10^{-2}	217	340	4340	13966	15204	$*$	$*$	$*$	$*$	$*$
10^{-4}	226	348	4396	14068	15204	$*$	$*$	$*$	$*$	$*$
10^{-6}	176	272	3232	11218	12211	191	291	3508	18326	19391
10^{-8}	185	289	3356	11582	12635	196	296	3700	20888	21973
10^{-10}	204	298	4626	15412	16511	190	286	3480	23979	25028

error-based termination residual-based termination

11.5 2D topology optimization

We now solve the following 2D topology optimization problem from Gersborg-Hansen, Bendsøe, and Sigmund (2006):

$$\begin{array}{ll} \underset{u\in H_0^1(\Omega),\, z\in L^2(\Omega)}{\text{minimize}} & \displaystyle\int_\Omega f u \, ds \\ \text{subject to} & \begin{aligned} \textstyle\int_\Omega z\, ds &\leqslant V \\ -\nabla\cdot(k(z)\nabla u) &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{in } \partial\Omega, \\ 0 \leqslant z &\leqslant 1, \end{aligned} \end{array} \tag{11.6}$$

where $k(z) : \Omega \to \Omega$ defined by $k(z)(s) = 10^{-3} + (1 - 10^{-3})z(s)^3$ for $s \in \Omega$. The domain is $\Omega = [0, 1]^2$, with load vector $f = 10^{-2}$, and $V = 0.4$. We discretize (11.6) using finite elements on a 64×64 grid as described by Gersborg-Hansen et al. (2006), resulting in a problem with 8321 variables and 4096 equality constraints. After discretization, we add a slack variable $\bar{s} \geqslant 0$ for the first inequality constraint, so we have only equality constraints and bounds. The problem has $n = 8322$ variables and $m = 4096$ constraints, with bounds on z and $\bar{s}$.

We perform the same experiment as in Section 11.3 (using one mesh), with $\sigma = 10^{-1}$ as the penalty parameter, initial point $u_0 = \frac{1}{2}V\mathbb{1}$, $z_0 = \frac{1}{2}V\mathbb{1}$, $\bar{s}_0 = V - \sum z_i = 0.2$, and KNITRO as the minimization algorithm. The results are recorded in Table 11.9. With (11.2a), the trend is like before: as η increases the number of Jacobian products decreases (and in this case, so do the numbers of outer iterations and Lagrangian Hessian products), but this is only true until η becomes too large and the linear solves become too coarse, causing slowed convergence. With (11.2b) as the termination criterion, we see a similar trend, except that when the linear solves are too coarse, KNITRO fails to converge.

Table 11.10: Results for problems with linear constraints (first three rows have only equality constraints). m_{lin} and m_{nln} are the number of linear and nonlinear constraints; $\sigma^\star_{\text{impl}}$ and $\sigma^\star_{\text{expl}}$ are threshold penalty parameters when the linear constraints are handled implicitly and explicitly; σ is the penalty parameter. The last two columns give the number of iterations before convergence; the symbol $*$ indicates that unboundedness was detected, and – that 100 iterations were performed without converging. The solver exits when unboundedness is detected or an iterate satisfies (11.1) with $\epsilon = 10^{-8}$.

Problem	n	m_{lin}	m_{nln}	$\sigma^\star_{\text{impl}}$	$\sigma^\star_{\text{expl}}$	σ	Impl.	Expl.
`Chain400`	802	402	1	0.0012	0	10^{-3}	$*$	10
						0.002	7	10
`Channel400`	1600	800	800	0	0	10^{-3}	–	5
						1	–	5
`hs113`	18	3	5	6.61	3.39	6	$*$	42
						7	28	17
`prodpl0`	69	25	4	211.9	13.7	40	–	43
						300	–	30
`prodpl1`	69	25	4	60.8	3.56	10	–	22
						70	89	41
`synthes3`	38	23	19	6.00	0.66	2	–	12
						7	35	18

11.6 Explicit linear constraints

We investigate the effect of maintaining the linear constraints explicitly (Section 9.3), using some problems from the CUTEst test set (Gould, Orban, and Toint, 2003) with linear constraints. We use KNITRO to minimize ϕ_σ with and without linear constraints, because it can handle them explicitly. We use the corrected semi-normal equations to perform linear solves, and Hessian approximation $B_1(x)$ (10.14a). The threshold penalty parameters (10.4) and (9.24) (adapted to bound constraints) are computed from earlier optimal solutions when the linear constraints were kept implicit ($\sigma^\star_{\text{impl}}$) and explicit ($\sigma^\star_{\text{expl}}$) respectively.

In Table 11.10, we observe that maintaining the linear constraints explicitly decreases the penalty parameter for all problems except `Channel400` ($\sigma^\star = 0$ in both cases). KNITRO fails to find an optimal solution when the linear constraints are implicit and $\sigma < \sigma^\star_{\text{impl}}$. This is because in the equality-constrained case ϕ_σ is unbounded, and otherwise KNITRO stalls without converging to a feasible solution. When σ is sufficiently large, both versions converge (with and without explicit constraints); in most cases keeping the constraints requires fewer iterations, except for `Chain400`. Although positive semidefiniteness of $\nabla^2\phi_\sigma(x^\star)$ is guaranteed in the relevant critical cone when $\sigma > \sigma^\star$ (both implicit and explicit cases), a larger value of σ may sometimes be required because the curvature of ϕ_σ away from the solution may be larger or ill-behaved.

For the `Channel` problems, the threshold parameter is zero in both cases. However, KNITRO converges quickly when the linear constraints are kept explicit, but otherwise fails to converge in a reasonable number of iterations. This phenomenon for the `Channel` problems appears to be independent of σ (more values were investigated than are reported here). Even if the threshold penalty parameter does not decrease, it appears beneficial to maintain some of the constraints explicitly.

Table 11.11: Convergence results for `hs061` (left) and `mss1` (right) when TRON and KNITRO minimize $\phi_\sigma(\cdot;\delta)$. The first rows show the iteration at which δ is updated, and the last two rows record the final primal and dual infeasibilities.

δ	TRON	KNITRO
10^{-1}	22	12
10^{-2}	23	13
10^{-4}	24	14
$\|c(\bar{x})\|$	10^{-10}	10^{-9}
$\|g_\sigma(\bar{x})\|$	10^{-7}	10^{-8}

δ	TRON	KNITRO
10^{-2}	46	33
10^{-4}	52	36
10^{-7}	53	37
$\|c(\bar{x})\|$	10^{-12}	10^{-14}
$\|g_\sigma(\bar{x})\|$	10^{-8}	10^{-9}

11.7 Regularization

We next solve problems where $A(x)$ is rank-deficient for some iterates, requiring that ϕ_σ be regularized (Section 5.5). We use the corrected semi-normal equations to solve the linear systems, with $B_2(x)$ as the Hessian approximation.

For problem `hs061` ($n = 3$ variables, $m = 3$ constraints) from the CUTEst test set (Gould et al., 2003) we use $x_0 = 0$, $\sigma = 10^2$, $\delta_0 = 10^{-1}$. For problem `mss1` ($n = 90$, $m = 73$) we use $x_0 = 0$, $\sigma = 10^3$, $\delta_0 = 10^{-2}$. In both cases we decrease δ according to $\nu(\delta) = \delta^2$ to retain local quadratic convergence. For both problems, $A(x_0)$ is rank-deficient and ϕ_σ is undefined, so the trust-region solvers terminate immediately. We therefore regularize ϕ_σ and record the iteration at which δ changed. For `mss1`, we set $\delta_{\min} = 10^{-7}$ to avoid ill-conditioning. The results are in Table 11.11.

The regularized problems converge with few iterations between δ updates, showing evidence of quadratic convergence. Note that a large δ can perturb $\phi_\sigma(\cdot;\delta)$ substantially, so that δ_0 may need to be chosen judiciously. We use $\delta_0 = 10^{-2}$ because neither TRON nor KNITRO would converge for `mss1` when $\delta_0 = 10^{-1}$.

Chapter 12

Contributions and future directions

We demonstrated that the smooth exact penalty method is a promising approach for nonlinearly constrained optimization, contrary to existing notions about its practical liminations. The method is particularly promising when the augmented linear systems (9.19) and (10.16) can be solved efficiently, for example, for PDE-constrained optimization problems.

12.1 Contributions

Efficient evaluation of the penalty function

We developed an efficient approach to evaluating the penalty function and its derivatives by solving the same linear system with different right-hand sides. We discussed the tradeoffs of using various solvers for solving these linear systems, with particular emphasis on the use of iterative methods in order to develop a factorization-free solver.

Extension to bound constraints

We developed a smooth extension to Fletcher's penalty function when bound constraints are present, compared to his original nonsmooth proposal in (8.9). This penalty function is exact when minimized over the bound constraints. We then showed how to evaluate the penalty function and its derivatives efficiently when bounds are present.

Extensions to special cases

We extended the penalty function to handle cases where the constraint Jacobian is rank-deficient away from the solution, and when some of the constraints are simple and can be kept explicit. When the constraint Jacobian is rank-deficient, we modify the penalty function to (9.34a). Algorithm 8 shows how to decrease the regularization parameter to zero in such a way that the overall convergence of the optimization algorithm is unimpeded. If some of the constraints are easy to maintain explicitly (e.g., linear constraints), we show how to modify the penalty function so that the threshold penalty parameter is lower. Maintaining linear constraints explicitly also tends to exhibit better convergence behaviour than when the constraints are only penalized.

12.2 Future directions

As members of the scientific computing community, we are not in the business of designing the one algorithm or method that handles all problems better than any other approach. Instead we aim to develop a suite of tools that efficiently handle as wide a range of problem types as possible, and gain a deep understanding of the mathematical and numerical properties of these problems and methods in order to match a problem to the method most likely to succeed efficiently. This philosophy appears time and time again, just as a method like LSQR can handle a linear system, yet is likely to be less effective than a specialized method like CG when said linear system is SPD.

This work is no different: Fletcher's penalty function demonstrates promise for constrained optimization problems satisfying a certain paradigm, but is by no means meant to be the one method to rule them all. However, several of the ideas used throughout this work can be applied to more general research in optimization; for example, exploiting inexactness, explicit parametrization of dual variables (such as in extended nonlinear programming (Rockafellar, 2000)), and exploring further applications of smooth exact penalty functions.

