Analysis of Critical Points in Smooth Optimization Benchmark Problems

Daniel Henderson

June 3, 2025

Abstract

The problem of locating and characterizing the critical points of a smooth objective function is fundametal to the design and analysis of optimization schemes using curvature information. We locate stationary points and analyze their local convexity properties for a set of benchmark problems in smooth, unconstrained optimization. For each problem, we review its origin, present the analytic form of the objective $f : \mathbb{R}^n \to \mathbb{R}$ and the standard starting point $\boldsymbol{x_0}$. We introduce a sampling-based procedure to classify the critical-point structure and verify the positive definiteness of the Hessian in a neighborhood of a strict local minimizer. Numerical experiments confirm that while some test functions exhibit strong local convexity, others contain narrow regions of non-convexity that can slow down standard schemes. **Keywords:** benchmarking, numerical optimization, stationary points, saddle points,

Contents

1 Intro

A bootlekneck of non-convex smooth optimization scheme is encountering plateaus in pursuit of a minimizer. A plateau is an almost "flat" region in some direction(s), leading to a critical point where the objective's gradient vanishes. We present theory and an algorithm to lacate and characterize the critical points of objective f. Each function f is chosen from `CUTEst` [2], the latest evolution of the *Constrained and Unconstrained Testing Environment* `CUTE` [1] which first appeared 30-years ago.

Strong local convexity near the objective's minimizer often guarantees rapid convergence for first-order schemes. In contrast, the presence of narrow plateau's can lead first-order schems to tack within the plateau. Under very mild regularity conditions on the objective, saddle points have little effect on the asymptotic behavior of first-order methods [5]. However, in practice such methods exhibit slow convergence and possible convergence to a saddle point.

We analyze the objective of each problem to reveal long, narrow valleys of weak/zero curvature surrounding strict minimizers. Additinally, we analyze the spectral pattern of the Hessian of f at critical points obtained from seeding standard schemes from randomly sampled starting locations surrounding each problems initial iterate. Moreover, we replicate the findings of Kok et al. for the generalized Rosenbrock problem [4] and extend their analysis to a larger set of benchmark problems.

Our contributions are:

- sampling-based classification algorithm for characterizing the curvature within a neighborhood of ϵ-first-order stationary points.
- systematically compares the local convexity profiles of standard benchmark problems in continuous optimization. (ref artificial article)
- supplemental material containing analytical expressions f, and the accepted initial iterate $\boldsymbol{x_0}$, as well as a discussion of each problem's provenance and known properties from the literature.

Introduction overview:

- 1.1 establishes the notation adopted throughout the manuscript.
- 1.2 formulates the smooth unconstrained optimization problem.
- 1.3 formulates minimization problem as the search for stationary equilibrium solutions to the gradient-flow dynamical system.
- 1.4 relates phase-space analysis of the dynamical system to Morse theory, yeilding thoeretical tools to classify the local convexity of critical points of f.
- 1.5 formalizes a constructive definiton for an *optimization scheme* for solving our unconstrained optimization problem, accompanied by illustrative examples that recover classical results for gradient descent.
- 1.6 enumerates the benchmark test functions considered in this study.

Organization of paper:.

- Section 2 details our problem selection criteria and introduces the benchmark problems under study.
- Section 3, we present numerical experiments for each problem, describe our methodology for determining strict local minimizers $\boldsymbol{x}^*$, and outline our sampling-based convexity classification algorithm
- Section 4 summarizes our findings, offers practical recommendations, and discusses directions for future work.
- Appendix 5 contains supplementary material, including code listing and analytical expressions for each problem's objective function f and the initial iterate $\boldsymbol{x_0}$.

1.1 Preliminaries

We fix our notation as declared in Appendix 5 as used by Nocedal and Wright [7], and we assume the following standing assumptions hold throughout the manuscript.

Assumption 1.1 (Standing Assumptions)**.**

- **Domain.** $\Omega \subset \mathbb{R}^n$ is path-connected, bounded, and open, and its boundary $\partial\Omega$ is a C^k $(k \geq 1)$ embedded hypersurface.

- **Manifold structure.** The closure $\overline{\Omega} = \Omega \cup \partial\Omega$ is compact and carries the structure of a C^k manifold with boundary $\partial\Omega$.

- **Objective function.** The objective satisfies $f \in C^k(\overline{\Omega})$ with $k \geq 2$, i.e., f and it's derivatives extend continuously to $\partial\Omega$.

TODO - Add ρ-Hessian Lipschitz and ℓ-smooth assumptions? - Critical Point Non-Degeneracy? - Path-Connectedness of Ω? Allows us to assume that f is convex of on sublevel sets... needed for morse theory and further results in dynamical systems, e.g., gradient flow retracts sub-level sets onto unstable manifolds..

1.2 Motivation

Consider the general form of a smooth unconstrained optimization problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ denotes the optimization variable and $f : \mathbb{R}^n \to \mathbb{R}$ denotes the objective function. An optimization scheme seeks an *optimal value* and *optimal solution*

$$\boldsymbol{x^*} \in \arg\min_{\boldsymbol{x} \in \overline{\Omega}} f(\boldsymbol{x}) \quad \text{and} \quad f^* = \min_{\boldsymbol{x} \in \overline{\Omega}} f(\boldsymbol{x}).$$

over some compact closed subset $\overline{\Omega} \subset \mathbb{R}^n$ from some initial starting point $\boldsymbol{x_0} \in \overline{\Omega}$. Finding an optimal value and solution for a nonconvex objective function over a continuous high-dimensional domain $\overline{\Omega}$ is a core problem to engineering and scienctific computing. When $\overline{\Omega} = \mathbb{R}^n$, as in 1, finding f^* and $\boldsymbol{x^*}$ for arbitary f is known to be NP-hard, as it is a search problem with an infinite state space. So, even in one-dimensional cases, there exist $f \in \mathbb{C}^k(\mathbb{R};\mathbb{R})$ where f^* and $\boldsymbol{x^*}$ cannot be determined in finite time.

cite

cite

We choose Ω in accordance to our standing assumptions 1.1; namely, Ω is a bounded and open subset of $\mathbb{R}^n$ so the Heine-Borel theorem asserts that the closure $\overline{\Omega} = \Omega \cup \partial\Omega$ is compact. By the extreme-value theorem, the objective f attains its maximum and minimum values on $\overline{\Omega}$, so there must exist a global optimal value on $\overline{\Omega}$. Note that we specify Ω based on the context, e.g., it may be a neighborhood about a stationary point or the sublevel set containing all the iterates of an optimization scheme, i.e., $\Omega = L^-_{f(\boldsymbol{x_0})}$.

A point where the gradient vanishes, (i.e., where $\nabla f = \mathbf{0}$) satisfies the first-order optimality condition and we refer to such points as *critical points*. Within a neighborhood surrounding such critical points, it is either a maximizer, minimizer, or a saddle point. From Nocedal and Wright [7], we fromally define minimizers of f:

- A *local minimizer* if there exists a neighborhood N of $\boldsymbol{x^*}$ such that $f(\boldsymbol{x^*}) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in N$.

- A *strict local minimizer* if there exists a neighborhood N of $\boldsymbol{x}^*$ such that $f(\boldsymbol{x}^*) < f(\boldsymbol{x})$ for all $\boldsymbol{x} \in N \setminus \{\boldsymbol{x}^*\}$.
- An *isolated local minimizer* if there exists a neighborhood N of $\boldsymbol{x}^*$ such that $\boldsymbol{x}^*$ is the unique local minimizer in N.
- A *global minimizer* if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$.

