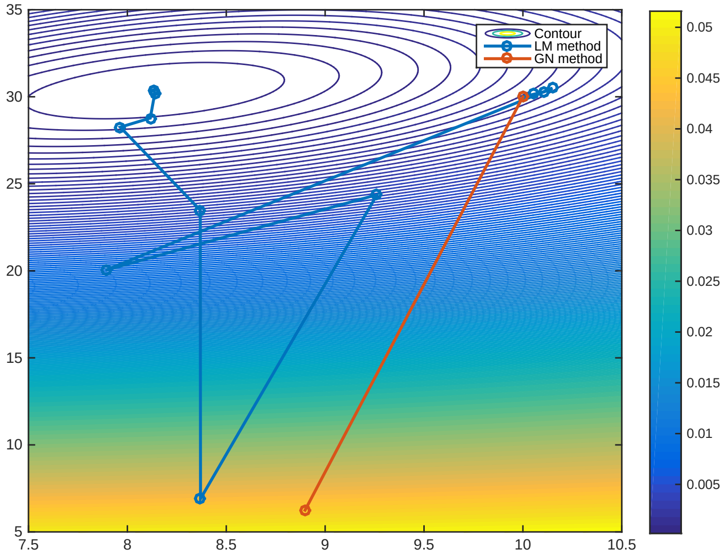
Problem 1 (graded by Kangchen) 30 points

(a) - 10 points

The code is shown below. Note that Gauss-Newton is just a special case of Levenberg-Marquardt when $\lambda = 0$.

```

function [M]=nonlinear_solver(x,y,ui,Minit)

%% for damped least square method

Mr1 = zeros(1000,length(Minit));

M = Minit;

misfit_old = 0;

misfit = 0;

for ii = 1:1:20

misfit = compute_misfit(x,y,M,ui);


[Gamma,Hess] = compute_gradient_approx_hess(x,y,M,misfit);

deltaM= -(Hess+(ii<=3)*10*diag(diag(Hess)))\Gamma;

Mr1(ii,:)=M;

M = M + deltaM;


if (norm(misfit - misfit_old)<1e-7)
    break;
end

misfit_old = misfit;

end

n1=ii;
%% for undamped least square method

Mr2 = zeros(1000,length(Minit));

M = Minit;

misfit_old = 0;

misfit = 0;

for ii = 1:1:2
 %% since LS method do not converge, we plot only the first 2 points.

misfit = compute_misfit(x,y,M,ui);


[Gamma,Hess] = compute_gradient_approx_hess(x,y,M,misfit);

deltaM= -(Hess)\Gamma;

Mr2(ii,:)=M;

M = M + deltaM;


if (norm(misfit - misfit_old)<1e-7)
    break;
end

misfit_old = misfit;

end
n2=ii;

disp(['Number of iterations:',num2str(ii)]);
%% plot the contour and convergence path
[Xs,P]=meshgrid(7.5:0.01:10.5,5:0.1:35);
[m,n]=size(Xs);
ERROR = Xs*0;
for ll = 1:1:m
    for kk = 1:1:n
        M0=[Xs(ll,kk),-5.1412,11.5066,P(ll,kk)]';
  %      M0 = [Xs(ll,kk),-10,10,P(ll,kk)];
        Misfit = compute_misfit(x,y,M0,ui);
        ERROR(ll,kk) = Misfit'*Misfit;
    end
end
contour(Xs,P,ERROR,300);
hold on
plot(Mr1(1:n1,1),Mr1(1:n1,4),'-o','linewidth',2);
hold on
plot(Mr2(1:n2,1),Mr2(1:n2,4),'-o','linewidth',2);
colorbar
legend('Contour','LM method','GN method')
print('fig1b.pdf','-dpdf');
end
```



(b)

10 points

Levenberg Marquardt method should converge to the same value as Gauss-Newton method for this problem. The contour of misfit function is ploted assuming y_s, z_s is the same as the best model. Note that

- λ control the length of updating step for LM method. First 3 step ($\lambda = 10$), the updating is smaller compared with Gauss-Newton method.
- The Gauss-Newton method failed to converge because $G^T G$ is not always invertible.

