A Solver for Non-Linear Optimization Problems with Box-Constraints

Matthias Mitterreiter

December 19, 2018

Abstract

In this work we present NOONTIME, a lightweight solver for non-linear optimization problems with box-constraints, based on Newton's method. It has minimal dependencies and is easy to use. The implementation was tested on a suitable subset of the CUTEst problem set and is compared to the open source solver library IPopt.

Contents

Chapter 1

Introduction

This work is about solving non-linear optimization problems. Mathematical optimization as a scientific field studies these problems and provides techniques to solve them. Among its numerous applications in engineering, economics and various other fields, many machine learning techniques rely heavily on optimization. For example, logistic regression requires the maximization of the likelihood function, support vector machines maximize the in-between margin of classes and the back-propagation algorithm for training deep neural nets requires the minimization of a loss function.

These functions are often non-linear or are defined over bounded subsets of $\mathbb{R}^n$. If a function of interest is twice differentiable, this allows to use a powerful technique called Newton's method to solve optimization problems of that kind.

This thesis is divided in two main parts. Chapter 2 introduces non-linear optimization problems with box-constraints and describes the theoretical background that enable the implementation of a solver for this kind of problem. Chapter 3 summarizes the results from testing NOONTIME on a subset of the CUTEst problems. For benchmarks we used the IPopt solver library and compared and evaluated the results.

Chapter 2

Theoretical Background

2.1 Definitions & Fundamentals

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. We say f is *convex* if:

$$f((1-t)x + ty) \leq (1-t)f(x) + tf(y) \quad \forall x, y \in \mathbb{R}^n,\ t \in [0,1]$$

We say f is *strictly convex* if:

$$f((1-t)x + ty) < (1-t)f(x) + tf(y) \quad \forall x, y \in \mathbb{R}^n, x \neq y,\ t \in (0,1)$$

We say f is *strongly convex* with parameter $\alpha > 0$ if:

$$f((1-t)x + ty) \leq (1-t)f(x) + tf(y) - t(1-t)\alpha\|x-y\|^2 \quad \forall x, y \in \mathbb{R}^n, t \in [0,1]$$

We say $\tilde{x} \in \mathbb{R}^n$ is a *local minimum* of f if there exists a $\delta > 0$ with:

$$f(x) \geq f(\tilde{x}) \quad \forall x \in \{x \in \mathbb{R}^n : \|x - \tilde{x}\| < \delta\}$$

and $\tilde{x} \in \mathbb{R}^n$ is a *global minimum* of f if:

$$f(x) \geq f(\tilde{x}) \quad \forall x \in \mathbb{R}^n$$

We say a symmetric matrix $A \in \mathbb{R}^{n \times n}$ is *positive definite* if

$$x^T A x > 0 \quad x \in \mathbb{R}^n \backslash \{0\}$$

and A is *positive semidefinite* if

$$x^T A x \geq 0 \quad x \in \mathbb{R}^n$$

We now present some important theorems, that we use throughout this thesis.

Theorem 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and let $\tilde{x} \in \mathbb{R}^n$ be a local minimum of f. Then $\tilde{x}$ is a global minimum of f over $\mathbb{R}^n$.*

Proof. Let $\tilde{x} \in \mathbb{R}^n$ be a local but not a global minimum of f. Therefore there exists a positive δ such that:

$$f(x) \geq f(\tilde{x}) \quad \forall x \in \mathbb{R}^n \text{ with } \|x - \tilde{x}\| < \delta$$

Now let x^* be a global minimum of f. Since f is convex, the following holds:

$$f((1-t)\tilde{x} + tx^*) \leq (1-t)f(\tilde{x}) + tf(x^*) < f(\tilde{x})$$

By choosing t such that $((1-t)\tilde{x} + tx^*) \in \{x \in \mathbb{R}^n : \|x - \tilde{x}\| < \delta\}$ we have a contradiction and therefore x^* does not exist. $\square$

Theorem 2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function. Then f is strongly convex if and only if there is a positive β such that*

$$x^T H(x) x \geq \beta x^T x \quad \forall x \in \mathbb{R}^n$$

Proof. See proof of [2, Thm. 3.2.14] □

Theorem 3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function. Then $H(x)$ is positive definite if and only if all its eigenvalues are positive.*

Proof.

⇒ Let $H(x)$ be positive definite and let λ be an eigenvalue of $H(x)$ with eigenvector $\hat{x}$.

$$H(x)\hat{x} = \lambda\hat{x} \Rightarrow \underbrace{\hat{x}^T H(x)\hat{x}}_{>0} = \lambda \underbrace{\hat{x}^T \hat{x}}_{>0} \Rightarrow \lambda > 0$$

⇐ Now let each eigenvalue $\lambda_i, i \in \{1, ..., n\}$ of $H(x)$ be positive. Since $H(x)$ is symmetric, there exits an orthogonal matrix V such that:

$$H(x) = VDV^T$$

where

$$D = diag(\lambda_1, ..., \lambda_n)$$

Now let $y \in \mathbb{R}^n$ be a non-zero vector and let $z = V^T y$. We can now write:

$$y^T H(x) y = z^T D z = \sum_{i=1}^{n} \lambda_i z_i^2$$

With y being non-zero and $z = V^T y$ being non-zero it follows that the sum above is positive and therefore $H(x)$ is positive definite.

□

Theorem 4. *Let A be a positive definite matrix. Then A is invertible.*

Proof. Let A be positive definite. By the invertible matrix theorem, we know that A is invertible iff the equation $Ax = 0$ has only the trivial solution. We write:

$$Ax = 0 \Rightarrow x^T A x = 0$$

Since A is positive definite, $x^T Ax$ is positive for a non-zero vector x. Therefore $x^T Ax = 0$ has only the trivial solution and it follows that A is invertible. □

2.2 Bounded strongly convex optimization problems

Consider the following optimization problem with twice differentiable, strongly convex objective function $f : \mathbb{R}^n \to \mathbb{R}$ and solution $\tilde{x}$:

$$\tilde{x} = \min_{x} \quad f(x) \tag{2.1a}$$

$$\text{s.t.} \quad l \leq x \leq b \tag{2.1b}$$

We call $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ the lower and upper bounds on the variable $x \in \mathbb{R}^n$, respectively. In the following we consecutively describe the building blocks of a solver for this kind of problems. Later on we will generalize it to non-convex optimization problems. We begin with the general descent method for unconstrained optimization problems [3, Ch. 9.1]:

Algorithm 1: General descent method

Input: starting point $x^{(0)} \in \mathbb{R}^n$

$k = 0$

while *not converged* **do**

  Determine descent direction $\Delta x^{(k)}$

  Line search. Chose a step size $\alpha^{(k)} > 0$

  Update. $x^{(k+1)} = x^{(k)} + \alpha^{(k)} \Delta x^{(k)}$

  k = k + 1

end

return $x^{(k)}$

2.3 Newton's method

Let f be the objective function and let $x^{(k)}$ be the current iterate with function value $f^{(k)}$, gradient $g^{(k)}$ and Hessian $H^{(k)}$. Since f is strongly convex, it follows that $H^{(k)}$ is positive definite and hence invertible by Theorem 4. We now consider the second order Taylor polynomial m_k of f in $x^{(k)}$, which we call model function:

$$m_k(x) = f^{(k)} + (g^{(k)})^T(x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T H^{(k)}(x - x^{(k)}) \tag{2.2}$$

The minimum $\tilde{x}^{(k)}$ of m_k is given by:

$$\begin{aligned} \nabla m_k(x) &= (g^{(k)}) + H^{(k)}(x - x^{(k)}) = 0 \\ \Rightarrow \tilde{x}^{(k)} &= x^{(k)} - (H^{(k)})^{-1} g^{(k)} \end{aligned}$$

from which we construct the search direction:

$$\Delta x^{(k)} = \tilde{x}^{(k)} - x^{(k)} = -(H^{(k)})^{-1} g^{(k)} \tag{2.3}$$

We call an algorithm of type (1) with descent direction (2.3) Newton's method. With $x^{(0)}$ being sufficiently close to $\tilde{x}$, the sequence of iterates converges to $\tilde{x}$ [5, Thm 3.5]. The positive definiteness property of the Hessian matrix is the key element to this method, since in this case the quadratic approximation (2.2) of f in $x^{(k)}$ is a strictly convex quadratic function with a unique solution $\tilde{x}^{(k)}$.

