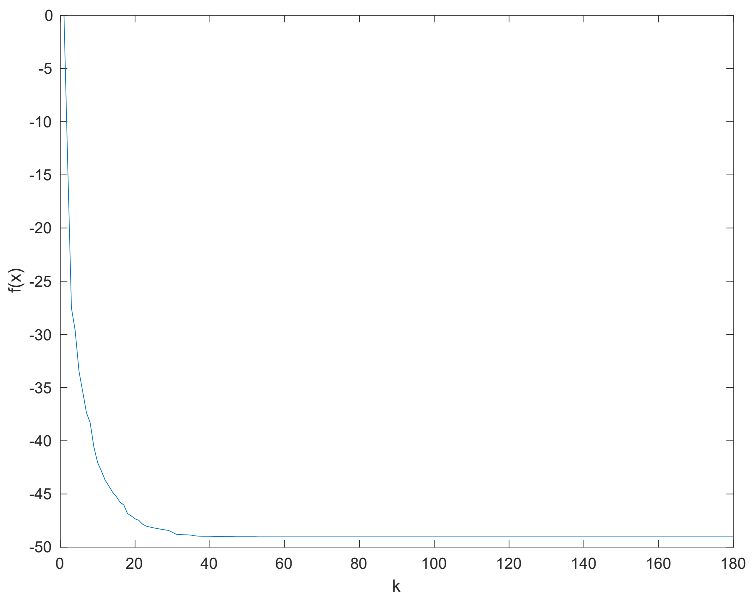
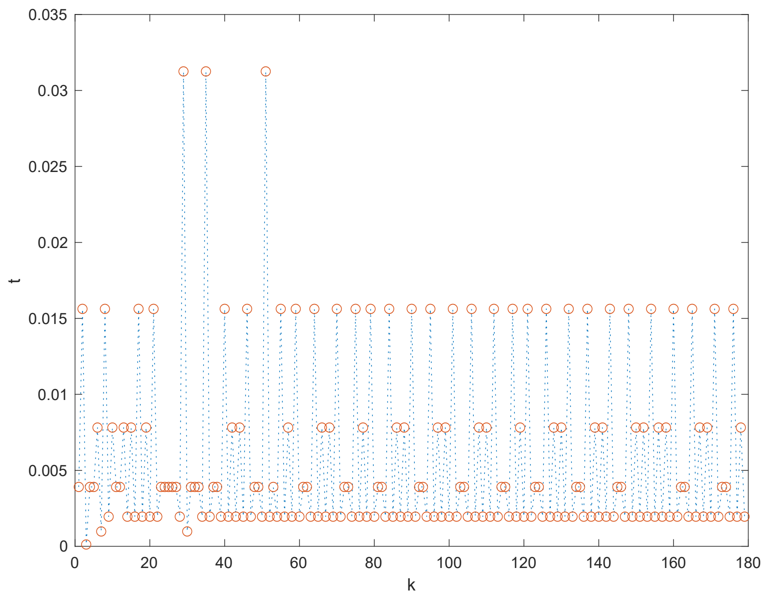
APPENDIX 9.30