The necessary condition for $\boldsymbol{x}^*$ to be a local minimizer of f is that $\boldsymbol{x}^*$ satisfies the first-order and second-order optimality conditions. The *second-order* optimality condition states that the Hessian of f is positive semidefinite at $\boldsymbol{x}^*$. If the Hessian is strictly positive definite at $\boldsymbol{x}^*$, it is both a necessary and a sufficient condition that $\boldsymbol{x}^*$ is a strict local minimizer. Moreover, a strict local minimizer is isolated minimizer.

1.3 Continuous Model

In this section, we establish a link between the minimization problem (1) and the trajectories of a dynamical system by studying the gradient flow associated with the objective f.

Definition 1. The *gradient-flow* dynamical system associated with the objective function f is defined by the autonomous ordinary differential equation (ODE)

$$\boldsymbol{\gamma}'(t) = -\nabla f(\boldsymbol{\gamma}(t)) \;\text{ subject to } \boldsymbol{\gamma}(0) = \boldsymbol{x_0} \in \overline{\Omega} \tag{GF}$$

An integral solution $\boldsymbol{\gamma}(t)$ to the gradient-flow ODE (GF) with initial condition $\boldsymbol{\gamma}(0) = \boldsymbol{x_0} \in \overline{\Omega}$ is called a *gradient flow-line*, or simply a *trajectory.*

Critical points of the objective function f correspond precisely to equilibrium solutions of the gradient-flow ODE. Indeed, a point $\boldsymbol{x}_0$ is a critical point if and only if $\nabla f(\boldsymbol{x}_0) = \mathbf{0}$, which implies that $\boldsymbol{\gamma}(t) = \boldsymbol{x}_0$ is a stationary solution of (GF). Next, we address the existence, uniqueness, and global well-posedness of trajectories emanating from any initial point $\boldsymbol{x_0} \in \overline{\Omega}$.

theorem (Well-Posedness of Gradient Flow)**.** Given $f \in C^k(\overline{\Omega})$ and the compactness of $\overline{\Omega}$, the gradient-flow ODE (GF) admits a unique trajectory $\boldsymbol{\gamma}_{\boldsymbol{x_0}}(t)$ defined for all $t \geq 0$ for each initial condition $\boldsymbol{x_0} \in \overline{\Omega}$.

Proof. Since f is continuous on compact $\overline{\Omega}$, it is globally Lipschitz on $\overline{\Omega}$ Since f is continously differentiable on compact $\overline{\Omega}$, ∇f is globally Lipschitz on $\overline{\Omega}$. Therefore, the Picard-Lindelöf theorem ensures global existence and uniqueness of the solution trajectory $\boldsymbol{\gamma}_{\boldsymbol{x_0}}(t)$ □

For any $\boldsymbol{x_0} \in \overline{\Omega}$, the trajectory $\boldsymbol{\gamma}_{\boldsymbol{x_0}}$ is driven by the steepest descent direction of the objective function f. But f is bounded below by f^* on the compact set $\overline{\Omega}$, so we expect the trajectory $\boldsymbol{\gamma}_{\boldsymbol{x_0}}$ cannot escape $\overline{\Omega}$ in finite time. Indeed, we aim to show that the trajectory $\boldsymbol{\gamma}_{\boldsymbol{x_0}}$ converges to a critical point of f in $\overline{\Omega}$ as $t \to \infty$. We start by characterizing the monotonicity of f along the trajectory $\boldsymbol{\gamma}_{\boldsymbol{x_0}}$ in the following lemma.

Lemma A. *Along any gradient-flow trajectory $\boldsymbol{\gamma}_{\boldsymbol{x_0}}(t)$ one has*

$$\begin{aligned}\frac{d}{dt}(f \circ \boldsymbol{\gamma}_{\boldsymbol{x_0}}(t)) &= \nabla f(\boldsymbol{\gamma}_{\boldsymbol{x_0}}(t)) \cdot (-\nabla f(\boldsymbol{\gamma}_{\boldsymbol{x_0}}(t))) \\ &= -\|\nabla f(\boldsymbol{\gamma}_{\boldsymbol{x_0}}(t))\|^2 \leq 0,\end{aligned}$$

Hence $f \circ \boldsymbol{\gamma}_{\boldsymbol{x_0}}$ is nonincreasing in t and strictly decreasing whenever $\nabla f(\boldsymbol{\gamma}_{\boldsymbol{x_0}}(t)) \neq 0$.

Proof. Using the chain rule to differentiate $f \circ \boldsymbol{\gamma}_{\boldsymbol{x}_0}$ w.r.t. t yields

$$\begin{aligned}\frac{d}{dt} f\big(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)\big) &= \nabla f\big(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)\big) \cdot \frac{d}{dt}\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t) \\ &= \nabla f\big(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)\big) \cdot \boldsymbol{x}'_{\boldsymbol{x}_0}(t) \\ &= \nabla f\big(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)\big) \cdot \big(-\nabla f\big(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)\big)\big) \\ &= \langle \nabla f\big(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)\big), -\nabla f\big(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)\big)\rangle \\ &= -\langle \nabla f\big(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)\big), \nabla f\big(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)\big)\rangle \\ &= -\|\nabla f\big(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)\big)\|^2 \\ &\leq 0.\end{aligned}$$

A direct consequence of the positive-definiteness property of a norm is that the final inequality is strict for all points $\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)$ that aren't critical. □

The monotone descent of f along trajectory $\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)$ ensures that the Lyapunov function $V(\boldsymbol{x}) := f(\boldsymbol{x}) - f^*$ is monotone on $\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)$. As shown in the proof of Lemma A (and since f^* is constant), differentiating V along any trajectory yeilds

$$\dot{V}(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)) = -\|\nabla f(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t))\|^2 \leq 0.$$

Hence, V is stricly decreasing until the trajectory $\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)$ reaches a critical point. The following Lemma demonstrates convergence by showing that trajectory $\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)$ remains in $\overline{\Omega}$ for all $t \geq 0$, by the campactness of $\overline{\Omega}$

Lemma B. *For any gradient-flow trajectory $\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)$, there exists atleast one accumulation point $\boldsymbol{x}^\infty \in \overline{\Omega}$ where*

$$\boldsymbol{x}_\infty := \lim_{t\to\infty} \boldsymbol{\gamma}_{\boldsymbol{x}_0}(t),$$

such that $\boldsymbol{x}^\infty \in \mathrm{crit}(f)(\overline{\Omega}) = \{\boldsymbol{x} \in \overline{\Omega} \mid \nabla f(\boldsymbol{x}) = \mathbf{0}\}$.

Proof. Since $\overline{\Omega}$ is compact and the trajectory $\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)$ remains within $\overline{\Omega}$ for all $t \geq 0$, the trajectory is bounded. By the Bolzano-Weierstrass theorem, the trajectory has at least one accumulation point $\boldsymbol{x}^\infty \in \overline{\Omega}$. From Lemma A, we have that $\frac{d}{dt} f(\boldsymbol{x}_{\boldsymbol{x}0}(t)) = -\|\nabla f(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t))\|^2 \leq 0$, implying that $f(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t))$ is non-increasing and bounded below by f^*. Therefore, the limit $\lim_{t\to\infty} f(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t))$ exists. Suppose the contrary that $\nabla f(\boldsymbol{x}^\infty) \neq \mathbf{0}$. Since ∇f is continuous, there exists a neighborhood around $\boldsymbol{x}^\infty$ where $|\nabla f(\boldsymbol{x})| \geq \delta > 0$. This would imply that $\frac{d}{dt} f(\boldsymbol{x}_{\boldsymbol{x}0}(t)) \leq -\delta^2$ in this neighborhood, leading to $f(\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)) \to -\infty$ as $t \to \infty$, which contradicts the boundedness of f. Hence, $\nabla f(\boldsymbol{x}^\infty) = \mathbf{0}$, and $\boldsymbol{x}^\infty$ is a critical point. □

clean
parag

In the above lemma, we demonstrated that $\overline{\Omega}$ is *positvely invariant* which means that if a trajectory starts in $\overline{\Omega}$, it remains $\overline{\Omega}$ for all future times. Next, we characterize the accumulation points as an invariant subset of the set of critical points of f on $\overline{\Omega}$.

theorem (LaSalle's Invariance Principle)**.** Any gradient-flow trajectory $\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)$ converges to the largest invariant set within the critical points of f on $\overline{\Omega}$.