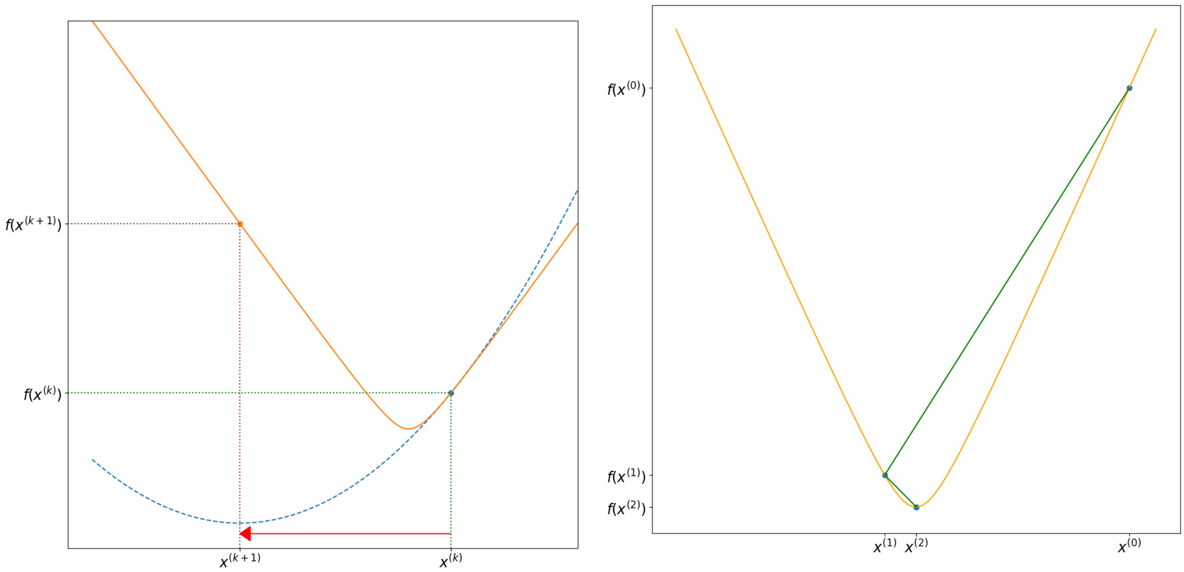
2.4 Line search

The line search is a method to choose a step length, that determines how far the algorithm moves the descent direction in any iteration.

2.4.1 Motivational example

Consider the function:

$$f(x) = (1 + x^2)^{\frac{1}{2}} \quad \text{with domain } x \in [-10, 10]$$



(a) Model function $m_k(x)$ (blue) in $x^{(k)} = 2$ and direction Δx (red).

(b) With line search, the method converges with start value $x^{(0)} = 8$ in 2 steps to the minimum in $x^{(2)} = 0$.

Figure 2.1: Divergence (left) and convergence (right) of Newton's method for the function $f(x) = (1 + x^2)^{\frac{1}{2}}$ (orange)

with:

$$f'(x) = x(1 + x^2)^{-\frac{1}{2}}$$
$$f''(x) = (1 + x^2)^{-\frac{3}{2}}$$

Since for $\beta = 1e-4$ the inequality $x^2 f''(x) \leq \beta x^2$ holds, it follows that $f''(x)$ is positive definite and therefore by Theorem 2 f is strongly convex. By using the descent direction (2.3) we get:

$$\begin{aligned}\Delta x^{(k)} &= -f''(x^{(k)})^{-1} f'(x^{(k)}) \\ &= -x^{(k)} - (x^{(k)})^3\end{aligned}$$

In the k-th step we can calculate the new iterate $x^{(k+1)}$ via:

$$x^{(k+1)} = x^{(k)} - x^{(k)} - (x^{(k)})^3 = -(x^{(k)})^3$$

From this follows:

$$f(x^{(k+1)}) = \begin{cases} f(x^{(k)}) & |x^{(k)}| = 1 \\ > f(x^{(k)}) & |x^{(k)}| > 1 \\ < f(x^{(k)}) & |x^{(k)}| < 1 \end{cases}$$

Here we can see, that Newton's method only converges for a start value with $|x^{(0)}| < 1$. In case $x^{(0)}$ is one, only the sign on the next iterate flips and in case $|x^{(0)}|$ is larger than one, our method diverges.
This is why Newton's method is called a *locally convergent* method. In Figure 2.1a it can be observed, that the model function m_k in $x^{(k)} = 2$ has its minimum in $x^{(k+1)}$, yet $f(^{(k+1)})$ is much larger than $f(x^{(k)})$.

2.4.2 Line search

In order to obtain a globally convergent method, a line search along $\Delta x^{(k)}$ can be used, such that

$$f(x^{(k)} + \alpha \Delta x^{(k)}) < f(x^{(k)}) \quad \forall x^{(k)} \in \mathbb{R}^n, \alpha > 0 \tag{2.4}$$

In order to guarantee (2.4) as well as a sufficient decrease of f, we enforce the strong Wolfe conditions [5, 3.7]:

$$f(x^{(k)} + \alpha^{(k)} \Delta x^{(k)}) \leq f(x^{(k)}) + c_1 \alpha^{(k)} (g^{(k)})^T \Delta x^{(k)} \tag{2.5}$$

$$|\nabla f(x^{(k)} + \alpha^{(k)} \Delta x^{(k)})^T \Delta x^{(k)}| \leq c_2 |(g^{(k)})^T \Delta x^{(k)}| \tag{2.6}$$

with

$$0 < c_1 < c_2 < 1$$

The full algorithm as well as implementation notes can be found in [5, Ch. 3]. By using the line search on our previous example, Newton's method converges even for start values $|x^{(0)}| > 1$ (see Figure 2.1b).

2.5 The gradient projection algorithm

Optimization problems with constraints of the form (2.1b) are called box-constraint problems. We use the gradient projection method presented in [4] to tackle these kind of problems.

For this, we define the active set A of a point x to be the set of indices, at which the components x_i of x lie on the bounds.

$$A(x) = \{i : x_i \in \{l_i, u_i\}\}$$

The set of free variables F of x is defined complementary as:

$$F(x) = \{i : l_i < x_i < u_i\}$$

The gradient projection is a two step algorithm, that generates a new descent direction $\Delta x^{(k)}$ by taking the bounds l and u on $x^{(k)}$ into account:

1. **Cauchy point computation**: Compute the Cauchy point $c^{(k)}$ (see section 2.5.1) to identify the set of active and free variables $A(c^{(k)})$ and $F(c^{(k)})$.

2. **Subspace minimization**: Solve a subspace minimization problem on the set of free variables $F(c^{(k)})$ with solution $s^{(k)}$.

Ultimately we calculate the search direction $\Delta x^{(k)} = s^{(k)} - x^{(k)}$ for the current iteration k.

2.5.1 Cauchy point computation

We start by projecting the direction of the steepest descent $-g^{(k)}$ onto the feasible region (2.1b), which can be expressed as a piecewise-linear-path:

$$x(t) = P(x^{(k)} - t g^{(k)}, l, u) \qquad t \geq 0 \tag{2.7}$$

with the projection function:

$$P(x, l, u)_i = \min\left(u_i, \max(x_i, l_i)\right) \tag{2.8}$$

The Cauchy point $c^{(k)}$ is then defined to be the minimum of the model function (2.2) along the piecewise-linear path (2.7):

$$\begin{aligned} t^* = \min_t \quad & m_k(x(t)) \\ \text{s.t.} \quad & t \geq 0 \end{aligned} \tag{2.9}$$

$$c^{(k)} = x(t^*)$$

For this purpose we calculate the set $T = \{t_i | i = 1, ..., n\}$ along the gradient direction.

$$t_i = \begin{cases} (x_i^{(k)} - u_i)/g_i^{(k)} & g_i^{(k)} < 0 \\ (x_i^{(k)} - l_i)/g_i^{(k)} & g_i^{(k)} > 0 \\ \infty & otherwise \end{cases} \tag{2.10}$$

By sorting the set T in increasing order, we obtain the ordered set $\{t^j : t^j \leq t^{j+1}, j = 1, .., n\}$. The Cauchy point $c^{(k)}$ can then be found by iteratively searching the intervals $[t^{j-1}, t^j]$ for t^*.

2.5.1.1 Interval search

In the following section we drop the outer index k, such that $g = g^{(k)}$, $H = H^{(k)}$ and define x^0 to be $x^{(k)}$. Superscripts denote the current interval.

