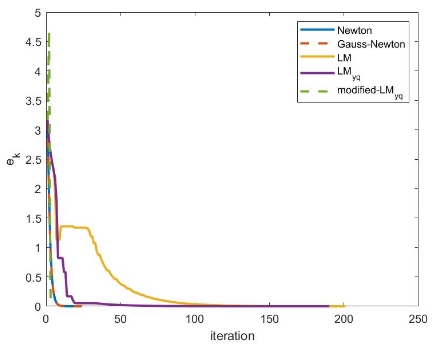
1 One-page report: Exercise 9

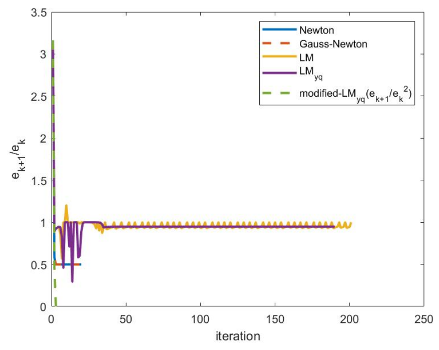
1.1 What makes the Powell's problem difficult to solve?

In the Powell's problem, the unique root x^* is $[0,0]^T$. However, after calculating the Jacobian matrix J, I found that the determinant of J is equal to 0 at x^*. It means that J is singular at x^*. Therefore, when the optimization method is approaching $x*$, the singularity of J increases, making the local approximation of $r(x)$ poor and the optimization harder to converge. That's why the Powell's problem is difficult to solve.

1.2 Comparisons of Newton's, Gauss-Newton, Levenberg-Marquardt



(a)



(b)

Differences and Similarities They all deal with nonlinear problems with local approximation. Gauss-Newton and L-M method are both modified from Newton's method. Newton's method needs to compute Hessian matrix, while the rest two use its approximation. L-M method belongs to trust-region methods. Newton's and Gauss-Newton all require start point not too far from the optimal point, while L-M method doesn't have such requirement.

Convergence rates According to my observation, the convergence rates of Newton's, Gauss-Newton and L-M method are both linear in our case (before changing the variables). Because their e_{k+1}/e_k all tend to be a less than constant between 0 and 1 when k is approaching infinite. In more general cases, as introduced in slides, Newton's method has quadratic convergence rate, while the rest two have only linear rate. But they can have quadratic convergence rate if the neglected term in the Hessian is very small.

Advantages and Limitations In most cases, Newton's method takes less iterations to converge than others whereas it's more computational expensive because of the Hessian matrix. It doesn't work properly when the Hessian matrix is not positive-definite. Gauss-Newton method is more computational efficient, but usually has linear convergence rate. And it doesn't work when Jacobian matrix is rank-deficient. L-M method is more robust than the other two methods. It often converge linearly. It works even when Jacobian matrix is rank-deficient. Plus, Newton's and Gauss-Newton all require start point not too far from the optimal point, while L-M method doesn't have such requirement.

1.3 In the last method, we simply changed the variables, then what happened when we called the L-M method? Why?

As it can be seen from figure 1(a) and 1(b) that, after the change, L-M method only took 2 iterations to converge. And the convergence rate became quadratic. The answer lies in the fact that after the change, the determinant of J is always 2: J is non-singular at all possible places, which significantly reduced the difficulty of the problem. Then the optimization method was able to work efficiently.

2 Rosenbrock problem

2.1 Calculate the Jacobian

Regarding Rosenbrock problem,

$$r(x) = \sqrt{2}\begin{bmatrix} 10(x_2 - x_1^2) \\ 1 - x_1 \end{bmatrix} \tag{1}$$

the Jacobian matrix is:

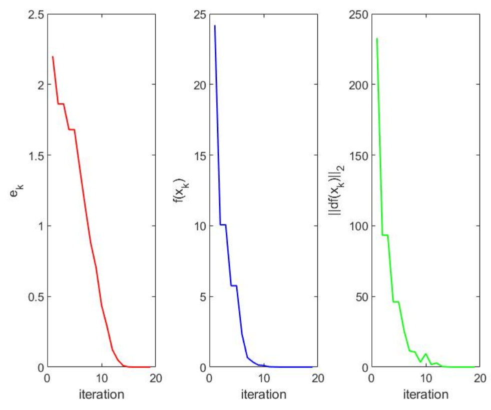
$$J = \begin{bmatrix} -20\sqrt{2}x_1 & 10\sqrt{2} \\ -\sqrt{2} & 0 \end{bmatrix} \tag{2}$$

