EXTERIOR-POINT ALGORITHMS FOR SOLVING LARGE-SCALE NONLINEAR OPTIMIZATION PROBLEMS

by

Veronica J. Bloom

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computational Sciences and Informatics

Committee:

______________________	Dr. Igor Griva, Dissertation Director
______________________	Dr. Chi Yang, Committee Member
______________________	Dr. Kirk Borne, Committee Member
______________________	Dr. Maria Emelianenko, Committee Member
______________________	Dr. Chi Yang, Interim Director, School of Physics, Astronomy, and Computational Science
______________________	Dr. Donna M. Fox, Associate Dean, Office of Student Affairs & Special Programs, College of Science
______________________	Dr. Peggy Agouris, Dean, College of Science

Date: ______________________ Spring 2014
George Mason University
Fairfax, VA

Exterior-Point Algorithms for Solving Large-Scale Nonlinear Optimization Problems

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Veronica J. Bloom
Master of Arts
Duke University, 2007
Bachelor of Science
Michigan Technological University, 2005

Director: Dr. Igor Griva, Associate Professor
School of Physics, Astronomy, and Computational Science

Spring 2014
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my children, who I hope come to value education and learning as much as their parents do.

Acknowledgments

Most importantly, I would like to thank my advisor Dr. Igor Griva for all of the time he put into helping and guiding me through the entire dissertation process.

I would like to thank my committee members Dr. Chi Yang, Dr. Kirk Borne and Dr. Maria Emelianenko, for all of their comments and helpful suggestions.

I would like to thank my husband Gedare, whose support, encouragement and endurance for chasing around a toddler really made this dissertation possible.

I would like to thank my family. My parents, Thomas and Debra, my sister Valerie, and my brother Zachary for all of their love and support.

I would like to thank the Summer Program for Women in Mathematics at George Washington University, both the organizers and all of the amazing women I have met through the program. Their enthusiasm for learning and mathematics is what encouraged me to pursue this dissertation to begin with.

Finally, I would like to thank my employer Northrop Grumman, for providing education assistance for this program. I would also like to thank all of my managers who were so supportive and willing to work with my flexible schedule which allowed me to pursue this dissertation.

Table of Contents

List of Tables

List of Figures

Abstract

EXTERIOR-POINT ALGORITHMS FOR SOLVING LARGE-SCALE NONLINEAR OPTIMIZATION PROBLEMS

Veronica J. Bloom, PhD

George Mason University, 2014

Dissertation Director: Dr. Igor Griva

Although many efficient solvers exist for solving nonlinear optimization problems, there is none that performs well on all problems or outperforms all methods. Therefore, developing new methods for solving optimization problems continues to be important. Exterior-point methods (EPMs) for solving nonlinear optimization problems have promising convergence properties established by theoretical analysis, but no numerically efficient implementation of exterior-point methods exists.

This dissertation extends development of primal-dual exterior point methods and investigates the application of exterior point methods in several directions. First, a numerically efficient algorithm for solving large scale optimization problems based on exterior point methods is investigated. Second, an implementation of the algorithm is developed in C++ and the algorithm is tested on several hundred nonlinear optimization problems from the CUTEr test set. Third, the use of exterior-point methods for solving the support vector machine (SVM) problem is investigated. Finally, properties of the EPM are utilized to implement active-passive strategies to reduce the size of the primal-dual system when solving optimization problems.

Numerical results indicate the primal-dual EPM solver is competitive with existing solvers, while also being able to solve some problems that existing solvers are unable to solve. Active-passive strategies decrease computation time for the primal-dual EPM solver. The EPM algorithm is also shown to be a feasible training algorithm for SVMs, and acceleration strategies applied to the SVM problem are shown to decrease computation time.

Chapter 1: Introduction

Large-scale optimization problems with thousands or millions of variables and constraints can arise in complicated engineering problems or business operations. Solving such problems can lead to better engineering solutions or increases in profits for businesses. Although a number of numerical methods exist for solving large optimization problems (see e.g. [2, 3]), there is no method that performs well on all problems or outperforms all methods [4, 5]. Therefore, developing new methods for solving optimization problems continues to be important.

In 2008, Griva and Polyak [6] introduced the primal-dual exterior-point method (PD EPM) for problems with both equality and inequality constraints. Compared to related techniques, which are surveyed in chapter 2, the PD EPM has the following properties and associated advantages:

1. It converges for a fixed bounded scaling parameter, so it can avoid the ill-conditioning seen in other methods.

2. Its convergence rate can be sped up by increasing the scaling parameter at each iteration [6] without ill-conditioning.

3. It solves the primal-dual system of equations simultaneously, which is more numerically efficient than methods that minimize at each iteration.

4. It converges at a 1.5-q-superlinear rate.

5. It does not contain complementarity conditions in the formulation, which can cause loss of accuracy in related techniques.

Despite the PD EPM's theoretical advantages, no efficient implementation of the PD EPM exists. Investigating PD EPMs as methods for solving large scale real world optimization problems is important because they may lead to solving problems that existing solvers are unable to solve.

This dissertation extends development of primal-dual exterior-point methods and investigates the application of exterior-point methods in several directions. Contributions of this dissertation include:

- **A numerically efficient PD EPM algorithm**

  A numerically efficient algorithm for solving large scale optimization problems based on exterior-point methods is developed. The design space for the algorithm is explored in detail in chapter 3.

- **Implementation of the PD EPM in C++**

  An implementation of the PD exterior-point algorithm is developed in C++. C++ was chosen for its efficiency—the compiled code is highly optimized for the specific CPU used—and support for object oriented programming, which includes the ability to use abstraction of classes. The algorithm is tested on several hundred nonlinear optimization problems from the CUTEr test set. These tests show the algorithm to be competitive with existing solvers. This is described in detail in chapter 4.

- **Application of the PD EPM to Support Vector Machines**

  The use of exterior-point methods for solving the support vector machine (SVM) problem is investigated and described in detail in chapter 5.

- **Active-Passive strategies applied to the PD EPM**

  The fact that the EPM allows iterates to approach the solution from the exterior of the feasible set is utilized to implement active-passive strategies to reduce the size of the primal-dual system when solving optimization problems. Active-passive strategies are applied to the PD EPM and tested on the CUTEr test set and SVM problems. These strategies are described in detail in sections 4.4 and 5.3 respectively.

The rest of the this dissertation is outlined as follows: Chapter 2 covers background material and work related to exterior-point methods. Chapter 3 describes an exploration of the design space for developing a numerically efficient algorithm. Chapter 4 describes the exterior-point algorithm that was developed as part of this dissertation and presents numerical results. Chapter 5 describes an application of the EPM to the SVM problem, and how properties of the EPM are utilized for solving the SVM problem efficiently and presents numerical results.

Chapter 2: Background and Related Work

This chapter describes optimization problems, the optimality conditions for a solution and primal-dual approaches as they play an important role in developing optimization methods. Then the PD EPM and related approaches are described, including penalty-barrier methods, the NRAL technique, and the PD IPM.

2.1 Optimization Problems

This dissertation focused on developing an exterior-point method algorithm for solving nonlinear optimization problems with both equality and inequality constraints. These problems can be represented as

$$\text{(P1)} \qquad \begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & c_j(x) \geq 0, j = 1, ..., p \\ & g_i(x) = 0, i = 1, ..., q \end{aligned}$$

for $x \in \mathbb{R}^n$, $f, g_i, c_j : \mathbb{R}^n \to \mathbb{R}$, and f, g_i, c_j are all twice continuously differentiable functions.

Problems with equality constraints differ from those with inequality constraints in important ways, which is explained in the following sections. These differences result in different approaches for solving problems with either inequalities or equalities. In the following sections we separately consider problems with just equality constraints and just inequality constraints, describe their optimality conditions and discuss primal-dual methods for solving these problems.

2.1.1 Equality Constrained Problem

A nonlinear optimization problem with only equality constraints can be represented as

$$\text{(P2)} \qquad \begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) = 0, i = 1, ..., q. \end{array}$$

for $x \in \mathbb{R}^n$ and $f, g_i : \mathbb{R}^n \to \mathbb{R}$.

Optimality Conditions

Methods for solving nonlinear optimization problems attempt to find a solution to the problem that satisfies the optimality conditions. Let $g^T(x) = (g_1(x), ..., g_q(x))$ be the vector of equality constraints and $\nabla g(x) = J(g(x))$ be the Jacobian of $g(x)$. Assume that the solution, x^*, to the equality constrained problem (P2) is a *regular point*, which means the gradients of the equality constraints at the solution $\nabla g_i(x_*), i = 1, ..., q$ are linearly independent, where $\nabla g(x)$ is the $q \times n$ Jacobian matrix, or equivalently

$$\text{rank}(\nabla g(x^*)) = q \leq n.$$

Then there exists a vector of dual variables $\lambda_* \in \mathbb{R}^q$ and primal variables $x^* \in \mathbb{R}^n$ such that the first order optimality conditions are satisfied. Lagrange suggested the first order optimality condition for a problem of the form (P2) as a solution to the system of equations

$$\nabla_x L(x^*, \lambda^*) = \nabla f(x^*) - \sum_{i=1}^{q} \lambda_i^* \nabla g_i(x^*) = 0$$

$$g_i(x^*) = 0, i = 1, ..., q.$$

Lagrange's condition is based on the Lagrangian function $L(x, \lambda) = f(x) - \sum_{i=1}^{q} \lambda_i g_i(x)$, which introduces a vector $\lambda = (\lambda_1, ..., \lambda_q)$ of dual variables for solving constrained optimization problems.

The solution x^* being a regular point together with the second order sufficient condition

$$\xi^T \nabla_{xx}^2 L(x^*, \lambda^*) \xi > 0 \text{ for all } \xi \in \mathbb{R}^n \text{ such that } \nabla g(x_*) \xi = 0$$

consitute the standard second order optimality conditions for the problem (P2). The second order sufficient condition together with the first order optimality conditions are the sufficient conditions for a solution to the equality constrained problem. If a point x^* satisfies the sufficient conditions, then x^* is a strict local minimum of $f(x)$ subject to the equality constraints. The second order optimality conditions are important because they guarantee efficient convergence of the PD EPM.

Primal-Dual Newton's Method

For the equality constrained problem, Newton's method can be used as a primal-dual approach for finding a solution that satisfies the first order optimality conditions. Using Newton's method, if x and λ are current iterates for the primal and dual variables, then the next iterate is

$$\hat{x} = x + \Delta x$$

$$\hat{\lambda} = \lambda + \Delta\lambda$$

where Δx and $\Delta\lambda$ are the solution to the system of equations

$$\begin{bmatrix} \nabla^2_{xx} L(x,\lambda) & -\nabla g(x)^T \\ \nabla g(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda) \\ -g(x) \end{bmatrix}$$

and $\nabla g(x)$ is the Jacobian of $g(x) = (g_1(x), ..., g_q(x))$. Under the second order optimality conditions, Newton's method will converge quadratically within a neighborhood of the solution. However, Newton's method does not necessarily converge globally, so if Newton's method is applied, a globalization strategy is required.

2.1.2 Inequality Constrained Problem

A nonlinear optimization problem with only inequality constraints can be represented as

$$\text{(P3)} \qquad \begin{array}{rl} \text{minimize} & f(x) \\ \text{subject to} & c_i(x) \geq 0, i = 1, ..., p \end{array}$$

for $x \in \mathbb{R}^n$ and $f, c_i : \mathbb{R}^n \to \mathbb{R}$.

Optimality Conditions

As with the equality constrained problem, methods for solving the inequality constrained problem attempt to find a solution, x^*, that satisfies the optimality conditions. Let $c_{(r)}(x)$ represent the active set of inequality constraints, i.e. $c_{(r)}^T(x) = (c_1(x), ..., c_r(x))$ where at the solution $c_i(x^*) = 0, i = 1, ..., r$. Let $\nabla c_{(r)}(x) = J(c_{(r)})$ be the Jacobian of $c_{(r)}(x)$ of dimension $r \times n$. Again we assume that the solution, x^*, to the inequality constrained problem (P3) is a regular point. For the inequality constrained problem, x^* being a regular point means that the gradients of the active constraints at the solution, $\nabla c_i(x^*), i = 1, ..., r$, are linearly independent, or equivalently

$$\operatorname{rank}(\nabla c_{(r)}(x^*)) = r \leq n.$$

Then there exists a vector $\lambda^* \in \mathbb{R}^p$ and $x^* \in \mathbb{R}^n$ such that the first order optimality conditions are satisfied.

The first order optimality conditions for inequality constrained problems—the Karush-Kuhn-Tucker (KKT) conditions [7,8]—state that a solution to (P3) must satisfy the following system for primal variables $x^* \in \mathbb{R}^n$ and dual variables $\lambda^* \in \mathbb{R}^p$

$$\nabla_x L(x^*, \lambda^*) = \nabla f(x^*) - \sum_{i=1}^{p} \lambda_i^* \nabla c_i(x^*) = 0 \qquad (2.1)$$

$$\lambda_i^* c_i(x^*) = 0, i = 1, ..., p$$

$$\lambda_i^* \geq 0, i = 1, ..., p$$

$$c_i(x^*) \geq 0, i = 1, ..., p.$$

In addition to equation (2.1), the KKT conditions include: primal feasibility $c_i(x^*) \geq 0, i = 1, ..., p$; dual feasibility $\lambda_i^* \geq 0, i = 1, ..., p$; and complementarity conditions $\lambda_i^* c_i(x^*) = 0, i = 1, ..., p$.

The standard second order optimality conditions for the problem (P3) include: the solution x^* being a regular point, the second order sufficient condition

$$\xi^T \nabla_{xx}^2 L(x^*, \lambda^*)\xi > 0 \text{ for all } \xi \in \mathbb{R}^n \text{ such that } \nabla c_{(r)}(x_*)\xi = 0,$$

and *strict complementarity.* Strict complementarity means the multipliers corresponding to the active constraints must be positive $\lambda_i^* > 0, i = 1, ..., r$. The second order sufficient condition together with the first order optimality conditions are the sufficient conditions for a solution to the inequality constrained problem. If a point x^* satisfies the sufficient conditions, then x^* is a strict local minimum of $f(x)$ subject to the inequality constraints. Again, the second order optimality conditions are important because they guarantee efficient convergence of the PD EPM.

Primal-Dual Newton's Method

This section shows that applying Newton's method to the first order optimality conditions for the problem with inequality constraints may lead to difficulties that do not occur for problems with equality constraints. Consider using Newton's method to find a solution that satisfies the KKT conditions. If x and λ are current iterates for the primal and dual variables, then the next iterate is

$$\hat{x} = x + \Delta x$$

$$\hat{\lambda} = \lambda + \Delta\lambda$$

where Δx and $\Delta\lambda$ are the solution to the system of equations

$$\begin{bmatrix} \nabla^2_{xx} L(x,\lambda) & -\nabla c(x)^T \\ \Lambda \nabla c(x) & C(x) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda) \\ -C(x)\Lambda e \end{bmatrix}$$

where $C(x) = \text{diag}(c_1(x), ..., c_p(x))$, $\Lambda = \text{diag}(\lambda_1, ..., \lambda_p)$ and $e = (1, ..., 1) \in \mathbb{R}^p$. Solving for $\Delta\lambda$ and Δx gives

$$\begin{aligned} \Delta\lambda =& - \Lambda e - C^{-1}(x)\Lambda\nabla c(x)\Delta x \\ \Delta x =& (\nabla^2_{xx} L(x,\lambda) + \nabla c(x)^T C^{-1}(x)\Lambda\nabla c(x))^{-1} \\ & (-\nabla_x L(x,\lambda) - \nabla c(x)^T \Lambda e). \end{aligned}$$

Note that the elements in the matrix $C(x)$, corresponding to the active constraints $\{i : c_i(x^*) = 0\}$, go to zero near the solution. For these elements, $C^{-1}(x)$ becomes large, which can cause a loss of accuracy due to finite precision arithmitic. Also, in order to satisfy primal and dual feasibility, we need to ensure that the initial values are feasible, $\lambda_0 \geq 0$ and $c_i(x_0) \geq 0, i = 1, ..., p$, which is a challenging problem in itself. Feasibility can be maintained by ensuring that $\lambda \geq 0$ and $c_i(x) \geq 0, i = 1, ..., p$ at each iteration, but this may decrease the convergence rate.

If the active constraints are known ahead of time, then Newton's method can be used to solve the system of nonlinear equations. However, this set is typically unknown, and finding it is a combinatorial problem. Researchers have developed methods, which are discussed in the following sections, that find a solution that satisfies the optimality conditions while avoiding a combinatorial search.

In conclusion, because the active constraints are unknown, developing methods for solving nonlinear optimization problems with inequality constraints is more challenging than those with equality constraints. The next two sections discuss methods for solving problems with equality constraints, inequality constraints, and both. The first section describes sequential unconstrained minimization techniques (SUMTs) and the second section describes primal-dual methods that can be derived from SUMT techniques.

2.2 Sequential Unconstrained Minimization Techniques (SUMTs)

Sequential unconstrained minimization techniques (SUMTs) solve a constrained minimization problem by deriving a sequence of solutions to smooth unconstrained minimization problems. This sequence approaches the solution to the constrained problem.

Both penalty and barrier methods, which is discussed in section 2.2.1, and the NRAL technique, which is discussed in section 2.2.2, are SUMTs. For penalty and barrier methods, multipliers are not used explicitly, but they appear as a by-product as is shown in section 2.3.1. The NRAL technique, and its constituent techniques the augmented Lagrangian and nonlinear rescaling techniques, introduce multipliers, which are updated at each iteration and drive the convergence of the methods.

Primal-dual methods, which are discussed in section 2.3, can be derived from penalty and barrier methods and the NRAL technique. Because SUMTs solve an unconstrained minimization problem at each iteration, they may not be computationally efficient. Primal-dual methods improve performance by removing the minimization at each iteration.

2.2.1 Penalty and Barrier Methods

In this section, penalty and barrier methods are described and their relationship to the NRAL technique and the PD IPM is explained.

Penalty Method

In 1943, Courant [9] introduced penalty methods for solving equality constrained problems. Penalty methods solve a sequence of unconstrained minimization problems where a penalty is incurred for violating the constraints. A penalty function takes the form

$$\pi_\rho(x) = f(x) + \rho\psi(x)$$

where $\psi(x) = 0$ if x satisfies the constraints and $\psi(x) > 0$ otherwise. One example is

$$\psi(x) = \frac{1}{2} g(x)^T g(x).$$

A penalty method solves the unconstrained minimization problem

$$\text{minimize } \pi_{\rho_k}(x)$$

for an increasing sequence ρ_k. As $\rho_k \to \infty$, the largest eigenvalue of the Hessian of the penalty function tends to infinity and becomes ill-conditioned, which makes the minimization problem more difficult to solve as the solution is approached.

Barrier Method

Barrier functions take the form

$$\beta_\mu(x) = f(x) + \mu\phi(x)$$

where $\phi(x) \to \infty$ as $c_i(x) \to 0_+$. The logarithmic barrier function, introduced by Frisch [10] , has $\phi(x) = -\sum_{i=1}^{p} \log(c_i(x))$ and the inverse barrier function, introduced by Carroll [11], has $\phi(x) = \sum_{i=1}^{p} \frac{1}{c_i(x)}$. Barrier methods use barrier functions to find a solution to the inequality constrained problem (P3) by solving a sequence of unconstrained minimization problems

$$\text{minimize } \beta_{\mu_k}(x)$$

for a decreasing barrier parameter μ_k. As $\mu_k \to 0$, the Hessian of the barrier function tends to infinity and becomes ill-conditioned, which makes the minimization problem more difficult to solve as the solution is approached.

Fiacco and McCormick [12] and Wright [13] survey penalty and barrier methods. Section 2.3.1 describes a PD IPM that can be derived from the barrier method. PD IPMs have an advantage over classical barrier methods since they do not suffer from ill-conditioning as the scaling parameter μ goes to 0. However, PD IPMs have their own disadvantages which are shown in section 2.3.1.

The next section describes the NRAL technique, which is a SUMT method that will converge for a fixed bounded scaling parameter. This convergence for a bounded scaling parameter occurs due to explicitly introducing dual variables, or multipliers, into the formulation.

2.2.2 Nonlinear Rescaling Augmented Lagrangian Technique

The NRAL technique [6] is a multiplier-based method that combines the augmented Lagrangian method for equality constraints [14, 15] with the nonlinear rescaling method for inequality constraints [16, 17]. Multiplier-based methods avoid the ill-conditioning seen in classical penalty and barrier methods by exploiting dual variables, or multipliers, as a driving force for the convergence of the methods to the solution.

Augmented Lagrangian Method

An augmented Lagrangian method [14, 15] iteratively minimizes an augmented Lagrangian function with respect to its primal variables and then updates its dual variables. Iteration continues until the first order optimality condition is satisfied. An example of an augmented Lagrangian function is

$$A_k(x, \lambda) = f(x) - \lambda^T g(x) + \frac{1}{2} k g(x)^T g(x)$$

where A_k is the Lagrangian with a penalty term. An augmented Lagrangian method iterates

$$x_{s+1} = \arg\min_x A_{k_s}(x, \lambda_s)$$

$$\lambda_{s+1} = \lambda_s - \rho_s g(x_s)$$

over s until $\nabla L(x_s, \lambda_s) = 0$. Bertsekas [18] describes augmented Lagrangian methods in his book. Conn, Gould and Toint [19] address computational issues related to augmented Lagrangian techniques using an early version of the LANCELOT solver.

Augmented Lagrangian techniques avoid ill-conditioning, but they perform an unconstrained minimization at each step, which can be computationally expensive. Primal-dual methods, described in section 2.3, remove the minimization at each step and update simultaneously both primal and dual variables resulting in substantially more efficient methods.

Modified Barrier Method

For solving inequality constrained optimization problems, Polyak introduced modified barrier methods [17]. Similar to augmented Lagrangian methods, modified barrier methods explicitly use dual variables to avoid the ill-conditioning seen in classical barrier methods. The modified barrier method iteratively minimizes a modified barrier function with respect to its primal variables then updates its dual variables. Iteration continues until the solution is found. An example of a modified barrier function is the logarithmic modified barrier function

$$S_k(x, \lambda) = f(x) - k^{-1} \sum_{i=1}^{p} \lambda_i \log(kc_i(x) + 1).$$

The modified barrier method iterates

$$x_{s+1} = \arg\min_x S_{k_s}(x, \lambda_s)$$

$$(\lambda_{s+1})_i = \frac{(\lambda_s)_i}{k_s c_i(x_{s+1}) + 1}$$

over s until $\nabla L(x_s, \lambda_s) = 0$ and $\lambda_i c_i(x) = 0$, $i = 1, ..., p$. This method will converge for a fixed $\mu_s > 0$, so it avoids the ill-conditioning of the Hessian seen in classical barrier methods.

One problem with the modified barrier method is that the modified barrier function is not defined for all real numbers, which can lead to numerical difficulties. In the next sections, we see that the modified barrier function can be extended to a more general transformation defined for all real numbers, which alleviates this problem.

Nonlinear Rescaling Method

Polyak and Teboulle [16] introduced the nonlinear rescaling method as a genearalization of the modified barrier method. The nonlinear rescaling method transforms the constraints $c_i(x)$ to an equivalent set using a class of functions ψ with the following properties [16]:

- $\psi(0) = 0$
- $\psi'(t) > 0$
- $\psi'(0) = 1$
- $\psi''(t) < 0$
- (a) $\psi'(t) \leq a(t+1)^{-1}$, (b) $-\psi''(t) \leq b(t+1)^{-2}$, $t \geq 0$, $a > 0$, $b > 0$.

For $k > 0$, the equivalent problem to (P3) is

$$\text{minimize } f(x)$$

$$\text{subject to } k^{-1}\psi(kc_i(x)) \geq 0, i = 1, ..., p$$

and the Lagrangian of the equivalent problem is

$$\mathcal{L}_k(x,\lambda) = f(x) - k^{-1}\sum_{i=1}^{p}\lambda_i\psi(kc_i(x)).$$

Similar to the modified barrier method, the nonlinear rescaling method can solve (P3) by minimizing the Lagrangian for the equivalent problem for fixed $k > 0$ and updating the multipliers λ. So the nonlinear rescaling method iterates

$$x_{s+1} = \arg\min_x \mathcal{L}_k(x,\lambda_s)$$

$$(\lambda_{s+1})_i = \psi'(kc_i(x_{s+1}))(\lambda_s)_i, i = 1, ..., p$$

over s until the solution is found. The nonlinear rescaling method will converge without increasing the scaling parameter k, so it avoids the ill-conditioning seen in classical barrier methods.

NRAL Technique

For problems with both equality and inequality constraints, Griva and Polyak [6] introduced the nonlinear rescaling-augmented Lagrangian (NRAL) technique. The NRAL technique combines the nonlinear rescaling method for inequality constraints [16, 17] with the augmented Lagrangian method for equality constraints [14, 15].

The NRAL technique can find a solution to the problem (P1)

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } c_i(x) \geq 0, i = 1, ..., p \\ &\qquad\qquad g_j(x) = 0, j = 1, ..., q \end{aligned}$$

with $x \in \mathbb{R}^n$, and twice continuously differentiable functions $f, c_i, g_j : \mathbb{R}^n \to \mathbb{R}$.

We assume that the solution, x^*, to this problem (P1) is a regular point. Let $c_{(r)}(x)$ represent the active set of inequality constraints, i.e. $c_{(r)}^T(x) = (c_1(x), ..., c_r(x))$ where at the solution $c_i(x^*) = 0, i = 1, ..., r$. Let $\nabla c_{(r)}(x) = J(c_{(r)})$ be the Jacobian of $c_{(r)}(x)$. Let $g^T(x) = (g_1(x), ..., g_q(x))$ be the vector of equality constraints and $\nabla g(x) = J(g(x))$ be the Jacobian of $g(x)$. For the problem with both equality and inequality constraints, x^* being a regular point means that the gradients of the active constraints at the solution, $\nabla c_i(x^*), i = 1, ..., r$, and the gradients of the equality constraints at the solution, $\nabla g_i(x_*), i = 1, ..., q$, are linearly independent, or equivalently

$$\text{rank} \begin{pmatrix} \nabla c_{(r)}(x^*) \\ \nabla g(x^*) \end{pmatrix} = q + r \leq n.$$

Then there exists vectors $\lambda^* \in \mathbb{R}^p$, $\nu^* \in \mathbb{R}^q$ and $x^* \in \mathbb{R}^n$ such that the first order optimality conditions, or KKT conditions, are satisfied:

$$\nabla_x L(x^*, \lambda^*, \nu^*) = \nabla f(x^*) - \sum_{i=1}^{p} \lambda_i^* \nabla c_i(x^*) - \sum_{j=1}^{q} \nu_j^* \nabla g_j(x^*) = 0$$

$$\lambda_i^* c_i(x^*) = 0, i = 1, ..., p$$

$$\lambda_i^* \geq 0, i = 1, ..., p$$

$$c_i(x^*) \geq 0, i = 1, ..., p$$

$$g_i(x^*) = 0, i = 1, ..., q.$$

The standard second order optimality conditions for the problem (P1) include: the solution x^* being a regular point, the second order sufficient condition

$$\xi^T \nabla^2_{xx} L(x^*, \lambda^*, \nu^*)\xi > 0 \text{ for all } \xi \in \mathbb{R}^n : \nabla c_{(r)}(x_*)\xi = 0 \text{ and } \nabla g(x_*)\xi = 0$$

and *strict complementarity.* Strict complementarity means the multipliers corresponding to the active constraints must be positive $\lambda_i^* > 0, i = 1, ..., r$.

The NRAL technique solves the problem (P1) by constructing an equivalent problem in which the constraints are rescaled by a function
$\psi : -\infty \leq t_0 < t < t_1 \leq +\infty$ such that $\psi(t)$ has the following properties:

- $\psi(0) = 0$
- $\psi'(t) > 0$
- $\psi'(0) = 1$
- $\psi''(t) < 0$
- (a) $\psi'(t) \leq a(t+1)^{-1}$, (b) $-\psi''(t) \leq b(t+1)^{-2}$, $t \geq 0$, $a > 0$, $b > 0$.

For $k > 0$, the following problem is equivalent to the original problem:

$$\text{minimize } f(x)$$

$$\text{subject to } k^{-1}\psi(kc_i(x)) \geq 0, j = 1, ..., p$$

$$g_j(x) = 0, i = 1, ..., q.$$

For this work, the transformation ψ is chosen as

$$\psi(t) = \begin{cases} \log(t+1) & \text{if } t > -.5 \\ -2t^2 + \log(.5) + .5 & \text{if } t \leq -.5. \end{cases}$$

Griva and Polyak [20] actively used this transformation for solving nonlinear optimization problems using the nonlinear rescaling method.

The augmented Lagrangian for this equivalent problem is

$$\mathcal{L}_k(x, \lambda, \nu) = f(x) - k^{-1} \sum_{i=1}^{p} \lambda_i \psi(kc_i(x)) - \sum_{j=1}^{q} \nu_j g_j(x) + \frac{k}{2} \sum_{j=1}^{q} g_j(x)^2$$

where $\lambda \in \mathbb{R}^p$ and $\nu \in \mathbb{R}^q$ are dual variables.

As a multiplier-based method, the NRAL technique minimizes the augmented Lagrangian for the equivalent problem with respect to the primal variables, and then updates the dual variables, or multipliers. If (x^s, λ^s, ν^s) is a current iterate of the NRAL technique then the next iterate $(x^{s+1}, \lambda^{s+1}, \nu^{s+1})$ is found by solving the unconstrained minimization problem

$$x^{s+1} = \arg \min_{x \in \mathbb{R}^n} \mathcal{L}_k(x, \lambda^s, \nu^s)$$

and updating the multipliers

$$\lambda_i^{s+1} = \lambda_i^s \psi'(kc_i(x^{s+1})), i = 1, ..., p$$

$$\nu_j^{s+1} = \nu_j^s - kg_j(x^{s+1}), j = 1, ..., q.$$

At each iteration of the NRAL technique, any unconstrained minimization routine can be used to find $\arg \min_{x \in \mathbb{R}^n} \mathcal{L}_k(x, \lambda^s, \nu^s)$. Unconstrained minimization methods investigated for the NRAL technique are described in section 3.3.

If (x_*, λ_*, ν_*) is the solution, then for k large enough, under second order optimality conditions, the iterates (x^s, λ^s, ν^s) derived using the NRAL technique will converge to the solution linearly.

Although the NRAL technique can avoid the ill-conditioning seen with penalty and barrier methods, since it converges for a fixed bounded scaling parameter, it still requires solving a minimization problem at each iteration, which can be computationally expensive. Also, increasing the scaling parameter at each iteration can help convergence, but ill-conditioning may occur if the scaling parameter becomes too large.

Next, primal-dual methods are discussed, which update both primal and dual variables simultaneously and may result in more efficient methods. Also, they do not suffer from the ill-conditioning seen in penalty and barrier or multiplier based methods.

2.3 Primal-Dual Methods

Primal-dual methods improve the computational efficiency of sequential unconstrained minimization techniques (SUMT) by applying one step of Newton's method to solve the system of equations which arise, rather than minimizing at each iteration.

2.3.1 Primal-Dual Interior-Point Method (PD IPM)

A primal-dual interior-point method (PD IPM) solves the primal-dual system of equations which arises in the logrithmic barrier method using Newton's method. Each iteration of the logarithmic barrier method solves

$$\text{minimize } \beta_{\mu_k}(x) = f(x) - \mu_k \sum_{i=1}^{p} \log(c_i(x))$$

which is equivalent to finding x as the solution to

$$\nabla f(x) - \mu_k \sum_{i=1}^{p} \frac{\nabla c_i(x)}{c_i(x)} = 0.$$

If we define λ as $\lambda_i = \dfrac{\mu_k}{c_i(x)}$, then λ and x result from solving the system

$$\nabla f(x) - \sum_{i=1}^{p} \lambda_i \nabla c_i(x) = 0$$

$$\lambda_i c_i(x) = \mu_k, i = 1, ..., p.$$

Consider applying a single step of Newton's method to solve the system of equations. If x and λ are current iterates, then the Newton direction $(\Delta x, \Delta\lambda)$ can be found by solving the system of equations

$$\begin{bmatrix} \nabla^2_{xx} L(x,\lambda) & -\nabla c(x)^T \\ \Lambda \nabla c(x) & C(x) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda) \\ -C(x)\Lambda e + \mu_k e \end{bmatrix}$$

where $C(x) = \text{diag}(c_1(x), ..., c_p(x))$, $\Lambda = \text{diag}(\lambda_1, ..., \lambda_p)$ and $e = (1, ..., 1) \in \mathbb{R}^p$.

Then the primal and dual variables can be updated as

$$\hat{x} = x + \Delta x \quad \hat{\lambda} = \lambda + \Delta\lambda.$$

Solving for Δx and $\Delta\lambda$, we obtain

$$\Delta\lambda = -\Lambda e + \mu_k C^{-1}(x)e - C^{-1}(x)\Lambda\nabla c(x)\Delta x$$

$$(\nabla^2_{xx} L(x,\lambda) + \nabla c(x)^T C^{-1}(x)\Lambda\nabla c(x))\Delta x =$$

$$(-\nabla_x L(x,\lambda) - \nabla c(x)^T \Lambda e + \mu_k \nabla c(x)^T C^{-1}(x)e).$$

The complementarity condition in the method, $\lambda_i c_i(x) = \mu_k, i = 1, ..., p$, causes the matrix $C(x)^{-1}$ to appear. The elements in the matrix $C(x)$ corresponding to the active constraints, $\{i : c_i(x^*) = 0\}$, go to zero near the solution causing $C^{-1}(x)$ to become large. So the complementarity condition can cause a loss of accuracy due to finite precision arithmitic.

Practical interior-point algorithms for nonlinear programming are addressed by Vanderbei [21], Vanderbei and Shanno [22] and Byrd, Hribar, and Nocedal [23].

The primal-dual exterior-point method—presented in the next section—has no complementarity condition in the formulation, so the method can potentially obtain more accuracy than the PD IPM.

2.3.2 Primal-Dual Exterior-Point Method (PD EPM)

In 2008, Griva and Polyak [6] introduced the primal-dual exterior-point method (PD EPM) for problems with both equality and inequality constraints. The PD EPM solves the primal-dual system of equations which arises in the NRAL technique using Newton's method, rather than solving a minimization problem followed by an update of the multipliers.

In the NRAL technique, minimizing the augmented Lagrangian of the equivalent problem with respect to the primal variables is equivalent to finding $\hat{x}$ such that

$$\nabla f(\hat{x}) - \sum_{i=1}^{p} \lambda_i \psi'(kc_i(\hat{x}))\nabla c_i(\hat{x}) - \sum_{j=1}^{q}(\nu_j - kg_j(\hat{x}))\nabla g_j(\hat{x}) = 0.$$

A single iteration of the NRAL method is equivalent to finding $\hat{x}, \hat{\lambda}, \hat{\nu}$ such that

$$\nabla f(\hat{x}) - \sum_{i=1}^{p} \lambda_i \psi'(kc_i(\hat{x}))\nabla c_i(\hat{x}) - \sum_{j=1}^{q}(\nu_j - kg_j(\hat{x}))\nabla g_j(\hat{x}) = 0$$

$$\hat{\lambda} - \Psi'(kc(\hat{x}))\lambda = 0$$

$$\hat{\nu} - \nu - kg(\hat{x}) = 0$$

where $\Psi'(kc(\hat{x})) = \text{diag}(\psi'(kc_i(\hat{x})))$ for $i = 1, ..., p$.

Replacing $\psi'(kc_i(\hat{x}))\lambda_i$ with $\hat{\lambda}_i$ and $\nu_j - kg_j(\hat{x})$ with $\hat{\nu}_j$ yields

$$\nabla f(\hat{x}) - \sum_{i=1}^{p} \hat{\lambda}_i \nabla c_i(\hat{x}) - \sum_{j=1}^{q} \hat{\nu}_j \nabla g_j(\hat{x}) = 0$$

$$\hat{\lambda} - \Psi'(kc(\hat{x}))\lambda = 0$$

$$\hat{\nu} - \nu - kg(\hat{x}) = 0.$$

Newton's method can be used to solve for $(\hat{x}, \hat{\lambda}, \hat{\nu})$. For one step of Newton's method, if (x, λ, ν) is a current iterate, then the Newton direction $(\Delta x, \Delta\lambda, \Delta\nu)$ is found by solving the system of equations

$$\begin{bmatrix} \nabla^2_{xx} L(x, \lambda, \nu) & -\nabla c^T(x) & -\nabla g^T(x) \\ -k\Lambda\Psi''(kc(x))\nabla c(x) & I_p & 0 \\ k\nabla g(x) & 0 & I_q \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x, \lambda, \nu) \\ \bar{\lambda} - \lambda \\ -kg(x) \end{bmatrix}$$

where $\Psi''(kc(x)) = \mathrm{diag}(\psi''(kc_i(x)))_{i=1}^p$, $\Lambda = \mathrm{diag}(\lambda_i)_{i=1}^p$, $\bar{\lambda} = \Psi'(kc(x))\lambda$ and I_p and I_q are $p \times p$ and $q \times q$ identity matrices.

Griva and Polyak [6] use a merit function to define the scaling parameter k so that, under second order optimality conditions, the PD EPM has a 1.5-q-superlinear rate of convergence. The merit function v measures the distance between the current approximation and the solution as

$$v(x, \lambda, \nu) = \max\{\|\nabla_x L(x, \lambda, \nu)\|, -\min_{1 \le i \le p} c_i(x), \max_{1 \le i \le q} |g_i(x)|,$$

$$\sum_{i=1}^{p} |\lambda_i||c_i(x)|, -\min_{1 \le i \le p} \lambda_i\}.$$

Then the scaling parameter k is

$$k = v(x, \lambda, \nu)^{-.5}.$$

The PD EPM consists of the following steps:

1. Calculate the scaling parameter $k = v(x, \lambda, \nu)^{-.5}$.

2. Compute the primal-dual Newton direction $(\Delta x, \Delta\lambda, \Delta\nu)$.

3. Update the primal-dual variables

$$\hat{x} = x + \Delta x \quad \hat{\lambda} = \lambda + \Delta\lambda \quad \hat{\nu} = \nu + \Delta\nu.$$

If (x_*, λ_*, ν_*) is the solution, then, under second order optimality conditions, the iterates $(\hat{x}, \hat{\lambda}, \hat{\nu})$ derived using the exterior-point method will converge to the solution at a 1.5-q-superlinear rate within a neighborhood of the solution. Thus, for $z = (x, \lambda, \nu)$, we have

$$||\hat{z} - z^*|| \le C||z - z^*||^{1.5}.$$

Despite the PD EPM's theoretical advantages over related techniques no efficient implementation of the PD EPM exists. The goal of this research is to develop a general purpose nonlinear optimization algorithm and numerically efficient solver based on exterior-point methods.

2.4 Nonlinear Optimization Solvers

Efficient solvers for nonlinear programming problems exist that implement interior-point methods and other techniques such as sequential quadratic programming methods and reduced gradient methods. The solvers listed below are designed for solving nonlinear programming problems and are available on the NEOS optimization server (www.neos-server.org).

- IPOPT [24, 24, 25]: Implements a PD IPM
- LANCELOT [19, 26, 27]: Implements a sequential augmented Lagrangian algorithm as part of GALAHAD
- LOQO [21, 22]: Implements a PD IPM
- MINOS [28]: Implements a reduced gradient algorithm for linear constraints and a projected augmented Lagrangian algorithm for nonlinear constraints

- SNOPT [29–32]: Implements a sequential quadratic programming algorithm
- CONOPT [33]: Implements a generalized reduced gradient algorithm
- DONLP2 [34]: Implements a sequential equality constrained quadratic programming method with an active set technique
- FSQP [35–41]: Implements a feasible sequential quadratic programming algorithm
- KNITRO [42]: Implements both an interior-point method and an active set method
- NPSOL [43, 44]: Implements a sequential quadratic programming method

Moré and Wright [45] give an overview of a subset of these solvers in their book. Bondarenko, Bortz and Moré [4] test five of these solvers—DONLP2, LANCELOT, MINOS, SNOPT, and LOQO—on problems from the COPS test set, a collection of large-scale nonlinearly constrained optimization problems. None of the solvers performs well on all problems and no solver consistently outperforms the others.

Although efficient solvers exist that implement interior-point methods and other techniques, no efficient solver implements exterior-point methods. Under this project, a numerically efficient algorithm for solving large scale optimization problems based on exterior point methods was investigated and an implementation of the algorithm in C++ was developed. This solver is tested on the CUTEr test set with numerical results reported in section 4.3. The rest of this dissertation outlines the development of the exterior-point method algorithm, and discusses the application of the exterior-point method algorithm to the support vector machine problem.

Chapter 3: Exterior-Point Algorithm Design Space Exploration

The exterior-point algorithm relies on methods which require efficient implementation. These methods include solvers for linear systems of equations, methods for matrix regularization, and methods for performing unconstrained minimization. The choice of implementation for these methods affects the efficiency of the exterior-point algorithm. This chapter explores the options for these methods as an essential step to building an efficient algorithm.

Section 3.1 discusses linear systems of equations, which arise in the PD EPM and in the NRAL technique if Newton's method is used for minimization. Section 3.2 provides details on matrix regularization, which is necessary for ensuring that the matrix that arises in the PD EPM is quasi-definite and that the matrix that arises in Newton's method for minimization is positive definite. Finally, section 3.3 presents unconstrained optimization methods investigated for solving the minimization problem that arises in the NRAL technique.

3.1 Selecting Open Source Linear Solvers for the EPM

Linear systems of equations arise in the PD EPM and in the NRAL technique if Newton's method is used for minimization. Efficient open source solvers are investigated for solving the linear system of equations that arise. The choice of solvers and methods to be used depends on the structure of the problem, which is discussed in the next sections.

Determining how to solve the linear system of equations that arise in the NRAL and PD EPM depends on the conditioning, structure, sparsity, and size of the system. The conditioning of the system will determine whether a direct or iterative method should be used, since iterative methods work better on well conditioned systems. The structure of the system—whether it is symmetric or not, and whether it is positive definite, quasi-definite, or neither—will determine which direct or iterative method to use. The sparsity of the system will determine whether to use a sparse or dense implementation of these methods. The next sections give an overview of direct and iterative methods for solving linear systems of equations, describe sparse and dense matrix representations and methods for solving linear systems, and discusses reducing the size of the PD EPM system.

3.1.1 Direct and Iterative Methods for Dense Systems

The conditioning and structure of a system will determine whether a direct or iterative method should be used and which direct or iterative method to use. Direct methods, such as Gaussian elimination and Cholesky factorization, solve a system directly in a single step. Iterative methods, such as the conjugate gradient method, solve a system in many steps by starting with an initial guess for the solution and iteratively improving the solution.

Both Cholesky factorization and the conjugate gradient method solve symmetric positive definite systems. Gaussian elimination and variations of the conjugate gradient method, such as QMR, GMRES and BiCgStab [46], can solve non-symmetric systems. Also, certain indefinite systems can be factored as LDL^T and solved using a Cholesky-like factorization [47].

For the NRAL technique, the Hessian of the Lagrangian should be positive definite, or regularized so that it is positive definite, so Cholesky factorization or conjugate gradient can be used. Section 3.2 shows how matrix regularization can be done as part of Cholesky factorization.

Symmetric Quasi-Definite System

For the PD EPM, we can write the system as a symmetric quasi-definite matrix and use a Cholesky-like factorization, LDL^T [47]. A symmetric quasi-definite system is a matrix of the form

$$K = \begin{bmatrix} E & A^T \\ A & -F \end{bmatrix}$$

where E and F are symmetric positive definite matrices.

The system that arises in the PD EPM is

$$\begin{bmatrix} \nabla^2_{xx} L(x,\lambda,\nu) & -\nabla c^T(x) & -\nabla g^T(x) \\ -k\Lambda\Psi''(kc(x))\nabla c(x) & I_p & 0 \\ k\nabla g(x) & 0 & I_q \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda,\nu) \\ \bar{\lambda} - \lambda \\ -kg(x) \end{bmatrix}$$

(see Chapter 2 for an explanation of the terms).

By multiplying through the second and third rows, this matrix becomes

$$\begin{bmatrix} \nabla^2_{xx} L(x,\lambda,\nu) & -\nabla c^T(x) & -\nabla g^T(x) \\ -\nabla c(x) & \frac{1}{k}(\Lambda\Psi''(kc(x)))^{-1} I_p & 0 \\ -\nabla g(x) & 0 & -\frac{1}{k} I_q \end{bmatrix}.$$

The Hessian of the Lagrangian $\nabla^2_{xx} L(x,\lambda,\nu)$ is symmetric positive definite or can be regularized so that it is positive definite. Since $\lambda_i \psi''(kc_i(x)) < 0$ for $i = 1...p$ (see the properties of $\psi(x)$ in section 2.2.2) the matrix

$$\begin{bmatrix} \frac{1}{k}(\Lambda\Psi''(kc(x)))^{-1} I_p & 0 \\ 0 & -\frac{1}{k} I_q \end{bmatrix}$$

is diagonal and, thus, symmetric negative definite. So the matrix that arises in the exterior-point method can be written as a symmetric quasi-definite system and solved using an LDL^T factorization [47].

Benchmarking Direct and Iterative Methods for Dense Systems

To create benchmark tests for a number of linear solve routines, an AMPL model is created to solve the following problem

$$\text{minimize } \frac{1}{2}x^TAx - x^Tb.$$

For a positive definite matrix, A, the solution to this problem is x such that $Ax = b$.

To create the $n \times n$ matrix A and $n \times 1$ vector b, the matrix A is defined as $A = HH^T$ where the elements of H are uniformly random in (0,1) and the elements of b are random in (0,n). In this way, the matrix A will be a dense symmetric positive definite matrix. In order to change test the methods for different condition numbers, the condition number of the matrix A is modified by adding a diagonal matrix to the matrix A. Here the matrices used are the identitiy matrix I, 1000*I, and D=diag$\{1,...,n\}$ where diag$\{1,...,n\}$ is a matrix with 1,...,n along the diagonal of the matrix.

Linear Solvers

The linear solve routines used are direct and iterative methods from open source linear algebra packages. The two direct methods benchmarked are a Cholesky factorization and solve routine and a Gaussian ellimination routine both from the LAPACK library. The Gaussian ellimination method solves a general real matrix with partial pivoting. The Cholesky factorization method solves a system of equations for a real symmetric matrix. More information can be found at http://www.netlib.org/lapack/.

The iterative routines benchmarked are the conjugate gradient method, the biconjugate gradient stabilized method (BiCgStab), the generalized minimal residual method (GMRES) and the quasi-minimal residual method (QMR). The iterative methods used for benchmarking are all from the GMM+ library. More information on the GMM++ library can be found at http://download.gna.org/getfem/html/homepage/gmm/index.html.

Experimental Setup

All of the experiments were run on a 2.2 GHz Intel Pentium Dual Core desktop computer with 2 GB of RAM and a Windows XP operating system.

Timing the Routines

The time taken was computed using the clock() routine defined as part of the header file time.h. The routine clock() returns the number of clock ticks ellapsed for the currently called process. By differencing two clock() values and dividing by the macro CLOCKS_PER_SEC, the total amount of time for the code can be computed in seconds.

Matrix Solve Routines

The following graphs show the amount of time taken to solve an N-by-N system of equations using two direct methods, Cholesky factorization and Gaussian Ellimination and four iterative methods, Conjugate Gradient, biconjugate gradient stabilized, GMRES and QMR. Each graph shows matrices constructed in different ways resulting in different sets of condition numbers.

Figure 3.1 shows solution times for solving a system of linear equations where the matrix is constructed as $A = HH^T + I$. The elements of H are uniformly random in (0,1) and the matrix I is an N-by-N identity matrix. For the solution times in figure 3.1, the condition number for each matrix is shown in table 3.1. From 3.1 we see that the Cholesky factorization and solve routine outperforms all other methods for matrices up to size $N = 3000$ with $A = HH^T + I$.