The piecewise linear path (2.7) can now be expressed as

$$x_i(t) = \begin{cases} x_i^0 - t g_i & t \leq t_i \\ x_i^0 - t_i g_i & otherwise \end{cases}$$

Given the interval $[t^{j-1}, t^j]$ with descent direction:

$$d_i^{j-1} = \begin{cases} -g_i & t^{j-1} < t_i \\ 0 & otherwise \end{cases}$$

and breakpoints

$$\begin{aligned} x^{j-1} &= x(t^{j-1}) \\ x^j &= x(t^j) \end{aligned}$$

on line segment $[x^{j-1}, x^j]$, the model function (2.2) can be written as

$$\begin{aligned} m(x) = f &+ g^T(x^{j-1} + (t - t^{j-1})d^{j-1} - x^0) \\ &+ \frac{1}{2}(x^{j-1} + (t - t^{j-1})d^{j-1} - x^0)^T H (x^{j-1} + (t - t^{j-1})d^{j-1} - x^0) \end{aligned} \tag{2.11}$$

With $\Delta t = t - t^{j-1}$ and $z^{j-1} = x^{j-1} - x^0$, we can expand (2.11) and write it as a quadratic function in Δt:

$$\hat{m}(\Delta t) = f_{j-1} + f'_{j-1}\Delta t + \frac{1}{2} f''_{j-1} \Delta t^2 \tag{2.12}$$

where

$$\begin{aligned} f_{j-1} &= f + g^T z^{j-1} + \frac{1}{2}(z^{j-1})^T H z^{j-1} \\ f'_{j-1} &= g^T d^{j-1} + (d^{j-1})^T H z^{j-1} \\ f''_{j-1} &= (d^{j-1})^T H d^{j-1} \end{aligned}$$

which yields a minimum in $\Delta t^* = -f'_{j-1}/f''_{j-1}$. If $t^{j-1} + \Delta t^*$ lies on $[t^{j-1}, t^j)$, we found our Cauchy point c. Otherwise c lies at $x(t^{j-1})$ if $f'_{j-1} \geq 0$ and beyond or at $x(t^j)$ if $f'_{j-1} < 0$.

2.5.1.2 Updates

For exploring the next interval $[t^j, t^{j+1}]$, we set:

$$
\begin{aligned}
\Delta t^{j-1} &= t^j - t^{j-1} \\
x^j &= x^{j-1} + \Delta t^{j-1} d^{j-1} \\
z^j &= z^{j-1} + \Delta t^{j-1} d^{j-1}
\end{aligned}
$$

Since at least one variable became active it remains to update the search direction accordingly

$$
d_i^j = \begin{cases} d_i^{j-1} & i \in F(x^{j-1}) \\ 0 & i \in A(x^{j-1}) \end{cases}
$$

2.5.2 Subspace minimization

Given the Cauchy point $c^{(k)}$ in iteration k, we proceed with minimizing $m_k(\mathrm{x})$ over the set of free variables $F(c^{(k)})$. Let $Z \in \{0,1\}^{n \times |F(c^{(k)})|}$ be the matrix of unit vectors, that span the subspace of free variables at $c^{(k)}$ and let $\hat{d}$ be a vector of dimension $|F(c^{(k)})|$. Rewriting (2.2) in terms of $\hat{d}$ yields:

$$
\begin{aligned}
\hat{d}^* = \min_{\hat{d}} \quad & \hat{m}_k(\hat{d}) = \hat{d}^T \hat{r} + \frac{1}{2} \tilde{H} \hat{d} + \gamma && \text{(2.13a)} \\
\text{s.t.} \quad & l_i - c_i \le \hat{d}_i, && \text{(2.13b)} \\
& u_i - c_i \ge \hat{d}_i && \text{(2.13c)}
\end{aligned}
$$

with reduced Hessian

$$
\tilde{H} = Z^T H^{(k)} Z
$$

and gradient

$$
\hat{r} = Z^T (g^{(k)} + H^{(k)} (c^{(k)} - x^{(k)}))
$$

The solution of the subspace minimization problem $s^{(k)} \in \mathbb{R}^n$ is now feasible to compute:

$$
s_i^{(k)} = \begin{cases} c_i^{(k)} & i \notin F(c^{(k)}) \\ c_i^{(k)} + (Z\hat{d}^*)_i & i \in F(c^{(k)}) \end{cases}
$$

This leads to the search direction

$$
\Delta x^{(k)} = x^{(k)} - s^{(k)}
$$

2.6 Termination conditions

Our algorithm stops, if the infinity norm of the projected gradient becomes sufficiently small:

$$
||P(x^{(k)} - g^{(k)}, l, u) - x^{(k)}||_\infty < g_{tol} \tag{2.14}
$$

Furthermore, we stop the process if the number of iterations k reaches a limit k_{max} or if the change of the objective function over two subsequent iterations is adequately small:

$$
|f^{(k)} - f^{(k-1)}| < f_{tol} \tag{2.15}
$$

2.7 Non-linear optimization problem

So far, we have required the objective function to be strongly convex. In this case, its Hessian is always positive definite and any local minimum is a global one. Since we also want to solve non-convex problems, we now drop the strong convexity condition.

2.7.1 Non-linear optimization problem

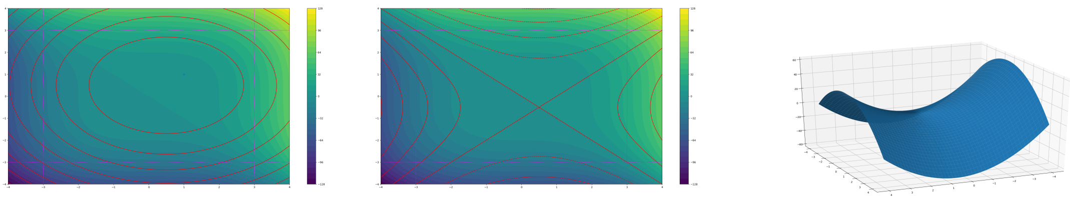
Consider the optimization problem of Section (2.2):

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & l \leq x \leq b \end{aligned} \tag{2.16}$$

We call (2.16) a non-linear optimization problem if the function $f : \mathbb{R}^n \to \mathbb{R}$ is non-linear. By necessity, we still require f to be twice differentiable.

By dropping the condition of strong convexity, we lose the guarantee that a local minimum is a global one. As a further consequence, the Hessian matrix is not necessarily positive definite. In this case, the quadratic approximation (2.2) of f in $x^{(k)}$ is not strictly convex and the descent direction (2.3) might not exist.

2.7.1.1 Example



(a) For $x = (0,0)$, $H(x)$ is positive definite and the model function (red) is strongly convex and has a unique minimum.

(b) For $x = (1,-1)$, $H(x)$ is indefinite and the model function (red) is not convex and does not have a minimum.

(c) 3D visualization of the model function for $x = (1,-1)$
.

Figure 2.2: Contour plots for problem (2.17) with bounds colored mangenta and the model function for different x colored in red.

As an example, consider the following optimization problem with non-convex objective function $f : \mathbb{R}^2 \to \mathbb{R}$:

$$\begin{aligned} \min_{x} \quad & f(x) = x_1^3 + x_2^3 \\ \text{s.t.} \quad & -3 \leq x_1 \leq 3, \\ & -3 \leq x_1 \leq 3 \end{aligned} \tag{2.17}$$

with:

$$g(x) = (3x_1^2, 3x_2^2)^T \qquad H(x) = \begin{pmatrix} 6x_1 & 0 \\ 0 & 6x_2 \end{pmatrix}$$

Depending on the value of x, the Hessian matrix $H(x)$ has different properties:

- For $x_1 = 1, x_2 = 0$, the Hessian is singular.
- For $x_1 = -1, x_2 = -1$ the Hessian is negative definite and since $\Delta x^T g > 0$, Δx is not a descent direction.

- For $x_1 = 1, x_2 = 1$ the Hessian is positive definite.
- For $x_1 = 1, x_2 = -1$ the Hessian is indefinite and since $\Delta x^T g = 0$, Δx is not a descent direction.

To overcome this obstacle, the Hessian matrix can be replaced by a positive definite one. In our case, we modify the Hessian and continue the process of minimizing the objective function with the modified Hessian.

2.7.2 Hessian Modification

The modification of the Hessian can be done in various ways [5, Ch. 3.4]. Our approach performs a spectral shift of the Hessian in case that it is not positive definite.

Let λ_{min} be the smallest eigenvalue of $H^{(k)}$ and let δ be a chosen lower bound for the eigenvalues of the modified Hessian. We can than calculate the modification parameter ω as follows:

$$\omega = \max\left(0, \delta - \lambda_{min}\right) \quad \delta \in \mathbb{R}^+$$

The modified Hessian is obtained by

$$\hat{H}^{(k)} = H^{(k)} + \omega \mathbb{I}$$

where all eigenvalues of $\hat{H}^{(k)}$ are all greater or equal to δ. It follows that $\hat{H}^{(k)}$ is positive definite.

