Instructions: Try to solve all problems on your own. If you have difficulties, ask the instructor or TAs. Questions marked as **[R]** should be answered in your report. Please follow the instructions for submitting your assignments and reports circulated earlier.

Exercise 1: Newton's Method for Unconstrained Minimization
Suppose we want to find stationary points of a function $f : \mathbb{R}^n \to \mathbb{R}$, starting from a given point $x^0 \in \mathbb{R}^n$, using Newton's method (covered in IE 601):

$k \leftarrow 0$;
while *not converged* **do**
  $d^k \leftarrow -\left(\nabla^2 f(x^k)\right)^{-1} \nabla f(x^k)$;
  $\alpha_k \leftarrow \text{argmin}_{\alpha \geq 0} f(x^k + \alpha d^k)$;
  $x^{k+1} \leftarrow x^k + \alpha_k d^k$;
  $k \leftarrow k + 1$;
end

1. **[R]** What convergence criteria would you use for this algorithm?
2. Implement the above algorithm in a Python function (or a language of your choice). Your implementation should be independent of the function used. You may assume that the function you want to minimize is available as another Python function (evalf) that you can call from your program. Also assume that the function to evaluate gradient (gradf) and hessian (hessf) are available as well. **For solving the inner minimization problem to compute α_k, use your own implementation from the previous labs.**
3. Test your algorithm for the function $f : \mathbb{R}^2 \to \mathbb{R}$, defined as $f(x) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$, starting from point $x^0 = (-1, -1)$.
4. **[R]** What are the stationary points of f above?
5. **[R]** How many iterations does your algorithm take to get close (according to your convergence criteria) to a stationary point?
6. **[R]** Plot the sequence of points $x^0, x^1, \ldots$ generated by your algorithm and comment on the plot.
7. **[R]** Plot the distance between the consecutive values of x as a function of number of iterations and comment on the plot.
8. **[R]** Plot the function value against the number of iterations and comment on the plot.

Exercise 2: BFGS
Recall the BFGS algorithm

$k \leftarrow 0$;
while *not converged* **do**
  $d^k \leftarrow -(B^k)^{-1} \nabla f(x^k)$;
  $\alpha_k \leftarrow \text{argmin}_{\alpha \geq 0} f(x^k + \alpha d^k)$;
  $x^{k+1} \leftarrow x^k + \alpha_k d^k$;
  $s^k = x^{k+1} - x^k$;
  $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$;
  $B^{k+1} \leftarrow B^k - \frac{B^k s^k (s^k)^T B^k}{(s^k)^T B^k s^k} + \frac{(y^k)^T y^k}{(y^k)^T s^k}$;
  $k \leftarrow k + 1$;
end

1. **[R]** What is the initial choice of B?
2. Implement this algorithm and verify whether it converges for the above problem starting from the same point.
3. **[R]** Plot the function value against the number of iterations and comment on the plot. Compare it against Newton's method.
4. **[R]** Is B^k positive definite in all iterations? Check by calcuating the eigenvalues of B^k. You may use the Numpy package in Python for it.
5. Find the minimum value of
$$100 - \sum_{i=1}^{199} \log(1 + [ix_i - (i+1)x_{i+1} - i]^2),$$
starting from the point $(50, 50, 50, \ldots, 50)$. Compare the performance of your two algorithms for this problem.

Exercise 3: Convergence Let $\{x_k\}$ be a sequence in $\mathbb{R}^n$ that converges to x^*. The convergence is said to be *Q-Linear* if there is a constant $r \in (0,1)$ such that

$$\frac{||x_{k+1} - x^*||}{||x_k - x^*||} \leq r, \text{ for all } k \text{ sufficiently large,}$$

Q-superlinear if

$$\lim_{k\to\infty} \frac{||x_{k+1} - x^*||}{||x_k - x^*||} = 0,$$

and *Q-quadratically* if

$$\frac{||x_{k+1} - x^*||}{||x_k - x^*||^2} \leq M, \text{ for all } k \text{ sufficiently large}$$

For each of following three sequences check empirically how fast they converge: (a)$(1 + 0.5)^{2^k}$, (b)$1 + (0.5)^k$, (c) $1 + k^{-k}$. Use plots carefully to depict the rates clearly.

Plot the convergence rates of three methods: steepest descent, Newton's method and BFGS on problem described in Q 2.5 above. Comment on the convergence rates of the three algorithms on this problem.