Report

Module design

There are five different classes in this program: *"FullMatrixSolver"*, *"EKV_Model"*, *"NewtonPowerLaw"*, *"Tests"*, *"Main"*.

The class *"FullMatrixSolver"* is for task 1. And the purpose of this class is to direct solve the equation $Ax = b$, in which A is a full matrix and b is a vector. There are four modules in this class. The first module is used to find the pivot with the minimum fill-in algorithm. The second module is to do the LU decomposition, and the third module is to do the backward substitution. Finally, the forth module integrates all these modules and solve the equation.

The class *"EKV_Model"* includes all the functions used to extract the parameters of EKV model with the help of given measured data. This class extracts the parameters in two different ways: Qusi-Newton and Secant. Roughly, there are five modules in this class. The first module is to compute the objective functions. In this model the functions are:

$$\frac{\partial I_d}{\partial I_s} = 0$$

$$\frac{\partial I_d}{\partial k} = 0$$

$$\frac{\partial I_d}{\partial V_{th}} = 0$$

The second module is used to compute *"delta_para"* for the Newton method. The first step of this module is to get the Hessian matrix. And this module provides two methods: Qusi-Newton method and Secant method to get the Hessian matrix. The second step of this module is to compute *"delta_para"* by solving the equation

$$H * delta_{para} = -f$$

The third module is used to compute the loss function, and the forth module is to integrate all these modules to do the Newton method and line search. Thus, the forth module will return the parameters extracted for the EKV model. And the convergence is also automatically checked in this module. If the loss of the model using the resulting set of parameters is less than 0.1, we will conclude that the method converges. As for the quadratic convergence check, we will utilize the following property of quadratic convergence:

$$\lim_{k \to \infty} \frac{||V_k||}{(||V_{k-1}||)^2} = const$$

During the convergence process, we will record the values of $\frac{\|V_k\|}{(\|V_{k-1}\|)^2}$ in an *ArrayList*. After this process comes to an end, we will compute the difference between the last two items in the *ArrayList*. If the difference is less than 10^{-3}, we will say that the quadratic convergence is detected. In this way, we can only detect the quadratic convergence in the cases that the Newton method needs more than three iterations to converge. Finally, the fifth module is to compute the norm which will be used in the line search of the forth module.

The third class *"NewtonPowerLaw"* is for the task 2. The purpose of this class is to extract the parameters for the power law using Newton method. The modules in this class is similar as the modules in *"EKV_Model"* class.

The *"Tests"* class includes all the functions used for doing tests and validations. To test the performance of the full matrix solver, I take a previous hacker practice as an example and use the full matrix solver to solve it. And I compute the second norm between $A * x$ and b. Only if the second norm is less than 10^{-7}, I conclude that the solver passes the test. The strategy of validating the power law parameter extraction is as following. First, we can transform the original model $y_i = a{x_i}^m$ into $\ln y_i = m * \ln x_i + \ln a$. Thus, after doing some modifications to the input data, we can extract the parameters by just doing linear regression. As a result, the second step is to get the parameters using linear regression. Finally, I compute the loss of the model using this set of parameters. If its loss is greater than the loss of the model using the set of the parameters got by the class *"NewtonPowerLaw"*, I will conclude that the power law parameter extraction passes the validation. For the task 7, the strategy is to compare the output of the model with the output of a desired approximation function. If the norm between these two outputs is less than 0.1, I will say that the model passes corresponding the validation.

The final class *"Main"* is to generate the desired output of each task.

Results discussion

The result of task 1: After solving the equation $A * x = b$ using the full matrix solver, I got the vector x. The second norm between $A * x$ and b is about 3.662×10^{-15}. Thus, the full matrix solver passes the test.