Substituting the Hessian in the previous sections by its modification, all results generalize to non-convex functions. Furthermore, it can be shown that direction $\Delta x^{(k)}$ with $\hat{H}$ is always a descent direction.

Chapter 3

Measurements and Results

3.1 Measurements

We tested our implementation on a subset of the CUTEst problem set. For this we selected only bounded and unbounded problems with variable-size $2 \leq n \leq 5000$. In total 309 problems were used for testing. Throughout the test runs, the following solver configuration was employed

$$f_{tol} = 1e^{-8} \text{ (see (2.15))} \qquad g_{tol} = 1e^{-8} \text{ (see (2.14))}$$

We set the limit for the number of iterations at

$$k_{max} = 5000$$

and confined the CPU runtime of the solving process to 360 seconds. The variables of interest were:

- **Success**: Was the problem solved or unsolved.
- **Function value**: The objective function value of the last iterate, which is the minimum if the solver terminated successfully, or the last iterate when the solver was aborted due to the time cap, or the limit on the number of iterations.
- **Iterations**: Number of iterations performed until the solver terminated.
- **Message** The result message of the solver.

The summary of the NOONTIME test run is listed in Table 3.1.

For comparison, we used the open source solver library IPopt (version 3.12.5) [1]. IPopt as part of the COIN-OR initiative is written in C++ and primarily optimized for large-scale optimization problems. It was initially released in 2005 and since then steadily improved. It is well recognized in both academics and industry [1]. This and the fact that IPopt uses Newton's method, makes it a good competitor for NOONTIME. We ran IPopt with its default configuration on the same problem set. A summary of the Ipopt test run is listed in Table 3.2.

The results of all the test runs for both, NOONTIME and IPopt can be found in Table 4.1.

For comparing the results from both of the solvers we use the *relative solver error*. Here we introduce the *relative solver error* on the values of the objective functions, but it is similarly defined and used for the number of iterations. Given a problem from our test set with results for the objective function value from NOONTIME, f_{nt}, and from IPopt, f_{opt}. Since the absolute error of both results $|f_{nt} - f_{pt}|$ depends heavily on the problem and does not represent the goodness of the final results very well, we define the *relative solver error* for f_{nt} and f_{opt} as:

$$err_{rel}(i) = \frac{f_i - f_{min}}{|f_{min}| + 1} \qquad f_{min} = min(f_{opt}, f_{nt}) \quad i \in \{nt, opt\} \tag{3.1}$$

This allows us to classify the result. With ϵ being appropriately selected, we denote three classes as:

$$\begin{aligned} f_{opt} \ll f_{nt} \quad &\Leftrightarrow \quad f_{opt} < f_{nt} \text{ and } err_{rel}(nt) > \epsilon \\ f_{opt} \approx f_{nt} \quad &\Leftrightarrow \quad f_{nt} \leq f_{opt} \text{ and } err_{rel}(opt) < \epsilon \text{ or} \\ &\qquad\; f_{opt} \leq f_{nt} \text{ and } err_{rel}(nt) < \epsilon \\ f_{opt} \gg f_{nt} \quad &\Leftrightarrow \quad f_{opt} < f_{nt} \text{ and } err_{rel}(opt) > \epsilon \end{aligned} \tag{3.2}$$

Informally, these classes can be interpreted verbally as:

- $f_{opt} \approx f_{nt}$: IPopt and NOONTIME did equally well.
- $f_{opt} \ll f_{nt}$: IPopt did better than NOONTIME.
- $f_{opt} \gg f_{nt}$: NOONTIME did better than IPopt.

The same holds for the number of iterations of IPopt, $Iter_{opt}$, and NOONTIME, $Iter_{nt}$. For the evaluation of our results, the relative solver error was computed with the following parameters:

- Objective function value: $\epsilon = 1e^{-4}$
- Number of iterations: $\epsilon = 1$

In the full result Table 4.1, we color-encoded these classes for both the number of iterations and the objective function values. Let $f_{opt} \geq f_{nt}$. Then the cell for f_{opt} is colored *dark green*. If $f_{opt} = f_{nt}$, then also the cell for f_{nt} is colored *dark green*. Otherwise if $f_{nt} \approx f_{opt}$, then it is colored *light green* and if $f_{opt} \ll f_{nt}$, it is colored *orange*. Informally, a cell is *dark green* if it holds the best result for this specific problem, it is *light green* if the result is similarly good as the best result and it is *orange* if it is sufficiently worse than the best result. The same rules apply to the columns of *#iter*.

NOONTIME

status	**code**	**count**	**reason**
solved	0	241	Optimal Solution Found.
unsolved	1	16	Maximum number of iterations exceeded.
	2	40	Timeout after 360 seconds.
	5	9	Invalid iterate encountered ($f^{(k+1)} > f^{(k)}$).
	6	1	Overflow encountered in double scalars.
	7	1	Eigenvalues did not converge.
	8	1	Invalid search direction encountered ($g^T \Delta x > 0$).
	Σ	309	

Table 3.1: Result breakdown of running NOONTIME on the CUTEst test set.

IPopt

status	**code**	**count**	**reason**
solved	0	290	Optimal Solution Found.
unsolved	1	11	Maximum Number of Iterations exceeded.
	2	5	Timeout after 360 seconds.
	3	1	Invalid number in NLP function or derivative detected.
	4	2	Error in step computation.
	Σ	309	

Table 3.2: Result breakdown of running IPopt on the CUTEst test set.

Problems solved by NOONTIME and IPopt

Iterations	**Function value**			
	$f_{opt} \ll f_{nt}$	$f_{opt} \approx f_{nt}$	$f_{opt} \gg f_{nt}$	Σ
$Iter_{opt} \ll Iter_{nt}$	4	27	0	31
$Iter_{opt} \approx Iter_{nt}$	6	160	3	169
$Iter_{opt} \gg Iter_{nt}$	7	24	4	35
Σ	17	211	7	235

Table 3.3: Comparing the results from all problems, that both, IPopt and NOONTIME solved.

3.2 Evaluation

On the given 309 problems, NOONTIME solved 241, which amounts a success rate of 77.99% and IPopt in comparison solved 290 problems, equivalent to 93.94%. The relationships of results regarding the success variable are listed in Table 3.4.

		NOONTIME		
		solved	**unsolved**	Σ
IPopt	**solved**	235	55	290
	unsolved	6	13	19
	Σ	241	68	309

Table 3.4: Results of testing NOONTIME and IPopt on the problem set.

3.2.1 Unsolved problems

For NOONTIME, the largest group of unsolved problems is the group with code number 2 where the solver process was aborted due to the cut-off time. It counts 40 problems in total. By comparison, for IPopt the same group contains only 5 members. This gap in performance can be explained easily: [1]

1. **Sparsity**: In contrast to IPopt, NOONTIME does not exploit sparsity structures in the Hessian matrix. Especially for problems with many variables, this slows down the solving process with expensive operations like eigenvalue computation or the solving of linear systems.

2. **Native Python**: The Cauchy point computation and the subspace minimization consist of many loops that are implemented in native Python which runs slower in comparison to a compiled version of the same.

The second largest group (code number 1) of unsolved problems with 16 members are the problems where the maximum number of iterations was exceeded. Here we are not much worse off than IPopt with 11 problems in this category. Exploratory test runs with modified solver configuration settings suggests, that the size of this group can be reduced by adjusting the solver configurations to the specific problems.

The three problems with exit code 6,7 and 8 in Table 3.1 could not be solved because of numerical issues in our implementation. The problems with exit code 5 in Table 3.1 could not be solved due to numerical issues in the *scipy* line search that we rely on.

3.2.2 Solved problems

As shown in Table 3.4 there are 235 problems that both IPopt and NOONTIME solved. We classified the results of these problems according to the criterion (3.2) which is summarized in Table 3.3.
From the 235 there are 191 problems or $81,3\%$ where NOONTIME was at least as good as IPopt, both in terms of the goodness of the minimum and the number of iterations. In contrast, for IPopt there are 197 problems or $83,9\%$, where it was at least as good as NOONTIME.

Now considering the 211 problems where $f_{opt} \approx f_{nt}$, we were interested in how fast (in terms of iterations) the two implementations converged. By correlating the relative number of problems solved with the relative overhead in terms of iterations as shown in in Figure 3.1, we conclude, that both solvers behave similarly regarding convergence speed.

[1] We note here, that we did not optimize for speed in terms of CPU time, which is also the reason why we did not measure and compare the runtimes in the results.