Figure 3.2 shows solution times for solving a system of linear equations where the matrix is constructed as $A = HH^T + 1000 * I$. The elements of H are uniformly random in (0,1) and the matrix I is an N-by-N identity matrix. For the solution times in figure 3.2, the condition number for each matrix is shown in table 3.2. From table 3.2 we see that the condition numbers for this set of matrices is much less than the condition numbers in table

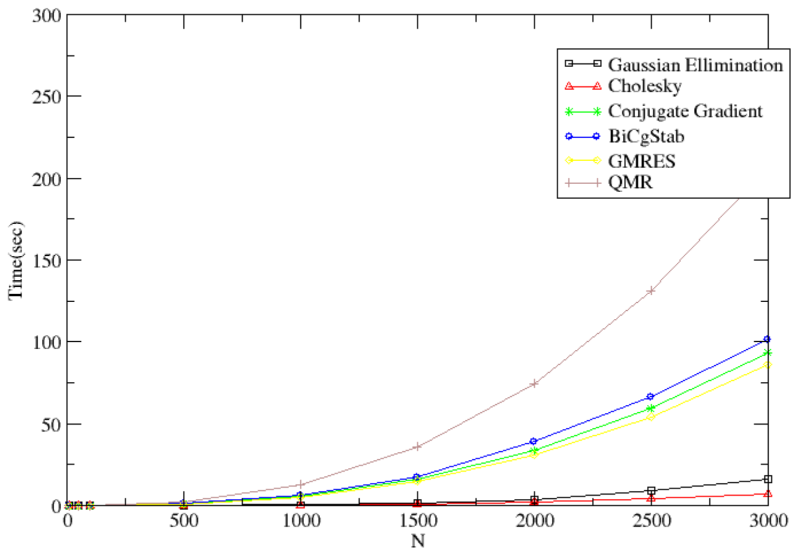

Figure 3.1: Solution time with $A = HH^T + I$

3.1. For smaller condition numbers, the iterative conjugate gradient, BiCgStab and GMRES methods outperform the direct methods as can be seen in figure 3.2.

Figure 3.3 shows solution times for solving a system of linear equations where the matrix is constructed as $A = HH^T + D$. The elements of H are uniformly random in (0,1) and the matrix $D = \text{diag}(1, ..., N)$ is a diagonal matrix with the numbers $1, ..., N$ along the diagonal. For the solution times in figure 3.3, the condition number for each matrix is shown in table 3.3. The condition numbers for this set of matrices shown in table 3.3 fall inbetween those in table 3.1 and table 3.2. Once again, the Cholesky factorization and solve method outperforms the other methods for matrices of up to size $N = 3000$. However, the solution time for the Cholesky factorization routine increases as N^3, which we can see in figure 3.3. Thus, this method may not outperform other methods for systems much larger than $N = 3000$.

Table 3.1: Condition numbers with $A = HH^T + I$

N	10	50	100	500	1000
Condition Number	25.3	620.94	2509.21	6.26E+004	2.50E+005
N	1500	2000	2500	3000	
Condition Number	5.63E+005	1.00E+006	1.56E+006	2.25E+006	

Table 3.2: Condition numbers with $A = HH^T + 1000 * I$

N	10	50	100	500	1000
Condition Number	1.03	1.64	3.55	63.64	251.26
N	1500	2000	2500	3000	
Condition Number	563.59	1000.31	1563.98	2250.89	

Table 3.3: Condition numbers with $A = HH^T + D$

N	10	50	100	500	1000
Condition Number	15.92	162.64	345.17	2192.6	4534.71
N	1500	2000	2500	3000	
Condition Number	6781.92	9218.39	11688.02	13874.58	

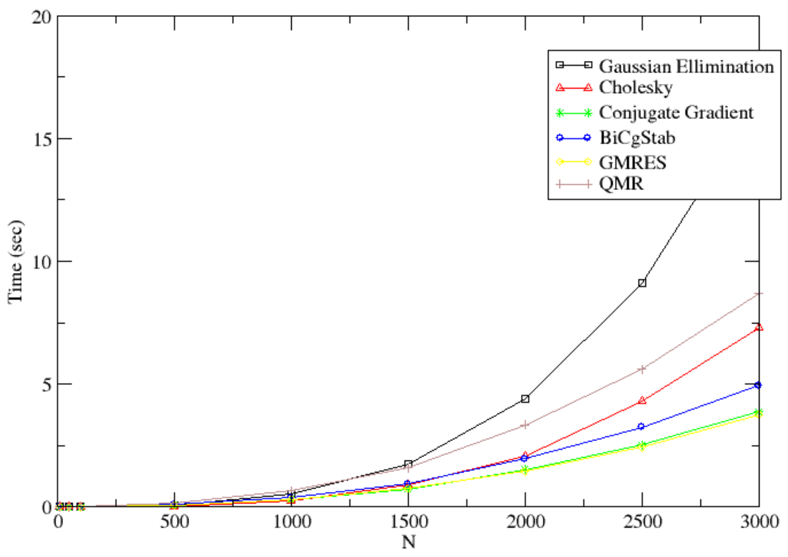

Figure 3.2: Solution time with $A = HH^T + 1000 * I$

Conclusions for Direct and Iterative Methods

For the direct methods, Cholesky factorization and Gaussian ellimination, the condition number does not affect the time taken to solve the systems. The time taken to solve each matrix for both Gaussian Ellimination and Cholesky factorization increases as n^3, with Cholesky factorization taking about half the time as Gaussian Ellimination.

Cholesky factorization outperforms all of the routines when the condition number is large and the size of the systems are small. When the condition number is small, the iterative GMRES, BiCgStab and Conjugate Gradient methods outperform Cholesky factorization.

Unfortunately, computing the condition number when deciding which routine to use is prohibitive, since the cost of computing the condition number is $O(n^3)$, which is on the same order as solving the system itself. Also, for the AMPL code, the systems should not be much larger than $N = 3000$ for dense systems, since this is a current memory limitation

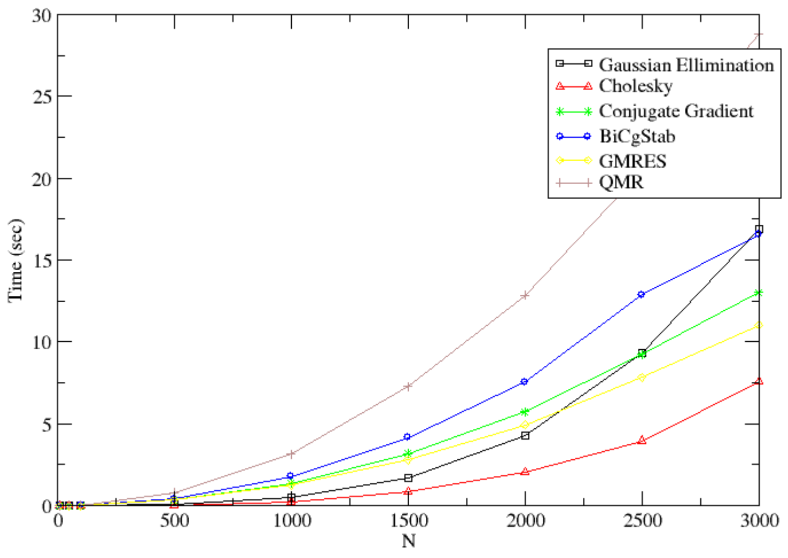

Figure 3.3: Solution time with $A = HH^T + D$

for the AMPL interface. Thus, in the current solver implementation, Cholesky factorization is used as default since it better handles smaller systems and systems with large condition numbers.

3.1.2 Dense and Sparse Systems

The density of a system affects the memory requirements and efficiency of a linear system solver. A solver designed for sparse systems will be more efficient on sparse systems than a dense solver and vice versa. Real world problems exist where the matrices that arise will have different degrees of sparsity. Consider the system that arises in the PD EPM

$$\begin{bmatrix} \nabla^2_{xx}L(x,\lambda,\nu) & -\nabla c^T(x) & -\nabla g^T(x) \\ -k\Lambda\Psi''(kc(x))\nabla c(x) & I_p & 0 \\ k\nabla g(x) & 0 & I_q \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda,\nu) \\ \bar{\lambda}-\lambda \\ -kg(x) \end{bmatrix}$$

where $\Psi''(kc(x)) = \text{diag}(\psi''(kc_i(x)))_{i=1}^p$, $\Lambda = \text{diag}(\lambda_i)_{i=1}^p$, $\bar{\lambda} = \Psi'(kc(x))\lambda$ and I_p and I_q are $p \times p$ and $q \times q$ identity matrices. And where

$$\nabla_x L(x,\lambda,\nu) = \nabla f(x) - \sum_{i=1}^{p} \lambda_i \nabla c_i(x) - \sum_{j=1}^{q} \nu_j \nabla g_j(x)$$

and

$$\nabla^2_{xx} L(x,\lambda,\nu) = \nabla^2 f(x) - \sum_{i=1}^{p} \lambda_i \nabla^2 c_i(x) - \sum_{j=1}^{q} \nu_j \nabla^2 g_j(x).$$

The degree of sparsity of the matrix that arises depends on the sparsity of the Hessian of the Lagrangian, $\nabla^2_{xx}L(x,\lambda,\nu)$, and the sparsity of the constraint gradients, $\nabla c_i(x)$ for $i = 1,...,p$ and $\nabla g_j(x)$ for $j = 1,...,q$. The following two problems demonstrate how different problems can have different degrees of sparsity.

Catenary Problem

The catenary problem looks to determine the shape of a hanging cable fixed at two ends [2]. One end is fixed at point x_a, y_a and the other end at point x_b, y_b. To determine the shape of the cable, we look to minimize the potential energy

$$\min \int_0^L m(l)y(l)dl$$

where L is the length of the cable, $l \in [0, L]$, and $\int_0^L m(l)dl = M$ is the total mass of the cable. If the cable is uniformly discretized into N segments using variables x_l, $l = 0,...,N$

and y_i, $i = 0, ..., N$ then we also require each segment length to be equal to L/N. Thus, we desire to solve the following optimization problem with discretized variables [2]

$$\begin{aligned} \text{minimize} \quad & \frac{M}{N}\sum_{l=0}^{N} y_l \\ \text{subject to} \quad & (x_l - x_{l-1})^2 + (y_l - y_{l-1})^2 = (\tfrac{L}{N})^2,\ l = 1, ..., N-1 \\ & x_0 = x_a,\ x_N = x_b \\ & y_0 = y_a,\ y_N = y_b. \end{aligned}$$

Recall again, that the degree of sparsity of the PD EPM matrix depends on the sparsity of the Hessian of the Lagrangian, $\nabla^2_{xx}L(x, \lambda, \nu)$, and the sparsity of the constraint gradients, $\nabla c_i(x)$ for $i = 1, ..., p$ and $\nabla g_j(x)$ for $j = 1, ..., q$.

Since the objective function for the catenery problem is linear, we have $\nabla^2 f(x) = 0$. Thus, the Hessian of the Lagrangian is $\nabla^2_{xx}L(x, \nu) = -\sum_{l=1}^{N-1} \nu_l \nabla^2 g_l(x)$ where $g_l(x) = (x_l - x_{l-1})^2 + (y_l - y_{l-1})^2 - (\frac{L}{N})^2$ for each of the $N-2$ non-linear equality constraints. So, the Hessian of the Lagrangian is a sparse tri-diagonal matrix with the only non-zero elements on the main diagonal and the first diagonals above and below the main diagonal.

Each of the constraint gradients is sparse as well. Since each constraint is $g_l(x) = (x_l - x_{l-1})^2 + (y_l - y_{l-1})^2 - (\frac{L}{N})^2$, each constraint gradient $\nabla g_l(x)$ has only 4 non-zero elements

$$\nabla g_l(x) = \begin{pmatrix} \vdots \\ -2(x_l - x_{l-1}) \\ 2(x_l - x_{l-1}) \\ \vdots \\ -2(y_l - y_{l-1}) \\ 2(y_l - y_{l-1}) \\ \vdots \end{pmatrix}$$

Since both the Hessian of the Lagrangian and the constraint gradients are sparse for the catenary problem, the matrix that arises in the PD EPM will be a sparse matrix with few non-zero elements.

Support Vector Machine Problem

Support vector machine (SVM) techniques are used for classifying m training points $x_i \in \mathbb{R}^n$ into 2 classes where a separating hyperplane is used to separate the classes. The training points are classified by specifying a scalar y_i such that if x_i is in a specific class, then $y_i = 1$, otherwise $y_i = -1$. The following optimization model is used to solve the support vector machine problem, which is derived in detail in section 5.1

$$\begin{aligned} \underset{\alpha \in \mathbb{R}^m}{\min}\text{imize} \; & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{subject to} \; & \sum_{i=1}^{m} \alpha_i y_i = 0 \\ & 0 \le \alpha_i \le C, i = 1, ..., m. \end{aligned}$$

Here, the objective function is non-linear, there is one equality constraint, and there are $2m$ bounds on the variables $\alpha \in \mathbb{R}^m$. The bounds on the variables can be folded over onto the Hessian of the Lagrangian to reduce the size of the PD EPM system. This is explained in detail for the SVM problem in section 5.2 and in general in section 3.1.3.

Folding over the bounds and setting $M(\alpha, \lambda, \nu) = \nabla^2_{\alpha\alpha} L(\alpha, \lambda, \nu) - k \nabla c^T(\alpha) \Lambda \Psi''(kc(\alpha)) \nabla c(\alpha)$, the PD EPM system is reduced to

$$\begin{bmatrix} M(\alpha, \lambda, \nu) & -\nabla g^T(\alpha) \\ k \nabla g(\alpha) & 1 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_\alpha L(\alpha, \bar{\lambda}, \nu) \\ -kg(\alpha) \end{bmatrix}.$$

From the optimization model, we have each element of the Hessian of the objective function as $\nabla^2 f(x)_{ij} = y_i y_j x_i^T x_j$ which is likely non-zero for each $i \in (1, m)$ and $j \in (1, m)$. Thus the Hessian of the objective function is dense, and the matrix $M(\alpha, \lambda, \nu)$ is dense as well.

The gradient of the equality constraint is also dense. Since the equality constraint is $g(\alpha) = \sum_{i=1}^{m} \alpha_i y_i$, we have $\nabla g^T(\alpha) = \begin{pmatrix} y_1 & y_2 & \cdots & y_m \end{pmatrix}$ which contains only non-zero elements.

Since both the Hessian of the Lagrangian and the constraint gradient is dense for the SVM problem, the matrix that arises in the PD EPM will be a dense matrix with mostly, if not all, non-zero elements.

Here we looked at two problems, the catenary problem and the SVM problem, where the different degrees of sparsity warrent using different solvers for the PD EPM system which arises. The next sections discuss dense and sparse matrix representations, and cover benchmarking of open source sparse and dense solvers.

Dense Systems

Dense systems have many non-zero elements that are typically stored in arrays. For a dense system, an $n \times m$ matrix is stored in a single array of size nm.

Consider the following 3×4 matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 6 & 3 & 9 & 5 \\ 7 & 1 & 4 & 8 \end{bmatrix}.$$

We represent this matrix row-wise by a single array A as

```
A = [ 1 2 3 4 6 3 9 5 7 1 4 8 ]
```

or column-wise by a single array A as

```
A = [ 1 6 7 2 3 1 3 9 4 4 5 8 ].
```

These representations store nm doubles in each array.

Different programming languages represent matrices in different ways. For example C and C++ use a row-wise representation, where as FORTRAN uses a column-wise representation.

Sparse Systems

Sparse systems are systems that have few non-zero elements and are stored such that only the non-zero elements are represented.

One way to represent sparse matrices is compressed sparse column format. If an $n \times m$ sparse matrix has k non-zero elements, the compressed sparse column format represents the matrix using the following 3 arrays:

1. A, of size k, contains the non-zero elements of the matrix in column major format
2. IA, of size k, contains the row position of each non-zero element
3. JA, of size $m + 1$, contains the position in A where each column starts.

For example, the 3×4 matrix

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 3 & 9 & 5 \\ 0 & 1 & 4 & 0 \end{bmatrix}$$

has compressed sparse column format representation

```
A = [ 1 2 3 1 9 4 5 ]

IA = [ 0 0 1 2 1 2 1 ]

JA = [ 0 1 4 6 7 ]
```

The compressed sparse column format stores $2k+m+1$ doubles in the three arrays.

Benchmarking Open Source Sparse and Dense Solvers

Whether to use a sparse or dense solver will depend on the sparsity of the system and the memory requirements. To create benchmark tests for sparse systems, first an $n \times n$ sparse symetric system, A, is created with a density p of non-zero elements uniformly random in (0,1). A matrix αI is added so as to make this system positive definite, with $\alpha \in \mathbb{R}$ and I an $n \times n$ identity matrix. Then a dense vector b is created with uniformly random elements in $(0, n)$ and an AMPL model is created to solve the following problem,

$$\text{minimize } \frac{1}{2} x^T A x - x^T b.$$

For a positive definite matrix, A, the solution to this problem is x such that $Ax = b$.

Sparse Linear Solvers

Three sparse linear system solver packages were looked at for the sparse solve routines.

SuperLU The SuperLU package includes C routines for solving sparse linear systems using a sparse Gaussian Elimination method. For this solver, the sparse matrix is represented using compressed sparse column format as is returned by AMPL. More information on SuperLU can be found at http://crd.lbl.gov/ xiaoye/SuperLU/.

SPARSKIT The SPARSKIT package includes FORTRAN routines for solving sparse

systems using iterative methods. For this solver, the sparse matrix is represented using compressed row format. This format is similar to compressed column format except the array of non-zeros is stored along each row, another array stores the column position of each non-zero element and another array stores the position of where each row starts. This package includes a number of iterative routines, including a conjugate gradient routine, a generalized minimal residual method (GMRES), and a biconjugate gradient stabalized (BiCgStab) routine. More information on SPARSKIT can be found at
http://www-users.cs.umn.edu/ saad/software/SPARSKIT/sparskit.html.

CHOLMOD The CHOLMOD package includes C routines for factoring and solving sparse linear systems using either an LL^T or an LDL^T factorization. This solver can be used to solve symetric positive-definite or quasi-definite matrices. For this solver, the upper or lower triangular portion of the matrix is stored in compressed sparse column format. More information on CHOLMOD can be found at http://www.cise.ufl.edu/research/sparse/cholmod/.

Experimental Setup

All of the experiments were run on a 2.2 GHz Intel Pentium Dual Core desktop computer with 2 GB of RAM and a Windows XP operating system.

Timing the Routines

The time taken for each portion of the code was computed using the clock() routine defined as part of the header file time.h. The routine clock() returns the number of clock ticks ellapsed for the currently called process. By differencing two clock() values and dividing by the macro CLOCKS_PER_SEC, the total amount of time for a particular portion of code can be computed in seconds.

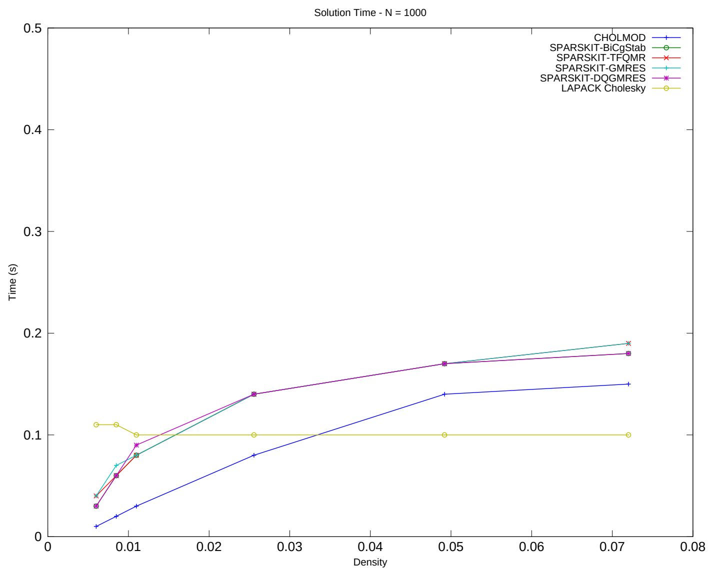

Figure 3.4: Solution time for system of size $N = 1000$

Sparse Matrix Solve Routines

The following graphs show the amount of time taken to solve an N-by-N system of equations using two sparse direct methods, Cholesky factorization from the CHOLMOD library and Gaussian Ellimination from the SuperLU library and three iterative methods from the SPARSKIT library: Conjugate Gradient, BiCgStab, and GMRES.

Figures 3.4-3.6 show solution times for solving a sparse system of linear equations $Ax = b$ versus density of non-zeros for systems of size N. The system A is created with a density p of non-zero elements uniformly random in (0,1). A matrix αI is added to A so as to make this system positive definite. Tables 3.4-3.6 show the condition number for each system versus the density. From these plots, we see that the CHOLMOD sparse Cholesky routine outperforms all other routines for matrices with density of less than about 2.5% non-zeros. For matrices with greater than 2.5% density of non-zeros, the LAPACK Cholesky routine outperforms all other routines.

Table 3.4: Condition numbers for systems of size $N = 1000$

Density	0.0059	0.0084	0.0109	0.0255	0.0491	0.0720
Condition #	11.9925	14.1825	15.630	22.1009	29.1307	35.4093
Density	0.1025	0.2107	0.5012	0.7502	1.0	
Condition #	41.4580	59.4668	90.4139	397.5144	519.1860	

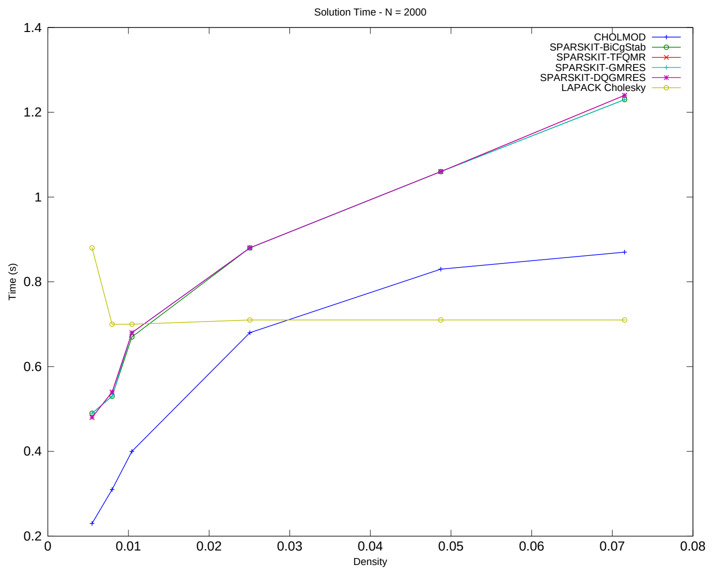

Figure 3.5: Solution time for system of size $N = 2000$

Table 3.5: Condition numbers for systems of size $N = 2000$

Density	0.0054	0.0079	0.0104	0.0250	0.0487	0.0715
Condition #	15.4408	18.7116	19.9866	29.8391	40.6878	49.1857
Density	0.1021	0.2101	0.5016	0.7499	1.0	
Condition #	58.0677	83.2285	127.6178	781.0443	1027.2610	

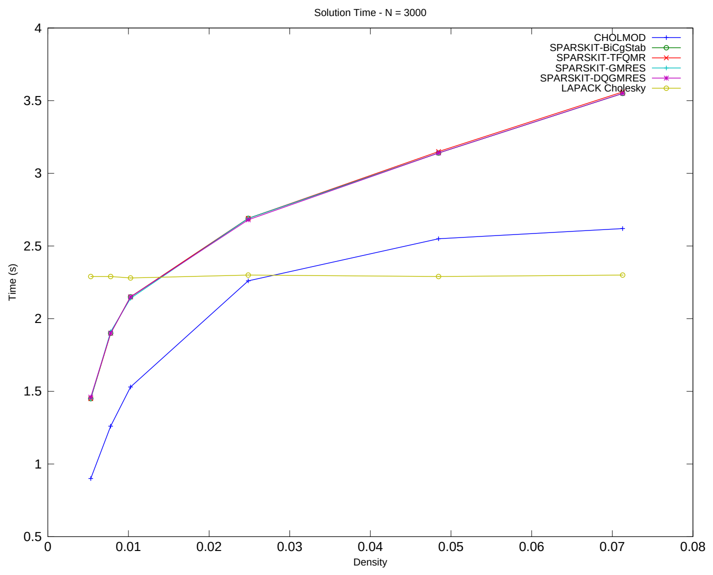

Figure 3.6: Solution time for system of size $N = 3000$

Table 3.6: Condition numbers for systems of size $N = 3000$

Density	0.0053	0.0077	0.0102	0.0248	0.0484	0.0712
Condition #	18.1724	21.6080	24.3459	35.6859	49.3490	59.5692
Density	0.1018	0.2099	0.4998	0.7504	1.0	
Condition #	70.8823	101.1742	786.6454	1163.4041	1532.5275	

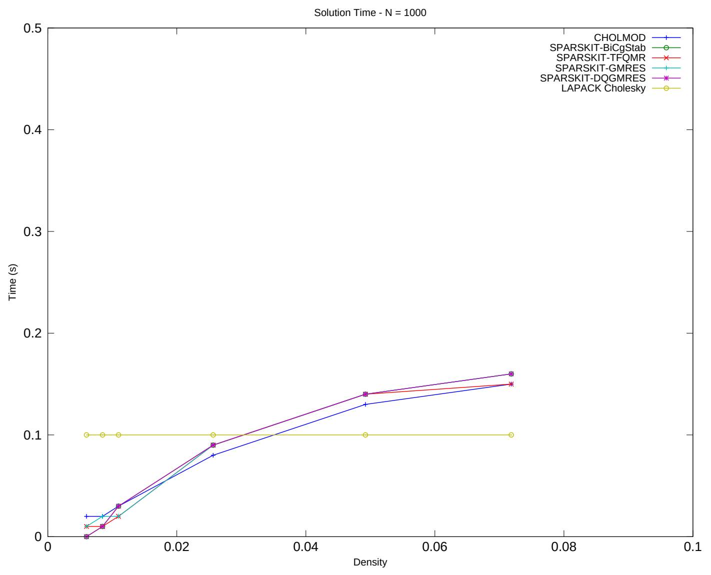

Figure 3.7: Solution time for system of size $N = 1000$

Figures 3.7-3.9 show solution times for solving a sparse system of linear equations $(A + D)x = b$ versus density of non-zeros for systems of size N. The system A is created with a density p of non-zero elements uniformly random in (0,1). A matrix αI is added to A so as to make this system positive definite and the matrix $D = \text{diag}\{1...N\}$. Tables 3.7-3.9 show the condition number for each system versus the density. Due to smaller condition numbers, the iterative routines now outperform the sparse Cholesky routine, CHOLMOD. From figures 3.7-3.9 we see that the SPARSKIT iterative routines outperform all other routines for matrices with density of less than about 5% non-zeros. For matrices with a greater density of non-zeros, the LAPACK Cholesky routine outperforms all other routines.

Figure 3.10 shows time taken to solve $Ax = b$ with just the sparse CHOLMOD routine and the LAPACK Cholesky routine for different densities of non-zeros. The time it takes for the sparse CHOLMOD routine to solve the system with matrices of density 2.5% is very close to the same as the time it takes for the dense LAPACK Cholesky routine to solve a dense system of the same size. From figure 3.10, it's clear that the LAPACK Cholesky

Table 3.7: Condition numbers for systems of size $N = 1000$

Density	0.0059	0.0084	0.0109	0.0256	0.0492	0.0718
Condition #	130.1541	120.5312	107.7516	82.9792	65.6894	55.0864
Density	0.1023	0.2101	0.5015	0.7496	1.0	
Condition #	47.9275	35.1742	24.5846	49.1700	60.1678	

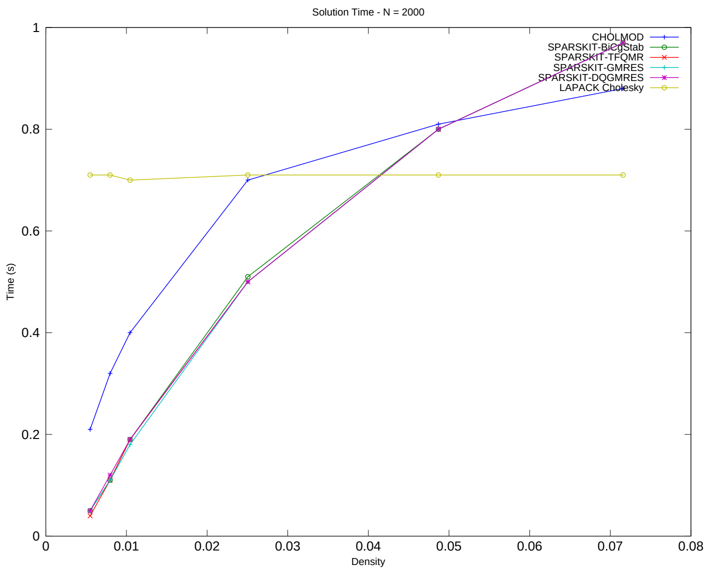

Figure 3.8: Solution time for system of size $N = 2000$

Table 3.8: Condition numbers for systems of size $N = 2000$

Density	0.0054	0.0079	0.0104	0.0250	0.0486	0.0715
Condition #	213.2479	191.3632	163.6736	123.9782	93.6639	79.5784
Density	0.1019	0.2102	0.5012	0.7505	1.0	
Condition #	67.5594	48.9418	34.8943	70.8404	86.6305	

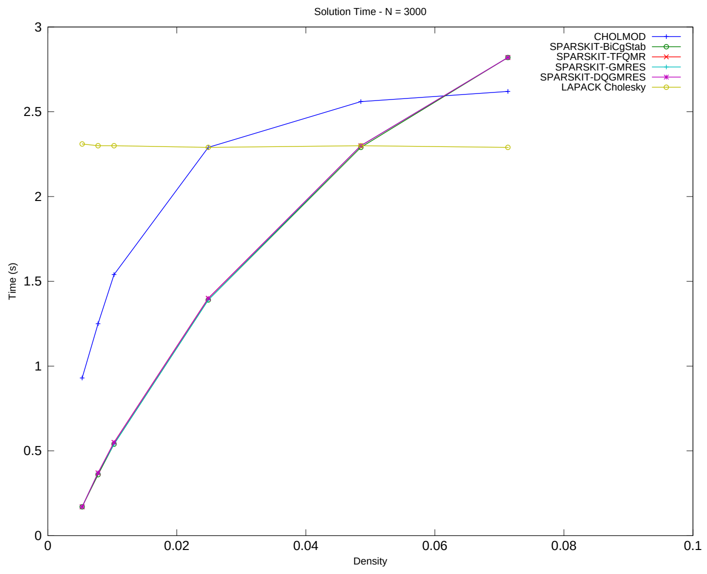

Figure 3.9: Solution time for system of size $N = 3000$

Table 3.9: Condition numbers for systems of size $N = 3000$

Density	0.0053	0.0077	0.0102	0.0248	0.0485	0.0713
Condition #	284.3150	250.5979	221.7237	160.9420	116.1884	98.3555
Density	0.1018	0.2100	0.5000	0.7500	1.0	
Condition #	83.4559	60.1442	83.4530	88.6431	107.2341	

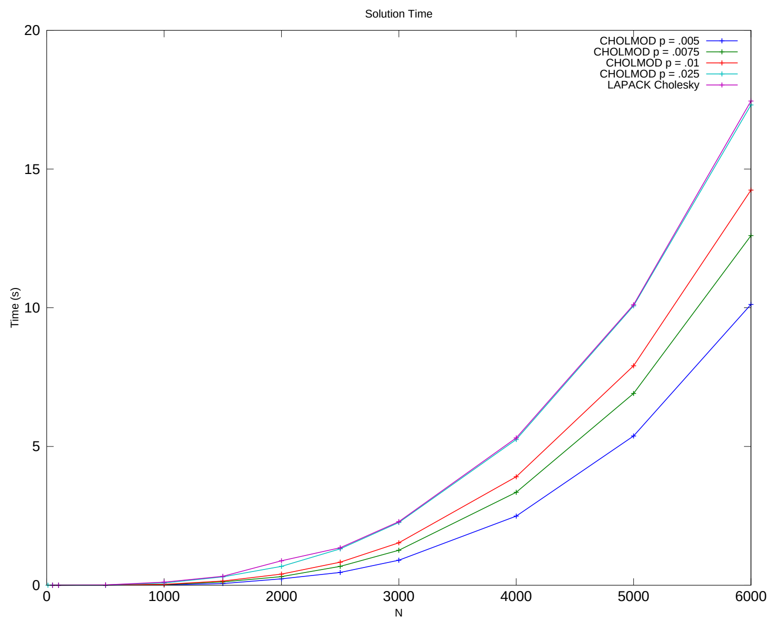

Figure 3.10: Time to solve $Ax = b$ with sparse CHOLMOD and dense LAPACK routines

routine performs about as well as the CHOLMOD routine for matrices with 2.5% density of non-zeros, with the CHOLMOD routine performing better for matrices with density less than 2.5%.

Figure 3.11 shows time taken to solve $(A + D)x = b$ with just the SPARSKIT sparse iterative DQGMRES routine and the LAPACK Cholesky routine. In figure 3.11, times are plotted for the LAPACK Cholseky routine and the SPARSKIT DQGMRES routine for matrices with density less than 5%. From this figure, it's clear that the SPARSKIT DQGM-RES routine outperforms the LAPACK Cholesky routine for matrices with 5% density of non-zeros or less, with the LAPACK Cholesky routine performing better for matrices with density greater than 5%.

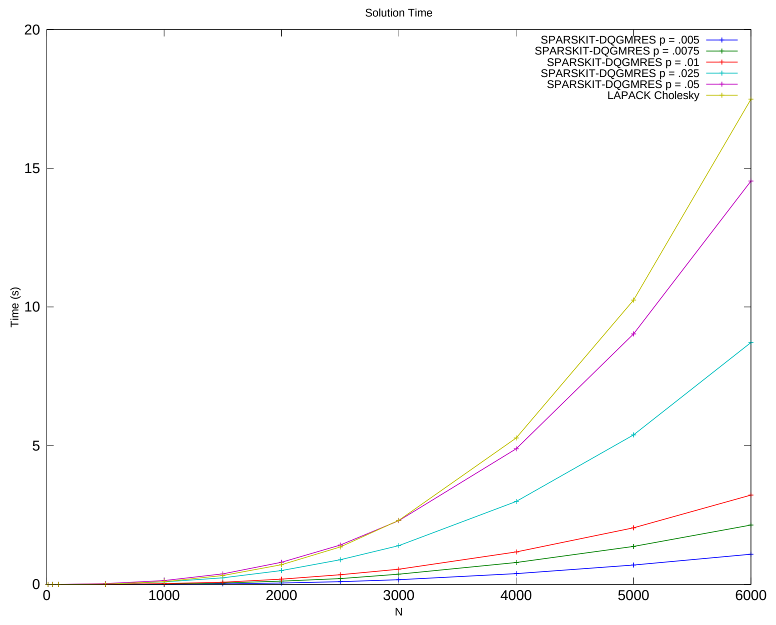

Figure 3.11: Time to solve $(A + D)x = b$ with sparse SPARSKIT and dense LAPACK routines

Conclusions for Sparse and Dense Methods

For direct methods, the sparse CHOLMOD routine outperforms the dense LAPACK Cholesky routine for matrices with less than 2.5% non-zeros. For iterative methods, the SPARSKIT routines can outperform the dense LAPACK Cholesky routine when the condition number is small. For example, when solving the system $(A + D)x = b$, the sparse iterative routines outperform the dense LAPACK Cholesky routine for matrices with density of non-zeros less than 5%.

Unfortunately, as in section 2.2.1, computing the condition number when deciding which routine to use is prohibitive, since the cost of computing the condition number is $O(n^3)$, which is on the same order as solving the system itself. Thus, in the current solver implementation, the dense LAPACK Cholesky routine is used for matrices with density of non-zeros greater than 2.5% and the sparse CHOLMOD routine is used for matrices with density of non-zeros less than 2.5%.

3.1.3 Reducing Matrix Size

Another way to solve the system that arises in the PD EPM is to write it as a smaller, but possibly dense system of equations. A smaller system may be more efficiently solved, but if a system that was originally sparse becomes dense, then it may become more challenging. One of the goals is to determine when the system will be more efficiently solved as a smaller system.

Let the diagonal matrix $D = -k\Lambda\Psi''(\cdot)$. Then the system of equations that arises is

$$\begin{bmatrix} \nabla^2_{xx}L(x) & -\nabla c^T(x) & -\nabla g^T(x) \\ -\nabla c(x) & -D^{-1} & 0 \\ -\nabla g(x) & 0 & -k^{-1}I_q \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x) \\ -D^{-1}(\bar{\lambda} - \lambda) \\ g(x) \end{bmatrix}.$$

If $y = \begin{bmatrix} \lambda \\ \nu \end{bmatrix}$, $A = \begin{bmatrix} -\nabla c(x) \\ -\nabla g(x) \end{bmatrix}$, $\rho = \begin{bmatrix} -D^{-1}(\bar{\lambda} - \lambda) \\ g(x) \end{bmatrix}$ and $\hat{D} = \begin{bmatrix} D^{-1} & 0 \\ 0 & k^{-1} I_q \end{bmatrix}$, then

$$\begin{bmatrix} \nabla^2_{xx} L(x) & A^T \\ A & -\hat{D} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x) \\ \rho \end{bmatrix}$$

and

$$\Delta y = (-\hat{D})^{-1}(\rho - A\Delta x)$$

$$(\nabla^2_{xx} L(x) + A^T \hat{D}^{-1} A)\Delta x = -\nabla_x L(x) + A^T \hat{D}^{-1} \rho.$$

The only system now is $n \times n$, where as the original system was $(n+p+q) \times (n+p+q)$. However, if the original linear system is sparse, the matrix $A^T \hat{D}^{-1} A$ could be dense if A has dense rows or columns.

Determining to reduce the size of the system which arises in the PD EPM depends on the size and sparsity of the original and reduced systems. Figure 3.10 can be used to help determine when to reduce matrix size. For example, for matrices of size N with density of non-zeros of 1% it is faster to solve a smaller dense matrix of size less than about $(N - 500)$ using LAPACK's Cholesky routine than a sparse matrix of size N using CHOLMOD, and for matrices with density of non-zeros of .5%, it is faster to solve a smaller dense matrix of size less than about $(N - 1000)$ using LAPACK's Cholesky routine than a sparse matrix of size N using CHOLMOD. However, figure 3.10 does not take into account the time it takes to construct the reduced system which can become prohibitive with dense constraints.

Section 4.2 discusses details on the design of the algorithm with section 4.2.2 discussing when matrix reduction is used. Also, section 4.2.4 discusses how bounds on the variables, which are very sparse constraints, are folded over to reduce matrix size.

3.2 Matrix Regularization

Matrix regularization techniques are used to ensure that the matrices arising in the unconstrained minimization in the NRAL technique are sufficiently positive definite. Matrix regularization is also used to ensure that the matrix arising in the PD EPM is a quasi-definite matrix.

3.2.1 Ensuring a Descent Direction

A positive definite matrix is required when using Newton's method for minimization to ensure a descent direction at a point x_k. For an iteration of Newton's method, we have

$$\nabla^2 f(x_k)p_k = -\nabla f(x_k)$$

so

$$p_k = -(\nabla^2 f(x_k))^{-1}\nabla f(x_k).$$

The direction p_k is a descent direction at the point x_k if $p_k^T \nabla f(x_k) < 0$. Using Newton's method we have

$$p_k^T \nabla f(x_k) = -(\nabla f(x_k))^T(\nabla^2 f(x_k))^{-1}\nabla f(x_k).$$

So p_k is a descent direction if $\nabla^2 f(x_k)$ is positive definite.

Let $A = \nabla^2 f(x_k)$. If A is not positive definite, a diagonal matrix E can be added such that $A + E$ is sufficiently positive definite. Then the Newton's method iteration is $(A+E)p_k = -\nabla f(x_k)$, and p_k is a descent direction at x_k.

3.2.2 Matrix Regularization During Cholesky Factorization

A simple way to perform matrix regularization is during Cholesky factorization. If the matrix A is not sufficiently positive definite then during its Cholesky factorization a diagonal element a_{kk} appears with $a_{kk} < \delta$, where $\delta > 0$ is a small number close to 0. We replace

a_{kk} with $|a_{kk}|$ if $|a_{kk}| > \delta$, and with δ otherwise. Replacing a_{kk} with δ or $|a_{kk}|$ is equivalent to adding a diagonal matrix E to A such that $A + E$ is sufficiently positive definite [2].

One goal of matrix regularization is to ensure that $\|E\|$ is not much larger than $\inf\{\Delta A : A + \Delta A \text{ is positive definite}\}$ [48]. Fang and O'Leary [48] give other methods for regularization during Cholesky factorization that are more complicated, but they ensure that $\|E\|$ does not become too large. One of these methods, the GMW81 algorithm, is described in more detail in the next section 3.2.3.

Determining how to perform matrix regularization depends on keeping $\|E\|$ from becoming too large and also on the computational complexity of the matrix regularization technique. In the EPM solver, two regularization techiques are implemented, one to ensure that the matrix which arises in the NRAL technique is sufficiently positive definite and one to ensure that the matrix which arises in the PD EPM iteration is quasi-definite.

The next two sections, 3.2.3 and 3.2.4, describe the two matrix regularization techniques which are used in the implementation of the algorithm. In the next chapter, section 4.2.3 discusses when each of these regularization methods is used in the implementation.

3.2.3 Positive Definite Regularization

For regularizing the positive definite matrix which arises in the NRAL technique, a matrix E is found such that $A + E$ can be factored as $A + E = L^T L$. To do this, the GMW81 algorithm is used as described by Gill, Murray and Wright [49].

First we set $\beta^2 = \max\{\eta, \frac{\xi}{\sqrt{n^2-1}}, \delta\}$ where ξ is the maximum magnitude of the off diagonal elements of A, η is the maximum magnitude of the diagonal elements of A and n is the size of the matrix A. Here we let $\delta = \epsilon_M$ (machine epsilon).

Then, at each iteration of the Cholesky routine, a_{kk} is checked to see if $a_{kk} > \delta$, and if not, l_{kk} is set as $\max\{\sqrt{\delta}, \sqrt{|\hat{a}_{kk}|}, \frac{\mu_k}{\beta}\}$ where $\mu_k = \max\{|\hat{a}_{kj}| : j = k+1, ..., n\}$ at iteration k.

Pseudo-code for the regularized Cholesky algorithm using the GMW81 algorithm is shown in Algorithm 1.

Algorithm 1 Cholesky Regularization - GMW81 Algorithm [50]

```
$\hat{A} = A$
For $k = 1 : n$
  If $\hat{a}_{kk} \geq \delta$
    $l_{kk} = \sqrt{a_{kk}}$
  Else
    $l_{kk} = \max\{\sqrt{\delta}, \sqrt{|\hat{a}_{kk}|}, \frac{\mu_k}{\beta}\}$
  End
  For $j = k + 1 : n$
    $l_{kj} = \hat{a}_{kj}/l_{kk}$
    For $i = k + 1 : j$
      $\hat{a}_{ij} = \hat{a}_{ij} - l_{kj}l_{ki}$
    End
  End
End
```

3.2.4 Quasi-Definite Regularization

Recall from subsection 3.1.1 the form of a quasi-definite matrix is

$$A = \begin{bmatrix} E & C^T \\ C & -F \end{bmatrix}$$

where $E \in \mathbb{R}^{nxn}$ and $F \in \mathbb{R}^{mxm}$ are symmetric positive definite matrices.

If the matrix A is not quasi-definite, then one of the matrices E or F are not positive definite and regularization can be performed as in [51]. If E is not positive definite, then during the LDL^T factorization of A, a diagonal element a_{kk} with $1 \leq k \leq n$ will appear with $a_{kk} < \delta$, where $\delta > 0$ is a small number close to 0. If F is not positive definite, then a diagonal element a_{kk} with $n+1 \leq k \leq n+m$ will appear with $a_{kk} > -\delta$, where $\delta > 0$ is a small number close to 0.

For elements $1 \leq k \leq n$, a_{kk} is replaced with $|a_{kk}|$ if $|a_{kk}| > \delta$, and with δ otherwise. For elements $n+1 \leq k \leq n+m$, a_{kk} is replaced with $-|a_{kk}|$ if $-|a_{kk}| < -\delta$, and with $-\delta$ otherwise. Replacing a_{kk} with δ or $|a_{kk}|$ for $1 \leq k \leq n$ is equivalent to adding a diagonal matrix D to E such that $E + D$ is sufficiently positive definite. Replacing a_{kk} with $-\delta$ or $-|a_{kk}|$ for $n+1 \leq k \leq n+m$ is equivalent to adding a diagonal matrix D to F such that $F + D$ is sufficiently positive definite.

After regularization, the resulting matrix is quasi-definite and can be solved with an LDL^T factorization routine.

3.3 Newton and Gradient Routines

A number of unconstrained minimization routines were investigated for solving the unconstrained minimization problem that arises in the NRAL technique. The NRAL technique requires use of either a Newton or gradient routine to solve the minimization problem

$$x^{s+1} = \arg\min_{x \in \mathbb{R}^n} \mathcal{L}_k(x, \lambda^s, \nu^s)$$

for x^{s+1} given λ^s and ν^s. Solving this problem is equivalent to finding x^{s+1} such that x^{s+1} solves the following system

$$\nabla f(x) - \sum_{i=1}^{p} \lambda_i^s \psi'(kc_i(x)) \nabla c_i(x) - \sum_{j=1}^{q} (\nu_j^s - kg_j(x)) \nabla g_j(x) = 0.$$

The three unconstrained minimization techniques investigated for solving this minimization problem are Newton's method, the optimal gradient method, and the steepest descent method. The next sections describe these methods in detail.

3.3.1 Newton's Method for Minimization

For optimization problems, Newton's method can be applied to the first-order optimality condition for a local minimizer

$$\nabla f(x) = 0.$$

If x_k is a current Newton's method iterate, then x_{k+1} can be found by solving

$$\nabla^2 f(x_k) p_k = -\nabla f(x_k)$$

for p_k and updating

$$x_{k+1} = x_k + \alpha_k p_k.$$

Here p_k is the search direction and α_k is chosen for the minimization problem such that $f(x_{k+1}) < f(x_k)$.

One method of choosing α_k is to choose the first element in the sequence $\{1, \frac{1}{2}, \frac{1}{4}, ..., 2^{-i}, ...\}$ such that α_k satisfies Armijo's rule.

Armijo's rule requires that

$$f(x_k + \alpha_k p_k) < f(x_k) + \mu \alpha_k p_k^T \nabla f(x_k)$$

for some $0 < \mu < 1$.

Newton's method iterates until $\|\nabla f(x_k)\| < \epsilon$ for some ϵ.

If x_* is a solution to the system of equations, then the sequence $\{x_k\}$ defined by Newton's method converges quadratically to x_*.

3.3.2 Steepest Descent Method

The steepest descent method solves unconstrained minimization problems. If x_k is a current iterate of the steepest descent method, then the next iterate can be found as

$$x_{k+1} = x_k + \alpha_k p_k$$

where

$$p_k = -\nabla f(x_k).$$

As with Newton's method, α_k can be chosen as the first element in the sequence $\{1, \frac{1}{2}, \frac{1}{4}, ..., 2^{-i}, ...\}$ such that α_k satisfies Armijo's rule. Iteration continues until $\|\nabla f(x_k)\| < \epsilon$ for some ϵ.

If x_* is a minimizer of a strongly convex function $f(x)$ then the sequence $\{x_k\}$ defined by the steepest descent method converges linearly to x_* with a rate constant bounded by

$$\left[\frac{\text{cond}(\nabla^2 f(x_*)) - 1}{\text{cond}(\nabla^2 f(x_*)) + 1}\right]^2.$$

A modification of the choice of α_k may speed up this method. Instead of choosing α_k as the first element in the sequence $\{1, \frac{1}{2}, \frac{1}{4}, ..., 2^{-i}, ...\}$ such that α_k satisfies Armijo's rule, α_k is chosen as the first element in the sequence $\{\frac{2}{n}, \frac{1}{n}, \frac{1}{2n}, ..., \frac{1}{2^{i-1}n}, ...\}$ where n is the dimension of the vector x.

