SANDIA REPORT

SAND2011-9346
Unlimited Release
Printed December, 2011

Numerical study of a matrix-free trust-region SQP method for equality constrained optimization

Denis Ridzal, Miguel Aguiló, Matthias Heinkenschloss

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.
Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2011-9346
Unlimited Release
Printed December, 2011

Numerical study of a matrix-free trust-region SQP method for equality constrained optimization

Denis Ridzal
Optimization and Uncertainty Quantification, MS-1320
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1320
dridzal@sandia.gov

Miguel Aguiló
Optimization and Uncertainty Quantification, MS-1320
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1320
maguilo@sandia.gov

Matthias Heinkenschloss
Department of Computational and Applied Mathematics, MS-134
Rice University
6100 Main Street
Houston, TX 77005-1892
heinken@rice.edu

Abstract

This is a companion publication to the paper "A Matrix-Free Trust-Region SQP Algorithm for Equality Constrained Optimization" [11]. In [11], we develop and analyze a trust-region sequential quadratic programming (SQP) method that supports the matrix-free (iterative, inexact) solution of linear systems. In this report, we document the numerical behavior of the algorithm applied to a variety of equality constrained optimization problems, with constraints given by partial differential equations (PDEs).

Acknowledgments

Denis Ridzal thanks the DOE Office of Science Advanced Scientific Computing Research (ASCR) and the DOE NNSA Advanced Simulation & Computing (ASC) programs for continuous support.

Miguel Aguiló thanks the DOE NNSA Advanced Simulation & Computing (ASC) program for support.

Matthias Heinkenschloss acknowledges funding by NSF grants ACI-0121360 and DMS-0915238.

Contents

1 Introduction . . . 9
2 Background . . . 10
3 The Matrix-Free Trust-Region SQP Algorithm with Inexact Linear System Solves . . . 13
3.1 Computation of the Quasi-Normal Step . . . 14
3.2 Solution of the Tangential Subproblem . . . 14
3.3 Computation of the Tangential Step . . . 16
3.4 Computation of the Lagrange Multipliers . . . 16
3.5 Trust-Region Update . . . 16
4 Example Problems . . . 17
4.1 Distributed Control of the Burgers Equation . . . 17
4.2 Boundary Control of the Navier-Stokes Equations . . . 18
4.3 Boundary Control of the Scalar Ginzburg-Landau Equation . . . 19
4.4 Distributed Control of the Poisson-Boltzmann Equation . . . 19
4.5 Parameter Identification in the Poisson Equation . . . 20
4.6 Parameter Identification in Linear Elastodynamics . . . 21
5 Numerical Results . . . 23
5.1 Linear System Solvers . . . 23
5.2 Algorithm Parameters . . . 24
5.3 Reporting of Results for Individual Test Problems . . . 24
5.4 Summary of Results . . . 27
5.5 Results for Individual Test Problems . . . 28
References . . . 42

Figures

2.1 An illustration of the composite-step approach. Left pane: the quasi-normal step n_k and the tangential step $t_k = \widetilde{t}_k$ computed using *exact* linear system solves. Right pane: the quasi-normal step n_k, the solution $\widetilde{t}_k$ of the tangential subproblem (2.5) and the tangential step t_k computed using *inexact* linear system solves. 11

4.1 Distributed control of the Burgers equation. Left pane: Control solution g^*. Right pane: Computed state solution u^* and desired state $\widehat{u}$. 17

4.2 Geometry of the backward–facing step channel. 18

4.3 Boundary control of the Navier-Stokes equations. Left pane: Plot of the velocity field for the uncontrolled flow, i.e. the solution of the PDEs (4.2b)-(4.2g) for $\mathbf{g} = \mathbf{0}$. Right pane: Plot of the velocity field for the controlled flow, i.e. the velocity solution $\mathbf{u}^*$ of the optimal control problem. 18

4.4 Sketch of a MESFET semiconductor device and the corresponding doping profile (a, b). 19

4.5 Parameter identification in the Poisson equation. Left pane: Target diffusion coefficient; $\kappa = 2$ within a circle of radius 0.3 centered at $(-0.25, -0.25)$, and $\kappa = 1$ everywhere else. Right pane: Computed diffusion coefficient, i.e. the solution κ^* of the inverse problem; for this illustration we used a finer computational grid, 128×128, a finer synthetic grid, 512×512, and $\alpha = 10^{-10}$. 20

4.6 Parameter identification in linear elastodynamics. Top left pane: Target shear modulus. Bottom left pane: Target bulk modulus. Top right pane: Computed shear modulus, i.e. the parameter solution μ^* of the inverse problem. Bottom right pane: Computed bulk modulus, i.e. the parameter solution κ^* of the inverse problem. 22

5.1 Plots of linear solver residuals in I-SQP run 26

5.2 Plots of linear solver residuals; distributed control of the Burgers equation. 29

5.3 Plots of linear solver residuals; boundary control of the Navier-Stokes equations. 31

5.4 Plots of linear solver residuals; boundary control of the scalar Ginzburg-Landau equation; incomplete-LU preconditioning. 33

5.5 Plots of linear solver residuals; boundary control of the scalar Ginzburg-Landau equation; domain-decomposition preconditioning. 35

5.6 Plots of linear solver residuals; distributed control of the Poisson-Boltzmann equation. 37

5.7 Plots of linear solver residuals; parameter identification in the Poisson equation. 39

5.8 Plots of linear solver residuals; parameter identification in linear elastodynamics. 41

Tables

5.1 Performance of the inexact SQP algorithm (I-SQP) for varying linear system solver tolerance tol^{LS}. 25

5.2 Inexact SQP algorithm (I-SQP), Algorithm 1, applied to the distributed control of the Burgers equation. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the incomplete-LU preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4\|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3. 28

5.3 Conventional SQP algorithm (C-SQP) applied to the distributed control of the Burgers equation. Failure is reported if the number of SQP iterations exceeds 100. 28

5.4 Inexact SQP algorithm (I-SQP), Algorithm 1, applied to the boundary control of the Navier-Stokes equations. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the incomplete-LU preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4\|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3. 30

5.5 Conventional SQP algorithm (C-SQP) applied to the boundary control of the Navier-Stokes equations. Failure is reported if the number of SQP iterations exceeds 100. 30

5.6 Inexact SQP algorithm (I-SQP), Algorithm 1, applied to the boundary control of the scalar Ginzburg-Landau equation, with an incomplete-LU preconditioner. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the incomplete-LU preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4\|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3. 32

5.7 Conventional SQP algorithm (C-SQP) applied to the boundary control of the scalar Ginzburg-Landau equation, with an incomplete-LU preconditioner. Failure is reported if the number of SQP iterations exceeds 100. 32

5.8 Inexact SQP algorithm (I-SQP), Algorithm 1, applied to the boundary control of the scalar Ginzburg-Landau equation, with a domain-decomposition preconditioner. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the domain-decomposition preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4\|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3. 34

5.9 Conventional SQP algorithm (C-SQP) applied to the boundary control of the scalar Ginzburg-Landau equation, with a domain-decomposition preconditioner. Failure is reported if the number of SQP iterations exceeds 100. 34

5.10 Inexact SQP algorithm (I-SQP), Algorithm 1, applied to the distributed control of the Poisson-Boltzmann equation. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the incomplete-LU preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4\|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3. 36

5.11 Conventional SQP algorithm (C-SQP) applied to the distributed control of the Poisson-Boltzmann equation. Failure is reported if the number of SQP iterations exceeds 100. 36

5.12 Inexact SQP algorithm (I-SQP), Algorithm 1, applied to parameter identification in the Poisson equation. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the incomplete-LU preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3. 38
5.13 Conventional SQP algorithm (C-SQP) applied to parameter identification in the Poisson equation. Failure is reported if the number of SQP iterations exceeds 1000. 38
5.14 Inexact SQP algorithm (I-SQP), Algorithm 1, applied to parameter identification in linear elastodynamics. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times a factorization of the approximate Schur preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3. 40
5.15 Conventional SQP algorithm (C-SQP) applied to parameter identification in linear elastodynamics. Failure is reported if the number of SQP iterations exceeds 1000. . . 40

1 Introduction

This is a companion publication to the paper "A Matrix-Free Trust-Region SQP Algorithm" [11]. In [11], we develop and analyze a trust-region sequential quadratic programming (SQP) method that supports the matrix-free (iterative, inexact) solution of linear systems. SQP methods compute an approximate solution of the NLP by solving a sequence of quadratic subproblems which are built from a quadratic model of the Lagrangian and a linear Taylor approximation of the constraints. The solution of these subproblems requires the solution of linear systems in which the system matrix involves the constraint Jacobian or its transpose. For large-scale problems, the direct solution of such linear systems is prohibitively expensive. Often, constraint Jacobians are not even formed explicitly, only their action and the action of their transpose on a vector are available. In these cases, iterative linear system solvers have to be applied. Consequently, stopping criteria for the iterative linear system solvers must be developed. The definition of the stopping criteria is a delicate matter, because they directly affect the progress of the optimization algorithm.

This paper summarizes the selection of linear solver tolerances developed in [11] and examines the computational performance of our matrix-free trust-region SQP algorithm. We document the numerical behavior of the algorithm applied to a variety of equality constrained nonlinear programming problems (NLPs), with constraints given by partial differential equations (PDEs).

2 Background

We provide the background necessary to specify the matrix-free trust-region SQP algorithm and to examine the numerical results. Additional background, results and the links to relevant literature are given in [11]. Our SQP algorithm solves equality constrained NLPs of the form

$$\min \quad f(x) \tag{2.1a}$$
$$\text{s.t.} \quad c(x) = 0, \tag{2.1b}$$

where $f : \mathcal{X} \to \mathbb{R}$ and $c : \mathcal{X} \to \mathcal{C}$ for some Hilbert spaces $\mathcal{X}$ and $\mathcal{C}$. We define the Lagrangian functional $\mathcal{L} : \mathcal{X} \times \mathcal{C} \to \mathbb{R}$ for (2.1) as

$$\mathcal{L}(x, \lambda) = f(x) + \langle \lambda, c(x) \rangle_{\mathcal{C}} .$$

We let x_k be the k-th SQP iterate, λ_k the Lagrange multiplier estimate at x_k, and $H_k = H(x_k, \lambda_k)$ the Hessian $\nabla_{xx}\mathcal{L}(x_k, \lambda_k)$ of the Lagrangian or a self-adjoint approximation thereof.

Trust-region SQP methods compute an approximate solution of (2.1) by approximately solving a sequence of subproblems derived from

$$\min \quad \tfrac{1}{2} \langle H_k s, s \rangle_{\mathcal{X}} + \langle \nabla_x \mathcal{L}(x_k, \lambda_k), s \rangle_{\mathcal{X}} + \mathcal{L}(x_k, \lambda_k) \tag{2.2a}$$
$$\text{s.t.} \quad c_x(x_k)s + c(x_k) = 0 \tag{2.2b}$$
$$\|s\|_{\mathcal{X}} \leq \Delta_k. \tag{2.2c}$$

To deal with the possible incompatibility of the constraints (2.2b), (2.2c) we apply a Byrd-Omojokun-like composite-step approach.

The trial step s_k is computed as the sum of a quasi-normal step n_k and a tangential step t_k. The role of the quasi-normal step n_k is to reduce linear infeasibility. It is computed as an approximate solution of

$$\min \quad \|c_x(x_k)n + c(x_k)\|_{\mathcal{C}}^2 \tag{2.3a}$$
$$\text{s.t.} \quad \|n\|_{\mathcal{X}} \leq \zeta \Delta_k, \tag{2.3b}$$

where $\zeta \in (0, 1)$ is a fixed constant.

Once the quasi-normal step n_k is computed, the tangential step t_k is computed as an approximate solution of the subproblem

$$\min \quad \tfrac{1}{2} \langle H_k(t + n_k), t + n_k \rangle_{\mathcal{X}} + \langle \nabla_x \mathcal{L}(x_k, \lambda_k), t + n_k \rangle_{\mathcal{X}} + \mathcal{L}(x_k, \lambda_k) \tag{2.4a}$$
$$\text{s.t.} \quad c_x(x_k)t = 0 \tag{2.4b}$$
$$\|t + n_k\|_{\mathcal{X}} \leq \Delta_k. \tag{2.4c}$$

This task is handled in two stages. In the first stage, we approximately solve the related problem

$$\min \quad \tfrac{1}{2} \left\langle H_k \widetilde{t}, \widetilde{t} \right\rangle_{\mathcal{X}} + \left\langle \widetilde{W}_k g_k, \widetilde{t} \right\rangle_{\mathcal{X}} \tag{2.5a}$$
$$\text{s.t.} \quad \widetilde{t} \in \text{Range}(\widetilde{W}_k) \tag{2.5b}$$
$$\|n_k + \widetilde{t}\|_{\mathcal{X}} \leq \Delta_k, \tag{2.5c}$$

where $g_k \stackrel{\text{def}}{=} \nabla_x \mathcal{L}(x_k, \lambda_k) + H_k n_k$. The operator $\widetilde{W}_k : \mathcal{X} \to \mathcal{X}$ is an inexact projection onto the null space $\text{Null}(c_x(x_k))$ of the linearized constraints. It is defined implicitly via $z = \widetilde{W}_k w$, where z is computed by iteratively solving the *augmented system*

$$\begin{pmatrix} I & c_x(x_k)^* \\ c_x(x_k) & 0 \end{pmatrix} \begin{pmatrix} z \\ y \end{pmatrix} = \begin{pmatrix} w \\ 0 \end{pmatrix}. \tag{2.6}$$

In the second stage, we compute the tangential step t_k by applying an approximate null-space projection to $\widetilde{t}_k$, the solution of (2.5). This additional projection, computed by solving another augmented system, restores as much of the linear feasibility lost during the first stage as is needed to ensure the global convergence of the SQP algorithm.

Figure 2.1 gives an illustration of the composite-step approach for two scenarios: exact linear system solves (left pane) and inexact linear system solves (right pane).

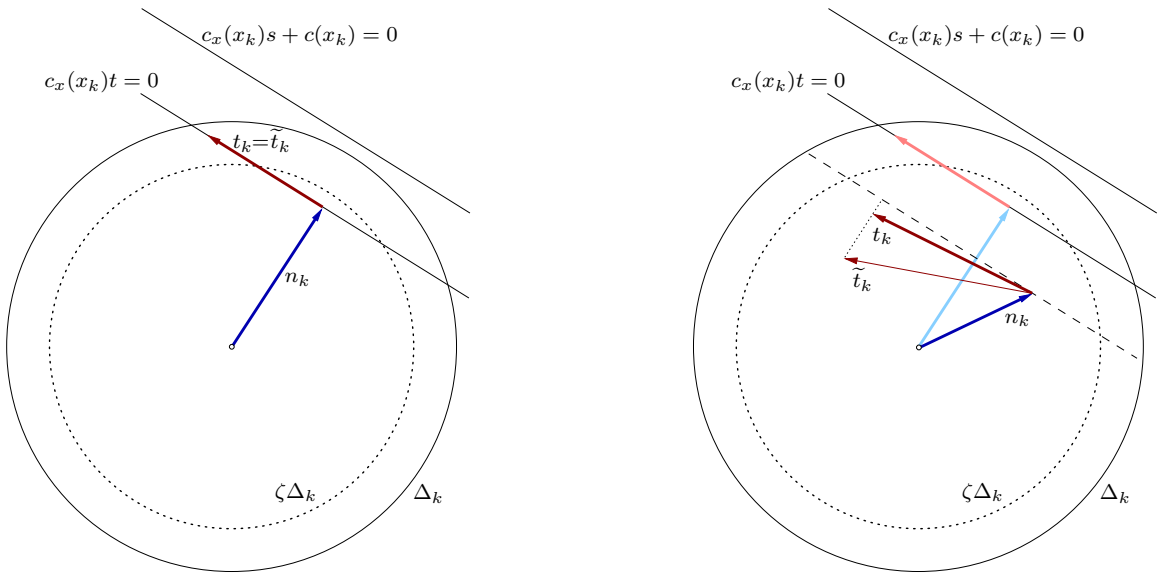

Figure 2.1: An illustration of the composite-step approach. Left pane: the quasi-normal step n_k and the tangential step $t_k = \widetilde{t}_k$ computed using *exact* linear system solves. Right pane: the quasi-normal step n_k, the solution $\widetilde{t}_k$ of the tangential subproblem (2.5) and the tangential step t_k computed using *inexact* linear system solves.