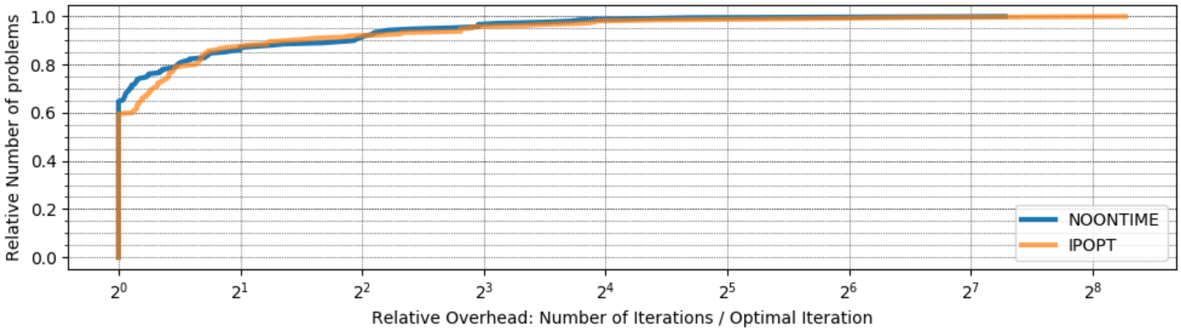


Figure 3.1: Convergence speed for the intersection of problems that were solved by IPopt and NOONTIME. The blue curve represents NOONTIME and the orange one IPopt.

Chapter 4

Appendix

name	*bounds*	*n*	*success* noontime	*success* ipopt	*f* noontime	*f* ipopt	*#iter* noontime	*#iter* ipopt	*code* noontime	*code* ipopt
3PK	True	30	True	True	1.720119E+00	1.720119E+00	1	11	0	0
AIRCRFTB	True	8	True	True	1.890693E-08	4.788247E-25	58	15	0	0
AKIVA	False	2	True	True	6.166042E+00	6.166042E+00	6	6	0	0
ALLINIT	True	4	True	True	1.670597E+01	1.670597E+01	7	11	0	0
ALLINITU	False	4	True	True	5.744385E+00	5.744385E+00	7	14	0	0
ARGLINA	False	200	True	True	2.000000E+02	2.000000E+02	1	1	0	0
ARGLINB	False	200	True	True	9.962547E+01	9.962547E+01	2	2	0	0
ARGLINC	False	200	True	True	1.011255E+02	1.011255E+02	2	2	0	0
ARGTRIGLS	False	10	True	True	3.762173E-19	7.054688E-25	5	8	0	0
ARWHEAD	False	5000	True	True	0.000000E+00	0.000000E+00	6	6	0	0
BA-L1LS	False	57	True	True	1.256169E-24	7.648105E-21	138	10	0	0
BA-L1SPLS	False	57	True	True	2.105604E-23	6.476196E-17	23	9	0	0
BARD	False	3	True	True	8.214877E-03	8.214877E-03	8	8	0	0
BDEXP	True	5000	False	True	7.652549E-04	1.612155E-06	39	18	2	0
BDQRTIC	False	5000	True	True	2.000626E+04	2.000626E+04	10	9	0	0
BEALE	False	2	True	True	2.895403E-16	4.342571E-18	6	8	0	0
BENNETT5LS	False	3	True	True	5.389879E-04	5.563289E-04	90	21	0	0
BIGGS3	True	6	True	True	6.490822E-09	4.080557E-17	118	9	0	0
BIGGS5	True	6	True	True	1.724764E-07	4.016507E-17	64	20	0	0
BIGGS6	False	6	True	True	1.579136E-09	3.748460E-17	320	79	0	0
BIGGSB1	True	5000	False	True	1.558711E-01	1.500460E-02	16	17	2	0
BLEACHNG	True	17	False	True	9.176758E+03	9.176759E+03	4	16	5	0
BOX2	True	3	True	True	5.403778E-19	5.403778E-19	7	8	0	0
BOX3	False	3	True	True	6.871809E-19	5.382223E-19	8	9	0	0
BOXBODLS	False	2	True	True	9.771500E+03	9.771500E+03	7	13	0	0
BQP1VAR	True	1	True	True	0.000000E+00	-7.443447E-09	1	5	0	0
BQPGABIM	True	50	True	True	-3.790343E-05	-3.789187E-05	2	15	0	0
BQPGASIM	True	50	True	True	-5.519814E-05	-5.516937E-05	2	15	0	0
BQPGAUSS	True	2003	False	True	-1.157101E-02	-3.625779E-01	20	23	2	0
BRKMCC	False	2	True	True	1.690427E-01	1.690427E-01	3	3	0	0
BROWNAL	False	200	True	True	5.586575E-25	1.091606E-21	5	5	0	0
BROWNBS	False	2	True	True	0.000000E+00	0.000000E+00	10	7	0	0
BROWNDEN	False	4	True	True	8.582220E+04	8.582220E+04	8	8	0	0
BROWNDENE	False	4	True	True	9.032343E+02	9.032343E+02	1	1	0	0
BROYDN3DLS	False	10	True	True	1.232595E-30	2.366583E-30	6	6	0	0
BROYDN7D	False	5000	True	True	1.775979E+03	1.515038E+03	54	121	0	0
BROYDNBDLS	False	10	True	True	1.695783E-17	7.957487E-18	11	11	0	0
BRYBND	False	5000	True	True	2.027851E-20	4.279822E-21	13	11	0	0
CAMEL6	True	2	True	True	-1.031628E+00	-1.031628E+00	6	10	0	0
CHAINWOO	False	4000	False	True	1.574620E+04	7.933124E+01	12	187	2	0

name	*bounds*	*n*	*success* noontime	*success* ipopt	*f* noontime	*f* ipopt	*#iter* noontime	*#iter* ipopt	*code* noontime	*code* ipopt
CHEBYQAD	True	100	True	True	1.196433E-02	4.877696E-03	47	273	0	0
CHENHARK	True	5000	False	True	9.995000E+02	-2.000002E+00	0	19	6	0
CHNROSNB	False	50	True	True	9.830587E-20	1.539369E-22	50	42	0	0
CHNRSNBM	False	50	True	True	3.598979E-19	8.488299E-16	56	52	0	0
CHWIRUT1LS	False	3	False	True	3.710101E+05	2.384477E+03	1	6	5	0
CHWIRUT2LS	False	3	False	True	1.326249E+07	5.130480E+02	1	6	5	0
CLIFF	False	2	True	True	1.997866E-01	2.072380E-01	27	23	0	0
CRAGGLVY	False	5000	True	True	1.688215E+03	1.688215E+03	14	14	0	0
CUBE	False	2	True	True	2.563981E-18	1.753568E-24	28	27	0	0
DANWOODLS	False	2	True	True	4.317308E-03	4.317308E-03	13	11	0	0
DECONVB	True	63	True	False	4.032415E-07	6.952463E-10	64	3000	0	1
DECONVU	True	63	True	True	1.038219E-06	4.362019E-11	88	182	0	0
DENSCHNA	False	2	True	True	1.102837E-23	1.102837E-23	6	6	0	0
DENSCHNB	False	2	True	True	8.366166E-27	9.860761E-32	7	7	0	0
DENSCHNC	False	2	True	True	8.018738E-23	2.177679E-20	11	10	0	0
DENSCHND	False	3	True	True	5.448220E-09	2.221899E-04	35	26	0	0
DENSCHNE	False	3	True	True	2.363081E-17	1.860553E-17	258	14	0	0
DENSCHNF	False	2	True	True	6.513246E-22	6.513246E-22	6	6	0	0
DIXMAANA	False	3000	True	True	1.000000E+00	1.000000E+00	6	7	0	0
DIXMAANB	False	3000	True	True	1.000000E+00	1.000000E+00	7	11	0	0
DIXMAANC	False	3000	True	True	1.000000E+00	1.000000E+00	8	9	0	0
DIXMAAND	False	3000	True	True	1.000000E+00	1.000000E+00	9	9	0	0
DIXMAANE	False	3000	True	True	1.000000E+00	1.000000E+00	9	10	0	0
DIXMAANF	False	3000	True	True	1.000000E+00	1.000000E+00	31	19	0	0
DIXMAANG	False	3000	True	True	1.000000E+00	1.000000E+00	32	16	0	0
DIXMAANH	False	3000	True	True	1.000000E+00	1.000000E+00	24	19	0	0
DIXMAANI	False	3000	False	True	1.166435E+00	1.000000E+00	100	18	2	0
DIXMAANJ	False	3000	False	True	1.000613E+00	1.000000E+00	83	20	2	0
DIXMAANK	False	3000	False	True	1.001476E+00	1.000000E+00	83	24	2	0
DIXMAANL	False	3000	False	True	1.013376E+00	1.000000E+00	83	27	2	0
DIXMAANM	False	3000	False	True	2.147182E+00	1.000000E+00	100	11	2	0
DIXMAANN	False	3000	False	True	1.007268E+00	1.000000E+00	81	25	2	0
DIXMAANO	False	3000	False	True	1.003826E+00	1.000000E+00	82	25	2	0
DIXMAANP	False	3000	False	True	1.002339E+00	1.000000E+00	81	28	2	0
DJTL	False	2	True	True	-8.951545E+03	-8.951545E+03	2541	1527	0	0
DQDRTIC	False	5000	True	True	0.000000E+00	5.916457E-29	1	1	0	0
DQRTIC	False	5000	False	True	6.240630E+17	3.935732E+01	0	23	2	0
DRCAV1LQ	True	4489	False	True	1.296927E-04	5.852343E-15	22	94	2	0
DRCAV2LQ	True	4489	False	True	2.002252E-04	4.727775E-08	22	169	2	0
DRCAV3LQ	True	4489	False	True	2.181732E-03	1.371340E-06	22	490	2	0
ECKERLE4LS	False	3	True	True	6.996961E-01	1.463589E-03	37	36	0	0
EDENSCH	False	2000	True	True	1.200328E+04	1.200328E+04	11	12	0	0
EG1	True	3	True	True	-1.132801E+00	-1.429307E+00	12	7	0	0
EG2	False	1000	True	True	-9.989474E+02	-9.989474E+02	3	4	0	0
EIGENALS	False	2550	True	False	3.038702E-12	None	136	None	0	2
EIGENBLS	False	2550	False	False	4.958431E-02	None	135	None	2	2
EIGENCLS	False	2652	False	False	1.502568E+03	None	126	None	2	2
ENGVAL1	False	5000	True	True	5.548668E+03	5.548668E+03	8	8	0	0
ENGVAL2	False	3	True	True	2.783412E-22	1.700242E-20	13	21	0	0
ENSOLS	False	9	True	True	7.885398E+02	7.885398E+02	9	7	0	0
ERRINROS	False	50	True	True	4.040449E+01	4.040449E+01	39	28	0	0
ERRINRSM	False	50	True	True	3.851945E+01	3.851945E+01	54	40	0	0
EXPFIT	False	2	True	True	2.405106E-01	2.405106E-01	6	8	0	0
EXPLIN	True	1200	True	True	-7.192548E+07	-7.192548E+07	72	57	0	0
EXPLIN2	True	1200	True	True	-7.199883E+07	-7.199883E+07	23	25	0	0
EXPQUAD	True	1200	False	True	-3.684941E+09	-3.684941E+09	138	32	5	0
EXTROSNB	False	1000	True	True	1.986815E-06	1.413033E-09	273	2718	0	0

