Answers: Exercise Sheet No. 9

1 Concepts and definitions

1. (a) If a function is convex than local minimizers are global minimizers.
   (b) The "standard" conditions of optimality require the first and second derivatives of a function to exist – if they don't then different (more complex) methods have to be used.
2. (a) Convex, but non-smooth (derivative does not exist at 0.)
   (b) Non-convex and smooth.
3. For minimization, A should be positive semi-definite.
4. n and $n \times n$
5. $\nabla f(x^*) = 0$ and the Hessian will be positive semi-definite.
6. No. We need knowledge of the Hessian matrix.

2 Steepest descent and Newton's method

1. It is taking a step 'down' by following the negative gradient. The parameter γ determines the length of this step.

2. Steepest descent only requires the calculation of the first derivatives while Newton's method requires the calculation of second derivatives. Thus per iteration Newton's method is more costly. However, Newton's method can have a faster rate of convergence than steepest descent, thus the number of steps (iterations) it needs to converge can be lower.

3. Steepest descent may step over this point. If it lands on this point it will not be able to leave as the gradient is zero.

4. No, Newton's method is not guaranteed to be positive definite at every step, however convex problems will always meet this condition. For non-convex problems one may need to regularise the Hessian to ensure that this condition is always met.

3 Implementation

```python
import autograd.numpy as np
from autograd import grad, hessian

# Define the function to minimize
def f(x):
    return (x[0]**2 + x[1] - 11)**2 + (x[0] + x[1]**2 - 7)**2

# Define the gradient and Hessian using Autograd
grad = grad(f)
hes = hessian(f)

def newton_method(f, x0, grad, hess, tol=1e-6, max_iter=1000):
    grad_f = grad(x)
    hess_f = hes(x)
    while iter_count < max_iter:
        # Compute the search direction using Newton's method
        p = np.linalg.solve(hess_f, -grad_f)

        # Update the current point
        x_new = x + p

        # Check if the convergence criterion is met
        if np.linalg.norm(x_new - x) < tol:
            break

        # Update the current point and iteration counter
        x = x_new
        iter_count += 1


    return x

sol1 = newton_method(f, np.array([-0.2, -1.1]),grad, hess)
sol2 = newton_method(f, np.array([4., -2.]),grad, hess)


def check_hess(sol):
    eig_val =  np.linalg.eigvals(hes(sol))
    if np.all(eig_val >= 0):
        print("The Hessian is positive semi-definite")
    if np.all(eig_val <= 0):
        print("The Hessian is negative semi-definite")

check_hess(sol1)
check_hess(sol2)
```

1. A maxima.
2. A minima.