In a conventional trust-region SQP algorithm, once the trial step $s_k = n_k + t_k$ is computed, one must decide whether to accept the step and how to update the trust-region radius Δ_k. In order to perform these tasks, one may use the *augmented Lagrangian merit function*

$$\phi(x, \lambda; \rho) = f(x) + \langle \lambda, c(x) \rangle_{\mathcal{C}} + \rho \|c(x)\|_{\mathcal{C}}^2 = \mathcal{L}(x, \lambda) + \rho \|c(x)\|_{\mathcal{C}}^2 \,. \tag{2.7}$$

The step s_k is accepted or rejected and the trust-region radius Δ_k is updated based on the ratio between the actual reduction

$$ared(s_k; \rho_k) = \phi(x_k, \lambda_k; \rho_k) - \phi(x_k + s_k, \lambda_{k+1}; \rho_k) \tag{2.8}$$

and the predicted reduction

$$\begin{aligned} \widehat{pred}(s_k; \rho_k) = \phi(x_k, \lambda_k; \rho_k) - \Big[& \mathcal{L}(x_k, \lambda_k) + \langle \nabla_x \mathcal{L}(x_k, \lambda_k), s_k \rangle_{\mathcal{X}} + \tfrac{1}{2} \langle H_k s_k, s_k \rangle_{\mathcal{X}} \\ & + \langle \lambda_{k+1} - \lambda_k, c_x(x_k) s_k + c(x_k) \rangle_{\mathcal{C}} + \rho_k \|c_x(x_k) s_k + c(x_k)\|_{\mathcal{C}}^2 \Big]. \end{aligned} \tag{2.9}$$

Here λ_{k+1} is a Lagrange multiplier estimate corresponding to the trial iterate $x_k + s_k$. It can be computed as an approximate solution of $\min \|\nabla f(x_k + s_k) + c_x(x_k + s_k)^* \lambda\|_{\mathcal{X}^*}$.

Defining

$$r_k^t = c_x(x_k) t_k \,,$$

which can be interpreted as a gauge of the loss of linear feasibility, the predicted reduction (2.9) can be split into two parts,

$$\widehat{pred}(s_k;\rho_k) \equiv pred(n_k,\widetilde{t}_k;\rho_k) + rpred(r_k^t;\rho_k),$$

with the following definitions:

$$\begin{aligned} & pred(n_k,\widetilde{t}_k;\rho_k) \\ = \; & -\left\langle \widetilde{W}_k g_k, \widetilde{t}_k \right\rangle_{\mathcal{X}} - \tfrac{1}{2}\left\langle H_k\widetilde{t}_k, \widetilde{t}_k\right\rangle_{\mathcal{X}} - \langle \nabla_x \mathcal{L}(x_k,\lambda_k), n_k\rangle_{\mathcal{X}} - \tfrac{1}{2}\langle H_k n_k, n_k\rangle_{\mathcal{X}} \\ & - \langle \lambda_{k+1}-\lambda_k, c_x(x_k)n_k + c(x_k)\rangle_{\mathcal{C}} \\ & +\rho_k\left(\|c(x_k)\|_{\mathcal{C}}^2 - \|c_x(x_k)n_k + c(x_k)\|_{\mathcal{C}}^2\right) \end{aligned} \tag{2.10}$$

and

$$rpred(r_k^t;\rho_k) = -\left\langle \lambda_{k+1}-\lambda_k, r_k^t\right\rangle_{\mathcal{C}} - \rho_k\|r_k^t\|_{\mathcal{C}}^2 - 2\rho_k\left\langle r_k^t, c_x(x_k)n_k + c(x_k)\right\rangle_{\mathcal{C}}. \tag{2.11}$$

In our algorithm, we first compute a penalty parameter ρ_k satisfying

$$pred(n_k,\widetilde{t}_k;\rho_k) \geq \frac{\rho_k}{2}\left(\|c(x_k)\|_{\mathcal{C}}^2 - \|c_x(x_k)n_k + c(x_k)\|_{\mathcal{C}}^2\right),$$

and then apply an approximate projection to $\widetilde{t}_k$ to compute a tangential step t_k that satisfies the requirement

$$|rpred(r_k^t;\rho_k)| \leq \eta_0\, pred(n_k,\widetilde{t}_k;\rho_k). \tag{2.12}$$

Here $\eta_0 \in (0, 1-\eta_1)$, and $\eta_1 \in (0,1)$ is the smallest acceptable ratio of the actual reduction defined in (2.8) and the predicted reduction $pred(n_k,\widetilde{t}_k;\rho_k)$ defined in (2.10).

As we have mentioned, the tangential step is computed in two stages. The approximate solution $\widetilde{t}_k$ of (2.5) can be computed immediately after the normal step has been computed. The final tangential step t_k, however, depends on $pred(n_k,\widetilde{t}_k;\rho_k)$, which in turn depends on the new Lagrange multiplier estimate. Furthermore, to guarantee convergence of our algorithm, all computed substeps must satisfy certain quality criteria. These are discussed in [11].

In the next section we state the complete algorithm and specify how the subproblems are being solved.

3 The Matrix-Free Trust-Region SQP Algorithm with Inexact Linear System Solves

We begin with the specification of the master algorithm. We make forward references to several subalgorithms, given in Sections 3.1–3.5.

Algorithm 1 [Trust-region SQP algorithm with inexact linear system solves]

1. Initialization: Choose initial point x_0, initial trust-region radius Δ_0, constants $0 < \alpha_1, \eta_1 < 1$, $0 < \eta_0 < 1 - \eta_1$, $\rho_{-1} \geq 1$, $\bar{\rho} > 0$, and $tol^{SQP} > 0$. Set $\Delta_{min}, \Delta_{max}$ so that $0 < \Delta_{min} < \Delta_{max}$. Set forcing parameters $\xi^{qn}, \xi^{pg}, \xi^{proj}, \xi^{tang}, \xi^{lmh} \in (0,1)$, $\xi^{lmg} > 0$ and $\xi_4 > 1$. Choose initial Lagrange multiplier λ_{-1} and compute λ_0 by solving (3.12) with linear solver tolerance (3.13).

2. **For** $k = 0, 1, 2, \ldots$

   (a) Convergence Check: If $\|\nabla_x \mathcal{L}(x_k, \lambda_k)\|_{\mathcal{X}} < tol^{SQP}$ and $\|c(x_k)\|_{\mathcal{C}} < tol^{SQP}$, then **stop**.

   (b) Compute quasi-normal step n_k using Algorithm 2 and linear solver tolerance (3.3).
   Compute $\widetilde{t}_k$, $\widetilde{t}_k^{cp}$ using Algorithm 3 and linear solver tolerances (3.6), (3.8).

   (c) **For** $i = 0, 1, 2, \ldots$

      i. **For** $j = 0, 1, 2, \ldots$

         A. Compute tangential step t_k by solving (3.9) with linear solver tolerance (3.10).

         B. Compute Lagrange multiplier estimate λ_{k+1} at $x_k + n_k + t_k$ by solving (3.12) with linear solver tolerance (3.13).

         C. Update the penalty parameter: If

$$pred(n_k, \widetilde{t}_k; \rho_{k-1}) \geq \frac{\rho_{k-1}}{2} \left(\|c(x_k)\|_{\mathcal{C}}^2 - \|c_x(x_k) n_k + c(x_k)\|_{\mathcal{C}}^2 \right)$$

         then set $\rho_k \leftarrow \rho_{k-1}$. Otherwise set

$$\rho_k \leftarrow \frac{-2\, pred(n_k, \widetilde{t}_k; \rho_{k-1})}{\|c(x_k)\|_{\mathcal{C}}^2 - \|c_x(x_k) n_k + c(x_k)\|_{\mathcal{C}}^2} + 2\rho_{k-1} + \bar{\rho}.$$

         D. If $|rpred(c_x(x_k) t_k; \rho_k)| > \eta_0\, pred(n_k, \widetilde{t}_k; \rho_k)$, set $\xi^{tang} \leftarrow 10^{-3}\, \xi^{tang}$, else **break**.

      End For

      Reset ξ^{tang} to its value at outer iteration i prior to Step 2(c)i.

      ii. If $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ and $\widetilde{t}_k = \widetilde{t}_k^{cp}$

         Set $\xi^{qn} \leftarrow 10^{-1}\, \xi^{qn}$, $\xi^{pg} \leftarrow 10^{-1} \xi^{pg}$, $\xi^{proj} \leftarrow 10^{-1} \xi^{proj}$, $\xi^{tang} \leftarrow 10^{-1}\, \xi^{tang}$.
         Recompute n_k using Algorithm 2 and linear solver tolerance (3.3).
         Recompute $\widetilde{t}_k$, $\widetilde{t}_k^{cp}$ using Algorithm 3 and linear solver tolerances (3.6), (3.8).

      Else If $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ and $\widetilde{t}_k \neq \widetilde{t}_k^{cp}$

         Set $\widetilde{t}_k \leftarrow \widetilde{t}_k^{cp}$.

      Else

         Optional: Reset ξ^{qn}, ξ^{pg}, ξ^{proj} and ξ^{tang} to their values from Step 1.
         break

   End For

   (d) Acceptance Test:

      i. Compute trial step $s_k \leftarrow n_k + t_k$.

      ii. Compute ratio $\theta_k \leftarrow ared(s_k; \rho_k) / pred(n_k, \widetilde{t}_k; \rho_k)$.

      iii. If $\theta_k \geq \eta_1$, set $x_{k+1} \leftarrow x_k + s_k$, and choose Δ_{k+1}, via Algorithm 4, such that

$$\max\{\Delta_{min}, \Delta_k\} \leq \Delta_{k+1} \leq \Delta_{max}.$$

      Otherwise set $x_{k+1} \leftarrow x_k$, $\lambda_{k+1} \leftarrow \lambda_k$, and $\Delta_{k+1} \leftarrow \alpha_1 \|s_k\|_{\mathcal{X}}$.

End For

3.1 Computation of the Quasi-Normal Step

We use a dogleg method to compute an approximate solution of (2.3). Let n_k^{cp} be the Cauchy point, i.e. the solution of $\min\left\{\|c_x(x_k)n + c(x_k)\|_{\mathcal{C}}^2 \,:\, n = -\alpha c_x(x_k)^* c(x_k), \alpha \geq 0\right\}$. It is easy to verify that

$$n_k^{cp} = -\frac{\|c_x(x_k)^* c(x_k)\|_{\mathcal{X}}^2}{\|c_x(x_k)c_x(x_k)^* c(x_k)\|_{\mathcal{C}}^2} c_x(x_k)^* c(x_k). \tag{3.1}$$

If $\|n_k^{cp}\|_{\mathcal{X}} \geq \zeta\Delta_k$, then we set the quasi-normal step to $n_k = \zeta\Delta_k n_k^{cp}/\|n_k^{cp}\|_{\mathcal{X}}$.

If $\|n_k^{cp}\|_{\mathcal{X}} < \zeta\Delta_k$, we compute an approximate minimum-norm solution n_k^N of $\min \|c_x(x_k)n + c(x_k)\|_{\mathcal{C}}^2$ and compute the quasi-normal step by moving from n_k^{cp} as far as possible toward n_k^N while staying within the trust region with radius $\zeta\Delta_k$. The vector n_k^N can be computed by solving the augmented system

$$\begin{pmatrix} I & c_x(x_k)^* \\ c_x(x_k) & 0 \end{pmatrix} \begin{pmatrix} n_k^N \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ -c(x_k) \end{pmatrix}.$$

If iterative solvers are used, we instead solve for $\delta n_k = n_k^N - n_k^{cp}$:

$$\begin{pmatrix} I & c_x(x_k)^* \\ c_x(x_k) & 0 \end{pmatrix} \begin{pmatrix} \delta n_k \\ y \end{pmatrix} = \begin{pmatrix} -n_k^{cp} + e^1 \\ -c_x(x_k)n_k^{cp} - c(x_k) + e^2 \end{pmatrix}. \tag{3.2}$$

The size of the residual $\begin{pmatrix} e^1 & e^2 \end{pmatrix} \in \mathcal{X} \times \mathcal{C}$ is restricted via

$$\|e^1\|_{\mathcal{X}}^2 + \|e^2\|_{\mathcal{C}}^2 \leq (\xi^{qn})^2 \, \|c_x(x_k)n_k^{cp} + c(x_k)\|_{\mathcal{C}}^2 \,, \tag{3.3}$$

where $0 < \xi^{qn} \leq 1$. Our algorithm for computing the quasi-normal step is given as follows.

Algorithm 2 [Dogleg method for the quasi-normal subproblem]

1. Compute n_k^{cp} as defined in (3.1).
2. If $\|n_k^{cp}\|_{\mathcal{X}} \geq \zeta\Delta_k$, then set $n_k \leftarrow \zeta\Delta_k n_k^{cp}/\|n_k^{cp}\|_{\mathcal{X}}$.
3. Else compute δn_k via (3.2) such that e^1 and e^2 satisfy (3.3).
   If $\|n_k^{cp} + \delta n_k\|_{\mathcal{X}} \leq \zeta\Delta_k$, then set $n_k \leftarrow n_k^N$.
   Else compute $\theta_k \in (0,1)$ such that $\|n_k^{cp} + \theta_k \delta n_k\|_{\mathcal{X}} = \zeta\Delta_k$, and set $n_k \leftarrow n_k^{cp} + \theta_k \delta n_k$.

3.2 Solution of the Tangential Subproblem

The tangential subproblem (2.5) is solved using a modified truncated Steihaug-Toint conjugate gradient (STCG) method. Aside from handling the nonstandard objective function in (2.5), our modifications involve a full orthogonalization of search directions (i.e. the conventional three-term recurrence relation enjoyed by CG is lost), several subtle yet crucial tunings of the STCG truncation criteria and the related exit computations, and a special termination condition related to an estimate of the null-space error accumulated by our STCG variant. The latter is discussed next.

Our algorithm for the solution of the tangential subproblem (2.5), Algorithm 3, repeatedly applies an inexact null-space projector $\widetilde{W}_k$ by iteratively solving augmented systems of type (2.6). We note that $\widetilde{W}_k$ is not explicitly available; only the results of its action on the vector g_k and its action on the STCG residuals $\widetilde{r}_i$ used in Algorithm 3 are known.[1] We introduce the operator $R_i : \mathbb{R}^{i+1} \to \mathcal{X}$, given by

$$R_i = [g_k, \widetilde{r}_1, ..., \widetilde{r}_i] \,,$$

[1] Only the scope of the index k extends from Algorithm 1 to Algorithm 3 – indices i and j are independent, i.e. their scope is local to each algorithm.

the operator $\widetilde{Y}_i \ : \ \mathbb{R}^{i+1} \to \mathcal{X}$, given by

$$\widetilde{Y}_i = [\widetilde{W}_k g_k, \widetilde{W}_k \widetilde{r}_1, ..., \widetilde{W}_k \widetilde{r}_i] \, ,$$

and the diagonal matrix

$$D_i = \mathrm{diag}(\|\widetilde{W}_k g_k\|_{\mathcal{X}}, \|\widetilde{W}_k \widetilde{r}_1\|_{\mathcal{X}}, \ldots, \|\widetilde{W}_k \widetilde{r}_i\|_{\mathcal{X}}).$$

Finally, we define the matrix

$$\widehat{S}_i = D_i^{-1}(\widetilde{Y}_i^T R_i - D_i^2) D_i^{-1} \, . \tag{3.4}$$

In [11], it is shown that $\|\widehat{S}_i\|_2$ can be used to control the cumulative effect of inexactness in the projections $\widetilde{W}_k$. Our modified STCG algorithm is specified next.