name	*bounds*	*n*	*success* noontime	*success* ipopt	*f* noontime	*f* ipopt	*#iter* noontime	*#iter* ipopt	*code* noontime	*code* ipopt
FBRAIN2LS	True	4	True	True	3.683882E-01	3.683882E-01	11	15	0	0
FBRAIN3LS	False	6	True	False	2.497709E-01	2.419554E-01	2004	3000	0	1
FBRAINLS	True	2	True	True	4.166029E-01	4.166029E-01	7	8	0	0
FLETBV3M	False	5000	False	True	-2.327752E+05	-2.249399E+05	26	235	2	0
FLETCBV2	False	5000	True	True	-5.002682E-01	-5.002863E-01	1	1	0	0
FLETCBV3	False	5000	False	False	-7.310101E+09	-8.243740E+06	26	3000	2	1
FLETCHBV	False	5000	False	False	-2.293644E+16	-8.272254E+14	26	3000	2	1
FLETCHCR	False	1000	True	True	1.326011E-16	5.294468E-20	1568	1473	0	0
FREUROTH	False	5000	True	True	6.081592E+05	6.081592E+05	12	8	0	0
GAUSS1LS	False	8	True	True	1.315822E+03	1.315822E+03	5	5	0	0
GAUSS2LS	False	8	True	True	1.247528E+03	1.247528E+03	5	5	0	0
GAUSS3LS	False	8	True	True	1.244485E+03	1.244485E+03	8	11	0	0
GAUSSIAN	False	3	True	True	1.127933E-08	1.127933E-08	2	2	0	0
GBRAINLS	False	2	True	True	2.851586E+01	2.851586E+01	6	6	0	0
GENHUMPS	False	5000	False	False	7.571708E+07	8.074689E+07	23	3000	2	1
GENROSE	False	500	True	True	1.000000E+00	1.000000E+00	344	382	0	0
GENROSEB	True	500	True	True	1.593945E+03	1.593945E+03	129	15	0	0
GROWTHLS	False	3	True	True	1.004041E+00	1.004041E+00	75	71	0	0
GULF	False	3	True	True	3.645471E-19	5.933775E-22	38	27	0	0
HADAMALS	True	400	True	True	1.963648E+02	1.914699E+02	187	162	0	0
HAHN1LS	False	7	False	True	4.727615E+07	3.338424E+01	1	78	5	0
HAIRY	False	2	True	True	2.000000E+01	2.000000E+01	33	52	0	0
HARKERP2	True	1000	True	True	-5.000000E-01	-4.708660E-01	8	23	0	0
HART6	True	6	True	True	-3.322887E+00	-3.322887E+00	5	8	0	0
HATFLDA	True	4	True	True	4.991961E-22	7.237037E-16	22	10	0	0
HATFLDB	True	4	True	True	5.572809E-03	5.572812E-03	19	10	0	0
HATFLDC	True	25	True	True	3.418825E-27	2.921509E-17	5	5	0	0
HATFLDD	False	3	True	True	6.615134E-08	6.615114E-08	19	21	0	0
HATFLDE	False	3	True	True	5.120000E-07	5.120377E-07	18	20	0	0
HATFLDFL	False	3	True	True	6.626283E-05	6.016514E-05	4	1233	0	0
HEART6LS	False	6	True	True	7.175581E-18	9.127834E-23	373	878	0	0
HEART8LS	False	8	True	True	7.013926E-19	6.309963E-29	475	106	0	0
HELIX	False	3	True	True	7.912620E-17	6.057699E-25	14	13	0	0
HIELOW	False	3	True	True	-4.789078E+06	8.741654E+02	2	8	0	0
HILBERTA	False	2	True	True	2.958228E-31	1.314768E-31	1	1	0	0
HILBERTB	False	10	True	True	2.370815E-29	7.067769E-30	1	1	0	0
HIMMELBB	False	2	True	True	7.334173E-17	1.401396E-17	5	18	0	0
HIMMELBF	False	4	False	True	3.185790E+02	3.185717E+02	5001	75	1	0
HIMMELBG	False	2	True	True	8.900056E-27	3.633000E-22	5	6	0	0
HIMMELBH	False	2	True	True	-1.000000E+00	-1.000000E+00	6	4	0	0
HIMMELP1	True	2	True	True	-2.389742E+01	-6.205394E+01	56	11	0	0
HOLMES	True	180	True	True	1.248150E+03	1.248150E+03	20	12	0	0
HS1	True	2	True	True	1.965084E-23	5.894626E-16	26	25	0	0
HS110	True	10	True	True	-4.577848E+01	-4.577848E+01	6	6	0	0
HS2	True	2	True	True	4.941229E+00	4.941229E+00	8	11	0	0
HS25	True	3	True	True	3.283500E+01	1.034604E-15	1	36	0	0
HS3	True	2	True	True	2.174944E-08	-7.494096E-09	31	4	0	0
HS38	True	4	True	True	1.533400E-18	2.761247E-19	41	40	0	0
HS3MOD	True	2	True	True	4.271062E-10	-7.494096E-09	12	5	0	0
HS4	True	2	True	True	2.666667E+00	2.666667E+00	1	5	0	0
HS45	True	5	True	True	1.000000E+00	1.000000E+00	3	7	0	0
HS5	True	2	True	True	-1.913223E+00	-1.913223E+00	5	8	0	0
HUMPS	False	2	True	True	2.098890E-14	1.879074E-23	87	323	0	0
HYDC20LS	False	99	False	True	7.556457E-03	7.695631E-02	5001	775	1	0
INDEF	False	5000	False	True	-5.243019E+09	-2.777566E+20	22	125	2	0
INTEQNELS	False	12	True	True	3.994096E-22	3.994096E-22	3	3	0	0
JENSMP	False	2	True	True	1.243622E+02	1.243622E+02	10	9	0	0