Gradient Method

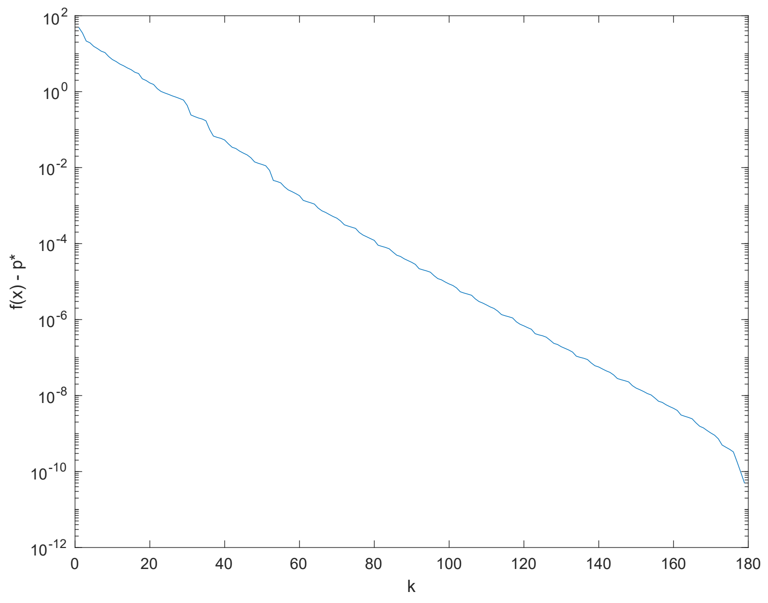
- As instructed by the problem, we generate a_i randomly based on some distribution. We select normal distribution within this case.
- We set $m = 100, n = 40$ for experiment.
- As for algorithm settings, we set iteration max as $k = 1000,\ \alpha = 0.25,\ \beta = 0.5$ for backtracking line search, and $\eta = 1e^{-4}$ for stopping criterion.
- Below then shows the corresponding results:
    - Objective versus iterations:



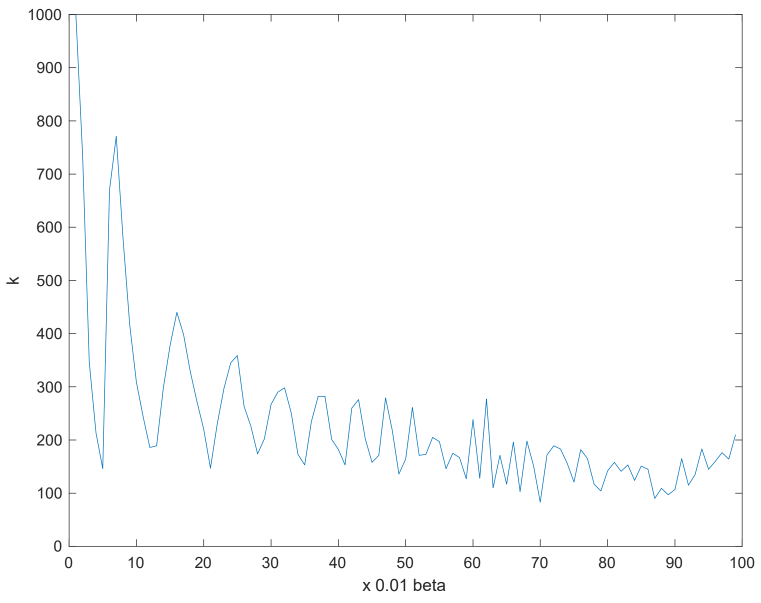
    - Step length versus iterations:



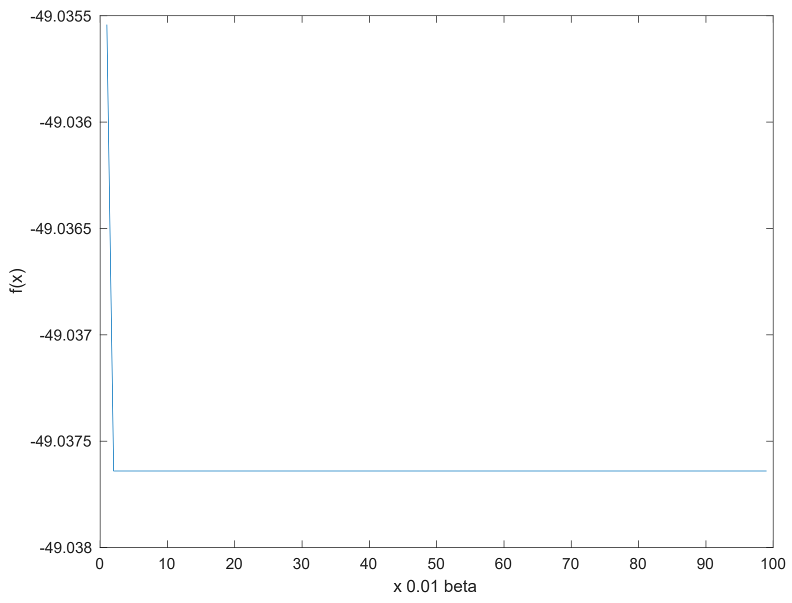
    - $f - p^*$ versus iterations:



- Brief discussion:
    - A few tests with different n and m are carried out, and similar trends in results are observed. For brevity, detailed graphs are omitted.
    - We also test the effects induced by setting different α, β:
        - With a fixed $\alpha = 0.49$, we vary β, and retrieve the following results:
            - Iteration versus beta:

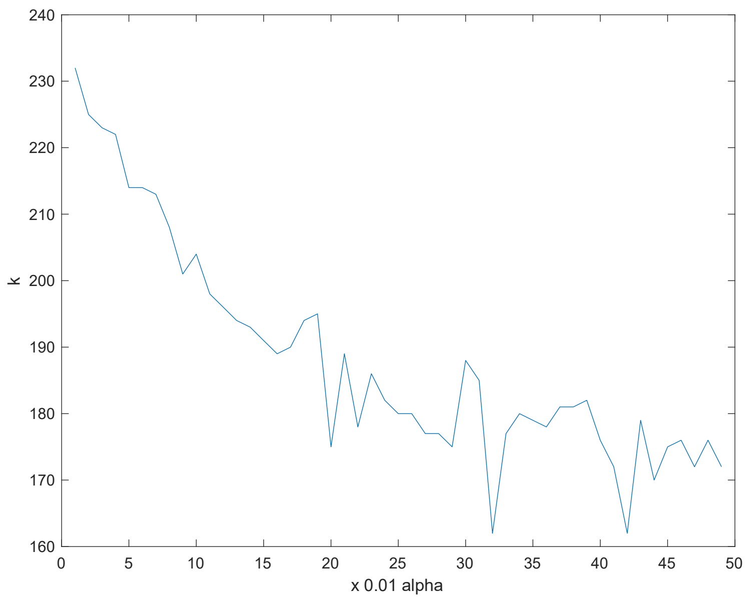


- Optimal value versus beta:

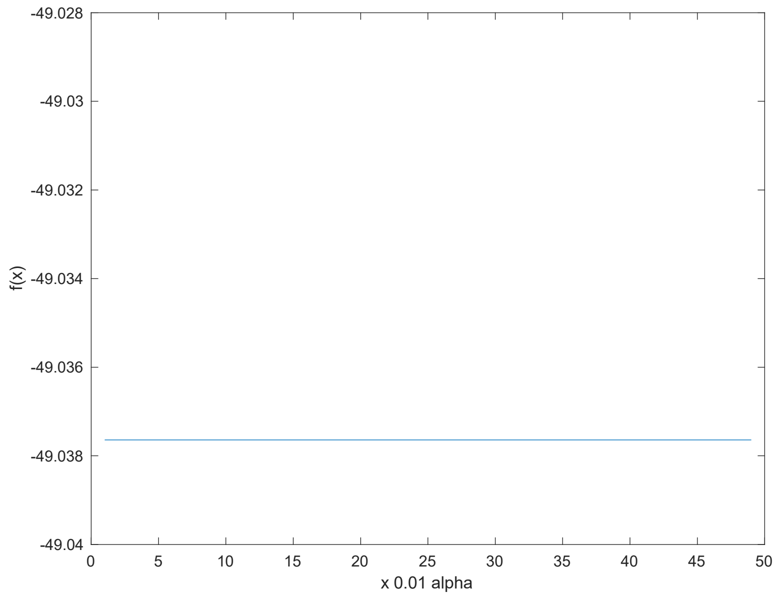


Which can be observed that, the reached optimal values are similar. Yet, the iteration k fluctuates with beta, and seemed to be having a decreasing trend as beta approaches 1. Note that we also tested with different α values, and retrieve **different trend**; for brevity we do not show all tests. Therefore, the above observation could only serve as empirical statement.