Algorithm 3 [STCG method with inexact null-space projections]

0. Given relative tolerance $tol^{CG} \in (0,1)$. Given iteration maximum $i^{CG}_{\max}$. Let $\widetilde{t}_{k,0} = 0 \in \mathcal{X}$. Compute $\widetilde{r}_0 \leftarrow \widetilde{W}_k g_k$ by solving (3.5) with the linear solver tolerance (3.6). If $\|\widetilde{r}_0\|_{\mathcal{X}} = 0$, **stop**.
1. **For** $i = 0, 1, 2, ..., i^{CG}_{\max}$
   (a) If $i = 0$ set $\widetilde{z}_0 \leftarrow \widetilde{r}_0$, else compute $\widetilde{z}_i \leftarrow \widetilde{W}_k \widetilde{r}_i$ via (3.7) with the linear solver tolerance (3.8). If $\|\widetilde{z}_i\|_{\mathcal{X}} \le tol^{CG} \|\widetilde{r}_0\|_{\mathcal{X}}$ and $i > 0$, return $\widetilde{t}_k \leftarrow \widetilde{t}_{k,i}$ and $\widetilde{t}^{cp}_k \leftarrow \widetilde{t}_{k,1}$, and **stop**.
   (b) Compute $\widehat{S}_i$ defined in (3.4). If $\|\widehat{S}_i\|_2 > 1/2$, return $\widetilde{t}_k \leftarrow \widetilde{t}_{k,i}$ and $\widetilde{t}^{cp}_k \leftarrow \widetilde{t}_{k,1}$, and **stop**.
   (c) Set $\widetilde{p}_i \leftarrow -\widetilde{z}_i + \sum_{j=0}^{i-1} \frac{\langle \widetilde{z}_i, H_k \widetilde{p}_j \rangle_{\mathcal{X}}}{\langle \widetilde{p}_j, H_k \widetilde{p}_j \rangle_{\mathcal{X}}} \widetilde{p}_j$.
   (d) If $\langle \widetilde{r}_i, \widetilde{p}_i \rangle_{\mathcal{X}} \neq 0$ and $\langle \widetilde{p}_i, H \widetilde{p}_i \rangle_{\mathcal{X}} \le 0$, compute θ such that $\mathrm{sign}(\theta) = \mathrm{sign}(-\langle \widetilde{r}_i, \widetilde{p}_i \rangle_{\mathcal{X}})$ and $\|n_k + \widetilde{t}_{k,i} + \theta \widetilde{p}_i\|_{\mathcal{X}} = \Delta_k$, and return $\widetilde{t}_k \leftarrow \widetilde{t}_{k,i+1} \leftarrow \widetilde{t}_{k,i} + \theta \widetilde{p}_i$ and $\widetilde{t}^{cp}_k \leftarrow \widetilde{t}_{k,1}$, and **stop**.
   If $\langle \widetilde{r}_i, \widetilde{p}_i \rangle_{\mathcal{X}} = 0$ and $\langle \widetilde{p}_i, H \widetilde{p}_i \rangle_{\mathcal{X}} < 0$, compute θ such that $\|n_k + \widetilde{t}_{k,i} + \theta \widetilde{p}_i\| = \Delta_k$, and return $\widetilde{t}_k \leftarrow \widetilde{t}_{k,i+1} \leftarrow \widetilde{t}_{k,i} + \theta \widetilde{p}_i$ and $\widetilde{t}^{cp}_k \leftarrow \widetilde{t}_{k,1}$, and **stop**.
   (e) If $\langle \widetilde{r}_i, \widetilde{p}_i \rangle_{\mathcal{X}} = 0$, return $\widetilde{t}_k \leftarrow \widetilde{t}_{k,i}$ and $\widetilde{t}^{cp}_k \leftarrow \widetilde{t}_{k,1}$, and **stop**.
   (f) Set $\widetilde{\alpha}_i \leftarrow -\frac{\langle \widetilde{r}_i, \widetilde{p}_i \rangle_{\mathcal{X}}}{\langle \widetilde{p}_i, H_k \widetilde{p}_i \rangle_{\mathcal{X}}}$.
   (g) Set $\widetilde{t}_{k,i+1} \leftarrow \widetilde{t}_{k,i} + \widetilde{\alpha}_i \widetilde{p}_i$.
   (h) If $\|n_k + \widetilde{t}_{k,i+1}\|_{\mathcal{X}} \ge \Delta_k$, compute θ such that $\mathrm{sign}(\theta) = \mathrm{sign}(\widetilde{\alpha}_i)$ and $\|n_k + \widetilde{t}_{k,i} + \theta \widetilde{p}_i\|_{\mathcal{X}} = \Delta_k$, and return $\widetilde{t}_k \leftarrow \widetilde{t}_{k,i+1} \leftarrow \widetilde{t}_{k,i} + \theta \widetilde{p}_i$ and $\widetilde{t}^{cp}_k \leftarrow \widetilde{t}_{k,1}$, and **stop**.
   (i) Set $\widetilde{r}_{i+1} \leftarrow \widetilde{r}_i + \widetilde{\alpha}_i H_k \widetilde{p}_i$.

   End For

The augmented system residuals related to the application of the inexact projector $\widetilde{W}_k$ are controlled as follows. In Step 0 of Algorithm 3 the inexact projected gradient $\widetilde{r}_0 = \widetilde{W}_k g_k$ is computed. The iterative linear system solver returns $\widetilde{r}_0$ satisfying

$$\begin{pmatrix} I & c_x(x_k)^* \\ c_x(x_k) & 0 \end{pmatrix} \begin{pmatrix} \widetilde{r}_0 \\ y \end{pmatrix} = \begin{pmatrix} g_k \\ 0 \end{pmatrix} + \begin{pmatrix} e^1 \\ e^2 \end{pmatrix} \, . \tag{3.5}$$

The residual $\begin{pmatrix} e^1 & e^2 \end{pmatrix} \in \mathcal{X} \times \mathcal{C}$ must satisfy

$$\|e^1\|_{\mathcal{X}} + \|e^2\|_{\mathcal{C}} \le \xi^{pg} \min \{ \|\widetilde{r}_0\|_{\mathcal{X}}, \Delta_k, \|g_k\|_{\mathcal{X}} \} \, , \tag{3.6}$$

where $0 < \xi^{pg} \le 1$. In Step 1a in Algorithm 3 we compute $\widetilde{z}_i = \mathcal{W}_k(\widetilde{r}_i) = \widetilde{W}_k \widetilde{r}_i$. The iterative linear system solver returns $\widetilde{z}_i$ satisfying

$$\begin{pmatrix} I & c_x(x_k)^* \\ c_x(x_k) & 0 \end{pmatrix} \begin{pmatrix} \widetilde{z}_i \\ y \end{pmatrix} = \begin{pmatrix} \widetilde{r}_i \\ 0 \end{pmatrix} + \begin{pmatrix} e^1_i \\ e^2_i \end{pmatrix} \, , \tag{3.7}$$

where the residual $\begin{pmatrix} e_i^1 & e_i^2 \end{pmatrix} \in \mathcal{X} \times \mathcal{C}$ is controlled by the condition

$$\|e_i^1\|_{\mathcal{X}} + \|e_i^2\|_{\mathcal{C}} \leq \xi^{proj} \min\left\{\|\widetilde{z}_i\|_{\mathcal{X}}, \|\widetilde{r}_i\|_{\mathcal{X}}\right\}, \tag{3.8}$$

with $0 < \xi^{proj} \leq 1$.

3.3 Computation of the Tangential Step

Once the approximate solution $\widetilde{t}_k$ of the tangential subproblem (2.5) has been obtained, the tangential step t_k is computed. The goal is to restore some of the linear feasibility lost in Algorithm 3. To this end, we perform another inexact null space projection,

$$\begin{pmatrix} I & c_x(x_k)^* \\ c_x(x_k) & 0 \end{pmatrix} \begin{pmatrix} t_k \\ y \end{pmatrix} = \begin{pmatrix} \widetilde{t}_k \\ 0 \end{pmatrix} + \begin{pmatrix} e^1 \\ e^2 \end{pmatrix}, \tag{3.9}$$

where the residual $\begin{pmatrix} e^1 & e^2 \end{pmatrix} \in \mathcal{X} \times \mathcal{C}$ must satisfy

$$\|e^1\|_{\mathcal{X}} + \|e^2\|_{\mathcal{C}} \leq \Delta_k \min\left\{\Delta_k, \|n_k + t_k\|_{\mathcal{X}}, \xi^{tang}\|\widetilde{t}_k\|_{\mathcal{X}}/\Delta_k\right\}, \tag{3.10}$$

for $0 < \xi^{tang} \leq 1$.

3.4 Computation of the Lagrange Multipliers

Let $\widehat{x}_k = x_k + n_k + t_k$. The Lagrange multiplier estimate λ_{k+1} is computed by approximately minimizing $\|\nabla f(\widehat{x}_k) + c_x(\widehat{x}_k)^*\lambda\|_{\mathcal{X}^*}$, which can be accomplished by solving the augmented system

$$\begin{pmatrix} I & c_x(\widehat{x}_k)^* \\ c_x(\widehat{x}_k) & 0 \end{pmatrix} \begin{pmatrix} z \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla_x f(\widehat{x}_k) \\ 0 \end{pmatrix}. \tag{3.11}$$

If iterative solvers are used, we instead solve for $\Delta\lambda = \lambda_{k+1} - \lambda_k$, where λ_k is the previous Lagrange multiplier estimate, as follows:

$$\begin{pmatrix} I & c_x(\widehat{x}_k)^* \\ c_x(\widehat{x}_k) & 0 \end{pmatrix} \begin{pmatrix} z \\ \Delta\lambda \end{pmatrix} = \begin{pmatrix} -\nabla_x f(\widehat{x}_k) - c_x(\widehat{x}_k)^*\lambda_k + e^1 \\ e^2 \end{pmatrix}. \tag{3.12}$$

The residual $\begin{pmatrix} e^1 & e^2 \end{pmatrix} \in \mathcal{X} \times \mathcal{C}$ must satisfy

$$\|e^1\|_{\mathcal{X}} + \|e^2\|_{\mathcal{C}} \leq \min\left\{\xi^{lmg}, \xi^{lmh}\|\nabla_x f(\widehat{x}_k) + c_x(\widehat{x}_k)^*\lambda_k\|_{\mathcal{X}}\right\}, \tag{3.13}$$

for $0 < \xi^{lmh} \leq 1$ and a fixed $\xi^{lmg} > 0$ independent of k.

3.5 Trust-Region Update

Whenever the trial step s_k is accepted, the radius Δ_{k+1} of the trust region is computed as follows.

Algorithm 4 [Trust-region update for accepted trial steps]

0. Given actual to predicted reduction ratio θ_k, current trust-region radius Δ_k, trial step s_k.
1. If $\theta_k \geq 0.9$, set
   $\Delta_{k+1} \leftarrow \min\left\{\max\left\{7\,\|s_k\|_{\mathcal{X}}, \Delta_k, \Delta_{min}\right\}, \Delta_{max}\right\}$
   Else If $\theta_k \geq 0.8$, set
   $\Delta_{k+1} \leftarrow \min\left\{\max\left\{2\,\|s_k\|_{\mathcal{X}}, \Delta_k, \Delta_{min}\right\}, \Delta_{max}\right\}$
   Else set
   $\Delta_{k+1} \leftarrow \max\left\{\Delta_k, \Delta_{min}\right\}$.

4 Example Problems

We study the performance of Algorithm 1 on six problems in PDE-constrained optimization. This study updates some of our previously published results, see [14, 10, 9]. Four problems come from optimal design and control, while two involve parameter identification. We label them as follows:

- distributed control of the Burgers equation,
- boundary control of the Navier-Stokes equations,
- boundary control of the scalar Ginzburg-Landau equation,
- distributed control of the Poisson-Boltzmann equation,
- parameter identification in the Poisson equation, and
- parameter identification in linear elastodynamics.

The problems are defined in this section. We limit the exposition to their formal statements using the strong form of the governing PDEs. For the implementation and solution, we follow the discretize-then-optimize approach. All problems are implemented in Matlab®. A few select solution plots are included in this section, for illustrative purposes only. The numerical study of the performance of the matrix-free trust-region SQP algorithm, our primary objective, is presented in Section 5.

4.1 Distributed Control of the Burgers Equation

We consider an optimal control problem involving the steady-state Burgers equation in one dimension. The infinite-dimensional problem formulation as well as the problem discretization are identical to those in [14]. Let $\Omega = [0, 1]$, $\partial\Omega = \{0, 1\}$. The problem is given by

$$\underset{u,g}{\text{minimize}} \quad \frac{1}{2}\int_\Omega (u - \widehat{u})^2\, dx + \frac{\alpha}{2}\int_\Omega g^2\, dx \tag{4.1a}$$

subject to

$$-\nu u_{xx} + u u_x = g \qquad \text{in} \quad \Omega \tag{4.1b}$$

$$u = 0 \qquad \text{on} \quad \partial\Omega\,. \tag{4.1c}$$

Here u and g are the *state* and *control* variables, respectively, $\alpha = 10^{-5}$ is a control penalty parameter, $\nu = 10^{-2}$ is the viscosity parameter, and $\widehat{u} = \widehat{u}(x) = \sin(2\pi x)$ is the desired state. The domain Ω is divided into 100 intervals of equal length. We discretize the Burgers equation using linear finite elements. The solution of the optimal control problem is shown in Figure 4.1.

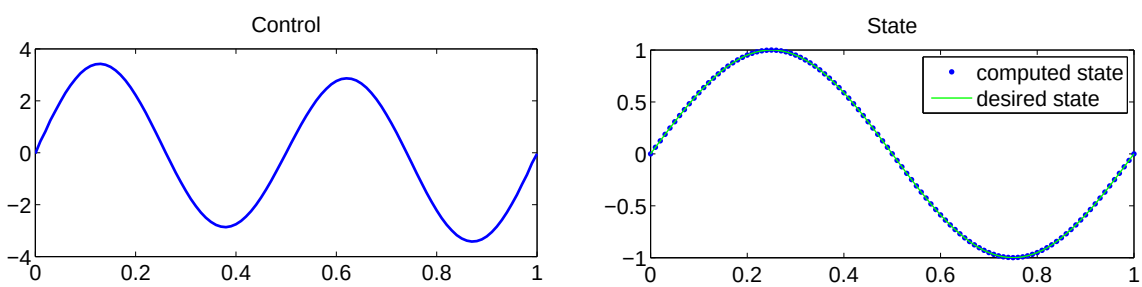

Figure 4.1: Distributed control of the Burgers equation. Left pane: Control solution g^*. Right pane: Computed state solution u^* and desired state $\widehat{u}$.

4.2 Boundary Control of the Navier-Stokes Equations

Let $\Omega \in \mathbb{R}^2$ be the channel with a backward facing step shown in Figure 4.2. We consider an optimal

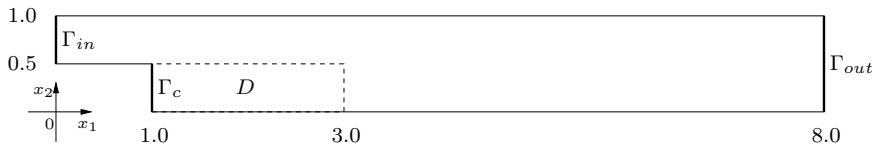

Figure 4.2: Geometry of the backward–facing step channel.

control problem governed by the Navier-Stokes equations formally stated as follows:

$$\underset{\mathbf{u},p,\mathbf{g}}{\text{minimize}} \quad \frac{1}{2}\int_D (\partial_{x_1}\mathbf{u}_2 - \partial_{x_2}\mathbf{u}_1)^2\,dx + \frac{\alpha}{2}\int_{\Gamma_c} |\mathbf{g}|^2\,ds, \tag{4.2a}$$

subject to

$$-\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{0} \quad \text{in} \quad \Omega, \tag{4.2b}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{in} \quad \Omega, \tag{4.2c}$$

$$(\nu\nabla\mathbf{u} - pI)\mathbf{n} = \mathbf{0} \quad \text{on} \quad \Gamma_{out}, \tag{4.2d}$$

$$(\nu\nabla\mathbf{u} - pI)\mathbf{n} + \frac{1}{\delta}\mathbf{u} = \frac{1}{\delta}\mathbf{g} \quad \text{on} \quad \Gamma_c, \tag{4.2e}$$

$$\mathbf{u} = \mathbf{b} \quad \text{on} \quad \Gamma_{in}, \tag{4.2f}$$

$$\mathbf{u} = \mathbf{0} \quad \text{on} \quad \partial\Omega \setminus (\Gamma_{in}\cup\Gamma_c\cup\Gamma_{out}). \tag{4.2g}$$

The infinite-dimensional problem formulation as well as the problem discretization are identical to those in [12]. Here $\mathbf{u}$ is the flow velocity, p is the pressure, $\mathbf{g}$ is the control, $\alpha = 10^{-1}$ is the control penalty parameter and $\nu = 5\times 10^{-3}$ is the viscosity parameter. By $\mathbf{n}$ we denote the unit outward normal; the flow at the inflow boundary $\Gamma_{in} = \{0\}\times[0.5,1]$ is given by $\mathbf{b}(x_1,x_2) = 8(x_2-0.5)(1-x_2)$. The boundary condition (4.2e) is a penalized form of the Dirichlet condition $\mathbf{u} = \mathbf{g}$ on $\Gamma_c = \{1\}\times[0,0.5]$, with penalty parameter $\delta = 10^{-5}$. To control the flow, we only apply suction and blowing in wall-normal direction. That is, $\mathbf{g} = g\mathbf{n}$. The computational domain is divided into 352 triangles with a finer mesh across the area in which recirculation occurs. We discretize the Navier-Stokes equations using Taylor-Hood finite elements. The uncontrolled and controlled flows are depicted in Figure 4.3.