name	*bounds*	*n*	*success* noontime	*success* ipopt	*f* noontime	*f* ipopt	*#iter* noontime	*#iter* ipopt	*code* noontime	*code* ipopt
JIMACK	False	3549	False	True	8.841238E-01	8.667933E-01	47	18	2	0
KIRBY2LS	False	5	False	True	1.869869E+06	3.905074E+00	1	11	5	0
KOEBHELB	True	3	True	True	7.751635E+01	7.751635E+01	77	345	0	0
KOWOSB	False	4	True	True	3.078009E-04	3.078009E-04	62	8	0	0
LANCZOS1LS	False	6	True	True	1.637700E-05	4.285594E-17	660	169	0	0
LANCZOS2LS	False	6	True	True	1.669310E-05	2.229943E-11	669	102	0	0
LANCZOS3LS	False	6	True	True	1.633548E-05	1.611719E-08	679	159	0	0
LIARWHD	False	5000	True	True	6.380775E-22	6.380775E-22	12	12	0	0
LINVERSE	True	1999	True	True	6.820000E+02	6.810000E+02	128	745	0	0
LOGHAIRY	False	2	True	True	6.102268E+00	1.823216E-01	565	2245	0	0
LOGROS	True	2	True	True	0.000000E+00	0.000000E+00	53	65	0	0
LSC1LS	False	3	True	True	7.711852E+00	7.711852E+00	67	16	0	0
LSC2LS	False	3	False	True	1.403637E+01	1.333415E+01	5001	44	1	0
LUKSAN11LS	False	100	True	True	8.530675E-18	1.949090E-27	344	333	0	0
LUKSAN12LS	False	98	True	True	4.228836E+03	4.292197E+03	33	25	0	0
LUKSAN13LS	False	98	True	True	2.518886E+04	2.518886E+04	19	19	0	0
LUKSAN14LS	False	98	True	True	1.239235E+02	1.239235E+02	11	11	0	0
LUKSAN15LS	False	100	True	True	4.168876E+02	3.569697E+00	8	9	0	0
LUKSAN16LS	False	100	True	True	3.569697E+00	3.569697E+00	7	6	0	0
LUKSAN17LS	False	100	True	True	4.931613E-01	4.931613E-01	27	16	0	0
LUKSAN21LS	False	100	True	True	3.729845E-20	2.610386E-17	289	11	0	0
LUKSAN22LS	False	100	True	True	8.689405E+02	8.689405E+02	22	16	0	0
MANCINO	False	100	True	True	1.239764E-21	1.433993E-21	15	18	0	0
MARATOSB	False	2	True	True	-1.000000E+00	-1.000000E+00	734	670	0	0
MAXLIKA	True	8	True	True	1.136307E+03	1.136307E+03	315	28	0	0
MCCORMCK	True	5000	True	True	-4.566581E+03	-4.566581E+03	6	7	0	0
MDHOLE	True	2	True	True	2.889774E-17	-2.261664E-09	38	43	0	0
MEXHAT	False	2	True	True	-4.001000E-02	-4.001000E-02	28	26	0	0
MEYER3	False	3	False	True	1.761481E+09	8.794586E+01	1	195	5	0
MGH09LS	False	4	False	True	3.514219E-03	3.075056E-04	5001	71	1	0
MGH10LS	False	3	False	True	1.417863E+09	8.794586E+01	5001	1724	1	0
MGH17LS	False	5	True	True	1.022432E+00	5.464895E-05	11	324	0	0
MINSURF	True	64	True	True	1.000000E+00	1.000000E+00	31	4	0	0
MISRA1ALS	False	2	True	True	1.245514E-01	1.245514E-01	37	40	0	0
MISRA1BLS	False	2	True	True	7.546468E-02	7.546468E-02	28	34	0	0
MISRA1CLS	False	2	True	True	4.096684E-02	4.096684E-02	18	14	0	0
MISRA1DLS	False	2	True	True	5.641930E-02	5.641930E-02	24	30	0	0
MOREBV	False	5000	True	True	7.704646E-09	5.831055E-15	4	1	0	0
MSQRTALS	False	1024	True	True	4.968121E-18	4.223720E-16	84	24	0	0
MSQRTBLS	False	1024	True	True	1.909138E-18	1.365103E-21	44	24	0	0
NCB20B	False	5000	True	True	7.351301E+03	7.351301E+03	9	16	0	0
NELSONLS	False	3	True	True	3.797683E+00	3.797683E+00	68	68	0	0
NONCVXU2	False	5000	False	False	1.317778E+08	None	18	None	2	2
NONCVXUN	False	5000	False	True	1.561126E+08	1.159976E+04	19	2490	2	0
NONDIA	False	5000	False	True	4.897136E-06	5.226135E-13	21	7	2	0
NONDQUAR	False	5000	False	True	4.775167E-03	2.069477E-10	18	19	2	0
NONMSQRT	False	4900	False	False	1.267592E+03	None	24	None	2	2
NONSCOMP	True	5000	True	True	2.198811E-12	1.233203E-05	12	21	0	0
OSBORNEA	False	5	True	True	4.902382E-02	5.464895E-05	126	64	0	0
OSBORNEB	False	11	True	True	4.013774E-02	4.013774E-02	31	19	0	0
OSCIPATH	False	10	True	False	9.999667E-01	9.994943E-01	2	3000	0	1
OSLBQP	True	8	True	True	6.250000E+00	6.250000E+00	1	14	0	0
PALMER1	True	4	True	True	1.175460E+04	1.175460E+04	8	392	0	0
PALMER1A	True	6	True	True	8.988306E-02	8.988306E-02	35	45	0	0
PALMER1B	True	4	True	True	3.447349E+00	3.447349E+00	15	20	0	0
PALMER1C	False	8	True	True	9.760505E-02	9.760505E-02	2	1	0	0
PALMER1D	False	7	True	True	6.526740E-01	6.526740E-01	1	1	0	0