- Similarly, we make an empirical observation here. With a fixed β, we vary α:
  - Iterations versus alpha:



  - Optimal value versus alpha:

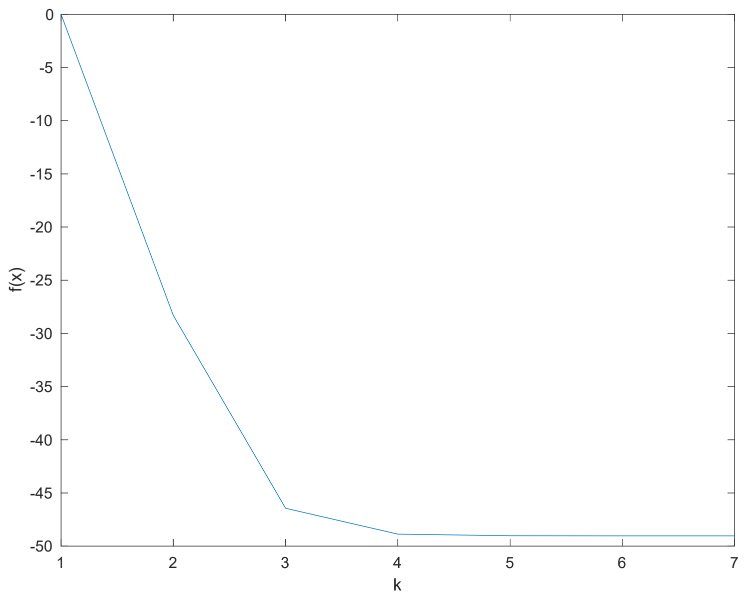


Which can be observed that with alpha approaches 0.5, the iteration tends to decrease; meanwhile, optimal value does not fluctuate much. Yet, again, this only serve as a pure empirical observation without rigorous analysis.

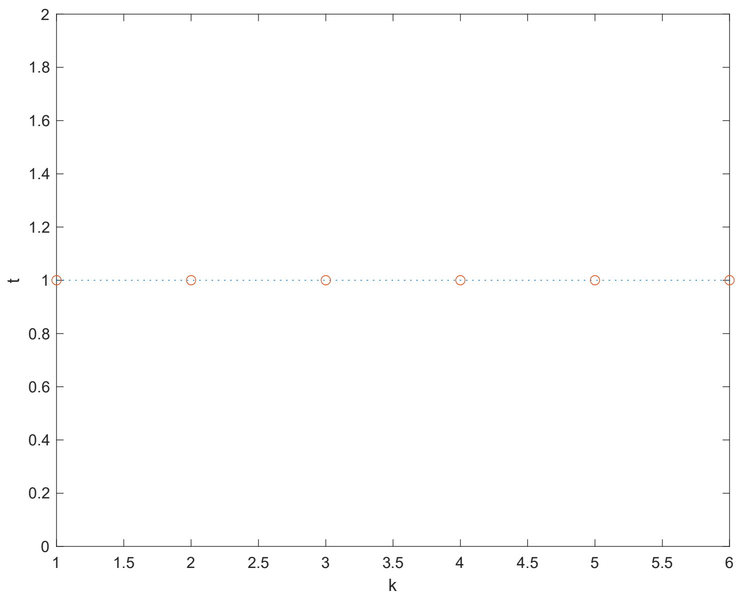
- The implementation code in MatLab script is shown on next page.

