The purpose of this homework is to get you thinking about the numerical aspects of maximum likelihood estimation. A side objective is to provide a programming task that is a bit more detail-oriented, and hence somewhat more challenging. We’ll be manually implementing the Newton-Raphson algorithm for a conditional Poisson regression, and then comparing our results to those obtained using the routines in the optimization toolbox.

Following up on the lab example, consider a Poisson regression with a conditional mean in which the expected value of the distribution is parameterized with exogenous variables:

\[ Y_i \sim \text{POISSON}(\lambda_i) \]
\[ \lambda_i = \exp(\beta'X_i), \]

where \( X_i \) is a \( k \times 1 \) vector of explanatory variables and \( \beta \) is a \( k \times 1 \) vector of unknown parameters to be estimated. Recall that the probability mass function for the Poisson is

\[ pr(Y_i) = \frac{\lambda_i^y \exp(-\lambda_i)}{Y_i!}, \quad E(Y_i) = \lambda_i, \]

and the likelihood for a sample \((y_i, X_i), i=1,\ldots,N\) is

\[
LL(\beta) = \sum_{i=1}^{N} y_i \ln[\exp(\beta'X_i)] - \exp(\beta'X_i) - \sum_{i=1}^{N} y_i \beta'X_i - \exp(\beta'X_i).
\]

Consider a Monte Carlo experiment using this data generating process. In particular, write code to generate a dataset with the following characteristics:

- Number of observations is 200
- \( \beta=[1.5, -0.5]' \)
- \( X_i \) contains a constant and an exogenous variable generated from a standard uniform distribution.
- \( y_i \) is drawn from a Poisson random number generator.

**Hint:** the command we used in the lab to generate Poisson realizations with a constant mean can also be used with an \( N \times 1 \) vector of conditional means.

**Part A: Implement NR Manually**

Using the generated data ‘estimate’ the unknown parameters \( \beta \) using maximum likelihood. In particular, implement the Newton-Raphson algorithm to find values of the parameters that maximize \( LL(\beta) \). Once the parameter values are found calculate and report the typical maximum likelihood output, including parameter estimates, parameter standard errors, and asymptotic t-statistics.
Hints:
Recall that NR is an iterative method based on a gradual updating of the parameter vector according to the formula

$$\beta_{t+1} = \beta_t + (-H_t^{-1})g_t,$$

where $H_t$ is the Hessian evaluated at $\beta_t$ and $g_t$ is the gradient at $\beta_t$. This suggests you will need the following in your program:

- Starting values for $\beta$.
- A loop in which the gradient and Hessian are calculated for the current value of $\beta$, from which the new value of $\beta$ is calculated.
- A means of monitoring progress
- A stopping criteria

The stopping criteria and looping structure are clearly related: you want to exit the loop, and consider the optimum found, once some threshold has been reached. Here is one example (not necessarily the best) of how to do this:

```matlab
b = [0.001, 0.001]; %starting values
check = 0;
tol0 = 0.000001;
while check == 0;
    (calculate gradient for b)
    (calculate Hessian for b)
    (update b to bnew from gradient and Hessian)

    tol = abs((bnew-b)./b);
    b = bnew;
    if all(tol<tol0)==1;
        check = 1;
    end;
end;
```

Note that the `all` command checks if all elements of the vector `tol` are less then the criteria, and returns the value of one if this is true.

To monitor progress you might consider including the following in your loop:

- A counter to keep track of the number of iterations
- An `fprintf` command to display the iteration number and current values of the parameters at each pass through the loop.

Once your routine has found the parameters that maximize $LL(\beta)$ you have all the information you need to report the maximum likelihood estimates. In particular:

- The final value of $\beta$ from the loop is the vector of parameter estimates
- The final value of the Hessian from the loop contains all the information needed to construct the standard errors of the parameter estimates.
Another Hint
I suggest you develop this program in stages. First code and debug the gradient, Hessian, and updating routine, making sure it works as you would expect for a single iteration away from the starting values. Once you feel confident with this construct the loop around it.

Part B: Estimate the parameters using the Optimization Toolbox

Using the same generated data ‘estimate’ the unknown parameters $\beta$ using maximum likelihood, but this time use the routines contained in the Optimization Toolbox. In particular, prepare a MATLAB function that contains the log-likelihood, gradient, and Hessian for the model similar to how we did in the lab. Use the `fminunc` command to find the parameters that maximize $LL(\beta)$. Once the parameter values are found calculate and report the typical maximum likelihood output, including parameter estimates, parameter standard errors, and asymptotic t-statistics. Compare your findings to those you derived manually.

A few things to note:

- The code for the likelihood, gradient, and Hessian will be very similar to above – but not exactly the same. Recall that `fminunc` minimizes the function. Watch that you have ‘−’ signs in the correct places!
- Likewise the way that you calculate the standard errors for the estimated parameters will vary slightly, again by a negative sign.
- Since one of the objectives is to compare the manual and canned optimization routines make sure you are using the same generated data for both. That is, generate and save a sample $(y_i, X_i), i=1,\ldots,N$, for use in both routines. Your maximum likelihood estimates will obviously be different if you are using different 'samples' for the different routines.