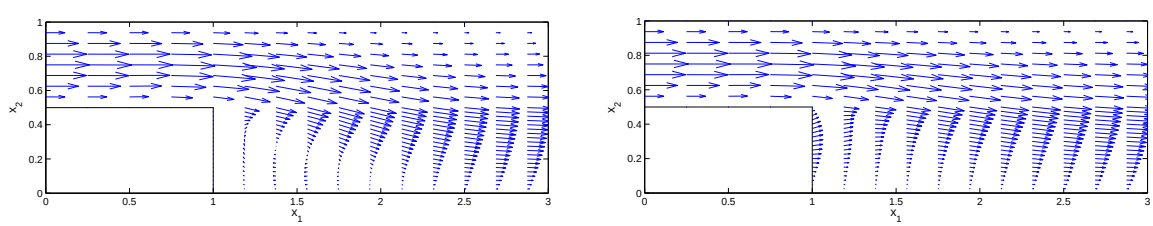

Figure 4.3: Boundary control of the Navier-Stokes equations. Left pane: Plot of the velocity field for the uncontrolled flow, i.e. the solution of the PDEs (4.2b)-(4.2g) for $\mathbf{g} = \mathbf{0}$. Right pane: Plot of the velocity field for the controlled flow, i.e. the velocity solution $\mathbf{u}^*$ of the optimal control problem.

4.3 Boundary Control of the Scalar Ginzburg-Landau Equation

Let $\Omega = [0,1]^2$. We consider an optimal control problem governed by a simplified scalar Ginzburg-Landau equation, stated as follows:

$$\underset{u,g}{\text{minimize}} \quad \frac{1}{2}\int_\Omega (u-\widehat{u})^2\,dx + \frac{1}{2}\int_{\partial\Omega} g^2\,ds \tag{4.3a}$$

subject to

$$-\Delta u + u^3 - u = f \ \text{ in } \Omega, \qquad \frac{\partial u}{\partial n} = g \text{ on } \partial\Omega. \tag{4.3b}$$

See, e.g., [7, 15, 10]. Here u is the state, g is the control, $\widehat{u} = \widehat{u}(x) = \cos(\pi x_1)\cos(\pi x_2)$ is the desired state and $f(x) = \cos(\pi x_1)\cos(\pi x_2)(2\pi^2 + \cos^2(\pi x_1)\cos^2(\pi x_2) - 1)$ is the forcing function. The domain Ω is subdivided into 2048 triangles by first subdividing it first into squares of size $1/32 \times 1/32$ and then subdividing each square into two triangles. The problem (4.3) is discretized using piecewise linear finite elements for states and controls. We omit the plots of the solution.

4.4 Distributed Control of the Poisson-Boltzmann Equation

We consider a distributed control problem related to the design of semiconductor devices. In such applications, a common objective is to match the current measured at an Ohmic contact Γ_o of the device to a prescribed value, by modifying a reference doping profile $\widehat{g}$, see [6]. We consider a simplified model problem, governed by the Poisson–Boltzmann potential equation. We solve

$$\underset{u,g}{\text{minimize}} \quad \frac{\alpha}{2}\int_{\Gamma_o} (\nabla u\cdot\nu - \nabla\widehat{u}\cdot\nu)^2\,ds + \frac{\beta}{2}\int_\Omega (g-\widehat{g})^2\,dx + \frac{\gamma}{2}\int_\Omega |\nabla g|^2\,dx \tag{4.4a}$$

subject to

$$-\Delta u = \exp(u) - \exp(-u) - g \qquad \text{in } \Omega, \tag{4.4b}$$

$$u = 0 \qquad \text{on } \Gamma_D, \tag{4.4c}$$

$$\nabla u\cdot\nu = 0 \qquad \text{on } \Gamma_N, \tag{4.4d}$$

where $\alpha, \beta, \gamma > 0$, ν is the outward unit normal, u is the potential, and g is the doping. The infinite-dimensional problem formulation as well as the problem discretization are identical to those in [9]. Our computational domain is $\Omega = (0,12)\times(0,6)$. We let $\Gamma_D = ((0,2)\cup(5,7)\cup(10,12))\times\{6\}$ (source, gate, drain, see Fig. 4.4), $\Gamma_N = \partial\Omega\setminus\Gamma_D$, and $\Gamma_o = (10,12)\times\{6\}$ (drain). We attempt to match the flux $\nabla\widehat{u}\cdot\nu = 0$ on Γ_o. The problem is discretized using piecewise linear finite elements on a regular mesh with 4096 (64×32×2) triangles. We solve (4.4) for a target doping profile $\widehat{g}$, characterized by two positive numbers a and b. A target doping (a,b) means that $\widehat{g}(x) = a$ for $x \in ([0,2]\times[3,6])\cup([10,12]\times[3,6])$, and $\widehat{g}(x) = b$ elsewhere, see Fig. 4.4. Cost functional parameters are $\alpha = b$, $\beta = b\cdot 10$, and $\gamma = b\cdot 10^{-5}$. We set $a = 10^3$ and $b = 10^{-3}$. The problem (4.4) is discretized using piecewise linear finite elements. We omit the plots of the solution.

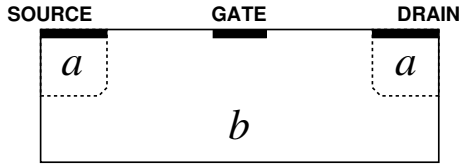

Figure 4.4: Sketch of a MESFET semiconductor device and the corresponding doping profile (a,b).

4.5 Parameter Identification in the Poisson Equation

A prototypical parameter identification (inverse) problem is to estimate the coefficient κ from measurements $\widehat{u}$ related to the solution u of the elliptic boundary value problem

$$-\nabla \cdot (\kappa \nabla u) = f \qquad \text{in } \Omega, \tag{4.5a}$$

$$u = 0 \qquad \text{on } \partial\Omega\,. \tag{4.5b}$$

Here $\Omega \subseteq \mathbb{R}^d$, $d \in \{1, 2, 3\}$, is the computational domain with boundary $\partial\Omega$. For a steady-state heat equation, κ is the coefficient of thermal diffusion, u is the temperature distribution, $\widehat{u}$ are the temperature measurements and f is a given heat source.

We let $\Omega = (-1, 1)^2$. We consider the problem

$$\underset{\{u,\kappa\}\in\, \mathcal{U}\times\mathcal{K}}{\text{minimize}} \quad \frac{1}{2}\int_\Omega (u - \widehat{u})^2\, dx \;+\; R(\kappa) \tag{4.6a}$$

subject to

$$-\nabla \cdot (\kappa \nabla u) = f \text{ in } \Omega, \tag{4.6b}$$

where $\mathcal{U} = \{u\colon u \in H^1(\Omega),\ u = 0 \text{ on } \partial\Omega\}$, $\mathcal{K} = \{\kappa\colon \kappa \in L^2(\Omega),\ \kappa > 0\}$ and $R(\cdot)$ denotes a regularization functional. For simplicity of implementation, we restrict the search space for the diffusion coefficient κ to $H^1(\Omega)$. This enables the use of the total variation regularization functional

$$R(\kappa) = \frac{\alpha}{2}\sqrt{|\nabla\kappa|^2 + \beta}\,. \tag{4.7}$$

We set $\alpha = 10^{-7}$, $\beta = 10^{-10}$. We do not enforce the constraint $\kappa > 0$ explicitly, but verify that it is satisfied, in the discrete sense, after the inverse problem is solved numerically. The problem is discretized using piecewise linear finite elements on a regular grid with 2048 ($32\times32\times2$) triangles. We call this grid the computational grid. We use a sinusoidal heat source, $f(x) = \sin(\pi x_1)\sin(\pi x_2)$. The measurements $\widehat{u}$ are synthesized from the target coefficient given in Figure 4.5 (left pane). To synthesize $\widehat{u}$ we use a finer grid, with 32768 ($128\times128\times2$) triangles, and project the result onto the computational grid. The result of parameter identification is shown in Figure 4.5 (right pane), albeit for a slightly different problem setup.

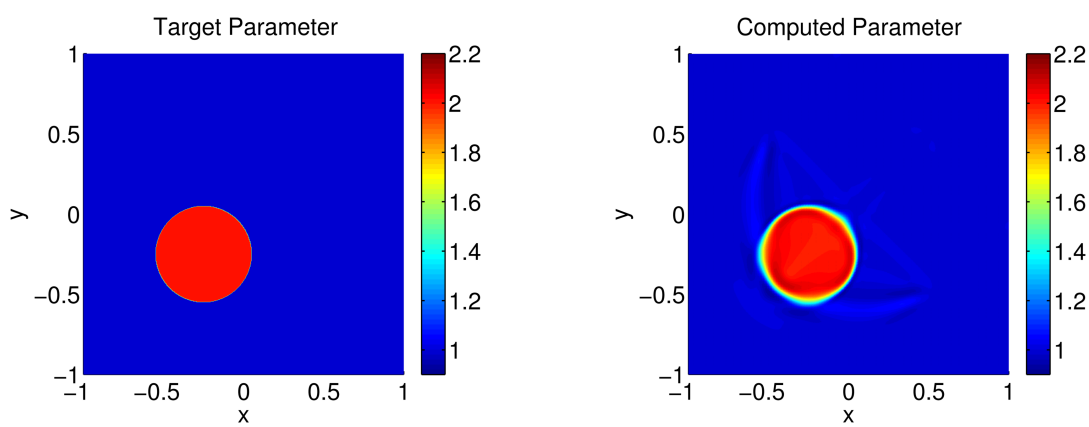

Figure 4.5: Parameter identification in the Poisson equation. Left pane: Target diffusion coefficient; $\kappa = 2$ within a circle of radius 0.3 centered at $(-0.25, -0.25)$, and $\kappa = 1$ everywhere else. Right pane: Computed diffusion coefficient, i.e. the solution κ^* of the inverse problem; for this illustration we used a finer computational grid, 128×128, a finer synthetic grid, 512×512, and $\alpha = 10^{-10}$.

4.6 Parameter Identification in Linear Elastodynamics

In this example we want to identify the shear modulus and the bulk modulus from measurements of the displacements in linear elastodynamics.

Let $\Omega \subseteq \mathbb{R}^d$, $d \in \{1, 2, 3\}$, be the computational domain with boundary $\partial\Omega$, and $\Gamma_u \subset \partial\Omega$ and $\Gamma_\tau \subset \partial\Omega$ are the regions where Dirichlet and Neumann conditions are applied, respectively. In the following we use standard tensor notation with Einstein summation for elasticity equations given in frequency domain. The indices i, j and k take on the values $1, \ldots, d$. Partial differentiation is denoted by a comma.

We consider the PDEs of linear elastodynamics in frequency domain. Here u_i is the (complex-valued) displacement in the i-th direction and τ_i is the surface traction in the i-th direction. Mass density is denoted by ρ, while ω stands for the angular frequency. The shear modulus is μ and the bulk modulus is κ.

The PDEs of linear elastodynamics in frequency domain are given by

$$\sigma_{ij,j} = -\omega^2 \rho u_i \qquad \text{in } \Omega, \tag{4.8a}$$

$$u_i = 0 \qquad \text{on } \Gamma_u, \tag{4.8b}$$

$$\sigma_{ij} n_j = \tau_i \qquad \text{on } \Gamma_\tau\,. \tag{4.8c}$$

The stress tensor σ_{ij} is given by

$$\sigma_{ij} = 2\mu(\epsilon_{ij} - \frac{1}{3}\epsilon_{kk}\delta_{ij}) + \kappa\epsilon_{kk}\delta_{ij}, \tag{4.9}$$

where δ_{ij} is the Kronecker delta and where the strain tensor ϵ_{ij} is defined to be the symmetric part of the displacement gradient, as follows,

$$\epsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}). \tag{4.10}$$

We let $\Omega = (-1, 1)^2$. Furthermore, let $\widehat{u}_i$ be the displacement measurements in the i-th direction We consider the problem

$$\underset{\{u_1, u_2, \mu, \kappa\} \in\, \mathcal{U}_1 \times \mathcal{U}_2 \times \mathcal{G} \times \mathcal{B}}{\text{minimize}} \quad \frac{1}{2}\int_\Omega \sum_{i=1}^{2} (u_i - \widehat{u}_i)\overline{(u_i - \widehat{u}_i)}\, dx \;+\; R(\mu) \;+\; R(\kappa) \tag{4.11a}$$

subject to

$$\sigma_{ij,j} = -\omega^2 \rho u_i \qquad \text{in } \Omega, \tag{4.11b}$$

$$\sigma_{ij} n_j = \tau_i \qquad \text{on } \Gamma_\tau, \tag{4.11c}$$

where $i, j, k \in \{1, 2\}$, $\mathcal{U}_i = \{u_i \colon u_i \in H^1(\Omega),\ u_i = 0 \text{ on } \Gamma_u\}$, $\mathcal{G} = \{\mu \colon \mu \in L^2(\Omega),\ \mu > 0\}$, $\mathcal{B} = \{\kappa \colon \kappa \in L^2(\Omega),\ \kappa > 0\}$ and the overline denotes complex conjugation. We note that the L^2 inner product in $\mathcal{U}_i$ is defined by

$$(u_i, v_i) := \int_\Omega u_i(x)\overline{v}_i(x)dx.$$

To simplify the implementation, we restrict the search spaces for μ and κ to $H^1(\Omega)$. This enables the use of the total variation regularization functional defined in (4.7). We set $\alpha = 5 \cdot 10^{-7}$, $\beta = 10^{-10}$. We do not enforce the constraints $\mu > 0$ and $\kappa > 0$ explicitly, but verify that they are satisfied, in the discrete sense, after the inverse problem is solved numerically. We define $\Gamma_u = [-1, 1] \times \{-1\}$ and $\Gamma_\tau = [-1, 1] \times \{1\}$, in other words Dirichlet and Neumann conditions are applied on the bottom

and top boundaries of the domain, respectively. To generate an elastic response, we use a constant surface traction on Γ_τ, with values $\tau_1 = 1 + i$ and $\tau_2 = 10 + 10i$. We set $\rho = 1$ and $\omega = 1$.

The problem is discretized using piecewise linear finite elements on a regular grid with 32768 ($128\times128\times2$) triangles. We call this grid the computational grid. The measurements $\{\widehat{u}_i\}$ are synthesized from the target shear and bulk moduli given in Figure 4.6 (top left and bottom left panes, respectively). The target values, in both cases, are 1 for the blue region (layer 1), 2 for the green region (layer 2), 2.4 for the yellow region (layer 3), 2.7 for the orange region (square) and 3 for the red region (circle). The straight lines separating layer 1 from layer 2 and layer 2 from layer 3 are given by

$$y(x) = 0.25x \quad \text{and} \quad y(x) = 0.6x - 0.3\,,$$

respectively. We note that the lines intersect near the point $(0.86, 0.21)$. The square is centered at $(0.33, -0.37)$, with side length 0.7. The circle is centered at $(0.53, -0.53)$, with radius 0.25.

To synthesize $\{\widehat{u}_i\}$ we use a finer grid, with 524288 ($512\times512\times2$) triangles, and project the result onto the computational grid. The result of parameter identification is shown in Figure 4.6 (top right and bottom right panes).