3.3.3 Optimal Gradient Method

The optimal gradient method solves an unconstrained minimization problem

$$\min_x f(x)$$

where f is a convex function [52]. Starting with x_0 and y_1 and with $\alpha_0 = 1$, the iterates of one instance of the optimal gradient method are

$$x_k = y_k - \frac{1}{L}\nabla f(y_k)$$

$$\alpha_{k+1} = (1 + \sqrt{4\alpha_k^2 + 1})/2$$

$$y_{k+1} = x_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(x_k - x_{k-1})$$

where L is the Lipschitz constant for $\nabla f(x)$.

The optimal gradient method converges to x_* as [52]

$$f(x_k) - f(x_*) \leq \frac{4L\|x_0 - x_*\|_2^2}{(k+2)^2}.$$

Iteration continues until $\|\nabla f(y_k)\| < \epsilon$ for some ϵ.

3.3.4 Benchmarking Newton and Gradient Routines

Each of the following unconstrained minimization problems was tested with the unconstrained Newton method, the unconstrained gradient method, the unconstrained gradient method with the initial α in Armijo's rule set to $2/N$, and the optimal gradient method. Plots shown are the time taken for each method to solve the problem.

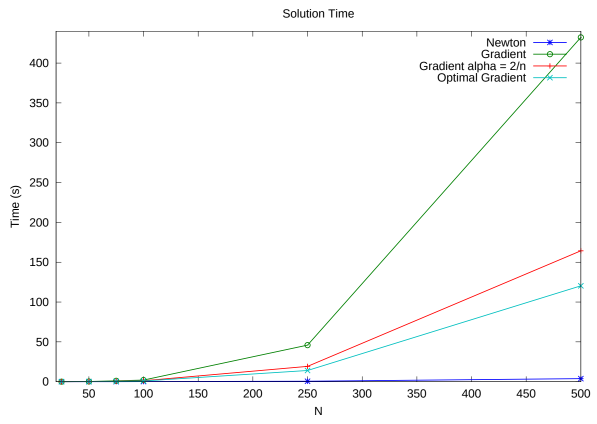

Figure 3.12: Solution time vs. problem size for Problem 1 with accuracy = 1e-5

Problem 1

$$\min_{x\in\mathbb{R}^n}\text{imize}\ \sum_{i=1}^{n} i x_i^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\sin(x_i x_j)$$

For this problem, the inital value is set at $x_1 = x_2 = 2, x_i = 0$ for $i = 3, ..., n$, accuracy is set to 1×10^{-5}, and the Lipshitz constant for the optimal gradient method is set to $2.5n$. Solution time versus problem size for this problem is shown in figure 3.12.

Problem 2

$$\min_{x\in\mathbb{R}^n}\text{imize}\ \sum_{i=1}^{n} i x_i^2 + \sum_{i=1}^{n}\sin(x_i + 5)$$

For this problem, the inital value is set at $x_1 = x_2 = 2, x_i = 0$ for $i = 3, ..., n$, accuracy is set to 1×10^{-5} and the Lipshitz constant for the optimal gradient method is set to $2n$. Solution time versus problem size for this problem is shown in figure 3.13.

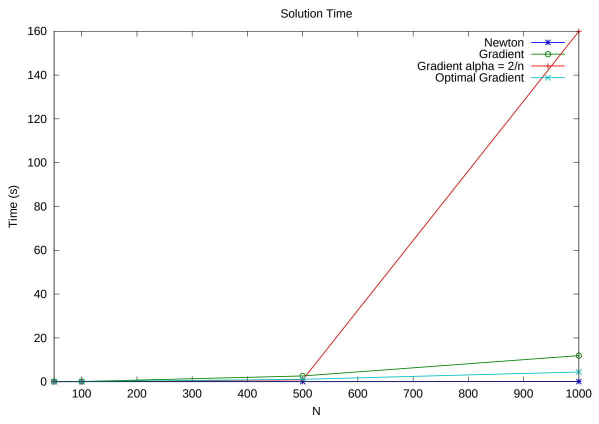

Figure 3.13: Solution time vs. problem size for Problem 2 with accuracy = 1e-5

Problem 3

$$\min_{x\in\mathbb{R}^n} \text{imize} \sum_{i=1}^{n-1}(x_{i+1} - i * x_i)^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\sin(x_i x_j)$$

For this problem, the inital value is set at $x_i = 1$ for i odd and $x_i = 2$ for i even, accuracy is set to 1×10^{-5} and the Lipshitz constant for the optimal gradient method is set to $2(n-1)^2 + .5n$. Solution time versus problem size for this problem is shown in figure 3.14.

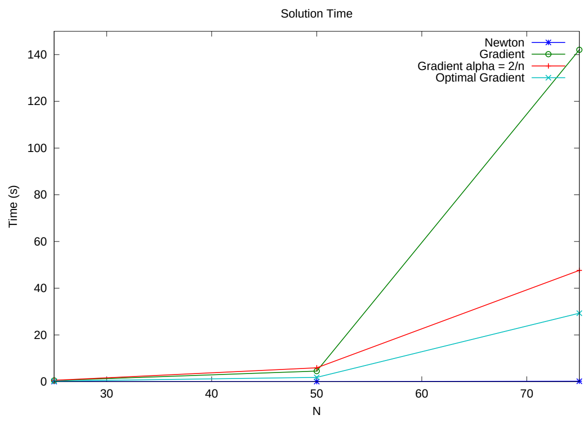

Figure 3.14: Solution time vs. problem size for Problem 3 with accuracy = 1e-5

Problem 4

$$\min_{x\in\mathbb{R}^n} \text{imize} \sum_{i=1}^{n-1} (x_{i+1} - i * x_i)^2$$

For this problem, the inital value is set at $x_i = 1$ for i odd and $x_i = 2$ for i even, accuracy is set to 1×10^{-5} and the Lipshitz constant for the optimal gradient method is set to $2(n-1)^2$. Solution time versus problem size for this problem is shown in figure 3.15.

3.3.5 Conclusions for Newton and Gradient Routines

For all of the problems, Newton's method outperforms all of the gradient methods in terms of solution time. This is due to the few number iterations required for Newton's method for these problems and the small size of the problems. For problems of larger size, where we are able to accurately estimate the Lipshitz constant for the optimal gradient method, the optimal gradient method may still be preferable. We will see this is the case for the support vector machine problem discussed in Chapter 5.

For three of the four problems, the unconstrained gradient method with the initial α in

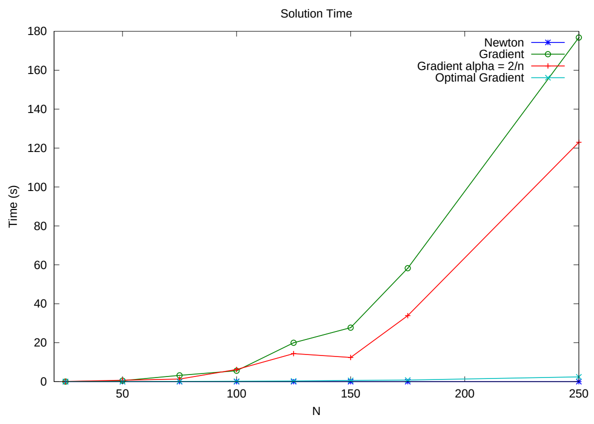

Figure 3.15: Solution time vs. problem size for Problem 4 with accuracy = 1e-5

Armijo's rule set to $2/N$ outperforms the unconstrained gradient method with the initial α set to 1. For all four problems, the optimal gradient method outperforms all of the other gradient methods as expected from theoretical analysis [52].

3.4 Summary

Since Cholesky factorization outperforms all of the direct and iterative routines when the condition number is large and the size of the systems are small, Cholesky factorization is used in the implementation. This is because when interfacing with AMPL, the systems will not become too large due to memory limitations within AMPL, and the cost of computing the condition number is prohibitive. Also the regularization methods described in section 3.2 can be performed during Cholesky factorization, making Cholesky factorization preferable.

For systems with fewer than 2.5 % non-zeros, the sparse matrix routine for computing the Cholesky factorization outperforms the dense matrix routine for computing the Cholesky factorization, regardless of matrix size. Thus, a sparse matrix factorization is used when a matrix has fewer than 2.5% non-zeros, and a dense matrix factorization is used when there are greater than 2.5% non-zeros.

When constraints are sparse, the PD EPM matrix can be reduced in size as described in section 3.1.3. Section 4.2.4 discusses how bounds on the variables, which are very sparse constraints, are folded over to reduce matrix size.

For all of the problems tested in section 3.3, Newton's method outperforms all of the gradient methods in terms of solution time. This is due to the small size of the problems when interfacing with AMPL. Thus, Newton's method is used for unconstrained minimization in the current implementation. We see in chapter 5 that gradient methods become useful for solving larger problems when not interfacing with AMPL.

The next chapter covers more details on the implemetation of the algorithm, provides numerical results for testing the algorithm on the CUTEr test set, and introduces an active-passive strategy for the general solver along with numerical results.

Chapter 4: Exterior-Point Algorithm and Numerical Results

This chapter discusses the exterior-point method algorithm that was implemented based on the design space exploration of chapter 3 and presents numerical results. Section 4.1 discusses the constrained nonlinear optimization algorithms used, including NRAL and PD EPM. Section 4.3 presents numerical results of testing the exterior-point algorithm on the CUTEr test set. Finally, section 4.4 describes an active-passive strategy applied to the exterior-point algorithm, similar to the active-passive strategy appled to the SVM algorithm in chapter 5.

4.1 Globally Convergent PD EPM

The constrained optimization algorithms implemented as part of the EPM solver include the NRAL technique, described in section 2.2.2, and the PD EPM, described in section 2.3.2. The PD EPM is only guaranteed to converge within a neighborhood of the solution, so a globally convergent method, similiar to the globally convergent PDNRD method [6,20], is implemented which uses both the NRAL technique and the PD EPM. An outline of this approach follows.

Consider solving the primal-dual system of equations that arises in the PD EPM where x, λ and ν are current iterates of the method

$$\begin{bmatrix} \nabla^2_{xx} L(x,\lambda,\nu) & -\nabla c^T(x) & -\nabla g^T(x) \\ -k\Lambda\Psi''(kc(x))\nabla c(x) & I_p & 0 \\ k\nabla g(x) & 0 & I_q \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda,\nu) \\ \bar{\lambda} - \lambda \\ -kg(x) \end{bmatrix}$$

and

$$\hat{x} = x + \Delta x \quad \hat{\lambda} = \lambda + \Delta\lambda \quad \hat{\nu} = \nu + \Delta\nu.$$

Let the merit function $r = v(x, \lambda, \nu)$ and $\hat{r} = v(\hat{x}, \hat{\lambda}, \hat{\nu})$. If $\hat{r}$ achieves a superlinear reduction, $\hat{r} \leq r^{\frac{3}{2}}$, then $\hat{x}, \hat{\lambda}, \hat{\nu}$, as defined above, are taken as the next iterates. Otherwise, only the primal variable $\hat{x}$ is updated as $\hat{x} = x + \alpha\Delta x$ and is used in a single iterate of the NRAL technique for fixed dual variables λ and ν. Then the dual variables are updated as in the NRAL technique and the process repeats.

A globally convergent PD EPM iterates the following steps:

1. Compute $\hat{x}, \hat{\lambda}, \hat{\nu}$ for the PD EPM

2. If $\hat{r} \leq \min\{r^{\frac{3}{2}-\theta}, \gamma r\}$ with $0 < \theta < \frac{1}{2}$ and $0 < \gamma < 1$:

    - Accept $\hat{x}, \hat{\lambda}, \hat{\nu}$

3. Else:

    - $\hat{x} = x + t\Delta x$ for t such that Armijo's rule is satisfied

    - Fix λ, ν and find $\hat{x} = \arg\min\limits_{x \in \mathbb{R}^n} \mathcal{L}_k(x, \lambda, \nu)$

    - Update $\hat{\lambda}_i = \lambda_i \psi'(kc_i(\hat{x})), i = 1, ..., p$ and
      $\hat{\nu_j} = \nu_j - kg_j(\hat{x}), j = 1, ..., q$.

4. Repeat

When the iterates are close enough to the solution, each iteration accepts both the primal and dual variable updates and the method will converge superlinearly. Algorithms 2 - 3 describe the algorithm in greater detail.

Algorithm 2 Exterior-Point Algorithm

```
1: Initialization
   An initial primal approximation $x^0 \in \mathbb{R}^m$ is given.
   An accuracy parameter $\epsilon > 0$ and the initial scaling parameter $k > 0$ are given.
   Parameters $0 < \gamma < 1$, $0 < \eta < .5$, $\beta > 1$, $\sigma > 0$, $\theta > 0$ are given.
   Set $x := x^0$, $\lambda^0 := (1, \dots, 1) \in \mathbb{R}^p$, $\nu_0 := 0$,
   $r := \mu\left(x, \lambda, \nu\right)$, $\lambda_g := \lambda^0$, $\nu_g = \nu^0$.
2: If $r \leq \epsilon$, stop. Output: $x$, $\lambda, \nu$.
3: Find direction: $(\Delta x, \Delta\lambda, \Delta\nu) :=$
     PrimalDualDirection $(x, \lambda, \nu)$.
   Set $\hat{x} := x + \Delta x$, $\hat{\lambda} := \lambda + \Delta\lambda$, $\hat{\nu} := \nu + \Delta\nu$.
4: If $\mu(\hat{x}, \hat{\lambda}, \hat{\nu}) \leq \min\left\{r^{\frac{3}{2}-\theta}, \gamma r\right\}$, set $x := \hat{x}$, $\lambda := \hat{\lambda}$, $\nu = \hat{\nu}$, $r := \mu(x, \lambda, \nu)$, $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$.
   Goto Step 2.
5: Set $t := 1$. Decrease $t := \frac{t}{2}$ until $\mathcal{L}_k\left(x + t\Delta x, \lambda_g, \nu_g\right)$
     $- \mathcal{L}_k\left(x, \lambda_g, \nu_g\right) \leq \eta t\left(\nabla \mathcal{L}_k\left(x, \lambda_g, \nu_g\right), \Delta x\right)$.
6: Set $\hat{\lambda} := \lambda_g \psi^{'}\left(kc\left(x + t\Delta x\right)\right)$, $\hat{\nu} := \nu_g - kg\left(x + t\Delta x\right)$,
   $x := x + t\Delta x$.
7: If $\|\nabla_x \mathcal{L}_k\left(x, \lambda_g, \nu_g\right)\| \leq \frac{\sigma}{k} \cdot \max\left\{\|\hat{\lambda} - \lambda_g\|, \|\hat{\nu} - \nu_g\|\right\}$, goto Step 9.
8: Find direction: $(\Delta x, \Delta\lambda, \Delta\nu) :=$
     PrimalDualDirection $(x, \lambda_g, \nu_g)$, goto Step 5.
9: If $\mu(x, \hat{\lambda}, \hat{\nu}) \leq \gamma r$, set $\lambda := \hat{\lambda}, \lambda_g := \hat{\lambda}, \nu := \hat{\nu}$,
   $\nu_g := \hat{\nu}, r := \mu\left(x, \lambda, \nu\right)$, $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$,
   goto Step 2.
10: Set $k := k\beta$, goto Step 8.
```

4.2 Notes on Algorithm Design

This section describes how the results from chapter 3 are incorporated into the exterior-point algorithm, and covers additional details about the algorithm and implementation.

4.2.1 Sparse or Dense Routines

At the beginning of the exterior-point algorithm execution, the AMPL routine sphes() is called which returns the number of non-zero elements in the Hessian of the objective function. AMPL also stores a variable, nzc, which holds the number of non-zeros in the Jacobian of the constraints. Using these two values and the number of constraints, the number of non-zeros in the PD EPM matrix can be computed.

Once the number of non-zero elements are computed, the percentage of non-zero elements can be determined using the matrix size. Using the result from section 3.1.2, sparse

Algorithm 3 Newton EPM Direction

```
function $(\Delta x, \Delta\lambda, \Delta\nu)$ :=*Primal Dual Direction* $(x, \lambda, \nu)$
begin
  $\bar{\lambda} := \psi^{'}\left(kc(x)\right)\lambda$
  Solve the quasidefinite system
  $$\begin{bmatrix} \nabla^2_{xx}L(x,\lambda,\nu) & -\nabla c^T(x) & -\nabla g^T(x) \\ -\nabla c(x) & \frac{1}{k}(\Lambda\Psi''(kc(x)))^{-1}I_p & 0 \\ -\nabla g(x) & 0 & -\frac{1}{k}I_q \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda,\nu) \\ \bar{\lambda}-\lambda \\ -kg(x) \end{bmatrix}$$
end
```

matrices and routines are used if the percentage of non-zero elements is less than 2.5% and dense matrices and routines are used otherwise.

Sparse matrices are constructed using compressed sparse column format, as described in section 3.1.2 and dense matrices are constructed as arrays in row-wise representation.

4.2.2 Solving Linear Systems

If dense matrices and routines are used, then LAPACK's LDL^T routine is used for computing the primal-dual direction of the quasi-definite matrix in step 3 of algorithm 2. In step 8 of algorithm 2, constraints are folded over and the system is reduced in size as described in section 3.1.3. In this way, the matrix is positive definite and only Δx is computed using Lapack's Cholesky factorization routine.

If sparse matrices and routines are used, then CHOLMOD's sparse LDL^T factorization routine is used for computing the primal-dual direction of the quasi-definite matrix in step 3 of algorithm 2. In step 8 of algorithm 2, if sparse matrices are used, CHOLMOD's sparse LDL^T factorization routine is used to solve the quasi-definite system. The reduced system is not used here because folding over the constraints is computationally inefficient with the sparse representation, and the reduced system can become dense even if the quasi-definite system is sparse.

4.2.3 Regularization

Regularization is implemented in LAPACK's Cholesky factorization routine using the GMW81 algorithm described in section 3.2.3. This ensures a decent direction for Newton minimization. The quasi-definite regularization method described in section 3.2.4 is implemented in LAPACK's LDL^T factorization routine and in CHOLMOD's sparse LDL^T factorization routine. This ensures that the matrices being factorized are sufficiently quasi-definite.

4.2.4 Folding Bounded Variables

If the optimization problem has bounded variables, then a lower bound of α on variable x_i corresponds to the constraint $c(x) = x_i - \alpha \geq 0$ and an upper bound of β on variable x_j corresponds to the constraint $c(x) = \beta - x_j \geq 0$. Thus, for a lower bound, the quasi-definite system used to compute the primal-dual direction contains the gradient $\nabla c^T(x) = (0, 0, ..., 1, ..., 0, 0)$ with the only non-zero value being 1 in the ith element of the vector. Similarly, for an upper bound, the quasi-definite system contains the gradient $\nabla c^T(x) = (0, 0, ..., -1, ..., 0, 0)$ with the only non-zero value being 1 in the jth element of the vector.

Since constraints corresponding to bounds on the variables are sparse, these constraints are folded over onto the diagonal of the Hessian, since the computational cost of folding over the constraint is low and it reduces the size of the system being solved. This is done similarly to reducing matrix size in section 3.1.3, but only the constraints corresponding to bounded variables are folded over.

4.3 EPM Numerical Results

I tested the EPM solver on the CUTEr test set of open source constrained and unconstrained optimization problems consisting of both linear and nonlinear problems [53]. I used the modeling language AMPL to provide test problems to the optimization solver [54]. AMPL can call an optimization solver and provide the solver with information necessary to solve an optimization problem. This information includes the number of variables and constraints, the objective and constraint function values, the gradient values and the Hessians.

The EPM solver was tested on 919 available optimization problems in the CUTEr test set. All tests were run on a 2.2 GHz Intel Pentium Dual Core desktop computer with 2 GB of RAM and a Windows XP operating system. These problems are available in AMPL format on LOQO's benchmarking website at www.princeton.edu/ rvdb/bench.html. Of these 919 problems, 80% of the problems were solved to an accuracy of 1e-6, meaning the merit function value was less than 1e-6 at the solution, and 82% of the problems were solved to an accuracy of 1e-3. It is worth noting that these results were obtained by running the EPM solver with the default parameters and without any tuning of the code. The results for running the EPM solver on the CUTEr test set is provided in Appendix A.

Since the exterior point method allows variables to leave the feasible, it is possible to get an exception in a function evaluation if a variable leaves the domain of the function. For example, an exception can be obtained if the operand of a square root or log evaluation becomes negative. Of the 919 CUTEr problems, 7% of these problems exited with exceptions to function evaluations.

These results can be compared to benchmarking results obtained for NITRO, SNOPT and LOQO at www.princeton.edu/ rvdb/bench.html. These benchmarks used the CUTEr test set and the results of the benchmarks are provided in Appendix A. These benchmarks did not report results for 32 of the problems that the EPM solver was tested on. Of the 887 problems that both the EPM solver was tested on and were reported for SNOPT, NITRO and LOQO, table 4.1 shows the percentage of problems solved by each code.

The unsolved problems for SNOPT, NITRO and LOQO were problems that timed out, reached the iteration limit, or contained an error. It is not clear from the benchmarking table to what accuracy these problems were solved. It is also stated that some tuning was done to obtain the results for each of these codes, unlike the EPM solver which was run with its default values.

From the results in table 4.1, we see that the EPM solver developed as part of this dissertation is competitive with both the SNOPT and NITRO codes in terms of number

Table 4.1: Benchmarking Results for Optimization Codes

Optimization Solver	% of CUTEr Problems Solved
SNOPT	82.9%
NITRO	79.3%
LOQO	95.7%
EPM solved to 1e-6	80.5 %
EPM solved to 1e-3	82.5 %

of problems solved. The LOQO code does much better, but it is unclear what tuning was done and what accuracy was achieved for the LOQO benchmarks.

Of the 889 problems reported in the benchmarks for SNOPT, NITRO and LOQO only 17 of the problems were unable to be solved by any of the codes. Out of these 17 problems, the EPM solver was able to solve 5 of these problems to 1e-3 accuracy and 4 of those 5 problems to 1e-6 accuracy. Of these 5 problems, 3 of the problems have quartic objective functions with no constraints. For problems without constraints, the method becomes just regularized Newton's method. This suggests that the regularization used with just Newton's method is implemented well. The other two problems have quadratic objectives with linear and cubic constraints. For these problems, the scaling parameter k does not need to be large for the EPM, so the presence of constraints does not cause ill-conditioning, which may occur with a large scaling parameter.

In general the EPM does well on convex problems, and can obtain high accuracy for these problems. One class of such problems are quadratic programming (QP) problems. A particular example of a QP problem is the one required for training support vector machines, which is investigated in detail in chapter 5. As with the other solvers, there are a number of problems that the EPM solver does not solve that the other solvers are able to. Many of these unsolved problems contain very ill-conditioned Hessian matrices with constraints. This suggests that more sophisiticated regularization methods may still be worth investigating. The EPM also does poorly on problems where iterations in the exterior of the feasible set causes challenges, such as encountering extremely large operands in exponentials or negative operands in square roots and logarithms.

It is not possible to compare execution times directly for the EPM solver tests and the benchmarks for SNOPT, NITRO and LOQO since the tests were performed on different machines. However, the execution times for the EPM solver are similar to the reported execution times for the benchmarked codes suggesting the EPM solver is competitive with these codes in terms of execution time.

4.4 EPM Active-Passive Strategy

4.4.1 Strategy for Primal Variables

If the optimization problem has bounded variables, an active-passive strategy can be applied to the EPM. Consider the following problem with inequality constraints, equality constraints and bounds on a subset of the variables

$$\text{(P1)} \qquad \begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & c_j(x) \geq 0, j = 1, ..., p \\ & g_i(x) = 0, i = 1, ..., q. \end{aligned}$$

If the optimization problem has bounded variables, then a lower bound of α on variable x_i corresponds to the constraint $c(x) = x_i - \alpha \geq 0$ and an upper bound of β on variable x_j corresponds to the constraint $c(x) = \beta - x_j \geq 0$.

Since, unlike interior point methods, the PD EPM allows iterates to leave the feasible set during an iteration, we can set $x_i = \alpha$ if x_i is bounded from below and $x_i < \alpha$ during an iteration or $x_i = \beta$ if x_i is bounded from above and $x_i > \beta$ during an iteration. Then the only primal variables which are active during an iteration are x_i such that x_i is bounded from below and $x_i > \alpha$, x_i is bounded from above and $x_i < \beta$ or x_i is not bounded. In this way, the PD system only needs to be solved for variables x_i which are active.

However, if a variable x_i leaves the active set, that variable needs to be brought back in to the active set if it left in error. This can be done by checking the value of the dual variable at the current value for x.

The value of the kth dual variable, $\underline{\lambda_k}$, for a lower bound constraint $x_k \geq \alpha$ can be calculated as

$$\underline{\lambda_k} = [\nabla f(x)]_k - \sum_{i=1}^{p} \hat{\lambda}_i [\nabla c_i(x)]_k - \sum_{j=1}^{q} \hat{\nu}_j [\nabla g_j(x)]_k$$

and for the uppper bound constraint, $x_l \leq \beta$ the value of the lth dual variable, $\overline{\lambda_l}$, can be calculated as

$$\overline{\lambda_l} = -[\nabla f(x)]_l + \sum_{i=1}^{p} \hat{\lambda}_i [\nabla c_i(x)]_l + \sum_{j=1}^{q} \hat{\nu}_j [\nabla g_j(x)]_l.$$

At the solution, the multipliers $\underline{\lambda_k}$ and $\overline{\lambda_l}$ must be nonnegative. Thus, if for some k, we have $x_k \leq \alpha$ and $\underline{\lambda_k} < 0$, then x_k is kept in the active set. Similarly, if we have $x_l \geq \beta$ and $\overline{\lambda_l} < 0$, then x_l is kept in the active set.

In this way, at each iteration, Δx is only computed for those variables which are in the active set. Recall, at each iteration of the PD EPM, the following system is solved

$$\begin{bmatrix} \nabla^2_{xx} L(x, \lambda, \nu) & -\nabla c^T(x) & -\nabla g^T(x) \\ -k\Lambda\Psi''(kc(x))\nabla c(x) & I_p & 0 \\ k\nabla g(x) & 0 & I_q \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x, \lambda, \nu) \\ \bar{\lambda} - \lambda \\ -kg(x) \end{bmatrix}.$$

Let $\mathcal{X}$ denote the set of variables x which are active during an iteration, and $\mathcal{I}$ denote the set of variables which are inactive. Then for $x \in \mathcal{I}$, we have $x = \alpha$ or $x = \beta$ and we have $\Delta x_{\mathcal{I}} = 0$. For $x \in \mathcal{X}$, the reduced system of equations is solved

$$\begin{bmatrix} \nabla^2_{xx} L(x,\lambda,\nu)_{\mathcal{X}\mathcal{X}} & -\nabla c^T(x)_{\mathcal{X}} & -\nabla g^T(x)_{\mathcal{X}} \\ -k\Lambda\Psi''(kc(x))\nabla c(x)_{\mathcal{X}} & I_p & 0 \\ k\nabla g(x)_{\mathcal{X}} & 0 & I_q \end{bmatrix} \begin{bmatrix} \Delta x_{\mathcal{X}} \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda,\nu)_{\mathcal{X}} \\ \bar{\lambda} - \lambda \\ -kg(x) \end{bmatrix}.$$

The size of the system is reduced from $n+p+q$ to $|\mathcal{X}|+p+q$, where $|\mathcal{X}| < n$ is the size of the set of active primal variables $\mathcal{X}$.

4.4.2 Strategy for Dual Variables

An active-passive strategy can also be employed for dual variables λ which correspond to inequality constraints. Again, the following problem with inequality constraints, equality constraints and bounds on a subset of the variables is solved

$$\text{(P1)} \qquad \begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & c_j(x) \geq 0, j = 1, ..., p \\ & g_i(x) = 0, i = 1, ..., q. \end{aligned}$$

If a constraint is not active at the solution, meaning we have strictly $c_j(x) > 0$, then from the KKT conditions, the corresponding dual variable is $\lambda = 0$.

At each iteration of the EPM, the value of the dual variable for the inequality constraints is checked. Since the PD EPM allows iterates to leave the feasible set during an iteration, if we have both $\lambda < 0$ and the constraint is satisfied $c(x) > 0$, we set $\lambda = 0$ for the next iteration and remove it from the set of active dual variables.

In this way, at each iteration, $\Delta\lambda$ is only computed for the set of active dual variables. Recall, at each iteration of the PD EPM, the following system is solved

$$\begin{bmatrix} \nabla^2_{xx}L(x,\lambda,\nu) & -\nabla c^T(x) & -\nabla g^T(x) \\ -k\Lambda\Psi''(kc(x))\nabla c(x) & I_p & 0 \\ k\nabla g(x) & 0 & I_q \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda,\nu) \\ \bar{\lambda}-\lambda \\ -kg(x) \end{bmatrix}.$$

Combining this with the strategy for primal variables, let $\mathcal{X}$ denote the set of active primal variables, $\mathcal{I}$ denote the set of inactive primal variables, $\mathcal{W}$ denote the set of active dual variables and $\mathcal{J}$ denote the set of inactive dual variables. Then only the reduced system of equations for the active primal and dual variables is solved

$$\begin{bmatrix} \nabla^2_{xx}L(x,\lambda,\nu)_{\mathcal{X}\mathcal{X}} & -\nabla c^T(x)_{\mathcal{X}\mathcal{W}} & -\nabla g^T(x)_{\mathcal{X}} \\ -k\Lambda\Psi''(kc(x))\nabla c(x)_{\mathcal{W}\mathcal{X}} & I_{\mathcal{W}} & 0 \\ k\nabla g(x)_{\mathcal{X}} & 0 & I_q \end{bmatrix} \begin{bmatrix} \Delta x_{\mathcal{X}} \\ \Delta\lambda_{\mathcal{W}} \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x,\lambda,\nu)_{\mathcal{X}} \\ \bar{\lambda}_{\mathcal{W}}-\lambda_{\mathcal{W}} \\ -kg(x) \end{bmatrix}.$$

Thus, the size of the system is further reduced from $n+p+q$ to $|\mathcal{X}|+|\mathcal{W}|+q$, where $|\mathcal{X}| < n$ is the size of the set of active primal variables $\mathcal{X}$ and $|\mathcal{W}| < p$ is the size of the set of active dual variables $\mathcal{W}$.

4.5 EPM Active-Passive Strategy Numerical Results

The active-passive strategies for both primal and dual variables were tested on the CUTEr test set of open source constrained and unconstrained optimization problems consisting of both linear and nonlinear problems [53]. All tests were run on a 2.2 GHz Intel Pentium Dual Core desktop computer with 2 GB of RAM and a Windows XP operating system.

Out of the 919 available problems the EPM solver was tested on with the active-passive strategies, 193 of the problems contained bounded variables with passive primal variables. Out of these 193 problems, the EPM solver with the active-passive strategy for primal variables was able to solve 54.9% of these problems to an accuracy of 1e-6. This is versus

66.3% of the 193 problems that are solved to an accuracy of 1e-6 without the primal active-passive strategy. Thus, the perfomance in terms of number of problems solved is not as good as without the active-passive strategy. However, looking at the total execution time for the 54.9% of problems that were solved with the EPM active-passive strategy and comparing those times to the total execution time of the EPM solver without the active-passive strategy, the decrease in total execution time is 73.4%.

Figure 4.1 displays the speedup for the EPM solver with the active-passive strategy for primal variables. Some problems do not see a speedup, but many more problems do. Problems with a recorded computation time of 0 both with and without the active-passive strategy are not shown. The results for running the EPM solver with the active-passive strategy for primal variables is provided in Appendix B.

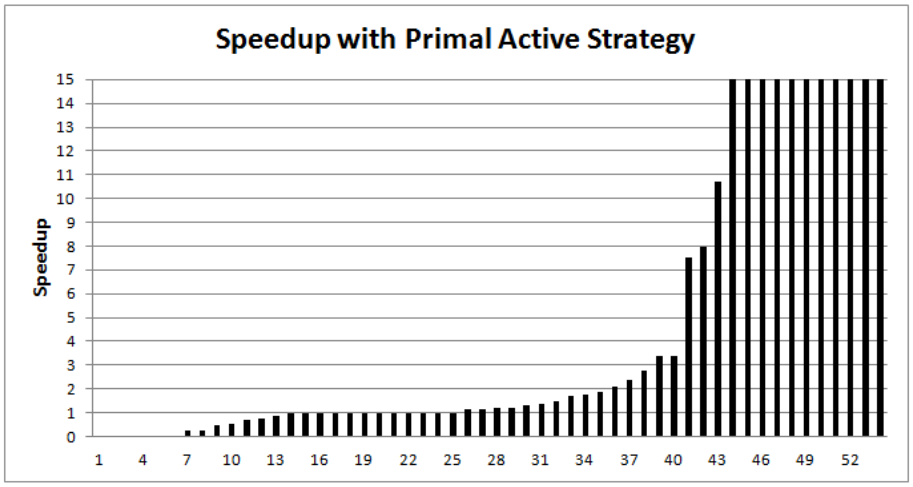

Figure 4.1: Speedup with active-passive strategy for primal variables

There were 156 of the 919 available problems in the CUTEr test set that contained passive dual variables, in order to test the active-passive strategy for dual variables. Out of these 156 problems, the EPM solver with the active-passive strategy for dual variables was able to solve 87.2% of these problems to an accuracy of 1e-6. This is versus 85.9% of the 156 problems that are solved to an accuracy of 1e-6 without the dual active-passive strategy. Thus, in terms of number of problems solved, the EPM with the dual active-passive strategy solves slightly more problems than the EPM without the dual active-passive strategy. Looking at the total execution time for all of the 87.2% of problems that were solved with the EPM active-passive strategy for dual variables and comparing those times to the total execution time of the EPM solver for these problems without the active-passive strategy, the decrease in total execution time is 60.2%.

Figure 4.2 displays the speedup for the EPM solver with the active-passive strategy for dual variables. Most problems see a speedup with the active-passive strategy for dual variables. Problems with a recorded computation time of 0 both with and without the active-passive strategy are not shown. The results for running the EPM solver with the active-passive strategy for dual variables is provided in Appendix B.

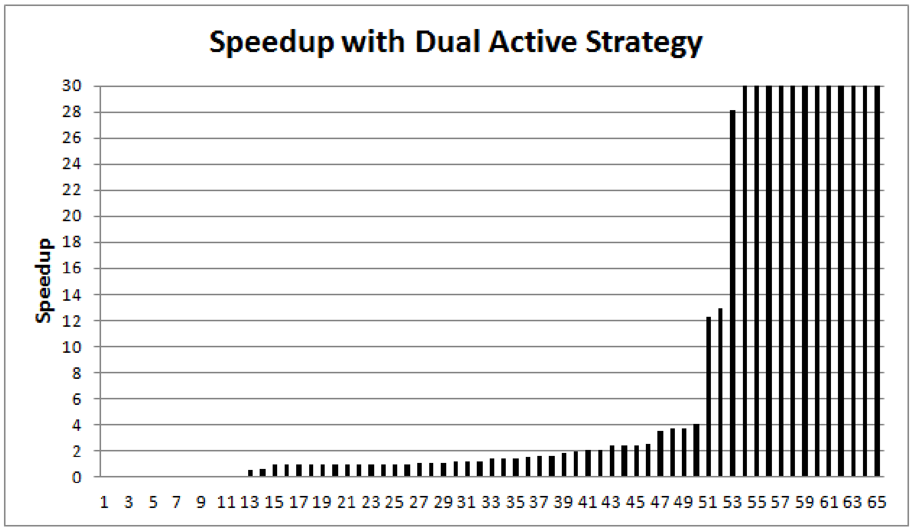

Figure 4.2: Speedup with active-passive strategy for dual variables

Table 4.2 displays these results for running the EPM solver on the CUTEr test set with the active-passive strategies for primal and dual variables.

Table 4.2: Active-Passive Strategies

Active-Passive Strategy	# Problems Tested	# Problems Solved	Decrease in Total Execution Time
Primal Variables	193	106	73.4%
Dual Variables	156	136	60.2%

4.6 Summary

In this chapter, the exterior-point algorithm that was implemented as part of this dissertation was discussed and numerical results presented. We also described active-passive strategies applied to the EPM for both primal and dual variables.

Numerical results for the EPM show the implemented algorithm to be competitive with existing solvers in terms of number of problems solved. Also, the implemented EPM algorithm was found to be able to solve some problems that none of the existing solvers, SNOPT, NITRO or LOQO, were able to solve.

Numerical results for the EPM with active-passive strategies show that the active-passive strategies for both primal and dual variables resulted in a decrease in total execution time on the sets of problems tested and solved by each strategy. Figures 4.1 and 4.2 show speedup with the active-passive strategies. For the dual active-passive strategy, the computation time for most of the problems was the same or decreased. In the next chapter, an active-passive strategy for primal variables is applied to the support vector machine problem.

Chapter 5: Support Vector Machines

Machine learning algorithms are popularly used for gaining insight into data. Today, data is accumulating at tremendous rates, motivating the need for machine learning algorithms that can solve large-scale big data problems. One such machine learning approach is the support vector machine (SVM). The SVM problem is a convex quadratic programming (QP) problem, which is a class of problems that are easily solved by the EPM to high accuracy. Despite advances in SVM algorithms, large-scale SVMs motivate development of new training algorithms. This chapter investigates the use and improvement of exterior-point methods for training large-scale SVMs. Section 5.1 discusses the formulation of the SVM problem, section 5.2 discusses application of the exterior-point algorithm to the SVM problem, section 5.3 discusses using properties of the EPM to speed up training the SVM, and sections 5.5 and 5.5.2 discuss using fast gradient methods to speed up training large-scale SVMs.

5.1 SVM Formulation

Support vector machine techniques are used for classifying m training points $x_i \in \mathbb{R}^n$ into 2 classes where a separating hyperplane with the largest margin is used to separate the classes. The training points are classified by specifying a scalar y_i such that if x_i is in a specific class, then $y_i = 1$, otherwise $y_i = -1$. The separating hyperplane takes the form $y = w^T x + b$ such that points with $y_i = 1$ lie on one side of the hyperplane, so that $w^T x_i + b > 0$, and points with $y_i = -1$ lie on the other side of the hyperplane, so that $w^T x_i + b < 0$.

The separation margin is $2/\|w\|$ and w should be found such that this separation margin is maximized. Maximizing this margin is equivalent to minimizing $w^T w$.

If the training data is not separable, a vector $\xi \in \mathbb{R}^m$ can be introduced to allow each training point, x_i, to violate the margin by as much as $\xi_i \geq 0$. The primal form of the SVM problem is

$$\text{minimize } \frac{1}{2}w^T w + C\sum_{i=1}^{m} \xi_i$$

$$\text{subject to } y_i(x_i^T w + b) \geq 1 - \xi_i, i = 1, ..., m$$

$$\xi_i \geq 0, i = 1, ..., m.$$

The parameter C penalizes separation errors. The larger C is, the more separation errors $\sum_{i=1}^{m} \xi_i$ penalize the minimization problem.

If $\alpha \in \mathbb{R}^m$ and $\eta \in \mathbb{R}^m$ are dual variables corresponding to the constraints $y_i(w^T x + b) \geq 1 - \xi_i, i = 1, ..., m$ and $\xi_i \geq 0, i = 1, ..., m$ respectively, then the Lagrangian of the SVM problem takes the form

$$L(w, b, \xi, \alpha, \eta) = \frac{1}{2}w^T w + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i(y_i(x_i^T w + b) - 1 + \xi_i) - \sum_{i=1}^{m} \eta_i \xi_i.$$

Let $e \in \mathbb{R}^m$ be a vector of ones. Set the gradient of the Lagrangian to zero so that

$$\nabla_w L = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0$$

$$\nabla_b L = \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\nabla_\xi L = Ce - \alpha - \eta = 0.$$

Using the above expressions and maximizing the Lagrangian, the dual SVM problem becomes

$$\begin{aligned} \text{maximize} \ & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{subject to} \ & \sum_{i=1}^{m} \alpha_i y_i = 0 \\ & 0 \le \alpha_i \le C, i = 1, ..., m. \end{aligned}$$

From the gradient of the Lagrangian, the relationship between the primal variables w and dual variables α is $w = \sum_{i=1}^{m} \alpha_i y_i x_i$. If $\alpha_j < C$ for some $j \in 1...m$, then from $\alpha_j + \eta_j = C$ we have $\eta_j \neq 0$. From complementary slackness this implies $\xi_j = 0$, so $y_j(x_j^T w + b) \geq 1$. If $\alpha_j > 0$ then this constraint is active. Thus, for α_j such that $0 < \alpha_j < C$, we have $y_j(x_j^T w + b) = 1$.

Once $\alpha \in \mathbb{R}^m$ is computed for the dual problem, w can be computed as $w = \sum_{i=1}^{m} \alpha_i y_i x_i$ and b can be computed for α_j such that $0 < \alpha_j < C$ as $b = -w^T x_j + y_j = -\sum_{i=1}^{m} \alpha_i y_i x_i^T x_j + y_j$. If multiple α_j's satisfy $0 < \alpha_j < C$, multiple b's can be computed and averaged to find the most accurate value for b.

A new point x_t can be classified by computing

$$y_t = w^T x_t + b = \sum_{i=1}^{m} \alpha_i y_i x_i^T x_t + b.$$

If $y_t > 0$, x_t is classified as being in the same class as the training points with $y = 1$, otherwise it is put into the same class as the training points with $y = -1$.

5.1.1 Kernels

Each data vector x can be mapped into a higher dimensional feature vector $\Phi(x)$ with a separating hyperplane taking the form $w^T\Phi(x)+b$. Using Φ in the dual problem and finding $\alpha \in \mathbb{R}^m$ results in $w^T\Phi(x_j) + b = \sum_{i=1}^{m} \alpha_i y_i \Phi(x_i)^T \Phi(x_j) + b$.

A kernel function $K(x, z) = \Phi(x)^T\Phi(z)$ can be used to compute the inner products $\Phi(x)^T\Phi(z)$ without explicitly forming Φ. A commonly used kernel function is the radial basis kernel $K(x, z) = e^{-\gamma\|x-z\|^2}$.

5.2 PD EPM for SVM

Note: Portions of this section will appear in publication. [1]

For the dual suppport vector machine problem, the maximization problem is changed a minimization problem by negating the objective function.

$$\min_{\alpha\in\mathbb{R}^m} \text{imize } -\sum_{i=1}^{m}\alpha_i + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j K(x_i, x_j)$$

$$\text{subject to } \sum_{i=1}^{m}\alpha_i y_i = 0$$

$$0 \le \alpha_i \le C, i = 1, ..., m$$

For each iteration of the PD EPM method we solve the system

$$\begin{bmatrix} \nabla^2_{\alpha\alpha} L(\alpha,\lambda,\nu) & -\nabla c^T(\alpha) & -\nabla g^T(\alpha) \\ -k\Lambda\Psi''(kc(\alpha))\nabla c(\alpha) & I_p & 0 \\ k\nabla g(\alpha) & 0 & I_q \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta\lambda \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_\alpha L(\alpha,\lambda,\nu) \\ \bar{\lambda}-\lambda \\ -kg(\alpha) \end{bmatrix}$$

where $\Psi''(kc(\alpha)) = \text{diag}(\psi''(kc_i(\alpha)))_{i=1}^p$, $\Lambda = \text{diag}(\lambda_i)_{i=1}^p$, $\bar{\lambda} = \Psi'(kc(\alpha))\lambda$ and I_p and I_q are $p \times p$ and $q \times q$ identity matrices.

Then the primal and dual variables are updated as

$$\hat{\alpha} = \alpha + \Delta\alpha \quad \hat{\lambda} = \lambda + \Delta\lambda \quad \hat{\nu} = \nu + \Delta\nu.$$

If the primal-dual direction $(\Delta\alpha, \Delta\lambda, \Delta\nu)$ does not produce a superlinear reduction of the merit function, then the iterates λ and ν are not updated and the direction $\Delta\alpha$ is used in the unconstrained minimization of the augmented Lagrangian for the equivalent rescaled problem

$$\mathcal{L}_k(\alpha,\lambda,\nu) = f(\alpha) - k^{-1}\sum_{i=1}^{p}\lambda_i\psi(kc_i(\alpha)) - \sum_{j=1}^{q}\nu_j g_j(\alpha) + \frac{k}{2}\sum_{j=1}^{q} g_j(\alpha)^2.$$

The size of the primal-dual system for the support vector machine probem is $3m+1$. Using the sparsity of the second row of the primal-dual system

$$-k\Lambda\Psi''(kc(\alpha))\nabla c(\alpha)\Delta\alpha + \Delta\lambda = \bar{\lambda} - \lambda$$

we can reduce the size of the system to $m+1$. After expressing $\Delta\lambda$ from this equation and inserting into the first row of the PD EPM system, we have

$$\nabla^2_{\alpha\alpha}L(\alpha,\lambda,\nu)\Delta\alpha + \nabla c^T(\alpha)\lambda - k\nabla c^T(\alpha)\Lambda\Psi''(kc(\alpha))\nabla c(\alpha)\Delta\alpha - \nabla c^T(\alpha)\bar{\lambda} - \nabla g^T(\alpha)\Delta\nu$$

$$= -\nabla f(\alpha) + \nabla c(\alpha)\lambda + \nabla g(\alpha)\nu$$

which is equivalent to

$$[\nabla^2_{\alpha\alpha} L(\alpha,\lambda,\nu) - k\nabla c^T(\alpha)\Lambda\Psi''(kc(\alpha))\nabla c(\alpha)]\Delta\alpha - \nabla g^T(\alpha)\Delta\nu = -\nabla_\alpha L(\alpha,\lambda,\nu).$$

Setting $M(\alpha,\lambda,\nu) = \nabla^2_{\alpha\alpha} L(\alpha,\lambda,\nu) - k\nabla c^T(\alpha)\Lambda\Psi''(kc(\alpha))\nabla c(\alpha)$, the PD EPM system is reduced to

$$\begin{bmatrix} M(\alpha,\lambda,\nu) & -\nabla g^T(\alpha) \\ k\nabla g(\alpha) & 1 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_\alpha L(\alpha,\bar{\lambda},\nu) \\ -kg(\alpha) \end{bmatrix}$$

or equivalently the quasidefinite system

$$\begin{bmatrix} M(\alpha,\lambda,\nu) & -\nabla g^T(\alpha) \\ -\nabla g(\alpha) & -\frac{1}{k} \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_\alpha L(\alpha,\bar{\lambda},\nu) \\ g(\alpha) \end{bmatrix}.$$

Algorithms 4-5 outline the method for solving the SVM problem using the PD EPM.

5.2.1 Numerical Results

This algorithm was tested on data sets from the UC Irvine Machine Learning Repository [55]. The data sets selected from the repository consisted of hundreds to a few thousand instances with attributes in numerical form and no missing attributes. The selected data sets consisted of two or more classes. For classifications with more than two classes, one class is arbitrarily labeled as positive and all remaining classes as negative.

All test were conducted on an Intel Core 2 Duo Processor P8700 (3-MB cache, 2.53 GHz) laptop computer with 4 GB of RAM and a Windows 7 operating system. The EPM QP solver is implemented in C++ and uses a radial basis kernel $K(x,z) = e^{-\gamma\|x-z\|^2}$ with $\gamma = 0.0521$. Therefore, all linear systems solved by the EPM algorithm are completely dense. The penalization parameter C is selected as $C = 100$. The EPM solver uses a stopping criteria of $\mu(z) \leq 10^{-6}$.

