1. Refer to script.

2. Refer to script. No matter which initial value x is set to, the script always converges to minimum x = -4 in one iteration. Because the first derivative of the function is a linear equation, any line tangent to the first derivative of the function is equal to the first derivative of the function, instantly finding the root in the first iteration for any initial x value.

3. Refer to script. Using the Newton-Raphson method results in an infinite loop. This is because it's solving for the root of the function (not the first derivative of the function), which makes it impossible to clear the stopping criterion, since it converges at x = 1.2915, a value found before x = -4. If the stopping criterion is changed to stop at |f(x)| < 10^-6, it will converge at x = 1.2915 in 4 iterations.

4. Refer to script. For function 1 (f(x) = 10*x1^2 + 10*x2^2) and starting points [10,2] I got 14 iterations and [x1,x2] = 3.725290298461914e-08;7.450580596923828e-09].

   For function 1 and starting points [10,0.1] I got 14 iterations and [x1,x2] = [3.725290298461914e-08;3.725290298461913e-10].

   For function 2 (f(x) = x1^2 + 10*x2^2) and starting points [10,2] I got 126 iterations and [x1,x2] = [4.931904753940478e-07;2.763573937630222e-76]

   For function 2 and starting points [10,0.1] I got 15 iterations and [x1,x2] = [0;-4.526227712631225e-08]

   As shown for function 1, changing the initial value affected that positions final value, but not the other. The large iteration for function 2 and starting points [10,2] could be attributed to the significantly smaller partial derivative of x1, which leads to a smaller gradient. This causes the algorithm to take a longer time to find the minimums. Function 2 was able to find the exact minimum at zero with initial values [10,0.1] because the gradient is larger due to the larger initial interval.

5. Refer to script. The only difference from the previous question is the removal of reducing t and changing how delta is evaluated. Instead of reducing t, the pure Newton method uses t=1. The changes in delta result in computing the inverse of the hessian and multipling it with the gradient. The hessian is defined as the partial derivatives of the gradient, i.e., [2*x1 + 2*x2] becomes [2 2]. Because these values do not change per iteration, the gradient is static, resulting in only a single iteration; finding the minimum instantly for all functions and initial values.

6. Refer to function.

7. Refer to script. Using the script, I found:
   c1 = 610/53, c2 = 610/53, c3 = 1010/53, c4 = 9910/583, c5 = 610/53

8. Refer to function. Using the function, I found:

c1 = 11.5094, c2 = 11.5094, c3 = 19.0566, c4 = 16.9983, c5 = 11.5094
These values are equivalent to what I found in problem 7; the function works.

9. Refer to function.