There remain several limitations of our approach, and addressing them is the subject of future work. One property of ϕ_σ observed from our experiments is that it is highly nonlinear and nonconvex; this appears as a root cause of many limitations. Even though superlinear or quadratic local convergence can be achieved, the high nonlinearity potentially results in many globalization iterations and small step-sizes. Some extensions include: developing a robust update for the penalty parameter; improving the solution of the trust-region subproblem; developing robust tolerance rules for solving the required linear systems; and relaxing (A2a) to weaker constraint qualifications. We discuss some of these ideas in detail below.

Updating the penalty parameter

As previously mentioned, we do not have a robust approach for updating the penalty parameter that ensures global convergence and protects iterates from stalling or diverging (ϕ_σ can be unbounded for all σ; $x^\star$ is only guaranteed to be a local minimum). This is especially important because the threshold parameter is often not known a priori. We currently use a heuristic that often works in practice, but not in all cases. Mukai and Polak (1975) develop a scheme for updating the penalty parameter for a variant of Fletcher's penalty function, which would be a helpful starting point for developing robust penalty parameter update rules.

Use as a merit function

One possibility is to use ϕ_σ as a merit function (Nocedal and Wright, 2006, §15.4) in conjunction with a constrained optimization method (e.g., an SQP method), rather than minimizing it directly. Using ϕ_σ as a merit function in a linesearch is probably inefficient because of repeated evaluation of ϕ_σ at different points to determine a stepsize. However, pure trust-region methods using a merit function to evaluate trial points may show promise as long as the constrained method guarantees a descent direction for a sufficiently small trust-region radius. Determining trial points using traditional constrained models (that require evaluation and derivatives of f and c only) and using ϕ_σ only to evaluate trial point quality may be more efficient than solving subproblems that use derivatives of ϕ_σ.

Connections with Sequential Linear Programming

For equality-constrained problems, the definition of the dual multiplier estimate (9.1b) is closely related to Sequential Linear Programming (SLP) (Palacios-Gomez, Lasdon, and Engquist, 1982). At the current iterate x_k, SLP solves the following linearized subproblem to determine a search direction:

$$\min_d \; f(x_k) + g(x_k)^T d \qquad \text{subject to} \qquad A(x_k)^T d + c(x_k) = 0, \;\; \|d\| \leqslant \Delta,$$

where $\Delta > 0$ ensures that the solution is bounded. The dual problem to the multiplier estimate problem (9.1b) is equivalent to

$$\min_{d} \ \tfrac{\sigma}{2}\|d\|^2 + g(x_k)^T d \qquad \text{subject to} \qquad A(x_k)^T d + c(x_k) = 0.$$

Further, the solution of the dual problem is $d = \sigma g_\sigma(x_k)$, the (scaled) gradient of the Lagrangian evaluated at x_k and $y_\sigma(x_k)$. These subproblems are nearly identical, and so it may be possible to use ϕ_σ as a merit function for SLP, if for example it can be shown that the the solution d of either problems is a descent direction for ϕ_σ.

Solving the trust-region subproblem

One major improvement would be developing a preconditioning approach for the trust-region subproblem. This is particularly nontrivial because the (approximate) Hessian is available only as an operator. Traditional approaches based on incomplete factorizations (Lin and Moré, 1999a) are not applicable. One possibility is to use a preconditioner for the Lagrangian Hessian H_σ as a preconditioner for the Hessian approximations B_i (9.17). This may be effective when $m \ll n$ because H_σ and B_i would differ only by a low-rank update; otherwise H_σ can be a poor approximation to B_i.

Further, products with B_i (9.17) are generally more expensive than typical Hessian-vector products, as they require solving a linear system. Products with a quasi-Newton approximation would be significantly faster. Also, exact curvature away from the solution may be less important than near the solution for choosing good directions; therefore a hybrid scheme that begins with quasi-Newton and switches to B_i may be effective. Another strategy, similar to Morales and Nocedal (2000), is to use quasi-Newton approximations to precondition the trust-region subproblems involving B_i. The approximation for iteration k can be updated with every $B_i(x_{k-1})$ product, or with every update step $x_k - x_{k-1}$.

Inexact Hessian-product trust-region method

Currently there does not exist a Newton-CG trust-region method that allows for variable-accuracy Hessian-vector products and can guarantee superlinear or quadratic convergence. Such a method would be similar to the inexact Krylov subspace solvers of (Simoncini and Szyld, 2003; van den Eshof and Sleijpen, 2004). It would be a useful kernel for inexact optimization in general, and particularly helpful to solve (PP). The dominant cost in minimizing ϕ_σ is the repeated products with the Hessian approximation B_i (9.17); each of these products requires solving two linear systems, and solving these systems inexactly results in an inexact Hessian-vector product. Therefore, a trust-region solver that can prescribe the accuracy for each Hessian-vector product while guaranteeing fast asymptotic convergence would substantially improve the performance of this approach.

Appendix A

An unstable SYMMLQ implementation

Just for fun, we introduce a linesearch-based implementation of SYMMLQ in Algorithm 11 that looks similar to the 7-line CG implementation (Hestenes and Stiefel, 1952) and CR (Stiefel, 1955). We have not found such an implementation elsewhere in literature. Although this version looks deceptively simple, it is numerically unstable and useless in practice.

Algorithm 11 Unstable SYMMLQ

```
Require: $A$, $b$
1: $x_0^L = 0$, $r_0 = w_0 = b$
2: $\delta_1 w_1 = Ab$
3: for $k = 1, \dots$ do
4:     $\zeta_k = \frac{r_{k-1}^T w_{k-1}}{\delta_k}$
5:     $x_k^L = x_{k-1} + \zeta_k w_k$
6:     $r_k = r_{k-1} - \zeta_k A w_k$
7:     if $k = 1$ then
8:         $\gamma_k = w_k^T A w_k$
9:         $\delta_{k+1} w_{k+1} = A w_k - \gamma_k w_k$
10:    else
11:        $\gamma_k = w_k^T A w_k$
12:        $\delta_{k+1} w_{k+1} = A w_k - \gamma_k w_k - \delta_k w_{k-1}$
13:    end if
14: end for
```

We can derive Algorithm 11 (in exact arithmetic) from the definition of SYMMLQ iterates:

$$x_k^L := \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \; \|x^\star - x\|^2, \;\text{ with } x \in \mathcal{K}_k(A, Ab).$$

Given an orthonormal basis $W_k = \begin{bmatrix} w_1 & \cdots & w_k \end{bmatrix}$ for $\mathcal{K}_k(A, Ab)$ and defining $x_k^L = W_k z_k = \sum_{j=1}^k \zeta_j w_j$ (where $z_k = (\zeta_1, \dots, \zeta_k)$), we have that for all $1 \leqslant k \leqslant n$:

1. $\zeta_k = w_k^T x^\star$,

2. $w_j^T x_k^L = 0$ for $j > k$, and therefore $w_j^T (x^\star - x_k^L) = 0$ for $j \leqslant k$.

To construct the orthonormal basis W_k, we apply the Lanczos process (Algorithm 1) to A with starting vector Ab. This produces the relation

$$AW_k = W_{k+1} \begin{bmatrix} \gamma_1 & \delta_2 & & \\ \delta_2 & \gamma_2 & \ddots & \\ & \ddots & \ddots & \delta_k \\ & & \delta_k & \gamma_k \\ & & & \delta_{k+1} \end{bmatrix}, \qquad W_k^T W_k = I, \; \delta_1 w_1 = Ab.$$

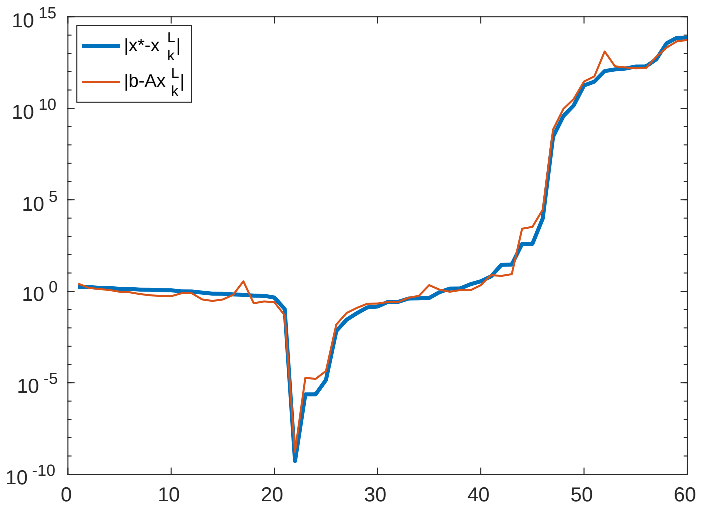


Figure A.1: Performance of Algorithm 11 on a small random example. After reaching a reasonably accurate solution, the method diverges.

Thus lines 2–3 and 8–13 of Algorithm 11 describe this Lanczos procedure, which produce the search directions w_k. Lines 6–7 update the iterate x_k^L and the current residual using the search directions and steplength. For $k \geqslant 2$ ($k = 1$ is a special case), by using the orthonormality of the w_k's, we can express the steplength as

$$\begin{aligned}\zeta_k = w_k^T x^\star &= w_k^T (x^\star - x_{k-1}^L) \\ &= \tfrac{1}{\delta_k} (A w_{k-1} - \gamma_{k-1} w_{k-1} - \delta_{k-1} w_{k-2})^T (x^\star - x_{k-1}^L) \\ &= \tfrac{1}{\delta_k} w_{k-1}^T A (x^\star - x_{k-1}^L) \\ &= \tfrac{w_{k-1}^T r_{k-1}}{\delta_k},\end{aligned}$$

which matches the expression on line 5.

Even though Algorithm 11 is mathematically equivalent to SYMMLQ, it is numerically unstable. This is best illustrated with a small example using a small linear system. We take a random 20×20 symmetric matrix A and random right-hand side b, and plot the error norm (with $x^\star = A \backslash b$ in Matlab) and residual norm at every iteration in Fig. A.1. After obtaining a reasonable accurate iterate, the method diverges. This instability likely stems from the failure of result 2 above. Previous search directions are not orthogogonal with the current error, yet this is explicitly invoked to derive the expression for ζ_k. Perhaps it is possible to modify Algorithm 11 so that it remains stable while keeping the implementation simple, but for now this algorithm simply remains an intellectual curiousity.