Algorithm 4 Exterior-Point Algorithm

```
1: Initialization
   An initial primal approximation $\alpha^0 \in \mathbb{R}^m$ is given.
   An accuracy parameter $\epsilon > 0$ and the initial scaling parameter $k > 0$ are given.
   Parameters $0 < \gamma < 1$, $0 < \eta < .5$, $\beta > 1$, $\sigma > 0$, $\theta > 0$ are given.
   Set $\alpha := \alpha^0$, $\lambda^0 := (1, \dots, 1) \in \mathbb{R}^p$, $\nu_0 := 0$,
   $r := \mu(\alpha, \lambda, \nu)$, $\lambda_g := \lambda^0$, $\nu_g = \nu^0$.
2: If $r \le \epsilon$, stop. Output: $\alpha$, $\lambda, \nu$.
3: Find direction: $(\Delta\alpha, \Delta\lambda, \Delta\nu) :=$
     PrimalDualDirection $(\alpha, \lambda, \nu)$.
   Set $\hat{\alpha} := \alpha + \Delta\alpha$, $\hat{\lambda} := \lambda + \Delta\lambda$, $\hat{\nu} := \nu + \Delta\nu$.
4: If $\mu(\hat{\alpha}, \hat{\lambda}, \hat{\nu}) \le \min\left\{r^{\frac{3}{2}-\theta}, \gamma r\right\}$, set $\alpha := \hat{\alpha}$, $\lambda := \hat{\lambda}$, $\nu = \hat{\nu}$, $r := \mu(\alpha, \lambda, \nu)$, $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$.
   Goto Step 2.
5: Set $t := 1$. Decrease $t := \frac{t}{2}$ until $\mathcal{L}_k(\alpha + t\Delta\alpha, \lambda_g, \nu_g)$
     $- \mathcal{L}_k(\alpha, \lambda_g, \nu_g) \le \eta t (\nabla\mathcal{L}_k(\alpha, \lambda_g, \nu_g), \Delta\alpha)$.
6: Set $\hat{\lambda} := \lambda_g \psi'(kc(\alpha + t\Delta\alpha))$, $\hat{\nu} := \nu_g - kg(\alpha + t\Delta\alpha)$,
   $\alpha := \alpha + t\Delta\alpha$.
7: If $\|\nabla_\alpha \mathcal{L}_k(\alpha, \lambda_g, \nu_g)\| \le \frac{\sigma}{k} \cdot \max\left\{\|\hat{\lambda} - \lambda_g\|, \|\hat{\nu} - \nu_g\|\right\}$, goto Step 9.
8: Find direction: $(\Delta\alpha, \Delta\lambda, \Delta\nu) :=$
     PrimalDualDirection $(\alpha, \lambda_g, \nu_g)$, goto Step 5.
9: If $\mu(\alpha, \hat{\lambda}, \hat{\nu}) \le \gamma r$, set $\lambda := \hat{\lambda}, \lambda_g := \hat{\lambda}, \nu := \hat{\nu}$,
   $\nu_g := \hat{\nu}, r := \mu(\alpha, \lambda, \nu)$, $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$,
   goto Step 2.
10: Set $k := k\beta$, goto Step 8.
```

Table 5.1 displays the the number of training instances and attributes, the number of EPM iterations and the solution time (in wall time seconds) to train the SVM. Since this SVM implementation does use not decomposition methods nor other acceleration techniques, the EPM is storing a matrix for the radial kernel function and solving a dense linear system of equations during each iteration.

These results indicate that the EPM is a feasible training algorithm for SVMs. In the next section we describe an active-passive strategy based on the EPM for speeding up the training time by reducing the size of the dense linear system solved at each iteration.

5.3 SVM Active-Passive Strategy

When the number of support vectors for the SVM problem is small compared to the number of training vectors, we can take advantage of the smaller number of support vectors by

Algorithm 5 Newton EPM Direction

```
function $(\Delta\alpha, \Delta\lambda, \Delta\nu)$ :=*Primal Dual Direction* $(\alpha, \lambda, \nu)$
begin
    $\bar{\lambda} := \psi^{'}(kc(\alpha))\,\lambda$
    Solve the quasidefinite system
    $\begin{bmatrix} M(\alpha,\lambda,\nu) & -\nabla g(\alpha) \\ -\nabla g^T(\alpha) & -\frac{1}{k} \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(\alpha,\bar{\lambda},\nu) \\ g(\alpha) \end{bmatrix}$
    $\Delta\lambda := \bar{\lambda} - \lambda + k\Lambda\Psi^{''}(kc(\alpha))\,\nabla c^T(\alpha)\Delta\alpha$
end
```

Table 5.1: SVM Numerical Results with PD EPM [1]

Data Set Name	Instances	Attributes	Iterations	Solution Time (s)
Arcene	100	10000	5	2.31
Balance Scale [56]	625	4	53	5.22
Breast Cancer Wisconsin (Diagnostic)	569	30	5	1.31
CNAE-9	1080	856	28	28.98
Contrapositive Method Choice	1473	9	91	130.06
Dexter [56]	300	20000	5	27.71
Haberman's Survival	306	3	42	0.81
Madelon [56]	2000	500	5	30.65
Page Block Classification	5473	10	47	2673.63
Pen-Based Recognition of Handwritten Digits	3498	16	6	459.09
Seeds	210	7	87	1.11
Statlog (Landsat Satellite)	4435	36	5	709.25

solving a smaller system of equations at each iteration.

Unlike interior point methods, EPMs allow variables to leave the feasible region during an iteration. We can heuristically set all such variables to the boundary of the feasible region, since at the solution all variables are contained within the feasible region. Thus, if during an iteration we have a variable $\alpha_i < 0$ we can set $\alpha_i = 0$ or if $\alpha_i < C$ we can set $\alpha_i = C$. We say these variables are passive for that iteration. We define an active variable of an iteration as α_i such that $0 < \alpha_i < C$ and the active set as the set of all active variables for an iteration. In this way, our active-passive strategy only solves the PD system of equations for the active variables at each iteration.

The active-passive strategy employs a heuristic that assumes that at the solution the passive variables will be on the boundary of the feasible region, but they may actually be within the feasible region at the solution. Thus, at each iteration the active-passive strategy checks if a passive variable should be moved to the active set as follows.

At iteration k, calculate the value of the ith dual variable, $\underline{\lambda_i}$, corresponding to the lower bound constraint $\alpha_i \geq 0$ as

$$\underline{\lambda_i} = ([\nabla^2_{\alpha\alpha} f(\alpha)]\alpha)_i - 1 + \nu y_i$$

and for the uppper bound constraint, $\alpha_i \leq C$, calculate the value of the ith dual variable, $\overline{\lambda_i}$, as

$$\overline{\lambda_i} = -([\nabla^2_{\alpha\alpha} f(\alpha)]\alpha)_i + 1 - \nu y_i.$$

At the solution, the multipliers $\underline{\lambda_i}$ and $\overline{\lambda_i}$ must be positive. Thus, if for some i, $\alpha_i \leq 0$ and $\underline{\lambda_i} < 0$, then move α_i to the active set from the passive set. Similarly, if $\alpha_i \geq C$ and $\overline{\lambda_i} < 0$, then move α_i to the active set. Also, each iteration must contain at least one vector from each class. Thus, if at some iteration, the active set only contains variables corresponding to the class with $y = 1$, then a variable corresponding to the class with $y = -1$ is moved from the passive set to the active set.

Each iteration of the PD EPM solves the quasi-definite system

$$\begin{bmatrix} M(\alpha,\lambda,\nu) & -\nabla g^T(\alpha) \\ -\nabla g(\alpha) & -\frac{1}{k} \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_\alpha L(\alpha,\bar{\lambda},\nu) \\ g(\alpha) \end{bmatrix}.$$

The active-passive strategy only computes $\Delta\alpha_i$ for variables contained in the active set. Let $\mathcal{A}$ denote the set of indices corresponding to the set of variables α_i that are active during an iteration, and $\mathcal{I}$ denote the set of indices corresponding to the set of variables that are passive. For $i \in \mathcal{I}$, let $\Delta\alpha_i = 0$. For the active variables, solve the reduced system of equations

$$\begin{bmatrix} M(\alpha,\lambda,\nu)_{\mathcal{A}\mathcal{A}} & -\nabla g^T(\alpha)_{\mathcal{A}} \\ -\nabla g(\alpha)_{\mathcal{A}} & -\frac{1}{k} \end{bmatrix} \begin{bmatrix} \Delta\alpha_{\mathcal{A}} \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_\alpha L(\alpha,\bar{\lambda},\nu)_{\mathcal{A}} \\ g(\alpha) \end{bmatrix}.$$

The size of the system is reduced from $m+1$ to $|\mathcal{A}|+1$, where $|\mathcal{A}| \leq m$ is the size of the active set $\mathcal{A}$.

Algorithms 6-8 outline the method for solving the SVM problem using the PD EPM with our active-passive strategy.

5.4 SVM Active-Passive Strategy - Small Active Set

When the number of support vectors for the SVM problem is small compared to the number of training vectors, we can further improve the performance of the EPM active-passive strategy by starting the routine with a small number of active variables, instead of setting all variables as active. For this method, choose $2 \leq p \leq m$ as the initial size of the set of active variables. At each iteration, algorithm 10 ensures at most p variables are contained in the active set, and increases p by Δp until all variables have been considered for the active set. This method only solves systems of the size $|\mathcal{A}|+1$, which may never be as large as $m+1$. In contrast, our strategy in the previous section always starts with a system of size $m+1$. Algorithms 9-10 describe this method for an active-passive strategy starting

Algorithm 6 Exterior-Point Algorithm with Active-Passive Strategy

```
1: Initialization
   An initial primal approximation $\alpha^0 \in \mathbb{R}^m$ is given.
   An accuracy parameter $\epsilon > 0$ and the initial scaling parameter $k > 0$ are given.
   Parameters $0 < \gamma < 1$, $0 < \eta < .5$, $\beta > 1$, $\sigma > 0$, $\theta > 0$ are given.
   Set $\alpha := \alpha^0$, $\lambda^0 := (1, \dots, 1) \in \mathbb{R}^p$, $\nu_0 := 0$,
   $r := \mu(\alpha, \lambda, \nu)$, $\lambda_g := \lambda^0$, $\nu_g = \nu^0$.
2: If $r \leq \epsilon$, stop. Output: $\alpha$, $\lambda, \nu$.
3: Find active set and set inactive variables: $(\mathcal{A}, \alpha_{\mathcal{I}}, \hat{\lambda}) :=$ ActiveCompute $(\alpha, \lambda, \nu)$.
4: Find direction: $(\Delta\alpha_{\mathcal{A}}, \Delta\lambda, \Delta\nu) :=$
     ReducedPrimalDualDirection $(\alpha_{\mathcal{A}}, \lambda, \nu)$.
   Set $\hat{\alpha}_{\mathcal{A}} := \alpha_{\mathcal{A}} + \Delta\alpha_{\mathcal{A}}$, $\hat{\lambda} := \lambda + \Delta\lambda$, $\hat{\nu} := \nu + \Delta\nu$.
5: If $\mu(\hat{\alpha}, \hat{\lambda}, \hat{\nu}) \leq \min\left\{r^{\frac{3}{2}-\theta}, \gamma r\right\}$, set $\alpha := \hat{\alpha}$, $\lambda := \hat{\lambda}$, $\nu = \hat{\nu}$, $r := \mu(\alpha, \lambda, \nu)$, $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$.
   Goto Step 2.
6: Set $t := 1$. Decrease $t := \frac{t}{2}$ until $\mathcal{L}_k(\alpha_{\mathcal{A}} + t\Delta\alpha_{\mathcal{A}}, \lambda_g, \nu_g)$
     $- \mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g) \leq \eta t\left(\nabla\mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g), \Delta\alpha_{\mathcal{A}}\right)$.
7: Set $\hat{\lambda} := \lambda_g \psi'(kc(\alpha + t\Delta\alpha))$, $\hat{\nu} := \nu_g - kg(\alpha + t\Delta\alpha)$
   $\alpha_{\mathcal{A}} := \alpha_{\mathcal{A}} + t\Delta\alpha_{\mathcal{A}}$.
8: If $\|\nabla_\alpha \mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g)\| \leq \frac{\sigma}{k} \cdot \max\left\{\|\hat{\lambda} - \lambda_g\|, \|\hat{\nu} - \nu_g\|\right\}$, goto Step 10.
9: Find direction: $(\Delta\alpha_{\mathcal{A}}, \Delta\lambda, \Delta\nu) :=$
     ReducedPrimalDualDirection $(\alpha_{\mathcal{A}}, \lambda_g, \nu_g)$, goto Step 6.
10: If $\mu(\alpha, \hat{\lambda}, \hat{\nu}) \leq \gamma r$, set $\lambda := \hat{\lambda}, \lambda_g := \hat{\lambda}, \nu := \hat{\nu}$,
    $\nu_g := \hat{\nu}, r := \mu(\alpha, \lambda, \nu)$, $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$,
    goto Step 2.
11: Set $k := k\beta$, goto Step 9.
```

with a small active set.

5.4.1 Numerical Results

Again, we tested the algorithm on data sets from the UC Irvine Machine Learning Repository [55] using data sets consiting of hundreds to a few thousand instances. All tests for the active-passive strategy were perfomed on an Intel Core i7-4770k (8-MB cache, 3.5 GHz) desktop computer with 16 GB of RAM and a Windows 7 operating system.

For testing the active-passive strategies, we use a radial basis kernel $K(x, z) = e^{-\gamma\|x-z\|^2}$ with $\gamma = .0521$ and $\gamma = 10^{-4}$. Using a smaller parameter for γ results in fewer active variables at the solution which better demonstrates the effectiveness of the strategy. All linear systems are still completely dense, but the systems will become smaller as the method

Algorithm 7 Reduced Newton EPM Direction

```
function $(\Delta\alpha_{\mathcal{A}}, \Delta\lambda, \Delta\nu)$ :=*Reduced Primal Dual Direction* $(\alpha_{\mathcal{A}}, \lambda, \nu)$
begin
    $\bar{\lambda} := \psi'(kc(\alpha))\,\lambda$
    Solve the quasidefinite system
    $\begin{bmatrix} M(\alpha,\lambda,\nu)_{\mathcal{A}\mathcal{A}} & -\nabla g^T(\alpha)_{\mathcal{A}} \\ -\nabla g(\alpha)_{\mathcal{A}} & -\frac{1}{k} \end{bmatrix} \begin{bmatrix} \Delta\alpha_{\mathcal{A}} \\ \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -\nabla_x L(\alpha,\bar{\lambda},\nu)_{\mathcal{A}} \\ \\ g(\alpha) \end{bmatrix}$
    $\Delta\lambda := \bar{\lambda} - \lambda + k\Lambda\Psi''(kc(\alpha))\,\nabla c^T(\alpha)\Delta\alpha$
end
```

progresses. The penalization parameter C is selected as $C = 100$ and the stopping criteria used is $\mu(z) \leq 10^{-6}$.

Table 5.2 displays results for training the SVM without using the active-passive strategy, with $\gamma = .0521$ and table 5.3 displays results for training the SVM without using the active-passive strategy, with $\gamma = 10^{-4}$.

Tables 5.4 and 5.5 displays results for training the SVM with the active-passive strategy, with $\gamma = .0521$ and $\gamma = 10^{-4}$ respectively. Both tables display the number of training instances and attributes, the number of EPM iterations and the solution time (in wall time seconds) to train the SVM. Tables 5.6 and 5.7 show results for training the SVM with and without the active-passive strategy for $\gamma = .0521$ and $\gamma = 10^{-4}$ respectively, along with the percentage of variables which are active at the solution, and the speedup with the active-passive strategy.

These results show that the active-passive strategy results in a speedup in computation time for most problems where there are fewer active variables at the solution. For problems with many fewer active variables at the solution, the speedup is significant. For example, with $\gamma = 10^{-4}$, the *Handwritten Digits* problem and the *Statlog* problem have 2% and 9.4% variables active at the solution respectively, and a speedup of 6.5 and 4.8 respectively.

Tables 5.8 and 5.9 display results for training the SVM with the active-passive strategy starting with fewer active variables, with $\gamma = .0521$ and $\gamma = 10^{-4}$ respectively. Both tables display the number of training instances and attributes, the number of EPM iterations and

Algorithm 8 Compute Active Set

```
function $(\mathcal{A}, \alpha_{\mathcal{I}}, \hat{\lambda})$ :=*ActiveCompute* $(\alpha, \lambda, \nu)$

For $i$ from 1 to $m$
    If $0 < \alpha_i < C$ then $i \in \mathcal{A}$
    Else if $\alpha_i < 0$ then $\alpha_i = 0$ and $i \in \mathcal{I}$
    Else if $\alpha_i > C$ then $\alpha_i = C$ and $i \in \mathcal{I}$
End for

If $y_i = 1$ for all $i \in \mathcal{A}$ then $i \in \mathcal{A}$ such that $y_i = -1$
If $y_i = -1$ for all $i \in \mathcal{A}$ then $i \in \mathcal{A}$ such that $y_i = 1$

For $i$ in $\mathcal{I}$:
    $(\lambda_i)$ :=*Compute Multiplier*$(i, \alpha, \nu)$
    If $\lambda_i < 0$ then $i \in \mathcal{A}$ and $i \notin \mathcal{I}$
    Else Set $\hat{\lambda}_i$ :=*Compute Multiplier*$(i, \alpha, \nu)$
End for
end

function $(\mathcal{A}, \alpha_{\mathcal{I}})$ :=*Compute Multiplier* $(i, \alpha, \nu)$
If $\alpha_i = 0$ then
    $\lambda_i = ([\nabla^2_{\alpha\alpha} f(\alpha)]\alpha)_i - 1 + \nu y_i$
Else if $\alpha_i = C$ then
    $\lambda_i = -([\nabla^2_{\alpha\alpha} f(\alpha)]\alpha)_i + 1 - \nu y_i$
end
```

the solution time (in wall time seconds) to train the SVM.

Tables 5.10 and 5.11 show results for training the SVM with and without the active-passive strategy starting with fewer active variables, along with the percentage of variables which are active at the solution, and the speedup with the active-passive strategy starting with fewer variables.

For $\gamma = .0521$, most problems show a speedup with the active-passive strategy starting with fewer variables. The problems that do not see a speedup either have all variables active at the solution, or a high percentage of variables active at the solution.

For $\gamma = 10^{-4}$, the larger problems with fewer numbers of support vectors see the most improvement from the active-passive strategy starting with fewer number of active variables. In particular the *Page Block Classification* problem with 5473 variables and 26% active variables sees a speedup of 2.7, and the *Statlog* problem with 4435 variables and 9% active variables sees a speedup of 1.8. And most notably, the *Recognition of Handwritten Digits* problem with 3498 variables and 2% active variables sees a speedup of 99.2.

Algorithm 9 Exterior-Point Algorithm with Active-Passive Strategy and Small Active Set

```
1: Initialization
   An initial primal approximation $\alpha^0 \in \mathbb{R}^m$ is given.
   An accuracy parameter $\epsilon > 0$ and the initial scaling parameter $k > 0$ are given.
   A parameter $fg\_switch > 0$ used for switching between fast gradient and Newton methods for minimization
   A parameter $p$ for the initial size of the active set and $\Delta p$ for increasing the size of the active set
   Parameters $0 < \gamma < 1$, $0 < \eta < .5$, $\beta > 1$, $\sigma > 0$, $\theta > 0$ are given.
   Set $\alpha := \alpha^0$, $\lambda^0 := (1, \ldots, 1) \in \mathbb{R}^p$, $\nu_0 := 0$,
   $r := \mu(\alpha, \lambda, \nu)$, $\lambda_g := \lambda^0$, $\nu_g = \nu^0$.
2: If $r \le \epsilon$, stop. Output: $\alpha$, $\lambda, \nu$.
3: Find active set and set inactive variables: $(\mathcal{A}, \alpha_{\mathcal{I}}, \hat{\lambda}) :=$ ActiveCompute $(\alpha, \lambda, \nu, p)$.
   Set $p = p + \Delta p$
4: Find direction: $(\Delta\alpha_{\mathcal{A}}, \Delta\lambda, \Delta\nu) :=$
      PrimalDualDirection $(\alpha_{\mathcal{A}}, \lambda, \nu)$.
   Set $\hat{\alpha}_{\mathcal{A}} := \alpha_{\mathcal{A}} + \Delta\alpha_{\mathcal{A}}$, $\hat{\lambda} := \lambda + \Delta\lambda$, $\hat{\nu} := \nu + \Delta\nu$.
5: If $\mu(\hat{\alpha}, \hat{\lambda}, \hat{\nu}) \le \min\left\{r^{\frac{3}{2}-\theta}, \gamma r\right\}$, set $\alpha := \hat{\alpha}$, $\lambda := \hat{\lambda}$, $\nu = \hat{\nu}$, $r := \mu(\alpha, \lambda, \nu)$.
   If $p < m$ then set $k := k_0$ else $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$.
   Goto Step 2.
6: Set $t := 1$. Decrease $t := \frac{t}{2}$ until $\mathcal{L}_k(\alpha_{\mathcal{A}} + t\Delta\alpha_{\mathcal{A}}, \lambda_g, \nu_g)$
      $- \mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g) \le \eta t\left(\nabla\mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g), \Delta\alpha_{\mathcal{A}}\right)$.
7: Set $\hat{\lambda} := \lambda_g \psi'(kc(\alpha + t\Delta\alpha))$, $\hat{\nu} := \nu_g - kg(\alpha + t\Delta\alpha)$
   $\alpha_{\mathcal{A}} := \alpha_{\mathcal{A}} + t\Delta\alpha_{\mathcal{A}}$.
8: If $\|\nabla_\alpha \mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g)\| \le \frac{\sigma}{k} \cdot \max\left\{\|\hat{\lambda} - \lambda_g\|, \|\hat{\nu} - \nu_g\|\right\}$, goto Step 10.
9: Find direction: $(\Delta\alpha, \Delta\lambda, \Delta\nu) :=$
      PrimalDualDirection $(\alpha_{\mathcal{A}}, \lambda_g, \nu_g)$, goto Step 6.
10: If $\mu(\alpha, \hat{\lambda}, \hat{\nu}) \le \gamma r$, set $\lambda := \hat{\lambda}, \lambda_g := \hat{\lambda}, \nu := \hat{\nu}$,
    $\nu_g := \hat{\nu}, r := \mu(\alpha, \lambda, \nu)$.
    If $p < m$ then set $k := k_0$ else $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$,
    Goto Step 2.
11: Set $k := k\beta$, goto Step 9.
```

5.5 SVM Fast Gradient Methods

When the number of active variables is large, we can use the fast gradient method to solve the unconstrained minimization of the augmented Lagrangian for the equivalent problem within the exterior-point algorithm.

Recall from section 2.4.3, the fast gradient method starts with α_0 and y_1 and with

Algorithm 10 Compute Active Set

```
function $(\mathcal{A}, \alpha_{\mathcal{I}}, \hat{\lambda})$ :=*ActiveCompute* $(\alpha, \lambda, \nu, p)$

For $i$ from 1 to $m$
    If $|\mathcal{A}| > p$ then break
    If $0 < \alpha_i < C$ then $i \in \mathcal{A}$
    Else if $\alpha_i < 0$ then $\alpha_i = 0$ and $i \in \mathcal{I}$
    Else if $\alpha_i > C$ then $\alpha_i = C$ and $i \in \mathcal{I}$
End for

If $y_i = 1$ for all $i \in \mathcal{A}$ then $i \in \mathcal{A}$ such that $y_i = -1$
If $y_i = -1$ for all $i \in \mathcal{A}$ then $i \in \mathcal{A}$ such that $y_i = 1$

For $i$ in $\mathcal{I}$:
    If $|\mathcal{A}| > p$ then break
    $(\lambda_i)$ :=Compute Multiplier$(i, \alpha, \nu)$
    If $\lambda_i < 0$ then $i \in \mathcal{A}$ and $i \notin \mathcal{I}$
End for

For $i \notin \mathcal{I}$:
    Set $\hat{\lambda}_i$ :=Compute Multiplier$(i, \alpha, \nu)$
End for

end

function $(\mathcal{A}, \alpha_{\mathcal{I}})$ :=Compute Multiplier $(i, \alpha, \nu)$
If $\alpha_i = 0$ then
    $\lambda_i = ([\nabla^2_{\alpha\alpha} f(\alpha)]\alpha)_i - 1 + \nu y_i$
Else if $\alpha_i = C$ then
    $\lambda_i = -([\nabla^2_{\alpha\alpha} f(\alpha)]\alpha)_i + 1 - \nu y_i$
end
```

$\beta_0 = 1$. One iteration of the fast gradient method is

$$\alpha_i = y_i - \frac{1}{L}\nabla f(y_i)$$

$$\beta_{i+1} = (1 + \sqrt{4\beta_i^2 + 1})/2$$

$$y_{i+1} = \alpha_i + \frac{\beta_i - 1}{\beta_{i+1}}(\alpha_i - \alpha_{i-1})$$

where L is the Lipschitz constant for $\nabla f(\alpha)$.

Table 5.2: SVM Numerical Results with PD EPM and $\gamma = .0521$

Data Set Name	Instances	Attributes	Iterations	Solution Time (s)
Arcene	100	10000	5	0.39
Balance Scale [56]	625	4	53	2.26
Breast Cancer Wisconsin (Diagnostic)	569	30	5	0.86
CNAE-9	1080	856	28	7.31
Contrapositive Method Choice	1473	9	91	49.94
Dexter [56]	300	20000	5	3.14
Haberman's Survival	306	3	42	0.26
Madelon [56]	2000	500	5	3.31
Page Block Classification	5473	10	46	1723.6
Pen-Based Recognition of Handwritten Digits	3498	16	6	411.32
Seeds	210	7	87	0.21
Statlog (Landsat Satellite)	4435	36	5	568.26

5.5.1 NRAL Fast Gradient Method

For the SVM problem we wish to minimize the augumentd Lagrangian for the equivalent problem

$$\mathcal{L}_k(\alpha,\lambda,\nu) = f(\alpha) - k^{-1}\sum_{i=1}^{m}((\lambda_l)_i\Psi(k\alpha_i) + (\lambda_u)_i\Psi(k(C-\alpha_i))) - \nu g(\alpha) + \frac{k}{2}g(\alpha)^2$$

with gradient

$$\nabla_\alpha\mathcal{L}_k(\alpha,\lambda,\nu) = \nabla f(\alpha) - \sum_{i=1}^{m}((\lambda_l)_i\Psi'(k\alpha_i) - (\lambda_u)_i\Psi'(k(C-\alpha_i)))e_i - (\nu - kg(\alpha))y.$$

Table 5.3: SVM Numerical Results with PD EPM and $\gamma = 10^{-4}$

Data Set Name	Instances	Attributes	Iterations	Solution Time (s)
Arcene	100	10000	5	.33
Balance Scale [56]	625	4	113	4.02
Breast Cancer Wisconsin (Diagnostic)	569	30	54	1.55
CNAE-9	1080	856	147	24.73
Contrapositive Method Choice	1473	9	506	197.88
Dexter [56]	300	20000	5	2.62
Haberman's Survival	306	3	137	0.7
Madelon [56]	2000	500	5	9.32
Page Block Classification	5473	10	259	5760.97
Pen-Based Recognition of Handwritten Digits	3498	16	77	486.03
Seeds	210	7	344	.73
Statlog (Landsat Satellite)	4435	36	88	1070.53

Estimating the Lipschitz Constant

The Lipshitz constant for the gradient $\nabla_\alpha \mathcal{L}_k(\alpha, \lambda, \nu)$ is related to the norm of the Hessian, $\nabla^2_{\alpha\alpha}\mathcal{L}_k(\alpha, \lambda, \nu)$ which is the maximum eigenvalue of the Hessian for the L_2 norm. The Hessian of the augmented Lagrangian for the equivalent problem is

$$\nabla^2_{\alpha\alpha}\mathcal{L}_k(\alpha, \lambda, \nu) = \nabla^2 f(\alpha) - k\sum_{i=1}^{m}((\lambda_l)_i\Psi''(k\alpha_i) + (\lambda_u)_i\Psi''(k(C-\alpha_i)))e_ie_i^T + kyy^T$$

where we use the fact that $\nabla g(\alpha) = y$.

Now, define 3 matrices, A, B, and C such that $\nabla^2_{\alpha\alpha}\mathcal{L}_k(\alpha, \lambda, \nu) = A + B + C$,

Table 5.4: SVM Numerical Results with PD EPM Active-Passive Strategy and $\gamma = .0521$

Data Set Name	Instances	Attributes	Iterations	Solution Time (s)
Arcene	100	10000	5	0.74
Balance Scale [56]	625	4	57	0.36
Breast Cancer Wisconsin (Diagnostic)	569	30	5	0.76
CNAE-9	1080	856	34	3.43
Contrapositive Method Choice	1473	9	99	33.52
Dexter [56]	300	20000	5	2.6
Haberman's Survival	306	3	41	0.6
Madelon [56]	2000	500	5	2.9
Page Block Classification	5473	10	52	1457.8
Pen-Based Recognition of Handwritten Digits	3498	16	7	389.84
Seeds	210	7	62	0.07
Statlog (Landsat Satellite)	4435	36	5	522.04

$$A = \nabla^2 f(\alpha)$$

$$B = -k \sum_{i=1}^{m} ((\lambda_l)_i \Psi''(k\alpha_i) + (\lambda_u)_i \Psi''(k(C-\alpha_i)))e_i e_i^T$$

$$C = kyy^T.$$

The triangle inequality gives us $||A+B+C||_2 \leq ||A||_2 + ||B||_2 + ||C||_2$. Let $\gamma_{max}(A)$ denote the maximum eigenvalue of matrix A. Then, since $||A||_2 = \gamma_{max}(A)$, this means that $\gamma_{max}(A+B+C) \leq \gamma_{max}(A) + \gamma_{max}(B) + \gamma_{max}(C)$.

For matrix $A = \nabla^2 f(\alpha)$ we have an element i, j of matrix A as $A_{ij} = y_i y_j e^{-\gamma||x_i - x_j||^2}$.

Table 5.5: SVM Numerical Results with PD EPM Active-Passive Strategy and $\gamma = 10^{-4}$

Data Set Name	Instances	Attributes	Iterations	Solution Time (s)
Arcene	100	10000	5	.34
Balance Scale [56]	625	4	108	2.45
Breast Cancer Wisconsin (Diagnostic)	569	30	45	.59
CNAE-9	1080	856	219	31.34
Contrapositive Method Choice	1473	9	521	148.96
Dexter [56]	300	20000	5	2.71
Haberman's Survival	306	3	162	0.69
Madelon [56]	2000	500	5	8.83
Page Block Classification	5473	10	325	5344.72
Pen-Based Recognition of Handwritten Digits	3498	16	46	74.81
Seeds	210	7	463	.79
Statlog (Landsat Satellite)	4435	36	97	221.91

Thus, for an off-diagonal element of A, we have $-1 \leq A_{ij} \leq 1$, and for a diagonal element $0 \leq A_{ij} \leq 1$. Since the trace of A is equal to the sum of the eigenvalues, and A is positive semi-definite, we have $\gamma_{max}(A) \leq tr(A) \leq m$.

For matrix $B = -k\sum_{i=1}^{m}((\lambda_l)_i\Psi''(k\alpha_i) + (\lambda_u)_i\Psi''(k(C-\alpha_i)))e_ie_i^T$ the off diagonal elements of B are $B_{ij} = 0$, and the diagonal elements of B are $B_{ii} = -k((\lambda_l)_i\psi''(k\alpha_i) + (\lambda_u)_i\psi''(k(C-\alpha_i)))$. Also, the multipliers $(\lambda_l)_i \geq 0$ and $(\lambda_u)_i \geq 0$ and from the properties of ψ in section 2.2.2 we have $\psi''(\cdot) < 0$. Thus, each element $B_{ii} \geq 0$. Using the log rescaling function defined in section 2.2.2, we have

$$\psi''(t) = \begin{cases} -\frac{1}{(t+1)^2} & \text{if } t > -.5 \\ -4 & \text{if } t \leq -.5. \end{cases}$$

Table 5.6: SVM Numerical Results with and without PD EPM Active-Passive Strategy and $\gamma = .0521$

Data Set Name	Solution Time (s) w/out APS	Solution Time (s) with APS	% Active	Speedup
Arcene	0.39	0.74	100	0.53
Balance Scale [56]	2.26	0.36	8	6.27
Breast Cancer Wisconsin (Diagnostic)	0.86	0.76	100	1.13
CNAE-9	7.31	3.43	15.28	2.13
Contrapositive Method Choice	49.94	33.52	60.08	1.49
Dexter [56]	3.14	2.6	100	1.21
Haberman's Survival	0.26	0.6	52.61	0.43
Madelon [56]	3.31	2.9	100	1.14
Page Block Classification	1723.6	1457.8	96	1.18
Pen-Based Recognition of Handwritten Digits	411.32	389.84	100	1.05
Seeds	0.21	0.07	16.67	3.00
Statlog (Landsat Satellite)	568.26	522.04	100	1.08

Thus, $\psi''(t) \geq -4$. Let the maximum multiplier value for upper and lower bounds be $\lambda_{max} = \max_i\{(\lambda_l)_i, (\lambda_u)_i\}$. Then, since $\gamma_{max}(B) = \max_i\{B_{ii}\}$, we have $\gamma_{max}(B) < 8k\lambda_{max}$.

Finally, the matrix $C = kyy^T$ has $m-1$ eigenvalues that are 0, and a single eigenvalue of km. Thus, $\gamma_{max}(C) = km$.

So, the Lipschitz constant is bounded from above by

$$L = \gamma_{max}(A+B+C) \leq \gamma_{max}(A) + \gamma_{max}(B) + \gamma_{max}(C)$$

$$\leq m + 8k\lambda_{max} + km.$$

Table 5.7: SVM Numerical Results with and without PD EPM Active-Passive Strategy and $\gamma = 10^{-4}$

Data Set Name	Solution Time (s) w/out APS	Solution Time (s) with APS	% Active	Speedup
Arcene	.33	.34	100	0.97
Balance Scale [56]	4.02	2.45	41.44	1.64
Breast Cancer WI Wisconsin (Diagnostic)	1.55	.59	26.01	2.63
CNAE-9	24.73	31.34	19.16	0.79
Contrapositive Method Choice	197.88	148.96	72.09	1.33
Dexter [56]	2.62	2.71	100	0.97
Haberman's Survival	0.7	0.69	53.59	1.01
Madelon [56]	9.32	8.83	100	1.06
Page Block Classification	5760.97	5344.72	26.69	1.07
Pen-Based Recognition Handwritten Digits	486.03	74.81	2.08	6.50
Seeds	.73	.79	60.95	0.92
Statlog (Landsat Satellite)	1070.53	221.91	9.40	4.82

To determine a lower bound on the Lipschitz constant, we use Weyl's monotonicity theorem [57]. Since the matrix $A + B$ is positive semi-definite, we have that $\gamma_{max}(C) \leq \gamma_{max}(A+B+C)$. Since $\gamma_{max}(C) = km$, the Lipschitz constant is bounded above and below by

$$km \leq L \leq m + 8k\lambda_{max} + km.$$

This bound is used to approximate the Lipschitz constant in the SVM code. Since the fast gradient method converges to x_* as $f(x_i) - f(x_*) \leq \frac{4L\|x_0 - x_*\|_2^2}{(i+2)^2}$, the fast gradient method performs better for the support vector machine problem when the multiplier k is

Table 5.8: SVM Numerical Results with PD EPM Active-Passive Strategy Starting with Small Active Set and $\gamma = .0521$

Data Set Name	Instances	Attributes	Iterations	Solution Time (s)
Arcene	100	10000	21	0.79
Balance Scale [56]	625	4	165	1.19
Breast Cancer Wisconsin (Diagnostic)	569	30	23	0.81
CNAE-9	1080	856	55	6.24
Contrapositive Method Choice	1473	9	285	44.52
Dexter [56]	300	20000	15	2.57
Haberman's Survival	306	3	186	0.3
Madelon [56]	2000	500	18	11.02
Page Block Classification	5473	10	38	952.39
Pen-Based Recognition of Handwritten Digits	3498	16	7	319.58
Seeds	210	7	212	0.21
Statlog (Landsat Satellite)	4435	36	25	419.87

small.

Fast Gradient Method with Active-Passive Strategy

If the fast gradient method is used along with the active-passive strategy, the gradient is only computed for variables which are in the active set. In this way, an iteration i of the fast gradient method for the support vector machine problem is

Table 5.9: SVM Numerical Results with PD EPM Active-Passive Strategy Starting with Small Active Set and $\gamma = 10^{-4}$

Data Set Name	Instances	Attributes	Iterations	Solution Time (s)
Arcene	100	10000	5	0.33
Balance Scale [56]	625	4	303	18.52
Breast Cancer Wisconsin (Diagnostic)	569	30	12	1.12
CNAE-9	1080	856	81	35.11
Contrapositive Method Choice	1473	9	418	234.2
Dexter [56]	300	20000	5	8.39
Haberman's Survival	306	3	226	3.75
Madelon [56]	2000	500	5	14.49
Page Block Classification	5473	10	162	2105.97
Pen-Based Recognition of Handwritten Digits	3498	16	51	4.9
Seeds	210	7	268	0.51
Statlog (Landsat Satellite)	4435	36	45	600.19

$$\alpha_{\mathcal{A},i} = y_{\mathcal{A},i} - \frac{1}{L}\nabla_y \mathcal{L}_k(y_{\mathcal{A},i}, \lambda, \nu)$$

$$\beta_{i+1} = (1 + \sqrt{4\beta_i^2 + 1})/2$$

$$y_{\mathcal{A},i+1} = \alpha_{\mathcal{A},i} + \frac{\beta_i - 1}{\beta_{i+1}}(\alpha_{\mathcal{A},i} - \alpha_{\mathcal{A},i-1})$$

where $L \approx k * |\mathcal{A}|$ is the Lipschitz constant for $\nabla_\alpha \mathcal{L}_k(\alpha_{\mathcal{A},i}, \lambda, \nu)$.

Algorithms 11-12 outline the method for solving the SVM problem using the PD EPM with an active-passive strategy and the fast gradient method.

Table 5.10: SVM Numerical Results with and without PD EPM Active-Passive Strategy Starting with Small Active Set and $\gamma = .0521$

Data Set Name	Solution Time (s) w/out APS	Solution Time (s) with APS	% Active	Speedup
Arcene	0.39	0.79	100	0.49
Balance Scale [56]	2.26	1.19	8	1.90
Breast Cancer Wisconsin (Diagnostic)	0.86	0.81	100	1.06
CNAE-9	7.31	6.24	15.28	1.17
Contrapositive Method Choice	49.94	44.52	60.08	1.12
Dexter [56]	3.14	2.57	100	1.22
Haberman's Survival	0.26	0.3	52.61	0.87
Madelon [56]	3.31	11.02	100	0.30
Page Block Classification	1723.6	952.39	96	1.81
Pen-Based Recognition of Handwritten Digits	411.32	319.58	100	1.29
Seeds	0.21	0.21	16.67	1
Statlog (Landsat Satellite)	568.26	419.87	100	1.35

5.5.2 Projected Fast Gradient Method

A projected fast gradient method for the SVM problem may improve convergence over the fast gradient method. In the projected fast gradient method, the augmented Lagrangian is used to satisfy the equality constraint, and the primal variables are projected onto the bounds to satisfy the bounds on the variables. Then the ith iteration of the projected fast gradient method is

Table 5.11: SVM Numerical Results with and without PD EPM Active-Passive Strategy Starting with Small Active Set and $\gamma = 10^{-4}$

Data Set Name	Solution Time (s) w/out APS	Solution Time (s) with APS	% Active	Speedup
Arcene	0.33	0.33	100	1.00
Balance Scale [56]	4.02	18.52	41.44	0.22
Breast Cancer Wisconsin (Diagnostic)	1.55	1.12	26.01	1.38
CNAE-9	24.73	35.11	19.17	0.70
Contrapositive Method Choice	197.88	234.2	72.1	0.84
Dexter [56]	2.62	8.39	100	0.31
Haberman's Survival	0.7	3.75	53.59	0.19
Madelon [56]	9.32	14.49	100	0.64
Page Block Classification	5760.97	2105.97	26.69	2.74
Pen-Based Recognition of Handwritten Digits	486.03	4.9	2.09	99.2
Seeds	0.73	0.51	60.95	0.31
Statlog (Landsat Satellite)	1070.53	600.19	9.4	1.78

$$\alpha_i = y_i - \frac{1}{L}\nabla f(y_i)$$

If $\alpha_{ji} <= 0$, set $\alpha_{ji} := 0$

If $\alpha_{ji} >= C$, set $\alpha_{ji} := C$

$$\beta_{i+1} = (1 + \sqrt{4\beta_i^2 + 1})/2$$

$$y_{i+1} = \alpha_i + \frac{\beta_i - 1}{\beta_{i+1}}(\alpha_i - \alpha_{i-1})$$

Algorithm 11 Exterior-Point Algorithm with Active-Passive Strategy and Fast Gradient

```
1:  Initialization
    An initial primal approximation $\alpha^0 \in \mathbb{R}^m$ is given.
    An accuracy parameter $\epsilon > 0$ and the initial scaling parameter $k > 0$ are given.
    A parameter $fg\_switch > 0$ used for switching between fast gradient and Newton methods for minimization
    Parameters $0 < \gamma < 1$, $0 < \eta < .5$, $\beta > 1$, $\sigma > 0$, $\theta > 0$ are given.
    Set $\alpha := \alpha^0$, $\lambda^0 := (1, \dots, 1) \in \mathbb{R}^p$, $\nu_0 := 0$,
    $r := \mu(\alpha, \lambda, \nu)$, $\lambda_g := \lambda^0$, $\nu_g = \nu^0$.
2:  If $r \le \epsilon$, stop. Output: $\alpha$, $\lambda, \nu$.
3:  Find active set and set inactive variables: $(\mathcal{A}, \alpha_{\mathcal{I}}, \hat{\lambda}) :=$ ActiveCompute $(\alpha, \lambda, \nu)$.
4:  If $|\mathcal{A}| < fg\_switch$ goto Step 6
5:  Unconstrained minimization with fast gradient: $\left(\alpha_{\mathcal{A}}, \hat{\lambda}, \hat{\nu}\right)$ :=FastGradient $(\alpha_{\mathcal{A}}, \lambda, \nu)$. Goto Step 12.
6:  Find direction: $(\Delta\alpha_{\mathcal{A}}, \Delta\lambda, \Delta\nu) :=$
        ReducedPrimalDualDirection $(\alpha_{\mathcal{A}}, \lambda, \nu)$.
    Set $\hat{\alpha}_{\mathcal{A}} := \alpha_{\mathcal{A}} + \Delta\alpha_{\mathcal{A}}$, $\hat{\lambda} := \lambda + \Delta\lambda$, $\hat{\nu} := \nu + \Delta\nu$.
7:  If $\mu(\hat{\alpha}, \hat{\lambda}, \hat{\nu}) \le \min\left\{r^{\frac{3}{2}-\theta}, \gamma r\right\}$, set $\alpha := \hat{\alpha}$, $\lambda := \hat{\lambda}$, $\nu = \hat{\nu}$, $r := \mu(\alpha, \lambda, \nu)$, $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$.
    Goto Step 2.
8:  Set $t := 1$. Decrease $t := \frac{t}{2}$ until $\mathcal{L}_k(\alpha_{\mathcal{A}} + t\Delta\alpha_{\mathcal{A}}, \lambda_g, \nu_g)$
        $- \mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g) \le \eta t\left(\nabla\mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g), \Delta\alpha_{\mathcal{A}}\right)$.
9:  Set $\hat{\lambda} := \lambda_g \psi'(kc(\alpha + t\Delta\alpha))$, $\hat{\nu} := \nu_g - kg(\alpha + t\Delta\alpha)$,
    $\alpha_{\mathcal{A}} := \alpha_{\mathcal{A}} + t\Delta\alpha_{\mathcal{A}}$.
10: If $\|\nabla_\alpha \mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g)\| \le \frac{\sigma}{k} \cdot \max\left\{\|\hat{\lambda} - \lambda_g\|, \|\hat{\nu} - \nu_g\|\right\}$, goto Step 12.
11: Find direction: $(\Delta\alpha_{\mathcal{A}}, \Delta\lambda, \Delta\nu) :=$
        ReducedPrimalDualDirection $(\alpha_{\mathcal{A}}, \lambda_g, \nu_g)$, goto Step 8.
12: If $\mu(\alpha, \hat{\lambda}, \hat{\nu}) \le \gamma r$, set $\lambda := \hat{\lambda}, \lambda_g := \hat{\lambda}, \nu := \hat{\nu}$,
    $\nu_g := \hat{\nu}, r := \mu(\alpha, \lambda, \nu)$, $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$,
    goto Step 2.
13: Set $k := k\beta$. If $|\mathcal{A}| < fg\_switch$ goto Step 11, else goto Step 5.
```

where L is the Lipschitz constant for $\nabla f(\alpha)$, and j is the jth element of the array.

For the projected fast gradient method applied to the SVM problem we wish to minimize the augumented Lagrangian,

$$\mathcal{L}_k(\alpha, \nu) = f(\alpha) - \nu g(\alpha) + \frac{k}{2} g(\alpha)^2$$

with gradient

$$\nabla_\alpha \mathcal{L}_k(\alpha, \nu) = \nabla f(\alpha) - (\nu - kg(\alpha))y.$$

Algorithm 12 Fast Gradient Minimization

```
    function $\left(\alpha_{\mathcal{A}}, \hat{\lambda}, \hat{\nu}\right)$ :=FastGradient $(\alpha_{\mathcal{A}}, \lambda, \nu)$
1: Initialize $\beta = 1.0$, $\hat{\alpha}_{\mathcal{A}} = \alpha_{\mathcal{A}}$
2: $\alpha_{\mathcal{A}} = y_{\mathcal{A}} - \frac{1}{L}\nabla_y \mathcal{L}_k(y_{\mathcal{A}}, \lambda, \nu)$
   $\beta = (1 + \sqrt{4\beta^2 + 1})/2$
   $y_{\mathcal{A}} = \alpha_{\mathcal{A}} + \frac{\beta - 1}{\beta}(\alpha_{\mathcal{A}} - \hat{\alpha}_{\mathcal{A}})$
   $\hat{\alpha}_{\mathcal{A}} = \alpha_{\mathcal{A}}$
3: Set $\hat{\lambda} := \lambda_g \psi^{'}\left(kc\left(\alpha_{\mathcal{A},i}\right)\right)$
   $\hat{\nu} := \nu_g - kg\left(\alpha_{\mathcal{A},i}\right)$.
4: If $\|\nabla_y \mathcal{L}_k\left(y_{\mathcal{A}}, \lambda_g, \nu_g\right)\| \leq \frac{\sigma}{k} \cdot \max\left\{\|\hat{\lambda} - \lambda_g\|, \|\hat{\nu} - \nu_g\|\right\}$ then break else goto Step 2.
```

The Lipschitz constant for the projected fast gradient method can be approximated from the maximum eigenvalue of the Hessian of the augmented Lagrangian. The Hessian of the augmented Lagrangian is

$$\nabla^2_{\alpha\alpha}\mathcal{L}_k(\alpha, \lambda, \nu) = \nabla^2 f(\alpha) + kyy^T.$$

Similar to section 5.5.1, the bounds on the Lipschitz constant are

$$km \leq L \leq m + km$$

which is used to approximate the Lipschitz constant in the SVM code. Since the fast gradient method converges to x_* as $f(x_i) - f(x_*) \leq \frac{4L\|x_0 - x_*\|_2^2}{(i+2)^2}$, the fast gradient method performs better for the support vector machine problem when the multiplier k is small.

If the projected fast gradient method is used along with the active-passive strategy, the gradient is only computed for variables which are in the active set. Then the ith iteration of the projected fast gradient method for the SVM problem is

$$\alpha_{\mathcal{A},i} = y_{\mathcal{A},i} - \frac{1}{L}\nabla_y \mathcal{L}_k(y_{\mathcal{A},i}, \lambda, \nu)$$

$$\text{If } \alpha_{ji} <= 0, \text{ set } \alpha_{ji} := 0$$

$$\text{If } \alpha_{ji} >= C, \text{ set } \alpha_{ji} := C$$

$$\beta_{i+1} = (1 + \sqrt{4\beta_i^2 + 1})/2$$

$$y_{\mathcal{A},i+1} = \alpha_{\mathcal{A},i} + \frac{\beta_i - 1}{\beta_{i+1}}(\alpha_{\mathcal{A},i} - \alpha_{\mathcal{A},i-1})$$

where $L \approx k * |\mathcal{A}|$ is the Lipschitz constant for $\nabla_\alpha \mathcal{L}_k(\alpha_{\mathcal{A},i}, \nu)$.

Algorithms 13-14 outline the method for solving the SVM problem using the PD EPM with an active-passive strategy and the projected fast gradient method.

5.5.3 Fast Gradient Methods Numerical Results

Again, we tested the fast gradient methods on data sets from the UC Irvine Machine Learning Repository [55] using data sets consiting of hundreds to a few thousand instances. All tests for the fast gradient methods were conducted on an Intel Core i7-Q720 (6-MB cache, 1.6 GHz) laptop computer with 8 GB of RAM and a Linux 2.6.35.14 operating system.

