CS1.404 (Spring 2024)
Optimization Methods
Deadline: 11.55 PM, March 9th, 2024

Instructions

1. Attempting all questions is mandatory.
2. You are expected to solve all the questions using python programming language.
3. Use of any in-built libraries to solve the problem directly is not allowed.
4. Submission Format: Check assignment description or announcement post for more details.
5. Plagiarism is a strict No. We will pass all codes through the plagiarism checking tool to verify if the code is copied from somewhere. In that case, you get F grade in the course.
6. If any two students codes are found exactly same (if they copy from each other), both will get F grade.

1 Functions

1.1 Trid Function

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

1.2 Three Hump Camel

$$f(\mathbf{x}) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$$

1.3 Styblinski-Tang Function

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

1.4 Rosenbrock Function

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

1.5 Root of Square Function

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

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

Algorithm 1 Backtracking

```
**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**
```

2.2 Bisection method with *Wolfe* condition

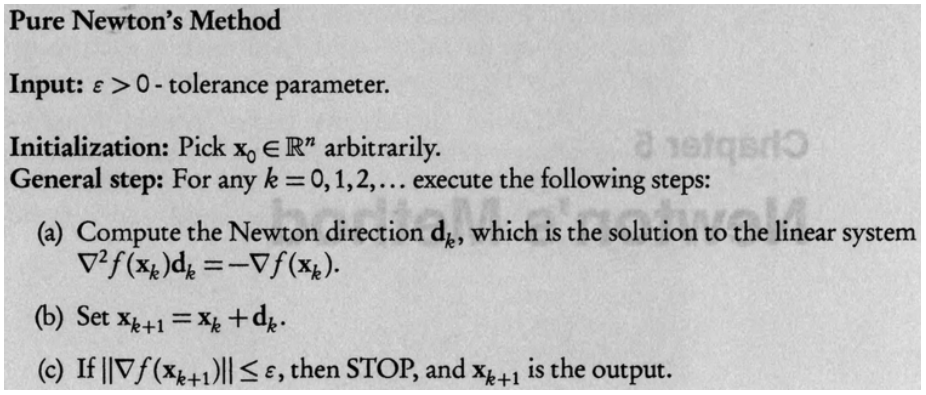
Algorithm 2 Bisection Method

```
**Initialization:** Set $c_1 = 0.001$, $c_2 = 0.1$, $\alpha = 0$, $t = 1$ and $\beta = 10^6, k = 0, \epsilon = 10^{-6}$
**while** $k \leq 10^4$ and $\|\nabla f(\mathbf{x}_k)\| > \epsilon$ **do**
  $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$
  **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**
```

3 Newton's Method

Implement the following variants of Newton's Method. Use the following stopping 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



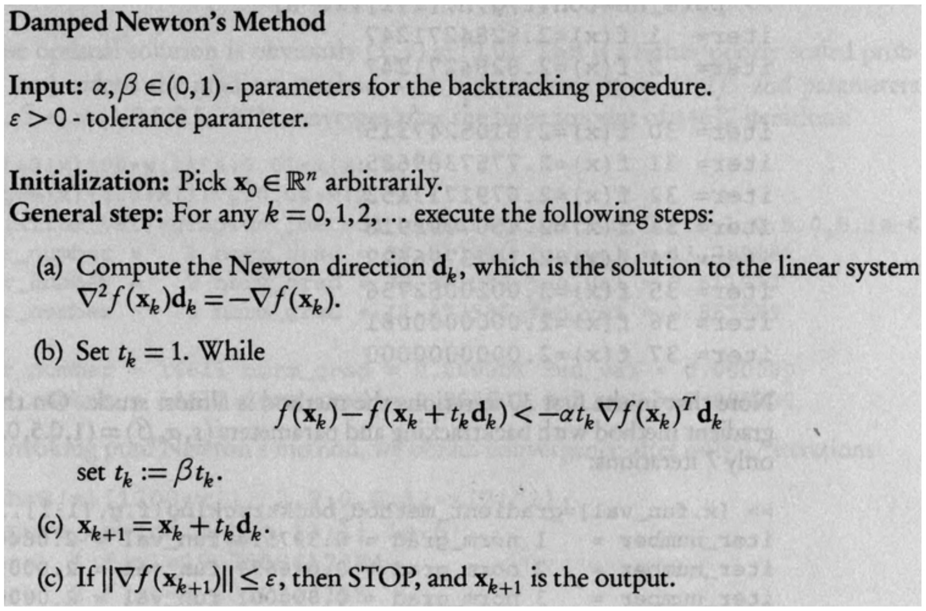
```
Pure Newton's Method

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.
```

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.
```

3.3 Levenberg-Marquardt Modification

Algorithm 3 Levenberg-Marquardt modification of Newton's Method

```
**Initialization:** $x_0 \in \mathbb{R}^n, k = 0, \epsilon = 10^{-6}$
**while** $k \leq 10^4$ and $\|\nabla f(x_k)\| > \epsilon$ **do**
  $\lambda_{min}$ = 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**
```

3.4 Combining Damped Newton's Method with Levenberg-Marquardt modification

Algorithm 4 Damped + 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**
  $\lambda_{min}$ = 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 $\alpha_k$ 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**
```

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 with three files

1. *algos.py*: Where you have to implement the algorithms.
2. *functions.py*: Which contains all the functions used for testing.
3. *main.py*: This is the file to be executed to test your code.

You are not allowed to create anymore files, nor are you allowed to modify *main.py* and *functions.py*. You can make changes to these files for your own test purposes but the final evaluations will use the original files for testing. In *algos.py* you are allowed to create your own new functions.

4.3 Report

You are required to submit a report (as a pdf) containing the following-

1. Derive the Jacobians and Hessians for all the functions
2. Using the Jacobians and Hessians, calculate the minimas for all functions except Rosenbrock.
3. State which algorithms failed to converge and under which circumstances.
4. Plot $f(x)$ vs *iterations* and $|f'(x)|$ vs *iterations*.
5. Make a contour plot with arrows indicating the direction of updates for all 2-d functions.

4.4 Submission Format

Create a folder with your Roll No. as its name containing the four files *(algos.py, functions.py, main.py, Report.pdf)*. Zip it and name it Roll_No.zip
Example- *2019111013.zip*