Appendix B

Empirical Check of SYMMLQ and CG Error Bounds

Table B.1: Empirical check on SYMMLQ error bounds using 140 SPD problems from UFL Sparse Matrix collection. Each run solves $Ax = \mathbb{1}/\sqrt{n}$ until $\epsilon_k^C \leqslant 10^{-10}\|x_\star\|$, where $x_\star$ is computed via Matlab's backslash operator. The second column gives the size of the matrix. The third column gives the total number of iterations. For each $\mu \in \{1-10^{-10}, 0.1\}$, we give two columns; the first indicates the number of iterations where $\epsilon_k^L \leqslant \|x_\star - x_k^L\|$, and the second is the fraction between the number of such iterations and the total number of iterations. The final column gives the condition number of the matrix.

Problem Name	n	Tot. Iter.	$\mu = 1 - 10^{-10}$		$\mu = 0.1$		$\kappa(A)$
ACUSIM/Pres_Poisson	14822	1784	0	0.000	0	0.000	2.00E+06
Bai/mhd3200b	3200	12800	0	0.000	0	0.000	1.60E+13
Bai/mhd4800b	4800	19200	0	0.000	0	0.000	8.20E+13
Bai/mhdb416	416	1664	0	0.000	0	0.000	4.00E+09
Bates/Chem97ZtZ	2541	150	0	0.000	0	0.000	2.50E+02
Bindel/ted_B	10605	544	0	0.000	0	0.000	1.90E+07
Bindel/ted_B_unscaled	10605	16	0	0.000	0	0.000	1.30E+11
Boeing/bcsstk34	588	1719	0	0.000	0	0.000	2.80E+04
Boeing/bcsstk36	23052	92208	0	0.000	0	0.000	7.40E+11
Boeing/crystm01	4875	7	0	0.000	0	0.000	2.30E+02
Boeing/crystm02	13965	5	0	0.000	0	0.000	2.50E+02
Boeing/crystm03	24696	5	0	0.000	0	0.000	2.60E+02
Boeing/msc00726	726	2351	287	0.122	177	0.075	4.20E+05
Boeing/msc01050	1050	4200	0	0.000	0	0.000	4.60E+15
Boeing/msc01440	1440	5760	0	0.000	0	0.000	3.30E+06
Boeing/msc04515	4515	7302	1685	0.230	1459	0.199	2.30E+06
Boeing/msc10848	10848	43392	0	0.000	0	0.000	1.00E+10
Boeing/msc23052	23052	92208	0	0.000	0	0.000	7.40E+11
Cannizzo/sts4098	4098	16392	0	0.000	0	0.000	2.20E+08
Cylshell/s1rmq4m1	5489	8667	0	0.000	0	0.000	1.80E+06
Cylshell/s1rmt3m1	5489	9348	0	0.000	0	0.000	2.50E+06
Cylshell/s2rmq4m1	5489	21729	0	0.000	0	0.000	1.80E+08
Cylshell/s2rmt3m1	5489	21956	8314	0.378	0	0.000	2.50E+08
Cylshell/s3rmq4m1	5489	21956	1050	0.047	0	0.000	1.80E+10
Cylshell/s3rmt3m1	5489	21956	0	0.000	0	0.000	2.50E+10
Cylshell/s3rmt3m3	5357	21428	0	0.000	0	0.000	2.40E+10
FIDAP/ex10	2410	9640	0	0.000	0	0.000	9.10E+11
FIDAP/ex10hs	2548	10192	0	0.000	0	0.000	5.50E+11
FIDAP/ex13	2568	10272	0	0.000	0	0.000	1.10E+15
FIDAP/ex15	6867	27468	0	0.000	0	0.000	8.60E+12
FIDAP/ex3	1821	7284	0	0.000	0	0.000	1.70E+10
FIDAP/ex33	1733	6932	0	0.000	0	0.000	7.00E+12
FIDAP/ex5	27	75	3	0.040	0	0.000	6.60E+07
FIDAP/ex9	3363	13452	0	0.000	0	0.000	1.20E+13
HB/1138_bus	1138	2479	0	0.000	0	0.000	8.60E+06

HB/494_bus	494	1425	0	0.000	0	0.000	2.40E+06
HB/662_bus	662	630	0	0.000	0	0.000	7.90E+05
HB/685_bus	685	632	0	0.000	0	0.000	4.20E+05
HB/bcsstk01	48	192	22	0.114	19	0.098	8.80E+05
HB/bcsstk02	66	48	0	0.000	0	0.000	4.30E+03
HB/bcsstk03	112	448	0	0.000	0	0.000	6.80E+06
HB/bcsstk04	132	528	0	0.000	0	0.000	2.30E+06
HB/bcsstk05	153	313	0	0.000	0	0.000	1.40E+04
HB/bcsstk06	420	1680	0	0.000	0	0.000	7.60E+06
HB/bcsstk07	420	1680	0	0.000	0	0.000	7.60E+06
HB/bcsstk08	1074	4296	0	0.000	0	0.000	2.60E+07
HB/bcsstk09	1083	311	23	0.073	17	0.054	9.50E+03
HB/bcsstk10	1086	4344	0	0.000	0	0.000	5.20E+05
HB/bcsstk11	1473	5892	0	0.000	0	0.000	2.20E+08
HB/bcsstk12	1473	5892	0	0.000	0	0.000	2.20E+08
HB/bcsstk13	2003	8012	0	0.000	0	0.000	1.10E+10
HB/bcsstk14	1806	7224	0	0.000	0	0.000	1.20E+10
HB/bcsstk15	3948	15792	0	0.000	0	0.000	6.50E+09
HB/bcsstk16	4884	731	81	0.110	42	0.057	4.90E+09
HB/bcsstk17	10974	38912	4622	0.118	3728	0.095	1.30E+10
HB/bcsstk18	11948	47792	0	0.000	0	0.000	3.50E+11
HB/bcsstk19	817	3268	0	0.000	0	0.000	1.30E+11
HB/bcsstk20	485	1940	0	0.000	0	0.000	3.90E+12
HB/bcsstk21	3600	14400	942	0.065	641	0.044	1.80E+07
HB/bcsstk22	138	460	0	0.000	0	0.000	1.10E+05
HB/bcsstk23	3134	12536	0	0.000	0	0.000	2.60E+12
HB/bcsstk24	3562	14248	0	0.000	0	0.000	1.90E+11
HB/bcsstk25	15439	61756	0	0.000	0	0.000	4.40E+12
HB/bcsstk26	1922	7688	0	0.000	0	0.000	1.70E+08
HB/bcsstk27	1224	1890	0	0.000	0	0.000	2.40E+04
HB/bcsstk28	4410	17640	3283	0.186	0	0.000	9.50E+08
HB/bcsstm02	66	11	0	0.000	0	0.000	8.80E+00
HB/bcsstm05	153	19	0	0.000	0	0.000	1.30E+01
HB/bcsstm06	420	136	0	0.000	0	0.000	3.50E+06
HB/bcsstm07	420	449	0	0.000	0	0.000	7.60E+03
HB/bcsstm08	1074	260	26	0.100	0	0.000	8.30E+06
HB/bcsstm09	1083	1	0	0.000	0	0.000	1.00E+04
HB/bcsstm11	1473	27	0	0.000	0	0.000	1.20E+05
HB/bcsstm12	1473	2899	0	0.000	0	0.000	6.30E+05
HB/bcsstm19	817	872	0	0.000	0	0.000	2.30E+05
HB/bcsstm20	485	548	0	0.000	0	0.000	2.60E+05
HB/bcsstm21	3600	2	0	0.000	0	0.000	2.40E+01
HB/bcsstm22	138	49	0	0.000	0	0.000	9.40E+02
HB/bcsstm23	3134	6533	0	0.000	0	0.000	9.50E+08
HB/bcsstm24	3562	14248	0	0.000	0	0.000	1.80E+13
HB/bcsstm25	15439	61756	0	0.000	0	0.000	6.10E+09
HB/bcsstm26	1922	2084	0	0.000	0	0.000	2.60E+05
HB/gr_30_30	900	41	0	0.000	0	0.000	1.90E+02
HB/lund_a	147	474	128	0.270	58	0.122	2.80E+06
HB/lund_b	147	442	0	0.000	0	0.000	3.00E+04
HB/nos1	237	948	0	0.000	0	0.000	2.00E+07
HB/nos2	957	3828	0	0.000	0	0.000	5.10E+09
HB/nos3	960	265	0	0.000	0	0.000	3.80E+04
HB/nos4	100	78	0	0.000	0	0.000	1.60E+03
HB/nos5	468	506	0	0.000	0	0.000	1.10E+04
HB/nos6	675	1657	0	0.000	0	0.000	7.70E+06
HB/nos7	729	2916	562	0.192	0	0.000	2.40E+09