Proof. Define the Lyapunov function $V(\boldsymbol{x}) := f(\boldsymbol{x}) - f^*$. From previous results, we have $\dot{V}(\boldsymbol{x}_{\boldsymbol{x}0}(t)) = -|\nabla f(\boldsymbol{x}\boldsymbol{x}_{\mathbf{0}}(t))|^2 \leq 0$, indicating that V is non-increasing along trajectories. Let

$$\begin{aligned} S &:= \boldsymbol{x} \in \overline{\Omega} \mid \dot{V}(\boldsymbol{x}) = 0 \\ &= \boldsymbol{x} \in \overline{\Omega} \mid \nabla f(\boldsymbol{x}) = \mathbf{0},\end{aligned}$$

the set of critical points. By LaSalle's Invariance Principle, the trajectory $\boldsymbol{\gamma}_{\boldsymbol{x}_0}(t)$ approaches the largest invariant set contained in S as $t \to \infty$. Therefore, the trajectory converges to an invariant subset of the critical points of f on $\overline{\Omega}$. □

check

The following corollary asserts a condition when $\boldsymbol{\gamma}_{\boldsymbol{x_0}}(t)$ converges to a single critical point, rather than, a set of accumulation points.

Corollary C. *Suppose all critical points of f are isolated. Then each trajectory $\boldsymbol{\gamma}_{\boldsymbol{x_0}}(t)$ converges to a unique critical point, asymptotically approaching a strict local minimizer of f.*

Proof. From the previous theorem, the trajectory converges to an invariant subset of the set of critical points. If all critical points are isolated, the only invariant subsets are the individual critical points themselves. So, the trajectory must converge to a single critical point. Such limit point must be a loccal minimizer, since f decreases along the trajectory until it reachs such critical point. And such critical point must be a strict local minizer by our assumption that all critical points of f are isolated. □

Each critical point attracts a certain region of initial conditions (its stable manifold).

Definition 2. Given any critical point $\boldsymbol{x}^* \in \text{crit}(f)$ we define the stable **stable manifold** W^s and **unstable manifold** W^u of $\boldsymbol{x}^*$ as

$$W^S(\boldsymbol{x}^*) := \{\boldsymbol{x} \in \overline{\Omega} : \lim_{t\to\infty} \boldsymbol{x}_{\boldsymbol{x}}(t) = \boldsymbol{x}^*\},$$
$$W^u(\boldsymbol{x}^*) := \{\boldsymbol{x} \in \overline{\Omega} : \lim_{t\to-\infty} \boldsymbol{x}_{\boldsymbol{x}}(t) = \boldsymbol{x}^*\}.$$

The stable manifold of $\boldsymbol{x}^*$ is the set of initial conditions that converge to $\boldsymbol{x}^*$, where as the unstable manifold of $\boldsymbol{x}^*$ is the set of initial conditions that asymptotically depart $\boldsymbol{x}^*$ as t increases.

A critical point $\boldsymbol{x}^*$ is *degenerate* if $\nabla^2 f(\boldsymbol{x}^*)$ is singular. Otherwise, we say $\boldsymbol{x}^*$ is *non-degenerate*, and it holds that $\nabla^2 f(\boldsymbol{x}^*)$ is invertible. Equivalently, non-degeneracy means all eigenvalues of $\nabla^2 f(\boldsymbol{x}^*)$ are nonzero.

Definition 3. The **index** of a non-degenerate critical point, denoted index$(\boldsymbol{x}^*)$, is the dimension of the negative eigenspace of $\nabla^2 f(\boldsymbol{x}^*)$.

Intuitively, the index of a non-degenerate critical point equals the number of linearly independent descent directions.

theorem (Equilibria Classification and Stability)**.** Let $\boldsymbol{x}^*$ be a non-degenerate critical point of f on $\overline{\Omega}$, then:

- if index$(\boldsymbol{x}^*) = 0$, $\boldsymbol{x}^*$ is a strict minimizer and an asymptotically stable equilibrium.
- if $0 <$ index$(\boldsymbol{x}^*) < n$, $\boldsymbol{x}^*$ is a saddlepoint of f and unstable in some directions.
- $\dim W^u(\boldsymbol{x}^*) = \text{index}(\boldsymbol{x}^*)$ and $\dim W^s(\boldsymbol{x}^*) = n - \text{index}(\boldsymbol{x}^*)$.

where $n = \dim \overline{\Omega}$.

Proof. At a non-degenerate critical point $\boldsymbol{x}^*$, the Hessian matrix $\nabla^2 f(\boldsymbol{x}^*)$ is invertible. The eigenvalues of the Hessian determine the nature of the critical point. If all eigenvalues are positive, $\boldsymbol{x}^*$ is a strict local minimum and the gradient flow is asymptotically stable in all directions. If some eigenvalues are negative, $\boldsymbol{x}^*$ is a saddle point, and the flow is unstable in the directions corresponding to the negative eigenvalues. The index index$(\boldsymbol{x}^*)$ counts the number of negative eigenvalues, which corresponds to the dimension of the unstable manifold $W^u(\boldsymbol{x}^*)$. The remaining $n - \text{index}(\boldsymbol{x}^*)$ directions correspond to the stable manifold $W^s(\boldsymbol{x}^*)$. □

check
use f
ment
rem
algeb

In summary, $\boldsymbol{\gamma}(t)$ is well-defined for $t \geq 0$, remains in $\overline{\Omega}$, and $f(\boldsymbol{x}(t))$ decreases monotonically to a limit point. Such point is unique when the critical points of f are isolated. Otherwise, analysis indicates that the dynamical system partitions $\overline{\Omega}$ into invariant sets associated with critical points. Morse theory formalizes this connection: critical points yield topological decompositions, where each nondegenerate critical point contributes to the manifold structure via its stable and unstable manifolds. We turn next to a detailed topological classification of these invariant sets using Morse theory, before considering optimization schemes that approximate the continuous dynamics.

1.4 Morse Theory

In this section, we apply the stability theory gradient-flow dynamical system to results in Morse theory. For a modern introduction to Morse theory, see Hall [3].

Assumption 1.2. Suppose f is analytic on $\overline{\Omega}$

The assumption that f is analytic implies that $f \in C^{\infty}(\overline{\Omega}; \mathbb{R})$. Taylor series expansion of f converges to f in a neighborhood of every point in $\overline{\Omega}$. As we aim to show, assumption 1.2 leads to the strict saddle point condition that all critical points of f are isolated. This is a key property of Morse functions, which are smooth functions with non-degenerate critical points.

Recall, by 1.1, $\overline{\Omega}$ is a compact Manifold. Proceeding we impose that the objective f is real-analytic on $\overline{\Omega}$, i.e., assumption 1.2 holds. Then analytic f implies the gradient-flow trajectory $\boldsymbol{\gamma}_{\boldsymbol{x_0}}$ emanating from a point $\boldsymbol{x_0}$ converges to a single critical point $\boldsymbol{x}^*$ at $t \to \infty$. Since trajectory $\boldsymbol{\gamma}_{\boldsymbol{x_0}}$ has only one limit point, such point is isolated and a strict local mininimizer of f on some neighborhood in $\overline{\Omega}$ (by theorem C). Consequently, $\boldsymbol{x}^*$ satisfies the second-order optimality condition that $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite, so eigenvalues have nonzero real parts greater than zero. Thus, $\det \nabla^2 f(\boldsymbol{x}^*) \neq 0$ implying that $\nabla^2 f$ is invertible at $\boldsymbol{x}^*$. Equivalently, $\boldsymbol{x}^*$ is non-degenerate strict local minimizer. So each critical point of real-anlytic function f on $\overline{\Omega}$ is non-degenerate, so, f is Morse on $\overline{\Omega}$.