A radial basis kernel $K(x, z) = e^{-\gamma\|x-z\|^2}$ was used with $\gamma = .0521$. The penalization parameter C is selected as $C = 100$ and the stopping criteria used is $\mu(z) \le 10^{-6}$.

The gradient method is used when the number of active variables is greater than 3000. Only three of the problems used from the UC Irvine Machine Learning Repository contained greater than 3000 variables. Two additional larger problems are tested with the projected fast gradient method, with 14,980 and 19,020 variables respectively. Table 5.12 displays results for the EPM without accelerations, with $\gamma = .0521$. Table 5.13 displays results for the NRAL fast gradient method, used to minimize the augmented Lagrangian for the

Algorithm 13 Exterior-Point Algorithm with Active-Passive Strategy and Projected Fast Gradient

```
1: **Initialization**
   An initial primal approximation $\alpha^0 \in \mathbb{R}^m$ is given.
   An accuracy parameter $\epsilon > 0$ and the initial scaling parameter $k > 0$ are given.
   A parameter $fg\_switch > 0$ used for switching between fast gradient and Newton methods for minimization
   Parameters $0 < \gamma < 1$, $0 < \eta < .5$, $\beta > 1$, $\sigma > 0$, $\theta > 0$ are given.
   Set $\alpha := \alpha^0$, $\lambda^0 := (1, \dots, 1) \in \mathbb{R}^p$, $\nu_0 := 0$,
   $r := \mu(\alpha, \lambda, \nu)$, $\lambda_g := \lambda^0$, $\nu_g = \nu^0$.
2: If $r \le \epsilon$, stop. **Output:** $\alpha$, $\lambda, \nu$.
3: Find active set and set inactive variables: $(\mathcal{A}, \alpha_{\mathcal{I}}, \hat{\lambda}) :=$ *ActiveCompute* $(\alpha, \lambda, \nu)$.
4: If $|\mathcal{A}| < fg\_switch$ goto Step 6
5: Unconstrained minimization with fast gradient: $(\alpha_{\mathcal{A}}, \hat{\nu}) :=$*ProjectedFastGradient* $(\alpha_{\mathcal{A}}, \nu, r)$.
   Goto Step 12.
6: Find direction: $(\Delta\alpha_{\mathcal{A}}, \Delta\lambda, \Delta\nu) :=$
     *ReducedPrimalDualDirection* $(\alpha_{\mathcal{A}}, \lambda, \nu)$.
   Set $\hat{\alpha}_{\mathcal{A}} := \alpha_{\mathcal{A}} + \Delta\alpha_{\mathcal{A}}$, $\hat{\lambda} := \lambda + \Delta\lambda$, $\hat{\nu} := \nu + \Delta\nu$.
7: If $\mu(\hat{\alpha}, \hat{\lambda}, \hat{\nu}) \le \min\left\{r^{\frac{3}{2}-\theta}, \gamma r\right\}$, set $\alpha := \hat{\alpha}$, $\lambda := \hat{\lambda}$, $\nu = \hat{\nu}$, $r := \mu(\alpha, \lambda, \nu)$, $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$.
   Goto Step 2.
8: Set $t := 1$. Decrease $t := \frac{t}{2}$ until $\mathcal{L}_k(\alpha_{\mathcal{A}} + t\Delta\alpha_{\mathcal{A}}, \lambda_g, \nu_g)$
     $- \mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g) \le \eta t \left(\nabla \mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g), \Delta\alpha_{\mathcal{A}}\right)$.
9: Set $\hat{\lambda} := \lambda_g \psi'(kc(\alpha + t\Delta\alpha))$, $\hat{\nu} := \nu_g - kg(\alpha + t\Delta\alpha)$,
   $\alpha_{\mathcal{A}} := \alpha_{\mathcal{A}} + t\Delta\alpha_{\mathcal{A}}$.
10: If $\|\nabla_\alpha \mathcal{L}_k(\alpha_{\mathcal{A}}, \lambda_g, \nu_g)\| \le \frac{\sigma}{k} \cdot \max\left\{\|\hat{\lambda} - \lambda_g\|, \|\hat{\nu} - \nu_g\|\right\}$, goto Step 12.
11: Find direction: $(\Delta\alpha_{\mathcal{A}}, \Delta\lambda, \Delta\nu) :=$
      *ReducedPrimalDualDirection* $(\alpha_{\mathcal{A}}, \lambda_g, \nu_g)$, goto Step 8.
12: If $\mu(\alpha, \hat{\lambda}, \hat{\nu}) \le \gamma r$, set $\lambda := \hat{\lambda}, \lambda_g := \hat{\lambda}, \nu := \hat{\nu}$,
    $\nu_g := \hat{\nu}, r := \mu(\alpha, \lambda, \nu)$, $k := \max\left\{\frac{1}{\sqrt{r}}, k\right\}$,
    goto Step 2.
13: Set $k := k\beta$. If $|\mathcal{A}| < fg\_switch$ goto Step 11, else goto Step 5.
```

equivalent problem. Table 5.14 displays results for the projected fast gradient method.

Table 5.15 displays speedup for the NRAL fast gradient method. Only 2 of the 3 problems converge before reaching the iteration limit of 500000 iterations. For the *Statlog* problem, the speedup is fairly significant at 2.67.

The results for the projected fast gradient method are better than the NRAL fast gradient method. Table 5.16 displays the speedup for the projected fast gradient method. There is a significant speedup for all of the problems. This speedup is 8.8 and 7 for the *Statlog* problem and the *Eyestate* problem respectively. For the *Telescope Data* problem, the speedup of 3.77 results in the projected fast gradient method taking 1.5 fewer days to

Algorithm 14 Projected Fast Gradient Minimization

```
   **function** $(\alpha_{\mathcal{A}}, \hat{\nu})$ :=*ProjectedFastGradient* $(\alpha_{\mathcal{A}}, \nu, r)$
1: Initialize $\beta = 1.0$, $\hat{\alpha}_{\mathcal{A}} = \alpha_{\mathcal{A}}$
2: $\alpha_{\mathcal{A}} = y_{\mathcal{A}} - \frac{1}{L}\nabla_y \mathcal{L}_k(y_{\mathcal{A}}, \nu)$
   For $j$ from 1 to $m$
     If $\alpha_j <= 0$ set $\alpha_j := 0$
     If $\alpha_j >= C$ set $\alpha_j := C$
   End for
   $\beta = (1 + \sqrt{4\beta^2 + 1})/2$
   $y_{\mathcal{A}} = \alpha_{\mathcal{A}} + \frac{\beta-1}{\beta}(\alpha_{\mathcal{A}} - \hat{\alpha}_{\mathcal{A}})$
   $\hat{\alpha}_{\mathcal{A}} = \alpha_{\mathcal{A}}$
3: Set $\hat{\nu} := \nu_g - kg\,(\alpha_{\mathcal{A},i})$.
4: If $\|\nabla_y \mathcal{L}_k\,(y_{\mathcal{A}}, \nu_g)\| \leq .1 * r$ then **break** else goto Step 2.
```

Table 5.12: SVM Results without Acceleration with $\gamma = .0521$

Data Set Name	Instances	Attributes	Iterations	Solution Time (s)
Pen-Based Recognition Digits	3498	16	7	427.83
Statlog (Landsat Satellite)	4435	36	5	1243.77
Page Block Classification	5473	10	68	4109.08
Eyestate	14980	14	11	35477.04
Telescope Data	19020	10	27	179439.58

Table 5.13: SVM Results with NRAL Fast Gradient Method with $\gamma = .0521$

Data Set Name	Instances	Attributes	Iterations	Solution Time (s)
Pen-Based Recognition Digits	3498	16	14699	549.58
Statlog (Landsat Satellite)	4435	36	6511	466.37
Page Block Classification	5473	10	Iter limit	Reached

solve the problem than without the projected fast gradient method.

Table 5.14: SVM Results with Projected Fast Gradient Method with $\gamma = .0521$

Data Set Name	Instances	Attributes	Iterations	Solution Time (s)
Pen-Based Recognition Digits	3498	16	2008	82.39
Statlog (Landsat Satellite)	4435	36	1957	140.93
Page Block Classification	5473	10	20518	1350.92
Eyestate	14980	14	6735	5051.56
Telescope Data	19020	10	35960	47619.07

Table 5.15: SVM Results with and without NRAL Fast Gradient Method

Data Set Name	Instances	Time(s) w/out Grad	Time(s) with Grad	Speedup
Recognition of Digits	3498	427.83	549.58	0.77
Statlog	4435	1243.77	466.37	2.67
Page Block	5473	4109.08	Reached Iter	Limit

Table 5.16: SVM Results with and without Projected Fast Gradient Method

Data Set Name	Instances	Time(s) w/out Grad	Time(s) with Grad	Speedup
Recognition of Digits	3498	427.83	82.39	5.19
Statlog	4435	1243.77	140.93	8.83
Page Block	5473	4109.08	1350.92	3.04
Eyestate	14980	35477.04	5051.56	7.02
Telescope Data	19020	179439.58	47619.07	3.77

5.6 Summary

This chapter discussed application of the exterior point method to the support vector machine problem. Also, an active-passive strategy was discussed which uses properties of the EPM to decrease the computation time of the exterior-point algorithm by reducing the size of the systems solved at each iteration. Finally, fast gradient methods were discussed to decrease the computation time even more for large problems.

Numerical results for the exterior-point algorithm applied to the SVM indicate that the EPM is a feasible training algorithm for SVMs. Numerical results for the active-passive strategy show that the strategy results in a speedup for most problems with passive variables at the solution, with this speedup being significant for problems with few active variables at the solution. Finally, results for the fast gradient methods show that the projected fast gradient method can provide a significant decrease in computation time as shown in table 5.16.

Chapter 6: Concluding Remarks

The main goal of this dissertation was to develop a numerically efficient implementation of a general purpose nonlinear optimization algorithm based on exterior-point methods to work well for many nonlinear problems. In addition to the general purpose optimization solver, efficient algorithms based on exterior-point methods for solving the support vector machine problem were investigated. Numerical results were presented for both the general purpose optimization algorithm and the application of the exterior point method algorithm to the support vector machine problem.

For the general purpose optimization algorithm, numerical results indicate the developed solver to be competitive with existing solvers, while also being able to solve some problems that the existing solvers SNOPT, NITRO or LOQO, were unable to solve.

Active-passive strategies were also investigated for the EPM for both primal and dual variables. Numerical results for these active-passive strategies showed computation time was the same or decreased for most problems solved with the primal strategy and for all problems solved with the dual strategy.

For the application of the exterior-point algorithm to the support vector machine problem, numerical results indicate the EPM to be a feasible training algorithm for SVMs. An active-passive strategy was also applied to the SVM problem, for which numerical results show a speedup for most problems with passive variables at the solution, with this speedup being significant for problems with few active variables at the solution. Fast gradient methods for use with large SVM problems were investigated. Numerical results showed the projected fast gradient method provided a significant speedup for large SVM problems.

Nonlinear optimization problems will continue to be of fundamental importance in science, engineering and mathematics. The wide variety of nonlinear optimization problems will continue to motivate the development of and improvement of optimization methods. Directions for future work in this area include the following. First, investigating more rigorous regularization methods for the general optimization code that could improve performance. Second, implementing a parallelized version of the SVM code for high performance computing starting the method with small sets of variables on multiple machines, similar to the method described in section 5.4. Third, investigating other problems that the EPM would work well on, as was done with the SVM problem here. Some examples of convex optimization problems that arise in engineering include model predictive control problems [62], filter design problems [63] and multi-antenna beamforming [64], amongst many others.

Appendix A: Benchmarking Results

Table A.1 displays results for benchmarking the EPM solver on the CUTEr test set.

Table A.1: Benchmarking Results for EPM Solver

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
3pk	30	30	6	6.65E-07	0.016
aircrfta	5	0	5	9.82E-13	0
aircrftb	5	0	16	4.33E-09	0.031
airport	84	210	41	7.74E-07	0.046
aljazzaf	3	4	38	8.47E-07	0
allinit	3	3	15	3.26E-08	0
allinitc	3	4	1003	0.000101	0.015
allinitu	4	0	8	2.37E-08	0
alsotame	2	5	37	3.81E-08	0
argauss	3	0	3	8.40E-11	0
arglina	100	0	2	2.82E-14	0.031
arglinb	10	0	2	4.01E-10	0
arglinc	8	0	2	3.27E-11	0
argtrig	100	0	8	2.79E-12	0.109
artif	5000	0	21	2.29E-07	1.89
arwhead	5000	0	7	5.77E-12	0.562
aug2d	20192	9996	56	5.05E-07	133
aug2dc	20200	10194	62	9.48E-07	149
aug2dcqp	20200	30196	148	9.80E-07	363
aug2dqp	20192	30188	147	7.80E-07	360
aug3d	3873	1000	5	5.81E-07	0.265
aug3dc	3873	1000	6	4.28E-09	0.437
aug3dcqp	3873	4873	56	8.48E-07	5
aug3dqp	3873	4873	27	1.41E-07	2.34
avgasa	6	18	8	1.74E-07	0
avgasb	6	18	10	2.26E-07	0
avion2	49	113	3	23000	0.031
bard	3	0	9	1.16E-11	0
batch	46	161	1061	0.0139	0.5
bdexp	5000	0	13	8.04E-07	1.14

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
bdqrtic	1000	0	11	2.20E-13	0.141
bdvalue	5000	0	2	3.17E-15	0.156
beale	2	0	2	0	0
bigbank	1773	4360	1	192000	0.031
biggs3	3	0	10	1.07E-08	0
biggs5	5	0	16	3.63E-08	0
biggs6	6	0	54	3.18E-09	0
biggsb1	1000	1998	31	2.96E-07	0.171
biggsc4	4	15	40	2.24E-07	0
blockqp1	2005	5011	47	8.15E-07	3.52
blockqp2	2005	5011	22	2.65E-07	1.61
blockqp3	2005	5011	61	9.46E-07	4.61
blockqp4	2005	5011	29	5.97E-07	2.17
blockqp5	2005	5011	61	9.58E-07	4.66
bloweya	2002	3004	430	0.0133	27.7
bloweyb	2002	3004	688	1.68E-07	18.8
bloweyc	2002	3004	824	3.56E-07	26.4
booth	2	0	2	3.55E-15	0
box2	2	0	13	5.70E-07	0
box3	3	0	8	1.57E-07	0
bqp1var	1	2	8	4.32E-09	0
bqpgabim	46	92	90	9.65E-07	0.031
bqpgasim	50	100	91	9.94E-07	0.031
brainpc0	6905	13805	1011	227	1690
brainpc1	6905	13805	1012	0.122	1710
brainpc2	13805	27605	1012	0.122	6690
brainpc3	6905	13805	1012	0.121	1690
brainpc4	6905	13805	1012	0.121	1710
brainpc5	6905	13805	1012	0.121	1700
brainpc6	6905	13805	1012	0.121	1700
brainpc7	6905	13805	1012	0.121	1690
brainpc8	6905	13805	1012	0.121	1690
brainpc9	6905	13805	1050	0.121	1760
bratu1d	1001	0	7	3.00E-07	0.078
bratu2d	4900	0	2	9.32E-07	0.187
bratu2dt	4900	0	5	5.46E-07	0.593
bratu3d	3375	0	4	3.96E-07	2.48

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
britgas	450	834	1012	0.852	8.69
brkmcc	2	0	4	5.64E-13	0
brownal	10	0	8	8.52E-09	0
brownbs	2	0	6	0	0
brownden	4	0	9	3.09E-10	0
broydn3d	10000	0	6	2.83E-08	1.64
broydn7d	1000	0	5	4.57	0.391
broydnbd	5000	0	9	1.02E-12	1.34
brybnd	5000	0	9	1.02E-12	1.33
bt1	2	1	14	8.28E-08	0
bt10	2	2	9	8.95E-08	0
bt11	5	3	19	1.44E-07	0
bt12	5	3	5	2.81E-13	0
bt13	5	2	33	2.28E-07	0
bt2	3	1	13	5.17E-11	0
bt3	5	3	6	1.46E-07	0
bt4	3	2	7	1.28E-11	0
bt5	3	2	28	4.11E-11	0
bt6	5	2	14	2.32E-07	0
bt7	5	3	140	9.76E-07	0
bt8	5	2	8	2.30E-07	0
bt9	4	2	18	1.46E-07	0
byrdsphr	3	2	24	1.87E-08	0
camel6	2	4	7	4.50E-11	0.015
cantilvr	5	6	15	1.27E-07	0
catena	32	11	131	8.26E-07	0.015
catenary	496	166	121	6.75E-07	0.312
cb2	3	3	13	2.74E-07	0
cb3	3	3	9	3.65E-07	0
cbratu2d	882	0	2	1.02E-08	0.031
cbratu3d	1024	0	2	4.22E-07	0.328
chaconn1	3	3	10	1.22E-07	0
chaconn2	3	3	7	6.78E-07	0
chainwoo	1000	0	83	4.69E-09	1.53
chandheq	100	100	17	6.26E-07	0.406
chebyqad	50	100	3	1.64	0.375
chemrcta	5000	5000	3	4840000	0.438

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
chemrctb	1000	1000	3	1920	0.046
chenhark	1000	1000	27	4.53E-07	0.14
chnrosnb	50	0	44	1.59E-07	0
cliff	2	0	28	1.14E-10	0
clnlbeam	1499	2996	1020	0.0261	41.5
clplatea	4970	0	7	1.56E-07	0.75
clplateb	4970	0	7	4.40E-08	0.75
clplatec	4970	0	2	2.44E-08	0.187
cluster	2	0	15	6.38E-07	0
concon	15	16	107	2.56E-07	0.015
congigmz	3	5	29	2.58E-07	0
coolhans	9	0	2	0	0.015
core1	65	165	108	1.87E-07	0.031
core2	157	367	1014	2.65E18	2.97
corkscrw	8997	15000	1099	0.15	919
coshfun	61	20	1028	0.0483	0.25
cosine	10000	0	7	3.39E-07	1.84
cragglvy	5000	0	15	1.53E-08	1.75
cresc100	6	205	1	16300	0.015
cresc132	6	2659	1	217000	0.093
cresc4	6	13	1	1720	0
cresc50	6	105	1	8160	0
csfi1	5	10	90	5.36E-08	0.015
csfi2	5	9	73	5.77E-07	0
cube	2	0	29	2.72E-09	0.015
curly10	10000	0	2	6.87E-07	0.562
curly20	10000	0	3	1.14E-15	1.12
curly30	10000	0	3	2.10E-15	1.39
cvxbqp1	10000	20000	16	5.41E-07	5.03
cvxqp1	1000	2500	386	1.42E-07	10.5
cvxqp2	10000	22500	766	1.88E-07	1400
cvxqp3	10000	27500	1006	0.0465	3360
dallasl	837	2272	1	193000	0.015
dallasm	164	447	1	89500	0
dallass	44	117	1	79900	0
deconvb	51	62	1012	32.9	0.421
deconvc	51	52	254	3.77E-07	0.109

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
deconvu	51	0	2	1.00E-10	0
degenlpa	20	54	220	4.29E-07	0.015
degenlpb	20	55	274	2.33E-10	0.031
demymalo	3	3	34	7.55E-08	0
denschna	2	0	7	6.64E-12	0
denschnb	2	0	7	8.39E-07	0.015
denschnc	2	0	12	7.68E-10	0.015
denschnd	3	0	38	8.35E-07	0
denschne	3	0	10	8.28E-08	0
denschnf	2	0	7	6.28E-10	0
dipigri	7	4	35	6.61E-07	0
disc2	28	35	250	1.68E-07	0.015
discs	33	78	1100	3.09	0.234
dittert	327	655	7	7.2	0.093
dixchlng	10	5	129	5.01E-07	0.015
dixchlnv	100	150	1	715000000	0.015
dixmaana	3000	0	7	5.35E-12	0.265
dixmaanb	3000	0	8	3.34E-18	0.421
dixmaanc	3000	0	2	NAN	0.218
dixmaand	3000	0	4	NAN	0.312
dixmaane	3000	0	89	4.30E-15	4.01
dixmaanf	3000	0	3	NAN	0.265
dixmaang	3000	0	3	NAN	0.265
dixmaanh	3000	0	4	NAN	0.328
dixmaani	3000	0	23	1.38E-11	0.921
dixmaanj	3000	0	6	1.7	1.42
dixmaank	3000	0	4	NAN	0.328
dixmaanl	3000	0	3	NAN	0.265
dixon3dq	10	0	2	8.88E-16	0
djtl	2	0	27	2.90E-07	0.015
dnieper	57	136	75	4.02E-07	0.078
dqdrtic	5000	0	2	0	0.125
dqrtic	5000	0	35	5.46E-07	2.91
drcav1lq	10000	0	2	NAN	3.72
drcav2lq	10000	0	3	NAN	64.1
drcav3lq	10000	0	2	NAN	4.33
drcavty1	10000	0	2	NAN	3.72

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
drcavty2	10000	0	3	NAN	64.1
drcavty3	10000	0	2	NAN	4.33
dtoc1l	14985	9990	10	2.78E-08	15.9
dtoc1na	1485	990	10	3.66E-08	1.09
dtoc1nb	1485	990	16	5.56E-08	1.66
dtoc1nc	1485	990	35	1.57E-08	3.69
dtoc1nd	735	490	4	0.426	3.22
dtoc2	5994	3996	211	9.55E-07	63
dtoc3	14996	9997	146	8.20E-07	241
dtoc4	14996	9997	114	9.61E-07	187
dtoc5	9998	4999	1075	0.000457	699
dtoc6	10000	5000	533	9.42E-07	332
dual1	85	171	57	8.60E-07	0.359
dual2	96	193	39	5.28E-07	0.343
dual3	111	223	52	5.11E-07	0.671
dual4	75	151	37	8.60E-07	0.171
dualc1	9	31	76	3.19E-07	0
dualc2	7	23	86	4.74E-07	0
dualc5	8	17	30	1.92E-07	0
dualc8	8	31	113	5.15E-07	0
edensch	2000	0	8	1.34E-10	0.203
eg1	3	4	18	9.15E-07	0
eg2	1000	0	4	5.96E-09	0.031
eg3	101	400	1042	0.501	4.03
eigena	110	110	85	9.65E-07	0.171
eigena2	110	55	373	4.14E-07	0.843
eigenaco	110	55	11	6.61E-12	0.015
eigenals	110	0	15	1.62E-07	0.062
eigenb	110	0	106	1.56E-10	0.265
eigenb2	110	55	42	7.29E-07	0.093
eigenbco	110	55	58	2.66E-10	0.109
eigenbls	110	0	148	3.62E-07	0.75
eigenc2	462	231	539	2.92E-09	35.7
eigencco	30	15	18	1.07E-07	0
eigmaxa	101	301	66	7.37E-07	0.14
eigmaxb	101	303	80	3.71E-08	0.109
eigmaxc	22	64	18	1.00E-07	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
eigmina	101	303	16	1.37E-07	0.046
eigminb	101	303	58	3.72E-08	0.093
eigminc	22	66	16	1.98E-07	0
engval1	5000	0	9	1.32E-12	0.75
engval2	3	0	18	1.13E-11	0
errinros	50	0	34	2.32E-07	0
expfit	2	0	20	3.95E-07	0
expfita	5	21	48	9.45E-08	0
expfitb	5	101	63	8.20E-07	0.015
expfitc	5	501	144	9.85E-07	1.42
explin	120	240	24	2.55E-07	0.015
explin2	120	240	26	4.11E-09	0
expquad	120	20	6	1200	0
extrasim	2	2	2	0	0
extrosnb	10	0	2	0	0
fccu	19	8	5	4.62E-07	0
fletcbv2	100	0	2	8.29E-07	0.015
fletcbv3	10000	0	1011	2	283
fletchbv	10000	0	1011	200000000	283
fletchcr	100	0	16	2.36E-12	0.015
fletcher	4	5	26	5.27E-09	0
flosp2hh	650	0	1005	0.00929	59
flosp2hl	650	0	3	2.93E-10	0.188
flosp2hm	650	0	6	9.90E-07	0.421
flosp2th	650	0	1006	6.71E-06	60.1
flosp2tl	650	0	2	1.96E-08	0.093
flosp2tm	650	0	5	8.73E-08	0.343
fminsrf2	1024	0	17	4.42E-09	0.25
fminsurf	1024	0	35	3.91E-08	20.4
freuroth	5000	0	2	NAN	0.39
gausselm	1495	4215	9	4.92E-09	1.01
genhs28	10	8	4	6.51E-09	0
genhumps	5	0	66	1.30E-07	0
genrose	500	0	5	3.16E-10	0.015
gigomez1	3	3	18	2.68E-07	0
gilbert	1000	1	29	3.14E-08	0.14
goffin	51	50	22	1.71E-07	0.046

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
gottfr	2	0	6	7.87E-09	0
gouldqp2	699	1747	67	9.37E-07	0.39
gouldqp3	699	1747	39	9.02E-07	0.25
gpp	250	498	114	9.38E-07	9.77
gridneta	8964	7053	277	3.38E-10	170
gridnetb	13284	6724	28	5.53E-07	29.8
gridnetc	7564	6365	181	9.26E-07	61.9
gridnetd					
gridnete					
gridnetf					
gridnetg	44	53	14	7.94E-09	0
gridneth	61	37	6	9.46E-07	0
gridneti	61	57	18	1.66E-07	0.015
grouping	100	325	16	4.88E-07	0.046
growth	3	0	9	8.17E-34	0
growthls	3	0	9	8.17E-34	0
gulf	3	0	27	2.65E-07	0.015
hadamals	90	180	249	1.42E-07	0.39
hadamard	65	257	9	2.44E-07	0.156
hager1	10000	5000	3	2.73E-12	0.859
hager2	10000	5000	3	1.93E-12	1.36
hager3	10000	5000	2	5.64E-07	0.781
hager4	10000	10000	502	6.48E-07	345
haifam	85	150	298	8.59E-07	0.531
haifas	7	9	17	4.54E-07	0
hairy	2	0	27	1.67E-17	0
haldmads	6	42	118	7.12E-07	0.031
hanging	288	180	46	2.84E-07	0.562
harkerp2	100	100	209	5.71E-07	0.359
hart6	6	12	15	8.38E-08	0
hatflda	4	4	1	3.23	0
hatfldb	4	5	1	3.23	0
hatfldc	4	6	6	6.26E-08	0
hatfldd	3	0	19	5.69E-07	0
hatflde	3	0	20	5.24E-07	0.015
hatfldf	3	0	37	1.87E-08	0
hatfldg	25	0	14	1.56E-08	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
hatfldh	4	15	34	2.24E-07	0
heart6	6	0	8	5.94E-11	0
heart6ls	6	0	8	5.94E-11	0
heart8	8	0	190	9.78E-10	0
heart8ls	8	0	160	6.19E-11	0
helix	3	0	16	1.32E-07	0
hilberta	10	0	2	1.52E-15	0
hilbertb	50	0	2	5.09E-14	0
himmelba	2	0	2	0	0.015
himmelbb	2	0	20	7.21E-07	0
himmelbc	2	0	8	2.36E-08	0
himmelbd	2	0	21	1.90E-10	0
himmelbe	3	0	7	2.59E-07	0
himmelbf	4	0	12	5.07E-12	0
himmelbg	2	0	5	1.02E-08	0
himmelbh	2	0	6	5.80E-08	0
himmelbi	100	112	254	0.00344	0.109
himmelbj	43	57	1	46.7	0
himmelbk	24	38	173	3.47E-07	0.062
himmelp1	2	4	23	2.23E-07	0
himmelp2	2	5	93	3.06E-08	0
himmelp3	2	6	6	3.26E-07	0
himmelp4	2	7	7	7.93E-11	0
himmelp5	2	7	16	2.44	0
himmelp6	2	8	11	3.39E-07	0.015
hong	4	9	24	3.59E-07	0
hs001	2	1	28	2.80E-08	0
hs002	2	1	17	8.98E-07	0
hs003	2	1	9	2.20E-07	0
hs004	2	2	8	4.82E-09	0
hs005	2	4	8	4.66E-09	0
hs006	2	1	67	6.01E-11	0
hs007	2	1	32	2.27E-10	0
hs008	2	2	8	4.13E-11	0
hs009	2	1	9	2.07E-11	0
hs010	2	1	12	8.85E-09	0
hs011	2	1	8	5.80E-07	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
hs012	2	1	18	2.60E-07	0
hs013	2	3	1008	0.000369	0.031
hs014	2	2	8	8.77E-07	0
hs015	2	3	25	2.28E-08	0
hs016	2	5	20	7.97E-07	0
hs017	2	5	43	9.23E-07	0.015
hs018	2	6	59	2.74E-07	0
hs019	2	6	44	6.73E-07	0
hs020	2	5	63	7.55E-09	0
hs021	2	5	27	4.67E-08	0
hs022	2	2	7	9.80E-07	0
hs023	2	9	25	2.76E-07	0
hs024	2	4	690	8.99E-05	0.015
hs025	3	6	14	5.55E-07	0.015
hs026	3	1	67	8.61E-07	0.015
hs027	3	1	18	6.27E-07	0
hs028	3	1	2	4.44E-16	0
hs029	3	1	13	4.26E-07	0
hs030	3	7	8	2.48E-07	0
hs031	3	7	8	2.05E-07	0
hs032	3	5	13	7.48E-07	0
hs033	3	6	20	3.59E-08	0
hs034	3	8	13	5.31E-07	0
hs035	3	4	7	5.96E-07	0
hs036	3	7	15	3.73E-07	0
hs037	3	7	15	3.73E-07	0
hs038	4	8	41	1.81E-13	0.015
hs039	4	2	18	1.46E-07	0
hs040	4	3	5	4.27E-08	0
hs041	4	9	17	2.04E-08	0
hs042	3	4	7	1.67E-09	0
hs043	4	3	10	4.45E-07	0
hs044	4	10	16	1.14E-07	0.015
hs045	5	10	7	3.54E-07	0
hs046	5	2	21	3.73E-07	0
hs047	5	3	42	3.34E-07	0
hs048	5	2	2	3.55E-15	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
hs049	5	2	17	9.03E-07	0
hs050	5	3	10	4.41E-07	0
hs051	5	3	2	2.22E-16	0
hs052	5	3	7	8.53E-09	0
hs053	5	13	6	1.94E-07	0
hs054	6	13	6	2.24E-07	0
hs055	6	14	16	2.33E-07	0
hs056	7	11	183	1.79E-07	0.015
hs057	2	3	2	0.865	0
hs059	2	7	12	1.84E-08	0
hs060	3	7	7	2.26E-07	0
hs061	3	2	19	2.15E-08	0
hs062	3	7	137	7.75E-07	0.015
hs063	3	5	29	1.21E-07	0
hs064	3	4	18	5.77E-09	0.015
hs065	3	7	76	2.49E-08	0
hs066	3	8	8	4.22E-07	0
hs067	10	27	138	1.47E-07	0
hs068	function	myerf	not	available	
hs069	function	myerf	not	available	
hs070	4	9	1	49.3	0
hs071	4	10	31	1.77E-08	0
hs072	4	10	46	4.08E-07	0.015
hs073	4	7	26	9.83E-07	0
hs074	4	12	40	3.15E-08	0
hs075	4	12	43	4.69E-08	0.015
hs076	4	7	20	1.08E-07	0
hs077	5	2	13	1.59E-07	0
hs078	5	3	5	1.34E-07	0
hs079	5	3	5	2.45E-07	0
hs080	5	13	7	1.66E-10	0
hs081	5	13	22	9.75E-08	0
hs083	5	13	32	5.04E-07	0
hs084	5	13	1044	1.21	0.062
hs085	5	46	1053	35900	0.218
hs086	5	11	23	5.92E-08	0
hs087	9	22	38	2.13E-08	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
hs088	2	1	250	9.28E-07	0.093
hs089	3	1	249	9.18E-07	0.234
hs090	4	1	249	9.30E-07	0.281
hs091	5	1	241	9.41E-07	0.421
hs092	6	1	249	9.31E-07	0.375
hs093	6	8	64	2.07E-07	0
hs095	6	16	58	6.31E-07	0
hs096	6	16	52	6.31E-07	0
hs097	6	16	1005	0.122	0.031
hs098	6	16	1027	0.122	0.015
hs099	19	28	15	5.96E-08	0
hs100	7	4	35	6.61E-07	0
hs100lnp	7	2	32	9.47E-11	0
hs100mod	7	4	41	4.80E-09	0.015
hs101	7	20	2	230	0
hs102	7	20	2	229	0
hs103	7	20	2	228	0
hs104	8	22	1	4.59	0
hs105	8	16	2	92.1	0.015
hs106	8	22	1100	0.484	0.062
hs107	9	14	788	3140	0.046
hs108	9	14	79	2.33E-07	0.015
hs109	9	26	1013	1020	0.078
hs110	10	20	8	8.78E-07	0
hs111	10	23	21	3.15E-07	0
hs111lnp	10	3	21	3.14E-07	0
hs112	10	13	1	37.4	0
hs113	10	8	9	2.72E-07	0
hs114	10	31	89	1.14E-07	0.015
hs116	13	41	266	9.18E-07	0.016
hs117	15	20	756	904000	0.078
hs118	15	47	31	6.29E-08	0
hs119	16	40	71	6.17E-07	0
hs21mod	7	9	21	4.79E-07	0
hs268	5	5	6	1.29E-07	0
hs35mod	2	3	7	7.01E-07	0
hs3mod	2	1	9	2.33E-10	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
hs44new	4	9	21	1.73E-07	0
hs99exp	28	35	27	8.24E-08	0
hubfit	2	2	12	7.63E-08	0.015
hues-mod	10000	10002	130	7.26E-08	36.2
huestis	10000	10002	1008	81.7	307
humps	2	0	77	8.22E-08	0
hvycrash	201	352	1082	0.00591	5.06
hypcir	2	0	8	2.14E-11	0
indef	1000	0	1005	10.2	13.9
integreq	100	0	4	2.49E-11	0.765
jensmp	2	0	11	2.05E-12	0
kissing	127	903	1089	0.0197	6.97
kiwcresc	3	2	44	7.94E-07	0
kowosb	4	0	11	6.46E-08	0
ksip	20	1001	397	9.93E-07	191
lakes	90	96	1	557000	0
launch	25	79	1	4250	0
lch	600	1	6	8.24E-11	0.031
lewispol	6	21	1003	3.44E-05	0.062
liarwhd	10000	0	13	6.59E-09	3.78
linspanh	72	176	13	1.24E-07	0.031
liswet1	10002	10000	1013	2.20E-06	1.22E+03
liswet10	10002	10000	181	9.88E-07	304
liswet11	10002	10000	190	9.96E-07	316
liswet12	10002	10000	1061	1.54E-06	1.25E+03
liswet2	10002	10000	190	9.93E-07	321
liswet3	10002	10000	188	1.00E-06	317
liswet4	10002	10000	189	9.90E-07	319
liswet5	10002	10000	188	1.00E-06	317
liswet6	10002	10000	190	9.89E-07	321
liswet7	10002	10000	190	9.95E-07	321
liswet8	10002	10000	190	9.96E-07	321
liswet9	10002	10000	1054	1.60E-06	1.22E+03
lminsurf	15129	0	49	9.19E-07	42.1
loadbal	31	73	22	4.72E-07	0
loghairy	2	0	218	1.24E-09	0.015
logros	2	2	64	8.99E-07	0.015

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
lootsma	3	6	20	3.59E-08	0
lotschd	12	19	14	2.95E-08	0
lsnnodoc	5	10	30	1.95E-07	0.015
lsqfit	2	2	11	3.89E-08	0
madsen	3	6	49	1.76E-08	0
madsschj	81	158	405	8.64E-07	3.97
makela1	3	2	17	2.14E-08	0
makela2	3	3	34	2.57E-07	0
makela3	21	20	11	4.42E-07	0
makela4	21	40	8	2.41E-09	0
mancino	100	0	6	4.66E-08	0.25
manne	1094	2189	5	0.73	0.125
maratos	2	1	10	3.29E-08	0
maratosb	2	0	2	6.25E-14	0.015
matrix2	6	6	12	9.74E-07	0
maxlika	8	16	2	92.1	0.031
mccormck	50000	100000	25	2.85E-07	162
mconcon	15	16	107	2.56E-07	0
mdhole	2	1	39	2.51E-07	0
methanb8	31	0	24	2.48E-08	0.015
methanl8	31	0	10	49600	0.015
mexhat	2	0	5	9.17E-13	0
meyer3	3	0	1007	0.000509	0.093
mifflin1	3	2	13	4.22E-08	0
mifflin2	3	2	28	3.54E-07	0
minc44	303	621	1002	0.00249	11.9
minmaxbd	5	20	1004	187	0.031
minmaxrb	3	4	94	2.67E-07	0
minperm	1113	2246	6	9.01	1.91
minsurf	36	0	9	1.42E-07	0
mistake	9	14	145	7.47E-07	0.015
model	60	152	1034	0.555	0.39
morebv	5000	0	2	3.24E-15	0.14
mosarqp1	2500	3200	66	8.34E-07	2.48
mosarqp2	900	1500	51	4.30E-07	0.718
msqrta	1024	0	4	4.74E-13	3.25
msqrtals	1024	0	4	4.74E-13	3.27

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
msqrtb	1024	0	4	4.72E-13	3.25
msqrtbls	1024	0	4	4.72E-13	3.25
mwright	5	3	18	1.94E-07	0
nasty	2	0	2	1.75E-26	0
ncvxbqp1	10000	20000	103	0.858	374
ncvxbqp2	10000	20000	5	0.121	2.47
ncvxbqp3	10000	20000	5	0.121	2.7
ncvxqp1	1000	2500	6	419	0.328
ncvxqp2	1000	2500	6	NAN	0.218
ncvxqp3	1000	2500	103	157	33
ncvxqp4	1000	2250	7	NAN	0.172
ncvxqp5	1000	2250	103	295	24.5
ncvxqp6	1000	2250	7	98.9	0.406
ncvxqp7	1000	2750	6	1780	0.375
ncvxqp8	1000	2750	5	153	0.218
ncvxqp9	1000	2750	6	104	0.39
ngone	97	1371	1087	0.0103	21.5
noncvxu2	1000	0	114	1.28E-11	0.609
noncvxun	1000	0	114	1.28E-11	0.625
nondia	9999	0	7	1.33E-09	1.83
nondquar	10000	0	22	3.22E-07	6.51
nonmsqrt	9	0	1009	0.000255	0.047
nonscomp	10000	20000	323	8.68E-07	95.7
obstclal	64	128	17	4.43E-07	0.015
obstclbl	64	128	13	6.81E-07	0
obstclbu	64	128	13	2.12E-07	0.015
odfits	10	16	8	9.65E-08	0
oet1	3	1002	75	7.34E-07	0.328
oet2	3	1002	78	8.92E-07	0.453
oet3	4	1002	395	9.97E-07	2.23
oet7	7	1002	634	9.90E-07	120
optcdeg2	1198	1998	1029	0.0513	17.7
optcdeg3	1198	1998	318	1.45E-07	5.39
optcntrl	28	50	90	3.73E-08	0
optctrl3	118	80	303	9.96E-07	1.44
optctrl6	118	80	303	9.96E-07	1.44
optmass	66	55	7	2.56E-09	0.015

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
optprloc	30	89	269	5.33E-08	0.046
orthrdm2	4003	2000	97	8.66E-10	13.1
orthrds2	203	100	1013	1.09	2.08
orthrega	517	256	169	6.95E-08	1.06
orthregb	27	6	2	2.11E-10	0
orthregc	10005	5000	1011	3.57	744
orthregd	10003	5000	206	3.38E-13	140
orthrege	36	21	1011	2.97	0.187
orthrgdm	10003	5000	205	2.01E-08	140
orthrgds	10003	5000	713	8.24E-13	486
osbornea	5	0	35	3.05E-09	0
osborneb	11	0	1	4.49	0.015
oslbqp	8	11	7	2.96E-07	0.015
palmer1	4	3	1009	5.20E-05	0.078
palmer1a	6	2	50	3.72E-07	0.015
palmer1b	4	2	35	7.78E-08	0
palmer1c	8	0	2	9.44E-08	0
palmer1d	7	0	2	1.23E-08	0
palmer1e	8	0	435	3.24E-10	0.015
palmer2	4	3	1013	40.8	0.031
palmer2a	6	2	58	5.84E-08	0
palmer2b	4	2	31	2.77E-07	0
palmer2c	8	0	2	6.96E-09	0
palmer2e	8	0	839	1.27E-09	0.031
palmer3	4	3	1013	242	0.031
palmer3a	6	2	69	6.91E-08	0
palmer3b	4	2	31	5.94E-10	0.015
palmer3c	8	0	2	3.36E-09	0
palmer3e	8	0	32	2.67E-10	0
palmer4	4	3	1021	0.000179	0.031
palmer4a	6	2	77	1.33E-09	0
palmer4b	4	2	33	5.23E-11	0
palmer4c	8	0	2	1.34E-08	0
palmer4e	8	0	1013	9.55E-05	0.046
palmer5a	8	2	1012	0.105	0.015
palmer5b	9	2	1014	0.053	0.015
palmer5c	6	0	2	1.51E-13	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
palmer5d	4	0	2	3.86E-12	0
palmer5e	8	1	1012	0.372	0.031
palmer6a	6	2	85	5.44E-08	0
palmer6c	8	0	2	2.01E-09	0
palmer6e	8	1	26	4.98E-07	0
palmer7a	6	2	1013	0.338	0.015
palmer7c	8	0	2	2.40E-08	0
palmer7e	8	1	1012	0.0976	0.031
palmer8a	6	2	92	2.76E-08	0.015
palmer8c	8	0	2	2.95E-09	0
palmer8e	8	1	23	1.53E-09	0
penalty1	1000	0	42	1.93E-07	28.4
penalty2	100	0	31	1.22E-09	0.046
pentagon	6	15	28	5.20E-07	0
pentdi	1000	1000	25	7.80E-07	0.187
pfit1	3	0	17	4.82E-07	0
pfit1ls	3	0	17	4.82E-07	0
pfit2	3	0	18	4.50E-07	0
pfit2ls	3	0	18	4.50E-07	0
pfit3	3	0	18	9.22E-07	0
pfit3ls	3	0	18	9.22E-07	0
pfit4	3	0	19	4.62E-07	0
pfit4ls	3	0	19	4.62E-07	0
polak1	3	2	11	4.99E-07	0
polak2	11	2	8	2.48E-07	0
polak3	12	10	103	2.46E-07	0.031
polak4	3	3	12	2.41E-10	0
polak5	3	2	201	1.37E-08	0.015
polak6	5	4	68	2.78E-07	0
porous1	4900	0	3	6.46E08	3.44
porous2	4900	0	61	5.45E08	130
portfl1	12	25	25	3.72E-07	0
portfl2	12	25	26	2.53E-07	0.015
portfl3	12	25	25	6.01E-07	0
portfl4	12	25	22	5.41E-07	0.015
portfl6	12	25	24	2.01E-07	0
powell20	1000	1000	1009	37	17

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
powellbs	2	0	22	5.50E-07	0
powellsq	2	0	13	7.57E-15	0
power	1000	0	2	0	0
probpenl	500	1000	12	4.10E-05	0.859
prodpl0	60	89	51	4.17E-07	0.015
prodpl1	60	89	69	5.90E-08	0.016
pspdoc	4	1	12	4.36E-07	0
pt	2	501	142	9.86E-07	0.25
qpcboei1	372	846	162	3.27E-07	0.812
qpcboei2	143	322	383	1.05E-08	0.812
qpcstair	385	741	699	2.10E-08	9.59
qpnboei1	372	846	269	NAN	1.31
qpnboei2	143	322	755	7.97	1.36
qpnstair	385	741	63	NAN	1.17
qr3d	155	10	111	2.72E-07	0.328
qr3dbd	127	10	59	2.72E-07	0.094
qr3dls	155	10	109	2.72E-07	0.437
qrtquad	120	20	25	2.55E-07	0.015
quartc	10000	0	37	3.84E-07	10.8
qudlin	12	24	27	8.91E-08	0
reading1	10001	25002	4	0.0234	4.03
reading2	15001	30002	1084	0.0114	2580
reading3	202	506	317	5.53E-08	1.86
recipe	3	0	20	5.20E-07	0
res	18	38	11	5.48E-07	0
rk23	17	17	991	2.76E-08	0.062
robot	7	2	65	3.37E-07	0
rosenbr	2	0	22	8.25E-07	0
rosenmmx	5	4	28	2.78E-07	0
s201.mod	2	0	2	0	0
s202.mod	2	0	8	5.68E-14	0
s203.mod	5	3	5	5.06E-07	0
s204.mod	2	0	5	1.75E-08	0
s205.mod	2	0	10	1.90E-08	0.015
s206.mod	2	0	5	1.11E-13	0
s207.mod	2	0	8	4.40E-12	0
s208.mod	2	0	22	8.25E-07	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
s209.mod	2	0	82	1.23E-08	0
s210.mod	2	0	358	1.33E-09	0
s211.mod	2	0	29	2.72E-09	0
s212.mod	2	0	11	9.29E-11	0
s213.mod	2	0	30	3.85E-07	0
s214.mod	2	0	87	0.00146	0
s215.mod	2	2	10	9.62E-07	0
s216.mod	2	1	10	5.93E-11	0
s217.mod	2	3	9	1.62E-08	0
s218.mod	2	2	44	4.08E-08	0
s219.mod	4	2	26	1.47E-07	0
s220.mod	2	3	12	3.46E-08	0
s221.mod	2	3	1011	0.000187	0.015
s222.mod	2	3	10	6.59E-08	0
s223.mod	2	6	39	6.16E-07	0
s224.mod	2	6	12	3.21E-07	0
s225.mod	2	5	25	2.76E-07	0
s226.mod	2	4	22	1.64E-08	0
s227.mod	2	2	10	5.88E-08	0
s228.mod	2	2	14	2.12E-08	0
s229.mod	2	4	24	1.47E-07	0
s230.mod	2	2	13	3.56E-07	0
s231.mod	2	2	26	1.39E-07	0
s232.mod	2	4	9	6.44E-07	0.015
s233.mod	2	1	13	3.47E-08	0
s234.mod	2	5	11	5.66E-07	0
s235.mod	3	1	29	6.27E-07	0
s236.mod	2	6	44	5.10E-09	0
s237.mod	2	6	33	4.65E-08	0
s238.mod	2	5	32	2.00E-08	0
s239.mod	2	5	9	2.94E-07	0
s240.mod	3	0	2	4.26E-14	0
s241.mod	8	5	40	1.75E-08	0
s242.mod	3	6	33	5.75E-07	0
s243.mod	3	0	5	2.75E-08	0
s244.mod	3	6	11	6.11E-07	0
s245.mod	3	0	9	4.36E-07	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
s246.mod	3	0	12	1.47E-11	0
s247.mod	4	4	20	2.84E-08	0
s248.mod	3	2	50	4.41E-09	0
s249.mod	3	2	13	1.30E-07	0
s250.mod	3	7	15	3.73E-07	0
s251.mod	3	7	15	3.73E-07	0
s252.mod	3	2	12	6.80E-07	0
s253.mod	3	4	15	4.06E-09	0
s254.mod	3	2	20	1.54E-09	0
s255.mod	4	0	2	4.44E-14	0
s256.mod	4	0	18	3.35E-07	0
s257.mod	4	2	18	4.60E-08	0
s258.mod	4	0	41	1.66E-13	0
s259.mod	4	1	14	6.35E-10	0
s260.mod	4	0	41	8.57E-14	0
s261.mod	4	0	16	3.26E-07	0
s262.mod	4	8	17	2.52E-09	0.015
s263.mod	4	4	33	7.70E-07	0
s264.mod	4	3	16	7.60E-07	0
s265.mod	4	10	202	0	0.015
s266.mod	5	0	8	3.49E-08	0
s267.mod	5	0	17	5.59E-07	0
s268.mod	5	5	6	1.29E-07	0.015
s269.mod	5	3	6	1.46E-07	0
s270.mod	5	5	11	5.33E-07	0
s271.mod	6	0	2	0	0
s272.mod	6	0	43	3.93E-08	0
s273.mod	6	0	11	2.90E-08	0
s274.mod	2	0	2	0	0
s275.mod	4	0	2	5.92E-16	0
s276.mod	6	0	2	1.84E-15	0
s277.mod	4	8	12	2.00E-08	0
s278.mod	6	12	12	6.00E-08	0
s279.mod	8	16	14	1.16E-08	0
s280.mod	10	20	16	3.14E-07	0
s281.mod	10	0	98	0.00764	0
s282.mod	10	0	62	1.26E-08	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
s283.mod	10	0	36	6.03E-07	0
s284.mod	15	10	31	1.48E-07	0
s285.mod	15	10	41	2.90E-07	0.015
s286.mod	20	0	22	8.25E-07	0
s287.mod	20	0	41	1.66E-13	0
s288.mod	20	0	18	3.35E-07	0
s289.mod	30	0	10	7.54E-07	0
s290.mod	2	0	2	0	0
s291.mod	10	0	2	0	0
s292.mod	30	0	2	0	0
s293.mod	50	0	2	1.09E-14	0
s294.mod	6	0	22	1.20E-10	0
s295.mod	10	0	28	8.92E-08	0
s296.mod	16	0	37	1.96E-08	0.015
s297.mod	30	0	62	3.26E-08	0
s298.mod	50	0	92	1.71E-11	0.015
s299.mod	100	0	168	1.48E-12	0.078
s300.mod	20	0	2	7.11E-15	0
s301.mod	50	0	2	1.42E-14	0
s302.mod	100	0	2	2.84E-14	0.015
s303.mod	20	0	12	2.12E-15	0
s304.mod	50	0	16	6.37E-09	0.015
s305.mod	100	0	20	1.11E-17	0.031
s307.mod	2	2	11	2.06E-12	0
s308.mod	2	0	11	1.15E-11	0
s309.mod	2	0	8	6.64E-12	0
s311.mod	2	0	8	2.36E-08	0
s312.mod	2	0	21	1.90E-10	0
s314.mod	2	0	4	5.64E-13	0
s315.mod	2	3	23	1.79E-07	0
s316.mod	2	1	28	8.32E-07	0
s317.mod	2	1	28	3.87E-07	0
s318.mod	2	1	27	1.47E-07	0
s319.mod	2	1	28	7.41E-07	0
s320.mod	2	1	28	1.54E-07	0
s321.mod	2	1	27	1.89E-07	0
s322.mod	2	1	29	2.46E-07	0

