CS1.404 (Spring 2025)
Optimization Methods

Assignment 1

Deadline: 11:55 PM, March 21st, 2025

Instructions

1. Attempting **all** questions is mandatory.
2. You are expected to solve all the questions using the **Python programming language**.
3. Use of any in-built libraries that directly solve the problem is **not allowed**.
4. Submission Format: Check assignment description or announcement post for details.
5. **Plagiarism is strictly prohibited.** All code will be checked for plagiarism. Any copied code or suspicious similarity will result in an **F grade**.
6. If any two students submit the exact same code, **both** get an **F grade**.

---

1 Functions

1.1 Trid Function

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

1.2 Three Hump Camel

$$f(x) = 2x_1^2 \; - \; 1.05\, x_1^4 \; + \; \frac{x_1^6}{6} \; + \; x_1 x_2 \; + \; x_2^2$$

1.3 Styblinski-Tang Function

$$f(x) = \frac{1}{2}\sum_{i=1}^{d}\left(x_i^4 - 16\, x_i^2 + 5\, x_i\right)$$

1.4 Rosenbrock Function

$$f(x) = \sum_{i=1}^{d-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$$

1.5 Root of Square Function

$$f(x) = \sqrt{1 + x_1^2} \; + \; \sqrt{1 + x_2^2}$$

---

2 Steepest Descent

Implement the Steepest Descent algorithm using inexact line search. For both the algorithms, use the following stopping condition: Terminate the algorithm when the magnitude of gradient is less than 10^{-6} or after 10^4 iterations.

2.1 Backtracking with Armijo Condition

Initialization: Set $\alpha_0 = 10.0, \rho = 0.75, c = 0.001, k = 0, \epsilon = 10^{-6}$
while $k \leq 10^4$ and $\|\nabla f(\mathbf{x}_k)\| > \epsilon$ **do**
  Set $\alpha \leftarrow \alpha_0$
  $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$
  while $f(\mathbf{x}_k + \alpha \mathbf{d}_k) > f(\mathbf{x}_k) + c\alpha \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$ **do**
    $\alpha \leftarrow \rho\alpha$
  end while
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{d}_k$
  $k \leftarrow k + 1$
end while

Figure 1: Armijo Condition

2.2 Backtracking with Armijo-Goldstein Condition

The condition is as follows.

$$f(\mathbf{x}_k) + (1 - c_1)\alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k \leq f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$

for some constant $c_1 \in (0, 1/2)$.

Figure 2: Armijo-Goldstein Condition

2.3 Bisection Method with Wolfe Condition

Initialization: Set $c_1 = 0.001$, $c_2 = 0.1$, $\alpha_0 = 0$, $t = 1$ and $\beta_0 = 10^6, k = 0, \epsilon = 10^{-6}$

while $k \leq 10^4$ and $\|\nabla f(\mathbf{x}_k)\| > \epsilon$ **do**
  $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$
  Set $\alpha \leftarrow \alpha_0$ and $\beta \leftarrow \beta_0$
  while True **do**
    if $f(\mathbf{x} + t\mathbf{d}_k) > f(\mathbf{x}) + c_1 t \nabla f(\mathbf{x})^T \mathbf{d}_k$ **then**
      set $\beta = t$ and reset $t = \frac{1}{2}(\alpha + \beta)$
    else if $\nabla f(\mathbf{x} + t\mathbf{d}_k)^T \mathbf{d}_k < c_2 \nabla f(\mathbf{x})^T \mathbf{d}_k$ **then**
      set $\alpha = t$ and reset $t = \frac{1}{2}(\alpha + \beta)$
    else
      STOP
    end if
  end while
  $\mathbf{x}_{k+1} = \mathbf{x}_k + t\mathbf{d}_k$
  $k \leftarrow k + 1$
end while

Figure 3: Wolfe Condition

---

3 Newton's Method

Implement the following variants of Newton's Method. Use the following stop- ping condition- Terminate the algorithm when the magnitude of gradient is less than 10^{-6} or after 10^4 iterations.

3.1 Pure Newton's Method

Pure Newton's Method

Input: $\varepsilon > 0$ - tolerance parameter.

Initialization: Pick $\mathbf{x}_0 \in \mathbb{R}^n$ arbitrarily.
General step: For any $k = 0, 1, 2, \ldots$ execute the following steps:

(a) Compute the Newton direction $\mathbf{d}_k$, which is the solution to the linear system $\nabla^2 f(\mathbf{x}_k)\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$.