2.2 Implement a Matlab function

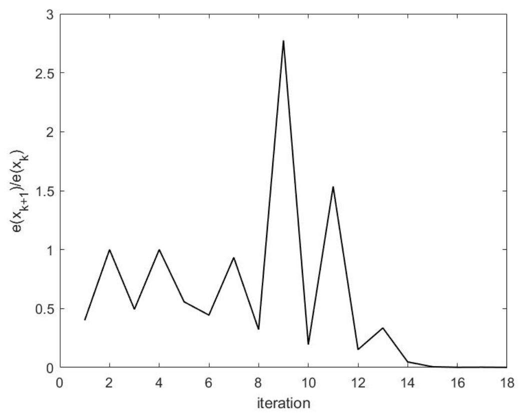
Here is the Matlab script.

```
function [r, J]=fun_rJ_Rosen(x)
x1 = x(1);
x2 = x(2);
r = sqrt(2) * [10*(x2 - x1^2); 1 - x1];
J = [-20*sqrt(2)*x1, 10*sqrt(2);
     -sqrt(2), 0];
end
```

2.3 L-M optimization



(c)



(d)

Before convergence, there are 18 iterations. From 1(d) we can see that the convergence rate is super-linear, since the ratio of e_k turned to be 0 when k is approaching infinite.

3 Linear least squares with weights

3.1 Compute the least squares fit without taking the difference in noise levels into account

With the help of LinearLSQ function, my solutions of all three parameters are

$$\begin{bmatrix} 0.7716 \\ 2.6843 \\ 0.2327 \end{bmatrix}$$

, and

$$||e||_2 = 0.8173$$

3.2 Take the difference of the noise levels into account

To take the difference of noise levels into account, we need to construct a weight matrix to add weights to the problem. The weight matrix is a diagonal matrix whose elements are each data point's corresponding standard deviation. Then, via multiplying weight matrix with 'A' and 'b' in the LSQ equation, weights were added. To clarify the process, I showed the Matlab script here.

```
data = load('data_exe3.mat');
x_star = [1.27; 2.04; 0.3];
phi_raw = [exp(-27.*data.t), exp(-8.*data.t), exp(0.*data.t)];
d_1 = ones(1,10)*1/0.5;
d_2 = ones(1,40)*1/0.1;
weight = diag([d_1 d_2]);
x = linearLSQ(weight*phi_raw, weight*data.y)
```

'x' in the script is the solution for this problem. Then we have the answer.

$$x = \begin{bmatrix} 1.1909 \\ 2.2734 \\ 0.2784 \end{bmatrix}$$

, and

$$||e||_2 = 0.2474$$

3.3 Which one is more accurate?

In line with the 2-norm of the absolute errors I gave above, the second one, which considered the difference of the noise levels,is more accurate.

4 Meyer's problem

4.1 Calculate the Jacobian

The calculated jacobian is

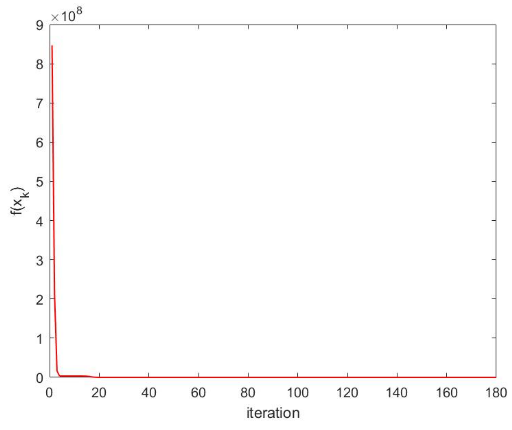
$$\begin{bmatrix} -exp(\frac{x_2}{t_i+x_3}) & -\frac{x_1 \cdot exp(\frac{x_2}{t_i+x_3})}{t_i+x_3} & \frac{x_1 x_2 \cdot exp(\frac{x_2}{t_i+x_3})}{(t_i+x_3)^2} \end{bmatrix} \tag{3}$$

for each $r_i(x)$. The corresponding Matlab scripts are:

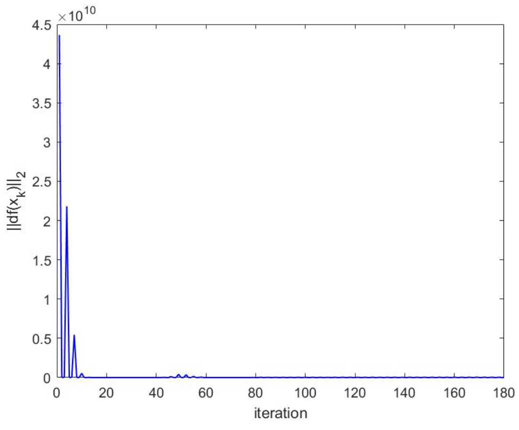
```
function [r, J]=fun_rJ_Meyer(x, t, y)
x1 = x(1); x2 = x(2); x3 = x(3);
r = y - x1 .* exp(x2./(t+x3));
J = [-exp(x2./(t+x3)), -(x1.*exp(x2./(t+x3)))./(t+x3) ,...
(x1.*x2.*exp(x2./(t+x3)))./(t+x3).^2];
end
```

4.2 L-M optimization

Here are the figures of $f(x_k)$ and $||\nabla f(x_k)||_2$:



(e)



(f)

L-M method took 179 iterations to converge here.

4.3 Calculate the Jacobian

The calculated jacobian is

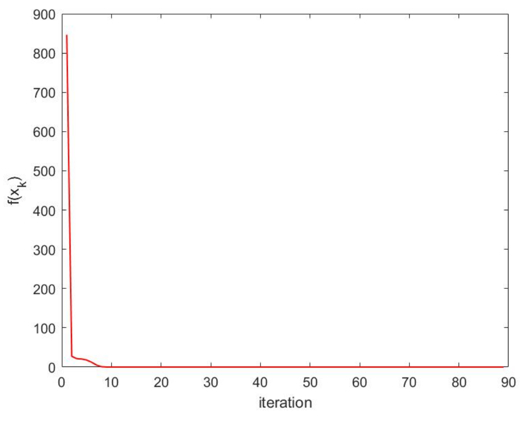
$$\left[\begin{array}{ccc} -exp(\frac{10z_2}{u_i+z_3}-13) & -\frac{10z_1\cdot exp(\frac{10z_2}{u_i+z_3}-13)}{u_i+z_3} & \frac{10z_1z_2\cdot exp(\frac{10z_2}{u_i+z_3}-13)}{(u_i+z_3)^2} \end{array}\right] \tag{4}$$

for each $rho_i(x)$. The corresponding Matlab scripts are:

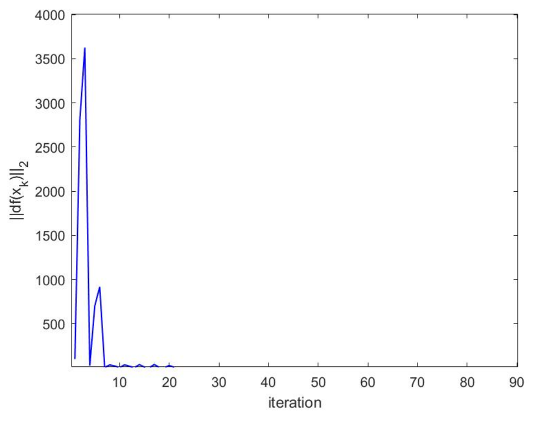
```
function [r, J] = fun_rJ_Meyer2(z, u, y)
z1 = z(1); z2 = z(2); z3 = z(3);
r = 1e-3.*y - z1.*exp(10.*z2./(u+z3)-13);
J = [-exp(10.*z2./(u+z3)-13),...
    -10.*z1.*exp(10.*z2./(u+z3)-13)./(u+z3) ,...
    10.*z1.*z2.*exp(10.*z2./(u+z3)-13)./(u+z3).^2];
end
```

4.4 L-M optimization

Here are the figures of $f(x_k)$ and $||\nabla f(x_k)||_2$:



(g)



(h)

L-M method took 88 iterations to converge here.

4.5 Compare both formulations

Comparing both formulations, the second one needs less iterations(179 V.S. 88). From the result I learned that by making some changes on variables, we can make the optimization problem easier.