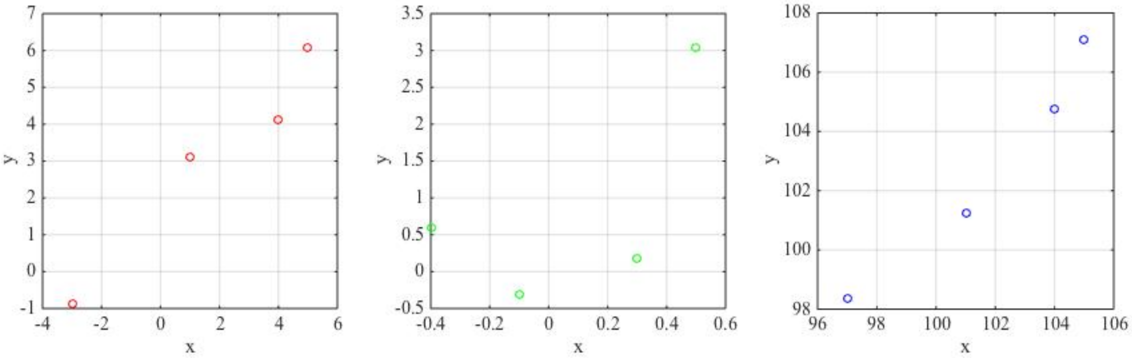
Problem 2 (graded by Kangchen) 30 points

(a) - 5 points

For the case of a linear regression model, we know that our G matrices always have the form:

$$G = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{pmatrix}$$

Our G matrices only know about where we took the measurements (x-values); they don't know anything about the corresponding y-values at these points. Thus there are infinitely many datasets that can correspond to these G matrices. The points DO NOT need to be on a line, or even nearly linear for that matter, for we can still find a "best fit linear model" for any arbitrary dataset. The only requirement is that the points be at x=1, -3, 4, 5 for G_1, x=-0.1, 0.3, -0.4, 0.5 for G_2, and x= 101, 97, 104, 105 for G_3.



(b) - 5 points

We plug in G_1, G_2, and G_3 into Matlab and simply use the SVD function to get the singular value decompostions. The outputs are:

$$U_1 = \begin{pmatrix} 0.1572 & 0.4897 & -0.5885 & -0.6239 \\ -0.3916 & 0.8240 & 0.2055 & 0.3543 \\ 0.5688 & 0.2390 & 0.7102 & -0.3390 \\ 0.7060 & 0.1554 & -0.3272 & 0.6086 \end{pmatrix}$$

$$S_1 = \begin{pmatrix} 7.2125 & 0 \\ 0 & 1.7262 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$V_1 = \begin{pmatrix} 0.1442 & 0.9895 \\ 0.9895 & -0.1442 \end{pmatrix}$$

$$U_2 = \begin{pmatrix} -0.4924 & 0.2653 & -0.8186 & -0.1304 \\ -0.5093 & -0.3073 & 0.3239 & -0.7357 \\ -0.4797 & 0.6948 & 0.4738 & 0.2504 \\ -0.5178 & -0.5936 & 0.0210 & 0.6157 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 2.0064 & 0 \\ 0 & 0.6960 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$V_2 = \begin{pmatrix} -0.9964 & 0.0850 \\ -0.0850 & -0.9964 \end{pmatrix}$$

$$U_3 = \begin{pmatrix} 0.4961 & 0.1357 & -0.5885 & -0.6239 \\ 0.4764 & 0.7780 & 0.2055 & 0.3543 \\ 0.5108 & -0.3460 & 0.7102 & -0.3390 \\ 0.5157 & -0.5066 & -0.3272 & 0.6086 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 203.6050 & 0 \\ 0 & 0.0611 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$V_3 = \begin{pmatrix} 0.0098 & 1.0000 \\ 1.0000 & -0.0098 \end{pmatrix}$$

(c) - 5 points

We can calculate G^TG for each of the matrices by hand. The results are:

$$G_1^TG_1 = \begin{pmatrix} 4 & 7 \\ 7 & 51 \end{pmatrix}$$

$$G_2^TG_2 = \begin{pmatrix} 4.00 & 0.30 \\ 0.30 & 0.51 \end{pmatrix}$$

$$G_3^TG_3 = \begin{pmatrix} 4 & 407 \\ 407 & 41451 \end{pmatrix}$$

We can compare these to the orthogonal eigendecomposition discussed in class. Here is the form of the orthogonal eigendecomposition:

$$G^TG = V\Sigma^TU^TU\Sigma V^T = V\Sigma^T\Sigma V^T$$

Note that U^TU should multiply out to the identity, we can check this with Matlab for U_1, U_2, and U_3. We also want to show that eigenvalues of G^TG are the squares of the singular values, and the corresponding eigenvectors are also shown in the columns of the V matrix. We can use the eig() function in Matlab to find the eigenvectors and eigenvalues for each G^TG.

```
EIGVECTORS1 =

   -0.9895    0.1442
    0.1442    0.9895


EIG1 =

    2.9796         0
         0   52.0204


SV1 =

    1.7262         0
         0    7.2125


EIGVECTORS2 =

    0.0850   -0.9964
   -0.9964   -0.0850
```

```
EIG2 =

    0.4844         0
         0    4.0256


SV2 =

    0.6960         0
         0    2.0064


EIGVECTORS3 =

   -1.0000    0.0098
    0.0098    1.0000


EIG3 =

   1.0e+04 *

    0.0000         0
         0    4.1455


SV3 =

    0.0611         0
         0  203.6050
```