Definition 4. A smooth function $f : \overline{\Omega} \to \mathbb{R}$ is **Morse** if all critical points of f are non-degenerate.

General Morse theory was developed from the realization that the critical points of f are constrained by the topology of f's pre-image. In 1939 by A.P. Morse demonstrated a particular are "rare" in the sense that they aren't a dense subset of the manifold $\overline{\Omega}$. Consequently, selecting a point in $\overline{\Omega}$ at random will almost never yeild a critical point of f.

theorem (Sard's theorem)**.** Let f be a Morse function on $\overline{\Omega}$, then the image $f(\text{crit}(f))$ has Lebesque measure zero in $\mathbb{R}$.

The property that $\boldsymbol{x}^* \in \overline{\Omega}$ being a *critical point* of a Morse function f is not dependent of the metric of $\overline{\Omega} \subset \mathbb{R}^n$ (and consequently, the norm induced by the metric) despite the fact that our analysis is based on the usual Euclidean norm, induced by the inner product.

theorem. For a Morse function f on $\overline{\Omega}$, the gradient of f is either zero or orthogonal to the tangent space of the level set L_c at $\boldsymbol{x} \in L_c$.

How
relat

theorem. Let f be a Morse function on Ω, then the Euler characteristic of Ω is given by

$$\chi(\Omega) = \sum_{i=0}^{n} (-1)^i c_i$$

where c_i is the number of critical points of index i.

Remark. The Euler characteristic $\chi(\Omega)$ is a topological invariant of the manifold Ω and is independent of the choice of Morse function f. The Euler characteristic is a measure of the "shape" of the manifold and can be used to distinguish between different topological spaces. The Euler characteristic may be defined by the alternating sum of the Betti numbers b_i of the manifold Ω

$$\chi(\Omega) = \sum_{i=0}^{n} (-1)^i b_i$$

where b_i is the i-th Betti number of the manifold Ω.

The above theorem implies that at a stationary point $\boldsymbol{x}^*$, a level set $L_{\boldsymbol{x}^*}$ is reduced to a single point when $\boldsymbol{x}^*$ is a local minimum or maximum. Otherwise, the level set may have a singularity such as a self-intersection or a cusp.

1.5 Optimization Schemes

We introduce a *scheme* for solving a *unconstrained optimization problem* based on the *gradient flow* dynamical system 1.

Definition 5. An **optimization scheme** is a one-parameter family of iteration operators $T_h : \overline{\Omega} \to \overline{\Omega}$, indexed by a step size $h \in (0, h_0]$ where h_0 is constant, that generates an iterative sequence using the rule

$$\boldsymbol{x}_{k+1} = T_h(\boldsymbol{x}_k) \text{ for } k = 0, 1, 2, \dots \tag{2}$$

starting from an initial point $\boldsymbol{x_0} \in \overline{\Omega}$. The scheme is well-defined such that the triplet $(\boldsymbol{x_0}, h, T_h)$ satisfy:

- Consistency: T_h is *consistent of order* $p \geq 1$ with the flow (GF) if

  $$\left\| T_h(\boldsymbol{x}) - \boldsymbol{x} + h\,\nabla f(\boldsymbol{x}) \right\| \;=\; \mathcal{O}\!\left(h^{p+1}\right) \quad \text{as } h \to 0,\ \forall \boldsymbol{x} \in \overline{\Omega}.$$

  A consistent scheme approximates the continuous gradient flow w/ a local error of $\mathcal{O}(h^{p+1})$ committed at each step k; where p is the global order. Consistent schemes reproduce the exact optimality condition in the limit as $h \to 0$.

- Stability: Let $\boldsymbol{x}^* \in \operatorname{crit}(f)$ be a *strict* local minimizer. The family $\{T_h\}$ is *stable* at $\boldsymbol{x}^*$ if

  $$\exists\, c > 0,\ h_{\max} > 0,\ \forall\, h \in (0, h_{\max}] : \qquad \rho\big(DT_h(\boldsymbol{x}^*)\big) \leq 1 - ch,$$

  where $\rho(\cdot)$ denotes the spectral radius. Equivalently,

  $$\|T_h(\boldsymbol{x}) - T_h(\boldsymbol{y})\| \leq (1 - ch)\|\boldsymbol{x} - \boldsymbol{y}\|$$

  for all $\boldsymbol{x}, \boldsymbol{y}$ in some neighborhood of $\boldsymbol{x}^*$. A stable scheme is contractive in a neighborhood of $\boldsymbol{x}^*$, meaning that the distance between iterates shrinks by at least a factor of $1 - ch$ at each step k.

If a scheme is order-p consistent and locally contractive at a strict minimizer $\boldsymbol{x}^*$, then for any fixed $h \in (0, h_{\max}]$ the iterates satisfy

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*\| \leq (1 - ch)^k \|\boldsymbol{x_0} - \boldsymbol{x}^*\|,$$

and hence $\boldsymbol{x}_k \to \boldsymbol{x}^*$ as $k \to \infty$.

Table 1.5 summarises four standard choices for T_h, stating their consistency order and the conditions under which local contractivity holds; see, e.g., Nocedal-Wright [7] for detailed discussions of these schemes.

Scheme	**Iteration operator** $T_h(\boldsymbol{x})$	**Order** p	**Stability near** $\boldsymbol{x}^*$
Gradient descent (GD)	$\boldsymbol{x} - h\,\nabla f(\boldsymbol{x})$	1	$\nabla^2 f(\boldsymbol{x}^*) \succeq \mu I \succ 0,\ 0 < h \leq 1/\ell$
Newton (NM)	$\boldsymbol{x} - h\,\nabla^2 f(\boldsymbol{x})^{-1}\nabla f(\boldsymbol{x})$	2	$\nabla^2 f(\boldsymbol{x}^*) \succ 0,\ 0 < h \leq 1$
Quasi–Newton (QN)	$\boldsymbol{x} - h\,B_k \nabla f(\boldsymbol{x}) \quad (B_k \to \nabla^2 f(\boldsymbol{x}^*)^{-1})$	1	Same as NM once $B_k \succ 0$ and $\|B_k\|$ is bounded
Trust region (TR)	$\boldsymbol{x} + \arg\min_{\|\boldsymbol{\tau}\| \leq \Delta} m_{\boldsymbol{x}}(\boldsymbol{\tau})$	2	Same as NM for sufficiently small Δ

Analysis of Gradient Descent (GD)

theorem. Assume f is ℓ-smooth and α-strongly convex and that $\epsilon > 0$. If we iterate the gradient descent *scheme* with $h = h_0 = \frac{1}{\ell}$ held fixed, i.e.,

$$T_h(\boldsymbol{x}_k) = \boldsymbol{x}_k - \frac{1}{\ell}\nabla f(\boldsymbol{x}_k),$$

then $d(x_k, x^*) \leq \epsilon$ for all $k > K$ where K is chosen to satisfy

$$\frac{2\ell}{\alpha} \cdot \log\left(\frac{d(x_0, x^*)}{\epsilon}\right) \leq K.$$