HB/plat362	362	511	0	0.000	0	0.000	2.20E+11
JGD_Trefethen/Trefethen_150	150	118	0	0.000	0	0.000	7.70E+02
JGD_Trefethen/Trefethen_20	20	19	0	0.000	0	0.000	6.30E+01
JGD_Trefethen/Trefethen_200	200	143	0	0.000	0	0.000	1.10E+03
JGD_Trefethen/Trefethen_2000	2000	563	0	0.000	0	0.000	1.60E+04
JGD_Trefethen/Trefethen_20000	20000	2055	0	0.000	0	0.000	2.00E+05
JGD_Trefethen/Trefethen_20000b	19999	1907	0	0.000	0	0.000	9.60E+04
JGD_Trefethen/Trefethen_200b	199	133	0	0.000	0	0.000	5.20E+02
JGD_Trefethen/Trefethen_20b	19	18	0	0.000	0	0.000	3.00E+01
JGD_Trefethen/Trefethen_300	300	185	0	0.000	0	0.000	1.80E+03
JGD_Trefethen/Trefethen_500	500	251	0	0.000	0	0.000	3.20E+03
JGD_Trefethen/Trefethen_700	700	306	0	0.000	0	0.000	4.70E+03
Lourakis/bundle1	10581	376	40	0.106	17	0.045	1.00E+03
MathWorks/Kuu	7102	559	0	0.000	0	0.000	1.60E+04
MathWorks/Muu	7102	50	0	0.000	0	0.000	7.70E+01
Nasa/nasa1824	1824	7296	0	0.000	0	0.000	1.90E+06
Nasa/nasa2146	2146	603	4	0.006	0	0.000	1.70E+03
Nasa/nasa2910	2910	11640	0	0.000	0	0.000	6.00E+06
Nasa/nasa4704	4704	18816	0	0.000	0	0.000	4.20E+07
ND/nd3k	9000	5046	0	0.000	0	0.000	1.60E+07
ND/nd6k	18000	6211	0	0.000	0	0.000	1.60E+07
Norris/fv1	9604	31	0	0.000	0	0.000	8.80E+00
Norris/fv2	9801	31	0	0.000	0	0.000	8.80E+00
Norris/fv3	9801	121	0	0.000	0	0.000	2.00E+03
Oberwolfach/gyro	17361	48763	19065	0.390	14011	0.287	1.10E+09
Oberwolfach/gyro_k	17361	48763	19065	0.390	14011	0.287	1.10E+09
Oberwolfach/gyro_m	17361	1096	0	0.000	0	0.000	2.50E+06
Oberwolfach/LF10	18	45	0	0.000	0	0.000	3.90E+06
Oberwolfach/LFAT5	14	30	4	0.133	0	0.000	1.40E+08
Oberwolfach/t2dah_e	11445	16763	0	0.000	0	0.000	7.20E+08
Oberwolfach/t2dal_e	4257	693	0	0.000	0	0.000	3.80E+07
Oberwolfach/t3dl_e	20360	72	0	0.000	0	0.000	6.00E+03
Pajek/Journals	124	200	0	0.000	0	0.000	9.80E+03
Pothen/bodyy4	17546	296	0	0.000	0	0.000	8.10E+02
Pothen/bodyy5	18589	929	0	0.000	0	0.000	7.90E+03
Pothen/bodyy6	19366	2874	0	0.000	0	0.000	7.70E+04
Pothen/mesh1e1	48	23	0	0.000	0	0.000	5.20E+00
Pothen/mesh1em1	48	38	0	0.000	0	0.000	1.90E+01
Pothen/mesh1em6	48	23	0	0.000	0	0.000	6.10E+00
Pothen/mesh2e1	306	133	0	0.000	0	0.000	2.90E+02
Pothen/mesh2em5	306	98	0	0.000	0	0.000	2.50E+02
Pothen/mesh3e1	289	28	0	0.000	0	0.000	8.90E+00
Pothen/mesh3em5	289	18	0	0.000	0	0.000	5.00E+00
Simon/olafu	16146	64584	0	0.000	0	0.000	7.60E+11
Simon/raefsky4	19779	79116	0	0.000	0	0.000	3.10E+13
TKK/cbuckle	13681	7152	0	0.000	0	0.000	3.30E+07
TKK/plbuckle	1282	2279	0	0.000	0	0.000	1.30E+06
UTEP/Dubcova1	16129	103	0	0.000	0	0.000	1.00E+03

Table B.2: Empirical check on CG error bounds using 140 SPD problems from UFL Sparse Matrix collection. Each run solves $Ax = \mathbb{1}/\sqrt{n}$ until $\epsilon_k^C \leqslant 10^{-10}\|x_\star\|$, where $x_\star$ is computed via Matlab's backslash operator. The second column gives the size of the matrix. The third column gives the total number of iterations. For each $\mu \in \{1 - 10^{-10}, 0.1\}$, we give two columns; the first indicates the number of iterations where $\epsilon_k^C \leqslant \|x_\star - x_k^C\|$, and the second is the fraction between the number of such iterations and the total number of iterations. The final column gives the condition number of the matrix.

Problem Name	n	Tot. Iter.	$\mu = 1 - 10^{-10}$		$\mu = 0.1$		$\kappa(A)$
ACUSIM/Pres_Poisson	14822	1784	0	0.000	0	0.000	2.0E+06
Bai/mhd3200b	3200	12800	0	0.000	0	0.000	1.6E+13
Bai/mhd4800b	4800	19200	0	0.000	0	0.000	8.2E+13
Bai/mhdb416	416	1664	0	0.000	0	0.000	4.0E+09
Bates/Chem97ZtZ	2541	150	0	0.000	0	0.000	2.5E+02
Bindel/ted_B	10605	544	0	0.000	0	0.000	1.9E+07
Bindel/ted_B_unscaled	10605	16	0	0.000	0	0.000	1.3E+11
Boeing/bcsstk34	588	1719	0	0.000	0	0.000	2.8E+04
Boeing/bcsstk36	23052	92208	0	0.000	0	0.000	7.4E+11
Boeing/crystm01	4875	7	0	0.000	0	0.000	2.3E+2
Boeing/crystm02	13965	5	0	0.000	0	0.000	2.5E+2
Boeing/crystm03	24696	5	0	0.000	0	0.000	2.6E+2
Boeing/msc00726	726	2351	262	0.111	175	0.074	4.2E+05
Boeing/msc01050	1050	4200	0	0.000	0	0.000	4.6E+15
Boeing/msc01440	1440	5760	0	0.000	0	0.000	3.3E+06
Boeing/msc04515	4515	7302	1687	0.231	1466	0.201	2.3E+06
Boeing/msc10848	10848	43392	0	0.000	0	0.000	1.0E+10
Boeing/msc23052	23052	92208	0	0.000	0	0.000	7.4E+11
Cannizzo/sts4098	4098	16392	0	0.000	0	0.000	2.2E+08
Cylshell/s1rmq4m1	5489	8667	0	0.000	0	0.000	1.8E+06
Cylshell/s1rmt3m1	5489	9348	0	0.000	0	0.000	2.5E+06
Cylshell/s2rmq4m1	5489	21729	0	0.000	0	0.000	1.8E+08
Cylshell/s2rmt3m1	5489	21956	3078	0.140	0	0.000	2.5E+08
Cylshell/s3rmq4m1	5489	21956	0	0.000	0	0.000	1.8E+10
Cylshell/s3rmt3m1	5489	21956	0	0.000	0	0.000	2.5E+10
Cylshell/s3rmt3m3	5357	21428	0	0.000	0	0.000	2.4E+10
FIDAP/ex10	2410	9640	0	0.000	0	0.000	9.1E+11
FIDAP/ex10hs	2548	10192	0	0.000	0	0.000	5.5E+11
FIDAP/ex13	2568	10272	0	0.000	0	0.000	1.1E+15
FIDAP/ex15	6867	27468	0	0.000	0	0.000	8.6E+12
FIDAP/ex3	1821	7284	0	0.000	0	0.000	1.7E+10
FIDAP/ex33	1733	6932	0	0.000	0	0.000	7.0E+12
FIDAP/ex5	27	75	2	0.027	0	0.000	6.6E+07
FIDAP/ex9	3363	13452	0	0.000	0	0.000	1.2E+13
HB/1138_bus	1138	2479	0	0.000	0	0.000	8.6E+06
HB/494_bus	494	1425	0	0.000	0	0.000	2.4E+06
HB/662_bus	662	630	0	0.000	0	0.000	7.9E+05
HB/685_bus	685	632	0	0.000	0	0.000	4.2E+05
HB/bcsstk01	48	192	22	0.115	19	0.099	8.8E+05
HB/bcsstk02	66	48	0	0.000	0	0.000	4.3E+03
HB/bcsstk03	112	448	0	0.000	0	0.000	6.8E+06
HB/bcsstk04	132	528	0	0.000	0	0.000	2.3E+06
HB/bcsstk05	153	313	0	0.000	0	0.000	1.4E+04
HB/bcsstk06	420	1680	0	0.000	0	0.000	7.6E+06
HB/bcsstk07	420	1680	0	0.000	0	0.000	7.6E+06
HB/bcsstk08	1074	4296	0	0.000	0	0.000	2.6E+07
HB/bcsstk09	1083	311	38	0.122	17	0.055	9.5E+03

HB/bcsstk10	1086	4344	0	0.000	0	0.000	5.2E+05
HB/bcsstk11	1473	5892	0	0.000	0	0.000	2.2E+08
HB/bcsstk12	1473	5892	0	0.000	0	0.000	2.2E+08
HB/bcsstk13	2003	8012	0	0.000	0	0.000	1.1E+10
HB/bcsstk14	1806	7224	0	0.000	0	0.000	1.2E+10
HB/bcsstk15	3948	15792	0	0.000	0	0.000	6.5E+09
HB/bcsstk16	4884	731	223	0.305	43	0.059	4.9E+09
HB/bcsstk17	10974	38912	6918	0.178	3737	0.096	1.3E+10
HB/bcsstk18	11948	47792	0	0.000	0	0.000	3.5E+11
HB/bcsstk19	817	3268	0	0.000	0	0.000	1.3E+11
HB/bcsstk20	485	1940	0	0.000	0	0.000	3.9E+12
HB/bcsstk21	3600	14400	923	0.064	616	0.043	1.8E+07
HB/bcsstk22	138	460	0	0.000	0	0.000	1.1E+05
HB/bcsstk23	3134	12536	0	0.000	0	0.000	2.6E+12
HB/bcsstk24	3562	14248	0	0.000	0	0.000	1.9E+11
HB/bcsstk25	15439	61756	0	0.000	0	0.000	4.4E+12
HB/bcsstk26	1922	7688	0	0.000	0	0.000	1.7E+08
HB/bcsstk27	1224	1890	0	0.000	0	0.000	2.4E+04
HB/bcsstk28	4410	17640	515	0.029	0	0.000	9.5E+08
HB/bcsstm02	66	11	0	0.000	0	0.000	8.8E+00
HB/bcsstm05	153	19	0	0.000	0	0.000	1.3E+01
HB/bcsstm06	420	136	0	0.000	0	0.000	3.5E+06
HB/bcsstm07	420	449	0	0.000	0	0.000	7.6E+03
HB/bcsstm08	1074	260	15	0.058	0	0.000	8.3E+06
HB/bcsstm09	1083	1	0	0.000	0	0.000	1.0E+04
HB/bcsstm11	1473	27	0	0.000	0	0.000	1.2E+05
HB/bcsstm12	1473	2899	0	0.000	0	0.000	6.3E+05
HB/bcsstm19	817	872	3	0.003	0	0.000	2.3E+05
HB/bcsstm20	485	548	0	0.000	0	0.000	2.6E+05
HB/bcsstm21	3600	2	0	0.000	0	0.000	2.4E+01
HB/bcsstm22	138	49	0	0.000	0	0.000	9.4E+02
HB/bcsstm23	3134	6533	0	0.000	0	0.000	9.5E+08
HB/bcsstm24	3562	14248	0	0.000	0	0.000	1.8E+13
HB/bcsstm25	15439	61756	0	0.000	0	0.000	6.1E+09
HB/bcsstm26	1922	2084	0	0.000	0	0.000	2.6E+05
HB/gr_30_30	900	41	0	0.000	0	0.000	1.9E+02
HB/lund_a	147	474	134	0.283	59	0.124	2.8E+06
HB/lund_b	147	442	0	0.000	0	0.000	3.0E+04
HB/nos1	237	948	0	0.000	0	0.000	2.0E+07
HB/nos2	957	3828	0	0.000	0	0.000	5.1E+09
HB/nos3	960	265	0	0.000	0	0.000	3.8E+04
HB/nos4	100	78	0	0.000	0	0.000	1.6E+03
HB/nos5	468	506	0	0.000	0	0.000	1.1E+04
HB/nos6	675	1657	0	0.000	0	0.000	7.7E+06
HB/nos7	729	2916	294	0.101	0	0.000	2.4E+09
HB/plat362	362	511	0	0.000	0	0.000	2.2E+11
JGD_Trefethen/Trefethen_150	150	118	0	0.000	0	0.000	7.7E+02
JGD_Trefethen/Trefethen_20	20	19	0	0.000	0	0.000	6.3E+01
JGD_Trefethen/Trefethen_200	200	143	0	0.000	0	0.000	1.1E+03
JGD_Trefethen/Trefethen_2000	2000	563	0	0.000	0	0.000	1.6E+04
JGD_Trefethen/Trefethen_20000	20000	2055	0	0.000	0	0.000	2.0E+05
JGD_Trefethen/Trefethen_20000b	19999	1907	0	0.000	0	0.000	9.6E+04
JGD_Trefethen/Trefethen_200b	199	133	0	0.000	0	0.000	5.2E+02
JGD_Trefethen/Trefethen_20b	19	18	0	0.000	0	0.000	3.0E+01
JGD_Trefethen/Trefethen_300	300	185	0	0.000	0	0.000	1.8E+03
JGD_Trefethen/Trefethen_500	500	251	0	0.000	0	0.000	3.2E+03
JGD_Trefethen/Trefethen_700	700	306	0	0.000	0	0.000	4.7E+03

