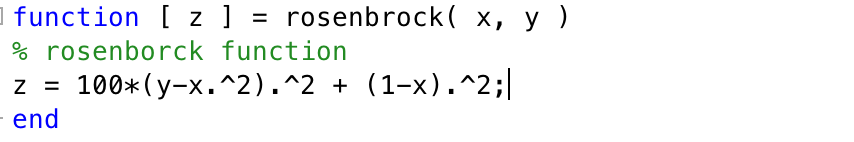
Introduction to Numerical Analysis Project5

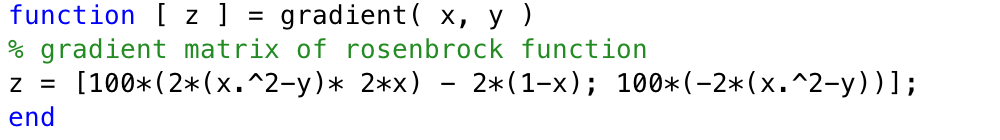
20120281김영은

Problem1. Rosenbrock’s function

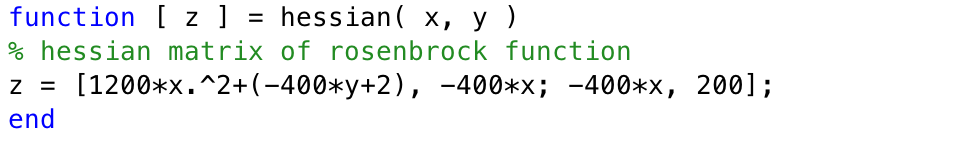
At first, define function rosenbrock.m, gradient.m which is gradient of rosenbrock function and hessian.m which is hessian matrix of rosenbrock function for utility.



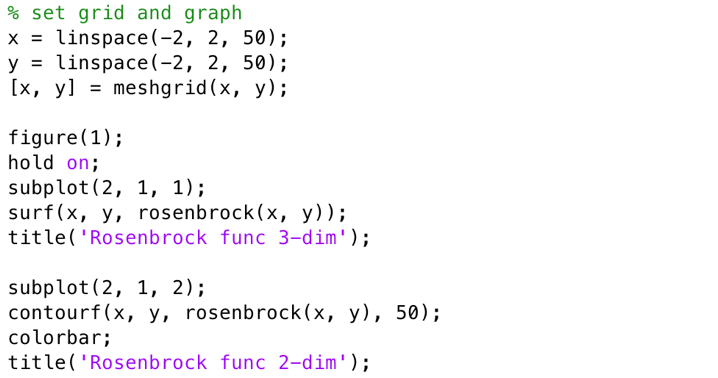
Code 1 rosenbrock.m



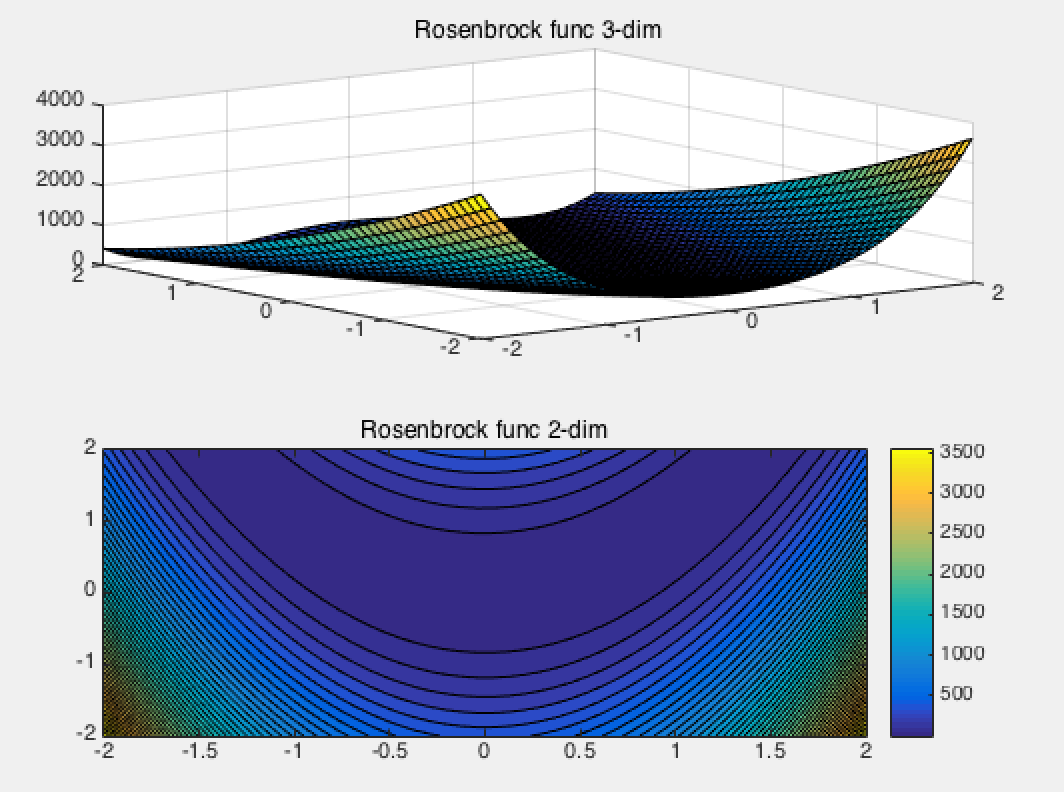
Code 2 gradient.m



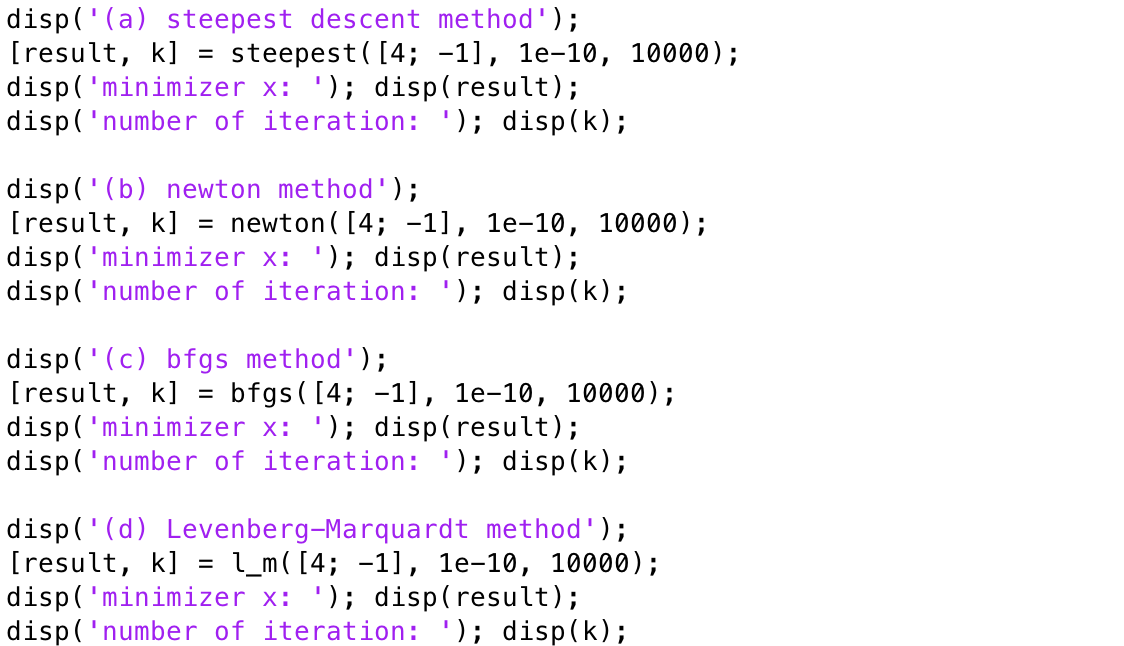
Code 3 hessian.m



Code 4 for graph of rosenbrock function in pj4.m



Graph 1 Rosenbrock function



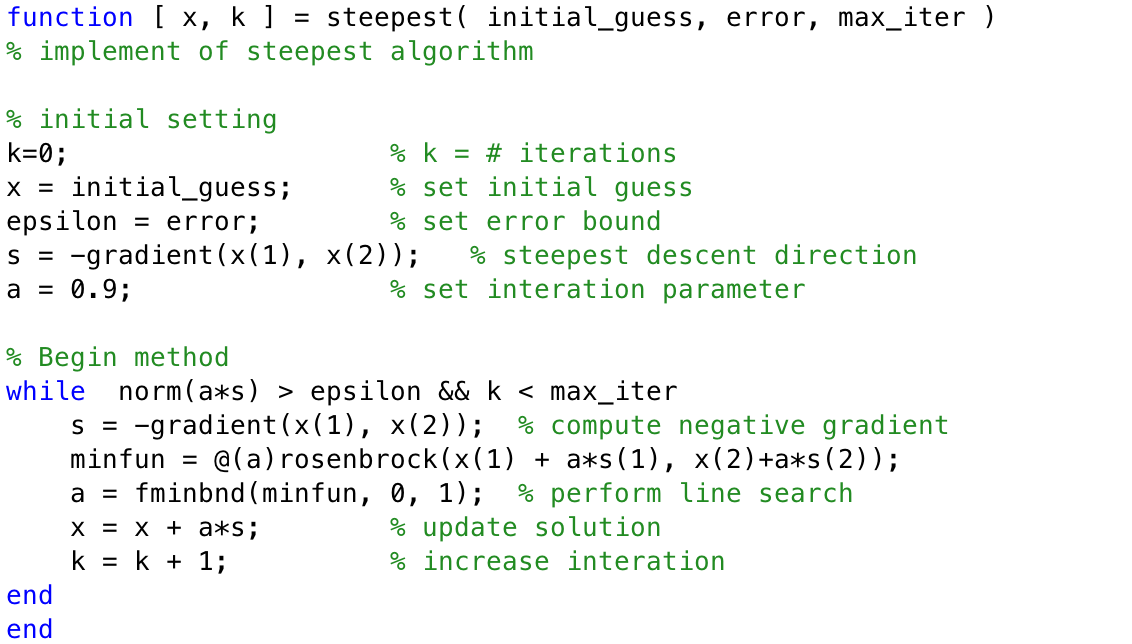
Code 5 code in pj5.m for executing np algorithms

(a) Steepest Descent

main idea of Steepest Descent method: For given function , taking as the direction for such a line search can take maximum possible benefit from movement in any downhill direction would be to attain the minimum of the objective function along that direction.

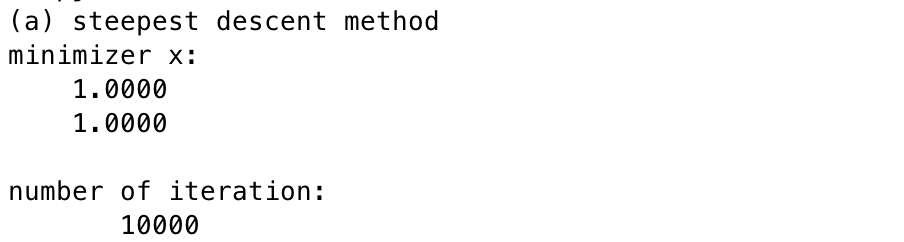
steepest function takes initial\_guess, error bound and maximum number of iteration as input, and return x which is the approximated minimizer and k which is the number of iteration satisfying given error bound. The gradient function and the rosenbrock function are that I declare before in other function file.

In steepest function, I used built-in function fminbnd which can find minimizer of given input function in given input interval for one variable. So, it is line search.

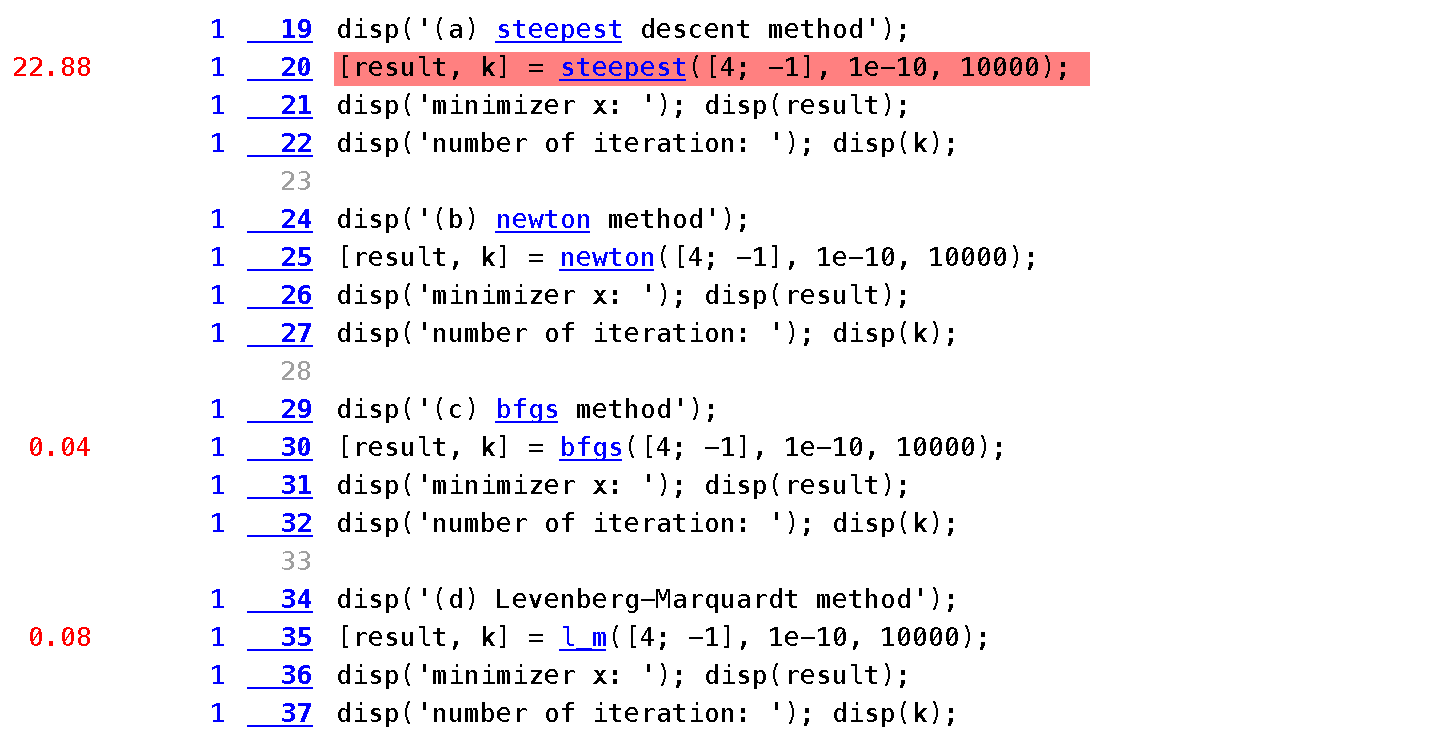


Code 6 steepest.m

This is the result of execution of steepest function in pj4.m code with input initial guess [4; -1], error bound 1e-10, and maximum number of iteration = 10000.



Result 1 steepest([4; -1], 1e-10, 10000)



Result 2 Profile for overall execution time

In profile, the steepest function takes significantly long time compare with other functions.

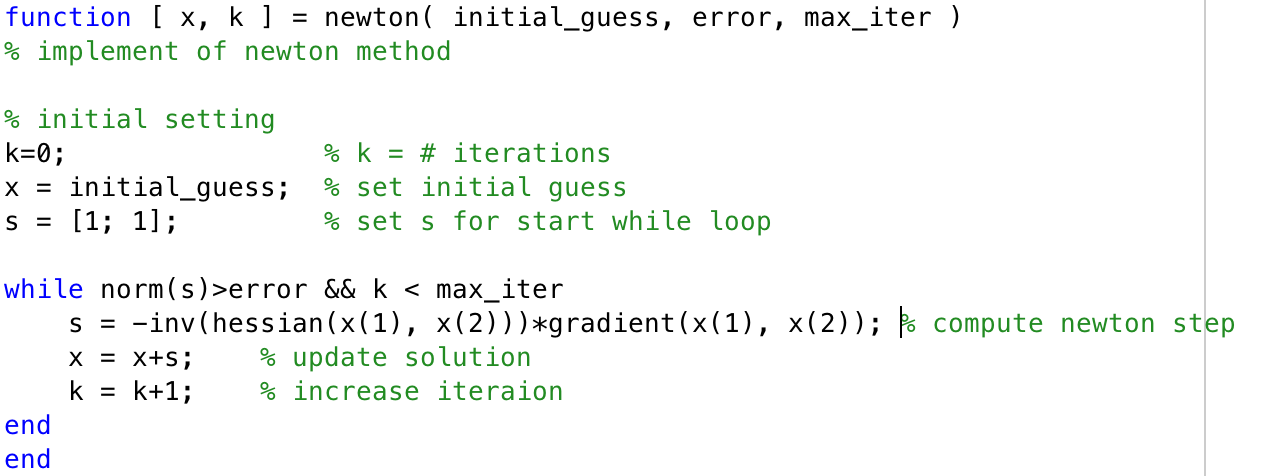


Result 3 Profile for steepest function

By the profile, the built-in function ‘fminbnd’ takes most of execution time for steepest function.

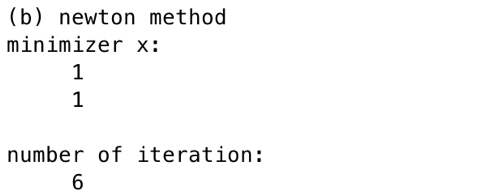
(b) Newton Method

main idea of Newton method: A border of the given function can be gained from a local quadratic approximation, which can be obtained from the truncated Taylor series expansion where is the Hessian matrix of second partial derivatives of , . The hessian matrix is the Jacobian matrix of the gradient, so this approach is equivalent to Newton’s method for solving the nonlinear system . So the code of this function is simple, and I used built-in function ‘inv’ which compute inverse of given matrix. Similar to steepest function, it takes three inputs; initial\_guess, error, max\_iter.



Code 7 newton.m

The result of execution is also depends on the initial guess and error bound, but the number of iteration until satisfying the error bound in newton method is significantly less than that in steepest method.

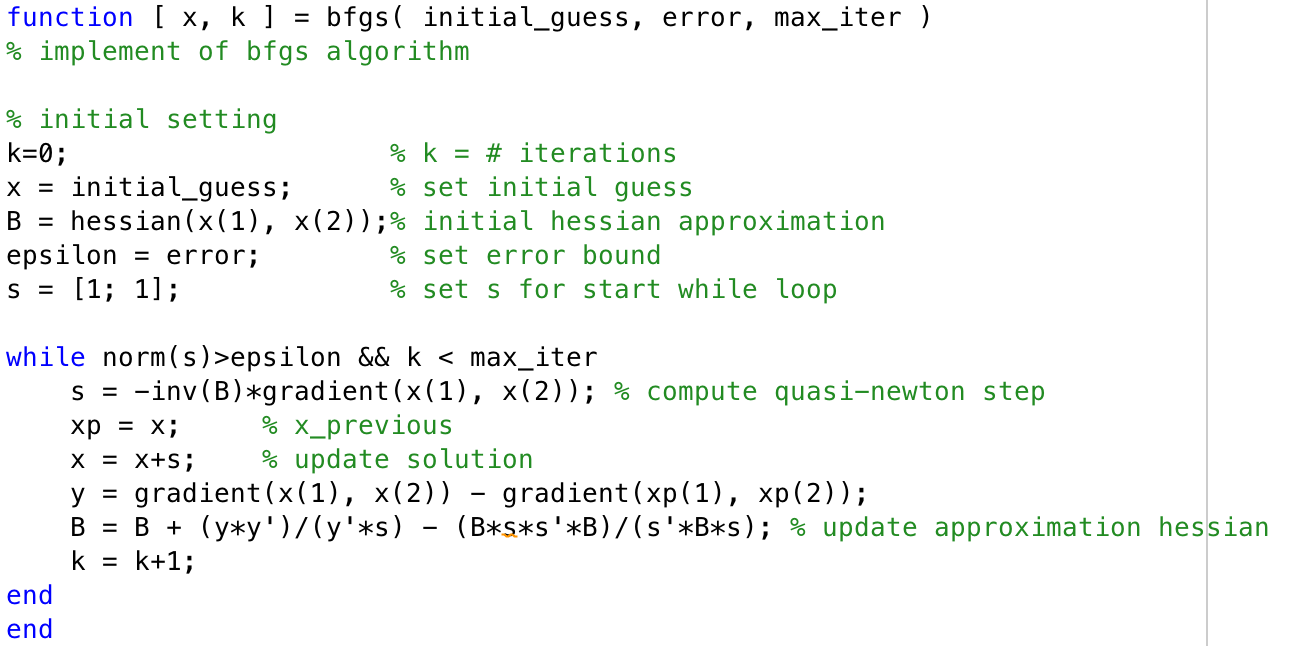


Result 4 newton([4; -1], 1e-10, 10000)

(c) BFGS

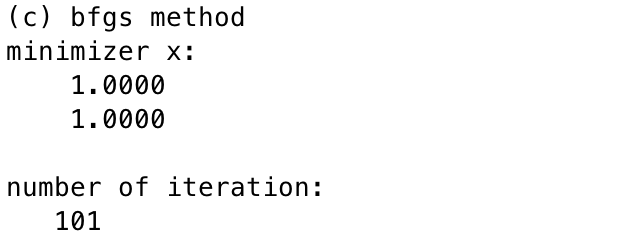
main idea of BFGS method: As with secant updating methods for solving nonlinear equations, the motivation for secant updating methods for optimization is to reduce the work per iteration of Newton’s method and possibly improve its robustness. One of the most effective of these secant updating methods for optimizations is BFGS. In each iterations, a factorization of is updated rather than itself, so that the linear system for the quasi-Newton step can be solved at a cost per iteration of rather than operations.

It needs no second derivatives unlike Newton’s method for optimization, and the second derivative information is gradually built up in the approximate hessian matrix by updating over successive iterations.



Code 8 bfgs.m

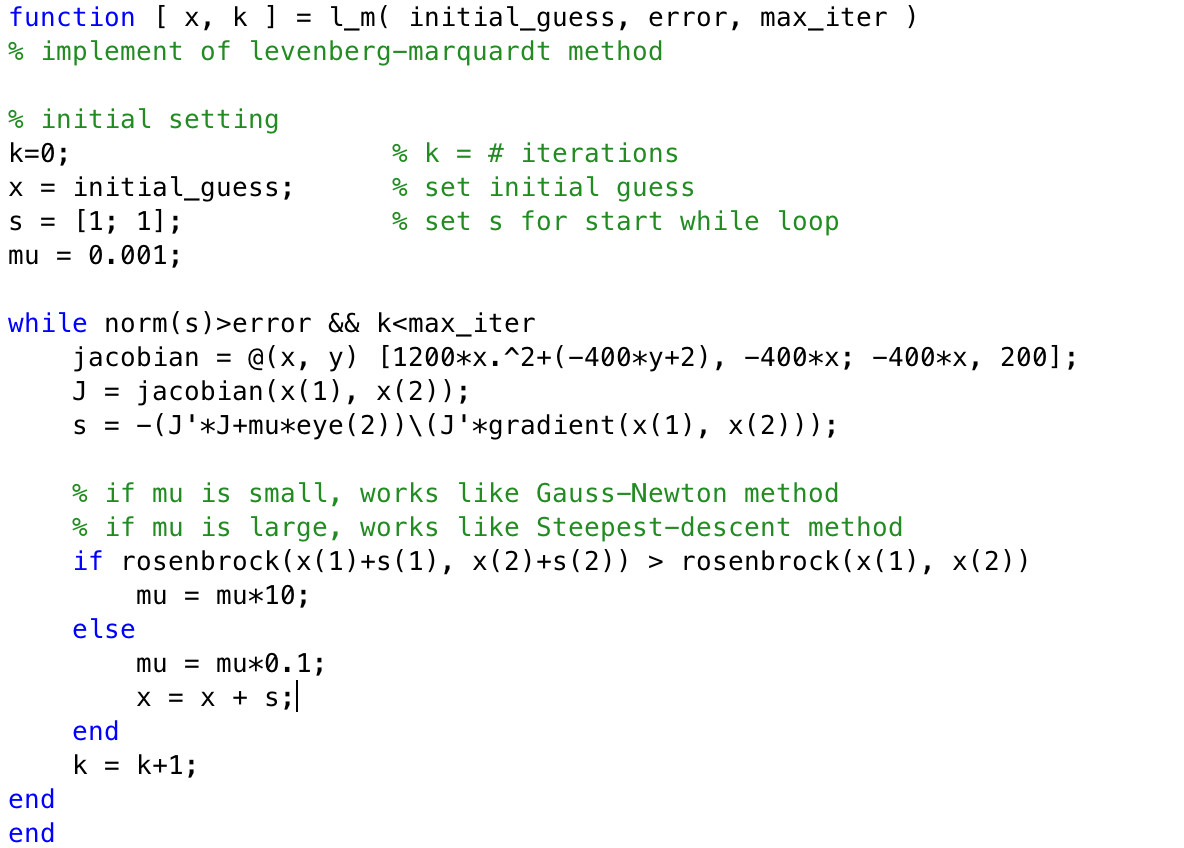
BFGS method executed 101 time iterations. Like most secant updating methods, BFGS normally has a superlinear convergence rate, even though the approximate hessian does not necessarily converge to the true hessian. A line search can also be used to enhance the effectiveness of the method.



Result 5 bfgs([4; -1], 1e-10, 10000)

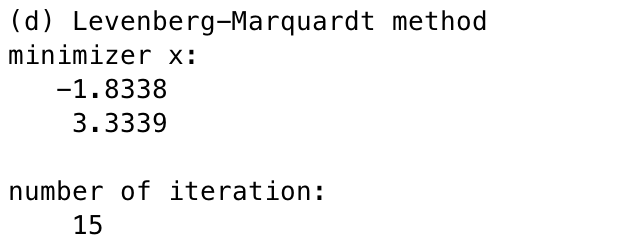
(d) Levenberg-Marquardt

main idea of Levenberg-Marquardt method: Steepest-descent + Gauss-Newton method. If the approximated solution is far from real solution, it works in the way of steepest descent, and if close to real solution works in the way of Gauss-Newton method. So this method is faster than Steepest Method and more likely to solve than Gauss-Newton method. This method is a useful alternative when the Gauss-Newton approximation yields an ill-conditioned or rank-deficient linear least squares subproblem. At each iteration of this method, the linear system for the step is of the form where is a nonnegative scalar parameter chosen by some strategy. The corresponding linear least squares problem to be solved is .



Code 9 l\_m.m

As the result of levenberg-marquardt method, the number of iteration to satisfy error bound is smaller than steepest descent method and bfgs method.



Result 6 l\_m([4; -1], 1e-10, 10000)