Assignment 3 Report

Clarification:

- Task 1: the implementation, testing and validation of matrix solver is in ***matrixSolver.java***
- The implementation of Task 2,4,6,7 are in ***mainP3.java***
- The code of Task 3 and Task 5 are in ***task3.matlab / task5.matlab*** in the file *Plots*
- Related methods of this project are in ***EKVModel.java***, ***rowFucntion.java***(basic method of matrix) and ***test.java***(method of second norm)
- Checks for additional helper functions are in ***testClassP3.java***
- There are detailed output for task1, task6 task7 and testClass.
- Simple output is in ***Project3SimpleOutput.txt***

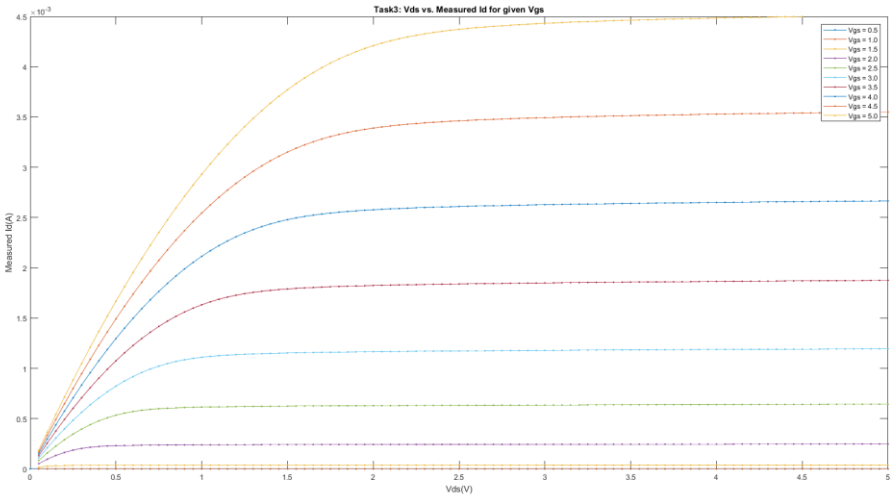
Task 1 : Using an ill-conditioned matrix to test full matrix solver, that is, creating a matrix with two nearly degenerate lines (the perturbation is changing from 0.1 to 10^{-21}) and obtaining the value of X in equation A*X=B.
The value of X starts to become NaN since perturbation equals to 10^{-16}, however, for X = (0.1~10^{-14}), the second norm between A*X and B is changing from 6.431098710768743E-13 to 0.25. In this case, we think that our matrix solver is robust enough to solve matrices with rank up to 4.
The detailed output is in Task1DetailedOutput.txt.

Task 2: We generated 10 data samples randomly with 10%-20% noise firstly and then we calculated a and m using newton method and linear regression respectively. What's more, we found that the initial guess is very important for quasi newton method, therefore, we tried different initial guess, including initial guess from class. Finally, we choose a=20, m=-1. And we get the following result:

From linear regression
the power m is -0.4986393577731087, and the coefficient a is 22.77749448598143
the norm of m is 0.5013606422268913, the norm of a is 2.7774944859814283
From Quasi Newton method
the power m is -1.0000000000000004, and the coefficient a is 20.000000000000004
the norm of m is 4.440892098500626E-16, the norm of a is 3.552713678800501E-15

From the result, we found that the quasi newton method has a better solution than linear regression, and it passed the validation since the norm of *a* and *m* are quite small.

Task3: For Task 3 with the quasi-Newton method, we did below plotting with MATLAB.



And following is our comments on the observation.

(1) Generate the plots of $\log I_D \ vs. V_{GS}$ with two V_{DS} for both $I_{Dmeasured}$ and I_{Dmodel}. We choose $V_{DS} = 1 \ and \ V_{DS} = 3$ respectively. The result of $I_{Dmeasured}$ and I_{Dmodel} are similar and we can observe the quadratic according to the graph.

(2) Generate the plots of $I_D \ vs. V_{DS}$ with the 10 values of V_{GS} for both $I_{Dmeasured}$ and I_{Dmodel}. When $V_{GS} = 0.5 \ V$, $I_D and \ V_{DS}$ are almost irrelevant because the graph is more like a random curve. When V_{GS} is larger than 0.5, we can observe the relationship between these two. Under a certain value of V_{GS}, $I_D and \ V_{DS}$ are quadratic when V_{DS} is relatively small. Then with V_{DS} increasing, I_D converges to a fixed value. Besides, the value of I_D increases with V_{GS} getting larger.