Remark. Under *ℓ-smoothness* and *α-strong convexity* assumptions in a neighborhood Ω about $\boldsymbol{x}^*$, it may be shown directly from the above theorem above that the *GD scheme* converges linearly to the optimal solution $\boldsymbol{x}^*$ at a rate of

$$\frac{d(\boldsymbol{x}_k, \boldsymbol{x}^*)}{d(\boldsymbol{x}_{k-1}, \boldsymbol{x}^*)} \leq 1 - \frac{\alpha}{\ell}$$

where $d(\boldsymbol{x}_k, \boldsymbol{x}^*)$ is the distance between the current iterate $\boldsymbol{x}_k$ and the optimal solution $\boldsymbol{x}^*$. The convergence rate is linear in the sense that the distance between the current iterate and the optimal solution decreases by a factor of $1 - \frac{\alpha}{\ell}$ at each iteration. (Ref: TODO)

Remark. Convergence to a first-order stationary point trivally implies convergence to a ϵ-first-order stationary point. Similarly, convergence to a second-order stationary point trivially implies convergence to a ϵ-second-order stationary point.

theorem. Assume f is ℓ-smooth, then for any $\epsilon > 0$, if we iterate the GD scheme with $h = h_0 = \frac{1}{\ell}$ held fixed starting from $\boldsymbol{x_0} \in \Omega$ where Ω is a neighborhood of $\boldsymbol{x}^*$, then the number of iterations K required to achieve the stopping condition $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon$ is at most

$$\left\lceil \frac{\ell}{\epsilon^2}\left(f(\boldsymbol{x_0}) - f(\boldsymbol{x}^*)\right) \right\rceil$$

Remark. **TODO and Questions**

- State how we use theorems in when performing analysis from the results of our experimewts.
- What is the relationship between ℓ and α?
- In practice do we know how to compute ℓ and α?
- What is the relationship between ℓ and ρ?
- In practice do we know how to compute ℓ and ρ?

1.6 Benchmark Problems

We select a subset of the `OptimizationProblems.jl` [6] Julia package that support automatic differentiation (AD) natively through operator overloading. (TODO: Cite ForwardDiff.jl, Flux.jl, etc.) Each problem is implemented as `ADNLPModel` instance, which is a wrapper around the `NLPModel` interface whose backend AD engine is configurable to support forward-mode or reverse-mode.

todo: - ref overleaf document and introduction section - enumerate the problems

2 Problem Selection Criterion and Software Stack

3 Numerical Experiments

An uniformed sampling of the problem space Ω is performed in Code Listings 1.

4 Conclusion

reference introduction section.

5 Apendix

References

References

[1] Ingrid Bongartz, Andrew R. Conn, Nick I. M. Gould, and Philippe L. Toint. Cute: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, 1995.

[2] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015.

[3] Stephen F. Hall. Morse theory. `https://stanford.edu/~sfh/morse.pdf`, 2011. Lecture notes, Stanford University.

[4] Schalk Kok and Carl Sandrock. Locating and characterizing the stationary points of the extended rosenbrock function. *Evolutionary Computation*, 17(3):437–453, 2009. © 2009 by the Massachusetts Institute of Technology.

[5] Jason D. Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. First-order methods almost always avoid saddle points. *arXiv preprint arXiv:1710.07406*, 2017.

[6] Tangi Migot, Dominique Orban, and Abel Soares Siqueira. Optimizationproblems.jl: A collection of optimization problems in julia. If you use this software, please cite it using the metadata from this file.

[7] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

Notation

We assume the following notation throughout

- $\|\cdot\|$ denotes the usual ℓ_2 norm for vectors $\boldsymbol{x}$ in $\mathbb{R}^n$ and $p = 2$ norm for matrices in $\mathbb{R}^{n\times m}$. i.e.,

$$\|x\| := \left(\sum_i x_i^2\right)^{1/2}$$
$$\|A\| := (\lambda_{\max}(A^\top A))^{1/2} = \max(\sigma(A))$$

- $\sigma(A) := \{\text{singular values of } A\}$.
- $A \in \mathbb{R}^{n\times n} \implies \sigma(A) = \{\text{eigenvalues of } A \text{ (i.e. spectrum)}\}$
- $\sigma_{\max}(A) := \max(\sigma(A))$ and $\sigma_{\min}(A) := \min(\sigma(A))$.
- $A \in \mathbb{R}^{n\times n} \implies \lambda_{\max}(A) := \sigma_{\max}(A)$ and $\lambda_{\min}(A) = \sigma_{\min}(A)$
- $\langle\cdot,\cdot\rangle$ denotes the usual inner product on $\mathbb{R}^n$, i.e.,

$$\langle \boldsymbol{x}, \boldsymbol{y}\rangle := \boldsymbol{x}^\top \boldsymbol{y} = \sum_{i=1}^n \boldsymbol{x}_i \boldsymbol{y}_i = \|\boldsymbol{x}\|\|\boldsymbol{y}\|\cos(\theta)$$

  where θ is the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$.

- $\mathcal{B}_r(\boldsymbol{x}) := \{\boldsymbol{y} \in \mathbb{R}^n : \|\boldsymbol{y} - \boldsymbol{x}\| < r\}$ is the open ball of radius r centered at $\boldsymbol{x} \in \mathbb{R}^n$.
- $\text{crit}(f) = \{\boldsymbol{x}^* \in \mathbb{R}^n : \nabla f(\boldsymbol{x}^*) = 0\}$ is the set of critical points of f.

5.1 Definitions

TODO: integrated into article as needed. At a minimum labled and references

Definition 6. f is **L-Lipschitz** if $\forall\ \boldsymbol{x_1}, \boldsymbol{x_2}$

$$\exists\ L \geq 0\ :\ \|f(\boldsymbol{x_1}) - f(\boldsymbol{x_2})\| \leq L\|\boldsymbol{x_1} - \boldsymbol{x_2}\|$$

Definition 7. f has **ℓ-Lipschitz gradient**, or, f is **ℓ-smooth** if $\forall\ \boldsymbol{x_1}, \boldsymbol{x_2}$

$$\exists\ \ell \geq 0\ :\ \|\nabla f(\boldsymbol{x_1}) - \nabla f(\boldsymbol{x_2})\| \leq \ell\|\boldsymbol{x_1} - \boldsymbol{x_2}\|$$

Definition 8. f has **ρ-Lipschitz Hessian** if $\forall\ \boldsymbol{x_1}, \boldsymbol{x_2}$

$$\exists\ \rho \geq 0\ :\ \|\nabla^2 f(\boldsymbol{x_1}) - \nabla^2 f(\boldsymbol{x_2})\| \leq \rho\|\boldsymbol{x_1} - \boldsymbol{x_2}\|$$

Definition 9. f is **convex** if $\forall\ \boldsymbol{x_1}, \boldsymbol{x_2}$

$$\begin{aligned} f(\boldsymbol{x_2}) &\geq f(\boldsymbol{x_1}) + \langle \boldsymbol{x_2} - \boldsymbol{x_1}, \nabla f(\boldsymbol{x_1})\rangle \\ &= f(\boldsymbol{x_1}) + \nabla f(\boldsymbol{x_1})^T(\boldsymbol{x_2} - \boldsymbol{x_1}) \end{aligned}$$

Definition 10. f is **strictly convex** if

$$\begin{aligned} &\exists\ \mu > 0\ : \nabla^2 f \succeq \mu I \\ &\iff \lambda_{\min}(\nabla^2 f) \geq \mu > 0 \end{aligned}$$

Definition 11. f is **α-strongly convex** if $\forall\ \boldsymbol{x_1}, \boldsymbol{x_2} \exists\ \alpha > 0$ s.t.

$$\begin{aligned} &f(\boldsymbol{x_2}) \geq f(\boldsymbol{x_1}) + \langle \nabla f(\boldsymbol{x_1}), \boldsymbol{x_2} - \boldsymbol{x_1}\rangle + \frac{\alpha}{2}\|\boldsymbol{x_2} - \boldsymbol{x_1}\|^2 \\ &\iff \lambda_{\min}(\nabla^2 f(\boldsymbol{x})) \geq -\alpha. \end{aligned}$$

Definition 12. $\boldsymbol{x^*}$ is a **first-order stationary point** if $\|\nabla f(x^*)\| = 0$.