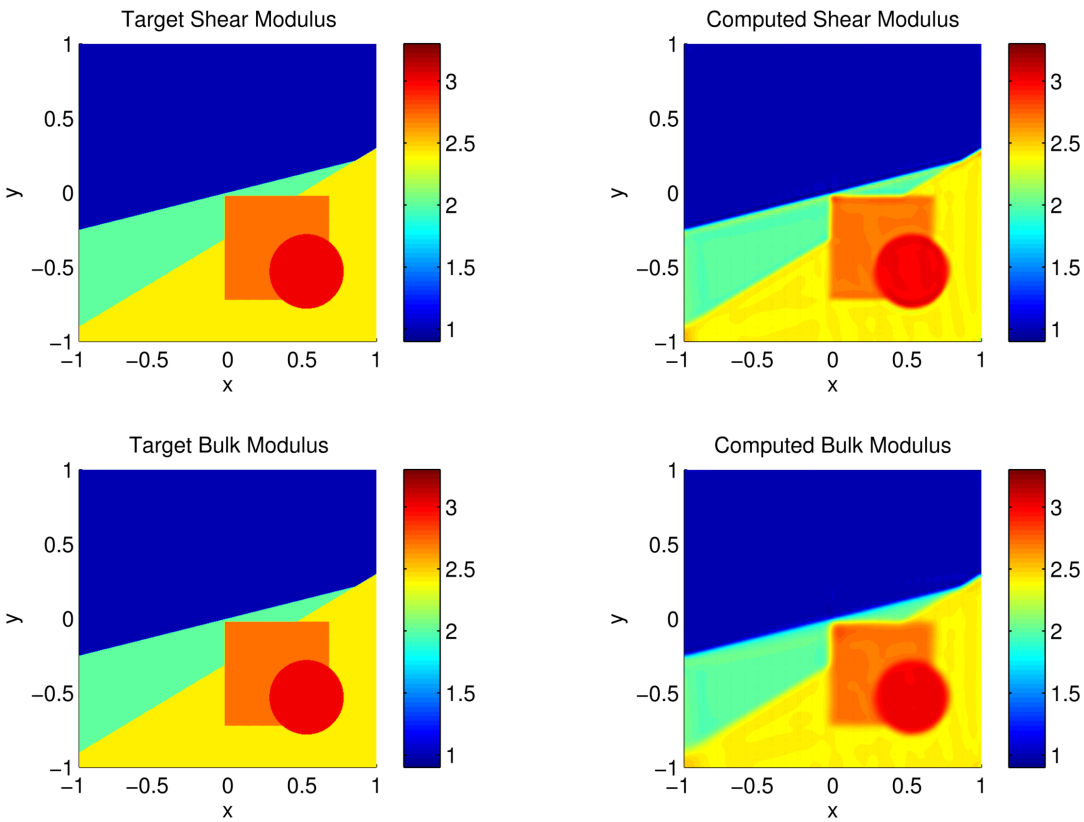

Figure 4.6: Parameter identification in linear elastodynamics. Top left pane: Target shear modulus. Bottom left pane: Target bulk modulus. Top right pane: Computed shear modulus, i.e. the parameter solution μ^* of the inverse problem. Bottom right pane: Computed bulk modulus, i.e. the parameter solution κ^* of the inverse problem.

5 Numerical Results

In this section we study the application of our trust-region SQP algorithm with inexact linear system solves, Algorithm 1, to the example problems defined in Section 4. We point out that the only linear systems that need to be solved in our implementation of Algorithm 1 are of the type

$$\begin{pmatrix} I & c_x(x_k)^* \\ c_x(x_k) & 0 \end{pmatrix} \begin{pmatrix} z \\ y \end{pmatrix} = \begin{pmatrix} b^1 \\ b^2 \end{pmatrix}. \tag{5.1}$$

These are Karush-Kuhn-Tucker (KKT) systems that correspond to convex quadratic programs

$$\begin{array}{ll} \min & \frac{1}{2}\langle z, z\rangle_{\mathcal{X}} - \langle b^1, z\rangle_{\mathcal{X}} \\ \text{s.t.} & c_x(x_k)z = b^2. \end{array}$$

Hence, iterative KKT solvers such as those discussed in [1, 2, 3, 4, 5, 13] can be safely used. This is true even if the subproblems

$$\begin{array}{ll} \min & \frac{1}{2}\langle H_k s, s\rangle_{\mathcal{X}} + \langle \nabla_x \mathcal{L}(x_k, \lambda_k), s\rangle_{\mathcal{X}} + \mathcal{L}(x_k, \lambda_k) \\ \text{s.t.} & c_x(x_k)s + c(x_k) = 0, \quad \|s\|_{\mathcal{X}} \le \Delta_k \end{array}$$

are not convex!

5.1 Linear System Solvers

In our numerical examples all instances of (5.1) are solved using a left-preconditioned GMRES algorithm. The GMRES stopping tolerances (3.3), (3.6), (3.8), (3.10) and (3.13) are imposed on the true (unpreconditioned) residual. The computation of the true residual at every GMRES iteration necessitates the solution of the projected least-squares problem (to obtain the iterate) and an additional application of the augmented system matrix (to compute the residual). In practice, one may perform the convergence checks less frequently — for instance, every third GMRES iteration. Right preconditioning may be used as well, however the cost of applying the preconditioner at every (or even every third) GMRES iteration to recover the iterate may be undesirable. We note that the augmented system matrices are symmetric. Thus if a symmetric positive definite preconditioner is used, GMRES can be replaced by MINRES or SYMMLQ.

For examples discussed in Sections 4.1, 4.2, 4.4 and 4.5 we use an incomplete-LU preconditioner. For the example discussed in Section 4.3 we report results with an incomplete-LU preconditioner and a domain-decomposition preconditioner related to [1, 8]. For the example discussed in Section 4.6 we use an approximate Schur preconditioner that is built as follows. Assuming a splitting of optimization variables x into state variables u and control variables (parameters) g, with $x_k = (u_k, g_k)$, the augmented system matrices may be written as 3×3 block matrices

$$\begin{pmatrix} I & 0 & c_u(x_k)^* \\ 0 & I & c_g(x_k)^* \\ c_u(x_k) & c_g(x_k) & 0 \end{pmatrix}.$$

Further assuming the existence of $(c_u(x_k))^{-1}$, our approximate Schur preconditioner is given by

$$\begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & c_u(x_k)^{-*}c_u(x_k)^{-1} \end{pmatrix}.$$

5.2 Algorithm Parameters

To examine the benefits of inexactness control in Algorithm 1, we compare it to a closely related conventional trust-region SQP algorithm in which the direct solves of augmented systems (5.1) are replaced by preconditioned GMRES solves with a fixed relative tolerance τ. In other words, for the solution of systems (5.1) that arise in the conventional SQP method, GMRES is stopped if the preconditioned residual is below $\tau(\|b^1\| + \|b^2\|)$. Here τ is a tolerance that is *fixed throughout* the SQP iterations. This is what one may do when forced to retrofit an existing NLP code with iterative linear system solves. It is important to point out that in addition to the static treatment of linear solver tolerances, the conventional SQP algorithm contains none of the global convergence safeguards of Algorithm 1 to deal with inexactness.

For brevity, we use the label I-SQP for our matrix-free algorithm with inexactness control, Algorithm 1, and the label C-SQP for the conventional algorithm described above. The algorithm parameters shared between I-SQP and C-SQP are[2]:

tol^{SQP}	tol^{CG}	$i^{CG}_{\max}$	ζ	Δ_0	Δ_{min}	Δ_{max}	α_1	η_1	η_0	ρ_{-1}	$\bar{\rho}$
10^{-6}	10^{-2}	200	0.8	10^2	10^{-10}	10^8	0.5	10^{-8}	0.5	1	10^{-8}

We introduce a *nominal* relative linear solver tolerance, tol^{LS}, which is varied in our experiments. Given tol^{LS}, the remaining I-SQP parameters are set to

$$\xi^{qn} = \xi^{pg} = \xi^{proj} = \xi^{tang} = \xi^{lmh} = tol^{LS}, \qquad \xi_4 = 2 \qquad \text{and} \qquad \xi^{lmg} = 10^4 .$$

The fixed relative solver tolerance used in C-SQP is set to $\tau = tol^{LS}$.

For the initial guesses x_0 and λ_{-1} we use constants[3]. To build incomplete-LU preconditioners we use the Matlab® function `luinc` with drop tolerances lut, applied to shifted augmented systems with the diagonal shift $lus \times I$. The corresponding values are:

Burgers	N.-Stokes	G.-Landau	P.-Boltzmann	Inverse Poisson	Inverse Elasticity
$x_0 = -1$	$x_0 = 0$	$x_0 = 1$	$x_0 = 0$	$\kappa_0 = 0.5$	$\mu_0 = \kappa_0 = 1.8$
$\lambda_{-1} = 0$	$\lambda_{-1} = 0$	$\lambda_{-1} = 0$	$\lambda_{-1} = 0$	$\lambda_{-1} = 0$	$\lambda_{-1} = 0$
$lut = 6\times10^{-3}$	$lut = 5\times10^{-5}$	$lut = 1\times10^{-2}$	$lut = 1\times10^{-3}$	$lut = 1\times10^{-2}$	N/A
$lus = 1\times10^{-2}$	$lus = 1\times10^{-7}$	$lus = 0$	$lus = 1\times10^{-4}$	$lus = 1\times10^{-4}$	N/A

For the example discussed in Section 4.3 (Ginzburg-Landau), where we also exercise a domain-decomposition preconditioner, the computational domain is divided uniformly into 64 (8×8) subdomains. The preconditioner is of the two-level balancing Neumann-Neumann type [8].

5.3 Reporting of Results for Individual Test Problems

In Section 5.5 below we summarize the performance of the inexact SQP method on the examples introduced in Section 4, in the form of tables and figures. Here we describe how to read the tables and figures.

[2]For the parameter identification examples, Examples 4.5 and 4.6, the SQP solution tolerance $tol^{SQP} = 10^{-8}$ is used in order to allow for sufficiently accurate solutions when the regularization parameters α are very small.

[3]For the parameter identification examples, Examples 4.5 and 4.6, we use constant initial guesses for the parameters κ and (μ, κ), respectively, and compute the initial PDE states by solving the governing PDEs. We note that the initial optimization iterates will therefore satisfy the constraints. At the same time, our SQP algorithm need not (and does not, for Examples 4.5 and 4.6) maintain feasibility of the subsequent iterates.

Table 5.1 is an example of a table that summarizes the performance of the inexact SQP algorithm (I-SQP), Algorithm 1, as the linear system solver tolerance tol^{LS} is changed. For each column in Table 5.1 we report:

- GMRES iters: This is the total number of GMRES iterations executed in I-SQP. For our examples, the computational cost of running I-SQP is typically proportional to the total number of GMRES iterations.
- GMRES calls: This is how often GMRES is called, i.e., how many augmented systems (5.1) are solved during an I-SQP run. Typically the total number of GMRES calls decreases as tol^{LS} decreases.
- GMRES iters/call: This is the average number of GMRES iterations executed, where the average is taken over the total number of GMRES iterations and the total number of GMRES calls during an I-SQP run.
- GMRES prec builds: This is how often the preconditioner is constructed. The preconditioner for (5.1) can be reused as long as x_k is unchanged.
- STCG iters: The total number of Steihaug-Toint CG iterations executed during an I-SQP run.
- SQP iters: The total number of I-SQP iterations executed. Note that we increase the iteration count also when an iteration is unsuccessful, i.e., $x_{k+1} = x_k$. Typically the total number of I-SQP iterations decreases as tol^{LS} decreases.
- a posteriori refine: Number of times during an I-SQP run the condition "$\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ and $\widetilde{t}_k = \widetilde{t}_k^{cp}$" in Step 2(c)ii of I-SQP Algorithm 1 is true. If this condition is true, the quasi-normal step, the tangential steps, and the Lagrange multipliers have to be recomputed with a finer linear solver tolerance.
- nonconvex QP: How often the Steihaug-Toint CG method for solving the tangential subproblem terminated because negative curvature was detected (Algorithm 3, Step 1d). Note that because inexact projections are used, negative curvature may occur even if the exact tangential subproblem is strictly convex, or negative curvature may not be detected even if the exact tangential subproblem is not convex.

Table 5.1: Performance of the inexact SQP algorithm (I-SQP) for varying linear system solver tolerance tol^{LS}.

Nominal Tol tol^{LS}		**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure		S	S	S	S	S	S	S	S	S
GMRES	iters	3809	3889	2255	2355	2704	2865	2990	3132	3273
	calls	407	471	188	173	186	188	188	188	188
	iters/call	9.4	8.3	12.0	13.6	14.5	15.2	15.9	16.7	17.4
	prec builds	109	106	26	21	21	21	21	21	21
STCG iters		163	240	130	124	135	137	137	137	137
SQP iters		78	76	19	15	15	15	15	15	15
a posteriori refine		2	0	0	0	0	0	0	0	0
nonconvex QP		3	6	7	7	7	7	7	7	7

In addition to a table, like Table 5.1, that summarizes the performance of the I-SQP Algorithm 1 as the linear system solver tolerance tol^{LS} is changed, we also provide a figure, like Figure 5.1, for

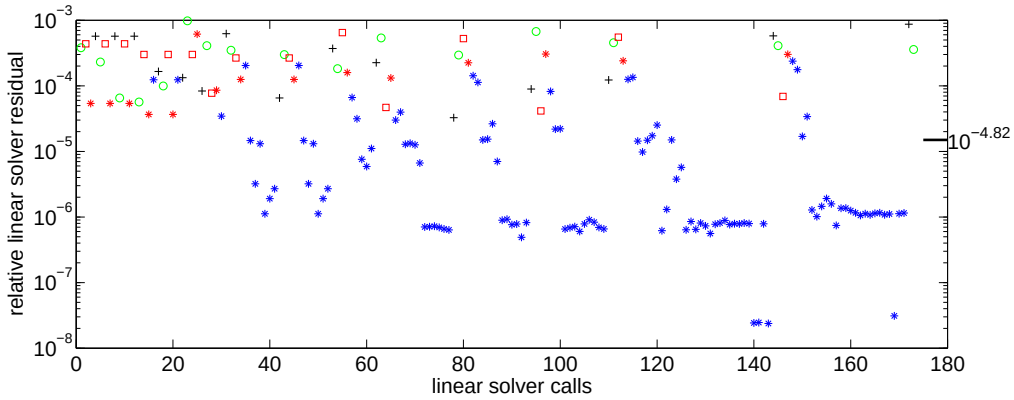

Figure 5.1: Plots of linear solver residuals in I-SQP run

a specific nominal solver tolerance. For every call to GMRES within the I-SQP Algorithm 1 this figure shows the relative linear system residuals recorded at the final GMRES iteration. Figure 5.1 corresponds to an I-SQP run with $tol^{LS} = 10^{-3}$. Table 5.1 shows that for $tol^{LS} = 10^{-3}$ GMRES was called 173 times. Figure 5.1 shows the achieved relative residuals for each of the 173 GMRES calls. The different colors indicate in which subproblem the GMRES call occurs.

- The red squares denote quasi-normal step computations with tolerance (3.3). Note that if the For-Loop in Step 2c of I-SQP Algorithm 1 is always executed with $i = 0$, then n_k^N and δn_k are computed at the beginning of every SQP iteration. For $tol^{LS} = 10^{-3}$, Table 5.1 shows that a posteriori refine equals 0, i.e., that each red square denotes the beginning of an SQP iteration.

- The red asterisks correspond to the first STCG iterations with tolerance (3.6). If the For-Loop in Step 2c of I-SQP Algorithm 1 is always executed with $i = 0$, then STCG is only called once per SQP iteration and the figure shows one red asterisk for each SQP iteration.

- The blue asterisks denote the subsequent STCG iterations with tolerance (3.8).

- The black crosses correspond to tangential step computations with tolerance (3.10). The tangential step t_k is computed in Step 2(c)iA of I-SQP Algorithm 1. Hence, if there is only one tangential step computation per I-SQP iteration (one black cross between red squares), then the For-Loop in Step 2c and the For-Loop in Step 2(c)i of I-SQP Algorithm 1 are executed with $i = 0$ and $j = 0$, respectively. This is the case in Figure 5.1.

- The green circles denote Lagrange multiplier computations with tolerance (3.13). The Lagrange multiplier λ_k is computed in Step 2(c)iB of I-SQP Algorithm 1. Hence, if there is only one Lagrange multiplier computation per I-SQP iteration (one green circle between red squares), then the For-Loop in Step 2c and the For-Loop in Step 2(c)i of I-SQP Algorithm 1 are executed with $i = 0$ and $j = 0$, respectively. This is the case in Figure 5.1.

In Section 5.5 below we also include figures with absolute linear system residuals. The marker colors and styles in those plots are identical to the ones in the relative residual plots.

We also note that each relative residual plot includes a number on the right vertical axis, which denotes the median value of the residuals on the logarithmic scale. This number is a good indication of how accurately, on average, the augmented systems (5.1) are solved for a given nominal tolerance tol^{LS}. The absolute residual plots do not include this number.

5.4 Summary of Results

All computations are performed using Matlab® Version 7.11.0.584 (R2010b) 64-bit, on a Dual Six Core Intel Xeon X5680 3.33GHz platform. Our results are summarized in Tables 5.2-5.13 and in Figures 5.2-5.7.