Task4

For ***quasi-newton method***, we get the best fit of Is, k and Vth in one loop. At first, we set the criteria to break loop as the norm of loss function ||V|| is less than 10^{*-7}, however, as V may step into local minimum and V may not be close to zero which leads to a result that the method could loop for thousands of times, we change the criteria to the norm of parameters is less than 10^{*-7}. In this case, the result converges in one loop.

The initial guess we use is Is = 10^{-7}A, k = 1, Vth = 1V.

The result we get is:

the counter is 1

the Is is 9.999999999999996E-8, the k is 0.9999999999999997, the Vth is 1.000000000000001, the second norm of v is 0.0027669813585974794, the delt is 1.9386019080010366

With ***Secant method***, we can also get the best fit of Is, k and Vth in one loop after we have tried several different initial guesses. This solver is very sensitive to the two initial guesses. We can obtain very small norm but will also get NaN under the same conditions just with different initial guesses.

The initial guess we use is Is(k) = 3 * 10^{-5} A, k(k) = 0.9, Vth(k) = 2.0V; Is(k-1) = 10^{-8}A, k(k-1) = 0.5, Vth(k-1) = 0.8V.

The result we get is:

counter is 1

the Is is 3.0681136821918327E-8, the k is 0.5000001112125411, the Vth is 0.7999998991968501, the second norm of V (absolute residual) is 1.0319181610760937, the delt is 9.403274749889826E-11

We also implement an automated check of the quadratic convergence for ||v|| as well as ||delt||, as there is only one loop for both of the solvers, this check is useless, but it is useful in task6.

Comparison: If we ca find a good initial guess when using secant method, then these two method will have similar performance. While secant solver will have very poor performance if the initial guess we choose is bad.

Task5

(1) For the Task5, we first repeat Task 3 by $S_{model} = I_D(V_{GS,} V_{DS}; I_{S,} \kappa, V_{\text{th}})/I_{Dmeasured}$. We also did plotting by MATLAB. And following is our comments on the observation.

When$V_{GS} = 0.5\,V$, the result is similar with what we obtained in task3: $I_D and\, V_{DS}$ are almost irrelevant. But what the difference is that with pre-processing the data, the graph is more like an impulse function rather than totally random, which means that most part of the graph is zero and a huge fluctuation exists.

When V_{GS} is larger than 0.5, we can also observe the quadratic relationship between $I_D and\, V_{DS}$ when V_{DS} is relatively small with a fixedV_{GS}. Then with V_{DS} increasing, I_D converges to a fixed value. Compared with result of non-normalized S_{model}, I_D converges faster.

On the contrary with the result of non-normalized S_{model}, with V_{GS} getting larger, the value of normalized I_D decreases.

(2) For the Task5, we also repeat Task 4 by $S_{model} = I_D(V_{GS,} V_{DS}; I_{S,} \kappa, V_{\text{th}})/I_{Dmeasured}$.

The analysis of results obtained with normalized data is shown in task6.

Task6: The detailed output is in Task6Output.txt

For ***quasi-newton method***: we found that the quasi-newton method can converge in most of cases if we choose the initial guess from the search space. And we can get the following result:

(1) For non-normalized data

The smallest ||V|| is 2.5381491606342493E-6, I_S is 1.0E-6, k is 0.9, Vth is 1.1.

(2) For normalized data

The smallest ||V|| is 115.42362771441067, I_S is 3.0E-6, k is 0.6, Vth is 1.7

As for quadratic convergence, the majority of initial guesses have one iteration, but we still can observe quadratic convergence in detailed output.

Then, the difference of smallest V between non-normalized data and normalized data was compared. We notice that the smallest V in normalized data is pretty larger than the one in non-normalized data. After discussion, we conclude that, normalization can adjust values measured on different scales to a notionally common scale, often prior to averaging. In this case, we will not ignore or amplify the value in different scale which is very 'fair' to every data sample.

With ***Secant method***, we can get the following result:

(1) For non-normalized data

The smallest ||V|| is 8.200055873781653E-4, I_S is 3.0E-6, k is 0.7, Vth is 1.0

(2) For normalized data

The smallest ||V|| is 3090291.2572293007, I_S is 3.0000000000000004E-8, k is 0.5, Vth is 0.8.

With Secant method, the absolute deviations increase sharply. Besides, with the small increment vector, most of the attempts will break in one iteration. So it is hard to observe quadratic convergence.