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
s323.mod	2	4	8	5.29E-07	0
s324.mod	2	3	59	1.92E-07	0
s325.mod	2	3	6	5.78E-09	0
s326.mod	2	4	17	3.51E-07	0
s327.mod	2	3	14	9.73E-08	0
s328.mod	2	4	27	2.21E-07	0
s329.mod	2	7	45	4.76E-10	0
s330.mod	2	5	17	2.81E-09	0
s331.mod	2	4	6	1.69E-08	0
s332.mod	2	5	102	55	0.109
s333.mod	3	0	9	1.59E-13	0
s334.mod	3	0	9	1.16E-11	0
s335.mod	3	2	32	5.34E-07	0
s336.mod	3	2	27	8.23E-07	0
s337.mod	3	3	8	2.05E-07	0
s338.mod	3	2	13	3.71E-07	0
s339.mod	3	4	24	9.69E-08	0
s340.mod	3	2	198	0.297	0
s341.mod	3	4	13	4.28E-07	0
s342.mod	3	4	12	6.52E-07	0
s343.mod	3	8	125	8.37E-10	0
s344.mod	3	1	7	2.25E-07	0.015
s345.mod	3	1	32	2.53E-09	0
s346.mod	3	8	125	8.37E-10	0
s347.mod	0				
s348.mod	3	5	1011	437	0.015
s350.mod	4	0	11	6.46E-08	0
s351.mod	4	0	12	1.65E-13	0
s352.mod	4	0	2	5.68E-14	0
s353.mod	4	7	8	1.73E-07	0
s354.mod	4	5	11	2.35E-07	0
s355.mod	4	5	172	7.59E-08	0
s356.mod	4	8	202	10.9	0.015
s357.mod	4	43	7	1.38E-07	0.015
s358.mod	5	10	84	5.59E-07	0
s359.mod	5	14	71	4.05E-08	0
s360.mod	5	11	217	1.14	0.031

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
s361.mod	5	14	904	2.04E-07	0.046
s364.mod	6	10	1	26.5	0.015
s365.mod	7	9	141	0.872	0
s365mod.mod	7	9	66	4.40E-07	0
s366.mod	7	28	96	9.40E-07	0
s367.mod	7	12	32	2.50E-07	0
s368.mod	8	16	6	1.90E-08	0
s369.mod	8	22	69	1.84E-07	0
s370.mod	6	0	14	6.28E-13	0.015
s371.mod	9	0	14	2.57E-08	0
s372.mod	9	18	8	917	0
s373.mod	9	6	6	8.33E-08	0
s374.mod	10	35	104	1.98E-08	0.031
s375.mod	10	9	34	9.05E-07	0
s376.mod	10	35	4	70.8	0.015
s377.mod	10	23	1	36.4	0
s378.mod	10	3	21	3.14E-07	0.015
s379.mod	11	0	1	4.49	0
s380.mod	12	27	2	3.02	0
s381.mod	13	17	37	9.33E-08	0
s382.mod	13	17	39	3.30E-07	0
s383.mod	14	29	442	6.94E-07	0.031
s384.mod	15	10	19	2.58E-07	0
s385.mod	15	10	65	9.27E-07	0.015
s386.mod	2	0	2	0	0
s387.mod	15	11	40	6.13E-07	0.015
s388.mod	15	15	92	4.44E-07	0.015
s389.mod	15	15	32	1.25E-08	0
s391.mod	30	0	1017	356	1.89
s392.mod	30	70	21	2.71E-09	0.015
s393.mod	48	75	66	8.40E-08	0.015
s394.mod	20	1	23	1.94E-08	0
s395.mod	50	1	21	2.79E-08	0
sawpath	589	782	1009	0.000732	9.28
scon1dls	1000	2000	170	0.154	1.3
scosine	10000	0	8	2.90E-08	2.2
scurly10	10000	0	23	9.34E-08	8.19

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
scurly20	10000	0	28	2.79E-07	11.7
scurly30	10000	0	26	2.48E-07	12.8
semicon1	1000	2000	170	0.154	1.28
semicon2	1000	2000	55	0.000936	0.437
sensors	1000	0	44	5.46E-10	3890
sim2bqp	2	2	11	2.02E-08	0.015
simbqp	2	2	11	1.51E-08	0
simpllpa	2	4	16	7.47E-07	0
simpllpb	2	5	19	1.02E-08	0
sineali	20	40	25	3.55E-08	0
sineval	2	0	44	3.66E-17	0
sinquad	10000	0	18	2.03E-08	5.8
sinrosnb	1000	1001	4	9.99E-08	0.14
sipow1	2	10000	135	9.86E-07	36.4
sipow1m	2	10000	133	9.84E-07	35.8
sipow2	2	5001	102	1.00E-06	7.23
sipow2m	2	5001	96	9.87E-07	6.76
sipow3	4	9999	406	9.94E-07	123
sipow4	4	10000	408	9.93E-07	125
sisser	2	0	15	4.81E-07	0
smbank	117	298	1	4160	0.031
smmpsf	720	983	1013	6.46	11.9
snake	2	2	1013	0.676	0.015
sosqp1					
sosqp2	20000	50001	1056	1.86E-06	6880
spanhyd	72	176	823	1.95E-07	0.609
spiral	3	2	76	7.14E-09	0
sreadin3	10000	25000	1011	1.92E-06	1150
srosenbr	10000	0	22	8.25E-07	5.95
sseblin	192	432	18	2.55E-07	0.015
ssebnln	192	456	1093	0.00403	3.5

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
ssnlbeam	31	60	182	1.54E-08	0.031
stancmin	3	5	21	1.97E-08	0
static3	434	240	175	9.95E-07	0.671
steenbra	432	540	42	9.41E-11	1.34
steenbrb	468	576	1	3460	0.062
steenbrc	540	666	1	10000	0
steenbrd	468	576	1	3460	0.062
steenbre	540	666	1	10000	0.093
steenbrf	468	576	1	2000	0
steenbrg	540	666	1	10000	0.078
supersim	2	3	8	3.65E-08	0
svanberg	5000	15000	239	9.85E-07	104
swopf	82	111	53	1.44E-07	0.078
synthes1	6	18	10	3.89E-07	0
tame	2	3	8	1.94E-07	0
fi2	3	10001	113	9.72E-07	29.8
tointqor	50	0	2	1.69E-14	0.015
trainf	20000	30002	1015	0.138	3200
trainh	20000	30002	1014	0.125	3350
tridia	10000	0	2	2.44E-14	0.328
trimloss	142	336	1	234	0
try-b	2	3	35	1.97E-08	0
twirism1	343	999	1015	0.772	57.4
twobars	2	6	16	3.65E-08	0
ubh1	17997	24006	6	1.50E-08	8.33
ubh5	19997	26006	1078	0.107	2730
vanderm1	100	99	2	1990	0.093
vanderm2	100	99	2	1990	0.078
vanderm3	100	99	3	85700	0.125
vanderm4	9	8	19	3.46E-07	0.015
vardim	100	0	26	1.71E-11	0.046

Problem	# Variables	# Constraints	# Iterations	Accuracy	Sln Time(s)
watson	31	0	440	9.69E-07	0.109
womflet	3	3	111	9.57E-07	0
woods	10000	0	41	2.58E-10	11.8
yao	2000	2001	1044	0.00393	54.6
yfit	3	1	37	7.03E-07	0
yfitu	3	0	33	2.52E-08	0
zangwil2	2	0	2	4.44E-16	0
zangwil3	3	0	2	4.26E-14	0
zecevic2	2	6	7	2.52E-07	0
zecevic3	2	6	17	2.96E-07	0
zecevic4	2	6	14	5.22E-07	0
zigzag	58	110	61	4.35E-07	0.031
zy2	3	5	13	2.51E-07	0

Table A.2 displays benchmarking results for SNOPT, NITRO and LOQO which can be found at http://www.princeton.edu/ rvdb/bench.html.

Table A.2: Benchmarking Results for SNOPT, NITRO and LOQO

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
3pk	30	0	991	23	19	0.83	0.14	0.03	1.72E+000	1.72E+000	1.72E+000
aircrfta	5	0	52	5	10	0.01	0.01	0.01	2.47E-012	4.02E-020	9.76E-018
aircrftb	5	0	94	20	16	0.01	0.02	0.02	9.62E-015	2.13E-011	1.46E-018
airport	84	42	130	13	24	0.8	0.25	0.36	4.80E+004	4.80E+004	4.80E+004
aljazzaf	3	1	64	148	49	0.02	0.15	0.07	7.50E+001	7.50E+001	7.50E+001
allinit	3	0	15	18	54	0	0.03	0.05	1.67E+001	1.67E+001	1.67E+001
allinitc	3	1			101			0.16	ERROR	ERROR	3.05E+001
allinitu	4	0	14	6	11	0	0.01	0.01	5.74E+000	5.74E+000	5.74E+000
alsotame	2	1	5	12	11	0	0.03	0.01	8.21E-002	8.21E-002	8.21E-002
argauss	3	0	6	2	6	0	0.01	0.01	1.13E-008	1.13E-008	1.13E-008
arglina	100	0	102	3	9	0.28	0.54	1.54	1.00E+002	1.00E+002	1.00E+002
arglinb	10	0	16	2	12	0.01	0.01	0.01	4.63E+000	4.63E+000	4.63E+000
arglinc	8	0	15	2	11	0.01	0.01	0.01	6.14E+000	6.14E+000	6.14E+000
argtrig	100	0	255	8	9	2.31	0.72	0.68	7.82E-015	8.34E-014	3.03E-017
artif	5000	0		112	41		10.77	6.71	(Time)	1.95E-013	1.14E-019
arwhead	5000	0		5	23		0.29	3.69	(Time)	1.38E-009	1.31E-009
aug2d	20192	9996			17			30.15	(Time)	ERROR	1.69E+006
aug2dc	20200	9996			19			37.9	(Time)	ERROR	1.82E+006
aug2dcqp	20200	9996			28			24.64	(Time)	ERROR	6.50E+006
aug2dqp	20192	9996			27			23.62	(Time)	ERROR	6.24E+006

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
aug3d	3873	1000	3647	6	14	585.03	1.14	9.74	5.54E+002	5.54E+002	5.54E+002
aug3dc	3873	1000	3990	4	14	1657.08	0.8	10.03	7.71E+002	7.71E+002	7.71E+002
aug3dcqp	3873	1000	2717	31	23	869.33	34.08	3.01	9.93E+002	9.93E+002	9.93E+002
aug3dqp	3873	1000	1857	30	28	196.82	19.8	3.46	6.75E+002	6.75E+002	6.75E+002
avgasa	6	6	15	20	13	0	0.05	0.01	-4.17E+000	-4.17E+000	-4.63E+000
avgasb	6	6	14	21	12	0.01	0.05	0.01	-4.13E+000	-4.13E+000	-4.13E+000
avion2	49	15	50	23	68	0.03	0.16	0.2	9.47E+007	9.47E+007	9.47E+007
bard	3	0	27	10	17	0.01	0.01	0.02	8.21E-003	8.21E-003	8.21E-003
batch	46	69	113	341	55	0.09	1.1	0.13	2.59E+005	2.59E+005	2.59E+005
bdexp	5000	0		13	33		0.9	3.97	(Time)	1.72E-003	4.55E-010
bdqrtic	1000	0	1194	12	13	124.87	0.32	0.47	3.98E+003	3.98E+003	3.98E+003
bdvalue	5000	0	0	1	8	0.09	30.37	1.08	1.04E-011	9.92E-012	9.29E-012
beale	2	0	17	7	10	0	0.01	0.01	1.74E-014	1.95E-016	6.47E-017
bigbank	1773	814		38	36		44.5	1.82	(Time)	-4.21E+006	-4.21E+006
biggs3	3	0	23	9	13	0	0.01	0.01	5.00E-013	7.07E-016	4.22E-016
biggs5	5	0	120	51	27	0.03	0.04	0.03	5.66E-003	6.65E-012	5.04E-017
biggs6	6	0	94	34	45	0.02	0.03	0.06	2.63E-006	9.54E-008	1.88E-018
biggsb1	1000	0		23	30		31.83	0.45	(IL)	1.51E-002	1.50E-002
biggsc4	4	7	8	29	21	0.01	0.06	0.02	-2.44E+001	-2.45E+001	-2.45E+001
blockqp1	2005	1001	2008	19	18	15.49	27.44	1.02	-9.97E+002	-9.96E+002	-9.97E+002
blockqp2	2005	1001	2362	18	12	18.18	5.66	0.72	-9.96E+002	-9.96E+002	-9.95E+002

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
blockqp3	2005	1001	2015	123	33	17.53	26.23	1.81	-4.97E+002	-4.97E+002	-4.98E+002
blockqp4	2005	1001	2356	17	18	18.18	4.25	1.06	-4.98E+002	-4.98E+002	-4.98E+002
blockqp5	2005	1001	2021	104	33	16.03	20.84	1.84	-4.97E+002	-4.97E+002	-4.98E+002
bloweya	2002	1002	603	12	76	2.79	7.84	4.81	-8.04E-006	-1.56E-002	-4.55E-002
bloweyb	2002	1002	401	14	138	1.98	3.37	8.84	-3.25E-020	2.71E-002	-3.05E-002
bloweyc	2002	1002	402	7	31	2.01	1.2	2.32	-3.21E-005	-1.33E-002	-3.03E-002
booth	2	0	6	2	9	0	0.01	0	3.66E-017	8.91E-029	5.20E-020
box2	2	0	15	5	12	0.01	0.01	0.01	3.71E-013	4.97E-012	6.83E-018
box3	3	0	26	7	11	0.01	0.01	0.01	1.22E-010	6.57E-012	2.62E-015
bqp1var	1	0	1	9	10	0	0.03	0.01	0.00E+000	1.28E-006	6.27E-009
bqpgabim	46	0	72	24	15	0.07	0.24	0.04	-3.79E-005	-3.54E-005	-3.79E-005
bqpgasim	50	0	83	26	15	0.09	0.26	0.04	-5.52E-005	-5.26E-005	-5.52E-005
brainpc0	6905	6900	7053		21	378.32		68.55	1.50E-003	ERROR	1.50E-003
brainpc1	6905	6900	9501		34	797.75		111.2	6.77E-008	ERROR	6.61E-007
brainpc2	13805	13800	9503		19	793.75		225.78	2.66E-007	ERROR	6.55E-007
brainpc3	6905	6900	9612		92	836.21		312.28	1.30E-006	ERROR	7.24E-007
brainpc4	6905	6900	9612		26	836.21		85.35	1.30E-006	ERROR	1.84E-006
brainpc5	6905	6900	9532		28	820.16		88.76	1.36E-006	ERROR	2.32E-006
brainpc6	6905	6900	9510		17	795.96		50.68	1.39E-007	ERROR	6.07E-007
brainpc7	6905	6900	9511		81	801.2		254.3	1.20E-007	ERROR	7.40E-007
brainpc8	6905	6900	9516		110	804.86		367.32	2.17E-007	ERROR	4.05E-004

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
brainpc9	6905	6900	9593		17	826.44		52.15	8.45E-007	ERROR	9.66E-007
bratu1d	1001	0		9	14		2.49	0.52	(IL)	-8.52E+000	-8.52E+000
bratu2d	4900	0		4	11		11.35	7.94	(Time)	1.94E-010	4.68E-016
bratu2dt	4900	0		4	14		10.65	9.47	(Time)	8.02E-005	1.23E-008
bratu3d	3375	0		5	11		1.79	83.61	(Time)	9.10E-013	1.98E-018
britgas	450	360	1038		15	3.67		0.75	0.00E+000	(IL)	2.31E-008
brkmcc	2	0	7	2	9	0	0.01	0.01	1.69E-001	1.69E-001	1.69E-001
brownal	10	0	31	7	12	0.01	0.01	0.02	7.60E-015	8.54E-015	1.91E-019
brownbs	2	0		30	33		0.03	0.03	ERROR	1.97E-031	1.57E-022
brownden	4	0	74	10	16	0.01	0.02	0.01	8.58E+004	8.58E+004	8.58E+004
broydn3d	10000	0		7	13		1.57	4.32	(Time)	2.63E-013	4.71E-023
broydn7d	1000	0	2867			1198.03			3.82E+000	(IL)	(IL)
broydnbd	5000	0		10	15		4.23	8.2	(Time)	1.44E-016	6.18E-024
brybnd	5000	0		10	15		4.16	8.31	(Time)	1.44E-016	6.18E-024
bt1	2	1	2	5	25	0.01	0.02	0.03	-1.00E+000	-1.00E+000	-1.00E+000
bt10	2	2	2	6	15	0.01	0.01	0.02	-1.00E+000	-1.00E+000	-1.00E+000
bt11	5	3	19	6	17	0.01	0.02	0.02	8.25E-001	8.25E-001	8.25E-001
bt12	5	3	10	6	15	0.01	0.02	0.02	6.19E+000	6.19E+000	6.19E+000
bt13	5	1	32	39	26	0.01	0.05	0.02	0.00E+000	2.56E-007	6.71E-021
bt2	3	1	19	12	18	0.01	0.02	0.02	3.26E-002	3.26E-002	3.26E-002
bt3	5	3	9	4	13	0	0.02	0.02	4.09E+000	4.09E+000	4.09E+000

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
bt4	3	2	11	6	14	0	0.02	0.02	-4.55E+001	-4.55E+001	-4.55E+001
bt5	3	2	24	6	11	0.01	0.02	0.01	9.62E+002	9.62E+002	9.62E+002
bt6	5	2	19	9	15	0	0.02	0.02	2.77E-001	2.77E-001	2.77E-001
bt7	5	3	18	30	27	0	0.04	0.04	3.07E+002	3.07E+002	3.60E+002
bt8	5	2	14	9	292	0.01	0.02	0.88	1.00E+000	1.00E+000	1.00E+000
bt9	4	2	18	14	17	0.01	0.02	0.02	-1.00E+000	-1.00E+000	-1.00E+000
byrdsphr	3	2		8	14		0.02	0.01	(IL)	-4.68E+000	-4.68E+000
camel6	2	0	12	14	11	0.01	0.03	0.01	-1.03E+000	-1.03E+000	-1.03E+000
cantilvr	5	1	29	20	17	0.01	0.04	0.02	1.34E+000	1.34E+000	1.34E+000
catena	32	11	99	68	30	0.21	0.1	0.06	-2.31E+004	-2.31E+004	-2.31E+004
catenary	496	166			41			0.77	(Time)	(IL)	-3.48E+005
cb2	3	3	7	21	13	0	0.04	0.02	1.95E+000	1.95E+000	1.95E+000
cb3	3	3	3	14	12	0.01	0.03	0.01	2.00E+000	2.00E+000	2.00E+000
cbratu2d	882	0	2417	1	8	245.3	0.11	0.7	2.18E-008	4.59E-012	3.74E-016
cbratu3d	1024	0	1142	2	8	122.78	0.16	2.75	1.64E-010	1.11E-012	3.59E-018
chaconn1	3	3	8	21	13	0.01	0.04	0.02	1.95E+000	1.95E+000	1.95E+000
chaconn2	3	3	3	14	11	0	0.03	0.01	2.00E+000	2.00E+000	2.00E+000
chainwoo	1000	0			60			1.63	(IL)	(IL)	6.36E+001
chandheq	100	0	125	22	29	0.65	2.48	3.38	2.42E-007	1.81E-007	2.72E-016
chebyqad	50	0	231	544	55	3.58	159.09	32.47	5.39E-003	5.39E-003	5.39E-003
chenhark	1000	0		646	18		3295.77	0.26	(IL)	-1.99E+000	-2.00E+000

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
chnrosnb	50	0	686	73	50	1.36	0.09	0.08	1.01E-013	2.07E-016	4.06E-020
cliff	2	0	26	27	33	0.01	0.02	0.03	2.00E-001	2.00E-001	2.00E-001
clnlbeam	1499	1000	1466	22	119	5.06	2.66	11.57	3.50E+002	3.45E+002	3.45E+002
clplatea	4970	0		14	10		3.31	2.94	(Time)	-1.26E-002	-1.26E-002
clplateb	4970	0		59	12		13.06	3.53	(Time)	-6.99E+000	-6.99E+000
clplatec	4970	0		3	9		19.97	2.62	(Time)	-5.02E-003	-5.02E-003
cluster	2	0	16	15	14	0.01	0.02	0.01	7.35E-010	1.38E-008	1.76E-014
concon	15	11	11	272	115	0	0.37	0.19	-6.23E+003	-6.23E+003	-6.23E+003
congigmz	3	5	4	31	34	0.01	0.05	0.03	2.80E+001	2.80E+001	2.80E+001
coolhans	9	0	0	7	0	0.01			0.00E+000	ERROR	3.74E-049
core1	65	50	87	334	81	0.03	2.52	0.24	9.11E+001	9.11E+001	9.11E+001
core2	157	122	198	478		0.17	9.55		7.29E+001	7.29E+001	ERROR
corkscrw	8997	7000	46387		33	1611.34		29.88	9.07E+001	ERROR	9.07E+001
coshfun	61	20	215		26	1.98		0.07	-7.73E-001	(IL)	-7.73E-001
cosine	10000	0		33	13		4.01	3.69	(Time)	-1.00E+004	-1.00E+004
cragglvy	5000	0		15	17		2.11	2.88	(Time)	1.69E+003	1.69E+003
cresc100	6	200	116		240	0.68		3.04	5.68E-001	(IL)	5.68E-001
cresc132	6	2654	1742		762.96				6.85E-001	ERROR	(IL)
cresc4	6	8	57		41	0.02		0.05	8.72E-001	(IL)	8.72E-001
cresc50	6	100	694		3.02				5.93E-001	(IL)	(IL)
csfi1	5	4	26	30	21	0.01	0.05	0.03	-4.91E+001	-4.91E+001	-4.91E+001

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
csfi2	5	4	39	51	26	0.01	0.07	0.03	5.50E+001	5.50E+001	5.50E+001
cube	2	0	35	38	34	0.01	0.03	0.03	1.06E-014	2.10E-016	7.33E-019
curly10	10000	0		23	18		1101.74	14.79	(Time)	-1.00E+006	-1.00E+006
curly20	10000	0			18			29.13	(Time)	ERROR	-1.00E+006
curly30	10000	0			18		2990.98	46.77	(Time)	-1.00E+006	-1.00E+006
cvxbqp1	10000	0	10000		18	121.86		39.76	2.25E+006	ERROR	2.25E+006
cvxqp1	1000	500	1593	12	29	6.03	5.67	6.93	1.09E+006	1.09E+006	1.09E+006
cvxqp2	10000	2500			25			1570.89	(Time)	ERROR	8.18E+007
cvxqp3	10000	7500	10217		38	417.44			1.16E+008	ERROR	(Time)
dallasl	837	598		263	40		40.53	1.19	ERROR	-2.03E+005	-2.03E+005
dallasm	164	119		134	77		2.62	0.48	ERROR	-4.82E+004	-4.82E+004
dallass	44	29	236		56	0.24		0.12	-3.24E+004	(IL)	-3.24E+004
deconvb	51	0	263	170	36	0.33	1.69	0.22	1.48E-008	2.16E-006	2.71E-003
deconvc	51	1	316	99	41	1.23	1.32	0.33	1.34E-008	1.12E-005	2.71E-003
deconvu	51	0	118	78	137	0.26	0.3	1.17	2.43E-007	1.03E-006	1.87E-011
degenlpa	20	14	28	141	29	0.01	0.31	0.03	3.06E+000	3.02E+000	3.06E+000
degenlpb	20	15	26	174	30	0.01	0.43	0.03	-3.07E+001	-3.08E+001	-3.07E+001
demymalo	3	3	14	27	17	0.01	0.04	0.01	-3.00E+000	-3.00E+000	-3.00E+000
denschna	2	0	11	5	9	0	0.01	0	7.06E-017	2.21E-012	9.09E-014
denschnb	2	0	8	5	10	0	0.01	0.01	1.78E-018	1.31E-015	1.22E-017
denschnc	2	0	14	11	16	0.01	0.01	0.01	1.83E-001	3.58E-014	1.91E-019

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
denschnd	3	0	141	37	37	0.02	0.03	0.03	7.29E-010	1.64E-008	2.82E-010
denschne	3	0	45	13	14	0	0.02	0.01	7.50E-013	2.01E-012	4.31E-014
denschnf	2	0	13	6	12	0	0.01	0.01	5.12E-018	6.51E-022	6.03E-020
dipigri	7	4	29	14	11	0.02	0.03	0.01	6.81E+002	6.81E+002	6.81E+002
disc2	28	23		35	48		0.08	0.11	(Inf)	1.56E+000	1.56E+000
discs	33	66			389			2.97	(IL)	(IL)	1.20E+001
dittert	327	264	3156	55	130	15.38	6.84	20.46	-2.00E+000	-2.00E+000	-2.00E+000
dixchlng	10	5	113	9	29	0.08	0.02	0.05	1.84E+003	2.47E+003	2.47E+003
dixchlnv	100	50	221	19		2.71	0.47		1.58E-014	2.31E-015	(IL)
dixmaana	3000	0	3007	8	12	1522.23	0.4	0.89	1.00E+000	1.00E+000	1.00E+000
dixmaanb	3000	0	3022	7	13	1613.16	0.7	1.82	1.00E+000	1.00E+000	1.00E+000
dixmaanc	3000	0	3028	9	14	1669.69	0.76	2.15	1.00E+000	1.00E+000	1.00E+000
dixmaand	3000	0	3031	10	15	1672.6	0.84	2.22	1.00E+000	1.00E+000	1.00E+000
dixmaane	3000	0	3170	36	18	2667.47	2.53	1.54	1.00E+000	1.00E+000	1.00E+000
dixmaanf	3000	0	3169		18	2656.85	2.48	3.07	1.00E+000	1.00E+000	1.00E+000
dixmaang	3000	0	3177	33	20	2711.34	2.52	3.31	1.00E+000	1.00E+000	1.00E+000
dixmaanh	3000	0	3130	35	22	2396.91	2.86	3.92	1.00E+000	1.00E+000	1.00E+000
dixmaani	3000	0		24	18		9.92	1.57	(Time)	1.00E+000	1.00E+000
dixmaanj	3000	0	3149	62	22	2526.37	13.38	3.97	1.09E+000	1.00E+000	1.00E+000
dixmaank	3000	0		55	24		10.6	4.33	(Time)	1.00E+000	1.00E+000
dixmaanl	3000	0	3293	61	24	3521.97	10.26	4.19	1.00E+000	1.00E+000	1.00E+000

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
dixon3dq	10	0	33	2	11	0.01	0.01	0.01	1.22E-012	1.42E-030	6.63E-016
djtl	2	0			23			0.03	ERROR	ERROR	-8.95E+003
dnieper	57	24	143	11	26	0.15	0.09	0.1	1.87E+004	1.87E+004	1.87E+004
dqdrtic	5000	0		8	12		0.47	1.14	(Time)	5.74E-018	1.35E-017
dqrtic	5000	0		38	64		1.89	5.05	(Time)	1.18E-005	1.65E-017
drcav1lq	10000	0			192				(Time)	(Time)	(Time)
drcav2lq	10000	0			184				(Time)	(Time)	(Time)
drcav3lq	10000	0			78				(Time)	(Time)	(Time)
drcavty1	10000	0			192				(Time)	(Time)	(Time)
drcavty2	10000	0			184				(Time)	(Time)	(Time)
drcavty3	10000	0			78				(Time)	(Time)	(Time)
dtoc1l	14985	9990			17			19.34	(Time)	ERROR	1.25E+002
dtoc1na	1485	990	2347	7	14	145.76	3.45	9.62	1.27E+001	1.27E+001	1.27E+001
dtoc1nb	1485	990	2750	6	12	153.71	3.16	7.78	1.59E+001	1.59E+001	1.59E+001
dtoc1nc	1485	990	2702	10	10	215.53	4.05	5.66	2.50E+001	2.50E+001	2.50E+001
dtoc1nd	735	490	1070	23	31	55.79	2.65	6.81	1.25E+001	1.26E+001	1.26E+001
dtoc2	5994	3996	4594	5	20	797.19	4.72	29.83	5.09E-001	5.09E-001	5.09E-001
dtoc3	14996	9997	11271		54	597.31		31.19	2.35E+002	ERROR	2.35E+002
dtoc4	14996	9997			20			81.47	(Time)	ERROR	2.87E+000
dtoc5	9998	4999		5	19		1.68	47.2	(Time)	1.53E+000	1.54E+000
dtoc6	10000	5000		13	27		4.05	66.08	(Time)	1.35E+005	1.35E+005

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
edensch	2000	0	2035	9	11	480.78	0.33	0.55	1.20E+004	1.20E+004	1.20E+004
eigena	110	0	197	70	24	0.42	1.56	0.59	1.78E-012	1.15E-005	2.30E-009
eigena2	110	55	141	34	110	1.36	0.42	4.47	8.25E+001	8.25E+001	8.25E+001
eigenaco	110	55	20	3	28	0.04	0.08	0.91	0.00E+000	0.00E+000	3.30E-020
eigenals	110	0	166	33	34	0.5	1.23	3.02	1.90E-013	1.06E-013	5.44E-021
eigenb	110	0		111	84		3.01	2.36	(IL)	2.64E-011	3.03E-020
eigenb2	110	55	142	96	36	1.46	0.85	1.33	1.60E+000	1.60E+000	1.60E+000
eigenbco	110	55	31	2		0.04	0.06		9.00E+000	9.00E+000	(IL)
eigenbls	110	0		119	83		6.06	5.84	(IL)	7.41E-011	1.18E-019
eigenc2	462	231			77			163.92	ERROR	ERROR	7.72E+002
eigencco	30	15	57	12	32	0.11	0.04	0.14	5.69E-013	2.09E-011	1.08E-018
eigmaxa	101	101	124	23	67	0.09	0.12	0.4	-7.00E+000	-1.00E+000	-2.00E+000
eigmaxb	101	101	141	26	20	2.02	0.11	0.17	-5.75E-001	-7.79E-002	-8.70E-003
eigmaxc	22	22	29	5	17	0.02	0.02	0.04	-1.00E+000	-1.00E+000	-2.00E+000
eigmina	101	101	110	23	73	0.06	0.12	0.43	1.00E+000	1.00E+000	1.00E+000
eigminb	101	101	103	26	13	0.21	0.11	0.11	8.70E-003	7.79E-002	9.67E-004
eigminc	22	22	31	20	17	0.02	0.05	0.05	1.00E+000	1.00E+000	1.00E+000
engval1	5000	0		10	16		0.57	1.78	(Time)	5.55E+003	5.55E+003
engval2	3	0	53	17	22	0.01	0.02	0.02	5.80E-016	9.87E-017	2.28E-021
errinros	50	0	670	92	50	1.27	0.1	0.09	3.99E+001	3.99E+001	3.99E+001
expfit	2	0	14	12	12	0	0.01	0.01	2.41E-001	2.41E-001	2.41E-001

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
expfita	5	21	37	32	23	0.01	0.07	0.03	1.14E-003	1.14E-003	1.14E-003
expfitb	5	101	191	48	31	0.11	0.23	0.08	5.02E-003	5.02E-003	5.02E-003
expfitc	5	501	1659	143	46	3.48	3.07	0.43	2.33E-002	2.33E-002	2.33E-002
explin	120	0	241	66	23	0.05	0.89	0.05	-7.24E+005	-7.24E+005	-7.24E+005
explin2	120	0	160	63	24	0.04	0.86	0.04	-7.24E+005	-7.24E+005	-7.24E+005
expquad	120	0	345	37	25	1.03	0.14	0.08	-3.62E+006	-3.62E+006	-3.62E+006
extrasim	2	1	1	11	11	0	0.03	0.01	1.00E+000	1.00E+000	1.00E+000
extrosnb	10	0	0		9	0		0.01	0.00E+000	ERROR	6.86E-020
fccu	19	8	29	6	19	0.01	0.02	0.04	1.11E+001	1.11E+001	1.11E+001
fletcbv3	10000	0							(Time)	(IL)	(IL)
fletchbv	100	0							(Time)	(IL)	(IL)
fletchcr	100	0	576	99	51	2.71	0.14	0.14	1.70E-012	7.81E-015	1.15E-020
fletcher	4	4		20	15		0.04	0.02	(Inf)	1.95E+001	1.95E+001
flosp2hh	650	0			161			28.67	(IL)	ERROR	3.89E+001
flosp2hl	650	0			8			0.58	(IL)	(IL)	3.89E+001
flosp2hm	650	0			8			0.62	(IL)	(IL)	3.89E+001
flosp2th	650	0			9			0.7	(IL)	(IL)	1.00E+001
flosp2tl	650	0			9			0.7	(IL)	(IL)	1.00E+001
flosp2tm	650	0			9			0.72	(IL)	(IL)	1.00E+001
fminsrf2	15625	0			394			979.79	(Time)	ERROR	1.00E+000
fminsurf	1024	0	1337	293	55	250.41	112.31	342.53	1.00E+000	1.00E+000	1.00E+000

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
freuroth	5000	0		12	21		1.62	6.95	(Time)	6.08E+005	6.08E+005
gausselm	1495	3690	2321	136		49.24	38.09		-1.00E+000	-1.75E+001	(IL)
genhs28	10	8	15	2	10	0	0.01	0.01	9.27E-001	9.27E-001	9.27E-001
genhumps	5	0	75	52	79	0.02	0.04	0.08	6.90E-013	1.87E-016	4.62E-013
genrose	500	0							(IL)	(IL)	(IL)
gigomez1	3	3	9	28	19	0.01	0.04	0.02	-3.00E+000	-3.00E+000	-3.00E+000
gilbert	1000	1	1046	33	37	1351.52	0.91	350.59	4.82E+002	4.82E+002	4.82E+002
goffin	51	50	25	45	13	0.05	0.74	0.1	-1.61E-013	1.28E-005	4.01E-009
gottfr	2	0	22	9	11	0	0.01	0.01	1.50E-016	1.42E-009	2.03E-015
gouldqp2	699	349	456	22	33	7.23	6.62	0.69	1.89E-004	1.94E-004	1.88E-004
gouldqp3	699	349	469	22	23	1.4	1.67	0.56	2.07E+000	2.07E+000	2.07E+000
gpp	250	498	1063	18	19	12.74	11.51	5.74	1.44E+004	1.44E+004	1.44E+004
gridneta	8964	6724	8773	24	24	1685.39	24.25	6.59	3.05E+002	3.05E+002	3.05E+002
gridnetb	13284	6724			18			16.74	(Time)	ERROR	1.43E+002
gridnetc	7564	3844			41			16.01	(Time)	ERROR	1.62E+002
gridnetd	3945	2644	2988	23	24	48.2	13.28	5.92	5.66E+002	5.66E+002	5.66E+002
gridnete	7565	3844		14	19		9.1	14.43	(Time)	2.07E+002	2.07E+002
gridnetf	7565	3844			37			23.63	(Time)	ERROR	2.42E+002
gridnetg	44	34	48	22	15	0.05	0.09	0.05	7.33E+001	7.33E+001	7.33E+001
gridneth	61	36	99	21	13	0.15	0.09	0.05	3.96E+001	3.96E+001	3.96E+001
gridneti	61	36	112	23	13	0.18	0.13	0.04	4.02E+001	4.02E+001	4.02E+001

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
grouping	100	125	44		17	0.03		0.15	1.39E+001	ERROR	1.39E+001
growth	3	0	4	164	79	0	0.11	0.08	3.54E+003	1.00E+000	1.00E+000
growthls	3	0	5	164	78	0	0.12	0.09	3.54E+003	1.00E+000	1.00E+000
gulf	3	0	95	31	27	0.05	0.07	0.08	2.85E-015	3.55E-014	1.09E-016
hadamals	90	0	526	46	336	0.44	1.1	7.2	2.53E+001	2.63E+001	2.78E+001
hadamard	65	256	9		12	0.04		0.12	1.00E+000	(IL)	1.00E+000
hager1	10000	5000		4	10		1.21	2.43	(Time)	8.81E-001	8.81E-001
hager2	10000	5000		4	10		1.74	4.06	(Time)	4.32E-001	4.32E-001
hager3	10000	5000		4	21		2.19	20.87	(Time)	1.41E-001	1.41E-001
hager4	10000	5000			17			7.13	(Time)	ERROR	2.79E+000
haifam	85	150	936	152	43	1.85	1.48	0.51	-4.50E+001	-4.50E+001	-4.50E+001
haifas	7	9	326	58	16	0.19	0.08	0.03	-4.50E-001	-4.50E-001	-4.50E-001
hairy	2	0	42	42	61	0.01	0.03	0.05	2.00E+001	2.00E+001	2.00E+001
haldmads	6	42	123	49	35	0.09	0.14	0.1	1.22E-004	1.24E-004	1.22E-004
hanging	288	180	993	41	17	20.55	1.33	0.26	-6.20E+002	-6.20E+002	-6.20E+002
harkerp2	100	0	338	26	30	0.25	0.49	0.62	-5.00E-001	-4.99E-001	-5.00E-001
hart6	6	0	27	23	25	0.01	0.05	0.02	-3.32E+000	-3.32E+000	-3.32E+000
hatflda	4	0	31	42	8	0.01	0.06	0.01	1.64E-014	1.89E-010	1.62E-015
hatfldb	4	0	28	15	11	0	0.03	0.01	5.57E-003	5.57E-003	5.57E-003
hatfldc	4	0	14	9	9	0	0.03	0.01	7.98E-014	7.69E-013	7.62E-022
hatfldd	3	0	40	26	25	0.01	0.03	0.03	2.55E-007	8.57E-008	6.62E-008

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
hatflde	3	0	34	22	23	0.01	0.03	0.03	2.73E-006	5.15E-007	5.12E-007
hatfldf	3	0	90	14	17	0.01	0.02	0.02	2.20E-013	1.01E-007	1.81E-016
hatfldg	25	0	90	15	15	0.07	0.02	0.02	8.36E-012	5.65E-014	1.73E-017
hatfldh	4	7	6	26	16	0	0.05	0.02	-2.45E+001	-2.45E+001	-2.45E+001
heart6	6	0							(IL)	(IL)	(IL)
heart6ls	6	0							(IL)	(IL)	(IL)
heart8	8	0			57			0.08	(IL)	(IL)	3.36E-022
heart8ls	8	0			55			0.11	(IL)	(IL)	2.30E-019
helix	3	0	28	17	14	0	0.02	0.01	5.44E-017	1.54E-021	3.97E-020
hilberta	10	0	35	3	10	0.01	0.01	0.01	2.29E-007	2.28E-007	1.62E-013
hilbertb	50	0	56	5	10	0.07	0.05	0.1	5.52E-013	2.07E-013	4.90E-016
himmelba	2	0	6	2	10	0	0.01	0.01	1.03E-023	5.05E-029	1.66E-019
himmelbb	2	0	11	12	13	0	0.01	0.02	1.91E-012	1.95E-020	4.80E-021
himmelbc	2	0	10	8	15	0	0.01	0.01	7.06E-016	1.54E-015	1.68E-020
himmelbd	2	0	62		26	0.01		0.02	5.92E+000	ERROR	5.92E+000
himmelbe	3	0	15	6	12	0	0.01	0.01	2.70E-015	2.04E-011	1.36E-018
himmelbf	4	0	93	13	23	0.01	0.02	0.03	3.19E+002	3.19E+002	3.19E+002
himmelbg	2	0	9	7	8	0	0.01	0.01	1.72E-014	1.20E-018	1.80E-015
himmelbh	2	0	7	4	10	0	0.01	0.01	-1.00E+000	-1.00E+000	-1.00E+000
himmelbi	100	12	106	25	25	0.22	0.19	0.06	-1.76E+003	-1.76E+003	-1.76E+003
himmelbj	43	14	538		240	0.32		1.15	-1.91E+003	ERROR	-1.91E+003

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
himmelbk	24	14	78	26	18	0.06	0.12	0.06	5.18E-002	5.18E-002	5.18E-002
himmelp1	2	0	8	18	13	0.01	0.04	0.01	-2.39E+001	-6.21E+001	-6.21E+001
himmelp2	2	1	60	18	17	0.02	0.04	0.02	-8.20E+000	-6.21E+001	-6.21E+001
himmelp3	2	2	7	13	16	0.01	0.03	0.02	-5.90E+001	-5.90E+001	-5.90E+001
himmelp4	2	3	10	13	16	0	0.03	0.01	-5.90E+001	-5.90E+001	-5.90E+001
himmelp5	2	3	23	35	103	0	0.06	0.18	-5.90E+001	-5.90E+001	-5.90E+001
himmelp6	2	4	0	16	30	0	0.04	0.02	-5.90E+001	-5.90E+001	-5.90E+001
hong	4	1	4	13	20	0	0.03	0.02	1.35E+000	1.35E+000	1.35E+000
hs001	2	0	33	44	33	0.01	0.05	0.02	5.50E-019	2.63E-013	6.89E-019
hs002	2	0	30	19	32	0	0.03	0.06	5.04E-002	4.94E+000	4.94E+000
hs003	2	0	5	10	11	0	0.03	0.01	5.05E-034	1.28E-006	6.09E-009
hs004	2	0	2	9	8	0	0.02	0	2.67E+000	2.67E+000	2.67E+000
hs005	2	0	10	11	10	0	0.03	0.01	-1.91E+000	-1.91E+000	-1.91E+000
hs006	2	1	5	11	11	0	0.02	0.01	4.93E-032	5.30E-016	8.06E-018
hs007	2	1	11	7	16	0	0.02	0.02	-1.73E+000	-1.73E+000	-1.73E+000
hs008	2	2	2	5	16	0	0.01	0.02	-1.00E+000	-1.00E+000	-1.00E+000
hs009	2	1	9	5	10	0.01	0.02	0.01	-5.00E-001	-5.00E-001	-5.00E-001
hs010	2	1	13	16	16	0	0.03	0.01	-1.00E+000	-1.00E+000	-1.00E+000
hs011	2	1	11	12	15	0.01	0.03	0.02	-8.50E+000	-8.50E+000	-8.50E+000
hs012	2	1	10	12	10	0	0.03	0.01	-3.00E+001	-3.00E+001	-3.00E+001
hs013	2	1	1	25		0.01	0.04		1.28E+000	1.01E+000	(IL)

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
hs014	2	2	2	12	13	0	0.03	0.02	1.39E+000	1.39E+000	1.39E+000
hs015	2	2	2	13	25	0	0.03	0.03	3.07E+002	3.60E+002	3.60E+002
hs016	2	2	1	14	18	0.01	0.03	0.01	2.31E+001	2.31E+001	2.50E-001
hs017	2	2	12	26	27	0.01	0.04	0.02	1.00E+000	1.00E+000	1.00E+000
hs018	2	2	15	32	18	0.01	0.05	0.02	5.00E+000	5.00E+000	5.00E+000
hs019	2	2	4	24	19	0.01	0.04	0.02	-6.96E+003	-6.96E+003	-6.96E+003
hs020	2	3	1	13	24	0	0.03	0.02	4.02E+001	4.02E+001	4.02E+001
hs021	2	1	3	23	11	0	0.04	0.01	-1.00E+002	-1.00E+002	-1.00E+002
hs022	2	2	2	12	9	0	0.03	0	1.00E+000	1.00E+000	1.00E+000
hs023	2	5	2	12	18	0	0.03	0.01	2.00E+000	2.00E+000	2.00E+000
hs024	2	2	6	17	13	0	0.03	0.01	-1.00E+000	-1.00E+000	-1.00E+000
hs025	3	0	0	35	26	0	0.11	0.07	3.28E+001	2.25E-010	6.51E-018
hs026	3	1	32	15	15	0.01	0.03	0.01	2.11E-011	1.31E-010	2.49E-010
hs027	3	1	30	22	58	0.01	0.03	0.12	4.00E-002	4.00E-002	4.00E-002
hs028	3	1	8	2	8	0.01	0.01	0	1.70E-027	6.41E-031	1.25E-016
hs029	3	1	16	13	10	0.01	0.03	0.01	-2.26E+001	-2.26E+001	-2.26E+001
hs030	3	1	5	232	9	0.01	0.21	0	1.00E+000	1.00E+000	1.00E+000
hs031	3	1	15	18	13	0.01	0.04	0.01	6.00E+000	6.00E+000	6.00E+000
hs032	3	2	4	17	23	0	0.03	0.02	1.00E+000	1.00E+000	1.00E+000
hs033	3	2	1	24	12	0.01	0.04	0.01	-4.00E+000	-4.59E+000	-4.59E+000
hs034	3	2	5	25	16	0.01	0.04	0.01	-8.34E-001	-8.34E-001	-8.34E-001

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
hs035	3	1	16	12	10	0.01	0.03	0	1.11E-001	1.11E-001	1.11E-001
hs036	3	1	3	10	16	0	0.03	0.01	-3.30E+003	-3.30E+003	-3.30E+003
hs037	3	1	7	10	11	0	0.03	0.01	-3.46E+003	-3.46E+003	-3.46E+003
hs038	4	0	123	62	44	0.01	0.08	0.03	3.17E-017	2.37E-014	3.32E-018
hs039	4	2	18	14	17	0.01	0.02	0.02	-1.00E+000	-1.00E+000	-1.00E+000
hs040	4	3	10	3	11	0	0.01	0.02	-2.50E-001	-2.50E-001	-2.50E-001
hs041	4	1	11	17	16	0	0.04	0.01	1.93E+000	1.93E+000	1.93E+000
hs042	3	1	12	10	12	0	0.03	0.02	1.39E+001	1.39E+001	1.39E+001
hs043	4	3	15	13	11	0.01	0.03	0.01	-4.40E+001	-4.40E+001	-4.40E+001
hs044	4	6	7	19	12	0	0.04	0.01	-1.50E+001	-1.50E+001	-1.50E+001
hs045	5	0	0	17	23	0.01	0.04	0.02	2.00E+000	1.00E+000	1.00E+000
hs046	5	2	53	15	20	0.02	0.03	0.02	2.90E-011	6.71E-010	1.34E-011
hs047	5	3	28	15	21	0.01	0.03	0.02	6.97E-011	3.96E-009	3.70E-011
hs048	5	2	12	2	8	0	0.01	0	1.21E-013	2.47E-031	9.13E-017
hs049	5	2	25	14	24	0.01	0.02	0.02	5.99E-009	1.18E-007	1.21E-011
hs050	5	3	27	9	16	0	0.02	0.02	2.07E-013	4.98E-030	7.66E-015
hs051	5	3	10	2	8	0.01	0.01	0.01	1.06E-025	2.47E-032	1.54E-017
hs052	5	3	9	2	8	0	0.01	0.01	5.33E+000	5.33E+000	5.33E+000
hs053	5	3	8	6	11	0.01	0.02	0.01	4.09E+000	4.09E+000	4.09E+000
hs054	6	1	8	12	12	0.01	0.03	0.01	1.93E-001	1.93E-001	1.93E-001
hs055	6	6	3		14	0		0.02	6.67E+000	ERROR	6.33E+000