name	*bounds*	*n*	*success* noontime	*success* ipopt	*f* noontime	*f* ipopt	*#iter* noontime	*#iter* ipopt	*code* noontime	*code* ipopt
PALMER1E	True	8	True	True	2.447114E+00	8.353481E-04	227	43	0	0
PALMER2	True	4	True	True	3.651098E+03	3.651098E+03	10	902	0	0
PALMER2A	True	6	True	True	1.710972E-02	1.710972E-02	76	87	0	0
PALMER2B	True	4	True	True	6.232669E-01	6.232669E-01	11	18	0	0
PALMER2C	False	8	True	True	1.436889E-02	1.436889E-02	2	1	0	0
PALMER2E	True	8	True	True	2.065035E-04	2.065044E-04	65	17	0	0
PALMER3	True	4	True	True	2.265958E+03	2.265958E+03	11	167	0	0
PALMER3A	True	6	True	True	2.043143E-02	2.043143E-02	67	80	0	0
PALMER3B	True	4	True	True	4.227647E+00	4.227647E+00	11	14	0	0
PALMER3C	False	8	True	True	1.953764E-02	1.953764E-02	1	1	0	0
PALMER3E	True	8	False	True	3.248139E-02	5.074119E-05	5001	29	1	0
PALMER4	True	4	True	True	2.285383E+03	2.285383E+03	15	329	0	0
PALMER4A	True	6	True	True	4.060614E-02	4.060614E-02	43	56	0	0
PALMER4B	True	4	True	True	6.835139E+00	6.835139E+00	12	15	0	0
PALMER4C	False	8	True	True	5.031069E-02	5.031069E-02	1	1	0	0
PALMER4E	True	8	True	True	3.708669E-01	1.480035E-04	292	30	0	0
PALMER5A	True	8	False	False	1.248709E-01	3.884813E-02	5001	3000	1	1
PALMER5B	True	9	False	True	7.685521E-02	9.752421E-03	5001	81	1	0
PALMER5C	False	6	True	True	2.128087E+00	2.128087E+00	1	1	0	0
PALMER5D	False	4	True	True	8.733939E+01	8.733939E+01	1	1	0	0
PALMER5E	True	8	True	False	4.524551E-02	2.088286E-02	139	3000	0	1
PALMER6A	True	6	True	True	5.594885E-02	5.594885E-02	99	123	0	0
PALMER6C	False	8	False	True	8.478351E-02	1.638744E-02	5001	1	1	0
PALMER6E	True	8	False	True	3.037913E-02	2.239541E-04	5001	30	1	0
PALMER7A	True	6	False	False	1.062846E+01	1.033491E+01	5001	3000	1	1
PALMER7C	False	8	False	True	2.443819E+00	6.019872E-01	5001	1	1	0
PALMER7E	True	8	False	False	7.851533E+00	6.574223E+00	5001	3000	1	1
PALMER8A	True	6	True	True	7.400970E-02	7.400970E-02	34	45	0	0
PALMER8C	False	8	False	True	4.319838E-01	1.597678E-01	5001	1	1	0
PALMER8E	True	8	True	True	6.339306E-03	6.339306E-03	3582	23	0	0
PARKCH	False	15	True	True	-8.974697E+06	1.623743E+03	5	17	0	0
PENALTY1	False	1000	False	True	1.114448E+17	6.439498E+00	0	23	2	0
PENALTY2	False	200	True	True	4.711628E+13	4.711628E+13	10	10	0	0
PENTDI	True	5000	True	True	-7.500000E-01	-7.500176E-01	1	15	0	0
PFIT1LS	True	3	True	True	1.244752E-07	9.577731E-16	212	307	0	0
PFIT2LS	True	3	True	True	8.160119E-09	1.623231E-16	64	102	0	0
PFIT3LS	True	3	True	True	9.000000E-09	2.799655E-15	146	141	0	0
PFIT4LS	True	3	True	True	4.613960E-09	4.404745E-16	245	235	0	0
POWELLBC	True	1000	False	False	3.104193E+05	0.000000E+00	1293	None	2	3
POWELLBSLS	False	2	True	True	6.558819E-07	2.562307E-26	30	91	0	0
POWELLSG	False	5000	True	True	4.204860E-08	8.332977E-09	19	19	0	0
PROBPENL	True	500	True	True	3.991981E-07	3.981010E-07	3	5	0	0
PSPDOC	True	4	True	True	2.414214E+00	2.414214E+00	3	7	0	0
QR3DLS	True	610	False	True	7.554681E-02	5.515677E-16	269	203	2	0
QRTQUAD	True	5000	False	True	-4.036489E+10	-2.648567E+11	13	376	2	0
QUARTC	False	5000	False	True	6.240630E+17	3.935732E+01	0	23	2	0
QUDLIN	True	5000	True	True	-1.250000E+09	-1.250000E+09	4	28	0	0
RAT42LS	False	3	False	True	1.991585E+04	8.056523E+00	0	28	7	0
RAT43LS	False	4	True	True	1.076462E+06	8.786405E+03	101	34	0	0
RAYBENDL	True	2050	False	False	9.786530E+01	-2.872387E+09	152	36	2	4
RAYBENDS	True	2050	False	False	9.789312E+01	-2.913347E+09	122	23	2	4
ROSENBR	False	2	True	True	2.124604E-18	3.743976E-21	21	21	0	0
ROSZMAN1LS	False	4	False	True	1.537246E-01	4.948485E-04	5001	28	1	0
S308	False	2	True	True	7.731991E-01	7.731991E-01	9	9	0	0
S368	True	8	True	True	-7.500000E-01	-7.500000E-01	51	11	0	0
SANTALS	True	21	True	True	1.224358E-05	1.224358E-05	48	33	0	0
SBRYBND	False	5000	True	True	8.311581E-27	1.173534E-20	11	13	0	0

name	*bounds*	*n*	*success* noontime	*success* ipopt	*f* noontime	*f* ipopt	*#iter* noontime	*#iter* ipopt	*code* noontime	*code* ipopt
SCHMVETT	False	5000	True	True	-1.499400E+04	-1.499400E+04	3	3	0	0
SCOSINE	False	5000	False	True	4.360693E+03	-4.999000E+03	1	129	8	0
SENSORS	False	100	True	True	-1.919531E+03	-1.987875E+03	24	36	0	0
SIM2BQP	True	2	True	True	0.000000E+00	-7.471066E-09	1	7	0	0
SIMBQP	True	2	True	True	0.000000E+00	-7.421861E-09	1	7	0	0
SINEALI	True	1000	True	True	-9.989947E+04	-9.990096E+04	9	26	0	0
SINEVAL	False	2	True	True	1.450661E-21	5.787363E-43	42	42	0	0
SINQUAD	False	5000	False	True	-6.757014E+06	-6.757014E+06	10	34	5	0
SISSER	False	2	True	True	2.105255E-09	6.331104E-13	13	18	0	0
SNAIL	False	2	True	True	2.431911E-29	1.472743E-28	67	63	0	0
SPARSINE	False	5000	False	True	2.147088E+06	1.295293E-07	19	15	2	0
SPECAN	True	9	True	True	1.645655E-13	2.306852E-13	9	10	0	0
SPMSRTLS	False	4999	False	True	3.192705E-08	1.855581E-15	14	22	2	0
SROSENBR	False	5000	True	True	2.681252E-23	3.301788E-22	9	8	0	0
SSBRYBND	False	5000	True	True	6.519279E-26	7.710062E-13	10	26	0	0
SSCOSINE	False	5000	False	True	4.306404E+03	-4.999000E+03	3	71	5	0
SSI	False	3	True	False	1.652794E-05	1.385069E-09	142	3000	0	1
STRATEC	False	10	True	True	-2.726163E+07	2.212262E+03	2	24	0	0
TESTQUAD	False	5000	True	True	0.000000E+00	0.000000E+00	1	1	0	0
THURBERLS	False	7	True	True	5.642708E+03	5.642708E+03	21	19	0	0
TOINTGOR	False	50	True	True	1.373905E+03	1.373905E+03	7	7	0	0
TOINTGSS	False	5000	True	True	1.000000E+01	1.000000E+01	1	1	0	0
TOINTPSP	False	50	True	True	2.255604E+02	2.255604E+02	13	20	0	0
TOINTQOR	False	50	True	True	1.175472E+03	1.175472E+03	1	1	0	0
TQUARTIC	False	5000	True	True	5.458963E-23	1.804881E-22	1	1	0	0
TRIDIA	False	5000	True	True	6.491900E-25	6.345166E-25	1	1	0	0
VARDIM	False	200	False	True	3.256542E+16	1.743420E-06	0	27	2	0
VAREIGVL	False	50	True	True	4.234969E-10	1.155963E-19	11	13	0	0
VESUVIALS	False	8	False	True	1.978977E+03	9.914100E+02	5001	48	1	0
VESUVIOLS	False	8	True	True	9.914100E+02	9.914100E+02	8	10	0	0
VESUVIOULS	False	8	True	True	4.771138E-01	4.771138E-01	8	8	0	0
VIBRBEAM	False	8	True	True	1.010408E+01	3.322376E-01	19	58	0	0
WALL10	True	1461	False	True	1.533520E+01	-4.559538E+05	528	32	2	0
WATSON	False	12	True	True	8.658558E-06	2.131709E-09	110	21	0	0
WEEDS	True	3	True	True	2.587277E+00	2.587277E+00	19	25	0	0
WOODS	False	4000	True	True	1.233828E-27	4.837167E-24	42	40	0	0
YATP1LS	False	2600	True	True	7.353642E-10	7.795463E-20	28	20	0	0
YATP2LS	False	2600	True	True	7.471263E-10	2.731200E-28	19	31	0	0
YFIT	True	3	True	True	6.669755E-13	6.717568E-13	36	49	0	0
YFITU	False	3	True	True	6.669755E-13	6.669727E-13	36	35	0	0
ZANGWIL2	False	2	True	True	-1.820000E+01	-1.820000E+01	1	1	0	0

Table 4.1: Results of testing Noontime and Ipopt on the the listed problems from the Cutest problem set. *name* refers to the name of the problem, *bounds* is *True* if the problem has bounds on the variables, otherwise its *False*, n refers to the number of variables of the respective problem, *success* is *True* if the problem was solved, otherwise its *False*. f denotes the objective function value of the found minimum or the objective function value of the last iterate, when the minimization process was aborted. *#iter* gives the number of iterations until the process terminated and *code* refers to the code of the result message that we encoded in Table 3.1 and in Table 3.2

Bibliography

[1] Ipopt, interior point optimizer. https://projects.coin-or.org/Ipopt.

[2] ALT, W. *Nichtlineare Optimierung: Eine Einführung in Theorie, Verfahren und Anwendungen.* vieweg studium; Aufbaukurs Mathematik. Vieweg+Teubner Verlag, 2011.

[3] BOYD, S., AND VANDENBERGHE, L. *Convex Optimization.* Cambridge University Press, New York, NY, USA, 2004.

[4] BYRD, R. H., LU, P., NOCEDAL, J., AND ZHU, C. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput. 16*, 5 (Sept. 1995), 1190–1208.

[5] NOCEDAL, J., AND WRIGHT, S. J. *Numerical Optimization*, second ed. Springer, New York, NY, USA, 2006.