Numerical optimization – Practice exercises

Zs. Németh and L. Lócsi

May 13, 2022

Exercises worth 1 point each. Everyone is assigned 4 exercises from each parts (max. 8 points total). Fractional scores are awarded for partial solutions.
A minimum of 4 points is required to pass. The practice mark is based on the overall score:
4 points = passing(2); 5 points = average(3); 6 points = good(4) and $7-8$ points = excellent(5).
Fractional totals are rounded using the floor function.

Part I

1. Suppose that the units of the plane $\mathbb{R}^2$ represent kilometers. From our country house at $(0,0)$, we want to go to the mart at point $(10,10)$ taking minimal time. There are three kinds of terrain on the way. The area

$$F = \{(x,y) : 2 < x < 5, y \in \mathbb{R}\}$$

is covered by forest, where we can move at a reduced speed of 3 km/h. Then,

$$S = \{(x,y) : 5 \leq x \leq 7, y \in \mathbb{R}\}$$

is a swamp, where passage is only possible at 1 km/h. All other terrain is neutral grassy area N, where we can proceed at 5 km/h.
Formalize the underlying optimization problem, i.e. introduce variables and constraints, explain what they represent and give a suitable objective function (i.e. the time we take) to minimize. Which known algorithm would you recommend to solve the problem with? Give the answer to this problem rounded to minute precision.

2. We want to place $n = 3$ charged particles along a circle centered at the origin. Suppose that the particles carry $c_1 = 1$, $c_2 = 2$ and $c_3 = 3$ units of charge, respectively. The particles shall be placed in such a way that the total potential of the system is minimized.
Formalize this optimization problem. Which known algorithm would you recommend to solve the problem with? Demonstrate an approximate solution to the problem.

3. In Epidemistan the new Tiara Bacteria is claiming its victims, and scientists have found out that the most effective way to prevent the disease: to have the input of vitamins F, G, H and I for the inhabitants of the country as balanced as possible. Now these vitamins are present in three fruits of the country: abbles, balalas and caranbolas. One pound of abbles contain no vitamin F, but 1 unit of vitamin G and I, and 5 units of vitamin H. One pound of balalas contain 4 units of vitamin F, 2 units of vitamin G and I, but no vitamin H. And one pound

of caranbolas contain 1 units of both vitamins F and H, and 3 units of vitamins G and I. You can mix one pound of fruit cocktail from these ingredients. What amount of each fruit should be present in the cocktail in order to minimize the sum of the squares of differences between each pair of vitamins?
Formalize the question as an unconstrained optimization problem. Which algorithm would you recommend to solve the problem with?

4. Implement a program for the visualization of the Armijo line search method in case of $\mathbb{R} \to \mathbb{R}$ functions. Show the effect of the parameters c and ϱ, i.e. plot the objective function, the separator line corresponding to c and the points examined by the algorithm. Find some nice test functions and configurations for demonstration.

5. Modify the implementation of the Armijo Gradient descent method, such that the plots also include the further points examined by the line search method underneath, and also the examined line should be visualised at each iteration.

6. Implement a program that plots a given $\mathbb{R}^2 \to \mathbb{R}$ function as a 3D surface, and its second order approximation (Taylor-polynomials) at a given point, step-by-step in each step of Newton's method (without any line search and the positive definite correction, i.e. based on `Newton_noLS.m`).

7. * Implement the unimodal Fibonacci line search algorithm, an improvement over the zero-order unimodal line search algorithm (`zero_unimodal.m`).

8. Implement two first-order unimodal line search algorithms of your choice (e.g. the bisection method and the secant method are possible choices).

9. Implement a non-unimodal line search method, which terminates when a point satisfying the Wolfe conditions was found. You may use the code for the Armijo conditions (`Armijo_LS.m`) as reference.

10. Modify the gradient descent algorithm (`grad_descent.m`) to closely approximate a local minimizer along the search direction at each iteration (instead of a point satisfying "only" the Armijo conditions). You may assume that the objective is differentiable any number of times.

11. * Implement a system of linear equations solver to iteratively approximate the solution of $Ax = b$, ($A \in \mathbb{R}^{n\times n}$ invertible but NOT necessarily positive definite, $b \in \mathbb{R}^n$) using the gradient descent algorithm (`grad_descent.m`) on a suitable objective. If possible, also try to avoid using line search, by moving to the exact directional minimum at each iteration. For partial score, the method for positive definite A can be implemented.

12. Implement a modified multivariable Newton method (based on `Newton.m`), which corrects the non positive definite Hessian by computing its LDL^T decomposition and changing negative and very small diagonal entries to some fixed positive δ value.

13. Implement the Polak–Ribiere conjugate gradient method. You may use the code for the Fletcher–Reeves conjugate gradient algorithm (`FR_conj_grad.m`) as reference. Use a first-order unimodal line search method to find the directional minimizer at each step.

14. * Implement a system of linear equations solver to iteratively compute the solution of $Ax = b$, ($A \in \mathbb{R}^{n \times n}$ symmetric positive definite, $b \in \mathbb{R}^n$) using the conjugate gradient algorithm with a suitable objective. Try to avoid using line search and ensure termination in a finite number of steps. Create a contour plot to visualize the steps of the algorithm, showing the quadratic objective, the search directions and the iteration points.

Part II

1. Implement the Broyden–Fletcher–Goldfarb–Shanno quasi-Newton method using exact line search (i.e. each linesearch terminates in a directional local minimizer). You may use the code for the Davidon–Fletcher–Powell method (`QN_DFP.m`) as reference.

2. Implement the second order finite differencing method for approximating Hessians, then modify the implementation of Newton's method (`Newton.m`) such that it only relies on the objective function's values to approximate both the required gradient and Hessian values.

3. Implement the pattern search method using one of the fixed direction sets D_n discussed on the lecture (e.g. the "coordinate direction set" or the "simplex set"). Use the sufficient decrease function $\rho(t) = 0$. Set some suitable hyperparameter values for γ_{tol}, θ and ϕ at the beginning of your function.

4. * Implement the derivative-free conjugate directions method in the general case (i.e. for arbitrary number of variables). You may implement the algorithm for 3 variables for partial score.

5. * Implement a constrained barrier method with logarithmic barrier. Make it able to explicitly compute the gradients and Hessians of the modified f_ρ objectives based on the exact gradients and Hessians of the g_i contraint functions, given in the input. Use Newton's method as the unconstrained subroutine. You may use the code for the Carrol barrier (`barrier_path.m`) as reference. You may implement the method with approximated Hessians for partial score.

6. Modify our implementation of the Carrol barrier method (`barrier_path.m`) to use a quasi-Newton method of your choice as an unconstrained subroutine. Be careful about not leaving the feasible region.

7. Improve our implementation of the Carrol barrier method (`barrier_path.m`), enabling it to find solutions closer to/right on the boundary of the feasible region: when finite differencing the Hessian for Newton's method, for each e_i canonical direction check if forward differencing would evaluate the gradient outside the feasible region, and if yes, try using a backward difference approximation instead for that column.

8. Extend the penalty method implementation `penalty_path.m`, making it able to solve generally constrained problems (GCP).

9. * Implement a linear least squares solver utilizing QR decomposition. Improve the numerical stability of the method for the case when $r_{i,i} \approx 0$ entries are appearing in the diagonal of the upper-triangle matrix: assume that if $r_{i,i} \approx 0$, then the column q_i is linearly dependent on the previous columns so we can replace q_i by its expansion in terms of $q_1, q_2, \ldots, q_{i-1}$ for columns $a_{i+1}, \ldots, a_n$. Use this idea to remove every critical column. You may use the code for the SVD-based solution (`linear_LSq.m`) as a starting point.

10. Implement a function for fitting a three dimensional paraboloid $z(x, y) = a \cdot x^2 + b \cdot xy + c \cdot y^2 + dx + ey + f$ to a sample of m points $(x_i, y_i, z_i) \in \mathbb{R}^3$, $(i = 1, \ldots, m)$ using linear least squares. Your method should find and return the optimal parameter values for a, b, c, d, e and f. You may use our SVD-based linear least squares solver (`linear_LSq.m`) after contstructing the coefficient matrix and the vector of expected values. Create a 3D plot of the input points and the returned paraboloid.

11. Formalize the problem of fitting a circle to a noisy sample of m points on the $\mathbb{R}^2$ plane (or in $\mathbb{C}$, if you prefer) as a nonlinear least squares problem, and adapt our implementation of the Gauss–Newton algorithm (`Gauss_Newton.m`) to solve it. The input should be the set of sample points, and the output is the center and radius of the resulting circle.

12. Implement a function to visualize the Carrol barrier's effect used in inequality constrained optimization problems for functions of two variables. The input should be the objective function f, the constraints g, and the parameter ϱ. The function should create a 3D plot of the original and the scaled objective functions. Provide some samples with bounded feasible regions (e.g. triangle, rectangle, circle or some other nice region of your choice).

13. Implement a function to visualize the square penalty's effect used in equality constrained optimization problems for functions of two variables. The input should be the objective function f, the constraints g, and the parameter ϱ. The function should create a 3D plot of the original and the modified objective functions. Provide some samples (e.g. optimizing on a line, on a circle or some other nice surface/curve of your choice).

14. Implement a function to visualize the optimization progress of the (derivative free) coordinate descent method while applied to a function of 3 variables. The input should be the objective function and the starting point. A 3D plot should be created with the points found in the optimization steps (subsequent points should be connected with line segments). Plotting the function values is not required, visualize only the trajectory of the approximate solution in the $\mathbb{R}^3$ domain. Provide a nice example where the coordinate directions are observable. You may use the code of `coord_descent.m` as a reference.