Definition 13. $\boldsymbol{x^*}$ is an **ϵ-first-order stationary point** if $\|\nabla f(x^*)\| \leq \epsilon$.

Definition 14. $\boldsymbol{x^*} \in \mathbb{R}^n$ is a **second-order stationary point** if $\|\nabla f(x^*)\| = 0$ and $\nabla^2 f(x^*) \succeq 0$.

Definition 15. if f has ρ-Lipschitz Hessian, $\boldsymbol{x^*} \in \mathbb{R}^n$ is a **ϵ-second-order stationary point** if

$$\|\nabla f(x^*)\| \leq \epsilon \text{ and } \nabla^2 f(x^*) \succeq -\sqrt{\rho\epsilon}$$

Remark. Note that the Hessian is not required to be positive definite, but it is required to have a small eigenvalue.

Definition 16. A point $\boldsymbol{x^*} \in \Omega$ is a **critical point** of f if the differentiable map $df_p : T_p\Omega \to \mathbb{R}$ is zero. (Here $T_p\Omega$ is a tangent space of the Manifold M at p.) The set of critical points of f is denoted by crit(f).

Definition 17. A point $\boldsymbol{x^*} \in \Omega$ is a **non-degenerate critical point** of f if the Hessian $H_p f$ is non-singular.

Definition 18. The **index** of a *non-degenerate critical point* $\boldsymbol{x^*}$ is defined to be the dimension of the negative eigenspace of the Hessian $H_p f$.

- local minima at $\boldsymbol{x^*}$ have index 0.
- local maxima at $\boldsymbol{x^*}$ have index n.

- saddle points at $\boldsymbol{x}^*$ have index k where $0 < k < n$.

We reserve the integers $c_0, c_1, \ldots, c_i, \ldots, c_n$ to denote the number of critical points of index i.

Remark. For each objective function f we are interested in determining the critical points of f

Remark. The **Morse function** is a smooth function $f : \Omega \to \mathbb{R}$ such that all critical points of f are non-degenerate.

5.2 Test Problems

1. **Extended Trigonometric function:**

$$f(x) = \sum_{i=1}^{n} \left((n - \sum_{j=1}^{n} \cos(x_j)) + i\,(1 - \cos(x_i)) - \sin(x_i) \right)^2,$$

$$x_0 = [0.2, 0.2, \ldots, 0.2].$$

2. **Generalized Rosenbrock function:**

$$f(x) = \sum_{i=1}^{n-1} \left(c(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right),$$

$$x_0 = [-1.2, 1, -1.2, 1, \ldots, -1.2, 1], \quad c = 100.$$

3. **Extended Penalty function:**

$$f(x) = \sum_{i=1}^{n-1} (x_i - 1)^2 + \left(\sum_{j=1}^{n} x_j^2 - 0.25 \right)^2,$$

$$x_0 = [1, 2, \ldots, n].$$

4. **Perturbed Quadratic function:**

$$f(x) = \sum_{i=1}^{n} i x_i^2 + \frac{1}{100} \left(\sum_{i=1}^{n} x_i \right)^2$$

$$x_0 = [0.5, 0.5, \ldots, 0.5].$$

5. **Generalized Tridiagonal 1 function:**

$$f(x) = \sum_{i=1}^{n-1} (x_i + x_{i+1} - 3)^2 + (x_i - x_{i+1} + 1)^4,$$

$$x_0 = [2, 2, \ldots, 2].$$

6. **Generalized White & Holst function:**

$$f(x) = \sum_{i=1}^{n-1} c(x_{i+1} - x_i^3)^2 + (1 - x_i)^2, \quad c = 100,$$

$$x_0 = [-1.2, 1, -1.2, 1, \ldots, -1.2, 1].$$

7. **Generalized PSC1 function:**

$$f(x) = \sum_{i=1}^{n-1} \left(x_i^2 + x_{i+1}^2 + x_i x_{i+1} \right)^2 + \sin^2(x_i) + \cos^2(x_i),$$

$$x_0 = [3, 0.1, \ldots, 3, 0.1].$$

8. **Full Hessian FH1 function:**

$$f(x) = (x_1 - 3)^2 + \sum_{i=2}^{n} \left(x_1 - 3 - 2 \left(\sum_{j=1}^{i} x_j \right)^2 \right)^2,$$

$$x_0 = [0.01, 0.01, \ldots, 0.01].$$

9. **Full Hessian FH2 function:**

$$f(x) = (x_1 - 5)^2 + \sum_{i=2}^{n} \left(\sum_{j=1}^{i} x_j - 1 \right)^2,$$

$$x_0 = [0.01, 0.01, \ldots, 0.01].$$

10. **Perturbed Quadratic Diagonal function:**

$$f(x) = \left(\sum_{i=1}^{n} x_i \right)^2 + \sum_{i=1}^{n} \frac{i}{100} x_i^2,$$

$$x_0 = [0.5, 0.5, \ldots, 0.5].$$

11. **Quadratic QF1 function:**

$$f(x) = \frac{1}{2} \sum_{i=1}^{n} i x_i^2 - x_n,$$

$$x_0 = [1, 1, 1, \ldots, 1]$$

12. **Extended quadratic penalty QP1 function:**

$$f(x) = \sum_{i=1}^{n-1} \left(x_i^2 - 2 \right)^2 + \sum_{i=1}^{n} \left(x_i^2 - 0.5 \right)^2$$

$$x_0 = [1, 1, 1, \ldots, 1]$$

13. **Extended quadratic penalty QP2 function:**

$$f(x) = \left(\sum_{i=1}^{n} x_i^2 - 100 \right)^2 + \sum_{i=1}^{n-1} \left(x_i^2 - \sin x_i \right)^2$$

$$x_0 = [1, 1, 1, \ldots, 1]$$

14. **Quadratic QF2 function:**

$$f(x) = \frac{1}{2} \sum_{i=1}^{n} i \left(x_i^2 - 1 \right)^2 - x_n$$

$$x_0 = [0.5, 0.5, 0.5, \ldots, 0.5]$$

15. **Extended Tridiagonal 2 function:**

$$f(x) = \sum_{i=1}^{n-1} (x_i x_{i+1} - 1)^2 + c(x_i + 1)(x_{i+1} + 1)$$

$$x_0 = [1, 1, 1, \ldots, 1], \quad c = 0.1$$

16. **FLETCBV3 function (CUTE):**

$$f(x) = \frac{1}{2} p(x_1^2 + x_n^2) + \frac{p}{2} \sum_{i=1}^{n-1} (x_i - x_{i+1})^2 - \sum_{i=1}^{n} \left(\frac{p(h^2 + 2)}{h^2} x_i + \frac{cp}{h^2} \cos(x_i) \right),$$

where:

$$p = \frac{1}{10^8}, \quad h = \frac{1}{n+1}, \quad c = 1,$$

$$x_0 = [h, 2h, \dots, nh].$$

17. **FLETCHCR function (CUTE):**

$$f(x) = \sum_{i=1}^{n-1} c(x_{i+1} - x_i + 1 - x_i^2)^2$$

$$x_0 = [0, 0, 0, \dots, 0], \quad c = 100$$

18. **BDQRTIC function (CUTE):**

$$f(x) = \sum_{i=1}^{n-4} (-4x_i + 3)^2 + \left(x_i^2 + 2x_{i+1}^2 + 3x_{i+2}^2 + 4x_{i+3}^2 + 5x_n^2\right)^2,$$