Lourakis/bundle1	10581	376	35	0.093	17	0.045	1.0E+03
MathWorks/Kuu	7102	559	0	0.000	0	0.000	1.6E+04
MathWorks/Muu	7102	50	0	0.000	0	0.000	7.7E+01
Nasa/nasa1824	1824	7296	0	0.000	0	0.000	1.9E+06
Nasa/nasa2146	2146	603	1	0.002	0	0.000	1.7E+03
Nasa/nasa2910	2910	11640	0	0.000	0	0.000	6.0E+06
Nasa/nasa4704	4704	18816	0	0.000	0	0.000	4.2E+07
ND/nd3k	9000	5046	0	0.000	0	0.000	1.6E+07
ND/nd6k	18000	6211	0	0.000	0	0.000	1.6E+07
Norris/fv1	9604	31	0	0.000	0	0.000	8.8E+00
Norris/fv2	9801	31	0	0.000	0	0.000	8.8E+00
Norris/fv3	9801	121	0	0.000	0	0.000	2.0E+03
Oberwolfach/gyro	17361	48763	18101	0.371	13997	0.287	1.1E+09
Oberwolfach/gyro_k	17361	48763	18101	0.371	13997	0.287	1.1E+09
Oberwolfach/gyro_m	17361	1096	0	0.000	0	0.000	2.5E+06
Oberwolfach/LF10	18	45	0	0.000	0	0.000	3.9E+06
Oberwolfach/LFAT5	14	30	2	0.067	0	0.000	1.4E+08
Oberwolfach/t2dah_e	11445	16763	0	0.000	0	0.000	7.2E+08
Oberwolfach/t2dal_e	4257	693	0	0.000	0	0.000	3.8E+07
Oberwolfach/t3dl_e	20360	72	0	0.000	0	0.000	6.0E+03
Pajek/Journals	124	200	0	0.000	0	0.000	9.8E+03
Pothen/bodyy4	17546	296	0	0.000	0	0.000	8.1E+02
Pothen/bodyy5	18589	929	0	0.000	0	0.000	7.9E+03
Pothen/bodyy6	19366	2874	0	0.000	0	0.000	7.7E+04
Pothen/mesh1e1	48	23	0	0.000	0	0.000	5.2E+00
Pothen/mesh1em1	48	38	0	0.000	0	0.000	1.9E+01
Pothen/mesh1em6	48	23	0	0.000	0	0.000	6.1E+00
Pothen/mesh2e1	306	133	0	0.000	0	0.000	2.9E+02
Pothen/mesh2em5	306	98	0	0.000	0	0.000	2.5E+02
Pothen/mesh3e1	289	28	0	0.000	0	0.000	8.9E+00
Pothen/mesh3em5	289	18	0	0.000	0	0.000	5.0E+00
Simon/olafu	16146	64584	0	0.000	0	0.000	7.6E+11
Simon/raefsky4	19779	79116	0	0.000	0	0.000	3.1E+13
TKK/cbuckle	13681	7152	0	0.000	0	0.000	3.3E+07
TKK/plbuckle	1282	2279	0	0.000	0	0.000	1.3E+06
UTEP/Dubcova1	16129	103	0	0.000	0	0.000	1.0E+03

Appendix C

Direct methods for augmented systems

We describe various direct methods for solving (10.16)

$$\begin{bmatrix} I & A \\ A^T & -\delta^2 I \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} w \\ z \end{bmatrix}, \qquad \text{where} \qquad \mathcal{K} = \begin{bmatrix} I & A \\ A^T & -\delta^2 I \end{bmatrix}, \tag{C.1}$$

required to evaluate ϕ_σ and its derivatives. Recall that $A_\delta = \begin{bmatrix} A^T & \delta I \end{bmatrix}^T$ when $\delta > 0$; otherwise $A_\delta = A$. For this section, given a matrix R and vector b, the shorthand notation $x \leftarrow R\backslash b$ means that x solves the system $Rx = b$ (via forward and/or backward substitution).

QR factorization

Algorithm 12 computes (p, q) using the thin QR factorization of $A_\delta = QR$, with Q orthogonal and $R \in \mathbb{R}^{m \times m}$.

Algorithm 12 Solving (10.16) using the QR factorization.

```
1: $Q, R \leftarrow \mathrm{qr}(A_\delta)$
2: $\bar{w} \leftarrow Q_{1:n,:}^T w$
3: $\bar{z} \leftarrow R^T \backslash z$
4: $p \leftarrow w - Q_{1:n,:}(\bar{w} - \bar{z})$
5: $q \leftarrow R\backslash(\bar{w} - \bar{z})$
6: **return** $(p, q)$
```

The advantage is that A_δ is factorized instead of $\mathcal{K}$, and this method is backward stable for both p and q (Golub and Van Loan, 2013, §5.3.6). If A_δ is sparse, R is likely to be sparse (for some column permutation of A_δ) but unfortunately Q is not. For large problems, it may not be practical to store Q in order to solve (10.16).

Corrected semi-normal equations

The R factor from $A_\delta = QR$ can be computed without storing Q. We can then solve the semi-normal equations $R^T R q = A^T w - z$ and set $p = w - Aq$. Björck and Paige (1994) show that this is not acceptable-error stable for p, possibly giving large error in p, particularly when $\|p\| \ll \|w\|$. Note that $p = g_\sigma$ in (10.13) means we may obtain large errors in the gradient near the solution if care is not taken. Fortunately, Björck and Paige (1994) show that one step of iterative refinement ensures p is acceptable-error stable; see Algorithm 13.

Algorithm 13 Solving (10.16) using the semi-normal equations.

```
1: $R \leftarrow \mathrm{qr}(A_\delta)$
2: $q \leftarrow (R^TR)\backslash(A^Tw - z) \qquad p \leftarrow w - Aq$
3: $\Delta q \leftarrow (R^TR)\backslash(A^Tp - \delta^2 q - z)$        ▷ Iterative refinement
4: $q \leftarrow q + \Delta q \qquad p \leftarrow p - A\Delta q$
5: **return** $(p, q)$
```

LDL and Bunch-Kaufman factorization

When it is not practical to store Q from the QR factors of A, or the semi-normal equations do not provide sufficient accuracy, it may be possible to compute the LDL or Bunch-Kaufman factorization of $\mathcal{K}$ directly. Although an $(n+m) \times (n+m)$ matrix is factorized (rather than an $n \times m$ matrix), the entire factorization is likely to be sparse, and the solution is typically more accurate than with the semi-normal equations.

Björck (1967) and Saunders (1995) discuss scaling of the $(1,1)$ identity block to improve the condition number of $\mathcal{K}$. Saunders (1995) also considers the case where $\mathcal{K}$ is regularized with $-\delta^2 I$ in the $(2,2)$ block.

Appendix D

Proof of Theorem 9.13

We repeat the assumptions of Theorem 9.13:

(B1) (NP) satisfies (A1b) and (A2a).

(B2) $x^\star$ is a second-order KKT point for (NP) satisfying $\nabla^2\phi_\sigma(x^\star) \succ 0$.

(B3) There exist $\bar{\delta} > 0$ and an open set $B(x^\star)$ containing $x^\star$ such that if $\widetilde{x}_0 \in B(x^\star)$ and $\delta \leqslant \bar{\delta}$, the sequence $\widetilde{x}_{k+1} = \widetilde{x}_k + G(\phi_\sigma(\cdot;\delta), \widetilde{x}_k)$ converges quadratically to $x(\delta)$ with constant M independent of δ.