	I_S	κ	V_{th}	$\|V\|$	$\|\Delta\|$
Task4(quasi Newton)	1.0E-6	0.9	1.1	2.5381491606342493E-6	4.954308893110746E-17
Task4(secant)	3.0E-6	0.7	1.0	8.200055873781653E-4	2.007923012523805 E-11
Task5(quasi Newton)	3.0E-6	0.6	1.7	115.42362771441067	0.02216113976991968
Task5(secant)	3.0000000000000004E-8	0.5	0.8	3090291.2572293007	5.7063097238377525

Task7:

1. Visualization has been realized, comments on the observation are shown in task 3 and task5. All the plots are in the file *Plots*.

2. Execute the validation checks.

If Vt =0.026, k<1 and Vgs < Vth, we can know that k(Vgs-Vth) <0, vice versa.

Similarly, if Vds > k(Vgs-Vth), k(Vgs-Vth) -Vds <0.

According to the approximation below:

$$\log^2\left(1+e^{x/2}\right)\approx e^x \qquad x << 0$$

$$\log^2\left(1+e^{x/2}\right)\approx \left(\frac{x}{2}\right)^2 \qquad x >> 1$$

The three checks can be translated as:

(1) Comparing the value of Id using

$$I_{D,EKV} = \underbrace{I_S \log^2\left(1+\exp\left(\frac{\kappa(V_{GB}-V_{th})-V_{SB}}{2V_T}\right)\right)} - \underbrace{I_S \log^2\left(1+\exp\left(\frac{\kappa(V_{GB}-V_{th})-V_{DB}}{2V_T}\right)\right)}$$

And Id = Is*e^{x1}-Is*e^{x2}, where x_1 = k*(Vgs-Vth)/Vt, x_2 = [k*(Vgs-Vth)-Vds]/Vt

(2) Comparing the value of Id using

$$I_{D,EKV} = \underbrace{I_S \log^2\left(1+\exp\left(\frac{\kappa(V_{GB}-V_{th})-V_{SB}}{2V_T}\right)\right)} - \underbrace{I_S \log^2\left(1+\exp\left(\frac{\kappa(V_{GB}-V_{th})-V_{DB}}{2V_T}\right)\right)}$$

And Id = Is*$(x_1/2)^2$-Is*e^{x2}, where x_1 = k*(Vgs-Vth)/ Vt, x_2 = [k*(Vgs-Vth)-Vds]/Vt

(3) Comparing the value of Id using

$$I_{D,EKV} = \underbrace{I_S \log^2\left(1+\exp\left(\frac{\kappa(V_{GB}-V_{th})-V_{SB}}{2V_T}\right)\right)} - \underbrace{I_S \log^2\left(1+\exp\left(\frac{\kappa(V_{GB}-V_{th})-V_{DB}}{2V_T}\right)\right)}$$

And Id = Is*$(x_1/2)^2$ -Is*$(x_2/2)^2$, where x_1 = k*(Vgs-Vth)/Vt, x_2 = [k*(Vgs-Vth)-Vds]/Vt

We implement the check by choosing the eligible parameters from 520 records generated from task6, and calculate the second norm of two Id. At the same time, for check 1 and check 2, we change the value of Vds to observe that if Id is sensitive to Vds. If the norm does not change a lot, we will say Id is insensitive to Vds. The detailed output is in Task7DetailedOutput.txt.

We get the following conclusion from the output:

(1) Id is an exponential function of Vgs and insensitive to Vds when k<1 and Vgs < Vth
The criteria of norm is 10^{-6}.
(2) Id is quadratic to Vgs and insensitive to Vds when Vds > (Vgs-Vth)*k and Vgs > Vth
The criteria of norm is 10^{-2}.
(3) Id is quadratic to Vgs when Vds < (Vgs-Vth)*k and Vgs > Vth
The criteria of norm is 10^{-2}.

Test:

The output of testClass.java is in testClassOutput.txt
We implement necessary checks for every helper function in testClassP3.java. The input and output are shown within the class.
However, for wrapped method, such as ***GetSmallestV()*** and ***quaNewSolver()***, they are tested in mainP3.java by observing the second norm or convergence of the result. Besides, we test every helper functions within it.
Similarly, the test for wrapped methods parameter extraction can be done from the implementation of task2. Known the ground truth of parameter a and m, we obtain the error between ground truth and calculated value. As the error is less than the criteria (usually 10^{*-7}), we think that this method has passed the test.