$$x_0 = [1, 1, \dots, 1].$$

19. **TRIDIA function (CUTE):**

$$f(x) = \gamma(\delta x_1 - 1)^2 + \sum_{i=2}^{n} i\left(\alpha x_i - \beta x_{i-1}\right)^2$$

where:

$$\alpha = 2, \quad \beta = 1, \quad \gamma = 1, \quad \delta = 1$$

$$x_0 = [1, 1, 1, \dots, 1]$$

20. **ARWHEAD function (CUTE):**

$$f(x) = \sum_{i=1}^{n-1} (-4x_i + 3) + \sum_{i=1}^{n-1} \left(x_i^2 + x_n^2\right)^2$$

$$x_0 = [1, 1, 1, \dots, 1]$$

21. **NONDIA function (CUTE):**

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} 100\left(x_1 - x_{i-1}^2\right)^2,$$

$$x_0 = [-1, -1, -1, \dots, -1]$$

22. **NONDQUAR function (CUTE):**

$$f(x) = (x_1 - x_2)^2 + \sum_{i=1}^{n-2} (x_i + x_{i+1} + x_n)^4 + (x_{n+1} + x_n)^2,$$

$$x_0 = [1, -1, , \dots, 1, -1]$$

23. **EG2 function (CUTE):**

$$f(x) = \sum_{i=1}^{n-1} \sin(x_1 + x_i^2 - 1) + \frac{1}{2}\sin(x_n^2)$$

$$x_0 = [1, 1, 1, \dots, 1]$$

24. **CURLY20 function (CUTE):**

$$f(x) = \sum_{i=1}^{n} \left(q_i^4 - 20q_i - 0.1q_i \right)$$

where:

$$q_i = \begin{cases} x_i + x_{i+1} + x_{i+2} + \cdots + x_{i+k}, & \text{if } i \le n-k \\ x_i + x_{i+1} + x_{i+2} + \cdots + x_n, & \text{if } i > n-k \end{cases}$$

$$k = 20, \quad x_0 = \left[\frac{0.001}{n+1}, \ldots, \frac{0.001}{n+1} \right].$$

25. **Partially Perturbed Quadratic function:**

$$f(x) = x_1^2 + \sum_{i=1}^{n} \left[ix_i^2 + \frac{1}{100} \left(\sum_{j=1}^{i} x_j \right)^2 \right],$$

$$x_0 = [0.5, 0.5, \ldots, 0.5].$$

26. **Broyden Tridiagonal function:**

$$f(x) = \left(3x_1 - 2x_1^2\right)^2 + \sum_{i=2}^{n-1} \left(3x_i - 2x_i^2 - x_{i-1} - 2x_{i+1} + 1\right)^2 + \left(3x_n - 2x_n^2 - x_{n-1} + 1\right)^2,$$

$$x_0 = [-1, -1, \ldots, -1].$$

27. **Perturbed Tridiagonal Quadratic function:**

$$f(x) = x_1^2 + \sum_{i=2}^{n-1} ix_i^2 + (x_{i-1} + x_i + x_{i+1})^2,$$

$$x_0 = [0.5, 0.5, \ldots, 0.5].$$

28. **Staircase 1 function:**

$$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2,$$

$$x_0 = [1, 1, \ldots, 1].$$

29. **Staircase 2 function:**

$$f(x) = \sum_{i=1}^{n} \left[\left(\sum_{j=1}^{i} x_j \right) - i \right]^2,$$

$$x_0 = [0, 0, \ldots, 0].$$

30. **ENGVAL1 function (CUTE):**

$$f(x) = \sum_{i=1}^{n-1} \left(x_i^2 + x_{i+1}^2 \right)^2 + \sum_{i=1}^{n-1} \left(-4x_i + 3 \right),$$

$$x_0 = [2, 2, \ldots, 2].$$

31. **EDENSCH function (CUTE):**

$$f(x) = 16 + \sum_{i=1}^{n-1} \left[(x_i - 2)^4 + (x_i x_{i+1} - 2x_{i+1})^2 + (x_{i+1} + 1)^2\right]$$

$$x_0 = [0, 0, \ldots, 0].$$

32. **INDEF function (CUTE):**

$$f(x) = \sum_{i=2}^{n-1} \cos(2x_i - x_n - x_1) + \sum_{i=1}^{n} x_i,$$

$$x_0 = \left[\frac{1}{n+1}, \frac{2}{n+1}, \ldots, \frac{n}{n+1}\right].$$

33. **CUBE function (CUTE):**

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} 100 \left(x_i - x_{i-1}^3\right)^2,$$

$$x_0 = [-1.2, 1, -1.2, 1, \ldots].$$

34. **EXPLIN1 function (CUTE):**

$$f(x) = \sum_{i=1}^{n} \exp(0.1 x_i x_{i+1}) - 10 \sum_{i=1}^{n} i x_i,$$

$$x_0 = [0, 0, \ldots, 0].$$

35. **BDEXP function (CUTE):**

$$f(x) = \sum_{i=1}^{n-2} (x_i + x_{i+1}) \exp(-x_{i+2}(x_i + x_{i+1}))$$

$$x_0 = [1, 1, \ldots, 1].$$

36. **HARKERP2 function (CUTE):**

$$f(x) = \left(\sum_{i=1}^{n} x_i\right)^2 - \sum_{i=1}^{n} (x_i + \frac{1}{2} x_i^2) + 2 \sum_{i=2}^{n} \left(\sum_{i=j}^{n} x_i\right)^2,$$

$$x_0 = [1, 2, ..., n].$$

37. **GENHUMPS function (CUTE):**

$$f(x) = \sum_{i=1}^{n-1} \left(\sin(2x_i)^2 \sin(2x_{i+1})^2 + 0.05(x_i^2 + x_{i+1}^2)\right),$$

$$x_0 = [-506, 506.2, \ldots, 506.2].$$

38. **MCCORMCK function (CUTE):**

$$f(x) = \sum_{i=1}^{n-1} \left(-1.5x_i + 2.5x_{i+1} + 1 + (x_i - x_{i+1})^2 + \sin(x_i + x_{i+1})\right)$$

$$x_0 = [1, 1, \ldots].$$

39. **NONSCOMP function (CUTE):**

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} 4(x_i - x_{i-1}^2)^2$$

$$x_0 = [3, 3, \dots].$$

40. **SINQUAD function (CUTE):**

$$f(x) = (x_1 - 1)^4 + \sum_{i=2}^{n-1} \left(\sin(x_i - x_n) - x_1^2 + x_i^2\right)^2 + (x_n^2 - x_1^2)^2,$$

$$x_0 = [0.1, 0.1, \dots].$$

41. **LIARWHD function (CUTE):**

$$f(x) = \sum_{i=1}^{n} 4\left(x_i^2 - x_1\right)^2 + \sum_{i=1}^{n} (x_i - 1)^2$$

$$x_0 = [4, 4, 4, \dots, 4]$$

42. **DIXON3DQ function (CUTE):**

$$f(x) = (x_1 - 1)^2 + \sum_{i=1}^{n-1} (x_i - x_{i+1})^2 + (x_n - 1)^2,$$

$$x_0 = [-1, -1, \dots, -1].$$

43. **COSINE function (CUTE):**

$$f(x) = \sum_{i=1}^{n-1} \cos(-0.5x_{i+1} + x_i^2),$$

$$x_0 = [1, 1, \dots, 1].$$

44. **SINE function:**

$$f(x) = \sum_{i=1}^{n-1} \sin(-0.5x_{i+1} + x_i^2),$$

$$x_0 = [1, 1, \dots, 1].$$

45. **BIGGSB1 function (CUTE):**

$$f(x) = (x_1 - 1)^2 + \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 + (1 - x_n)^2,$$

$$x_0 = [0, 0, \dots].$$

46. **Generalized Quartic function:**

$$f(x) = \sum_{i=1}^{n-1} x_i^2 + (x_{i+1} + x_i^2)^2,$$

$$x_0 = [1, 1, \dots].$$

47. **Full Hessian FH3 function:**

$$f(x) = \left(\sum_{i=1}^{n} x_i\right)^2 + \sum_{i=1}^{n} (x_i \exp(x_i) - 2x_i - x_i^2),$$

$$x_0 = [1, 1, \ldots, 1].$$

5.3 Code Listings

The following code is available in the source code repository https://github.com/danphenderson/numerical-optimization/blob/main/ju