Lemma D.1 *Under the assumptions of Theorem 9.13:*

1. *$\phi_\sigma(\cdot;\delta)$ has two continuous derivatives for $\delta > 0$ and $x \in \mathbb{R}^n$ by (B1).*

2. *There exists an open set $B_1(x^\star)$ containing $x^\star$ such that $\phi_\sigma(x)$ is well-defined and has two continuous derivatives for all $x \in B_1(x^\star)$ by (B1).*

3. *$\nabla^2\phi_\sigma(x^\star) = \nabla^2\phi_\sigma(x^\star;0) \succ 0$ and $\phi_\sigma(x;\delta)$ is second-order smooth in both x and δ, so by assumption (B2) there exists an open set $B_2(x^\star)$ containing $x^\star$ and $\widetilde{\delta} > 0$ such that $\nabla^2\phi_\sigma(x;\delta) \succ 0$ for $(x,\delta) \in B_2(x^\star) \times [0,\widetilde{\delta}]$.*

4. *By Theorem 9.10, there exists $\hat{\delta}$ such that for $\delta \leqslant \hat{\delta}$, $x(\delta)$ is continuous in δ. Therefore there exists an open set $B_3(x^\star)$ such that $x(\delta) \in B_3(x^\star)$ for $\delta \leqslant \hat{\delta}$.*

5. *There exists a neighborhood $B_4(x^\star)$ where Newton's method is quadratically convergent (with factor N) on $\phi_\sigma(x)$ by (B2).*

6. *Given $\delta_0 \leqslant \bar{\delta}$, where $\bar{\delta}$ is defined in (B3), there exists a neighborhood $B_5(x^\star)$ such that $\|\nabla\phi_\sigma(x;\delta_0)\| \leqslant \delta_0$ for all $x \in B_5(x^\star)$.*

We define

$$B'(x^\star) := B(x^\star) \cap \left(\bigcap_{i=1}^{5} B_i(x^\star)\right) \qquad \text{and} \qquad \delta' := \min\{\bar{\delta}, \widetilde{\delta}, \hat{\delta}, 1\},$$

and note that x_k defined by Algorithm 8 satisfies $x_k \in B'(x^\star)$ for all k by (B3). Because $\phi_\sigma(x;\delta)$ is a $\mathcal{C}_2$ function in $B'(x^\star) \times [0,\delta']$, there exist positive constants $K_1,\ldots,K_5$ such that

7. *$\|\nabla\phi_\sigma(x;\delta)\| \leqslant K_1$, $\|\nabla^2\phi_\sigma(x;\delta)^{-1}\| \leqslant K_2$ for $x \in B'(x^\star)$ and $\delta \leqslant \delta'$;*

8. *$\|\nabla P_\delta(x)\| \leqslant K_3$, $\|\nabla^2 P_\delta(x)\| \leqslant K_4$ for $x \in B'(x^\star)$ and $\delta \leqslant \delta'$;*

9. *$\|x_k - x^\star\| \leqslant K_5\|\nabla\phi_\sigma(x_k)\|$.*

Proof. Statements 1–2 follow from (B1). Statements 3 and 5 follow from (B2). Statement 4 follows from Theorem 9.10.

Now consider Statement 6. For a given δ_0, we have $\nabla\phi_\sigma(x(\delta_0);\delta_0) = 0$ and so there exists a neighbourhood $\widetilde{B}$ around $x(\delta_0)$ such that $\|\nabla\phi_\sigma(x;\delta_0)\| \leqslant \delta_0$ for all $x \in \widetilde{B}$. Further, $x(\delta_0) \in B_3(x^\star)$, so let $B_5(x^\star) = \widetilde{B} \cap B_3(x^\star)$. ∎

We first give some technical results. All assume that $x_k \in B'(x^\star)$ and $\delta_k \leqslant \delta'$.

Lemma D.2 *Assume $\delta_{k-1} \leqslant \delta_0 \leqslant \delta'$. For δ_k defined according to (9.36),*

$$\|\nabla\phi_\sigma(x_k;\delta_k)\| = O(\delta_k).$$

Proof. The result holds for $k = 1$ in view of observation 6 of Lemma D.1.

Because $x_k \in B'(x^\star)$ and $\delta_k, \delta_{k-1} \leqslant \delta'$, observation 8 of Lemma D.1 gives $\|\nabla P_{\delta_{k-1}}(x_k)\| \leqslant K_3$ and $\|\nabla P_{\delta_k}(x_k)\| \leqslant K_3$. Using (9.35), we have

$$\begin{aligned}
\|\nabla\phi_\sigma(x_k;\delta_k)\| &= \|\nabla\phi_\sigma(x_k;\delta_{k-1}) - \delta_{k-1}^2\nabla P_{\delta_{k-1}}(x_k) + \delta_k^2\nabla P_{\delta_k}(x_k)\| \\
&\leqslant \|\nabla\phi_\sigma(x_k;\delta_{k-1})\| + \delta_{k-1}^2\|\nabla P_{\delta_{k-1}}(x_k)\| + \delta_k^2\|\nabla P_{\delta_k}(x_k)\| \\
&\leqslant \|\nabla\phi_\sigma(x_k;\delta_{k-1})\| + (\delta_{k-1}^2 + \delta_k^2)K_3 \\
&= \|\nabla\phi_\sigma(x_k;\delta_{k-1})\| + O(\delta_k), \qquad (D.1)
\end{aligned}$$

where the last inequality follows from $\delta_k^2 \leqslant \delta_k$ and (9.36), implying that $\delta_k \geqslant \nu(\delta_{k-1}) = \delta_{k-1}^2$.

We consider two cases. If $\|\nabla\phi_\sigma(x_k;\delta_{k-1})\| \leqslant \delta_{k-1}$, (9.36) implies that

$$\delta_k = \max(\|\nabla\phi_\sigma(x_k;\delta_{k-1})\|,\, \delta_{k-1}^2) \geqslant \|\nabla\phi_\sigma(x_k;\delta_{k-1})\|,$$

and therefore (D.1) gives $\|\nabla\phi_\sigma(x_k;\delta_k)\| = O(\delta_k)$.

Otherwise, (9.36) yields $\delta_k = \max(\delta_{k-1},\, \delta_{k-1}^2) = \delta_{k-1}$, and there exists $\ell \leqslant k-1$ such that $\delta_k = \delta_{k-1} = \cdots = \delta_\ell < \delta_{\ell-1}$, or $\ell = 1$. If $\delta_\ell < \delta_{\ell-1}$, step 3 of Algorithm 8 implies that $\|\nabla\phi_\sigma(x_\ell;\delta_{\ell-1})\| < \delta_{\ell-1}$, which by the above sequence of inequalities implies that $\|\nabla\phi_\sigma(x_\ell;\delta_\ell)\| = O(\delta_\ell)$. Then, because $\delta_k = \delta_\ell$,

$$\|\nabla\phi_\sigma(x_\ell,\delta_k)\| = \|\nabla\phi_\sigma(x_\ell,\delta_\ell)\| = O(\delta_\ell) = O(\delta_k).$$

Define the sequence $\{\widetilde{x}_j\}$ with $\widetilde{x}_0 = x_\ell$ and $\widetilde{x}_{j+1} = \widetilde{x}_j + G(\phi_\sigma(\cdot;\delta_\ell),\widetilde{x}_j)$. By (B3), $\widetilde{x}_j \to x(\delta_\ell)$ quadratically, so after j iterations of this procedure, for some $\widetilde{M}$, we have

$$\|\nabla\phi_\sigma(\widetilde{x}_j;\delta_k)\| \leqslant \widetilde{M}^j\|\nabla\phi_\sigma(\widetilde{x}_0;\delta_k)\|^{2^j} \leqslant \widetilde{M}^j K_1^{2^{j-1}}\|\nabla\phi_\sigma(\widetilde{x}_0;\delta_k)\|^2.$$

Then after $j = O(1)$ iterations of this procedure (depending only on $\widetilde{M}$ and K_1), we have $\widetilde{M}^j K_1^{2^{j-1}} \leqslant 1$, so that

$$\|\nabla\phi_\sigma(\widetilde{x}_j;\delta_k)\| \leqslant \|\nabla\phi_\sigma(\widetilde{x}_0;\delta_k)\|^2 = \|\nabla\phi_\sigma(x_\ell;\delta_k)\|^2 = O(\delta_k^2) < \delta_k.$$

Therefore, $k - \ell \leqslant j = O(1)$, and by (B3),

$$\|\nabla\phi_\sigma(x_k;\delta_{k-1})\| = O\left(M^{k-\ell}\|\nabla\phi_\sigma(x_\ell;\delta_{k-1})\|^{2^{k-\ell}}\right) = O\left(M^{k-\ell}\delta_k^{2^{k-\ell}}\right) = O(\delta_k).$$ ∎

Lemma D.3 *For p_k defined by step 4 of Algorithm 8, $\|p_k\| = O(\delta_k)$.*

Proof. According to (B3), we may apply Lemma 9.12 with $\tau = 2$ and view step 4 of Algorithm 8 as an inexact-Newton step, i.e., there exists a constant $N_2 > 0$ such that

$$\nabla^2\phi_\sigma(x_k;\delta_k)p_k = -\nabla\phi_\sigma(x_k;\delta_k) + r_k, \qquad \|r_k\| \leqslant N_2\|\nabla\phi_\sigma(x_k;\delta_k)\|^2.$$

Therefore by Lemma D.2,

$$\begin{aligned}\|p_k\| &= \|\nabla^2\phi_\sigma(x_k;\delta_k)^{-1}(-\nabla\phi_\sigma(x_k;\delta_k) + r_k)\| \\ &\leqslant \|\nabla^2\phi_\sigma(x_k;\delta_k)^{-1}\|\left(\|\nabla\phi_\sigma(x_k;\delta_k)\| + \|r_k\|\right) \\ &\leqslant K_2\left(\|\nabla\phi_\sigma(x_k;\delta_k)\| + N_2\|\nabla\phi_\sigma(x_k;\delta_k)\|^2\right) \\ &\leqslant K_2\left(O(\delta_k) + O(\delta_k^2)\right) = O(\delta_k).\end{aligned}$$ □

Lemma D.4 *Let p_k be defined by step 4 of Algorithm 8 and q_k be the Newton direction for the unregularized penalty function defined by $\nabla^2\phi_\sigma(x_k)q_k = -\nabla\phi_\sigma(x_k)$. Then $\|p_k - q_k\| \in O(\delta_k^2)$.*

Proof. According to (B3), we may apply Lemma 9.12 with $\tau = 2$ and view step 4 of Algorithm 8 as an inexact-Newton step, i.e.,

$$\nabla^2\phi_\sigma(x_k;\delta_k)p_k = -\nabla\phi_\sigma(x_k;\delta_k) + r_k, \tag{D.2a}$$
$$\|r_k\| = O(\|\nabla\phi_\sigma(x_k;\delta_k)\|^2). \tag{D.2b}$$

We premultiply (D.2a) by $\nabla^2\phi_\sigma(x_k)^{-1}$ and use (9.35) to obtain

$$p_k + \delta_k^2\nabla^2\phi_\sigma(x_k)^{-1}\nabla^2P_{\delta_k}(x_k)p_k = q_k + \delta_k^2\nabla^2\phi_\sigma(x_k)^{-1}\nabla P_{\delta_k}(x_k) + \nabla^2\phi_\sigma(x_k)^{-1}r_k.$$

Lemma D.2, Lemma D.3 and the triangle inequality then yield

$$\begin{aligned}\|p_k - q_k\| &= \left\|\delta_k^2\nabla^2\phi_\sigma(x_k)^{-1}\left(\nabla P_{\delta_k}(x_k) - \nabla^2P_{\delta_k}(x_k)p_k\right) + \nabla^2\phi_\sigma(x_k)^{-1}r_k\right\| \\ &\leqslant \delta_k^2\|\nabla^2\phi_\sigma(x_k)^{-1}\|\left(\|\nabla P_{\delta_k}(x_k)\| + \|\nabla^2P_{\delta_k}(x_k)p_k\|\right) + \|\nabla^2\phi_\sigma(x_k)^{-1}r_k\| \\ &\leqslant \delta_k^2K_2\left(K_3 + O(\delta_k)\right) + O(\delta_k^2) = O(\delta_k^2).\end{aligned}$$ □

Using the previous technical results, we are in position to establish our main result.