(d) - 5 points

We can see that the singular values vary for each of the G matrices from part a. For G_1 and G_2. the singular values are only separated by 1 order of magnitude. This is due to the fact that the measurement points (x-values) are separated by roughly the same order of magnitude as their values: order of 1 for G_1, and order of 0.1 for G_2. However, the singular values for G_3 show a much greater disparity (4 orders of magnitude). This is due to the fact that the measurement points are on the order of 100 but are only separated by 1 or 4 each (2 orders of magnitude difference).

The ratio between the singular value is related with the error propagation from the data to the model parameters, as we will show.

The V matrices are made up of two vectors that are the orthonormal basis of the model space (eigenvectors of G^TG). $(G^TG)m$ is a stretch of m vector in the eigenvector directions by the eigenvalues, which should equal to the projection of data vector in the model space (G^Td). If there is an error in the data vector, the model parameters that are more related to the eigenvectors with large eigenvalues (or singular values) are more well constrained (with less error in the model parameters), and those that are related to the eigenvectors with smaller eigenvalues are not well constrained. If we look at the case for G_3 we see that the first singular value is large and the second is not. The eigenvector associated with the small value is:

$$\begin{pmatrix} 1.0000 \\ -0.0098 \end{pmatrix}$$

It means the first parameter (intercept) is not well constrained. It is same as our intuition. Let's think about two patches of data points. The first patch includes the data points associated with G_3, which are far away from 0; the second patch includes data points with similar separations but much closer to 0. For each patch, we can set a local coordinate centered at the data points. The errors in the slope and the intercept (because of the errors in data), would be similar for the two patches viewing from the local coordinate of each. However, viewing from the original coordinate, the error in the intercept will be much larger for the first patch because of the leverage effect.

Problem 3 (graded by Yiran) 40 points

(a)

In the absence of testable information, our intuition should tell us that the probabilities $p(A)$ and $p(B)$ are equal. Because the total probability is 1, we find

$$p(A) = p(B) = \frac{1}{2}.$$

(b)

The number of ways for choosing M items among N items is:

$$\left(\begin{array}{c} M \\ N \end{array} \right) = \frac{N!}{M!(N-M)!}.$$

(c)

We know that the binomial coefficients can be arranged in Pascale's triangle. It is easy to see that the maximum values in Pascale's triange are exacly the center values. Thus, the M that maximizes $F(M)$ is given by $M = N/2$. Thus, we have

$$p = \frac{M}{N} = \frac{1}{2}.$$

It agrees with the intuitive answer in (a).

(d)

Let $F(M) = \frac{N!}{M!(N-M)!}$. Maximizing $F(M)$ is equivalent to maximize $\log(F(M))$.

Note that $M = pN$.

$$\begin{aligned} \log(F(M)) &\approx N\log(N) - M\log(M) - (N-M)\log(N-M) \\ &= N\log(N) - Np\log(N) - Np\log(p) \\ &- N(1-p)\log(1-p) - N(1-p)\log(N) \\ &= N\left[-p\log(p) + (1-p)\log(1-p)\right] \\ &= NS(p) \end{aligned}$$

where we assumed large $N, M, N-M$ (using Stirling's approximation for the factorials). Thus, we can maximize S with respect to p instead of maximizing F with respect to M.

(e)

In order to find the maximum, we need to take the first derivative of $S(p)$ and set it to zero. We have

$$S'(p) = -[1 + \log(p) - \frac{1}{1-p} + \frac{p}{1-p} - \log(1-p)] = -\log\left(\frac{p}{1-p}\right) = 0.$$

This is equivalent to $1 - p = p$, which yields $p = 1/2$. In order for this to be a maximum, we need that $S'' < 0$. The second derivate is given by

$$S''(p) = -\frac{1-p}{p}\frac{1-p+p}{1-p} = -\frac{1}{p} < 0,\ \forall p > 0.$$

Thus, the maximum entropy solution is given by $p = 1/2$.