Code 1: Algorithm 16.5

```julia
using CSV, DataFrames, OptimizationProblems, ADNLPModels, NLPModels
using Random, Arpack, Optim, LineSearches
using LinearAlgebra, Statistics, Distributions

Random.seed!(1234)


function get_test_set(n::Int=40)
    """
    List of unconstrained scalable ADNLPModels within OptimizationProblems.jl
    """
    # meta = OptimizationProblems.meta
    # names_pb_vars = meta[
    # (meta.variable_nvar .== true) .& (meta.ncon .== 0) .& (5 .<= meta.nvar .<= 100),
    #     [:nvar, :name]
    # ]
    # test_set_generator = (
    #     eval(Meta.parse("ADNLPProblems.$(pb[:name])(n=$n)")) for pb in eachrow(names_pb_vars)
    # )
    # return [p for p in test_set_generator]
    return [
        "genrose",
        "arglina",
        "freuroth",
        "eg2",
        "cosine",
        "arglinb",
        "arglinc",
        "argtrig",
        "arwhead",
        "bdqrtic",
        "brownal",
        "broyden3d",
        "chnrosnb_mod",
        "cragglvy",
        "cragglvy2",
        "curly10",
        "curly10",
        "curly20",
        "curly30",
        "dixon3dq",
        "dqdrtic",
        "dqrtic",
        "edensch",
        "engval1",
        "errinros_mod",
        "extrosnb",
        "fletcbv2",
        "fletcbv3_mod",
        "fletchcr",
        "genhumps",
        "genrose_nash",
        "indef_mod",
        "integreq",
        "liarwhd",
        "morebv",
        "noncvxu2",
        "noncvxun",
        "nondia",
        "nondquar",
        "penalty1",
        "penalty2",
        "penalty3",
        "power",
        "quartc",
        "sbrybnd",
        "schmvett",
        "scosine",
        "sinquad",
        "tointgss",
        "tquartic",
        "tridia",
        "vardim"];
end

function get_problem(name::String, n::Int=40)
    """
    Get a problem by name
    """
    return eval(Meta.parse("ADNLPProblems.$(name)(n=$n)"))
end

function get_optim_options()
    """
    Using really stict conditions in low dimensions.

    Tacking earlier in routine may be obpuscated by extra
    iterations to obtain terminal convergence.
    """
    return Optim.Options(
        iterations = 10000000,
        g_abstol = eps(),
        store_trace = false, # Trace has a lot of useful stuff...
        show_trace = false,
        extended_trace = false,
    )
end

function build_sample_box(problem::ADNLPModel)
    """
    Box centered around the minimizer
    """
    x0 = problem.meta.x0
    scale_vector = 2 .* abs.(x0)
    scale_vector[scale_vector .<= 1.0] .= 1.0
    return scale_vector
end

function pull_sample(problem, box::Vector)
    """
    Pulls a sample uniformly from the box box surrounding x0
    """
    return rand.(Uniform.(-box, box))
end

```

```julia
function bfgs_linesearch()
    """
    Define the algorithm for the optimization
    """
    return BFGS(;
        alphaguess = LineSearches.InitialStatic(),
        linesearch = LineSearches.HagerZhang(),
        initial_invH = x -> Matrix{eltype(x)}(I, length(x), length(x)),
        manifold = Flat(),
    )
end

function gradiant_descent_linesearch()
    """
    Defines the Quasi-Newton algorithm for the optimization.

    P is our H. Currently P = \nabla^2 f(x) = I and we fallback
    to gradient descent.

    TODO: Accept P as an argument.
    """
    GradientDescent(;
        alphaguess = LineSearches.InitialHagerZhang(),
        linesearch = LineSearches.HagerZhang(),
        P = nothing,
        precondprep = (P, x) -> nothing
    )
end

function newton_trust_region()
    """
    Defines the Newton Trust Region algorithm for the optimization.
    """
    return NewtonTrustRegion(; initial_delta = 1.0,
        delta_hat = 100.0,
        eta = 0.1,
        rho_lower = 0.25,
        rho_upper = 0.75)
end

function eigs_hess(problem::ADNLPModel, x_cp::Vector)
    H = hess(problem, x_cp)
    λ, _ = eigs(H, nev=problem.meta.nvar - 1, maxiter=10000, which=:LM)
    λmin, _ = eigs(H, nev=1, maxiter=10000, which=:SR)
    push!(λ, pop!(λmin))
    return λ
end

function run_sample(problem::ADNLPModel, sample::Vector)
    """
    Run the optimization algorithm on the problem
    """
    # Define objective and in-place gradient aligning with optim's interface.
    x0 = sample
    f(x) = obj(problem, x)
    g!(G, x) = grad!(problem, x, G)
    h!(H, x) = hess!(problem, x, H)

    # Run the optimization
    res = Optim.optimize(
        f,
        g!,
        x0,
        bfgs_linesearch(),
        get_optim_options()
    )

    # Reset problem counters.
    NLPModels.reset!(problem)
    return res
end

function run(problem)
    box = build_sample_box(problem)
    df = DataFrame(
        "initial_objective" => Vector{Float64}(),
        "final_objective" => Vector{Float64}(),
        "g_residual" => Vector{Float64}(),
        "is_saddle" => Vector{Bool}(),
        "pos_curvature_directions" => Vector{Int}(),
        "neg_curvature_directions" => Vector{Int}(),
        "zero_curvature_directions" => Vector{Int}(),
        "max_lambda" => Vector{Float64}(),
        "min_lambda" => Vector{Float64}(),
        "median_lambda" => Vector{Float64}(),
        "iterations" => Vector{Int}(),
        "critical_point" => Vector{Vector{Float64}}(),
    )
    for _ in 1:1000
        sample = pull_sample(problem, box)
        fx0 = obj(problem, sample)
        try
            res = run_sample(problem, sample)
            if !res.f_converged
                continue
            end
            λ = eigs_hess(problem, res.minimizer)
            push!(df, (
                initial_objective = fx0,
                final_objective = res.minimum,
                g_residual = res.g_residual,
                is_saddle = λ[end] < 0,
                pos_curvature_directions = sum(λ .> 0),
                neg_curvature_directions = sum(λ .< 0),
                zero_curvature_directions = sum(abs.(λ) .< 1e-12),
                max_lambda = maximum(λ),
                min_lambda = minimum(λ),
                median_lambda = median(λ),
                iterations = res.iterations,
                critical_point = res.minimizer,
            ))
        catch
            continue
        end
    end
    return df
end

function unique_critical_points(df::DataFrame)
    """
    Compares the critical points locations to determine
    the number of unique critical points.
    """
    critical_points = df.critical_point
    critical_points = [round.(x, digits=4, base=2) for x in critical_points] # HACK d
    return length(unique(critical_points))
end

function run_all()
    test_set = get_test_set()
    mkpath("public/saddles/dim-40")
    for pb in test_set
        problem = get_problem(pb, 40)
        df = run(problem)
        CSV.write("public/saddles/dim-40/$(pb).csv", df)
        println("$(pb) has $(unique_critical_points(df)) unique critical points.")
        println("   $(pb) total saddles $(sum(df.is_saddle))")
    end
end
```