Proof of Theorem 9.13. We show that for $x_0 \in B'(x^\star)$ we achieve R-quadratic convergence, by showing that $\|x_k - x^\star\| = O(\delta_k)$ and that $\delta_k \to 0$ quadratically. By observation 9 of Lemma D.1, (9.35), the triangle inequality, Lemma D.2, and observation 8 of Lemma D.1, we have

$$\begin{aligned}\|x_k - x^\star\| &\leqslant K_5\|\nabla\phi_\sigma(x_k)\| \\ &= K_5\|\nabla\phi_\sigma(x_k;\delta_k) - \delta_k^2\nabla P_{\delta_k}(x_k)\| \\ &\leqslant K_5(\|\nabla\phi_\sigma(x_k;\delta_k)\| + \delta_k^2\|\nabla P_{\delta_k}(x_k)\|) \\ &\leqslant K_5\left(O(\delta_k) + \delta_k^2K_3\right) = O(\delta_k).\end{aligned}$$

Let q_k be the Newton direction defined in Lemma D.4. There exists a constant $N > 0$ such that

$$\begin{aligned}\|x_{k+1} - x^\star\| &= \|x_k + p_k - x^\star\| \\ &\leqslant \|x_k + q_k - x^\star\| + \|p_k - q_k\| \\ &\leqslant N\|x_k - x^\star\|^2 + \|p_k - q_k\| = O(\delta_k^2).\end{aligned}$$

It remains to show that δ_k decreases quadratically. If $\|\nabla\phi_\sigma(x_{k+1}, \delta_k)\| \leqslant \delta_k^2$,

$$\delta_{k+1} = \max\{\min\{\|\nabla\phi_\sigma(x_{k+1}, \delta_k)\|, \delta_k\}, \delta_k^2\} \leqslant \max\{\|\nabla\phi_\sigma(x_{k+1}, \delta_k)\|, \delta_k^2\} = \delta_k^2.$$

Assume now that $\|\nabla\phi_\sigma(x_{k+1}, \delta_k)\| > \delta_k^2$. We have from (9.35) and observations 7–8 of Lemma D.1 that

$$\begin{aligned}\delta_{k+1} &= \max\{\min\{\|\nabla\phi_\sigma(x_{k+1}, \delta_k)\|, \delta_k\}, \delta_k^2\} \\ &\leqslant \|\nabla\phi_\sigma(x_{k+1}, \delta_k)\| \\ &\leqslant \|\nabla\phi_\sigma(x_{k+1})\| + \delta_k^2\|\nabla P^2_{\delta_k}(x_{k+1})\| \\ &\leqslant K_2^{-1}\|x_{k+1} - x^\star\| + \delta_k^2 K_3 = O(\delta_k^2).\end{aligned}$$

Thus we have $\|x_k - x^\star\| = O(\delta_k)$ and $\delta_{k+1} = O(\delta_k^2)$, which means that $x_k \to x^\star$ R-quadratically. ∎

Bibliography

M. Anitescu. On solving mathematical programs with complementarity constraints as nonlinear programs. Technical Report ANL/MCS-P864-1200, Argonne National Laboratory, 2000.

M. Arioli. Generalized Golub-Kahan bidiagonalization and stopping criteria. *SIAM J. Matrix Anal. Appl.*, 34(2):571–592, 2013. doi: 10.1137/120866543.

S. Arreckx and D. Orban. A regularized factorization-free method for equality-constrained optimization. *SIAM J. Optim.*, 28(2):1613–1639, 2018. ISSN 1052-6234. doi: 10.1137/16M1088570.

G. Auchmuty. A posteriori error estimates for linear equations. *Numer. Math.*, 61(1):1–6, 1992. doi: 10.1007/BF01385494.

D. P. Bertsekas. Necessary and sufficient conditions for a penalty method to be exact. *Math. Program.*, 9:87–99, 1975.

D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York, 1982.

A. Björck. Iterative refinement of linear least squares solutions. I. *Nordisk Tidskr. Informations-Behandling (BIT)*, 7:257–278, 1967.

A. Björck and C. C. Paige. Solution of augmented linear systems using orthogonal factorizations. *BIT*, 34(1):1–24, 1994. doi: 10.1007/BF01935013.

P. T. Boggs, J. W. Tolle, and A. J. Kearsley. A merit function for inequality constrained nonlinear programming problems. Internal Report NISTIR 4702, Applied and Computational Mathematics Division, National Institute of Standards and Technology, Gaithersburg, MD, USA, 1992.

C. Brezinski. Error estimates for the solution of linear systems. *SIAM J. Sci. Comput.*, 21(2):764–781, 1999. doi: 10.1137/S1064827597328510.

R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In G. di Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, pages 35–59. Springer-Verlag, New York, 2006.

X. Chen. Smoothing methods for complementarity problems and their applications: a survey. *J. Oper. Res. Soc. Japan*, 43(1):32–47, 2000. ISSN 0453-4514. doi: 10.1016/S0453-4514(00)88750-5. New trends in mathematical programming (Kyoto, 1998).

A. R. Conn, N. I. M. Gould, and P. L. Toint. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM J. Numer. Anal.*, 28(2):545–572, 1991. ISSN 0036-1429. doi: 10.1137/0728030. URL https://doi.org/10.1137/0728030.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for Large-scale Nonlinear Optimization (Release A).* Springer Series in Computational Mathematics. Springer-Verlag, Berlin, 1992.

A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*, volume 1 of *MPS/SIAM Series on Optimization.* SIAM, Philadelphia, USA, 2000. doi: 10.1137/1.9780898719857.

R. Courant. Variational methods for the solution of problems with equilibrium and variation. *Bull. Amer. Math. Soc.*, 49:1–23, 1943.

J. E. Craig. The N-step iteration procedures. *Journal of Mathematics and Physics*, 34(1): 64–73, 1955.

G. Dahlquist, S. C. Eisenstat, and G. H. Golub. Bounds for the error of linear systems of equations using the theory of moments. *J. Math. Anal. Appl.*, 37:151–166, 1972. doi: 10.1016/0022-247X(72)90264-8.

G. Dahlquist, G. H. Golub, and S. G. Nash. Bounds for the error in linear systems. In *Semi-infinite Programming*, volume 15 of *Lecture Notes in Control and Information Science*, pages 154–172. Springer, Berlin-New York, 1979.

T. A. Davis. Algorithm 930: FACTORIZE: An object-oriented linear system solver for matlab. *ACM Trans. Math. Softw.*, 39(4):28:1–28:18, July 2013. doi: 10.1145/2491491.2491498.

T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Software*, 38(1):1:1–1:25, Dec. 2011. doi: 10.1145/2049662.2049663.

R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19(2):400–408, 1982.

J. W. Demmel. *Applied numerical linear algebra.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997. ISBN 0-89871-389-7. doi: 10.1137/1.9781611971446.

G. Di Pillo and L. Grippo. A continuously differentiable exact penalty function for nonlinear programming problems with inequality constraints. *SIAM J. Control Optim.*, 23(1):72–84, 1985. ISSN 0363-0129. doi: 10.1137/0323007.

G. Di Pillo and L. Grippo. An exact penalty function method with global convergence properties for nonlinear programming problems. *Math. Program.*, 36(1):1–18, 1986.

R. Estrin and C. Greif. SPMR: a family of saddle-point minimum residual solvers. *SIAM J. Sci. Comput.*, 40(3):A1884–A1914, 2018. ISSN 1064-8275.

R. Estrin, M. P. Friedlander, D. Orban, and M. A. Saunders. Implementing a smooth exact penalty function for equality-constrained nonlinear optimization. Technical report, 2019a. Working paper.

R. Estrin, M. P. Friedlander, D. Orban, and M. A. Saunders. Implementing a smooth exact penalty function for constrained nonlinear optimization. Technical report, 2019b. Working paper.

R. Estrin, D. Orban, and M. A. Saunders. LSLQ: An iterative method for linear least-squares with an error minimization property. *SIAM J. Matrix Anal. Appl.*, 40(1):254–275, 2019c. ISSN 0895-4798. doi: 10.1137/17M1113552.

R. Estrin, D. Orban, and M. A. Saunders. Euclidean-norm error bounds for SYMMLQ and CG. *SIAM J. Matrix Anal. Appl.*, 40(1):235–253, 2019d. ISSN 0895-4798. doi: 10.1137/16M1094816.

R. Estrin, D. Orban, and M. A. Saunders. LNLQ: An iterative method for least-norm problems with an error minimization property. *SIAM J. Matrix Anal. Appl.*, 2019e. In review.

B. Fischer. *Polynomial based iteration methods for symmetric linear systems*. Wiley-Teubner Series Advances in Numerical Mathematics. John Wiley & Sons, Ltd., Chichester; B. G. Teubner, Stuttgart, 1996. ISBN 0-471-96796-3. doi: 10.1007/978-3-663-11108-5.

R. Fletcher. A class of methods for nonlinear programming with termination and convergence properties. In J. Abadie, editor, *Integer and Nonlinear Programming*, pages 157–175. North-Holland, Amsterdam, 1970.