Tables 5.2, 5.4, 5.6, 5.8, 5.10, 5.12 and 5.14 document the performance of the I-SQP algorithm as the nominal relative linear solver tolerance tol^{LS} is decreased from 0.5 to 10^{-8}. First, we observe that I-SQP is extremely robust. All nominal tolerances yield successful runs, for all examples. As reported in Figures 5.2-5.8 the algorithm dynamically chooses linear solver tolerances as it makes progress toward a solution. The nominal tolerances tol^{LS} can be interpreted as upper bounds on the relative augmented system residuals used by I-SQP. At the same time, the *median* relative residuals are not significantly smaller than tol^{LS}, which means that our algorithm keeps the oversolving of linear systems in check.

Second, in contrast, C-SQP exhibits lack of robustness, shown in Tables 5.3, 5.5, 5.7, 5.9, 5.11, 5.13 and 5.15. The nominal linear solver tolerances required for convergence can be extremely small. In three cases, Table 5.11 (Poisson-Boltzmann), Table 5.13 (inverse Poisson) and Table 5.15 (inverse elasticity) *none* of the nominal tolerances yield successful runs. This very clearly demonstrates the importance of inexactness control.

Third, focusing on the rows GMRES iters/call and SQP iters in Tables 5.2, 5.4, 5.6, 5.8, 5.10 and 5.12, it is evident that I-SQP enables tradeoffs between the accuracy of linear system solves and the number of optimization iterations. This is especially pronounced in Table 5.8 (Ginzburg-Landau, domain decomposition), with values 2.0/17 and 9.2/8 for $tol^{LS} = 0.5$ and $tol^{LS} = 10^{-8}$, respectively, Table 5.10 (Poisson-Boltzmann), with values 1.7/34 and 13.3/18 for $tol^{LS} = 0.5$ and $tol^{LS} = 10^{-8}$, respectively, and Table 5.12 (inverse Poisson), with values 4.8/323 and 22.7/10 for $tol^{LS} = 0.5$ and $tol^{LS} = 10^{-8}$, respectively.

Fourth, it is clear that using very fine nominal linear solver tolerances tol^{LS} is unnecessary. Instead, the nominal tolerance $tol^{LS} = 10^{-3}$ is an excellent choice, for all examples.

Fifth, we note that only for the very coarse nominal tolerance of $tol^{LS} = 0.5$ the safeguard "$\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ and $\widetilde{t}_k = \widetilde{t}_k^{cp}$" is activated in Algorithm 1. Moreover, this happens only once or twice per run, and only for three of the seven studies (Burgers, Navier-Stokes and Ginzburg-Landau with incomplete-LU preconditioning), see the a posteriori refine rows of Tables 5.2, 5.4 and 5.6. In other words, there are virtually no recomputations of optimization substeps. Additionally, we observed that for the nominal tolerances $tol^{LS} < 10^{-1}$ the For-Loop in Step 2c and the For-Loop in Step 2(c)i of I-SQP Algorithm 1 are executed consistently with $i = 0$ and $j = 0$. This means that I-SQP automatically selects, *a priori*, adequate solver tolerances for augmented systems. We also remark that for simplicity we used a single nominal tolerance tol^{LS} for ξ^{qn}, ξ^{pg}, ξ^{proj}, ξ^{tang} and ξ^{lmh}. Better individual choices of nominal tolerances are possible for increased computational performance; in particular, setting $\xi^{tang} \to 10^{-1}\xi^{proj}$ executes the For-Loop in Step 2(c)i of I-SQP Algorithm 1 with $j = 0$ for all experiments and reduces the total number of GMRES iterations.

Sixth, nonconvexity in the quadratic subproblems is common. It can be present regardless of the accuracy of linear system solves, see e.g. the last rows of Table 5.2 (Burgers), Tables 5.6 and 5.8 (Ginzburg-Landau), Table 5.10 (Poisson-Boltzmann) and Table 5.14 (inverse elasticity). Furthermore, nonconvexity may be induced by inexact linear system solves, as evidenced by the last row of Table 5.4, $tol^{LS} = 10^{-2}$, or magnified, see the last row of Table 5.14, $tol^{LS} = 10^{-1}$. Therefore, for the chosen studies in PDE-constrained optimization, it is important that the optimization algorithm handle nonconvexity in the quadratic models. The I-SQP algorithm, Algorithm 1, does this robustly and efficiently.

This concludes our study.

5.5 Results for Individual Test Problems

5.5.1 Distributed Control of the Burgers Equation

Table 5.2: Inexact SQP algorithm (I-SQP), Algorithm 1, applied to the distributed control of the Burgers equation. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the incomplete-LU preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3.

Nominal Tol tol^{LS}		**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure		S	S	S	S	S	S	S	S	S
GMRES	iters	3809	3889	2255	2355	2704	2865	2990	3132	3273
	calls	407	471	188	173	186	188	188	188	188
	iters/call	9.4	8.3	12.0	13.6	14.5	15.2	15.9	16.7	17.4
	prec builds	109	106	26	21	21	21	21	21	21
STCG iters		163	240	130	124	135	137	137	137	137
SQP iters		78	76	19	15	15	15	15	15	15
a posteriori refine		2	0	0	0	0	0	0	0	0
nonconvex QP		3	6	7	7	7	7	7	7	7

Table 5.3: Conventional SQP algorithm (C-SQP) applied to the distributed control of the Burgers equation. Failure is reported if the number of SQP iterations exceeds 100.

Nominal Tol tol^{LS}	**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure	F	F	F	F	S	S	S	S	S
GMRES iters	—	—	—	—	3940	11861	9312	7090	6114
SQP iters	—	—	—	—	16	18	16	15	15

Figure 5.2 (*next page*):
Relative and absolute residuals for all augmented systems solved by the I-SQP algorithm, applied to the distributed control of the Burgers equation. First and second pane: $tol^{LS} = 10^{-1}$ (yielding the smallest number of iters/call). Third and fourth pane: $tol^{LS} = 10^{-3}$. Note: the red squares denote quasi-normal step computations with tolerance (3.3); the red asterisks correspond to the first STCG iterations with tolerance (3.6); the blue asterisks denote the subsequent STCG iterations with tolerance (3.8); the black crosses correspond to tangential step computations with tolerance (3.10); the green circles denote Lagrange multiplier computations with tolerance (3.13). Median (log scale) relative residuals are marked by bold lines on the right vertical axes of relative residual plots.

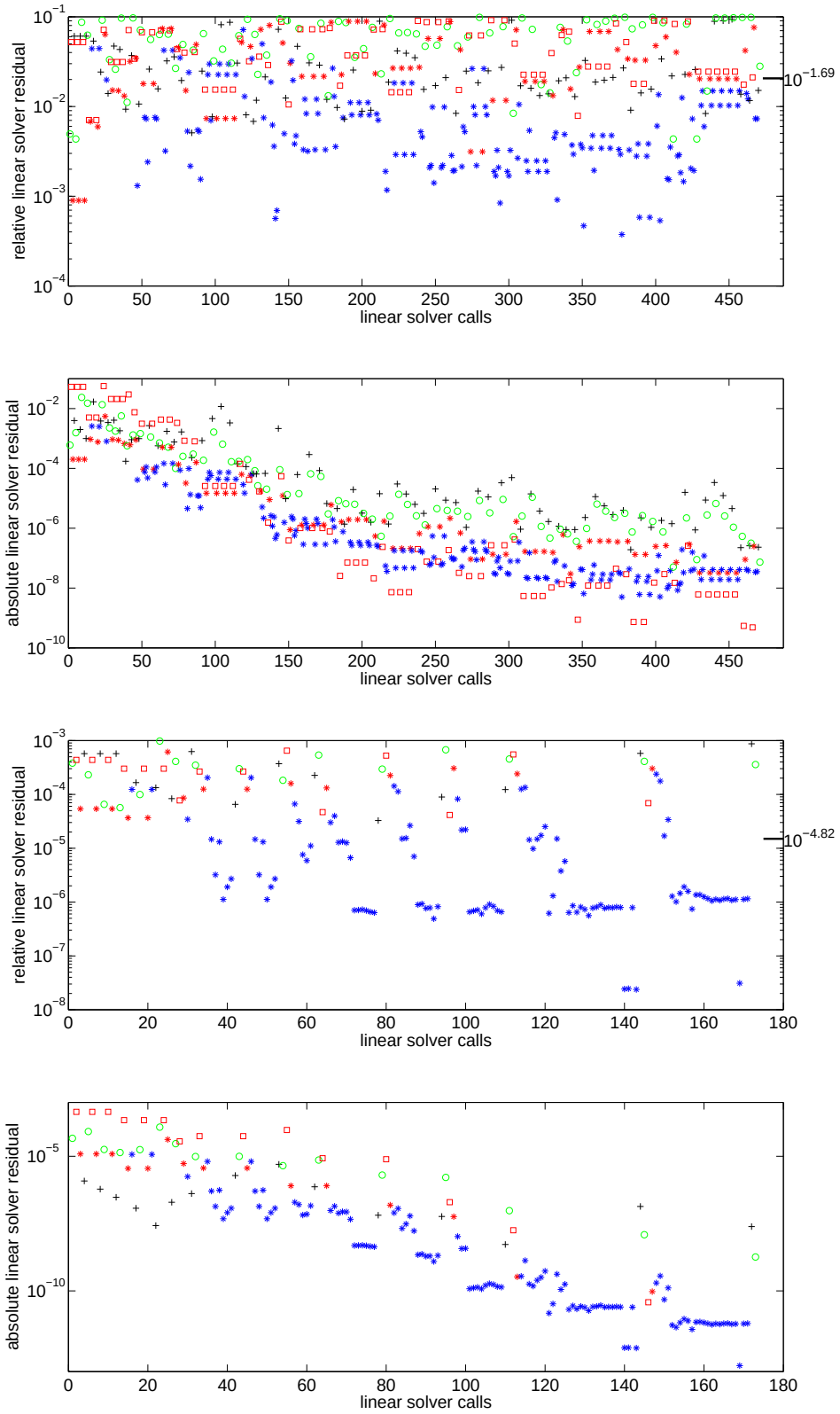

Figure 5.2: Plots of linear solver residuals; distributed control of the Burgers equation.

5.5.2 Boundary Control of the Navier-Stokes Equations

Table 5.4: Inexact SQP algorithm (I-SQP), Algorithm 1, applied to the boundary control of the Navier-Stokes equations. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the incomplete-LU preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3.

Nominal Tol tol^{LS}		**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure		S	S	S	S	S	S	S	S	S
GMRES	iters	2568	3186	2493	2352	2250	2600	2924	3304	3603
	calls	310	474	287	193	176	176	176	176	176
	iters/call	8.3	6.7	8.7	12.2	12.8	14.8	16.6	18.8	20.5
	prec builds	29	34	21	11	9	9	9	9	9
STCG iters		243	389	231	156	143	143	143	143	143
SQP iters		17	28	16	9	8	8	8	8	8
a posteriori refine		2	0	0	0	0	0	0	0	0
nonconvex QP		0	0	2	0	0	0	0	0	0

Table 5.5: Conventional SQP algorithm (C-SQP) applied to the boundary control of the Navier-Stokes equations. Failure is reported if the number of SQP iterations exceeds 100.

Nominal Tol tol^{LS}	**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure	F	F	F	F	F	F	F	S	S
GMRES iters	—	—	—	—	—	—	—	2906	3180
SQP iters	—	—	—	—	—	—	—	8	8

Figure 5.3 (*next page*):
Relative and absolute residuals for all augmented systems solved by the I-SQP algorithm, applied to the boundary control of the Navier-Stokes equations. First and second pane: $tol^{LS} = 10^{-1}$ (yielding the smallest number of iters/call). Third and fourth pane: $tol^{LS} = 10^{-3}$. Note: the red squares denote quasi-normal step computations with tolerance (3.3); the red asterisks correspond to the first STCG iterations with tolerance (3.6); the blue asterisks denote the subsequent STCG iterations with tolerance (3.8); the black crosses correspond to tangential step computations with tolerance (3.10); the green circles denote Lagrange multiplier computations with tolerance (3.13). Median (log scale) relative residuals are marked by bold lines on the right vertical axes of relative residual plots.

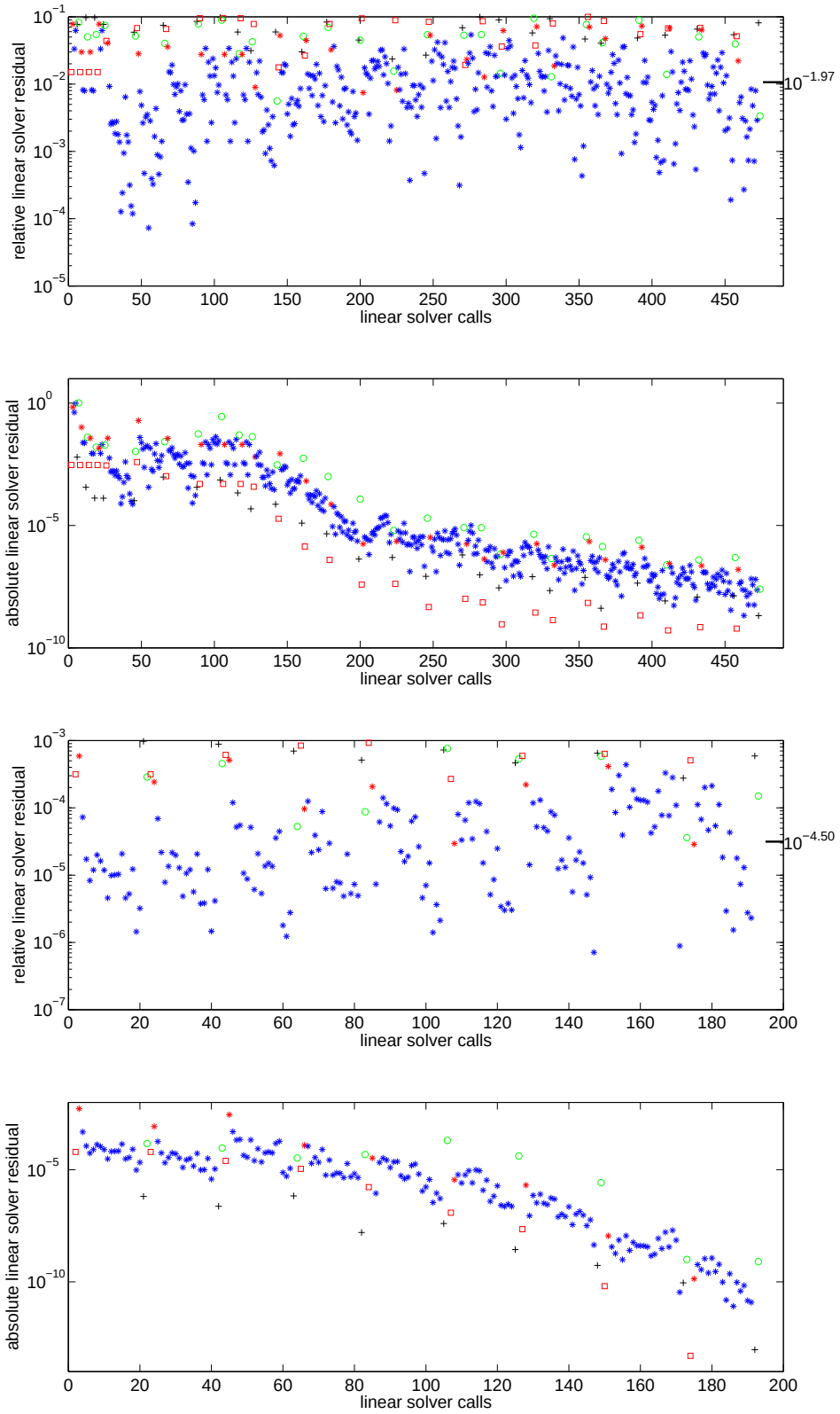

Figure 5.3: Plots of linear solver residuals; boundary control of the Navier-Stokes equations.

5.5.3 Boundary Control of the Scalar Ginzburg-Landau Equation

Incomplete-LU Preconditioner

Table 5.6: Inexact SQP algorithm (I-SQP), Algorithm 1, applied to the boundary control of the scalar Ginzburg-Landau equation, with an incomplete-LU preconditioner. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the incomplete-LU preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3.

Nominal Tol tol^{LS}		**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure		S	S	S	S	S	S	S	S	S
GMRES	iters	1206	1224	1038	990	1094	1180	1262	1340	1412
	calls	115	109	75	64	64	64	64	64	64
	iters/call	10.5	11.2	13.8	15.5	17.1	18.4	19.7	20.9	22.1
	prec builds	23	22	17	16	16	16	16	16	16
STCG iters		65	64	36	29	29	29	29	29	29
SQP iters		14	14	11	10	10	10	10	10	10
a posteriori refine		1	0	0	0	0	0	0	0	0
nonconvex QP		3	6	6	6	6	6	6	6	6

Table 5.7: Conventional SQP algorithm (C-SQP) applied to the boundary control of the scalar Ginzburg-Landau equation, with an incomplete-LU preconditioner. Failure is reported if the number of SQP iterations exceeds 100.

