# EECE 5550 HW3 Problem 1: Optimization Methods

**To begin, make a copy of this notebook so you can save changes**.

After completing these problems, please save this notebook as a pdf and combine it with your solutions to the written problems so that you can upload a single pdf to Gradescope.

## These first few cells just implement some methods you may find useful

```
import numpy as np
import matplotlib.pyplot as plt
from typing import Callable
```

```
def draw_plot(xs: np.ndarray, f: Callable, k: int = 1) -> None:

    # Draw 2 plots:
    # On the left subplot, show:
    # - the function f(x) as contours
    # - the values of x as we search for the minimizer
    # On the right subplot, show:
    # - value of f(x) vs. current xi at each iteration of the optimization

    plt.subplots(1, 2)

    plt.subplot(1, 2, 2)
    z = f(xs)
    plt.plot(np.arange(len(z)), z)
    plt.xlabel('Iteration')
    plt.ylabel('f(x)')
    plt.gca().set_yscale('log')

    plt.subplot(1, 2, 1)

    # Optimization
    plt.plot(xs[:, 0], xs[:, 1], '-x')

    x_low = -2
    x_high = 2
    y_low = -2
    y_high = 2

    # Contour Plot
    num_xs = 20
    num_ys = 20
    x_vals = np.linspace(x_low, x_high, num_xs)
    y_vals = np.linspace(y_low, y_high, num_ys)
    X, Y = np.meshgrid(x_vals, y_vals)
    xy_pairs = np.vstack([X.ravel(), Y.ravel()]).T
    z = f(xy_pairs, k=k)
    Z = z.reshape(num_xs, num_ys)
    plt.gca().contour(X, Y, Z)

    plt.xlim([x_low, x_high])
    plt.ylim([y_low, y_high])
    plt.show()
```

```
def f(x: np.ndarray, k: int = 1) -> float:
    # f(x) = x^2 - xy + ky^2
    if x.ndim == 1:
        return x[0]**2 - x[0]*x[1] + k*x[1]**2
    elif x.ndim == 2:
        return x[:, 0]**2 - x[:, 0]*x[:, 1] + k*x[:, 1]**2
```

## 1.1 [2pts] Implement the gradient and hessian of f(x; k)

```
def gradf(x: np.ndarray, k: int = 1) -> np.ndarray:
    return np.array([2 * x[0] - x[1], -x[0] + 2 * k * x[1]])

def hessf(x: np.ndarray, k: int = 1) -> np.ndarray:
    return np.array([[2, -1], [-1, 2 * k]])
```

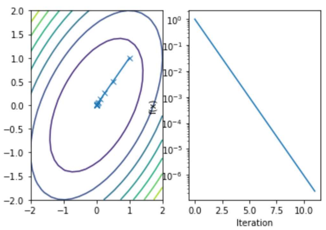
## 1.2 [2pts] Implement Gradient Descent

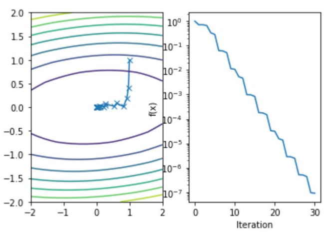
```
def gradient_descent(
        f: Callable,
        gradf: Callable,
        x0: np.ndarray,
        c: float,
        tau: float,
        epsilon: float,
        plot: bool = False,
        k: int = 1,
        ) -> np.ndarray:
    xs = [x0]
    while True:
```

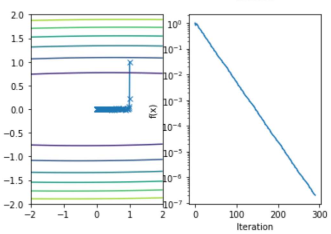
```
        x = xs[-1]
        p = -gradf(x, k)
        if np.linalg.norm(p) < epsilon:
            break
        alpha = 1
        while f(x + alpha*p, k) > f(x, k) - c*alpha*np.linalg.norm(p)**2:
            alpha *= tau
        xs.append(x + alpha*p)
    xs = np.array(xs)
    if plot:
        draw_plot(xs, f, k=k)
    return xs
```

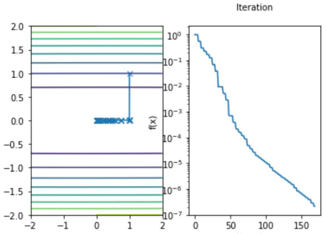
## 1.2 [1pt] Run your gradient descent for various values of $\kappa$

```
x0 = np.array([1., 1.])
c = 0.5
tau = 0.5
epsilon = 0.001
xs = gradient_descent(f, gradf, x0, c, tau, epsilon, k=1, plot=True)
xs = gradient_descent(f, gradf, x0, c, tau, epsilon, k=10, plot=True)
xs = gradient_descent(f, gradf, x0, c, tau, epsilon, k=100, plot=True)
xs = gradient_descent(f, gradf, x0, c, tau, epsilon, k=1000, plot=True)
```




For full credit on this problem, add some text description of what you observe about these results and why that is occurring (replace this text cell with your own thoughts).

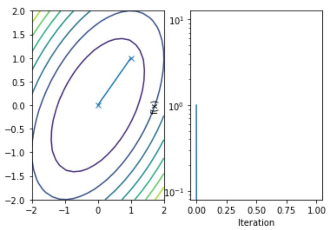
## 1.4 [2pts] Implement Newton's Method

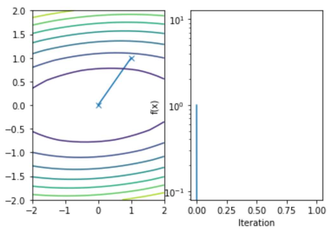
```
def newton(
        f: Callable,
        gradf: Callable,
        hessf: Callable,
        x0: np.ndarray,
        epsilon: float,
```

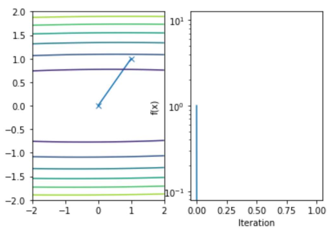
```
        plot: bool = False,
        k: int = 1
) -> np.ndarray:
    xs = [x0]
    while True:
        x = xs[-1]
        grad = gradf(x, k=k)
        hess = hessf(x, k=k)
        if np.linalg.norm(grad) < epsilon:
            break
        xs.append(x - np.linalg.inv(hess) @ grad)
    xs = np.array(xs)
    if plot:
        draw_plot(xs, f, k=k)
    return xs
```

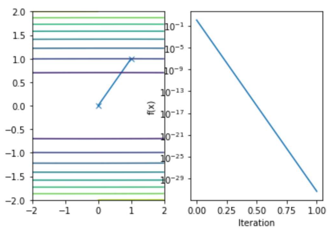
## 1.5 [1pt] Run your Newton's method for various values of $\kappa$

```
x0 = np.array([1., 1.])
epsilon = 0.001
xs = newton(f, gradf, hessf, x0, epsilon, k=1, plot=True)
xs = newton(f, gradf, hessf, x0, epsilon, k=10, plot=True)
xs = newton(f, gradf, hessf, x0, epsilon, k=100, plot=True)
xs = newton(f, gradf, hessf, x0, epsilon, k=1000, plot=True)
```




For full credit on this problem, add some text description of what you observe about these results and why that is occurring (replace this text cell with your own thoughts).