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
hs056	7	4	19	22	17	0	0.04	0.02	-3.46E+000	-3.46E+000	-3.46E+000
hs057	2	1	5	12	16	0.01	0.04	0.02	3.06E-002	3.06E-002	2.85E-002
hs059	2	3	18	22	22	0	0.04	0.02	-7.80E+000	-6.75E+000	-7.80E+000
hs060	3	1	12	9	18	0.01	0.03	0.02	3.26E-002	3.26E-002	3.26E-002
hs061	3	2	16	6	13	0.01	0.02	0.02	-1.44E+002	-1.44E+002	-1.44E+002
hs062	3	1	14	8	13	0.01	0.02	0.01	-2.63E+004	-2.63E+004	-2.63E+004
hs063	3	2	30	10	11	0.01	0.03	0.01	9.62E+002	9.62E+002	9.62E+002
hs064	3	1	37	17	30	0.01	0.03	0.03	6.30E+003	6.30E+003	6.30E+003
hs065	3	1	22	25	21	0.01	0.04	0.02	9.54E-001	9.54E-001	9.54E-001
hs066	3	2	7	22	16	0	0.04	0.01	5.18E-001	5.18E-001	5.18E-001
hs070	4	1	37	22	27	0.03	0.08	0.08	9.40E-003	9.40E-003	1.75E-001
hs071	4	2	10	14	16	0.01	0.03	0.02	1.70E+001	1.70E+001	1.70E+001
hs072	4	2	33	38	26	0.01	0.06	0.03	7.27E+002	7.28E+002	7.28E+002
hs073	4	3	11	13	21	0	0.03	0.02	2.99E+001	2.99E+001	2.99E+001
hs074	4	4	20	19	21	0	0.04	0.02	5.13E+003	5.13E+003	5.13E+003
hs075	4	4	9	108	21	0	0.14	0.02	5.17E+003	5.17E+003	5.17E+003
hs076	4	3	10	11	11	0	0.03	0.01	-4.68E+000	-4.68E+000	-4.68E+000
hs077	5	2	20	9	16	0	0.02	0.03	2.42E-001	2.42E-001	2.42E-001
hs078	5	3	12	4	12	0.01	0.02	0.02	-2.92E+000	-2.92E+000	-2.92E+000
hs079	5	3	14	5	9	0	0.02	0.01	7.88E-002	7.88E-002	7.88E-002
hs080	5	3	11	12	12	0	0.03	0.02	5.39E-002	5.39E-002	5.39E-002

			Iterations			Solution Time			Objective Value		
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
hs081	5	3	12	12	19	0	0.03	0.03	5.39E-002	5.39E-002	5.39E-002
hs083	5	3	3	11	16	0	0.03	0.01	-3.07E+004	-3.07E+004	-3.07E+004
hs084	5	3	20	8	23	0.01	0.02	0.03	-5.28E+006	-5.28E+006	-5.28E+006
hs085	5	38	19	49	191	0.03	0.11	0.74	-1.91E+000	-1.91E+000	-1.91E+000
hs086	5	6	16	19	13	0	0.04	0.02	-3.23E+001	-3.23E+001	-3.23E+001
hs087	11	6	25	198	24	0.01	0.3	0.03	8.83E+003	8.83E+003	8.83E+003
hs088	2	1	37		29	0.04		0.19	1.36E+000	(IL)	1.36E+000
hs089	3	1	61		34	0.1		0.29	1.36E+000	(IL)	1.36E+000
hs090	4	1	40		34	0.06		0.42	1.36E+000	(IL)	1.36E+000
hs091	5	1	44		35	0.11		0.57	1.36E+000	(IL)	1.36E+000
hs092	6	1	47		28	0.09		0.65	1.36E+000	(IL)	1.36E+000
hs093	6	2	36	9	13	0.01	0.03	0.02	1.35E+002	1.35E+002	1.35E+002
hs095	6	4	1	23	15	0	0.05	0.02	1.56E-002	1.57E-002	1.56E-002
hs096	6	4	1	23	16	0.01	0.05	0.02	1.56E-002	1.57E-002	1.56E-002
hs097	6	4	10	21	34	0.01	0.04	0.05	3.14E+000	3.14E+000	4.07E+000
hs098	6	4	10	22	40	0.01	0.05	0.06	3.14E+000	3.14E+000	4.07E+000
hs099	19	14	51	10	24	0.03	0.03	0.04	-8.31E+008	-8.31E+008	-8.31E+008
hs100	7	4	29	14	11	0.01	0.03	0.01	6.81E+002	6.81E+002	6.81E+002
hs100lnp	7	2	24	9	13	0	0.02	0.02	6.81E+002	6.81E+002	6.81E+002
hs100mod	7	4	27	16	15	0.01	0.04	0.01	6.79E+002	6.79E+002	6.79E+002
hs101	7	6	529	281	145	0.48	0.42	0.42	1.81E+003	1.81E+003	1.81E+003

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
hs102	7	6	539	288	100	0.48	0.42	0.27	9.12E+002	9.12E+002	9.12E+002
hs103	7	6	352	321	44	0.31	0.48	0.1	5.44E+002	5.44E+002	5.44E+002
hs104	8	6	31	35	14	0.02	0.06	0.02	3.95E+000	3.95E+000	3.95E+000
hs105	8	0	214	37	17	0.41	1.08	0.54	1.14E+003	1.14E+003	1.14E+003
hs106	8	6	32	90	61	0.01	0.13	0.08	7.05E+003	7.05E+003	7.05E+003
hs107	9	6	24		290	0.01		0.74	5.06E+003	(IL)	5.06E+003
hs108	9	13	49	38	19	0.02	0.07	0.03	-8.66E-001	-6.75E-001	-8.66E-001
hs109	9	10	43	162	77	0.01	0.29	0.15	5.33E+003	5.33E+003	5.33E+003
hs110	10	0	22	18	11	0.01	0.04	0.02	-4.58E+001	-4.58E+001	-4.58E+001
hs111	10	3	74	18	17	0.05	0.05	0.03	-4.78E+001	-4.78E+001	-4.78E+001
hs111lnp	10	3	120	12	21	0.1	0.03	0.04	-4.77E+001	-4.78E+001	-4.78E+001
hs112	10	3	51	11	27	0.01	0.03	0.06	-4.78E+001	-4.78E+001	-4.78E+001
hs113	10	8	32	15	16	0.02	0.04	0.01	2.43E+001	2.43E+001	2.43E+001
hs114	10	11	35	41	27	0.02	0.08	0.03	-1.77E+003	-1.77E+003	-1.77E+003
hs116	13	15	69	38	86	0.03	0.1	0.15	9.76E+001	9.76E+001	9.76E+001
hs117	15	5	68	39	19	0.03	0.09	0.02	3.23E+001	3.23E+001	3.23E+001
hs118	15	17	30	19	15	0.01	0.06	0.01	6.65E+002	6.65E+002	6.65E+002
hs119	16	8	56	39	32	0.01	0.11	0.05	2.45E+002	2.45E+002	2.45E+002
hs21mod	7	1	3	27	22	0	0.05	0.02	-9.60E+001	-9.60E+001	-9.60E+001
hs268	5	5	68	30	27	0.01	0.06	0.03	-7.22E-012	2.56E-007	6.53E-009
hs35mod	2	1	9	16	16	0.01	0.03	0.01	2.50E-001	2.50E-001	2.50E-001

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
hs3mod	2	0	6	12	12	0	0.03	0.01	7.35E-038	1.28E-006	3.08E-009
hs44new	4	5	9	12	21	0	0.03	0.02	-1.50E+001	-1.50E+001	-1.50E+001
hs99exp	28	21	188	17	318	0.16	0.05	0.63	-1.01E+009	-1.01E+009	-1.01E+009
hubfit	2	1	8	23	12	0	0.04	0.01	1.69E-002	1.69E-002	1.69E-002
hues-mod	10000	2			283			60.27	(Time)	ERROR	3.48E+007
huestis	10000	2			63			12.66	(Time)	ERROR	3.48E+011
humps	2	0	97	239	212	0.01	0.12	0.19	5.41E-011	6.11E-010	8.04E-013
hvycrash	201	150	2146	109	24	10.1	1.35	0.34	-2.19E-001	-1.68E-004	-2.19E-001
hypcir	2	0	9	5	13	0	0.01	0.01	2.64E-018	1.00E-017	1.64E-019
indef	1000	0							(Unb)	(IL)	(IL)
integreq	100	0	106	3	8	0.43	4.72	10.36	4.37E-014	8.95E-013	1.00E-017
jensmp	2	0	37	9	14	0	0.01	0.01	1.24E+002	1.24E+002	1.24E+002
kissing	127	903		87	33		16.7	3.14	(Inf)	8.45E-001	1.00E+000
kiwcresc	3	2	12	19	15	0	0.04	0.01	-1.66E-007	2.56E-006	-1.41E-008
kowosb	4	0	29	13	11	0.01	0.02	0.01	3.08E-004	3.08E-004	3.08E-004
ksip	20	1000	2438	33	47	25.85	25.04	1.83	5.76E-001	5.76E-001	5.76E-001
lakes	90	78	163		275	0.17		1.22	3.51E+005	ERROR	3.51E+005
launch	25	29							(Inf)	ERROR	(IL)
lch	600	1			29			49.66	(Time)	(IL)	-4.32E+000
lewispol	6	9							(Inf)	ERROR	(IL)
liarwhd	10000	0		13	22		2.26	7.72	(Time)	1.10E-009	7.97E-023

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
linspanh	72	32	7	14	12	0.01	0.11	0.03	-7.70E+001	-7.70E+001	-7.70E+001
liswet1	10002	10000			17			7.23	ERROR	ERROR	2.50E+001
liswet10	10002	10000			14			6.03	ERROR	ERROR	2.50E+001
liswet11	10002	10000	33		15	9.69		7.01	4.95E+001	ERROR	2.50E+001
liswet12	10002	10000			15			6.91	ERROR	ERROR	-5.03E+003
liswet2	10002	10000	33		12	4.82		4.77	2.50E+001	ERROR	2.50E+001
liswet3	10002	10000	49		41	11.11		12.38	2.50E+001	ERROR	2.50E+001
liswet4	10002	10000	51		46	10.13		14.26	2.50E+001	ERROR	2.50E+001
liswet5	10002	10000	53		33	12.77		10.67	2.50E+001	ERROR	2.50E+001
liswet6	10002	10000	50		19	11.01		5.8	2.50E+001	ERROR	2.50E+001
liswet7	10002	10000			10			3.82	ERROR	ERROR	2.50E+001
liswet8	10002	10000			10			3.72	ERROR	ERROR	2.50E+001
liswet9	10002	10000			10			3.77	ERROR	ERROR	2.50E+001
lminsurf	15129	0			148			429.49	(Time)	ERROR	9.00E+000
loadbal	31	31	87	23	24	0.08	0.1	0.04	4.53E-001	4.53E-001	4.53E-001
loghairy	2	0	167	79	89	0.03	0.05	0.09	1.82E-001	6.34E+000	1.82E-001
logros	2	0	144	65	72	0.01	0.07	0.05	0.00E+000	1.47E-013	0.00E+000
lootsma	3	2	1	24	12	0	0.04	0.01	2.00E+000	1.41E+000	1.41E+000
lotschd	12	7	6	14	15	0	0.04	0.02	2.40E+003	2.40E+003	2.40E+003
lsnnodoc	5	4	4	19	21	0	0.04	0.02	1.23E+002	1.23E+002	1.23E+002
lsqfit	2	1	7	23	12	0	0.04	0.01	3.38E-002	3.38E-002	3.38E-002

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
madsen	3	6	17	18	24	0.01	0.04	0.02	6.16E-001	6.16E-001	6.16E-001
madsschj	81	158	414	62	27	2.74	6.74	1.24	-7.97E+002	-7.97E+002	-7.97E+002
makela1	3	2	9	19	16	0.01	0.03	0.02	-1.41E+000	-1.41E+000	-1.41E+000
makela2	3	3	6	13	14	0.01	0.03	0.01	7.20E+000	7.20E+000	7.20E+000
makela3	21	20	128	18	18	0.06	0.05	0.03	-2.01E-011	5.12E-006	1.65E-009
makela4	21	40	1	19	12	0	0.06	0.02	0.00E+000	1.02E-005	1.35E-008
mancino	100	0		9	19		2.27	4.16	ERROR	2.81E-018	3.12E-021
manne	1094	730	729			1.77			-9.75E-001	(Time)	(IL)
maratos	2	1	5	3	9	0	0.02	0.01	-1.00E+000	-1.00E+000	-1.00E+000
maratosb	2	0	5	2	18	0	0.01	0.02	-1.00E+000	-1.00E+000	-1.00E+000
matrix2	6	2	19	28	29	0.01	0.04	0.03	1.49E-008	6.78E-006	2.03E-008
mccormck	50000	0			10			17.42	(Time)	ERROR	-4.57E+004
mconcon	15	11	11	272	115	0.01	0.37	0.19	-6.23E+003	-6.23E+003	-6.23E+003
mdhole	2	0	108	30	20	0.02	0.05	0.02	3.01E-032	1.28E-006	2.04E-010
mexhat	2	0	24	4	8	0	0.01	0.01	-4.01E-002	-4.01E-002	-4.01E-002
meyer3	3	0			183			0.2	ERROR	ERROR	8.79E+001
mifflin1	3	2	8	10	9	0	0.03	0.01	-1.00E+000	-1.00E+000	-1.00E+000
mifflin2	3	2	13	26	16	0.01	0.04	0.02	-1.00E+000	-1.00E+000	-1.00E+000
minc44	303	262	969	34		4.19	3.8		2.57E-003	2.57E-003	ERROR
minmaxbd	5	20	53	91	35	0.02	0.16	0.05	1.16E+002	1.16E+002	1.16E+002
minmaxrb	3	4	8	31	30	0	0.04	0.04	8.88E-016	5.12E-006	8.71E-009

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
minsurf	36	0	51	13	9	0.04	0.02	0.02	1.00E+000	1.00E+000	1.00E+000
model	60	32	36	57	15	0.01	0.24	0.03	5.74E+003	5.74E+003	5.74E+003
morebv	5000	0	0	1	8	0.14	31.19	1.19	1.04E-011	9.92E-012	9.30E-012
mosarqp1	2500	700	4578	28	18	344.92	31.41	1.04	-9.53E+002	-9.53E+002	-9.53E+002
mosarqp2	900	600	2574	24	22	77.89	17.96	0.87	-1.60E+003	-1.60E+003	-1.60E+003
msqrta	1024	0		48	33		555.46	844.18	(IL)	7.09E-012	8.20E-020
msqrtals	1024	0		48	33		554.07	845.35	(IL)	7.09E-012	8.20E-020
msqrtb	1024	0		47	26		361.79	766.17	(IL)	4.01E-011	2.69E-019
msqrtbls	1024	0		47	26		368.85	768.96	(IL)	4.01E-011	2.69E-019
mwright	5	3	17	6	16	0.01	0.02	0.03	2.50E+001	2.50E+001	2.50E+001
ncvxbqp1	10000	0	10000		539	121.96		1285	-1.99E+010	ERROR	-1.99E+010
ncvxbqp2	10000	0	10954		900	142.78		2125	-1.33E+010	ERROR	-1.33E+010
ncvxbqp3	10000	0	12440		403	175.3		974.3	-6.56E+009	ERROR	-6.52E+009
ncvxqp1	1000	500	626	59	422	1.63	9.97	375.2	-7.15E+007	-7.16E+007	-7.16E+007
ncvxqp2	1000	500	891		217	2.6	7.52	187.4	-5.78E+007	-5.78E+007	-5.78E+007
ncvxqp3	1000	500	1476	75	78	4.96	26.03	53.62	-3.14E+007	-3.08E+007	-3.14E+007
ncvxqp4	1000	250	811		399	1.48	2.52	188.67	-9.39E+07	-9.40E+07	-9.40E+07
ncvxqp5	1000	250	883		210	1.63	2.71	98.83	-6.64E+07	-6.63E+07	-6.63E+07
ncvxqp6	1000	250	1227	86	124	2.75	5.54	53.72	-3.53E+007	-3.46E+007	-3.48E+007
ncvxqp7	1000	750	609		459	1.98	11.27	457.97	-4.35E+007	-4.34E+007	-4.35E+007
ncvxqp8	1000	750	657		191	2.21	7.55	176.47	-3.05E+007	-3.05E+007	-3.05E+007

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
ncvxqp9	1000	750	1033		116	4.21	9.95	107.01	-2.16E+007	-2.15E+007	-2.16E+007
ngone	97	1273	235	97	32	2.33	10.62	1.76	-6.09E-001	-6.41E-001	-6.33E-001
nondia	9999	0		5	13		0.8	4.5	(Time)	5.76E-010	2.46E-022
nondquar	10000	0		163	24		33.9	6.36	(Time)	3.10E-006	7.18E-011
nonmsqrt	9	0		73	238		0.05	0.24	ERROR	7.52E-001	7.52E-001
nonscomp	10000	0			33			8.58	(Time)	ERROR	2.41E-009
odfits	10	6	30	12	12	0.01	0.03	0.01	-2.38E+003	-2.38E+003	-2.38E+003
oet1	3	1002	159	36	16	0.83		0.22	5.38E-001	ERROR	5.38E-001
oet2	3	1002	349	71	107	2.05	37.45	2.59	8.72E-002	8.72E-002	8.72E-002
oet3	4	1002	150	44	17	0.7	24.53	0.28	4.50E-003	4.51E-003	4.51E-003
oet7	7	1002	2186	88	228	60.68	85.7	12.24	4.42E-005	2.09E-003	4.45E-005
optcdeg2	1198	799	2902	29	74	13.89	2.13	3.72	2.30E+002	2.30E+002	2.30E+002
optcdeg3	1198	799	2778	36	55	13.2	3.15	2.84	4.61E+001	4.61E+001	4.61E+001
optcntrl	28	20	29	46	44	0.01	0.12	0.08	5.50E+002	5.50E+002	5.50E+002
optctrl3	118	80		34	36		0.15	0.32	(IL)	2.05E+003	2.05E+003
optctrl6	118	80		34	36		0.15	0.31	(IL)	2.05E+003	2.05E+003
optmass	66	55	106	21	18	0.1	0.08	0.06	-1.90E-001	-1.90E-001	-1.90E-001
optprloc	30	29	80	75	82	0.05	0.23	0.22	-1.64E+001	-1.64E+001	-1.64E+001
orthrdm2	4003	2000	4023	6	333	3363.35	35.8	250.98	1.56E+002	1.56E+002	1.05E+004
orthrds2	203	100			897			20.23	ERROR	ERROR	1.04E+003
orthrega	517	256	541	7	63	20.8	0.21	2.72	1.66E+003	1.66E+003	1.41E+003

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
orthregb	27	6	31	3	26	0.03	0.02	0.08	7.26E-012	4.06E-017	4.44E-019
orthregc	10005	5000			23			96.45	(Time)	ERROR	1.90E+002
orthregd	10003	5000			442			1656.74	(Time)	ERROR	4.25E+004
orthrege	36	20	261	488	28	1.01	0.75	0.1	3.87E+000	3.66E+000	5.90E+000
orthrgdm	10003	5000			12			46.83	(Time)	ERROR	1.51E+003
orthrgds	10003	5000			295			973.04	(Time)	ERROR	2.60E+004
osbornea	5	0	147	66	19	0.04	0.07	0.03	5.46E-005	5.47E-005	5.46E-005
osborneb	11	0	106	21	28	0.08	0.07	0.13	4.01E-002	4.01E-002	8.76E-002
oslbqp	8	0	6	16	19	0	0.04	0.01	6.25E+000	6.25E+000	6.25E+000
palmer1	4	0	41		16	0		0.02	1.18E+004	ERROR	1.18E+004
palmer1a	6	0	219	101	46	0.04	0.13	0.04	8.99E-002	8.99E-002	8.99E-002
palmer1b	4	0	116	35	39	0.02	0.06	0.05	3.45E+000	3.45E+000	3.45E+000
palmer1c	8	0	138		43	0.03		0.05	9.76E-002	ERROR	9.76E-002
palmer1d	7	0	91		138	0.02		0.14	6.53E-001	(IL)	6.53E-001
palmer1e	8	0	328		103	0.08		0.16	8.35E-004	(IL)	8.35E-004
palmer2	4	0	45		10	0.01		0.01	3.65E+003	ERROR	3.65E+003
palmer2a	6	0	168		95	0.03		0.12	1.72E-002	(IL)	1.72E-002
palmer2b	4	0	62	37	30	0.01	0.06	0.03	6.23E-001	6.23E-001	6.23E-001
palmer2c	8	0	46		48	0.01		0.07	1.44E-002	(IL)	1.44E-002
palmer2e	8	0	373		95	0.08		0.13	2.15E-004	(IL)	2.15E-004
palmer3	4	0	15		13	0.01		0.02	2.42E+003	ERROR	2.27E+003

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
palmer3a	6	0	198		88	0.03		0.12	2.04E-002	(IL)	2.04E-002
palmer3b	4	0	73	39	21	0.01	0.06	0.02	4.23E+000	4.23E+000	4.23E+000
palmer3c	8	0	123		44	0.02		0.07	1.95E-002	(IL)	1.95E-002
palmer3e	8	0	420		143	0.09		0.21	5.07E-005	(IL)	5.07E-005
palmer4	4	0	15		13	0		0.02	2.42E+003	ERROR	2.29E+003
palmer4a	6	0	129		71	0.02		0.1	4.06E-002	(IL)	4.06E-002
palmer4b	4	0	46	34	22	0.01	0.05	0.02	6.84E+000	6.84E+000	6.84E+000
palmer4c	8	0	127		41	0.02		0.06	5.03E-002	(IL)	5.03E-002
palmer4e	8	0	214		88	0.05		0.13	1.48E-004	(IL)	1.48E-004
palmer5a	8	0							(IL)	(IL)	(IL)
palmer5b	9	0			237			0.36	(IL)	(IL)	9.75E-003
palmer5c	6	0	24	32	36	0.01	0.03	0.04	2.13E+000	2.13E+000	2.13E+000
palmer5d	4	0	41	45	29	0	0.03	0.02	8.73E+001	8.73E+001	8.73E+001
palmer5e	8	0			38			0.06	(IL)	(IL)	1.63E+000
palmer6a	6	0	304		163	0.05		0.17	5.59E-002	ERROR	5.59E-002
palmer6c	8	0	53	105	155	0.01	0.08	0.13	1.64E-002	1.64E-002	1.64E-002
palmer6e	8	0	244	168	134	0.05	0.2	0.15	2.24E-004	2.24E-004	2.24E-004
palmer7a	6	0							(IL)	(IL)	(IL)
palmer7c	8	0	157		31	0.03		0.03	6.02E-001	(IL)	6.02E-001
palmer7e	8	0			104			0.15	(IL)	(IL)	1.02E+001
palmer8a	6	0	109		53	0.02		0.06	7.40E-002	(IL)	7.40E-002

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
palmer8c	8	0	221	890	46	0.04	0.59	0.05	1.60E-001	1.60E-001	1.60E-001
palmer8e	8	0	153		64	0.04		0.08	6.34E-003	(IL)	6.34E-003
penalty1	1000	0	1170	46	56	61.84	14.8	338.67	9.69E-003	9.69E-003	9.69E-003
penalty2	100	0	231	19	22	0.79	0.16	0.26	9.71E+004	9.71E+004	9.71E+004
pentagon	6	12	23	37	27	0.01	0.06	0.03	1.37E-004	1.38E-004	1.37E-004
pentdi	1000	0	4	19	20	0.04	0.8	0.48	-7.50E-001	-7.45E-001	-7.50E-001
pfit1	3	0	1263	521	235	0.14	0.3	0.22	4.87E-015	5.31E-007	1.13E-022
pfit1ls	3	0	1263	521	235	0.14	0.31	0.22	4.87E-015	5.31E-007	1.13E-022
pfit2	3	0	1122		88	0.13		0.08	4.18E-017	ERROR	1.08E-020
pfit2ls	3	0	1122		88	0.13		0.08	4.18E-017	ERROR	1.08E-020
pfit3	3	0	1236	266	116	0.14	0.16	0.1	2.37E-017	2.15E-009	2.41E-022
pfit3ls	3	0	1236	266	116		0.16	0.1	2.15E-009	2.41E-022	
pfit4	3	0	1332		214	0.15		0.2	2.10E-019	ERROR	1.67E-020
pfit4ls	3	0	1332		214	0.15		0.2	2.10E-019	ERROR	1.67E-020
polak1	3	2	14	14	14	0	0.03	0.01	2.72E+000	2.72E+000	2.72E+000
polak2	11	2	97	74	26	0.06	0.08	0.03	5.46E+001	5.46E+001	5.46E+001
polak3	12	10	88	85	22	0.1	0.16	0.04	5.93E+000	5.93E+000	5.93E+000
polak4	3	3	7	730	13	0.01	0.68	0.01	-1.18E-008	3.84E-006	2.29E-013
polak5	3	2	27	15	71	0.01	0.03	0.09	5.00E+001	5.00E+001	5.00E+001
polak6	5	4	36	25	31	0.01	0.05	0.04	-4.40E+001	-4.40E+001	-4.40E+001
porous1	4900	0		183	40		716.95	81.96	(Time)	1.61E-015	7.82E-013

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
porous2	4900	0		356	38		164.12	86	(Time)	2.39E-017	2.16E-020
portfl1	12	1	41	12	19	0.02	0.05	0.03	2.05E-002	2.05E-002	2.05E-002
portfl2	12	1	33	11	20	0.01	0.04	0.03	2.97E-002	2.97E-002	2.97E-002
portfl3	12	1	34	12	18	0.01	0.04	0.03	3.27E-002	3.28E-002	3.27E-002
portfl4	12	1	35	12	18	0.01	0.04	0.03	2.63E-002	2.63E-002	2.63E-002
portfl6	12	1	33	11	19	0.01	0.04	0.03	2.58E-002	2.58E-002	2.58E-002
powell20	1000	1000	501	55	27	2.38	2.13	0.94	5.21E+007	5.21E+007	5.21E+007
powellbs	2	0		43	87		0.03	0.07	(IL)	4.49E-006	2.15E-017
powellsq	2	0	11	17	13	0	0.02	0.01	1.62E-009	7.69E-016	8.56E-017
power	1000	0	2626	8	13	695.27	1.92	0.18	2.57E-016	4.02E-015	7.25E-020
probpenl	500	0	1003	6	17	6.19	0.56	22.92	3.99E-007	3.99E-007	-1.39E-005
prodpl0	60	29	88	22	70	0.03	0.09	0.13	6.09E+001	6.09E+001	6.09E+001
prodpl1	60	29	101	20	58	0.03	0.09	0.1	5.30E+001	5.30E+001	5.30E+001
pspdoc	4	0	14	17	11	0	0.03	0.01	2.41E+000	2.41E+000	2.41E+000
pt	2	501	1		16	0.04	13.85	0.09	1.78E-001	1.78E-001	1.78E-001
qpcboei1	372	288	2347		76	4.98	13.52	1.02	1.44E+007	1.44E+007	1.44E+007
qpcboei2	143	125	514	23	103	0.46		0.52	8.29E+006	(IL)	8.29E+006
qpcstair	385	356	775	362	309	2.66	24.12	6.42	6.20E+006	6.20E+006	6.20E+006
qpnboei1	372	288	2600	516		6.51	43.52		8.46E+006	8.51E+006	(IL)
qpnboei2	143	125	514	197	207	0.53	4.8	2.26	1.27E+006	1.27E+006	1.27E+006
qpnstair	385	356	823	232	453	3.12	21.56	21.32	5.15E+006	5.15E+006	5.15E+006

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
qr3dls	155	0		73	52		4.94	4.01	(IL)	1.06E-012	9.52E-021
qrtquad	120	0	340	34	343	1.01	0.13	3.01	-3.65E+006	-3.65E+006	-3.65E+006
quartc	10000	0		40	69		3.89	11.68	(Time)	3.69E-005	2.41E-018
qudlin	12	0	12	35	19	0	0.06	0.02	-7.20E+003	-7.20E+003	-7.20E+003
reading1	10001	5000			225			717.61	(Time)	ERROR	-1.60E-001
reading2	15001	10000	17172		24	996.02		10.93	-8.14E-003	ERROR	-1.26E-002
reading3	202	102	102	94	56	0.08	1.09	0.81	0.00E+000	-4.83E-007	-4.51E-013
recipe	3	0	35	16	21	0.01	0.02	0.02	1.74E-009	7.07E-009	2.42E-009
res	18	2	0	30	25	0	0.08	0.02	1.72E-028	1.28E-006	9.51E-009
rk23	17	11	22	33	13	0.01	0.05	0.02	8.33E-002	8.33E-002	8.33E-002
robot	7	2	57	6	17	0.03	0.02	0.02	3.85E+001	6.59E+000	6.59E+000
rosenbr	2	0	36	26	26	0	0.02	0.02	2.23E-013	5.37E-021	4.62E-020
rosenmmx	5	4	27	29	15	0	0.05	0.01	-4.40E+001	-4.40E+001	-4.40E+001
s201	2	0	6	2	10	0	0.01	0.01	6.31E-024	0.00E+000	2.77E-020
s202	2	0	26	6	13	0.01	0.01	0.01	4.90E+001	4.90E+001	4.90E+001
s203	5	3	13	4	23	0	0.02	0.03	8.39E-015	3.17E-019	1.48E-017
s204	2	0	13	4	8	0.01	0.01	0.01	1.84E-001	1.84E-001	1.84E-001
s205	2	0	17	8	13	0	0.01	0.02	5.74E-022	2.93E-022	3.51E-018
s206	2	0	15	4	12	0.01	0.01	0.01	8.38E-015	2.24E-013	2.32E-019
s207	2	0	11	8	12	0	0.01	0.01	1.19E-013	9.82E-019	4.58E-016
s208	2	0	36	26	26	0.01	0.02	0.02	2.23E-013	5.37E-021	4.62E-020

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
s209	2	0	104	122	88	0.01	0.07	0.07	1.36E-015	4.56E-016	3.43E-022
s210	2	0	366	728	0.03	0.37			8.47E-022	1.02E-012	(IL)
s211	2	0	35	38	34	0	0.03	0.03	1.06E-014	2.10E-016	7.33E-019
s212	2	0	16	11	11	0	0.01	0.01	2.99E-017	2.70E-015	5.12E-016
s213	2	0	119	27	33	0.01	0.02	0.02	1.65E-008	1.48E-008	4.09E-010
s214	2	0	61	0.09					ERROR	ERROR	1.87E-008
s215	2	1	2	17	25	0	0.03	0.02	0.00E+000	1.92E-006	1.64E-009
s216	2	1	16	10	24	0	0.02	0.04	9.99E-001	9.99E-001	9.99E-001
s217	2	2	2	15	19	0.01	0.03	0.02	-8.00E-001	-8.00E-001	-8.00E-001
s218	2	1	2	19	11	0	0.03	0.01	0.00E+000	2.56E-006	2.38E-009
s219	4	2	69	24	30	0.02	0.03	0.04	-1.00E+000	-1.00E+000	-1.00E+000
s220	2	1	0	38	8	0	0.05	0.01	1.00E+000	1.01E+000	1.00E+000
s221	2	1	2	15	0	0.03			-9.01E-001	-9.94E-001	(IL)
s222	2	1	2	19	12	0	0.03	0.02	-1.50E+000	-1.50E+000	-1.50E+000
s223	2	2	4	17	12	0.01	0.04	0.01	-8.34E-001	-8.34E-001	-8.34E-001
s224	2	2	5	11	12	0	0.03	0.01	-3.04E+002	-3.04E+002	-3.04E+002
s225	2	5	2	12	19	0.01	0.03	0.02	2.00E+000	2.00E+000	2.00E+000
s226	2	2	6	14	9	0.01	0.03	0	-5.00E-001	-5.00E-001	-5.00E-001
s227	2	2	3	12	11	0.01	0.03	0.01	1.00E+000	1.00E+000	1.00E+000
s228	2	2	3	16	10	0.01	0.03	0.01	-3.00E+000	-3.00E+000	-3.00E+000

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
s229	2	0	67	32	26	0.01	0.05	0.02	6.27E-016	1.64E-012	1.32E-018
s230	2	2	3	11	9	0	0.03	0.01	3.75E-001	3.75E-001	3.75E-001
s231	2	2	40	40	30	0.01	0.05	0.03	1.22E-013	2.37E-013	1.68E-019
s232	2	2	5	15	16	0	0.03	0.02	-1.00E+000	-1.00E+000	-1.00E+000
s233	2	1	37	17	13	0.01	0.03	0.01	2.12E+000	4.82E-012	1.35E-018
s234	2	1	0	25	18	0.01	0.04	0.01	-8.00E-001	-8.00E-001	-8.00E-001
s235	3	1	26	18	27	0	0.03	0.03	4.00E-002	4.00E-002	4.00E-002
s236	2	2	9	22	21	0	0.04	0.02	-5.89E+001	-5.89E+001	-5.89E+001
s237	2	3	16	31	352	0	0.05	1.03	-5.89E+001	-5.89E+001	-5.89E+001
s238	2	3	16	29	28	0.01	0.05	0.03	-5.89E+001	-5.89E+001	-5.89E+001
s239	2	1	21	16	15	0.01	0.03	0.02	-5.89E+001	-5.89E+001	-5.89E+001
s240	3	0	8	4	15	0	0.01	0.01	5.99E-015	1.39E-028	1.14E-015
s241	8	5	21	19	14	0.01	0.03	0.02	3.09E-013	1.73E-015	1.42E-011
s242	3	0	31	37	25	0	0.06	0.02	2.24E-016	1.43E-007	1.26E-009
s243	3	0	17	4	7	0	0.01	0	7.97E-001	7.97E-001	7.97E-001
s244	3	0	23	17	16	0	0.04	0.01	9.11E-015	1.71E-007	4.23E-008
s245	3	0	17	14	19	0.01	0.02	0.02	2.67E-012	4.98E-013	1.73E-016
s246	3	0	27	10	15	0	0.01	0.01	7.49E-016	2.00E-021	1.74E-019
s247	4	1	3	27	58	0.01	0.05	0.09	8.10E-015	3.42E-014	6.54E-018
s248	3	2	98	17	20	0.03	0.03	0.02	-8.00E-001	-8.00E-001	-8.00E-001
s249	3	1	13	16	10	0.01	0.03	0.01	1.00E+000	1.00E+000	1.00E+000

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
s250	3	1	3	10	16	0	0.03	0.01	-3.30E+003	-3.30E+003	-3.30E+003
s251	3	1	7	10	11	0	0.03	0.01	-3.46E+003	-3.46E+003	-3.46E+003
s252	3	1	24	36	21	0	0.05	0.02	4.00E-002	4.00E-002	4.00E-002
s253	3	1	9	27	15	0.01	0.05	0.02	6.93E+001	6.93E+001	6.93E+001
s254	3	2	13	10	22	0.01	0.02	0.03	-1.73E+000	-1.73E+000	-1.73E+000
s255	4	0	8	0				t	(Unb)	(IL)	-9.61E+001
s256	4	0	55	15	20	0.01	0.02	0.02	8.63E-013	4.62E-009	1.58E-010
s257	4	0	26	15	37	0	0.03	0.03	2.29E-014	1.50E-014	2.93E-018
s258	4	0	91	61	47	0.01	0.04	0.04	2.63E-016	8.14E-018	1.75E-020
s259	4	0	48	20	12	0.01	0.03	0.01	-8.54E+000	-8.54E+000	-8.54E+000
s260	4	0	92	61	47	0.01	0.04	0.04	2.63E-016	8.14E-018	1.75E-020
s261	4	0	56	12	19	0.01	0.02	0.01	1.11E-008	3.49E-008	1.02E-011
s262	4	4	4	11	13	0.01	0.03	0	-1.00E+001	-1.00E+001	-1.00E+001
s263	4	4	13	36	19	0.01	0.05	0.02	-1.00E+000	-1.00E+000	-1.00E+000
s264	4	3	12	11	11	0	0.03	0.01	-4.41E+001	-4.41E+001	-4.41E+001
s265	4	2	2	3	15	0	0.02	0.02	9.75E-001	1.90E+000	1.90E+000
s266	5	0	21	7	9	0.01	0.02	0.02	1.00E+000	1.00E+000	1.00E+000
s267	5	0	66	25	40	0.02	0.02	0.05	1.42E-013	1.50E-002	3.12E-016
s268	5	5	69	30	27	0.01	0.05	0.02	7.22E-012	2.56E-007	6.53E-009
s269	5	3	9	2	12	0	0.01	0.01	4.09E+000	4.09E+000	4.09E+000
s270	5	1	21	27	17	0.01	0.05	0.01	-1.00E+000	2.49E-007	2.19E-011

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
s271	6	0	16	4	11	0.01	0.01	0.01	1.17E-014	1.00E-014	1.44E-020
s272	6	0	72	266	53	0.01	0.2	0.07	5.66E-003	2.43E-001	7.71E-018
s273	6	0	43	11	17	0.01	0.01	0.02	1.49E-014	2.99E-017	4.89E-020
s274	2	0	6	3	9	0	0.01	0.01	7.91E-032	1.44E-034	9.90E-015
s275	4	0	12	3	9	0.01	0.01	0	1.40E-011	5.98E-012	4.31E-014
s276	6	0	16	3	9	0	0.01	0.01	2.92E-011	1.98E-012	2.46E-014
s277	4	4	6	18	13	0	0.04	0.01	5.08E+000	5.08E+000	5.08E+000
s278	6	6	7	18	13	0	0.04	0.01	7.84E+000	7.84E+000	7.84E+000
s279	8	8	10	22	14	0.01	0.05	0.01	1.06E+001	1.06E+001	1.06E+001
s280	10	10	10	24	16	0	0.06	0.01	1.34E+001	1.34E+001	1.34E+001
s281	10	0	73	0.15					ERROR	(IL)	1.82E-010
s282	10	0	154	15	65	0.03	0.02	0.06	1.15E-014	3.98E+000	1.14E-018
s283	10	0	575	33	43	0.12	0.03	0.04	9.29E-010	3.40E-009	4.45E-013
s284	15	10	121	30	27	0.08	0.1	0.05	-1.84E+003	-1.84E+003	-1.84E+003
s285	15	10	176	15	15	0.13	0.05	0.03	-8.25E+003	-8.25E+003	-8.25E+003
s286	20	0	81	27	26	0.03	0.02	0.03	1.14E-016	5.16E-014	4.62E-019
s287	20	0	145	56	47	0.07	0.04	0.05	2.56E-014	1.42E-015	5.49E-020
s288	20	0	61	15	20	0.04	0.02	0.02	6.24E-010	2.68E-008	7.89E-010
s289	30	0	34	5	10	0.01	0.01	0.02	7.53E-013	8.50E-009	2.72E-012
s290	2	0	6	2	8	0	0.01	0.01	1.03E-025	0.00E+000	4.06E-015
s291	10	0	26	4	9	0.01	0.01	0.01	1.75E-014	6.70E-013	2.83E-016

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
s292	30	0	62	5	9	0.06	0.01	0.02	8.26E-013	1.25E-014	2.51E-015
s293	50	0	93	5	9	0.16	0.01	0.02	1.11E-012	6.84E-014	6.96E-015
s294	6	0	56	21	24	0.01	0.02	0.03	4.29E-017	3.97E+000	3.97E+000
s295	10	0	80	34	30	0.02	0.03	0.03	1.62E-015	3.99E+000	3.99E+000
s296	16	0	129	42	39	0.05	0.04	0.04	9.54E-015	3.99E+000	3.99E+000
s297	30	0	90	81	67	0.08	0.08	0.08	9.47E-015	4.81E-014	2.44E-019
s298	50	0	112	126	98	0.22	0.14	0.17	3.70E-014	2.12E-013	4.29E-020
s299	100	0	216	222	174	0.84	0.39	0.47	1.10E-013	1.83E-015	2.44E-020
s300	20	0	54	3	13	0.03	0.01	0.01	-2.00E+001	-2.00E+001	-2.00E+001
s301	50	0	122	6	17	0.22	0.02	0.03	-5.00E+001	-5.00E+001	-5.00E+001
s302	100	0	5	19	0.02	0.05			(IL)	-1.00E+002	-1.00E+002
s303	20	0	78	11	16	0.03	0.02	0.02	1.60E-016	7.12E-030	2.67E-018
s304	50	0	100	15	21	0.12	0.03	0.07	1.52E-019	1.62E-020	5.52E-019
s305	100	0	339	19	25	1.44	0.08	0.27	8.46E-013	3.38E-026	5.69E-020
s307	2	0	11	14	0	0.01			1.24E+002	ERROR	1.24E+002
s308	2	0	11	9	13	0	0.01	0.01	7.73E-001	7.73E-001	7.73E-001
s309	2	0	10	6	13	0	0.01	0.02	-3.99E+000	2.89E-001	2.89E-001
s311	2	0	10	8	15	0	0.01	0.02	7.06E-016	1.54E-015	1.68E-020
s312	2	0	63	26	0	0.02			5.92E+000	ERROR	5.92E+000
s314	2	0	7	2	9	0	0.01	0.01	1.69E-001	1.69E-001	1.69E-001
s315	2	3	5	17	16	0	0.03	0.01	-8.00E-001	-8.00E-001	-8.00E-001

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
s316	2	1	7	6	16	0	0.02	0.02	3.34E+002	3.34E+002	3.34E+002
s317	2	1	12	6	15	0.01	0.02	0.02	3.72E+002	3.72E+002	3.72E+002
s318	2	1	15	6	18	0	0.02	0.03	4.13E+002	4.13E+002	4.13E+002
s319	2	1	15	7	16	0	0.02	0.02	4.52E+002	4.52E+002	4.52E+002
s320	2	1	18	7	17	0	0.02	0.01	4.86E+002	4.86E+002	4.86E+002
s321	2	1	21	7	18	0.01	0.02	0.01	4.96E+002	4.96E+002	4.96E+002
s322	2	1	28	22	0.01	0.03			5.00E+002	(IL)	5.00E+002
s323	2	2	10	18	10	0.01	0.04	0.01	3.80E+000	3.80E+000	3.80E+000
s324	2	2	15	15	20	0.01	0.03	0.02	5.00E+000	5.00E+000	5.00E+000
s325	2	3	5	11	11	0.01	0.03	0.01	3.79E+000	3.79E+000	3.79E+000
s326	2	2	3	11	12	0.01	0.03	0.01	-7.98E+001	-7.98E+001	-7.98E+001
s327	2	1	5	11	16	0	0.04	0.02	3.06E-002	3.06E-002	2.85E-002
s328	2	0	13	9	25	0	0.03	0.02	1.74E+000	1.74E+000	1.74E+000
s329	2	3	12	33	20	0.01	0.05	0.02	-6.96E+003	-6.96E+003	-6.96E+003
s330	2	1	15	14	14	0.01	0.03	0.02	1.62E+000	1.62E+000	1.62E+000
s331	2	1	11	9	8	0.01	0.03	0	4.26E+000	4.26E+000	4.26E+000
s332	2	1							ERROR	ERROR	(IL)
s333	3	0	5	13	0	0.02			5.61E+000	4.33E-002	(IL)
s334	3	0	27	10	17	0	0.01	0.01	8.21E-003	8.21E-003	8.21E-003
s335	3	2	55	36	29	0.02	0.04	0.04	-4.47E-003	-4.47E-003	-4.47E-003
s336	3	2	65	7	22	0.02	0.02	0.03	-3.38E-001	-3.38E-001	-3.38E-001

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
s337	3	1	15	18	14	0	0.03	0.02	6.00E+000	6.00E+000	6.00E+000
s338	3	2	19	9	22	0.01	0.02	0.03	-7.21E+000	-7.21E+000	-7.21E+000
s339	3	1	13	12	11	0	0.03	0.01	3.36E+000	3.36E+000	3.36E+000
s340	3	1	12	0.01					-5.40E-002	(IL)	(IL)
s341	3	1	17	13	11	0.01	0.03	0.01	-2.26E+001	-2.26E+001	-2.26E+001
s342	3	1	33	18	22	0.01	0.04	0.01	-2.26E+001	-2.26E+001	-2.26E+001
s343	3	2	7	29	27	0	0.05	0.02	-5.68E+000	-5.68E+000	-5.68E+000
s344	3	1	12	5	10	0.01	0.02	0.01	3.26E-002	3.26E-002	3.26E-002
s345	3	1	20	10	17	0	0.02	0.02	3.26E-002	3.26E-002	3.26E-002
s346	3	2	7	29	27	0	0.05	0.02	-5.68E+000	-5.68E+000	-5.68E+000
s347	6	4	7	24	40	0	0.05	0.01	1.76E+004	1.74E+004	1.76E+004
s348	3	1	6	303	0.01	0.75			3.70E+001	(IL)	3.70E+001
s350	4	0	29	13	11	0	0.02	0.02	3.08E-004	3.08E-004	3.08E-004
s351	4	0	95	13	22	0.01	0.02	0.02	3.19E+002	3.19E+002	3.19E+002
s352	4	0	14	6	12	0	0.01	0.01	9.03E+002	9.03E+002	9.03E+002
s353	4	3	2	12	23	0	0.03	0.02	-3.99E+001	-3.99E+001	-3.99E+001
s354	4	1	30	16	16	0	0.03	0.01	1.14E-001	1.14E-001	1.14E-001
s355	4	1	88	348	20	0.01	0.36	0.03	6.97E+001	6.97E+001	6.97E+001
s356	4	5	31	15	21	0.01	0.03	0.03	1.88E+000	1.88E+000	1.88E+000
s357	4	35	22	19	12	0.05	0.33	0.19	3.58E-001	3.58E-001	3.58E-001
s358	5	0	101	27	29	0.03	0.06	0.04	5.46E-005	5.46E-005	5.46E-005

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
s359	5	14	6	13	17	0	0.03	0.02	-5.50E+006	-8.54E+000	-5.50E+006
s360	5	2	20	8	30	0.01	0.02	0.04	-5.28E+006	-5.28E+006	-5.28E+006
s361	5	6	13	13	30	0.01	0.03	0.04	-1.53E+004	-1.53E+004	-1.53E+004
s365	7	5	25	36	42	0.01	0.06	0.05	5.21E+001	1.24E+002	5.21E+001
s365mod	7	5	25	28	26	0.01	0.05	0.04	5.21E+001	1.24E+002	5.22E+001
s366	7	14	23	20	37	0.01	0.05	0.06	1.23E+003	1.23E+003	1.23E+003
s367	7	5	71	13	34	0.02	0.03	0.03	-3.40E+001	-3.74E+001	-3.74E+001
s368	8	0	0	8	19	0	0.02	0.03	0.00E+000	0.00E+000	-2.84E-014
s369	8	6	34	218	19	0.01	0.32	0.03	7.05E+003	7.05E+003	7.05E+003
s370	6	0	79	12	17	0.02	0.02	0.02	2.29E-003	2.29E-003	2.29E-003
s371	9	0	134	13	18	0.06	0.02	0.03	6.68E-006	1.40E-006	1.40E-006
s372	9	12	40	42	35	0.02	0.08	0.05	1.34E+004	1.34E+004	1.34E+004
s373	9	6	20	14	30	0.01	0.03	0.04	1.34E+004	1.34E+004	1.34E+004
s374	10	35	222	156	0.24	0.52			2.33E-001	(IL)	2.33E-001
s375	10	9	17	9	23	0.01	0.03	0.04	-1.56E+001	-1.56E+001	-1.52E+001
s376	10	15	16	32	94	0.01	0.08	0.21	-4.43E+003	-4.43E+003	-4.43E+003
s377	10	3	7	15	140	0	0.04	0.22	-7.95E+002	-7.95E+002	-7.95E+002
s378	10	3	120	12	21	0.1	0.03	0.05	-4.77E+001	-4.78E+001	-4.78E+001
s379	11	0	103	21	28	0.08	0.07	0.13	4.01E-002	4.01E-002	8.76E-002
s380	12	3	246	241	0.2	0.46			3.17E+000	(IL)	3.17E+000
s381	13	4	14	11	15	0	0.03	0.01	1.01E+000	1.01E+000	1.01E+000