Nominal Tol tol^{LS}	**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure	F	F	F	F	S	S	S	S	S
GMRES iters	—	—	—	—	935	996	1110	1171	1233
SQP iters	—	—	—	—	9	9	9	9	9

Figure 5.4 (*next page*):
Relative and absolute residuals for all augmented systems solved by the I-SQP algorithm, applied to the boundary control of the scalar Ginzburg-Landau equation, with an incomplete-LU preconditioner. First and second pane: $tol^{LS} = 0.5$ (yielding the smallest number of iters/call). Third and fourth pane: $tol^{LS} = 10^{-3}$. Note: the red squares denote quasi-normal step computations with tolerance (3.3); the red asterisks correspond to the first STCG iterations with tolerance (3.6); the blue asterisks denote the subsequent STCG iterations with tolerance (3.8); the black crosses correspond to tangential step computations with tolerance (3.10); the green circles denote Lagrange multiplier computations with tolerance (3.13). Median (log scale) relative residuals are marked by bold lines on the right vertical axes of relative residual plots.

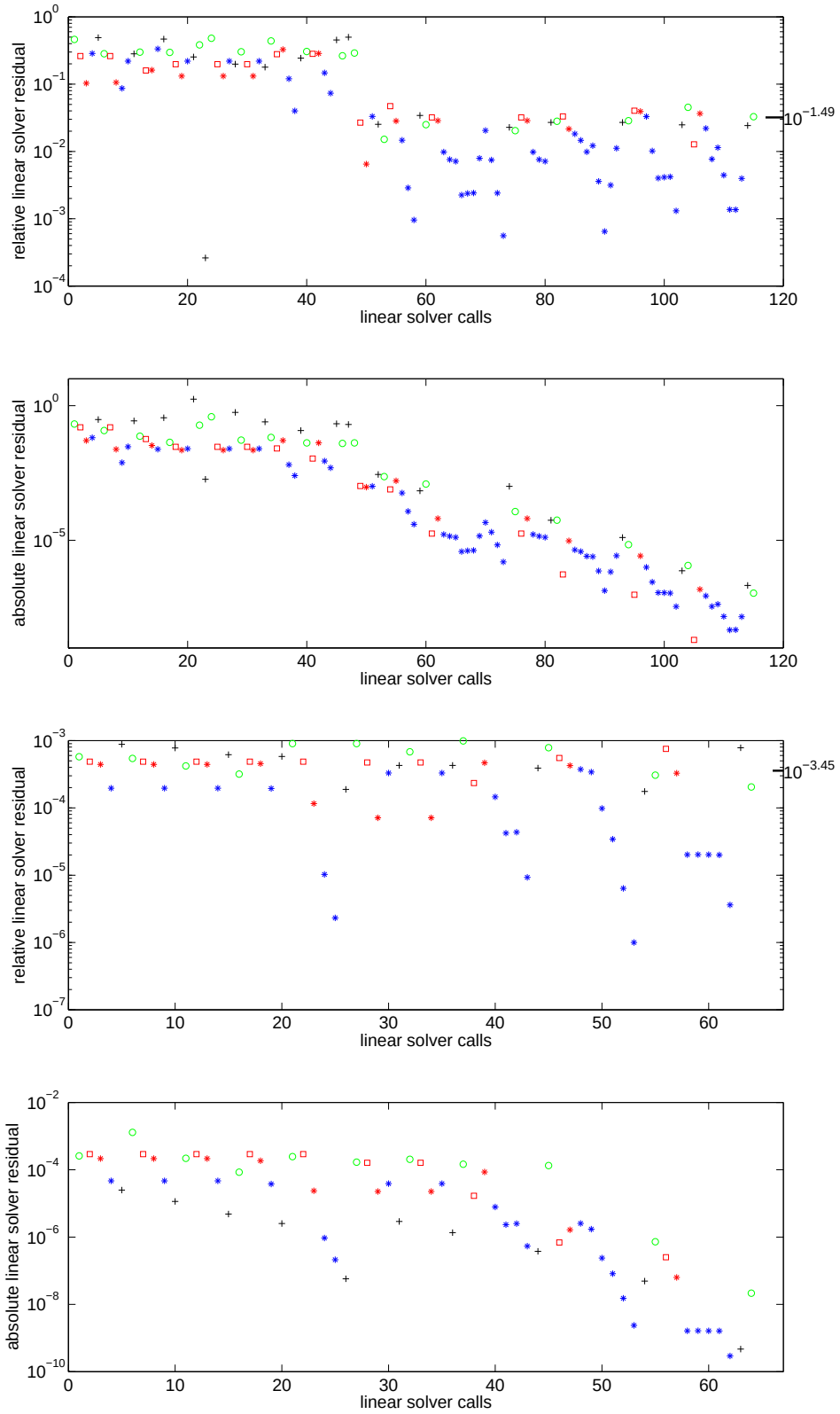

Figure 5.4: Plots of linear solver residuals; boundary control of the scalar Ginzburg-Landau equation; incomplete-LU preconditioning.

Domain-Decomposition Preconditioner.

Table 5.8: Inexact SQP algorithm (I-SQP), Algorithm 1, applied to the boundary control of the scalar Ginzburg-Landau equation, with a domain-decomposition preconditioner. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the domain-decomposition preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3.

Nominal Tol tol^{LS}		**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure		S	S	S	S	S	S	S	S	S
GMRES	iters	359	203	203	254	318	374	447	491	551
	calls	181	86	60	60	60	60	60	60	60
	iters/call	2.0	2.4	3.4	4.2	5.3	6.2	7.5	8.2	9.2
	prec builds	27	15	12	12	12	12	12	12	12
STCG iters		119	52	33	33	33	33	33	33	33
SQP iters		17	10	8	8	8	8	8	8	8
a posteriori refine		0	0	0	0	0	0	0	0	0
nonconvex QP		3	3	3	3	3	3	3	3	3

Table 5.9: Conventional SQP algorithm (C-SQP) applied to the boundary control of the scalar Ginzburg-Landau equation, with a domain-decomposition preconditioner. Failure is reported if the number of SQP iterations exceeds 100.

Nominal Tol tol^{LS}	**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure	F	F	F	S	S	S	S	S	S
GMRES iters	—	—	—	296	370	437	512	560	637
SQP iters	—	—	—	10	10	10	10	10	10

Figure 5.5 (*next page*):
Relative and absolute residuals for all augmented systems solved by the I-SQP algorithm, applied to the boundary control of the scalar Ginzburg-Landau equation, with a domain-decomposition preconditioner. First and second pane: $tol^{LS} = 0.5$ (yielding the smallest number of iters/call). Third and fourth pane: $tol^{LS} = 10^{-3}$. Note: the red squares denote quasi-normal step computations with tolerance (3.3); the red asterisks correspond to the first STCG iterations with tolerance (3.6); the blue asterisks denote the subsequent STCG iterations with tolerance (3.8); the black crosses correspond to tangential step computations with tolerance (3.10); the green circles denote Lagrange multiplier computations with tolerance (3.13). Median (log scale) relative residuals are marked by bold lines on the right vertical axes of relative residual plots.

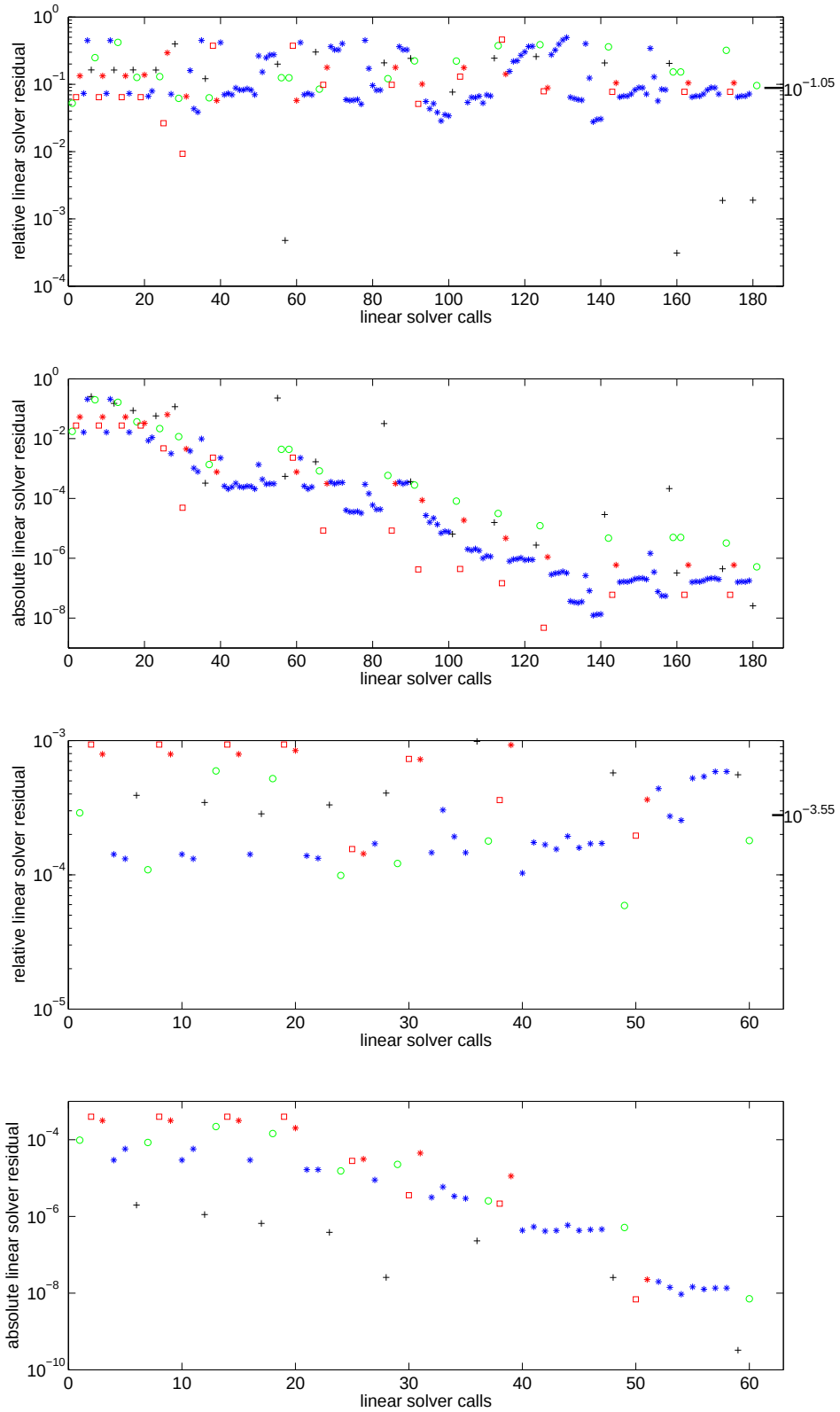

Figure 5.5: Plots of linear solver residuals; boundary control of the scalar Ginzburg-Landau equation; domain-decomposition preconditioning.

5.5.4 Distributed Control of the Poisson-Boltzmann Equation

Table 5.10: Inexact SQP algorithm (I-SQP), Algorithm 1, applied to the distributed control of the Poisson-Boltzmann equation. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the incomplete-LU preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3.

Nominal Tol tol^{LS}		**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure		S	S	S	S	S	S	S	S	S
GMRES	iters	381	380	514	671	844	1032	1218	1408	1601
	calls	223	136	120	120	120	120	120	120	120
	iters/call	1.7	2.8	4.3	5.6	7.0	8.6	10.2	11.7	13.3
	prec builds	42	29	24	24	24	24	24	24	24
STCG iters		117	68	59	59	59	59	59	59	59
SQP iters		34	21	18	18	18	18	18	18	18
a posteriori refine		0	0	0	0	0	0	0	0	0
nonconvex QP		2	2	2	2	2	2	2	2	2

Table 5.11: Conventional SQP algorithm (C-SQP) applied to the distributed control of the Poisson-Boltzmann equation. Failure is reported if the number of SQP iterations exceeds 100.

Nominal Tol tol^{LS}	**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure	F	F	F	F	F	F	F	F	F
GMRES iters	—	—	—	—	—	—	—	—	—
SQP iters	—	—	—	—	—	—	—	—	—

Figure 5.6 (*next page*):
Relative and absolute residuals for all augmented systems solved by the I-SQP algorithm, applied to the distributed control of the Poisson-Boltzmann equation. First and second pane: $tol^{LS} = 0.5$ (yielding the smallest number of iters/call). Third and fourth pane: $tol^{LS} = 10^{-3}$. Note: the red squares denote quasi-normal step computations with tolerance (3.3); the red asterisks correspond to the first STCG iterations with tolerance (3.6); the blue asterisks denote the subsequent STCG iterations with tolerance (3.8); the black crosses correspond to tangential step computations with tolerance (3.10); the green circles denote Lagrange multiplier computations with tolerance (3.13). Median (log scale) relative residuals are marked by bold lines on the right vertical axes of relative residual plots.

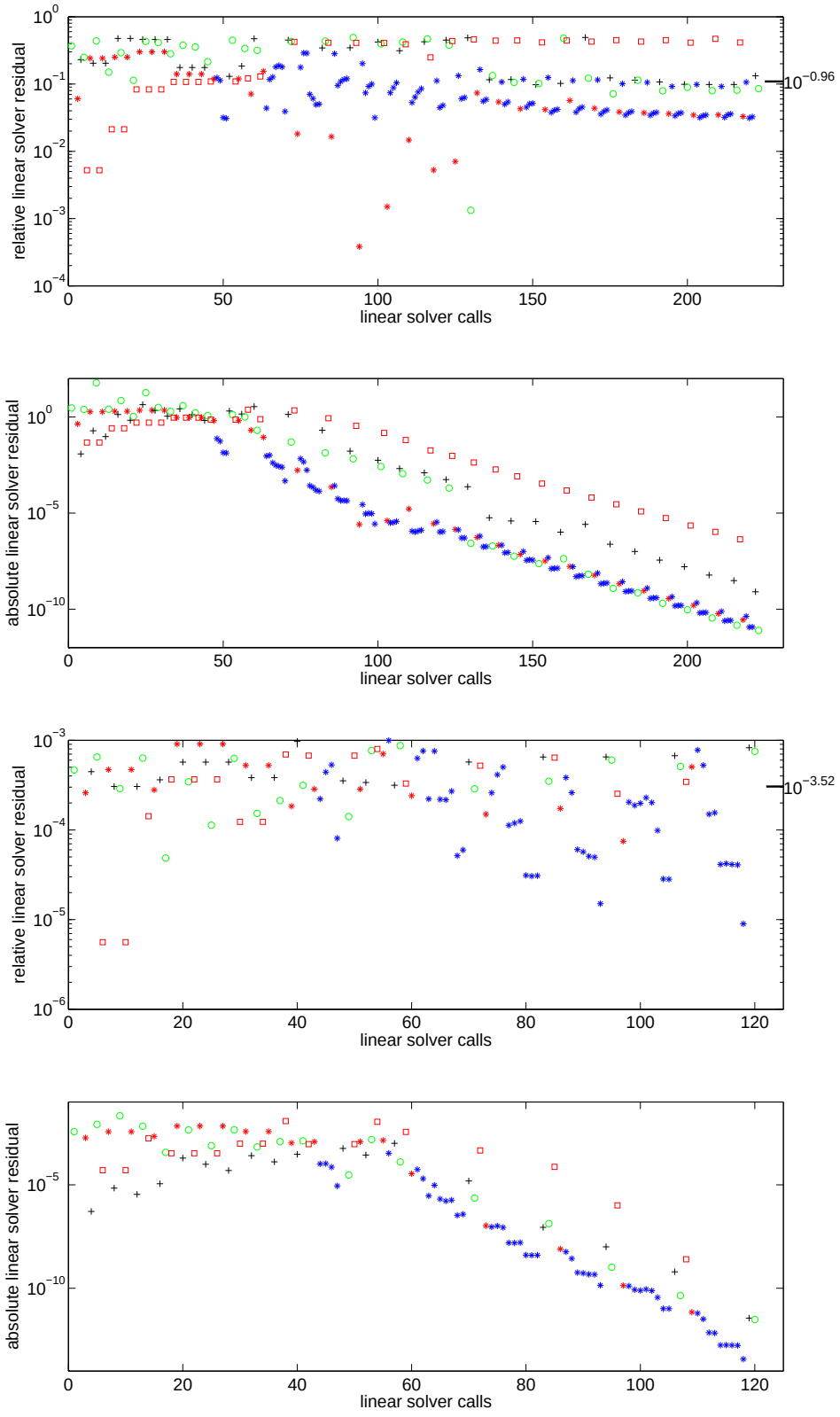

Figure 5.6: Plots of linear solver residuals; distributed control of the Poisson-Boltzmann equation.