R. Fletcher. A class of methods for nonlinear programming: III. rates of convergence. In F. A. Lootsma, editor, *Numerical Methods for Nonlinear Optimization*. Academic Press, New York, 1973a.

R. Fletcher. An exact penalty function for nonlinear programming with inequalities. *Math. Programming*, 5:129–150, 1973b. ISSN 0025-5610. doi: 10.1007/BF01580117.

R. Fletcher and S. Lill. A class of methods for nonlinear programming: II. computational experience. In O. L. M. B. Rosen and K. Ritter, editors, *Nonlinear Programming*. Academic Press, 1971.

D. C.-L. Fong. Minimum-residual methods for sparse least-squares using Golub-Kahan bidiagonalization, December 2011.

D. C.-L. Fong and M. A. Saunders. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM J. Sci. Comput.*, 33(5):2950–2971, 2011. doi: 10.1137/10079687X.

R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60(3):315–339, 1991. ISSN 0029-599X. doi: 10.1007/BF01385726.

A. Frommer, K. Kahl, T. Lippert, and H. Rittich. 2-norm error bounds and estimates for Lanczos approximations to linear systems and rational matrix functions. *SIAM J. Matrix Anal. Appl.*, 34(3):1046–1065, 2013. doi: 10.1137/110859749.

A. Gersborg-Hansen, M. P. Bendsøe, and O. Sigmund. Topology optimization of heat conduction problems using the finite volume method. *Struct. Multidiscip. Optim.*, 31(4): 251–259, 2006. ISSN 1615-147X. doi: 10.1007/s00158-005-0584-3.

P. E. Gill, M. A. Saunders, and J. R. Shinnerl. On the stability of Cholesky factorization for symmetric quasidefinite systems. *SIAM J. Matrix Anal. Appl.*, 17(1):35–46, 1996.

G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.*, 2(2):205–224, 1965. doi: 10.1137/0702016.

G. H. Golub and G. Meurant. Matrices, moments and quadrature. In *Numerical analysis 1993 (Dundee, 1993)*, volume 303 of *Pitman Res. Notes Math. Ser.*, pages 105–156. Longman Sci. Tech., Harlow, 1994.

G. H. Golub and G. Meurant. Matrices, moments and quadrature II; how to compute the norm of the error in iterative methods. *BIT*, 37(3):687–705, 1997. doi: 10.1007/BF02510247.

G. H. Golub and G. Meurant. *Matrices, moments and quadrature with applications*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2010. ISBN 978-0-691-14341-5.

G. H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM J. Numer. Anal.*, 10:413–432, 1973. Collection of articles dedicated to the memory of George E. Forsythe.

G. H. Golub and Z. Strakŏs. Estimates in quadratic formulas. *Numer. Algor.*, 8(2-4):241–268, 1994. doi: 10.1007/BF02142693.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 4th edition, 2013. ISBN 978-1-4214-0794-4.

N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. *ACM Trans. Math. Softw.*, 29(4):373–394, Dec. 2003.

M. Hegland. *On the computation of breeding values*, pages 232–242. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990. ISBN 978-3-540-46597-3. doi: 10.1007/3-540-53065-7_103.

M. Hegland. Description and use of animal breeding data for large least squares problems. Technical Report TR/PA/93/50, CERFACS, Toulouse, France, 1993.

M. Heinkenschloss and D. Ridzal. A matrix-free trust-region SQP method for equality constrained optimization. *SIAM J. Optim.*, 24(3):1507–1541, 2014.

M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

M. R. Hestenes. Multiplier and gradient methods. *J. Optim. Theory Appl.*, 4(5):303–320, 1969.

M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409–436, 1952.

W. J. Kammerer and M. Z. Nashed. On the convergence of the conjugate gradient method for singular linear operator equations. *SIAM J. Numer. Anal.*, 9(1):165–181, 1972. doi: 10.1137/0709016.

D. P. Kouri, M. Heinkenschloss, D. Ridzal, and B. G. van Bloemen Waanders. Inexact objective function evaluations in a trust-region algorithm for PDE-constrained optimization under uncertainty. *SIAM J. Sci. Comput.*, 36(6):A3011–A3029, 2014. doi: 10.1137/140955665.

C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards*, 45(4):255–282, 1950.

S. Leyffer. Complementarity constraints as nonlinear equations: theory and numerical experience. In *Optimization with Multivalued Mappings*, volume 2 of *Springer Optim. Appl.*, pages 169–208. Springer, New York, 2006. doi: 10.1007/0-387-34221-4_9.

C.-J. Lin and J. J. Moré. Incomplete cholesky factorizations with limited memory. *SIAM J. Sci. Comput.*, 21(1):24–45, 1999a.

C.-J. Lin and J. J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM J. Optim.*, 9(4):1100–1127, 1999b.

N. Maratos. *Exact Penalty Function Algorithms for Finite Dimensional and Optimization Problems.* PhD thesis, Imperial College of Science and Technology, London, UK, 1978.

Ph. L. Toint. Towards an efficient sparsity exploiting Newton method for minimization. In I. S. Duff, editor, *Sparse Matrices and Their Uses*, pages 57–88, London, 1981. Academic Press.

G. Meurant. The computation of bounds for the norm of the error in the conjugate gradient algorithm. *Numer. Algor.*, 16(1):77–87, 1997. doi: 10.1023/A:1019178811767.

G. Meurant. Estimates of the l_2 norm of the error in the conjugate gradient algorithm. *Numer. Algor.*, 40(2):157–169, 2005. doi: 10.1007/s11075-005-1528-0.

G. Meurant and P. Tichý. A new algorithm for computing quadrature-based bounds in conjugate gradients, 2014. URL http://www.cs.cas.cz/tichy/download/present/2014Spa.pdf.

G. Meurant and P. Tichý. On the numerical behavior of quadrature based bounds for the A-norm of the error in CG. 2015. URL http://www.cs.cas.cz/~tichy/download/present/2015ALA.pdf.

G. Meurant and P. Tichỳ. Approximating the extreme ritz values and upper bounds for the a-norm of the error in cg. *Numerical Algorithms*, pages 1–32, 2018.

J. L. Morales and J. Nocedal. Automatic preconditioning by limited memory quasi-Newton updating. *SIAM J. Optim.*, 10(4):1079–1096, 2000.

H. Mukai and E. Polak. A quadratically convergent primal-dual algorithm with global convergence properties for solving optimization problems with equality constraints. *Math. Programming*, 9(3):336–349, 1975.

J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, second edition, 2006.

D. Orban and M. Arioli. *Iterative Solution of Symmetric Quasi-definite Linear Systems*, volume 3 of *SIAM Spotlights*. SIAM, Philadelphia, PA, 2017.

J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. SIAM, Philadelphia, 2000.

C. C. Paige. Bidiagonalization of matrices and solution of linear equations. *SIAM J. Numer. Anal.*, 11(1):197–209, 1974. doi: 10.1137/0711019.

C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–629, 1975.

C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8(1):43–71, 1982a. doi: 10.1145/355984.355989.

C. C. Paige and M. A. Saunders. Algorithm 583; LSQR: Sparse linear equations and least-squares problems. *ACM Trans. Math. Software*, 8(2):195–209, 1982b. doi: 10.1145/355993.356000.

F. Palacios-Gomez, L. Lasdon, and M. Engquist. Nonlinear optimization by successive linear programming. *Management Science*, 28(10):1106–1120, 1982. ISSN 00251909, 15265501.

G. D. Pillo and L. Grippo. A class of continuously differentiable exact penalty function algorithms for nonlinear programming problems. In E. P. Toft-Christensen, editor, *System Modelling and Optimization*, pages 246–256. Springer-Verlag, Berlin, 1984.

M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, chapter 19. Academic Press, London and New York, 1969.

R. T. Rockafellar. Extended nonlinear programming. In *Nonlinear optimization and related topics*, pages 381–399. Springer, 2000.

Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986. doi: 10.1137/0907058.

M. A. Saunders. Solution of sparse rectangular systems using LSQR and CRAIG. *BIT*, 35: 588–604, 1995. doi: 10.1007/BF01739829.

M. A. Saunders. CME 338 class notes 4: Iterative methods for symmmetric $Ax = b$, 2019. URL http://stanford.edu/class/msande318/notes.html.

V. Simoncini and D. B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25(2):454–477, 2003. ISSN 1064-8275. doi: 10.1137/S1064827502406415.

T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.*, 20(3):626–637, 1983.

G. W. Stewart. The QLP approximation to the singular value decomposition. *SIAM J. Sci. Comput.*, 20(4):1336–1348, 1999. doi: 10.1137/S1064827597319519.

E. Stiefel. Relaxationsmethoden bester Strategie zur Lösung linearer Gleichungssysteme. *Comment. Math. Helv.*, 29:157–179, 1955. ISSN 0010-2571. doi: 10.1007/BF02564277.

Z. Strakoš and P. Tichý. On error estimation in the conjugate gradient method and why it works in finite precision computations. *Elec. Trans. Numer. Anal.*, 13:56–80 (electronic), 2002.

Z. Strakoš and P. Tichý. On error estimation in the conjugate gradient method and why it works in finite precision. *Elec. Trans. Numer. Anal.*, 13, 2002.

D. B. Szyld and O. B. Widlund. Variational analysis of some conjugate gradient methods. *East-West J. Numer. Math.*, 1(1):51–74, 1993.

J. van den Eshof and G. L. G. Sleijpen. Inexact Krylov subspace methods for linear systems. *SIAM J. Matrix Anal. Appl.*, 26(1):125–153, 2004. ISSN 0895-4798. doi: 10.1137/S0895479802403459.

T. van Leeuwen and F. J. Herrmann. A penalty method for PDE-constrained optimization in inverse problems. *Inverse Problems*, 32(1):015007, 26, 2016. ISSN 0266-5611. doi: 10.1088/0266-5611/32/1/015007.

R. J. Vanderbei. Symmetric quasi-definite matrices. *SIAM J. Optim.*, 5(1):100–113, 1995. doi: 10.1137/0805005.

S. J. Wright. Superlinear convergence of a stabilized SQP method to a degenerate solution. *Comput. Optim. Appl.*, 11(3):253–275, 1998.

V. M. Zavala and M. Anitescu. Scalable nonlinear programming via exact differentiable penalty functions and trust-region Newton methods. *SIAM J. Optim.*, 24(1):528–558, 2014.