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
s382	13	4	29	15	17	0	0.04	0.02	1.04E+000	1.04E+000	1.04E+000
s383	14	1	64	10	33	0.01	0.03	0.03	7.29E+005	7.29E+005	7.29E+005
s384	15	10	157	14	15	0.1	0.05	0.03	-8.31E+003	-8.31E+003	-8.31E+003
s385	15	10	153	18	19	0.09	0.06	0.03	-8.31E+003	-8.31E+003	-8.31E+003
s386	2	0	6	2	10	0	0.01	0.01	6.31E-024	0.00E+000	2.77E-020
s387	15	11	173	18	21	0.13	0.06	0.03	-8.25E+003	-8.25E+003	-8.25E+003
s388	15	15	113	16	18	0.09	0.06	0.03	-5.82E+003	-5.82E+003	-5.82E+003
s389	15	15	110	14	19	0.1	0.06	0.03	-5.81E+003	-5.81E+003	-5.81E+003
s391	30	0						(Unb)	(IL)	(IL)	
s392	30	25	72	59	29	0.03	0.21	0.04	-1.10E+006	-1.10E+006	-1.10E+006
s393	48	3	277	51	75	1.24	0.37	0.27	8.63E-001	8.63E-001	8.74E-001
s394	20	1	125	27	19	0.22	0.04	0.03	1.92E+000	1.92E+000	1.92E+000
s395	50	1	211	31	22	3.1	0.07	0.07	1.92E+000	1.92E+000	1.92E+000
sawpath	589	782	1124	11	74	4.35	47.28	2.66	1.82E+002	1.82E+002	1.82E+002
scon1dls	1000	0			292			16.04	(IL)	(Time)	4.62E-020
scosine	10000	0			80			27.3	(Time)	(IL)	-1.00E+004
scurly10	10000	0			103			110.2	(Time)	(Time)	-1.00E+006
scurly20	10000	0			94			180.38	(Time)	(Time)	-1.00E+006

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
scurly30	10000	0			88			266.98	(Time)	(Time)	-1.00E+006
semicon1	1000	0			292			15.4	(IL)	(Time)	4.62E-020
semicon2	1000	0			70			3.22	(IL)	(Time)	1.49E-016
sensors	1000	0	1067		40	725.51		1779.02	-2.11E+005	ERROR	-2.02E+005
sim2bqp	2	0	4	11	14	0	0.03	0.01	0.00E+000	1.28E-006	6.31E-009
simbqp	2	0	5	13	13	0	0.03	0.01	9.63E-034	1.28E-006	2.07E-009
simpllpa	2	2	3	20	13	0	0.04	0.01	1.00E+000	1.00E+000	1.00E+000
simpllpb	2	3	1	20	13	0	0.04	0.01	1.10E+000	1.10E+000	1.10E+000
sineali	20	0			14			0.02	(IL)	(IL)	-1.90E+003
sineval	2	0	122	64	47	0.01	0.04	0.04	3.53E-019	1.50E-016	4.99E-018
sinquad	10000	0		217	70		45.8	54.26	(Time)	2.22E-006	6.17E-012
sinrosnb	1000	999	0		6	0.05		0.45	-9.99E+004	ERROR	-9.99E+004
sipow1	2	10000	850		17	55.34		4.63	-1.00E+000	ERROR	-1.00E+000
sipow1m	2	10000	851		16	55.04		4.56	-1.00E+000	ERROR	-1.00E+000
sipow2	2	5000	739		16	18.78	11.1	1.59	-1.00E+000	-1.00E+000	-1.00E+000
sipow2m	2	5000	739	22	16	18.75	10.75	1.56	-1.00E+000	-1.00E+000	-1.00E+000
sipow3	4	9998	75		19	6.33		5.69	5.36E-001	ERROR	5.36E-001
sipow4	4	10000	20		19	8.63		7.03	2.73E-001	ERROR	2.73E-001
sisser	2	0	15	13	16	0.01	0.02	0.02	2.69E-009	1.02E-008	8.71E-010
smbank	117	64	550	45	58	1.17	0.58	0.35	-7.13E+006	-7.13E+006	-7.13E+006
smmpsf	720	263			109			1.96	ERROR	(IL)	1.05E+006

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
snake	2	2	3		58	0.01		0.09	0.00E+000	(IL)	-3.63E-009
sosqp2	20000	10001			18			16.15	(Time)	ERROR	-5.00E+003
spanhyd	72	32	92		145	0.08		1.12	2.40E+002	ERROR	2.40E+002
spiral	3	2	161		217	0.03		0.26	-1.60E-010	(IL)	-2.19E-008
sreadin3	10000	5000			41			128.14	(Time)	ERROR	-7.31E-005
srosenbr	10000	0		26	28		1.99	4.94	(Time)	8.59E-009	1.38E-021
sseblin	192	72	228	214	11	0.07	1.97	0.04	1.62E+007	1.62E+007	1.62E+007
ssebnln	192	96	233		59	0.18		0.28	1.62E+007	ERROR	1.62E+007
ssnlbeam	31	20	54	14	63	0.03	0.05	0.13	3.42E+002	3.38E+002	3.38E+002
stancmin	3	2	5	81	17	0	0.09	0.01	4.25E+000	4.25E+000	4.25E+000
static3	434	96							(Unb)	(IL)	(P/D I)
steenbra	432	108	198	142	22	0.35	7.34	3.32	1.70E+004	1.70E+004	1.70E+004
steenbrb	468	108	7109		282	56.01		49.87	9.08E+003	(IL)	9.08E+003
steenbrc	540	126	1350			2.22			1.83E+004	(IL)	(IL)
steenbrd	468	108	5579		452	50.29	34.35	94.76	9.03E+003	9.03E+003	9.03E+003
steenbre	540	126	6410		220	80.18		132.04	2.75E+004	(IL)	2.75E+004
steenbrf	468	108	2922			4.27	9.09		3.19E+002	2.83E+002	(IL)
steenbrg	540	126	11376		223	101.11		60.87	2.84E+004	ERROR	2.74E+004
supersim	2	2	1	2	11	0	0.01	0.01	6.67E-001	6.67E-001	6.67E-001
svanberg	5000	5000			20			35.96	(Time)	ERROR	8.36E+003
swopf	82	91	130	22	23	0.24	0.13	0.09	6.79E-002	6.79E-002	6.79E-002

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
synthes1	6	6	13	20	15	0	0.04	0.01	7.59E-001	7.59E-001	7.59E-001
tame	2	1	3	2	24	0	0.01	0.04	1.00E-016	0.00E+000	0.00E+000
tfi2	3	10000	62		25	8.26		7.33	6.49E-001	ERROR	6.49E-001
tointqor	50	0	92	6	8	0.15	0.02	0.01	1.18E+003	1.18E+003	1.18E+003
trainf	20000	10002			81			428.39	(Time)	ERROR	3.10E+000
trainh	20000	10002			92			538.31	(Time)	ERROR	1.24E+001
tridia	10000	0		8	12		8.61	1.93	(Time)	5.24E-013	9.99E-019
trimloss	142	72	226	55	66	0.17	0.96	0.33	9.06E+000	9.06E+000	9.06E+000
try-b	2	1	1	15	27	0	0.03	0.04	1.00E+000	4.10E-013	5.33E-022
twirism1	343	313	5323		346	33.76		32.84	-1.01E+000	ERROR	-1.01E+000
twobars	2	2	11	14	10	0	0.03	0.01	1.51E+000	1.51E+000	1.51E+000
ubh1	17997	12000			35			25.56	(Time)	ERROR	1.12E+000
ubh5	19997	14000			285			801.69	(Time)	ERROR	1.12E+000
vanderm1	100	99	349	33	27	5.91	2.96	3.26	1.61E-009	8.34E-008	5.61E-014
vanderm2	100	99	336	33	27	4.92	3	3.25	5.98E-009	8.35E-008	5.61E-014
vanderm3	100	99	498	41	40	7.76	3.64	4.53	5.73E-009	1.00E-007	2.21E-014
vanderm4	9	8	80	30	28	0.03	0.06	0.03	4.22E-009	8.37E-008	3.79E-012
vardim	100	0	292	26	34	0.94	0.09	0.34	2.62E-011	2.07E-025	1.07E-020
watson	31	0	276	13	18	0.36	0.05	0.07	2.59E-007	1.14E-008	9.16E-013
womflet	3	3	11	19	13	0	0.04	0.02	-4.77E-018	3.84E-006	6.05E+000
woods	10000	0		63	48		6.13	11.24	(Time)	1.32E-014	4.47E-022

				Iterations			Solution Time			Objective Value	
Problem	n	m	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO	SNOPT	NITRO	LOQO
yao	2000	1999	2	20	202	0.13	2.46	12.12	2.73E+002	1.87E+002	1.98E+002
yfit	3	0	131	76	41	0.02	0.09	0.05	6.67E-013	2.20E-010	6.67E-013
yfitu	3	0	131	73	42	0.02	0.06	0.05	6.67E-013	1.41E-012	6.67E-013
zangwil2	2	0	4	2	9	0.01	0.01	0	-1.82E+001	-1.82E+001	-1.82E+001
zangwil3	3	0	8	4	15	0	0.01	0.01	5.99E-015	1.39E-028	1.14E-015
zecevic2	2	2	4	20	11	0.01	0.04	0.01	-4.13E+000	-4.12E+000	-4.13E+000
zecevic3	2	2	12		12	0		0.01	9.73E+001	ERROR	9.73E+001
zecevic4	2	2	9	21	15	0.01	0.04	0.01	7.56E+000	7.56E+000	7.56E+000
zigzag	58	50	102	39	28	0.04	0.14	0.07	3.16E+000	3.16E+000	3.16E+000
zy2	3	1	3	12	13	0	0.03	0.01	2.00E+000	2.00E+000	2.00E+000

Appendix B: Results for EPM Active Set Strategies

Table B.1 displays results for benchmarking the EPM solver with the active set strategy for primal varialbes on the CUTEr test set.

Table B.1: Benchmarking Results for Primal Active Set Strategy

Problem	Variables	Constraints	Iterations	Accuracy	% Active Primal	Sln Time(s)
aljazzaf.mod	3	4	38	8.47E-07	0.667	0
allinitc.mod	3	4	1012	1.45E-05	0.667	0.015
aalsotame.mod	2	5	15	2.82E-07	0.5	0
aug2dc.mod	20200	10194	62	9.48E-07	0.995	150
aug2dcqp.mod	20200	30196	88	2.19E-07	0.989	153
aug2dqp.mod	20192	30188	1071	0.991	0.99	2.95E+03
aug3dcqp.mod	3873	4873	19	4.33E-07	0.887	1.47
aug3dqp.mod	3873	4873	24	2.01E-07	0.93	2.03
avion2.mod	49	113	3	2.47E+04	0.796	0.015
batch.mod	46	161	566	0.251	0.326	0.093
biggsb1.mod	1000	1998	17	9.35E-07	0.998	0.093
blockqp1.mod	2005	5011	72	1.50E-11	0.501	3.62
blockqp2.mod	2005	5011	107	3.84E-07	0.501	5.91
blockqp3.mod	2005	5011	66	7.65E-07	0.501	3.95
blockqp4.mod	2005	5011	1011	0.0128	0.501	63.7
blockqp5.mod	2005	5011	101	7.66E-07	0.501	6.17
bloweya.mod	2002	3004	1027	0.127	0.83	20.9
bloweyb.mod	2002	3004	1045	0.00495	0.996	30.6
bloweyc.mod	2002	3004	248	3.54E-06	0.993	5.81
bqp1var.mod	1	2	9	1.55E-07	0	0
bqpgabim.mod	46	92	184	9.95E-07	0.957	0.046
bqpgasim.mod	50	100	188	9.52E-07	0.98	0.062
chenhark.mod	1000	1000	22	5.10E-07	0.794	0.109
clnlbeam.mod	1499	2996	1032	0.011	0.793	51.9
concon.mod	15	16	336	2.42E-09	0.733	0.015
core1.mod	65	165	1014	0.714	0.646	0.265
core2.mod	157	367	1017	3.60E+19	0.669	4.26
corkscrw.mod	8997	15000	420	2.49	0.689	376
csfi2.mod	5	9	75	5.59E-07	0.8	0
cvxbqp1.mod	10000	20000	3	4.40E-07	0	0.468

Problem	Variables	Constraints	Iterations	Accuracy	% Active Primal	Sln Time(s)
cvxqp1.mod	1000	2500	426	2.04E-09	0.614	5.58
cvxqp2.mod	10000	22500	401	4.18E-11	0.469	176
cvxqp3.mod	10000	27500	1019	0.0321	0.625	1.69E+03
dallasl.mod	837	2272	1	1.93E+05	0.995	0.015
dallasm.mod	164	447	1	8.95E+04	0.963	0
dallass.mod	44	117	1	7.99E+04	0.886	0
deconvb.mod	51	62	1000	0.00154	0.765	0.375
deconvc.mod	51	52	118	1.90E-07	0.804	0.062
degenlpa.mod	20	54	1004	0.0869	0.75	0.062
degenlpb.mod	20	55	1004	0.102	0.4	0.062
disc2.mod	28	35	1008	117	0.893	0.109
discs.mod	33	78	1045	5.66	0.879	0.234
dittert.mod	327	655	7	7.2	0.997	0.109
djtl.mod	exit code					
himmelp6.mod	2	8	7	2.44E-08	0	0
hong.mod	4	9	10	5.80E-09	0	0
hs002.mod	2	1	16	1.30E-12	0.5	0
hs004.mod	2	2	2	0	0	0
hs013.mod	2	3	1000	8.31E-05	0.5	0.015
hs015.mod	2	3	13	7.01E-08	0.5	0
hs016.mod	2	5	18	9.76E-07	0.5	0
hs020.mod	2	5	32	7.12E-07	0.5	0
hs021.mod	2	5	20	2.66E-07	0.5	0
hs025.mod	3	6	7	2.95	0.667	0
hs030.mod	3	7	15	1.64E-09	0.667	0
hs032.mod	3	5	6	1.43E-07	0.333	0
hs033.mod	3	6	20	3.56E-08	0.667	0
hs034.mod	3	8	15	1.92E-07	0.667	0
hs041.mod	4	9	1000	2	0.75	0.015
hs044.mod	4	10	24	8.21E-09	0.5	0

Problem	Variables	Constraints	Iterations	Accuracy	% Active Primal	Sln Time(s)
hs055.mod	6	14	37	3.19E-08	0.667	0
hs067.mod	10	27	1097	1.67	0.6	0.093
hs071.mod	4	10	1000	850	0.5	0.015
hs073.mod	4	7	33	5.94E-08	0.75	0
hs074.mod	4	12	1060	39.1	0.75	0.031
hs075.mod	4	12	1023	401	0.75	0.031
hs076.mod	4	7	19	1.33E-08	0.75	0
hs083.mod	5	13	63	2.64E-07	0.4	0
hs085.mod	5	46	1021	0.0019	0.6	0.281
hs087.mod	9	22	60	8.57E-07	0.556	0
hs095.mod	6	16	60	2.66E-08	0.167	0
hs096.mod	6	16	58	2.44E-08	0.167	0
hs097.mod	6	16	1000	0.37	0.167	0.031
hs098.mod	6	16	1000	0.37	0.167	0.031
hs106.mod	8	22	1000	1.11	0.875	0.046
hs107.mod	9	14	474	6.08E-11	0.778	0.015
hs109.mod	9	26	1049	4.44E+04	0.667	0.078
hs114.mod	10	31	1083	0.593	0.7	0.046
hs116.mod	13	41	1028	0.0245	0.615	0.062
hs117.mod	15	20	762	9.04E+05	0.6	0.078
hs118.mod	15	47	31	6.28E-08	0.8	0.015
hs119.mod	16	40	116	7.68E-07	0.75	0.015
hs21mod.mod	7	9	27	5.04E-07	0.714	0
hs44new.mod	4	9	21	2.21E-07	0.5	0
hues-mod.mod	10000	10002	54	6.96E-10	0.945	10.6
huestis.mod	10000	10002	215	1.79E-10	0.945	54.6
hvycrash.mod	201	352	1087	0.0176	0.811	4.51
linspanh.mod	72	176	42	4.72E-07	0.875	0.015
loadbal.mod	31	73	42	1.54E-08	0.677	0
lootsma.mod	3	6	20	3.56E-08	0.667	0
lotschd.mod	12	19	11	3.93E-08	0.583	0
lsnnodoc.mod	5	10	14	7.01E-09	0.4	0

Problem	Variables	Constraints	Iterations	Accuracy	% Active Primal	Sln Time(s)
manne.mod	1094	2189	11	0.0497	0.645	0.156
matrix2.mod	6	6	15	6.26E-07	0.667	0
mconcon.mod	15	16	336	2.42E-09	0.733	0.015
mdhole.mod	2	1	36	0	0.5	0
minc44.mod	303	621	1004	0.005	0.713	7.25
minperm.mod	1113	2246	8	9.01	0.999	2.89
model.mod	60	152	1083	10.1	0.583	0.406
mosarqp1.mod	2500	3200	104	4.43E-07	0.558	2.89
mosarqp2.mod	900	1500	60	9.19E-07	0.879	0.734
ncvxbqp1.mod	10000	20000	103	0.858	0.959	336
ncvxbqp2.mod	10000	20000	16	0.999	0.75	35.8
ncvxbqp3.mod	10000	20000	103	0.998	0.375	27.1
ncvxqp1.mod	1000	2500	10	419	0.596	0.125
ncvxqp2.mod	1000	2500	14	295	0.599	0.578
ncvxqp3.mod	1000	2500	103	157	0.586	2.55
ncvxqp4.mod	1000	2250	103	102	0.717	2.88
ncvxqp5.mod	1000	2250	18	295	0.575	0.156
ncvxqp6.mod	1000	2250	16	98.9	0.63	0.203
ncvxqp7.mod	1000	2750	103	1.78E+03	0.613	3.41
ncvxqp8.mod	1000	2750	8	153	0.67	0.187
ncvxqp9.mod	1000	2750	103	104	0.581	3.16
ngone.mod	97	1371	1027	0.0322	0.948	19.8
obstclal.mod	64	128	19	9.72E-07	0.594	0.015
obstclbl.mod	64	128	38	1.69E-07	0.391	0.015
obstclbu.mod	64	128	17	2.01E-07	0.25	0.015
optcdeg2.mod	1198	1998	1011	0.21	0.653	13.9
optcdeg3.mod	1198	1998	1015	0.0503	0.889	31.6
optcntrl.mod	28	50	296	6.79E-09	0.643	0.046
optprloc.mod	30	89	1031	0.00394	0.7	0.125
oslbqp.mod	8	11	14	6.76E-07	0.75	0.015
palmer3.mod	4	3	103	882	0.75	0
palmer4.mod	4	3	108	6.33E-09	0.75	0

Problem	Variables	Constraints	Iterations	Accuracy	% Active Primal	Sln Time(s)
pentagon.mod	6	15	63	4.35E-07	0.833	0
pentdi.mod	1000	1000	13	5.64E-07	0.496	0.062
portfl3.mod	12	25	24	4.36E-07	0.833	0
prodpl0.mod	60	89	70	1.13E-07	0.467	0.015
prodpl1.mod	60	89	70	2.02E-09	0.433	0.015
pspdoc.mod	4	1	8	9.60E-10	0.75	0
qpcboei1.mod	372	846	813	4.47E-11	0.613	3.45
qpcboei2.mod	143	322	386	3.50E-07	0.72	0.593
qpcstair.mod	385	741	1027	22.8	0.678	8.09
qpnboei1.mod	372	846	370	1.42E+03	0.589	1.2
qpnboei2.mod	143	322	1025	1.19E+07	0.888	2.66
qpnstair.mod	385	741	365	22.7	0.571	2.61
qrtquad.mod	120	20	24	1.08E-08	0.967	0.015
qudlin.mod	12	24	27	1.63E-10	0.083	0
reading3.mod	202	506	582	1.53E-08	0.53	1.59
res.mod	18	38	27	4.92E-07	0.889	0
rk23.mod	17	17	65	2.29E-07	0.706	0
s220.mod	2	3	3	0	0	0
s221.mod	2	3	1000	3.48E-05	0.5	0.015
s222.mod	2	3	9	9.21E-07	0.5	0
s232.mod	2	4	6	4.19E-08	0.5	0
s234.mod	2	5	8	9.20E-08	0	0
s236.mod	2	6	12	2.10E-08	0	0
s237.mod	2	6	77	6.34E-11	0	0
s242.mod	3	6	6	3.02	0.667	0
s249.mod	3	2	17	1.50E-12	0.667	0
s252.mod	3	2	12	6.69E-07	0.667	0
s262.mod	4	8	17	1.29E-08	0.5	0
s265.mod	4	10	108	0.000479	0.5	0
s339.mod	3	4	2	1.32	0.667	0.015
s348.mod	3	5	1012	590	0.667	0.015
s353.mod	4	7	8	1.46E-07	0.5	0

Problem	Variables	Constraints	Iterations	Accuracy	% Active Primal	Sln Time(s)
s355.mod	4	5	164	6.68E-10	0.5	0
s356.mod	4	8	2	10.9	0.75	0
s360.mod	5	11	527	6.22E-07	0.2	0.031
s361.mod	5	14	6	7.56E-07	0.8	0
s365.mod	7	9	1000	6.96	0.857	0.015
s365mod.mod	7	9	1000	1	0.714	0.031
s376.mod	10	35	1	0	0.5	0
s381.mod	13	17	37	9.29E-08	0.615	0
s382.mod	13	17	38	3.30E-07	0.615	0
s383.mod	14	29	1000	0.234	0	0.046
s392.mod	30	70	20	3.03E-07	0.633	0
s393.mod	48	75	58	6.73E-07	0.708	0.031
scon1dls.mod	1000	2000	35	0.439	0.999	0.296
semicon1.mod	1000	2000	35	0.439	0.999	0.312
sim2bqp.mod	2	2	9	2.89E-08	0.5	0.015
simbqp.mod	2	2	9	2.89E-08	0.5	0
simpllpa.mod	2	4	16	8.80E-07	0.5	0.015
sineali.mod	20	40	21	3.04E-07	0.95	0
smmpsf.mod	720	983	1036	40.7	0.643	12
sosqp1.modsosqp2.mod	20000	50001	1017	0.0016	0.789	5.93E+03
spanhyd.mod	72	176	1089	710	0.694	0.671
sreadin3.mod	10000	25000	647	6.27E-13	0.5	502
sseblin.mod	192	432	1039	15.8	0.526	2.61
ssebnln.mod	192	456	1073	15.8	0.516	3.2
ssnlbeam.mod	31	60	1026	0.000568	0.548	0.203
stancmin.mod	3	5	20	4.56E-07	0.667	0
static3.mod	434	240	114	8.86E-07	0.995	0.453
steenbra.mod	432	540	42	2.43E-07	0.169	0.187
steenbrd.mod	468	576	1	3.46E+03	0.996	0.062
steenbre.mod	540	666	1	1.00E+04	0.996	0.078
swopf.mod	82	111	55	6.46E-07	0.963	0.078
synthes1.mod	6	18	35	1.31E-07	0.667	0

Problem	Variables	Constraints	Iterations	Accuracy	% Active Primal	Sln Time(s)
trainh.mod	20000	30002	1025	0.00999	0.75	2.99E+03
trimloss.mod	142	336	1	234	0.169	0
twirism1.mod	343	999	1015	15.6	0.51	38.4
zigzag.mod	58	110	76	4.33E-07	0.966	0.031
zy2.mod	3	5	11	2.88E-07	0.333	0

Table B.2 displays results for benchmarking the EPM solver with the active set strategy for dual varialbes on the CUTEr test set.

Table B.2: Benchmarking Results for Dual Active Set Strategy

Problem	Variables	Constraints	Iterations	Accuracy	% Active Dual	Sln Time(s)
avgasa.mod	6	18	8	1.74E-07	0.667	0
avgasb.mod	6	18	10	2.26E-07	0.5	0
batch.mod	46	161	536	4.37E-11	0.281	0.187
biggsc4.mod	4	15	40	2.24E-07	0.308	0
cb2.mod	3	3	13	2.74E-07	0.667	0
chaconn1.mod	3	3	10	1.22E-07	0.667	0
congigmz.mod	3	5	29	2.58E-07	0.6	0
core1.mod	65	165	108	1.93E-07	0.667	0.031
corkscrw.mod	8997	15000	1067	0.15	0	918
coshfun.mod	61	20	379	0.0483	0.4	0.14
csfi1.mod	5	10	90	5.36E-08	0.333	0
csfi2.mod	5	9	73	5.77E-07	0.333	0
disc2.mod	28	35	350	3.21E-07	0.333	0.031
discs.mod	33	78	1058	3.09	0.292	0.218
dualc1.mod	9	31	75	3.19E-07	0	0.015
dualc2.mod	7	23	90	4.53E-07	0	0.015
dualc8.mod	8	31	113	5.15E-07	0	0
expfita.mod	5	21	48	8.40E-08	0.19	0.015
expfitb.mod	5	101	63	8.20E-07	0.0396	0
expfitc.mod	5	501	144	9.85E-07	0.00998	0.406
fletcher.mod	4	5	26	5.26E-09	0.333	0
gausselm.mod	1495	4215	9	3.31E-09	0.0429	0.75
gpp.mod	250	498	114	9.38E-07	0.229	7.38
hadamard.mod	65	257	9	2.44E-07	0.667	0.14
haifas.mod	7	9	17	4.53E-07	0.333	0
haldmads.mod	6	42	118	7.12E-07	0.167	0.015
hatfldh.mod	4	15	34	2.24E-07	0.308	0
himmelbi.mod	100	112	18	0.00343	0.0833	0.015
himmelp4.mod	2	7	7	7.93E-11	0	0
himmelp6.mod	2	8	11	3.39E-07	0	0.015
hs015.mod	2	3	25	2.28E-08	0.5	0

Problem	Variables	Constraints	Iterations	Accuracy	% Active Dual	Sln Time(s)
hs016.mod	2	5	20	7.97E-07	0	0
hs018.mod	2	6	59	2.74E-07	0.5	0
hs020.mod	2	5	63	7.55E-09	0.333	0
hs021.mod	2	5	27	4.67E-08	0	0
hs023.mod	2	9	25	2.77E-07	0.4	0
hs029.mod	3	1	13	4.26E-07	0	0.015
hs032.mod	3	5	13	7.48E-07	0	0
hs036.mod	3	7	15	3.73E-07	0	0
hs037.mod	3	7	15	3.73E-07	0	0
hs043.mod	4	3	10	4.45E-07	0.667	0
hs044.mod	4	10	16	1.14E-07	0.333	0
hs059.mod	2	7	12	1.84E-08	0.333	0
hs074.mod	4	12	40	3.15E-08	0.5	0
hs075.mod	4	12	43	4.69E-08	0.5	0
hs076.mod	4	7	20	1.08E-07	0.667	0
hs083.mod	5	13	32	5.04E-07	0.333	0.015
hs085.mod	5	46	1040	4.75E+04	0.417	0.14
hs086.mod	5	11	23	5.92E-08	0.667	0
hs095.mod	6	16	58	6.31E-07	0.25	0
hs096.mod	6	16	52	6.31E-07	0.25	0
hs097.mod	6	16	1022	0.122	0.25	0.031
hs098.mod	6	16	1026	0.122	0.25	0.015
hs109.mod	9	26	1013	1.02E+03	0.5	0.063
hs113.mod	10	8	9	2.72E-07	0.75	0
hs114.mod	10	31	89	1.14E-07	0.5	0
hs116.mod	13	41	268	9.32E-07	0.733	0.015
hs118.mod	15	47	31	6.29E-08	0.414	0
hs21mod.mod	7	9	21	4.79E-07	0	0
hs44new.mod	4	9	21	1.73E-07	0.4	0
ksip.mod	20	1001	342	9.97E-07	0.018	6.76
liswet12.mod	10002	10000	1045	1.57E-06	0.976	1.22E+03
loadbal.mod	31	73	22	4.72E-07	0	0
madsen.mod	3	6	49	1.76E-08	0.5	0

Problem	Variables	Constraints	Iterations	Accuracy	% Active Dual	Sln Time(s)
madsschj.mod	81	158	405	8.64E-07	0.513	3.94
makela2.mod	3	3	34	2.57E-07	0.667	0
minmaxbd.mod	5	20	218	9.35E-08	0.15	0.015
mistake.mod	9	14	168	6.35E-07	0.462	0
model.mod	60	152	1034	0.555	0.833	0.39
mosarqp1.mod	2500	3200	78	8.34E-07	0.486	2.72
mosarqp2.mod	900	1500	51	1.38E-07	0.367	0.375
oet1.mod	3	1002	75	7.34E-07	0.013	0.281
oet2.mod	3	1002	78	8.92E-07	0.013	0.421
oet3.mod	4	1002	341	9.96E-07	0.0309	0.546
oet7.mod	7	1002	634	9.90E-07	0.278	59.2
optcntrl.mod	28	50	90	3.73E-08	0	0.015
optctrl3.mod	118	80	303	9.96E-07	0	0.609
optctrl6.mod	118	80	303	9.96E-07	0	0.609
optmass.mod	66	55	7	2.56E-09	0	0.015
optprloc.mod	30	89	269	5.33E-08	0.724	0.031
pentagon.mod	6	15	28	5.20E-07	0.417	0
polak3.mod	12	10	103	2.46E-07	0.3	0.015
polak6.mod	5	4	68	2.78E-07	0.75	0.015
powell20.mod	1000	1000	1009	37	0.704	12.4
prodpl0.mod	60	89	51	4.17E-07	0.667	0.015
prodpl1.mod	60	89	69	5.90E-08	0.667	0.031
pt.mod	2	501	142	9.86E-07	0.014	0.125
qpcboei1.mod	372	846	163	3.27E-07	0.372	0.656
qpcboei2.mod	143	322	283	1.49E-08	0.5	0.5
qpcstair.mod	385	741	563	1.29E-08	0.32	7.69
qpnboei1.mod	372	846	266	9.67E-05	0.997	1.28
qpnboei2.mod	143	322	528	7.97	0.721	1.01
qpnstair.mod	385	741	59	24.6	0.483	1.09
res.mod	18	38	11	5.48E-07	0	0
rosenmmx.mod	5	4	28	2.78E-07	0.75	0
s223.mod	2	6	39	6.16E-07	0.5	0
s224.mod	2	6	12	3.21E-07	0.5	0

Problem	Variables	Constraints	Iterations	Accuracy	% Active Dual	Sln Time(s)
s225.mod	2	5	25	2.77E-07	0.4	0
s226.mod	2	4	22	1.64E-08	0.5	0.015
s228.mod	2	2	14	2.12E-08	0.5	0
s232.mod	2	4	9	6.45E-07	0	0
s234.mod	2	5	11	5.66E-07	0	0
s236.mod	2	6	44	5.10E-09	0	0
s237.mod	2	6	33	4.65E-08	0	0
s238.mod	2	5	32	2.00E-08	0	0.015
s239.mod	2	5	9	2.94E-07	0	0
s250.mod	3	7	15	3.73E-07	0	0
s251.mod	3	7	15	3.73E-07	0	0.015
s262.mod	4	8	17	2.52E-09	0.333	0
s264.mod	4	3	16	7.60E-07	0.667	0
s270.mod	5	5	11	5.33E-07	0	0
s284.mod	15	10	31	1.46E-07	0.2	0
s315.mod	2	3	23	1.79E-07	0.667	0
s323.mod	2	4	8	5.29E-07	0.5	0
s324.mod	2	3	59	1.92E-07	0.5	0
s327.mod	2	3	14	9.73E-08	0	0
s329.mod	2	7	45	4.76E-10	0.667	0
s332.mod	2	5	102	55	0.5	0.093
s341.mod	3	4	13	4.28E-07	0	0
s353.mod	4	7	8	1.73E-07	0.5	0
s356.mod	4	8	213	10.9	0.8	0.015
s357.mod	4	43	7	1.38E-07	0	0.015
s359.mod	5	14	71	4.05E-08	0.357	0
s360.mod	5	11	338	0	0.5	0.015
s361.mod	5	14	904	2.04E-07	0	0.046
s365mod.mod	7	9	66	4.40E-07	0.8	0
s366.mod	7	28	96	9.40E-07	0.429	0
s367.mod	7	12	32	2.50E-07	0.667	0.015
s374.mod	10	35	104	2.00E-08	0.171	0.015
s382.mod	13	17	39	3.30E-07	0.667	0.015

Problem	Variables	Constraints	Iterations	Accuracy	% Active Dual	Sln Time(s)
s384.mod	15	10	19	2.58E-07	0.9	0
s385.mod	15	10	63	9.15E-07	0.8	0
s388.mod	15	15	32	2.10E-08	0.667	0
s389.mod	15	15	32	1.25E-08	0.667	0
s392.mod	30	70	21	2.71E-09	0.72	0
sawpath.mod	589	782	1011	0.000732	0.245	8.17
simpllpb.mod	2	5	19	1.02E-08	0.667	0
sipow1.mod	2	10000	135	9.86E-07	0.0022	9.77
sipow1m.mod	2	10000	133	9.84E-07	0.0022	9.73
sipow2.mod	2	5001	102	1.00E-06	0.0022	2.88
sipow2m.mod	2	5001	96	9.87E-07	0.0024	2.86
sipow3.mod	4	9999	373	9.86E-07	0.0016	9.53
sipow4.mod	4	10000	383	9.94E-07	0.0051	10.2
smmpsf.mod	720	983	1047	6.46	0.957	12.3
svanberg.mod	5000	15000	361	9.97E-07	0.806	168
swopf.mod	82	111	53	1.44E-07	0	0.078
synthes1.mod	6	18	10	3.89E-07	0.5	0
tfi2.mod	3	10001	113	9.72E-07	0.0114	18.2
twobars.mod	2	6	16	3.65E-08	0.5	0
vanderm4.mod	9	8	19	3.47E-07	0	0
yao.mod	2000	2001	1044	0.00393	0.999	54.7
zecevic2.mod	2	6	7	2.52E-07	0.5	0
zecevic3.mod	2	6	17	2.96E-07	0.5	0
zecevic4.mod	2	6	14	5.22E-07	0.5	0
zigzag.mod	58	110	61	4.35E-07	0.85	0.031
zy2.mod	3	5	13	2.51E-07	0	0

Bibliography

Bibliography

[1] V. Bloom, I. Griva, B. Kwon, and A.-R. Wolff, “Exterior-point method for support vector machines,” *IEEE Transactions on Neural Networks and Learning Systems*, to appear.

[2] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization, Second Edition*, 2nd ed. Society for Industrial Mathematics, Dec. 2008.

[3] J. Nocedal and S. Wright, *Numerical Optimization.* Springer, 2000.

[4] A. S. Bondarenko, D. M. Bortz, and J. J. Mor, *COPS: Large-scale nonlinearly constrained optimization problems*, 2000. [Online]. Available: http://www.osti.gov/energycitations/servlets/purl/751934-2BmkFr/native/

[5] E. D. Dolan, J. J. Mor, and T. S. Munson, “Benchmarking optimization software with COPS 3.0,” Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, Illinois, Technical Report ANL/MCS-TM-273, 2004, 00078.

[6] I. Griva and R. A. Polyak, “1.5-q-superlinear convergence of an exterior-point method for constrained optimization,” *Journal of Global Optimization*, vol. 40, no. 4, p. 679695, 2008. [Online]. Available: http://www.springerlink.com/content/323465620v133776/

[7] W. Karush, “Minima of functions of several variables with inequalities as side constraints,” M. Sc. Dissertation, Dept. of Mathematics, University of Chicago, Chicago, Illinois, 1939.

[8] H. Kuhn and A. W. Tucker, “Nonlinear programming,” in *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, J. Neyman, Ed. Berkeley, CA: University of California Press, 1951, p. 481492.

[9] C. R, “Variational methods for the solution of problems of equilibrium and vibrations,” *Bulletin of American Mathematical Society*, vol. 49, p. 123, 1943, 01256. [Online]. Available: http://ci.nii.ac.jp/naid/10015583032/

[10] K. Frisch, “The logarithmic potential method of convex programming,” Tech. Rep., 1955, 00321.

[11] C. W. Carroll, “The created response surface technique for optimizing nonlinear, restrained systems,” *OPERATIONS RESEARCH*, vol. 9, no. 2, p. 169184, Mar. 1961. [Online]. Available: http://or.journal.informs.org/cgi/content/abstract/9/2/169

[12] A. V. Fiacco and G. P. McCormick, *Nonlinear programming: sequential unconstrained minimization techniques.* SIAM, 1990.

[13] M. H. Wright, “Interior methods for constrained optimization,” *Acta Numerica*, vol. 1, p. 341407, 1992. [Online]. Available: http://journals.cambridge.org/action/displayAbstract?fromPage=onlineaid=1771092

[14] M. R. Hestenes, “Multiplier and gradient methods,” *Journal of Optimization Theory and Applications*, vol. 4, no. 5, p. 303320, Nov. 1969. [Online]. Available: http://www.springerlink.com.mutex.gmu.edu/content/r1r316w078l36u06/

[15] M. Powell, "A method for nonlinear constraints in minimization problems," in *Optimization*, R. Fletcher, Ed. New York: Academic Press, 1969, p. 283298.

[16] R. Polyak and M. Teboulle, "Nonlinear rescaling and proximal-like methods in convex optimization," *Mathematical Programming*, vol. 76, no. 2, p. 265284, Feb. 1997, 00090. [Online]. Available: http://www.springerlink.com.mutex.gmu.edu/content/2112135w13045t17/

[17] R. Polyak, "Modified barrier functions (theory and methods)," *Mathematical Programming*, vol. 54, no. 1-3, p. 177222, Feb. 1992, 00285. [Online]. Available: http://www.springerlink.com.mutex.gmu.edu/content/u773j5v570095072/

[18] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, 1st ed. Athena Scientific, Jan. 1996, 00000.

[19] A. Conn, G. Gould, and P. Toint, *Lancelot: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, 1st ed. Springer, Aug. 1992, 00003.

[20] I. Griva and R. A. Polyak, "Primal-dual nonlinear rescaling method with dynamic scaling parameter update," *Mathematical Programming*, vol. 106, no. 2, p. 237259, 2006, 00040. [Online]. Available: http://www.springerlink.com/content/v0kw5437856g364w/

[21] R. J. Vanderbei, "LOQO: an interior point code for quadratic programming," *OPTIMIZATION METHODS AND SOFTWARE*, vol. 12, 00500. [Online]. Available: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.2191

[22] R. J. Vanderbei and D. F. Shanno, "An interior-point algorithm for nonconvex nonlinear programming," *Computational Optimization and Applications*, vol. 13, p. 231252, Apr. 1999, ACM ID: 316355. [Online]. Available: http://portal.acm.org.mutex.gmu.edu/citation.cfm?id=316300.316355

[23] R. H. Byrd, M. E. Hribar, and J. Nocedal, *An Interior Point Algorithm for Large Scale Nonlinear Programming*, 1998, 00000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.39.4793

[24] J. Nocedal, A. Wachter, and R. A. Waltz, "Adaptive barrier strategies for nonlinear interior methods," IBM T.J. Watson Research Center, Yorktown, USA, Research Report RC23563, 2005.

[25] A. Wachter and L. T. Biegler, "Line search filter methods for nonlinear programming: Motivation and global convergence," *SIAM Journal on Optimization*, vol. 16, no. 1, p. 1, 2005, 00219. [Online]. Available: http://link.aip.org.mutex.gmu.edu/link/SJOPE8/v16/i1/p1/s1Agg=doi

[26] N. I. M. Gould, D. Orban, and P. L. Toint, "GALAHAD, a library of thread-safe fortran 90 packages for large-scale nonlinear optimization," *ACM Transactions on Mathematical Software*, vol. 29, no. 4, p. 353372, Dec. 2003, 00145. [Online]. Available: http://academic.research.microsoft.com/Publication/805014/galahad-a-library-of-thread-safe-fortran-90-packages-for-large-scale-nonlinear-optimization

[27] ——, "Results from a numerical evaluation of LANCELOT b," Rutherford Appleton Laboratory, Chilton, England, Tech. Rep. 2002-1, 2002.

[28] B. A. Murtagh and M. A. Saunders, "MINOS 5.51 user's guide," Stanford University, Technical Report SOL 83-20R, 1983, 00000.

[29] P. E. Gill, W. Murray, and M. A. Saunders, "User's guide for SNOPT version 7: Software for large-scale nonlinear programming," Tech. Rep., 2008.

[30] ——, "SNOPT: an SQP algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, p. 979, 2002. [Online]. Available: http://link.aip.org.mutex.gmu.edu/link/SJOPE8/v12/i4/p979/s1Agg=doi

[31] ——, “SNOPT: an SQP algorithm for large-scale constrained optimization,” *SIAM Review*, vol. 47, no. 1, p. 99131, Mar. 2005, ArticleType: research-article / Full publication date: Mar., 2005 / Copyright 2005 Society for Industrial and Applied Mathematics. [Online]. Available: http://www.jstor.org.mutex.gmu.edu/stable/20453604

[32] ——, “User’s guide for SQOPT version 7: Software for large-scale linear and quadratic programming,” Tech. Rep., 2006, 00008.

[33] A. S. Drud, “CONOPTA large-scale GRG code,” *INFORMS JOURNAL ON COMPUTING*, vol. 6, no. 2, p. 207216, Jan. 1994, 00356. [Online]. Available: http://joc.journal.informs.org/cgi/content/abstract/6/2/207

[34] P. Spellucci, “DONLP2 user’s guide,” Technical University at Darmstadt, Department of Mathematics, Darmstadt, Germany, Tech. Rep.

[35] J. F. Bonnans, E. R. Panier, A. L. Tits, and J. L. Zhou, “Avoiding the maratos effect by means of a nonmonotone line search II. inequality constrained ProblemsFeasible iterates,” *SIAM Journal on Numerical Analysis*, vol. 29, no. 4, p. 11871202, 1992, 00100 ArticleType: research-article / Full publication date: Aug., 1992 / Copyright 1992 Society for Industrial and Applied Mathematics. [Online]. Available: http://www.jstor.org.mutex.gmu.edu/stable/2157999

[36] E. R. Panier and A. L. Tits, “On combining feasibility, descent and superlinear convergence in inequality constrained optimization,” *Mathematical Programming*, vol. 59, no. 1-3, p. 261276, Mar. 1993. [Online]. Available: http://www.springerlink.com.mutex.gmu.edu/content/pvr216552xq7l258/

[37] J. L. Zhou and A. L. Tits, “Nonmonotone line search for minimax problems,” *Journal of Optimization Theory and Applications*, vol. 76, no. 3, p. 455476, Mar. 1993, 00104. [Online]. Available: http://www.springerlink.com.mutex.gmu.edu/content/r561m32096064411/

[38] ——, “An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions,” Technical Report ISR; TR 1993-83, 1993, 00000. [Online]. Available: http://drum.lib.umd.edu/handle/1903/5427

[39] J. L. Zhou, A. L. Tits, and C. T. Lawrence, “User’s guide for FFSQP version 3.7: A FORTRAN code for solving constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints,” Electrical Engineering Department and Institute for Systems Research, University of Maryland, College Park, MD, Tech. Rep., 00000.

[40] C. T. Lawrence, J. L. Zhou, and A. L. Tits, “User’s guide for CFSQP version 2.5: A c code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints,” Electrical Engineering Department and Institute for Systems Research, University of Maryland, College Park, MD, Tech. Rep.

[41] C. T. Lawrence and A. L. Tits, “Nonlinear equality constraints in feasible sequential quadratic programming,” *Optimization Methods and Software*, vol. 6, no. 4, p. 265, 1996, 00078. [Online]. Available: http://www.informaworld.com/10.1080/10556789608805638

[42] R. Byrd, J. Nocedal, and R. A. Waltz, “KNITRO: an integrated package for nonlinear optimization,” in *Large-scale nonlinear optimization*, G. D. Pillo and M. Roma, Eds. Birkhuser, 2006.

[43] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, “Some theoretical properties of an augmented lagrangian merit function,” Technical Report SOL 86-6R, 1986, 00086.

[44] ——, “User’s guide for NPSOL (version 5.0): A fortran package for nonlinear programming,” Technical Report SOL 86-6, 2001.

[45] J. J. Mor and S. J. Wright, *Optimization Software Guide.* Society for Industrial Mathematics, Jan. 1987.

[46] M. T. Heath, *Scientific Computing*, 2nd ed. The McGraw-Hill Companies, Inc., Jul. 2002.

[47] R. J. Vanderbei, "Symmetric quasi-definite matrices," Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ, Technical Report SOR-91-10, 1991.

[48] H.-r. Fang and D. P. OLeary, "Modified cholesky algorithms: a catalog with new approaches," *Mathematical Programming*, vol. 115, no. 2, p. 319349, Jul. 2007. [Online]. Available: http://www.springerlink.com/content/q4m5438141j0v028/

[49] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization.* Academic Press, 1981.

[50] P. E. Gill and W. Murray, "Newton-type methods for unconstrained and linearly constrained optimization," *Mathematical Programming*, vol. 7, no. 1, p. 311350, Dec. 1974. [Online]. Available: http://www.springerlink.com/content/k041450456h75793/

[51] A. Altman and J. Gondzio, "Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 275–302, 1999, 00096. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10556789908805754

[52] Y. Nesterov, *Introductory lectures on convex optimization: a basic course.* Springer, 2004, 00908.

[53] N. I. M. Gould, D. Orban, and P. Toint, "General CUTEr documentation," CERFACS, Technical Report TR/PA/02/13, 2005, 00000.

[54] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, 2nd ed. Duxbury Press, 2002, 00000.

[55] K. Bache and M. Lichman, "UCI machine learning repository," Ph.D. dissertation, University of California, Irvine, School of Information and Computer Sciences, 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[56] I. Guyon, A. B. Hur, S. Gunn, and G. Dror, "Result analysis of the NIPS 2003 feature selection challenge," in *Advances in Neural Information Processing Systems 17.* MIT Press, 2004, p. 545552.

[57] Y. Ikebe, T. Inagaki, and S. Miyamoto, "The monotonicity theorem, cauchy's interlace theorem, and the courant- fischer theorem," *The American Mathematical Monthly*, vol. 94, no. 4, p. 352, Apr. 1987. [Online]. Available: http://www.jstor.org/discover/10.2307/2323096?uid=3739936uid=2uid=4uid=3739256sid=21103375422941

[58] C.-C. Chang and C.-J. Lin, "{LIBSVM}: introduction and benchmarks," Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, Tech. Rep., 2000.

[59] ——, "LIBSVM: a library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, p. 27:127:27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199

[60] C.-W. Hsu and C.-J. Lin, "A simple decomposition method for support vector machines," *Mach. Learn.*, vol. 46, no. 1-3, p. 291314, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1012427100071

[61] T. Joachims, "Making large-scale {svm} learning practical," in *Advances in Kernel Methods*, B. Schlkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, p. 169184. [Online]. Available: http://dl.acm.org/citation.cfm?id=299094.299104

[62] S. Wright, “Applying new optimization algorithms to model predictive control,” in *Fifth International Conference on Chemical Process Control CPC V.* CACHE Publications, 1997, p. 147155.

[63] T. Davidson, “Enriching the art of FIR filter design via convex optimization,” *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 89–101, May 2010.

[64] A. B. Gershman, N. D. Sidiropoulos, S. Shahbazpanahi, M. Bengtsson, and B. Ottersten, *Convex Optimization-based Beamforming From receive to transmit and network designs*, 2010.

Biography

Veronica J. Bloom grew up in Wisconsin. She received her Bachelor of Science degree in mathematics from Michigan Technological University in 2005, and her Master of Arts degree in mathematics from Duke University in 2007. She is currently employed as a mathematician at Northrop Grumman where she has worked since 2005.