5.5.5 Parameter Identification in the Poisson Equation

Table 5.12: Inexact SQP algorithm (I-SQP), Algorithm 1, applied to parameter identification in the Poisson equation. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times the incomplete-LU preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t_k}\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3.

Nominal Tol tol^{LS}		**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure		S	S	S	S	S	S	S	S	S
GMRES	iters	8623	10166	9474	5267	4270	3113	3389	3538	3997
	calls	1778	1660	1042	356	256	171	172	166	176
	iters/call	4.8	6.1	9.1	14.8	16.7	18.2	19.7	21.3	22.7
	prec builds	442	389	275	32	15	10	10	10	12
STCG iters		612	625	444	289	213	134	135	129	137
SQP iters		323	342	199	22	12	9	9	9	10
a posteriori refine		0	0	0	0	0	0	0	0	0
nonconvex QP		0	0	0	0	0	0	0	0	0

Table 5.13: Conventional SQP algorithm (C-SQP) applied to parameter identification in the Poisson equation. Failure is reported if the number of SQP iterations exceeds 1000.

Nominal Tol tol^{LS}	**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure	F	F	F	F	F	F	F	F	F
GMRES iters	—	—	—	—	—	—	—	—	—
SQP iters	—	—	—	—	—	—	—	—	—

Figure 5.7 (*next page*):
Relative and absolute residuals for all augmented systems solved by the I-SQP algorithm, applied to parameter identification in the Poisson equation. First and second pane: $tol^{LS} = 0.5$ (yielding the smallest number of iters/call). Third and fourth pane: $tol^{LS} = 10^{-3}$. Note: the red squares denote quasi-normal step computations with tolerance (3.3); the red asterisks correspond to the first STCG iterations with tolerance (3.6); the blue asterisks denote the subsequent STCG iterations with tolerance (3.8); the black crosses correspond to tangential step computations with tolerance (3.10); the green circles denote Lagrange multiplier computations with tolerance (3.13). Median (log scale) relative residuals are marked by bold lines on the right vertical axes of relative residual plots.

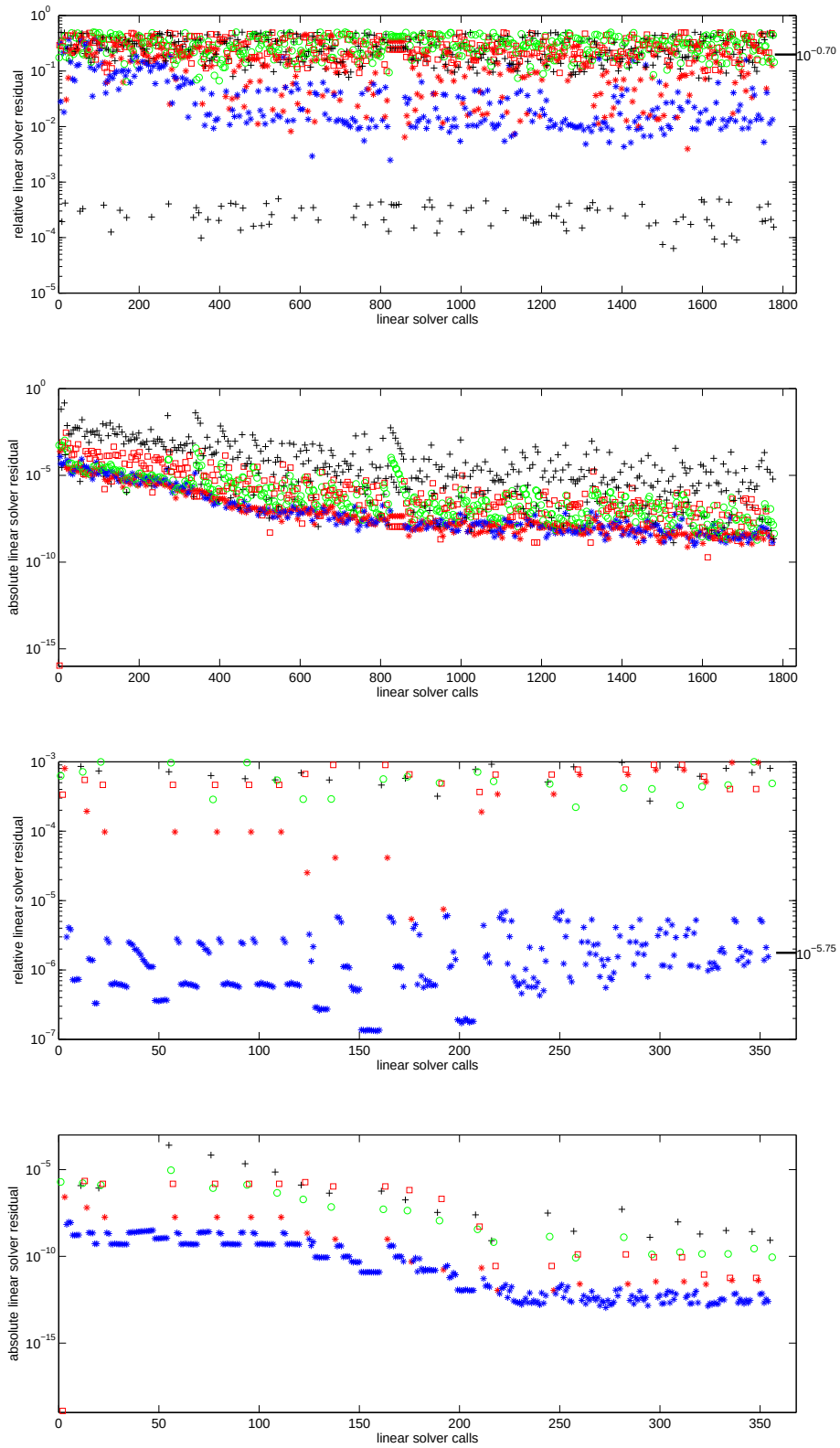

Figure 5.7: Plots of linear solver residuals; parameter identification in the Poisson equation.

5.5.6 Parameter Identification in Linear Elastodynamics

Table 5.14: Inexact SQP algorithm (I-SQP), Algorithm 1, applied to parameter identification in linear elastodynamics. Note: all numbers are totals across the entire SQP run; *prec builds* denotes the number of times a factorization of the approximate Schur preconditioner for augmented systems is computed; *a posteriori refine* denotes the number of times the safeguard $\|\widetilde{t}_k\|_{\mathcal{X}} > \xi_4 \|n_k + t_k\|_{\mathcal{X}}$ is activated in Algorithm 1; *nonconvex QP* denotes the number of times zero or negative curvature is encountered in STCG, Algorithm 3.

Nominal Tol tol^{LS}		**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure		S	S	S	S	S	S	S	S	S
GMRES	iters	259938	92370	140080	83298	53030	58834	64265	69334	79837
	calls	13780	4256	5566	2780	1584	1591	1591	1593	1594
	iters/call	18.9	21.7	25.2	30.0	33.5	37.0	40.4	43.5	50.1
	prec builds	1743	539	277	128	114	114	114	114	114
STCG iters		9720	3021	4860	2494	1323	1332	1332	1334	1335
SQP iters		945	270	235	96	87	87	87	87	87
a posteriori refine		0	0	0	0	0	0	0	0	0
nonconvex QP		4	8	7	7	7	7	7	7	7

Table 5.15: Conventional SQP algorithm (C-SQP) applied to parameter identification in linear elastodynamics. Failure is reported if the number of SQP iterations exceeds 1000.

Nominal Tol tol^{LS}	**0.5**	**10^{-1}**	**10^{-2}**	**10^{-3}**	**10^{-4}**	**10^{-5}**	**10^{-6}**	**10^{-7}**	**10^{-8}**
(S)uccess/(F)ailure	F	F	F	F	F	F	F	F	F
GMRES iters	—	—	—	—	—	—	—	—	—
SQP iters	—	—	—	—	—	—	—	—	—

Figure 5.8 (*next page*):
Relative and absolute residuals for all augmented systems solved by the I-SQP algorithm, applied to parameter identification in linear elastodynamics. First and second pane: $tol^{LS} = 0.5$ (yielding the smallest number of iters/call). Third and fourth pane: $tol^{LS} = 10^{-3}$. Note: the red squares denote quasi-normal step computations with tolerance (3.3); the red asterisks correspond to the first STCG iterations with tolerance (3.6); the blue asterisks denote the subsequent STCG iterations with tolerance (3.8); the black crosses correspond to tangential step computations with tolerance (3.10); the green circles denote Lagrange multiplier computations with tolerance (3.13). Median (log scale) relative residuals are marked by bold lines on the right vertical axes of relative residual plots.

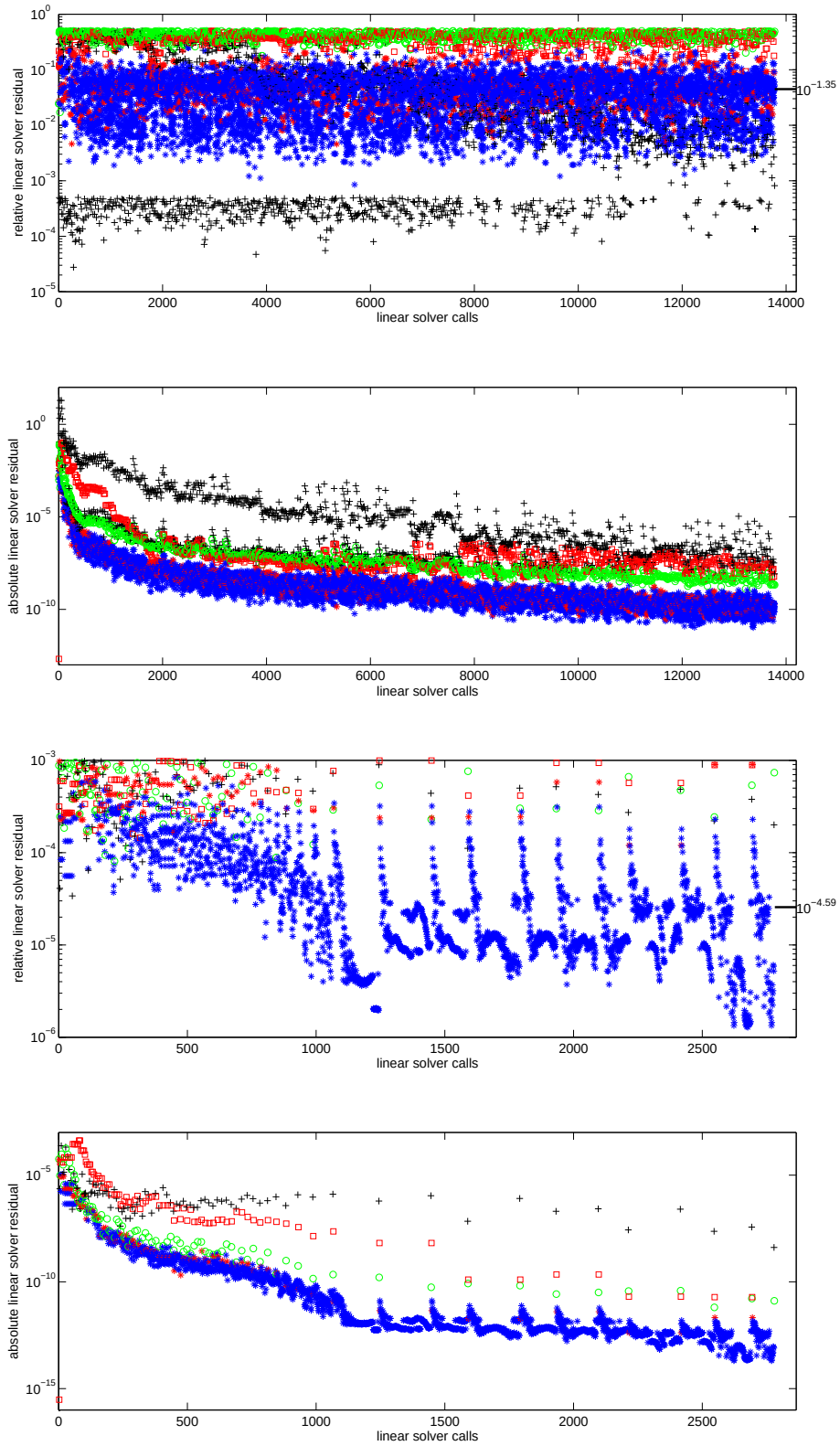

Figure 5.8: Plots of linear solver residuals; parameter identification in linear elastodynamics.

References

[1] R. A. Bartlett, M. Heinkenschloss, D. Ridzal, and B. G. van Bloemen Waanders. Domain decomposition methods for advection dominated linear-quadratic elliptic optimal control problems. *Comput. Methods Appl. Mech. Engrg.*, 195:6428–6447, 2006.

[2] M. Benzi, G. H Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14(1):1–137, 2005.

[3] G. Biros and O. Ghattas. Parallel Lagrange–Newton–Krylov–Schur Methods for PDE-Constrained Optimization. Part I: The Krylov–Schur Solver. *SIAM J. Sci. Comput.*, 27(2):687–713, 2005.

[4] A. Borzì and G. Borzì. An efficient algebraic multigrid method for solving optimality systems. *Computing and Visualization in Science*, 7:183–188, 2004.

[5] A. Borzì and V. Schulz. Multigrid methods for pde optimization. *SIAM Review*, 51(2):361–395, 2009.

[6] M. Burger and R. Pinnau. Fast optimal design of semiconductor devices. *SIAM Journal on Applied Mathematics*, 64(1):108–126, 2003.

[7] M. D. Gunzburger, L. S. Hou, and T. P. Svobotny. Finite element approximations of an optimal control problem associated with the scalar Ginzburg-Landau equation. *Comput. Math. Appl.*, pages 123–131, 1991.

[8] M. Heinkenschloss and H. Nguyen. Neumann–neumann domain decomposition preconditioners for linear-quadratic elliptic optimal control problems. *SIAM Journal on Scientific Computing*, 28(3):1001–1028, 2006.

[9] M. Heinkenschloss and D. Ridzal. An Inexact Trust-Region SQP Method with Applications to PDE-Constrained Optimization. In *Numerical Mathematics and Advanced Applications: Proceedings of ENUMATH 2007*, pages 613–620. Springer Verlag, 2008.

[10] M. Heinkenschloss and D. Ridzal. Integration of Sequential Quadratic Programming and Domain Decomposition Methods for Nonlinear Optimal Control Problems. In U. Langer, M. Discacciati, D. Keyes, O. Widlund, and W. Zulehner, editors, *Domain Decomposition Methods in Science and Engineering XVII*, volume 60 of *Lecture Notes in Computational Science and Engineering*, pages 69–80, Berlin, Heidelberg, New York, 2008. Springer Verlag.

[11] M. Heinkenschloss and D. Ridzal. A matrix-free trust-region SQP method for equality constrained optimization. Technical Report TR11–17, Department of Computational and Applied Mathematics, Rice University, 2011.

[12] L. S. Hou and S. S. Ravindran. Numerical approximation of optimal control problems by a penalty method: error estimates and numerical results. *SIAM Journal of Scientific Computing*, 20:1753–1777, 1999.

[13] E. E. Prudencio, R. Byrd, and X. C. Cai. Parallel Full Space SQP Lagrange–Newton–Krylov–Schwarz Algorithms for PDE–Constrained Optimization Problems. *SIAM Journal on Scientific Computing*, 27(4):1305–1328, 2006.

[14] D. Ridzal. *Trust–Region SQP Methods with Inexact Linear System Solves for Large–Scale Optimization*. PhD thesis, Department of Computational and Applied Mathematics, Rice University, Houston, TX, April 2006.

[15] F. Tröltzsch. *Optimale Steuerung partieller Differentialgleichungen: Theorie,Verfahren und Anwendungen*. Vieweg, 2005.

DISTRIBUTION:

1 MS 0899
RIM-Reports Management (electronic copy), 9532

Sandia National Laboratories