(b) Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$.

(c) If $||\nabla f(\mathbf{x}_{k+1})|| \leq \varepsilon$, then STOP, and $\mathbf{x}_{k+1}$ is the output.

Figure 4: Pure Newton

3.2 Damped Newton's Method

Set $\alpha = 0.001$ and $\beta = 0.75$.

Damped Newton's Method

Input: $\alpha, \beta \in (0, 1)$ - parameters for the backtracking procedure.
$\varepsilon > 0$ - tolerance parameter.

Initialization: Pick $\mathbf{x}_0 \in \mathbb{R}^n$ arbitrarily.
General step: For any $k = 0, 1, 2, \ldots$ execute the following steps:

(a) Compute the Newton direction $\mathbf{d}_k$, which is the solution to the linear system $\nabla^2 f(\mathbf{x}_k)\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$.

(b) Set $t_k = 1$. While

$$f(\mathbf{x}_k) - f(\mathbf{x}_k + t_k \mathbf{d}_k) < -\alpha t_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$

set $t_k := \beta t_k$.

(c) $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \mathbf{d}_k$.

(c) If $||\nabla f(\mathbf{x}_{k+1})|| \leq \varepsilon$, then STOP, and $\mathbf{x}_{k+1}$ is the output.

Figure 5: Damped Newton

3.3 Levenberg-Marquardt Modification

Initialization: $x_0 \in \mathbb{R}^n, k = 0, \epsilon = 10^{-6}$
while $k \leq 10^4$ and $\|\nabla f(x_k)\| > \epsilon$ **do**
  λ_{min} = Smallest eigen value of $\nabla^2 f(x_k)$
  if $\lambda_{min} \leq 0$ **then**
    $\mu_k = -\lambda_{min} + 0.1$
    $\mathbf{d}_k = -(\nabla^2 f(x_k) + \mu_k \mathbf{I})^{-1} \nabla f(x_k)$
  else
    $\mathbf{d}_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$
  end if
  $\mathbf{x_{k+1}} = \mathbf{x_k} + \mathbf{d}_k$
  $k \leftarrow k + 1$
end while

Figure 6: Levenberg-Marquardt

3.4 Combining Damped Newton's Method with Levenberg-Marquardt

Initialization: $x_0 \in \mathbb{R}^n, k = 0, \epsilon = 10^{-6}$
while $k \leq 10^4$ and $\|\nabla f(x_k)\| > \epsilon$ **do**
  λ_{min} = Smallest eigen value of $\nabla^2 f(x_k)$
  if $\lambda_{min} \leq 0$ **then**
    $\mu_k = -\lambda_{min} + 0.1$
    $\mathbf{d}_k = -(\nabla^2 f(x_k) + \mu_k \mathbf{I})^{-1} \nabla f(x_k)$
  else
    $\mathbf{d}_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$
  end if
  Calculate α_k using Backtracking with armijo condition with descent direction as $\mathbf{d}_k$
  $\mathbf{x_{k+1}} = \mathbf{x_k} + \alpha_k \mathbf{d}_k$
  $k \leftarrow k + 1$
end while

Figure 7: Combined

---

4 Submission Instructions

4.1 Allowed Packages

1. Python 3.11
2. NumPy
3. Matplotlib
4. PrettyTables

4.2 Boiler Plate

You have been provided three files:

- `algos.py`: Where you must implement all the algorithms.
- `functions.py`: Contains all the functions used for testing.
- `main.py`: The file to be executed to test your code.

You are not allowed to create additional files nor modify `main.py` and `functions.py` for the final submission. In `algos.py` you may create new helper functions if needed.

4.3 Report

You are required to submit a **Report (as a PDF)** that includes:

1. Derivations of the Jacobians and Hessians for all the functions.
2. Using those Jacobians and Hessians, manually compute the minima for all functions *except* Rosenbrock.
3. Mention which algorithms failed to converge (if any) and under what circumstances.
4. Plot $f(x)$ vs. iterations and $\|\nabla f(x)\|$ vs. iterations.
5. Make a contour plot with arrows indicating the update directions for all 2D functions.

4.4 Submission Format

- Create a folder named with **your Roll Number**, containing:
  - `algos.py`
  - `functions.py`
  - `main.py`
  - `Report.pdf`
- Compress the folder into `RollNo.zip`. For example: `2021112011.zip`.