The result of task 2: For task 2, I generate the data points randomly with 10%-20% noises. I found that the convergence of the Newton method strongly depends on the choice of initial guess. Some initial guesses will lead to the set of parameters with the loss lager than 2000. After trying many initial guesses, I choose the initial guess $a = 20,\ m = -1$. And it will lead to the set of parameters $a = 10.0918,\ m = -0.5032$. The loss of the model using this set of parameters is 0.0374. In the validation process, I got another set of parameters using linear regression. Using this set of parameters, the loss of the model is 8.6662 which is larger than the loss of the model using the previous set of parameters. Thus, the Newton method passes the validation.

The result of task 3: I use Excel to plot the measurement data file. Here are the observations. When $V_{GS} = 0.5$, the measured I_D is very small and there is no obvious relationship between I_D and V_{DS}. When the value of V_{GS} is greater than 0.5 and the value of V_{DS} is less than a certain value, I_D is quadratic to V_{DS}. But when the value of V_{DS} is greater than a certain value, I_D is nearly insensitive to V_{DS} and converges to a certain value. And this value will increase as the value V_{GS} getting greater.

The result of task 4: I use the initial guess $I_s = 10^{-7}A,\ k = 1$ and $V_{th} = 1V$ and this guess will converge to a set of parameters for both Quasi-Newton method and Secant method (the second initial guess is $I_s = 7.3218 * 10^{-7}A,\ k = 0.9588$ and $V_{th} = 1.1665V$). The results are shown in the file *consoleOutput.txt.* For the Quasi-Newton method, the loss of the model using its resulting parameters is larger than the loss of the model with the parameters got by Secant method. As for the sensitivities of each parameter, they will all converge to 1. And the increment vector keeps decreasing during the process of converging.

Both methods converge according to the criteria we set in the code. But neither the Quasi-Newton method nor the Secant method was detected quadratic convergence. For the quasi-Newton method, it is because the method converges with only one iteration. Thus, the method that we used to detect quadratic convergence cannot successfully detect the quadratic convergence.

The result of task 5: For this task, I use the set of parameters that I got from task 4 using Quasi-Newton method. And then I repeat the task 3, so that I plot the measured data again with the y-axis is $I_D(V_{GS}, V_{DS};\ I_S, k, V_{th})/I_{Dmeasured}$. Here are the observations. When $V_{GS} = 0.5$, the I_D output by our model is very small, so the value of y-axis is almost 0 for each data point. When the value of V_{GS} is greater than 0.5 and the value of V_{DS} is less than a certain value, the value of the y-axis is quadratic to V_{DS}. But when the value of V_{DS} is greater than a certain value, the value of the y-axis is nearly insensitive to V_{DS} and converges to 1.

The result of task 6: The output of this task is written in the file *task6Output.txt*. Here are the observations. For all the initial guesses in the searching space, the Quasi-Newton method will always converge. But some combinations of these initial guesses in the searching space will lead divergence for the Secant method. For the Quasi-Newton method, the best initial guess is $I_s = 3 * 10^{-6}A$, $k = 0.5$ and $V_{th} = 1.0V$, since the resulting parameters will lead to the minimum loss of the model. And the second norm of the loss is 0.0016. For the Secant method, the best initial guess is $I_{s1} = 1 * 10^{-5}A$, $k_1 = 0.7$, $V_{th1} = 1.0V$ and $I_{s2} = 3 * 10^{-6}A$, $k_2 = 0.6$ and $V_{th} = 0.9V$. And the second norm of the loss is 0.0016.

As for the quadratic convergence, we find that in most cases the quadratic convergence is not detected and we have the following analysis. For the Quasi-Newton method, this method usually converges with less than three iterations. But for the Secant method, it just seldom meets the criteria we set for the detection of quadratic convergence.

The result of task 7: In this task, all the three conclusions are validated. For the first conclusion, I compare the value of the output of the model with the value of a reasonable exponential function (Using the equation (8) in the handout). And the second norm computed by the validation method is less than 0.1. Then I randomly change the value of V_{DS}, the model will still pass the validation. Thus, the first validation is validated. For the other two functions, I change the approximation functions, and the results are shown in the file *consoleOutput.txt.*