Newton Method

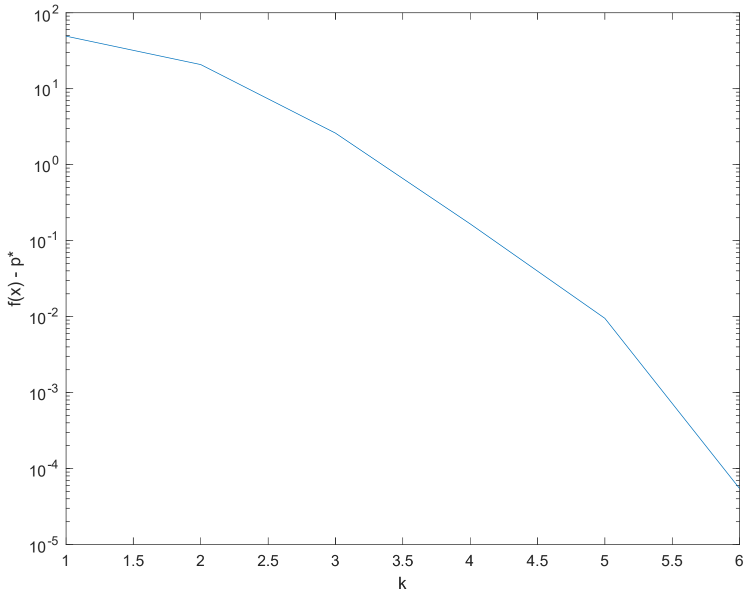
- We set an identical problem as in 9.30(a).
- Below shows the results. It could be seen that it yields a much faster converge rate with less iteration.
    - Objective versus iterations:



    - Step length versus iterations:



    - $f - p^*$ versus iterations:

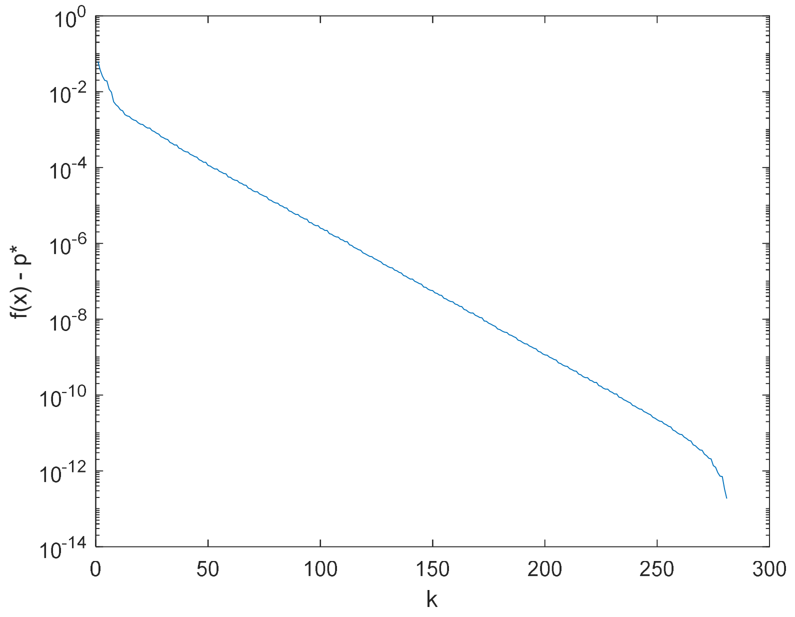


- On next page, we show the implementation of this problem with Newton method in MatLab script.

APPENDIX 9.32

Gauss-Newton Method

- As instructed, we generated $A_i \in \boldsymbol{S}_{++}^n$, and with $b_i^T A_i^{-1} b_i \leq 2$, for $i = 1, \dots, m$.
- In particular, we set $m = 100, n = 40$.
- For step size, we implement backtracking line searching method, and set $\alpha = 0.25$, $\beta = 0.5$.
- Below shows the result of Gauss-Newton method on nonlinear least-squares problems:

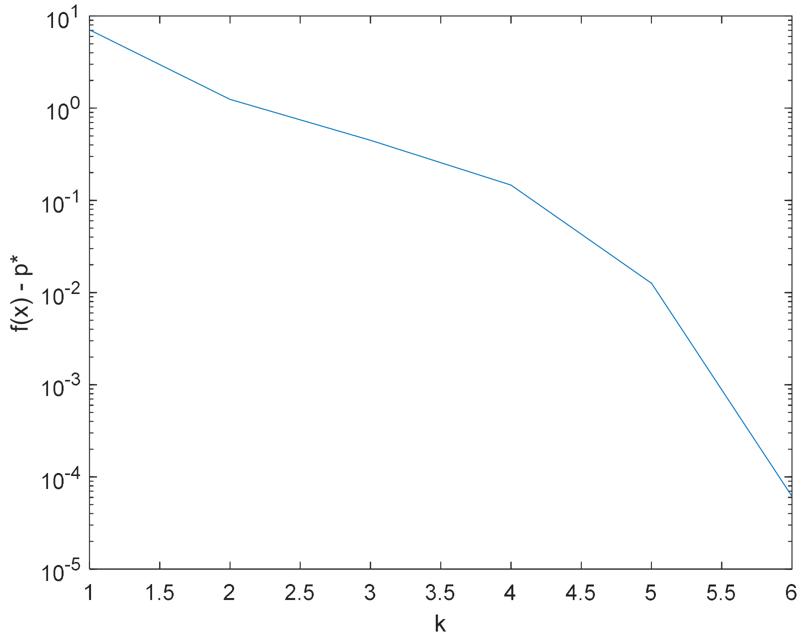


- From above, it yields a linear convergence rate, which is slower than Newton method; it is due to the fact that Gauss-Newton is an approximation of Newton method. Despite the lower convergence rate, in practice, the method is adopted if the approximation is close, and could then alleviate the computation complexity on calculating the Hessian matrix.
- On next page, we show the implementation of MatLab code.

APPENDIX 10.15

Newton Method for equality constrained minimization

- As instructed, we set $p = 30, n = 100$.
- In addition, we generate A randomly with full rank n, while generate $\hat{x}$ with uniform distribution within $[0,1]$.
- We then set equality constraint $b = A\hat{x}$.
- The Newton step is calculated through KKT matrix.
- Below shows the final numerical result. In which a quadratic convergence rate could be observed:

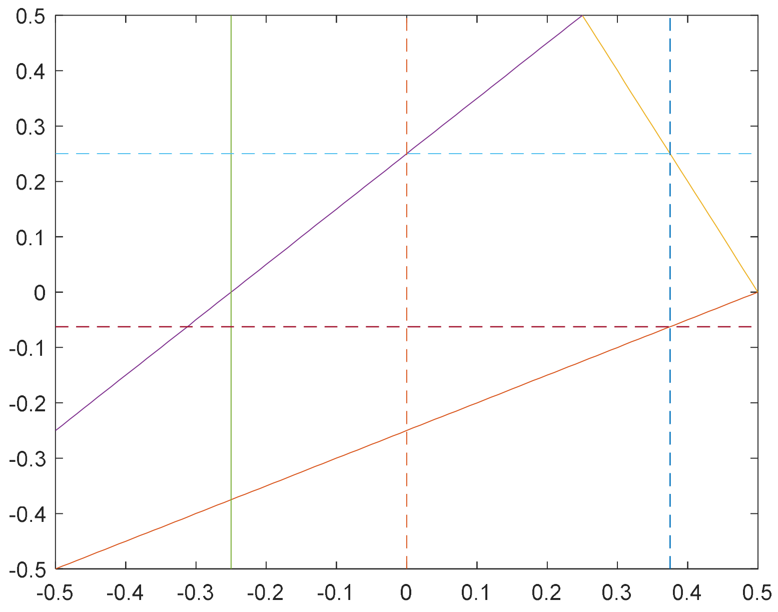


- The MatLab code of this implementation is shown on next page:

APPENDIX 11.22

Barrier Method

- We set $\eta = 1e^{-4}$, and $\epsilon = 1e^{-4}$.
- For step size, we implement backtracking line searching method, and set $\alpha = 0.25$, $\beta = 0.5$.
- We use Newton Method to calculate step.
- Below shows the final result:



- The MatLab code of this implementation is shown on next page: