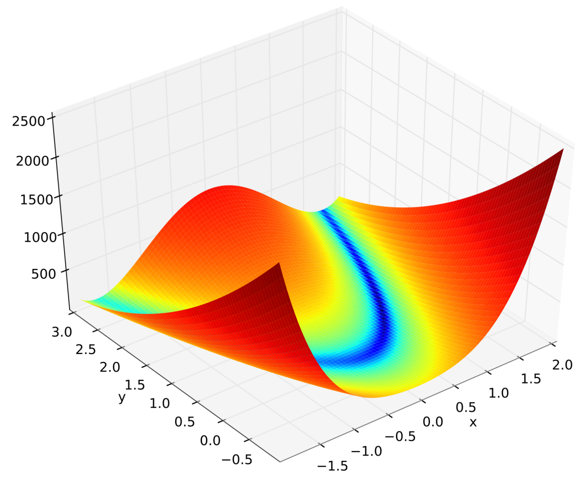
Implementation of Quasi-Newton methods with application to penalty problems



Project in Optimization, FMAN61. Group 24

January 7, 2022.

Faculty of Engineering at Lund University

Simon Danielsson	Erik Hu	Carl Nauclér
990530-1512	000907-4717	990606-6338
si7660da-s@student.lu.se	er0153hu-s@student.lu.se	carlwnaucler@gmail.com

Introduction

The problem of finding an optimal value for a suitable mathematical function arises in many disciplines: maximizing the utility function in micro economics, minimizing energy of a physical system or minimizing some loss function in machine learning to obtain the best possible predictive model. Generally, the objective function is not sufficiently regular for one to algebraically find an exact optimum. One then has to resort to iterative, numerical methods to find the solution to the optimization problem.

In the following, we implement and analyze the behaviour of two *Quasi-Newton methods* for optimization, namely the *Davidon-Fletcher-Powell* (DFP) and *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) algorithms.

Division of work

The majority of the coding was conducted with the whole group in order to be able to discuss and solve any obstacles or problems. The testing was distributed between all members since that phase is easily parallelized. In the final part of the project, namely the writing of the report, each group member wrote the part in which they had worked the most with during the coding phase. Nevertheless this could have been done differently as the background and all subsequent findings are familiar to everyone in the group.

Background

Before diving into the implementation and results of the algorithms we first need some theoretical background about the methods used. This is presented in the following.

Multi-dimensional search

Netwon's method takes a twice differentiable function ***f(x)*** and approximates the function near x_k by Taylor's formula as $q(x) = f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$. $H(x_k)$ denotes the Hessian of f at x_k. If the Hessian is positive definite, q has a minimum where

$\nabla q(x) \ = 0$, if $q(x)$ is defined everywhere. By differentiation of q we obtain $\nabla q(x) \ = 0 \Leftrightarrow$ $x_{k+1} = x_k - H(x_k)^{-1} \nabla f(x_k)$. This formula is used for finding the next iteration step, and Newton's method implies using $d_k = - H(x_k)^{-1} \nabla f(x_k)$ as the search direction for a unit step to find the minimum of the function along the line search.

In order for Newton's method to work the Hessian of the function must be invertible. For cases when the objective function does not have an invertible Hessian or when the Hessian requires a lot of computer power to be computed, Quasi-Newton methods can be utilized. The general idea of all Quasi-Newton methods is to replace the inverse Hessian with an approximation, using only first order derivatives and no matrix inversion (i.e. solving a system of equations). The general iteration step of the method is thus $x_{k+1} = x_k - \lambda_k D_k \nabla f(x_k)$. The general algorithm is as follows:

```
$For\ k := 1\ until\ termination\ do$
$y_1 := x_k$
$D_1 := I$
$For\ j := 1\ to\ n\ do\ (inner\ loop)$
        $d_j := - D_j \nabla f(y_j)$
        $Let\ \lambda_j\ be\ the\ solution\ to\ the\ one\ dimensional\ line\ search\ problem$
                $min\ f(y_j + \lambda d_j)$
        $Define\ y_{j+1} := y_j + \lambda d_j$
        $Update\ D_j\ according\ to\ formulas\ below$
$Let\ x_{k+1} := y_{n+1}$
$Decide\ whether\ to\ terminate\ or\ continue$
```

The distinction between the used methods DFP and BFGS is based on how D_j is updated in the inner loop of the method. DFP uses the following algorithm:

$$p_j := \lambda_j d_j \equiv y_{j+1} - y_j$$

$$q_j := \nabla f(y_{j+1}) - \nabla f(y_j)$$

$$D_{j+1} := D_j + \frac{1}{p_j^T q_j} p_j p_j^T - \frac{1}{q_j^T D_j q_j} D_j q_j q_j^T D_j.$$

BFGS instead utilizes the following algorithm to update D_j, which is known to perform even better than the DFP method in practice.

$$D_{j+1} := D_j + \left(1 + \frac{q_j^T D_j q_j}{p_j^T q_j}\right) \frac{1}{p_j^T q_j} p_j p_j^T - \frac{1}{q_j^T D_j q_j} \left(p_j q_j^T D_j + D_j q_j p_j^T\right).$$

Line search methods

When choosing between different line search methods there are a lot of different factors one might want to take into consideration, and a lot of different methods to choose from. Some of the more basic line search methods function without the use of derivatives. These include the uniform, dichotomous, golden section and fibonacci search. All of these find the minimum of a function *F* by constructing intervals of uncertainty that decrease in size for each function evaluation until an approximation of the optimal point has been found. While all methods of this nature are mainly used when the objective function *F* is not differentiable or simply too complicated to differentiate, some might still be preferred over others. Fibonacci search and golden section search are for example more effective in terms of number of function evaluations needed, in comparison to the other two methods.

If the objective function is differentiable, one should for the same reason opt for line search methods that use derivatives, as these are generally even more effective at reaching a certain level of accuracy. The bisection method for example halves the interval of uncertainty for each new iteration step. In the case of the objective function being twice differentiable, Newton's method can be used. Unlike all the previously mentioned methods, Newton's method does not use intervals of uncertainty and does not converge linearly. Instead, it iteratively minimizes a quadratic approximation of the objective function, and picks the minimum point of the approximation as the next point in the sequence. Thus, it constructs a "sequence of numbers $\{x_k\}$

which is expected to converge to the minimum point" (Böiers, 2015), which allows it to converge quadratically to the minimum of the objective function. On the other hand, this all comes at the cost of not guaranteeing a convergence to the global minimum, since a prerequisite for Newton's method to work well is to have chosen an initial point close enough to the minimum point. Also, in its pure form it requires the Hessian matrix of the objective function to be positive definite at every x_k in order for the extreme point to indeed be a minimum point. Alternate methods also include Armijo's rule which is an inexact line search method that produces an acceptable interval for the parameter λ in the line search method. It produces a coarse approximation of a minimum point in each step and converges slower in comparison to Newton's method, although it requires less functional evaluations and thus computer power.

Evidently, a (positive-definite) quadratic function will be minimized exactly in one iteration using Newton's method (since it is indeed equal to its second order Taylor expansion). This is not necessarily the case for Quasi-Newton methods since the inverse Hessian is only approximated.

Newton's method was chosen as it is the fastest of the examined methods in most cases (quadratic convergence for sufficiently regular functions), although it requires more computational power. If the examined objective functions had been far more expensive to evaluate using Newton's method (due to complex derivatives), Armijo's rule could have been used instead. As the computational time using Newton's method was comparable to that of the preliminary presented material, it was deemed sufficient for its purpose.

Stopping conditions

The choice of a suitable termination criterion is an important part of optimization algorithms, since it, if done correctly, allows for a good balance between effectiveness and precision of the algorithm. On the one end, the termination criterion needs to make sure computational resources are not wasted by preventing unnecessary iterations. On the other, it has to let the algorithm run for long enough so that a good estimate of the optimal point or value is found. In our specific implementation the termination criterion in the line search is based on the absolute value of the change in function value, whereas the one in the multidimensional search is based on the relative change in function value. When the change/relative change in function value is small in the latest iteration, it might indicate that the sequence has reached a stationary point. However, when the algorithm is applied to an objective function with a slow growth rate, it will be interpreted as having stationary points everywhere. Hence, our choice of termination criteria works best if the objective function does not have too slow of a growth rate. Furthermore, tests with various criteria were carried out and this is what seemed to work best for our specific algorithm. For example, if the relative function change criterion is applied to our line search, it immediately stops working. Of course, this can probably be quite easily fixed. We simply chose to pick the criteria based on our algorithm instead of building/adapting the algorithm based on a criterion.

Termination criterion for line search: $\left|f(x_{k+N}) - f(x_k)\right| < \varepsilon$

Termination criterion for multidimensional search: $\frac{\left|f(x_{k+1}) - f(x_k)\right|}{\left|f(x_k)\right|} < \varepsilon$

Results

Exact optimal values for each problem

Table 1 summarizes the optimal points and corresponding function values for the objective functions on which tests are conducted. The dimension *n* of the objective function's domain is positive integer.

Table 1		
Objective function	**Optimal point**	**Optimal function value**
Positive-definite quadratic form	(0, 0, …, 0)	0
Negative-definite quadratic	Does not exist	Does not exist

form		
Indefinite quadratic form	Does not exist	Does not exist
Rosenbrock function	(1, 1)	0
Booth function $f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$	(1, 3)	0
Styblinski-Tang $f(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i}{2}$	≈ (-2.904, -2.904, …, -2.904)	$-39.16617n < f(\mathbf{x}) < -39.16616n$

Consistency in the program behaviour

In order to test the consistency in the program behaviour, the optimization algorithm is tested on a set of functions for which information about its (potentially) existing optimum is already known. This is presented in the following.

Quadratic forms

First, the program is tested on positive-definite, negative-definite and indefinite quadratic forms for which only the positive-definite forms are convex and thus have a (unique) minimum. Such forms are easily constructed since all hermitian matrices are congruent to some diagonal matrix (corresponding to the same quadratic form but in another basis). Some of the results of the tests on positive definite quadratic forms are found in Table 2, using a tolerance of 1e-6 and a simple starting point. Other starting points and Hermitian matrices were used in subsequent trials, and still the two methods yielded essentially the same point of convergence in exactly the same number of iterations. Thus these results have been omitted from the tables. (The interested reader can download the source code and run it him/herself to verify this).

Table 2
Minimizing positive definite quadratic form, min at origin
Starting at point [1, 2],

Method

DFP

```
outer it.   iteration      x       step size      f(x)     norm(grad)  ls iters   lambda
   init                +1.00e+00               2.30e+01
                       +2.00e+00

    1           1      +3.80e-01   2.16e+00    4.55e-01    2.09e+01       3      1.03e-01
                       -6.83e-02

                2      +7.12e-10   3.86e-01    2.97e-17    2.38e+00       3      1.63e-01
                       +2.37e-09

    2           1      +7.12e-10   0.00e+00    2.97e-17    2.41e-08       1      0.00e+00
                       +2.37e-09


x =

   1.0e-08 *

    0.0712
    0.2372


no_its =

     3


normg =

   2.4101e-08
```

BFGS

```
outer it.   iteration      x       step size      f(x)     norm(grad)  ls iters   lambda
   init                +1.00e+00               2.30e+01
                       +2.00e+00

    1           1      +3.80e-01   2.16e+00    4.55e-01    2.09e+01       3      1.03e-01
                       -6.83e-02

                2      +7.12e-10   3.86e-01    2.96e-17    2.38e+00       3      1.61e-01
                       +2.37e-09

    2           1      +7.12e-10   0.00e+00    2.96e-17    2.41e-08       1      0.00e+00
                       +2.37e-09


x =

   1.0e-08 *

    0.0712
    0.2372


no_its =

     3


normg =

   2.4100e-08
```

Proceeding, negative definite and indefinite quadratic forms are minimized using the program. Here, the program throws an error. Table 3 displays such a situation when trying to minimize a negative definite quadratic form using the DFP method: the situation is analogous when using BFGS and other initial points (and have thus been omitted here).

Table 3

```
Minimizing negative definite quadratic form, has no minimum
Executing minimization of function @(x)x'*H*x
Starting at point [5, 9],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init               +5.00e+00               -1.06e+02
                      +9.00e+00

[Warning: The function does not seem to be convex: cannot be
minimized.]
```

As expected, the program is only able to minimize the positive-definite quadratic forms. There seems to be no trouble minimizing positive-definite quadratic forms whose Hessian matrix has vastly different magnitudes of its eigenvalues. Also, the fact that the number of iterations required for the minimum to be found exceeds one is expected, since only Newton methods minimize quadratic functions exactly in one iteration. This is not the case for Quasi-Newton methods where we are not using the inverse Hessian of the objective function to determine the subsequent point in the sequence. Also, when optimizing simple functions such as quadratic forms, we are not able to detect any difference in performance between the two methods. We need to test the methods on some other functions in order to see a significant difference.

Less regular functions

Proceeding, the program is tested on less regular functions. In Table 4 the results of the tests are found for the Rosenbrock function using initial point (200, 200) and tolerance 1e-6. Evidently the program is able to minimize the Rosenbrock function for this initial point: in fact it seems to converge for initial points a lot further away from the origin, such as (3990, -7111) (see Appendix 1). For the initial point [200, 200] we require more iterations to converge using BFGS than DFP. In contrast, for the initial points (399, -711) and (3990, -7111) the required number of iterations was in fact the same (and equal to 9), see Appendix 1 for full details. Thus the different methods seem to perform similarly more often than not.

For the Booth function, the program converges to the true minimum for all the tested initial points and using both methods. In Table 5 one finds the output of the program for the optimization problem using DFP and starting point (1139, 9991). It does so in very few iterations. The result is identical for the BFGS method and other initial points (see Appendix 1 for further details).

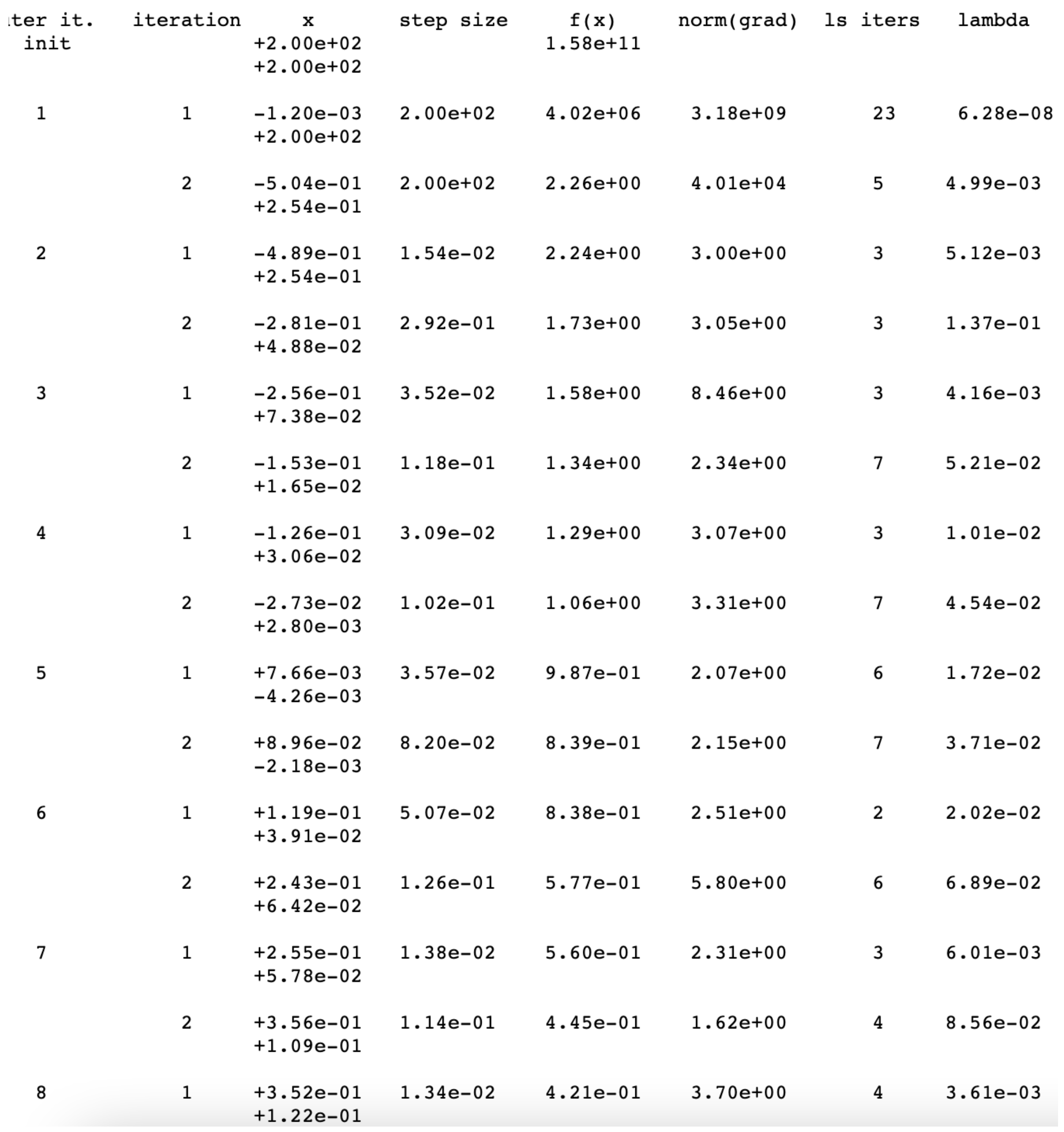
Table 5
Initial point [1139, 9991]
Using DFP method
Executing minimization of Booth function, min at (1, 3)

```
outer it.   iteration      x        step size     f(x)     norm(grad)  ls iters   lambda
   init                +1.14e+03              5.96e+08
                       +9.99e+03

    1           1      -3.97e+03   7.95e+03    3.09e+07    1.42e+05       3      5.59e-02
                       +3.89e+03

                2      +1.04e+00   5.56e+03    3.28e-02    1.12e+04       3      5.00e-01
                       +3.05e+00

    2           1      +9.97e-01   6.04e-02    2.25e-05    1.09e+00       3      5.56e-02
                       +3.00e+00

                2      +9.97e-01   0.00e+00    2.25e-05    9.50e-03       1      0.00e+00
                       +3.00e+00

    3           1      +9.97e-01   0.00e+00    2.25e-05    9.50e-03       1      0.00e+00
                       +3.00e+00


x =

    0.9966
    3.0034


no_its =

     5


normg =

    0.0095
```

Table 4
Minimizing Rosenbrock function, min at (1, 1)
Starting at point [200, 200],

Method

DFP



```
ter it.   iteration      x        step size     f(x)      norm(grad)  ls iters    lambda
 init                +2.00e+02              1.58e+11
                     +2.00e+02

  1           1      -1.20e-03   2.00e+02   4.02e+06    3.18e+09       23      6.28e-08
                     +2.00e+02

              2      -5.04e-01   2.00e+02   2.26e+00    4.01e+04       5     4.99e-03
                     +2.54e-01

  2           1      -4.89e-01   1.54e-02   2.24e+00    3.00e+00       3     5.12e-03
                     +2.54e-01

              2      -2.81e-01   2.92e-01   1.73e+00    3.05e+00       3     1.37e-01
                     +4.88e-02

  3           1      -2.56e-01   3.52e-02   1.58e+00    8.46e+00       3     4.16e-03
                     +7.38e-02

              2      -1.53e-01   1.18e-01   1.34e+00    2.34e+00       7     5.21e-02
                     +1.65e-02

  4           1      -1.26e-01   3.09e-02   1.29e+00    3.07e+00       3     1.01e-02
                     +3.06e-02

              2      -2.73e-02   1.02e-01   1.06e+00    3.31e+00       7     4.54e-02
                     +2.80e-03

  5           1      +7.66e-03   3.57e-02   9.87e-01    2.07e+00       6     1.72e-02
                     -4.26e-03

              2      +8.96e-02   8.20e-02   8.39e-01    2.15e+00       7     3.71e-02
                     -2.18e-03

  6           1      +1.19e-01   5.07e-02   8.38e-01    2.51e+00       2     2.02e-02
                     +3.91e-02

              2      +2.43e-01   1.26e-01   5.77e-01    5.80e+00       6     6.89e-02
                     +6.42e-02

  7           1      +2.55e-01   1.38e-02   5.60e-01    2.31e+00       3     6.01e-03
                     +5.78e-02

              2      +3.56e-01   1.14e-01   4.45e-01    1.62e+00       4     8.56e-02
                     +1.09e-01

  8           1      +3.52e-01   1.34e-02   4.21e-01    3.70e+00       4     3.61e-03
                     +1.22e-01
```

```
9          1     +4.84e-01   1.25e-02    2.67e-01    4.79e+00       3      2.61e-03
                 +2.32e-01

           2     +6.13e-01   1.82e-01    1.76e-01    7.50e-01       5      2.45e-01
                 +3.60e-01

10          1     +6.07e-01   9.24e-03    1.55e-01    4.57e+00       3      2.02e-03
                 +3.66e-01

           2     +7.23e-01   1.84e-01    9.39e-02    5.03e-01       6      3.69e-01
                 +5.09e-01

11          1     +7.17e-01   6.75e-03    8.00e-02    4.15e+00       3      1.63e-03
                 +5.14e-01

           2     +8.18e-01   1.78e-01    4.26e-02    3.24e-01       6      5.50e-01
                 +6.60e-01

12          1     +8.14e-01   4.71e-03    3.45e-02    3.46e+00       3      1.36e-03
                 +6.62e-01

           2     +8.98e-01   1.61e-01    1.49e-02    1.94e-01       5      8.27e-01
                 +8.00e-01

13          1     +8.95e-01   3.06e-03    1.10e-02    2.58e+00       3      1.18e-03
                 +8.01e-01

           2     +9.58e-01   1.30e-01    3.17e-03    1.02e-01       4      1.27e+00
                 +9.14e-01

14          1     +9.57e-01   1.67e-03    1.87e-03    1.56e+00       3      1.07e-03
                 +9.15e-01

           2     +9.57e-01   0.00e+00    1.87e-03    4.00e-02       1      0.00e+00
                 +9.15e-01

15          1     +9.57e-01   0.00e+00    1.87e-03    4.00e-02       1      0.00e+00
                 +9.15e-01

x =

    0.9568
    0.9153


no_its =

    29


normg =

    0.0400
```

```
outer it.   iteration      x       step size      f(x)     norm(grad)  ls iters    lambda
  init                 +2.00e+02               1.58e+11
                       +2.00e+02

    1           1      -1.20e-03   2.00e+02    4.02e+06    3.18e+09      23      6.28e-08
                       +2.00e+02

                2      -5.04e-01   2.00e+02    2.26e+00    4.01e+04       5     4.99e-03
                       +2.54e-01

    2           1      -4.89e-01   1.54e-02    2.24e+00    3.00e+00       3     5.12e-03
                       +2.54e-01

                2      -4.13e-01   1.06e-01    2.00e+00    3.05e+00       9     2.44e-02
                       +1.79e-01

    3           1      -3.50e-01   1.02e-01    1.88e+00    2.23e+00       4     4.59e-02
                       +9.89e-02

                2      -2.17e-01   1.64e-01    1.67e+00    7.62e+00       5     6.05e-03
                       +3.67e-03

    4           1      -1.89e-01   4.76e-02    1.42e+00    1.06e+01       3     4.48e-03
                       +4.24e-02

                2      -1.08e-01   8.87e-02    1.23e+00    2.30e+00       7     3.77e-02
                       +7.26e-03

    5           1      -5.65e-02   5.42e-02    1.17e+00    2.55e+00       5     2.12e-02
                       +2.56e-02

                2      -5.65e-02   0.00e+00    1.17e+00    4.76e+00       1     0.00e+00
                       +2.56e-02

    6           1      -5.63e-02   5.57e-04    1.16e+00    4.76e+00       3     1.17e-04
                       +2.50e-02

                2      +1.70e-02   7.40e-02    9.88e-01    4.66e+00       9     2.72e-02
                       +1.51e-02

    7           1      +3.14e-02   2.51e-02    9.42e-01    3.62e+00       3     6.94e-03
                       -5.48e-03

                2      +8.13e-02   5.00e-02    8.53e-01    2.26e+00       7     1.87e-02
                       -3.01e-03
```

BFGS

8	1	+9.80e-02 +1.81e-02	2.69e-02	8.21e-01	2.45e+00	3	1.10e-02	
	2	+1.13e-01 +2.06e-02	1.51e-02	7.93e-01	2.72e+00	6	3.72e-03	
9	1	+1.30e-01 +8.12e-03	2.10e-02	7.65e-01	2.64e+00	3	7.96e-03	
	2	+1.75e-01 +1.96e-02	4.66e-02	6.93e-01	2.17e+00	7	1.66e-02	
10	1	+1.82e-01 +3.69e-02	1.86e-02	6.71e-01	2.38e+00	4	7.83e-03	
	2	+2.33e-01 +5.45e-02	5.36e-02	5.89e-01	2.06e+00	7	1.97e-02	
11	1	+2.57e-01 +5.30e-02	2.44e-02	5.69e-01	1.58e+00	3	1.54e-02	
	2	+2.82e-01 +6.63e-02	2.83e-02	5.33e-01	2.60e+00	6	5.71e-03	
12	1	+2.82e-01 +7.92e-02	1.29e-02	5.16e-01	2.63e+00	3	4.90e-03	
	2	+3.83e-01 +1.37e-01	1.17e-01	3.91e-01	1.42e+00	5	7.23e-02	
13	1	+3.82e-01 +1.45e-01	8.47e-03	3.82e-01	2.12e+00	3	4.01e-03	
	2	+5.16e-01 +2.48e-01	1.69e-01	2.68e-01	1.11e+00	4	1.36e-01	
14	1	+5.09e-01 +2.57e-01	1.14e-02	2.41e-01	4.63e+00	3	2.47e-03	
	2	+6.36e-01 +3.89e-01	1.83e-01	1.57e-01	6.94e-01	6	2.61e-01	
15	1	+6.30e-01 +3.95e-01	8.75e-03	1.37e-01	4.54e+00	3	1.93e-03	
	2	+7.43e-01 +5.40e-01	1.84e-01	8.12e-02	4.62e-01	6	3.96e-01	

```
    16           1     +7.38e-01   6.30e-03    6.86e-02    4.02e+00       3     1.57e-03
                      +5.44e-01

               2     +8.36e-01   1.75e-01    3.53e-02    2.94e-01       6     5.94e-01
                      +6.89e-01

    17           1     +8.32e-01   4.34e-03    2.82e-02    3.29e+00       3     1.32e-03
                      +6.92e-01

               2     +9.12e-01   1.55e-01    1.15e-02    1.73e-01       5     8.96e-01
                      +8.25e-01

    18           1     +9.09e-01   2.74e-03    8.23e-03    2.37e+00       3     1.16e-03
                      +8.27e-01

               2     +9.67e-01   1.20e-01    2.09e-03    8.74e-02       4     1.38e+00
                      +9.32e-01

    19           1     +9.66e-01   1.40e-03    1.17e-03    1.33e+00       2     1.05e-03
                      +9.33e-01

               2     +9.66e-01   0.00e+00    1.17e-03    3.14e-02       1     0.00e+00
                      +9.33e-01

    20           1     +9.66e-01   0.00e+00    1.17e-03    3.14e-02       1     0.00e+00
                      +9.33e-01

x =

    0.9659
    0.9327


no_its =

    39


normg =

    0.0314
```

Moving on, the Styblinski-Tang function seems to be a bit more troublesome: see Table 6. Only when the initial point is chosen sufficiently close to the minimum the program seems to find the optimum. Specifically, the method does not converge to the true solution if the initial point has coordinates far away from the minimum (approximately -2.904). It does indeed converge for the initial points (-5, -5), (-3, -3), and (-1.5, -1.5), see Appendix 1. We see in Table 6 that for starting point (-1, -1), the method does indeed converge, but not to the right solution. For some reason it stops after only one iteration. Hence, it seems to converge for all initial points (*a*, *a*) for -5 < *a* < -1, but fails when -1 < a.

Table 6 *Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)* *Using BFGS method*	**Starting point**

```
outer it.   iteration      x        step size      f(x)      norm(grad)  ls iters    lambda
   init                 -5.00e+00                2.00e+02
                        -5.00e+00

     1           1      -2.90e+00   2.96e+00    -7.83e+01    2.37e+02       5      1.25e-02
                        -2.90e+00

                 2      -2.90e+00   2.62e-07    -7.83e+01    2.71e-05       2      2.47e-01
                        -2.90e+00

     2           1      -2.90e+00   7.40e-07    -7.83e+01    2.61e-05       2      2.84e-02
                        -2.90e+00

                 2      -2.90e+00   1.26e-08    -7.83e+01    7.11e-07       2      1.95e-02
                        -2.90e+00


x =

   -2.9035
   -2.9035


no_its =

     4


normg =

   7.1054e-07
```
	(-5, -5)

```
outer it.   iteration      x        step size      f(x)      norm(grad)  ls iters    lambda
   init                 -1.00e+00               -2.00e+01
                        -1.00e+00

     1           1      -1.00e+00   0.00e+00    -2.00e+01    2.33e+01       1      0.00e+00
                        -1.00e+00


x =

    -1
    -1


no_its =

     1


normg =

   23.3345
```
	(-1, -1)

Finally, we want to compare the effect the restart parameter has on the performance and accuracy of the DFP and BFGS methods, respectively. The results are found in Table 7, using starting point (200, 200) and tolerance 1e-6.

Table 7
Minimizing rosenbrock function, min at (1, 1)
Initial point [200, 200]

Method

DFP

```
outer it.   iteration      x        step size     f(x)      norm(grad)  ls iters   lambda
   init                 +2.00e+02               1.58e+11
                        +2.00e+02

    1           1       -1.20e-03   2.00e+02    4.02e+06    3.18e+09      23      6.28e-08
                        +2.00e+02

                2       -5.04e-01   2.00e+02    2.26e+00    4.01e+04      5      4.99e-03
                        +2.54e-01


x =

   -0.5043
    0.2543


no_its =

     2


normg =

   4.0100e+04
```

BFGS

```
outer it.   iteration      x        step size     f(x)      norm(grad)  ls iters   lambda
   init                 +2.00e+02               1.58e+11
                        +2.00e+02

    1           1       -1.20e-03   2.00e+02    4.02e+06    3.18e+09      23      6.28e-08
                        +2.00e+02

                2       -5.04e-01   2.00e+02    2.26e+00    4.01e+04      5      4.99e-03
                        +2.54e-01


x =

   -0.5043
    0.2543


no_its =

     2


normg =

   4.0100e+04
```

We see that the result is identical for the two methods. Obviously, now we only get one outer iteration, which means the total number of iterations performed is only 2. In this few number of iterations we do not precisely reach the true minimum of (1, 1): the point of convergence is about one order of magnitude away. However, this displays the strength of the methods since these calculations were very cheap in comparison to when we let the algorithm restart which required about 20 times more iterations for convergence. Thus, depending on the problem and how close we need to be to the true minimum, not letting the method restart might actually be a suitable alternativ.

To conclude, the performed tests indicate that the program has behaviour consistent with theory as long as the objective function is sufficiently regular. Occasionally, the choice of initial point determines if the method converges to the true minimum. Also, in some cases the program fails completely, for instance when the function values become too large. Furthermore, the two methods seem to behave in a very similar manner in most problems: and occasionally the BFGS method requires more iterations to converge. Further refinement of the implementation would have to be conducted in order to increase the stability and predictability of the program. An example of this could be to further improve the way derivatives are checked and computed such that we always get the desired result.

Constrained optimization using penalty function

Now it is time for the optimization procedure to be tested on a constrained optimization problem. Consider the following minimization problem

$$\text{(pen)} \qquad \text{minimize } e^{x_1x_2x_3x_4x_5} \qquad \text{subject to } \begin{cases} x_1^2+x_2^2+x_3^2+x_4^2+x_5^2=10, \\ x_2x_3=5x_4x_5, \\ x_1^3+x_3^3=-1. \end{cases}$$

This problem is solved by introducing a penalty function α and instead minimizing the auxiliary objective function $q_k(x) = f(x) + \mu_k\alpha(x)$ for some strictly increasing sequence $\{\mu_k\}_k$. Using this, the problem can be solved by iteratively minimizing $q_k(x)$, using the solution to this k'th problem as the starting point of the $(k+1)$'th problem of minimizing $q_{k+1}(x)$. By formulating some suitable stopping condition one obtains a sequence of solutions which hopefully converges to the exact solution. The number of elements in this sequence, i.e. the required number of iterations, we denote by *final*.

The result of the program applied to the (pen) problem is found in Table 8**.** The right-most column x_{final} denotes the point of convergence. Evidently, decreasing the tolerance (for this particular starting point and using the DFP method) requires the program to perform more iterations to converge. Nevertheless, they converge to (essentially) the same solution. According to Table 9 we see that the same is true for the BFGS method.

Using μ_k*= 2k* *With restart* *Using DFP method* *Starting at point [-2, 2, 2, -1, -1],*	**Table 8**		*Using* μ_k*= 2k* *With restart* *Using BFGS method* *Starting at point [-2, 2, 2, -1, -1],*	**Table 9**
$[x_1,\ x_2,\ \ldots,\ x_{final}]$	**tolerance**		$[x_1,\ x_2,\ \ldots,\ x_{final}]$	**tolerance**
-2.0000 -1.7182 2.0000 1.8279 2.0000 1.5972 -1.0000 -0.7666 -1.0000 -0.7666	1		-2.0000 -1.9511 2.0000 1.8288 2.0000 1.6820 -1.0000 -0.8410 -1.0000 -0.8410	1
-2.0000 -1.7182 -1.7177 -1.7175 2.0000 1.8279 1.8276 1.8275 2.0000 1.5972 1.5965 1.5962 -1.0000 -0.7666 -0.7651 -0.7646 -1.0000 -0.7666 -0.7651 -0.7646	1e-3		-2.0000 -1.9511 -1.7177 -1.7175 2.0000 1.8288 1.8276 1.8275 2.0000 1.6820 1.5965 1.5962 -1.0000 -0.8410 -0.7651 -0.7646 -1.0000 -0.8410 -0.7651 -0.7646	1e-3

Altering the starting point yields a similar result: see Tables 10, 11. However, in this case, when decreasing the tolerance the number of iterations required for convergence is increased even more in comparison when the previous starting point was used. This is true for both DFP and BFGS, but most prominent for BFGS. However, we note that the change in the coordinates is very small during the additional interactions performed when the tolerance is decreased to 1e-3 compared to a tolerance of 1.

Using μ_k *= 2k* *With restart* *Using DFP method* *Starting at point [2, -2, -2, 1, 1],*	**Table 10**
$[x_1, x_2, \ldots, x_{final}]$	**tolerance**
$\begin{matrix} 2.0000 & 2.2025 & 2.2025 \\ -2.0000 & -0.0002 & -0.0001 \\ -2.0000 & -2.2691 & -2.2692 \\ 1.0000 & 0.0168 & 0.0092 \\ 1.0000 & 0.0168 & 0.0092 \end{matrix}$	1
$\begin{matrix} 2.0000 & 2.2025 & 2.2025 & 2.2025 & 2.2025 & 2.2025 & 2.2025 \\ -2.0000 & -0.0002 & -0.0001 & -0.0001 & -0.0000 & -0.0000 & -0.0000 \\ -2.0000 & -2.2691 & -2.2692 & -2.2691 & -2.2692 & -2.2692 & -2.2692 \\ 1.0000 & 0.0168 & 0.0092 & 0.0079 & 0.0052 & 0.0041 & 0.0037 \\ 1.0000 & 0.0168 & 0.0092 & 0.0079 & 0.0052 & 0.0041 & 0.0037 \end{matrix}$	1e-3

Using μ_k *= 2k* *With restart* *Using BFGS method* *Starting at point [2, -2, -2, 1, 1],*	**Table 11**
$[x_1, x_2, \ldots, x_{final}]$	**tolerance**
$\begin{matrix} 2.0000 & 2.2024 & 2.2025 \\ -2.0000 & -0.0004 & -0.0001 \\ -2.0000 & -2.2691 & -2.2691 \\ 1.0000 & 0.0174 & 0.0093 \\ 1.0000 & 0.0174 & 0.0093 \end{matrix}$	1
$\begin{matrix} 2.0000 & 2.2024 & 2.2025 & 2.2025 & 2.2025 & 2.2025 & 2.2025 & 2.2025 & 2.2025 \\ -2.0000 & -0.0004 & -0.0001 & -0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0000 \\ -2.0000 & -2.2691 & -2.2691 & -2.2692 & -2.2692 & -2.2692 & -2.2692 & -2.2692 & -2.2692 \\ 1.0000 & 0.0174 & 0.0093 & 0.0055 & 0.0037 & 0.0027 & 0.0017 & 0.0008 & 0.0008 \\ 1.0000 & 0.0174 & 0.0093 & 0.0055 & 0.0037 & 0.0027 & 0.0017 & 0.0008 & 0.0008 \end{matrix}$	1e-3

Now we try some different sequences $\{\mu_k\}_k$ and see how it affects the convergence rate. To more clearly see the result we only display the required number of iterations for convergence, i.e. the number $final$. The required number of iterations for $\mu_k = k$, $\mu_k = 2k$, and $\mu_k = 5k$, $k = 1, 2, \ldots, final$ for problem (pen) with the same starting point as in Tables 10, 11 for tolerance 1e-3 is found in Table 12. Evidently, this indicates that the required number of iterations for convergence is low when using $\mu_k = 2k$. In fact, the sequence $\mu_k = 5k$ seems to be too restrictive: when using the starting point x_0 =(10, -5, 8, -0.5, -2), the algorithm fails to converge due to the extremely large penalty for large *k* (all derivatives become infinite). Thus, in the following, we choose the sequence $\mu_k = 2k$ which seems to be a good middle-ground choice of penalty: too low of a penalty yields slow convergence; too large of a penalty makes the program fail.

With restart *Starting at point [-2, 2, 2, -1, -1],*		**Table 12**
Required number of iterations **DFP**	**Required number of iterations** **BFGS**	μ_k
6	8	*k*
3	3	*2k*
8	5	5*k*

Finally, we compare the required number of iterations for convergence when the multidimensional search is done *without restart*. Thus each subproblem is solved using (at most) *n* iterations, where *n* denotes the number of dimensions of the input vector. Table 13 shows the required number of subproblem iterations which have to be performed in order for the program to converge *with and without restart*. Here we use tolerance equal to 1e-3 and $\mu_k = 2k$. This indicates that the number of subproblems that have to be solved increases significantly when we turn off the restart condition of the multidimensional search. This is expected since the restart parameter increases the accuracy of the solution of each subproblem (at an additional cost) which should in turn require a lower number of required subproblems to be solved.

Starting at point [10, -5, 8, -0.5, -2],		**Table 13**
Required number of iterations DFP	**Required number of iterations BFGS**	**With/without restart**
3	3	Restart
11	32	Without restart

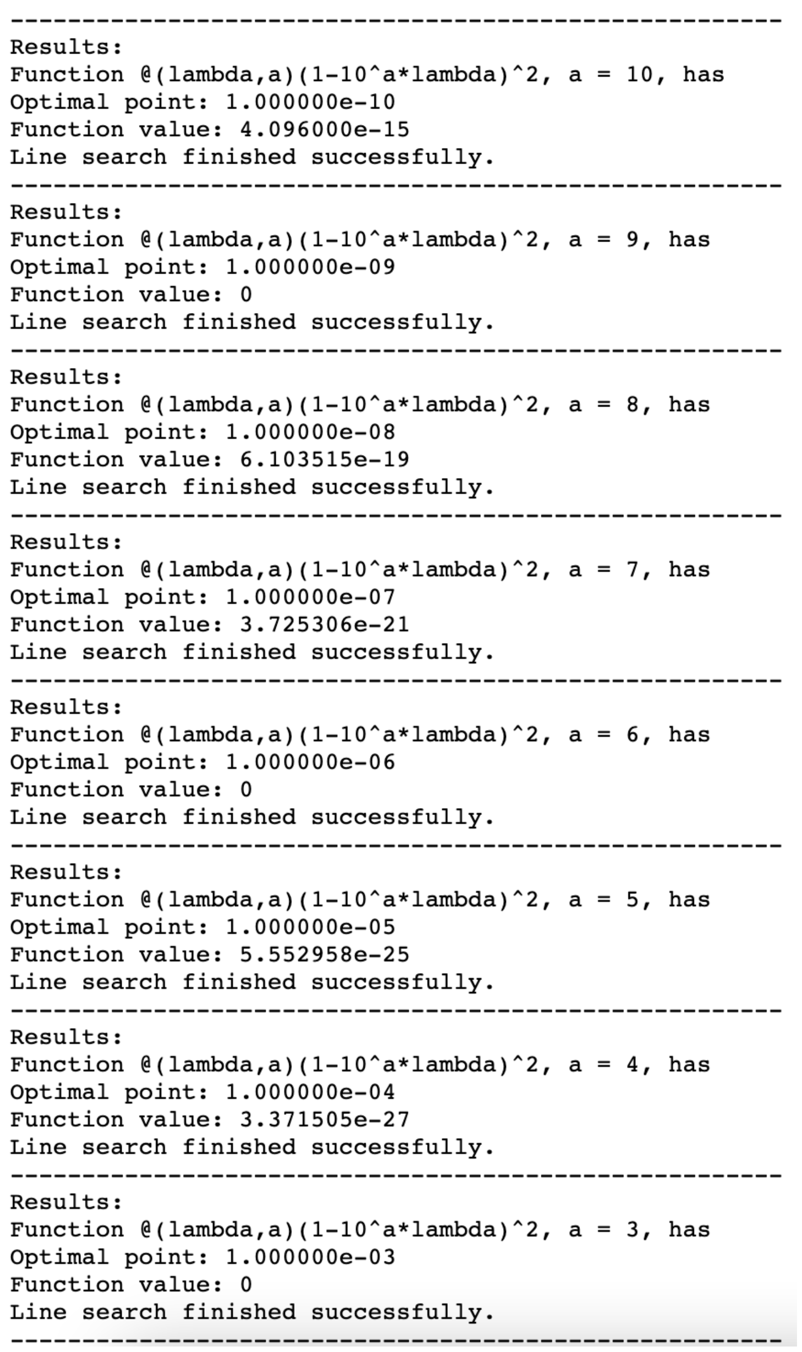
Deviations of the implementation from the pure algorithm

The choice of using Newton's method in the line search sub-problem may come with a few concerns which one has to consider when implementing an algorithm. The potential problems are connected to the assumptions that Newton's method makes (which are required for the method to have the desired behaviour).

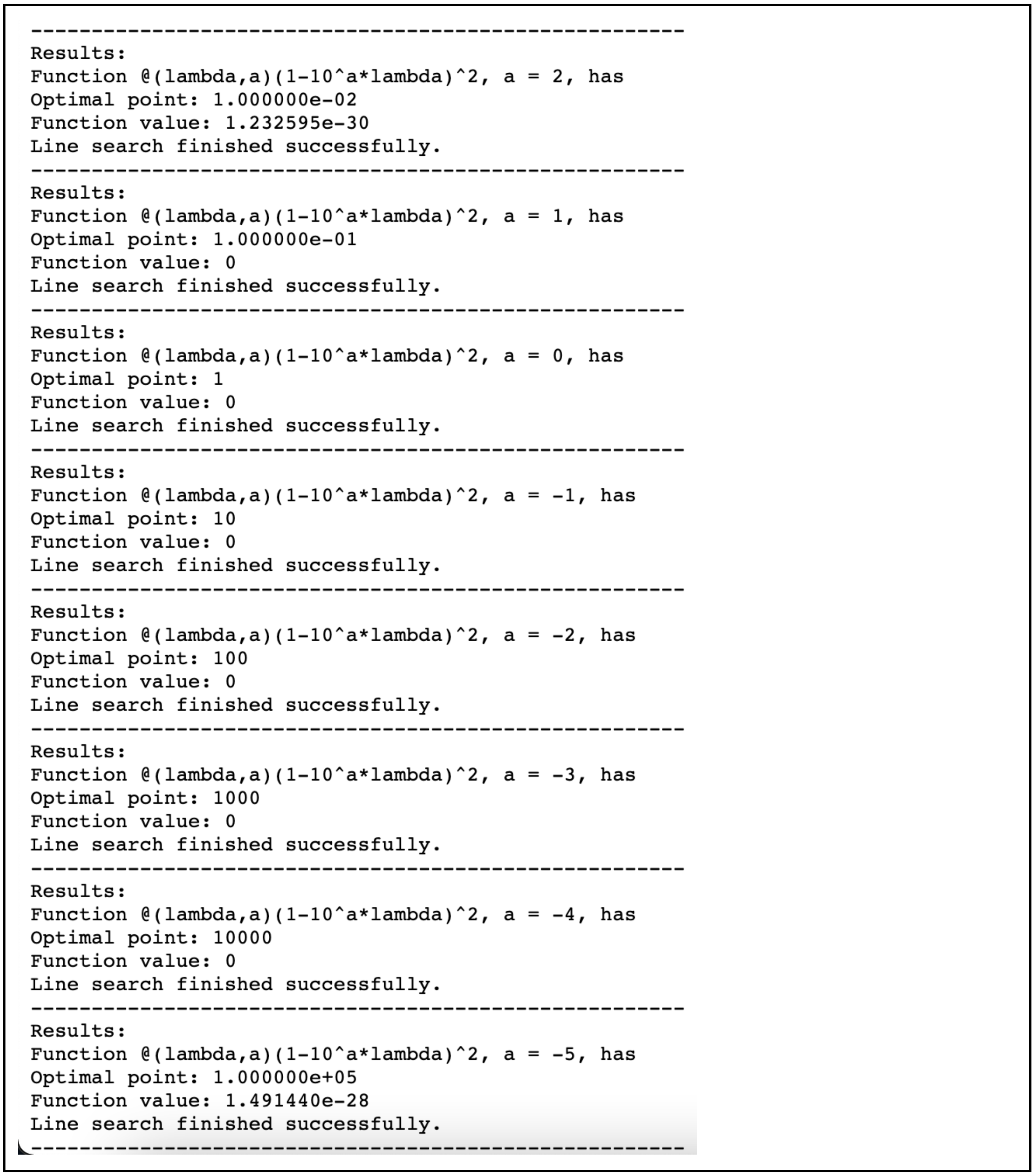
The first problem comes as a consequence of the numerical differentiation which is performed in our algorithm. Even though the function might actually be convex at a certain point, the numerical differentiation (which comes with a certain level of accuracy) might say that the function is non-convex (i.e. yield a non-positive second derivative). This problem was solved by incrementing the value of the step size h used in the numerical differentiation: increasing it until a positive second derivative is obtained. In the cases when h could be incremented to some large value $h_{max} \approx 1e10$ without finding a positive second derivative the function is stated to be non-convex and the program is stopped. This approach enable the program to successfully minimize the function $x \mapsto (1 - 10^{a}x)^{2}$ for all -10 ≤ a ≤ 10, see Table 14. Specifically, the cases when a ≤ -8 required this modification of the usual Newton line search algorithm because of the extremely small value of the second derivative everywhere. In fact, we require the second derivative not only to be positive, but greater than a very small prescribed value (around 1e-20) to be on the safe side.

Table 14

*Performing line search on functions (1-10^a*x)^2 for a in [-10, 10].*



```
-----------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 10, has
Optimal point: 1.000000e-10
Function value: 4.096000e-15
Line search finished successfully.
-----------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 9, has
Optimal point: 1.000000e-09
Function value: 0
Line search finished successfully.
-----------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 8, has
Optimal point: 1.000000e-08
Function value: 6.103515e-19
Line search finished successfully.
-----------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 7, has
Optimal point: 1.000000e-07
Function value: 3.725306e-21
Line search finished successfully.
-----------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 6, has
Optimal point: 1.000000e-06
Function value: 0
Line search finished successfully.
-----------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 5, has
Optimal point: 1.000000e-05
Function value: 5.552958e-25
Line search finished successfully.
-----------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 4, has
Optimal point: 1.000000e-04
Function value: 3.371505e-27
Line search finished successfully.
-----------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 3, has
Optimal point: 1.000000e-03
Function value: 0
Line search finished successfully.
-----------------------------------------------------
```



```
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 2, has
Optimal point: 1.000000e-02
Function value: 1.232595e-30
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 1, has
Optimal point: 1.000000e-01
Function value: 0
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = 0, has
Optimal point: 1
Function value: 0
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = -1, has
Optimal point: 10
Function value: 0
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = -2, has
Optimal point: 100
Function value: 0
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = -3, has
Optimal point: 1000
Function value: 0
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = -4, has
Optimal point: 10000
Function value: 0
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = -5, has
Optimal point: 1.000000e+05
Function value: 1.491440e-28
Line search finished successfully.
------------------------------------------------------
```

```
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = -6, has
Optimal point: 9.999999e+05
Function value: 5.108799e-15
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = -7, has
Optimal point: 1.000161e+07
Function value: 2.592823e-08
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = -8, has
Optimal point: 9.999997e+07
Function value: 7.823388e-14
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = -9, has
Optimal point: 1.000000e+09
Function value: 3.882002e-23
Line search finished successfully.
------------------------------------------------------
Results:
Function @(lambda,a)(1-10^a*lambda)^2, a = -10, has
Optimal point: 1.000000e+10
Function value: 1.810054e-13
Line search finished successfully.
```

The second problem that occurred was that the function value at the next generated point in the sequence could in fact be greater than the function value at the previous point, *although the second derivative was positive* (implying convexity). Once more, this might stem from the numerical differentiation obviously not being exact. This problem was mitigated by checking if the second derivative was positive but still the function value of the next point was greater than at the current point. If that was the case, the value of h was decreased, and using this new step size the derivatives and the next point in the sequence was recomputed. This alteration enabled the algorithm to minimize the Rosenbrock function.

Source code

The source code is for convenience uploaded to Github at:
https://github.com/simondanielsson/optimization

Appendix 1

Below, the result of all test performed on the function nonlinearmin() is displayed.

```
>> nonlinearmin_test(1, 0)
-------------------------------------------------------------------------------------------
Minimizing positive definite quadratic form, min at origin
Executing minimization of function @(x)x'*H*x
Starting at point [1, 2],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init             +1.00e+00             2.30e+01
                   +2.00e+00

    1          1     +3.80e-01   2.16e+00    4.55e-01    2.09e+01       3     1.03e-01
                   -6.83e-02

               2     +7.12e-10   3.86e-01    2.97e-17    2.38e+00       3     1.63e-01
                   +2.37e-09

    2          1     +7.12e-10   0.00e+00    2.97e-17    2.41e-08       1     0.00e+00
                   +2.37e-09


x =

   1.0e-08 *

    0.0712
    0.2372


no_its =

     3


normg =

   2.4101e-08

Executing minimization of function @(x)x'*H*x
```

```
Starting at point [1, 2],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +1.00e+00              2.30e+01
                    +2.00e+00

    1           1    +3.80e-01   2.16e+00    4.55e-01    2.09e+01       3     1.03e-01
                    -6.83e-02

               2    +7.12e-10   3.86e-01    2.96e-17    2.38e+00       3     1.61e-01
                    +2.37e-09

    2           1    +7.12e-10   0.00e+00    2.96e-17    2.41e-08       1     0.00e+00
                    +2.37e-09


x =

   1.0e-08 *

    0.0712
    0.2372


no_its =

     3


normg =

   2.4100e-08

------------------------------------------------------------------------------------------
Minimizing positive definite quadratic form, min at origin
Executing minimization of function @(x)x'*H*x
Starting at point [-5, -3],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              -5.00e+00              1.20e+02
                    -3.00e+00
    1           1     -1.25e+00   5.30e+00    7.50e+00    4.24e+01       3     1.25e-01
                     +7.50e-01

                2     +1.96e-08   1.46e+00    3.06e-15    1.06e+01       3     1.42e-01
                     +1.96e-08

    2           1     +1.96e-08   0.00e+00    3.06e-15    2.28e-07       1     0.00e+00
                     +1.96e-08


x =

   1.0e-07 *

    0.1957
    0.1957


no_its =

     3


normg =

   2.2820e-07

Executing minimization of function @(x)x'*H*x
Starting at point [-5, -3],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init               -5.00e+00             1.20e+02
                     -3.00e+00

    1           1     -1.25e+00   5.30e+00    7.50e+00    4.24e+01       3     1.25e-01
                     +7.50e-01

                2     +1.96e-08   1.46e+00    3.08e-15    1.06e+01       3     1.33e-01
                     +1.96e-08

    2           1     +1.96e-08   0.00e+00    3.08e-15    2.29e-07       1     0.00e+00
                     +1.96e-08
```

```
x =

   1.0e-07 *

    0.1961
    0.1961


no_its =

     3


normg =

   2.2874e-07

-------------------------------------------------------------------------------------------
Minimizing positive definite quadratic form, min at origin
Executing minimization of function @(x)x'*H*x
Starting at point [1, 2],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +1.00e+00              1.00e+06
                    +2.00e+00

    1           1     -3.94e-11   1.00e+00    1.20e+01    2.00e+06       3     5.00e-07
                    +2.00e+00

               2     +2.42e-09   2.00e+00    5.84e-12    1.20e+01       3     1.67e-01
                    +9.89e-13

    2           1     +2.42e-09   0.00e+00    5.84e-12    4.83e-03       1     0.00e+00
                    +9.89e-13


x =

   1.0e-08 *

    0.2417
    0.0001
```

```
no_its =

     3


normg =

    0.0048

Executing minimization of function @(x)x'*H*x
Starting at point [1, 2],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init               +1.00e+00               1.00e+06
                     +2.00e+00


     1            1     -3.94e-11   1.00e+00    1.20e+01    2.00e+06       3     5.00e-07
                     +2.00e+00


                  2     +2.42e-09   2.00e+00    5.84e-12    1.20e+01       3     1.67e-01
                     +9.90e-13


     2            1     +2.42e-09   0.00e+00    5.84e-12    4.83e-03       1     0.00e+00
                     +9.90e-13



x =

   1.0e-08 *

    0.2417
    0.0001



no_its =

     3



normg =
    0.0048

-------------------------------------------------------------------------------------------
Minimizing positive definite quadratic form, min at origin
Executing minimization of function @(x)x'*H*x
Starting at point [7, 7, 7, 7, 7],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init            +7.00e+00              1.22e+03
                  +7.00e+00
                  +7.00e+00
                  +7.00e+00
                  +7.00e+00

    1          1     +6.06e+00   1.21e+01    1.36e+02    1.80e+02       3     6.73e-02
                  +4.17e+00
                  +2.29e+00
                  +4.00e-01
                  -1.49e+00

               2     +4.77e+00   4.60e+00    3.51e+01    4.52e+01       3     1.05e-01
                  +1.44e+00
                  -4.09e-01
                  -7.64e-01
                  +3.70e-01

               3     +3.25e+00   2.61e+00    1.29e+01    1.85e+01       3     1.53e-01
                  -2.87e-01
                  -4.70e-01
                  +3.72e-01
                  -8.84e-02

               4     +1.47e+00   2.05e+00    4.20e+00    9.84e+00       3     2.41e-01
                  -6.55e-01
                  +3.54e-01
                  -1.20e-01
                  +1.82e-02

               5     -3.43e-08   1.66e+00    1.88e-13    6.30e+00       3     3.27e-01
                  -8.64e-08
                  -1.08e-07
                  -9.85e-08
                  -6.44e-08
   2          1     -3.43e-08   0.00e+00    1.88e-13    2.17e-06       1     0.00e+00
                   -8.64e-08
                   -1.08e-07
                   -9.85e-08
                   -6.44e-08


x =

   1.0e-06 *

  -0.0343
  -0.0864
  -0.1084
  -0.0985
  -0.0644


no_its =

     6


normg =

   2.1664e-06

Executing minimization of function @(x)x'*H*x
Starting at point [7, 7, 7, 7, 7],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init               +7.00e+00              1.22e+03
                   +7.00e+00
                   +7.00e+00
                   +7.00e+00
                   +7.00e+00

   1          1     +6.06e+00   1.21e+01    1.36e+02    1.80e+02       3     6.73e-02
                   +4.17e+00
                   +2.29e+00
                   +4.00e-01
                   -1.49e+00
```

```
          2    +4.77e+00   4.60e+00    3.51e+01    4.52e+01       3     9.88e-02
                +1.44e+00
                -4.09e-01
                -7.64e-01
                +3.70e-01

          3    +3.25e+00   2.61e+00    1.29e+01    1.85e+01       3     1.30e-01
                -2.87e-01
                -4.70e-01
                +3.72e-01
                -8.84e-02

          4    +1.47e+00   2.05e+00    4.20e+00    9.84e+00       3     1.80e-01
                -6.55e-01
                +3.54e-01
                -1.20e-01
                +1.82e-02

          5    -5.77e-08   1.66e+00    1.64e-13    6.30e+00       3     2.12e-01
                -8.98e-08
                -7.88e-08
                -7.36e-08
                -8.63e-08

   2       1    -5.77e-08   0.00e+00    1.64e-13    2.10e-06       1     0.00e+00
                -8.98e-08
                -7.88e-08
                -7.36e-08
                -8.63e-08


x =

   1.0e-07 *

   -0.5768
   -0.8985
   -0.7882
   -0.7361
   -0.8634


no_its =
    6


normg =

   2.0982e-06

-------------------------------------------------------------------------------------------
Minimizing negative definite quadratic form, has no minimum
Executing minimization of function @(x)x'*H*x
Starting at point [5, 9],
Using DFP method.

outer it.   iteration     x       step size    f(x)     norm(grad)  ls iters   lambda
   init              +5.00e+00              -1.06e+02
                    +9.00e+00

[Warning: The function does not seem to be convex: cannot be
minimized.]

Executing minimization of function @(x)x'*H*x
Starting at point [5, 9],
Using BFGS method.

outer it.   iteration     x       step size    f(x)     norm(grad)  ls iters   lambda
   init              +5.00e+00              -1.06e+02
                    +9.00e+00

[Warning: The function does not seem to be convex: cannot be
minimized.]

-------------------------------------------------------------------------------------------
Minimizing indefinite quadratic form, has no minimum
Executing minimization of function @(x)x'*H*x
Starting at point [5, 9],
Using DFP method.

outer it.   iteration     x       step size    f(x)     norm(grad)  ls iters   lambda
   init              +5.00e+00              3.30e+02
                    +9.00e+00


    1          1     +8.57e+00   1.13e+01    -2.06e+02    9.49e+01       3     1.19e-01
                    -1.71e+00
```

```
[Warning: The function does not seem to be convex: cannot be
minimized.]

Executing minimization of function @(x)x'*H*x
Starting at point [5, 9],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +5.00e+00              3.30e+02
                    +9.00e+00

    1          1     +8.57e+00   1.13e+01    -2.06e+02    9.49e+01       3      1.19e-01
                    -1.71e+00

[Warning: The function does not seem to be convex: cannot be
minimized.]

------------------------------------------------------------------------------------------

Minimizing rosenbrock function, min at (1, 1)
Initial point [200, 200]
Executing minimization of function rosenbrock
Starting at point [200, 200],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +2.00e+02              1.58e+11
                    +2.00e+02

    1          1     -1.20e-03   2.00e+02    4.02e+06    3.18e+09       23     6.28e-08
                    +2.00e+02

               2     -5.04e-01   2.00e+02    2.26e+00    4.01e+04       5     4.99e-03
                    +2.54e-01

    2          1     -4.89e-01   1.54e-02    2.24e+00    3.00e+00       3     5.12e-03
                    +2.54e-01

               2     -2.81e-01   2.92e-01    1.73e+00    3.05e+00       3     1.37e-01
                    +4.88e-02

    3          1     -2.56e-01   3.52e-02    1.58e+00    8.46e+00       3     4.16e-03
                    +7.38e-02
```

	2	-1.53e-01	1.18e-01	1.34e+00	2.34e+00	7	5.21e-02	+1.65e-02
4	1	-1.26e-01	3.09e-02	1.29e+00	3.07e+00	3	1.01e-02	+3.06e-02
	2	-2.73e-02	1.02e-01	1.06e+00	3.31e+00	7	4.54e-02	+2.80e-03
5	1	+7.66e-03	3.57e-02	9.87e-01	2.07e+00	6	1.72e-02	-4.26e-03
	2	+8.96e-02	8.20e-02	8.39e-01	2.15e+00	7	3.71e-02	-2.18e-03
6	1	+1.19e-01	5.07e-02	8.38e-01	2.51e+00	2	2.02e-02	+3.91e-02
	2	+2.43e-01	1.26e-01	5.77e-01	5.80e+00	6	6.89e-02	+6.42e-02
7	1	+2.55e-01	1.38e-02	5.60e-01	2.31e+00	3	6.01e-03	+5.78e-02
	2	+3.56e-01	1.14e-01	4.45e-01	1.62e+00	4	8.56e-02	+1.09e-01
8	1	+3.52e-01	1.34e-02	4.21e-01	3.70e+00	4	3.61e-03	+1.22e-01
	2	+4.91e-01	1.71e-01	2.97e-01	1.10e+00	4	1.62e-01	+2.22e-01
9	1	+4.84e-01	1.25e-02	2.67e-01	4.79e+00	3	2.61e-03	+2.32e-01
	2	+6.13e-01	1.82e-01	1.76e-01	7.50e-01	5	2.45e-01	+3.60e-01
10	1	+6.07e-01	9.24e-03	1.55e-01	4.57e+00	3	2.02e-03	+3.66e-01
	2	+7.23e-01	1.84e-01	9.39e-02	5.03e-01	6	3.69e-01	

```
                 +5.09e-01

   11          1    +7.17e-01   6.75e-03    8.00e-02    4.15e+00      3     1.63e-03
                 +5.14e-01

             2    +8.18e-01   1.78e-01    4.26e-02    3.24e-01      6     5.50e-01
                 +6.60e-01

   12          1    +8.14e-01   4.71e-03    3.45e-02    3.46e+00      3     1.36e-03
                 +6.62e-01

             2    +8.98e-01   1.61e-01    1.49e-02    1.94e-01      5     8.27e-01
                 +8.00e-01

   13          1    +8.95e-01   3.06e-03    1.10e-02    2.58e+00      3     1.18e-03
                 +8.01e-01

             2    +9.58e-01   1.30e-01    3.17e-03    1.02e-01      4     1.27e+00
                 +9.14e-01

   14          1    +9.57e-01   1.67e-03    1.87e-03    1.56e+00      3     1.07e-03
                 +9.15e-01

             2    +9.57e-01   0.00e+00    1.87e-03    4.00e-02      1     0.00e+00
                 +9.15e-01

   15          1    +9.57e-01   0.00e+00    1.87e-03    4.00e-02      1     0.00e+00
                 +9.15e-01


x =

   0.9568
   0.9153


no_its =

   29


normg =

   0.0400
```

```
------------------------------------------------------------------------------------------
Minimizing rosenbrock function, min at (1, 1)
Initial point [399, -711]
Using DFP method
Executing minimization of function rosenbrock
Starting at point [399, -711],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +3.99e+02              2.56e+12
                    -7.11e+02

    1          1     -5.79e-04   3.99e+02    5.05e+07    2.55e+10       24     1.56e-08
                    -7.10e+02

               2     +8.87e-01   7.11e+02    1.28e-02    1.42e+05       5     5.01e-03
                    +7.86e-01

    2          1     +8.87e-01   6.98e-05    1.28e-02    2.27e-01       2     3.08e-04
                    +7.86e-01

               2     +9.51e-01   1.30e-01    4.40e-03    1.84e-01       4     1.17e+00
                    +8.99e-01

    3          1     +9.49e-01   1.96e-03    2.62e-03    1.81e+00       3     1.08e-03
                    +9.00e-01

               2     +9.90e-01   8.82e-02    3.51e-04    4.77e-02       3     1.85e+00
                    +9.78e-01

    4          1     +9.90e-01   0.00e+00    3.51e-04    6.80e-01       1     0.00e+00
                    +9.78e-01


x =

    0.9899
    0.9782


no_its =

     7
```

```
normg =

    0.6804

-------------------------------------------------------------------------------------------
Minimizing rosenbrock function, min at (1, 1)
Initial point [399, -711]
Using BFGS method
Executing minimization of function rosenbrock
Starting at point [399, -711],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
  init             +3.99e+02              2.56e+12
                  -7.11e+02

    1          1    -5.79e-04   3.99e+02    5.05e+07    2.55e+10       24     1.56e-08
                  -7.10e+02

               2    +8.87e-01   7.11e+02    1.28e-02    1.42e+05       5     5.01e-03
                  +7.86e-01

    2          1    +8.87e-01   6.98e-05    1.28e-02    2.27e-01       2     3.08e-04
                  +7.86e-01

               2    +9.51e-01   1.31e-01    4.28e-03    1.84e-01       4     8.98e-01
                  +9.00e-01

    3          1    +9.50e-01   1.94e-03    2.55e-03    1.79e+00       3     1.08e-03
                  +9.01e-01

               2    +9.90e-01   8.73e-02    3.35e-04    4.70e-02       3     1.86e+00
                  +9.79e-01

    4          1    +9.90e-01   0.00e+00    3.35e-04    6.67e-01       1     0.00e+00
                  +9.79e-01


x =

    0.9901
    0.9788
```

```
no_its =

     7


normg =

    0.6666

-------------------------------------------------------------------------------------------
Minimizing rosenbrock function, min at (1, 1)
Initial point [3990, -7111]
Using DFP method
Executing minimization of function rosenbrock
Starting at point [3990, -7111],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +3.99e+03              2.54e+16
                    -7.11e+03

    1          1      -4.58e-05   3.99e+03    5.06e+09    2.54e+13       30     1.57e-10
                    -7.11e+03

               2     +7.83e-01   7.11e+03    4.71e-02    1.42e+06       4     5.00e-03
                    +6.13e-01

    2          1      +7.84e-01   8.82e-04    4.70e-02    4.34e-01       2     2.03e-03
                    +6.13e-01

               2     +8.69e-01   1.59e-01    2.37e-02    2.76e-01       5     6.83e-01
                    +7.47e-01

    3          1      +8.66e-01   3.78e-03    1.80e-02    3.03e+00       3     1.24e-03
                    +7.49e-01

               2     +9.84e-01   2.37e-01    1.76e-02    1.34e-01       2     1.77e+00
                    +9.55e-01

    4          1      +9.78e-01   5.93e-03    4.63e-04    5.78e+00       3     1.03e-03
                    +9.57e-01
```

```
             2    +9.78e-01   0.00e+00    4.63e-04    1.96e-02       1     0.00e+00
                  +9.57e-01

    5        1    +9.78e-01   0.00e+00    4.63e-04    1.96e-02       1     0.00e+00
                  +9.57e-01


x =

    0.9785
    0.9574


no_its =

     9


normg =

    0.0196

------------------------------------------------------------------------------------------
Minimizing rosenbrock function, min at (1, 1)
Initial point [3990, -7111]
Using BFGS method
Executing minimization of function rosenbrock
Starting at point [3990, -7111],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +3.99e+03              2.54e+16
                  -7.11e+03

    1        1    -4.58e-05   3.99e+03    5.06e+09    2.54e+13      30     1.57e-10
                  -7.11e+03

             2    +7.83e-01   7.11e+03    4.71e-02    1.42e+06       4     5.00e-03
                  +6.13e-01

    2        1    +7.84e-01   8.82e-04    4.70e-02    4.34e-01       2     2.03e-03
                  +6.13e-01

             2    +8.69e-01   1.59e-01    2.37e-02    2.76e-01       5     4.87e-01
```

```
                 +7.47e-01

   3          1     +8.66e-01   3.78e-03    1.80e-02    3.03e+00       3     1.24e-03
                 +7.49e-01

              2    +9.84e-01   2.37e-01    1.76e-02    1.34e-01       2     1.76e+00
                 +9.55e-01

   4          1     +9.78e-01   5.93e-03    4.64e-04    5.78e+00       3     1.03e-03
                 +9.57e-01

              2    +9.78e-01   0.00e+00    4.64e-04    1.96e-02       1     0.00e+00
                 +9.57e-01

   5          1     +9.78e-01   0.00e+00    4.64e-04    1.96e-02       1     0.00e+00
                 +9.57e-01


x =

   0.9785
   0.9573


no_its =

    9


normg =

   0.0196

-------------------------------------------------------------------------------------------
Minimizing rosenbrock function, min at (1, 1)
Initial point [200, 200]
Using DFP method
Executing minimization of function rosenbrock
Starting at point [200, 200],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +2.00e+02               1.58e+11
                 +2.00e+02
```

1	1	-1.20e-03 +2.00e+02	2.00e+02	4.02e+06	3.18e+09	23	6.28e-08
	2	-5.04e-01 +2.54e-01	2.00e+02	2.26e+00	4.01e+04	5	4.99e-03
2	1	-4.89e-01 +2.54e-01	1.54e-02	2.24e+00	3.00e+00	3	5.12e-03
	2	-2.81e-01 +4.88e-02	2.92e-01	1.73e+00	3.05e+00	3	1.37e-01
3	1	-2.56e-01 +7.38e-02	3.52e-02	1.58e+00	8.46e+00	3	4.16e-03
	2	-1.53e-01 +1.65e-02	1.18e-01	1.34e+00	2.34e+00	7	5.21e-02
4	1	-1.26e-01 +3.06e-02	3.09e-02	1.29e+00	3.07e+00	3	1.01e-02
	2	-2.73e-02 +2.80e-03	1.02e-01	1.06e+00	3.31e+00	7	4.54e-02
5	1	+7.66e-03 -4.26e-03	3.57e-02	9.87e-01	2.07e+00	6	1.72e-02
	2	+8.96e-02 -2.18e-03	8.20e-02	8.39e-01	2.15e+00	7	3.71e-02
6	1	+1.19e-01 +3.91e-02	5.07e-02	8.38e-01	2.51e+00	2	2.02e-02
	2	+2.43e-01 +6.42e-02	1.26e-01	5.77e-01	5.80e+00	6	6.89e-02
7	1	+2.55e-01 +5.78e-02	1.38e-02	5.60e-01	2.31e+00	3	6.01e-03
	2	+3.56e-01 +1.09e-01	1.14e-01	4.45e-01	1.62e+00	4	8.56e-02
8	1	+3.52e-01	1.34e-02	4.21e-01	3.70e+00	4	3.61e-03

```
          +1.22e-01

       2   +4.91e-01  1.71e-01   2.97e-01   1.10e+00     4    1.62e-01
          +2.22e-01

9       1   +4.84e-01  1.25e-02   2.67e-01   4.79e+00     3    2.61e-03
          +2.32e-01

       2   +6.13e-01  1.82e-01   1.76e-01   7.50e-01     5    2.45e-01
          +3.60e-01

10       1   +6.07e-01  9.24e-03   1.55e-01   4.57e+00     3    2.02e-03
          +3.66e-01

       2   +7.23e-01  1.84e-01   9.39e-02   5.03e-01     6    3.69e-01
          +5.09e-01

11       1   +7.17e-01  6.75e-03   8.00e-02   4.15e+00     3    1.63e-03
          +5.14e-01

       2   +8.18e-01  1.78e-01   4.26e-02   3.24e-01     6    5.50e-01
          +6.60e-01

12       1   +8.14e-01  4.71e-03   3.45e-02   3.46e+00     3    1.36e-03
          +6.62e-01

       2   +8.98e-01  1.61e-01   1.49e-02   1.94e-01     5    8.27e-01
          +8.00e-01

13       1   +8.95e-01  3.06e-03   1.10e-02   2.58e+00     3    1.18e-03
          +8.01e-01

       2   +9.58e-01  1.30e-01   3.17e-03   1.02e-01     4    1.27e+00
          +9.14e-01

14       1   +9.57e-01  1.67e-03   1.87e-03   1.56e+00     3    1.07e-03
          +9.15e-01

       2   +9.57e-01  0.00e+00   1.87e-03   4.00e-02     1    0.00e+00
          +9.15e-01

15       1   +9.57e-01  0.00e+00   1.87e-03   4.00e-02     1    0.00e+00
          +9.15e-01
```

```
x =

   0.9568
   0.9153


no_its =

   29


normg =

   0.0400

-------------------------------------------------------------------------------------------
Minimizing rosenbrock function, min at (1, 1)
Initial point [200, 200]
Using BFGS method
Executing minimization of function rosenbrock
Starting at point [200, 200],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
  init              +2.00e+02              1.58e+11
                   +2.00e+02

   1          1     -1.20e-03   2.00e+02    4.02e+06    3.18e+09       23     6.28e-08
                   +2.00e+02

             2     -5.04e-01   2.00e+02    2.26e+00    4.01e+04       5     4.99e-03
                   +2.54e-01

   2          1     -4.89e-01   1.54e-02    2.24e+00    3.00e+00       3     5.12e-03
                   +2.54e-01

             2     -4.13e-01   1.06e-01    2.00e+00    3.05e+00       9     2.44e-02
                   +1.79e-01

   3          1     -3.50e-01   1.02e-01    1.88e+00    2.23e+00       4     4.59e-02
                   +9.89e-02

             2     -2.17e-01   1.64e-01    1.67e+00    7.62e+00       5     6.05e-03
```

```
           +3.67e-03

4       1    -1.89e-01   4.76e-02    1.42e+00    1.06e+01      3     4.48e-03
           +4.24e-02

        2    -1.08e-01   8.87e-02    1.23e+00    2.30e+00      7     3.77e-02
           +7.26e-03

5       1    -5.65e-02   5.42e-02    1.17e+00    2.55e+00      5     2.12e-02
           +2.56e-02

        2    -5.65e-02   0.00e+00    1.17e+00    4.76e+00      1     0.00e+00
           +2.56e-02

6       1    -5.63e-02   5.57e-04    1.16e+00    4.76e+00      3     1.17e-04
           +2.50e-02

        2    +1.70e-02   7.40e-02    9.88e-01    4.66e+00      9     2.72e-02
           +1.51e-02

7       1    +3.14e-02   2.51e-02    9.42e-01    3.62e+00      3     6.94e-03
           -5.48e-03

        2    +8.13e-02   5.00e-02    8.53e-01    2.26e+00      7     1.87e-02
           -3.01e-03

8       1    +9.80e-02   2.69e-02    8.21e-01    2.45e+00      3     1.10e-02
           +1.81e-02

        2    +1.13e-01   1.51e-02    7.93e-01    2.72e+00      6     3.72e-03
           +2.06e-02

9       1    +1.30e-01   2.10e-02    7.65e-01    2.64e+00      3     7.96e-03
           +8.12e-03

        2    +1.75e-01   4.66e-02    6.93e-01    2.17e+00      7     1.66e-02
           +1.96e-02

10        1    +1.82e-01   1.86e-02    6.71e-01    2.38e+00      4     7.83e-03
           +3.69e-02

        2    +2.33e-01   5.36e-02    5.89e-01    2.06e+00      7     1.97e-02
           +5.45e-02
```

11	1	+2.57e-01	2.44e-02	5.69e-01	1.58e+00	3	1.54e-02	+5.30e-02
	2	+2.82e-01	2.83e-02	5.33e-01	2.60e+00	6	5.71e-03	+6.63e-02
12	1	+2.82e-01	1.29e-02	5.16e-01	2.63e+00	3	4.90e-03	+7.92e-02
	2	+3.83e-01	1.17e-01	3.91e-01	1.42e+00	5	7.23e-02	+1.37e-01
13	1	+3.82e-01	8.47e-03	3.82e-01	2.12e+00	3	4.01e-03	+1.45e-01
	2	+5.16e-01	1.69e-01	2.68e-01	1.11e+00	4	1.36e-01	+2.48e-01
14	1	+5.09e-01	1.14e-02	2.41e-01	4.63e+00	3	2.47e-03	+2.57e-01
	2	+6.36e-01	1.83e-01	1.57e-01	6.94e-01	6	2.61e-01	+3.89e-01
15	1	+6.30e-01	8.75e-03	1.37e-01	4.54e+00	3	1.93e-03	+3.95e-01
	2	+7.43e-01	1.84e-01	8.12e-02	4.62e-01	6	3.96e-01	+5.40e-01
16	1	+7.38e-01	6.30e-03	6.86e-02	4.02e+00	3	1.57e-03	+5.44e-01
	2	+8.36e-01	1.75e-01	3.53e-02	2.94e-01	6	5.94e-01	+6.89e-01
17	1	+8.32e-01	4.34e-03	2.82e-02	3.29e+00	3	1.32e-03	+6.92e-01
	2	+9.12e-01	1.55e-01	1.15e-02	1.73e-01	5	8.96e-01	+8.25e-01
18	1	+9.09e-01	2.74e-03	8.23e-03	2.37e+00	3	1.16e-03	+8.27e-01

```
            2     +9.67e-01   1.20e-01    2.09e-03    8.74e-02       4     1.38e+00
                  +9.32e-01

   19          1     +9.66e-01   1.40e-03    1.17e-03    1.33e+00       2     1.05e-03
                  +9.33e-01

            2     +9.66e-01   0.00e+00    1.17e-03    3.14e-02       1     0.00e+00
                  +9.33e-01

   20          1     +9.66e-01   0.00e+00    1.17e-03    3.14e-02       1     0.00e+00
                  +9.33e-01


x =

   0.9659
   0.9327


no_its =

   39


normg =

   0.0314

-------------------------------------------------------------------------------------------
Minimizing Booth, min at (1, 3)
Initial point [9, 10]
Using DFP method
Executing minimization of function @(x)(x(1)+2*x(2)-7)^2+(2*x(1)+x(2)-5)^2
Starting at point [9, 10],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
  init              +9.00e+00              1.01e+03
                  +1.00e+01

   1          1     +1.44e+00   1.06e+01    3.95e-01    1.91e+02       3     5.56e-02
                  +2.56e+00
            2    +1.00e+00   6.29e-01    8.12e-16    1.26e+00      3    5.00e-01
                 +3.00e+00

   2        1    +1.00e+00   0.00e+00    8.12e-16    1.71e-07      1    0.00e+00
                 +3.00e+00


x =

    1.0000
    3.0000


no_its =

     3


normg =

   1.7099e-07

-------------------------------------------------------------------------------------------
Minimizing Booth, min at (1, 3)
Initial point [9, 10]
Using BFGS method
Executing minimization of function @(x)(x(1)+2*x(2)-7)^2+(2*x(1)+x(2)-5)^2
Starting at point [9, 10],
Using BFGS method.

outer it.  iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +9.00e+00              1.01e+03
                     +1.00e+01

   1        1    +1.44e+00   1.06e+01    3.95e-01    1.91e+02      3    5.56e-02
                 +2.56e+00

            2    +1.00e+00   6.29e-01    8.12e-16    1.26e+00      3    5.00e-01
                 +3.00e+00

   2        1    +1.00e+00   0.00e+00    8.12e-16    1.71e-07      1    0.00e+00
                 +3.00e+00
```

```
x =

    1.0000
    3.0000


no_its =

     3


normg =

   1.7099e-07

-------------------------------------------------------------------------------------------
Minimizing Booth, min at (1, 3)
Initial point [1139, 9991]
Using DFP method
Executing minimization of function @(x)(x(1)+2*x(2)-7)^2+(2*x(1)+x(2)-5)^2
Starting at point [1139, 9991],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +1.14e+03              5.96e+08
                    +9.99e+03

    1          1     -3.97e+03   7.95e+03    3.09e+07    1.42e+05       3     5.59e-02
                    +3.89e+03

               2     +1.04e+00   5.56e+03    3.28e-02    1.12e+04       3     5.00e-01
                    +3.05e+00

    2          1     +9.97e-01   6.04e-02    2.25e-05    1.09e+00       3     5.56e-02
                    +3.00e+00

               2     +9.97e-01   0.00e+00    2.25e-05    9.50e-03       1     0.00e+00
                    +3.00e+00

    3          1     +9.97e-01   0.00e+00    2.25e-05    9.50e-03       1     0.00e+00
                    +3.00e+00


x =
    0.9966
    3.0034


no_its =

     5


normg =

    0.0095

------------------------------------------------------------------------------------------
Minimizing Booth, min at (1, 3)
Initial point [1139, 9991]
Using BFGS method
Executing minimization of function @(x)(x(1)+2*x(2)-7)^2+(2*x(1)+x(2)-5)^2
Starting at point [1139, 9991],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +1.14e+03              5.96e+08
                    +9.99e+03

     1          1     -3.97e+03   7.95e+03    3.09e+07    1.42e+05       3     5.59e-02
                    +3.89e+03

                2     +1.04e+00   5.56e+03    3.29e-02    1.12e+04       3     4.97e-01
                    +3.05e+00

     2          1     +9.97e-01   6.04e-02    2.26e-05    1.09e+00       3     5.56e-02
                    +3.00e+00

                2     +9.97e-01   0.00e+00    2.26e-05    9.50e-03       1     0.00e+00
                    +3.00e+00

     3          1     +9.97e-01   0.00e+00    2.26e-05    9.50e-03       1     0.00e+00
                    +3.00e+00


x =
    0.9966
    3.0034


no_its =

     5


normg =

    0.0095


x04 =

    -1
    -1


x05 =

     1
     1

------------------------------------------------------------------------------------------
Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)
Initial point [-3, -3]
Using DFP method
Executing minimization of function @(x)1/2*(x(1)^4-16*x(1)^2+5*x(1)+x(2)^4-16*x(2)^2+5*x(2))
Starting at point [-3, -3],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              -3.00e+00               -7.80e+01
                    -3.00e+00

    1           1     -2.90e+00   1.36e-01    -7.83e+01    4.95e+00       4     2.76e-02
                    -2.90e+00

               2     -2.90e+00   0.00e+00    -7.83e+01    0.00e+00       1     0.00e+00
                    -2.90e+00

    2           1     -2.90e+00   0.00e+00    -7.83e+01    0.00e+00       1     0.00e+00
```

```
                    -2.90e+00


x =

   -2.9035
   -2.9035


no_its =

     3


normg =

     0

--------------------------------------------------------------------------------------------
Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)
Initial point [-3, -3]
Using BFGS method
Executing minimization of function @(x)1/2*(x(1)^4-16*x(1)^2+5*x(1)+x(2)^4-16*x(2)^2+5*x(2))
Starting at point [-3, -3],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              -3.00e+00              -7.80e+01
                    -3.00e+00

    1          1     -2.90e+00   1.36e-01    -7.83e+01    4.95e+00       4     2.76e-02
                    -2.90e+00

               2     -2.90e+00   0.00e+00    -7.83e+01    0.00e+00       1     0.00e+00
                    -2.90e+00

    2          1     -2.90e+00   0.00e+00    -7.83e+01    0.00e+00       1     0.00e+00
                    -2.90e+00


x =

   -2.9035
   -2.9035
```

```
no_its =

     3


normg =

     0

--------------------------------------------------------------------------------------------
Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)
Initial point [-1.5, -1.5]
Using DFP method
Executing minimization of function @(x)1/2*(x(1)^4-16*x(1)^2+5*x(1)+x(2)^4-16*x(2)^2+5*x(2))
Starting at point [-1.5, -1.5],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init             -1.50e+00               -3.84e+01
                   -1.50e+00

    1          1     -2.16e+00   9.37e-01    -6.38e+01    2.79e+01       9     3.36e-02
                   -2.16e+00

               2     -2.90e+00   1.05e+00    -7.83e+01    2.39e+01       6     1.91e-01
                   -2.90e+00

    2          1     -2.90e+00   1.34e-07    -7.83e+01    5.12e-06       2     2.61e-02
                   -2.90e+00

               2     -2.90e+00   6.71e-09    -7.83e+01    7.11e-07       2     1.05e-02
                   -2.90e+00


x =

   -2.9035
   -2.9035


no_its =
    4


normg =

   7.1054e-07

-------------------------------------------------------------------------------------------
Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)
Initial point [-1.5, -1.5]
Using BFGS method
Executing minimization of function @(x)1/2*(x(1)^4-16*x(1)^2+5*x(1)+x(2)^4-16*x(2)^2+5*x(2))
Starting at point [-1.5, -1.5],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              -1.50e+00              -3.84e+01
                    -1.50e+00

    1          1     -2.16e+00   9.37e-01    -6.38e+01    2.79e+01       9     3.36e-02
                    -2.16e+00

               2     -2.90e+00   1.05e+00    -7.83e+01    2.39e+01       6     1.91e-01
                    -2.90e+00

    2          1     -2.90e+00   1.33e-07    -7.83e+01    4.14e-06       2     3.21e-02
                    -2.90e+00

               2     -2.90e+00   3.77e-12    -7.83e+01    1.00e-06       2     4.00e-06
                    -2.90e+00


x =

  -2.9035
  -2.9035


no_its =

    4


normg =
   1.0049e-06

-------------------------------------------------------------------------------------------
Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)
Initial point [-5, -5]
Using DFP method
Executing minimization of function @(x)1/2*(x(1)^4-16*x(1)^2+5*x(1)+x(2)^4-16*x(2)^2+5*x(2))
Starting at point [-5, -5],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              -5.00e+00              2.00e+02
                    -5.00e+00

    1          1     -2.90e+00   2.96e+00    -7.83e+01    2.37e+02       5     1.25e-02
                    -2.90e+00

               2     -2.90e+00   2.62e-07    -7.83e+01    2.71e-05       2     2.47e-01
                    -2.90e+00

    2          1     -2.90e+00   7.40e-07    -7.83e+01    2.61e-05       2     2.84e-02
                    -2.90e+00

               2     -2.90e+00   3.32e-09    -7.83e+01    7.11e-07       2     5.13e-03
                    -2.90e+00


x =

   -2.9035
   -2.9035


no_its =

     4


normg =

   7.1054e-07

-------------------------------------------------------------------------------------------
```

```
Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)
Initial point [-5, -5]
Using BFGS method
Executing minimization of function @(x)1/2*(x(1)^4-16*x(1)^2+5*x(1)+x(2)^4-16*x(2)^2+5*x(2))
Starting at point [-5, -5],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              -5.00e+00              2.00e+02
                    -5.00e+00

    1           1     -2.90e+00   2.96e+00    -7.83e+01    2.37e+02       5     1.25e-02
                    -2.90e+00

               2     -2.90e+00   2.62e-07    -7.83e+01    2.71e-05       2     2.47e-01
                    -2.90e+00

    2           1     -2.90e+00   7.40e-07    -7.83e+01    2.61e-05       2     2.84e-02
                    -2.90e+00

               2     -2.90e+00   1.26e-08    -7.83e+01    7.11e-07       2     1.95e-02
                    -2.90e+00


x =

   -2.9035
   -2.9035


no_its =

     4


normg =

   7.1054e-07

------------------------------------------------------------------------------------------
Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)
Initial point [-1, -1]
Using DFP method
Executing minimization of function @(x)1/2*(x(1)^4-16*x(1)^2+5*x(1)+x(2)^4-16*x(2)^2+5*x(2))
```

```
Starting at point [-1, -1],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              -1.00e+00              -2.00e+01
                    -1.00e+00

    1           1     -1.00e+00   0.00e+00    -2.00e+01    2.33e+01       1     0.00e+00
                    -1.00e+00


x =

    -1
    -1


no_its =

     1


normg =

   23.3345

--------------------------------------------------------------------------------------------
Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)
Initial point [-1, -1]
Using BFGS method
Executing minimization of function @(x)1/2*(x(1)^4-16*x(1)^2+5*x(1)+x(2)^4-16*x(2)^2+5*x(2))
Starting at point [-1, -1],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              -1.00e+00              -2.00e+01
                    -1.00e+00

    1           1     -1.00e+00   0.00e+00    -2.00e+01    2.33e+01       1     0.00e+00
                    -1.00e+00


x =
    -1
    -1


no_its =

     1


normg =

   23.3345

------------------------------------------------------------------------------------------
Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)
Initial point [1, 1]
Using DFP method
Executing minimization of function @(x)1/2*(x(1)^4-16*x(1)^2+5*x(1)+x(2)^4-16*x(2)^2+5*x(2))
Starting at point [1, 1],
Using DFP method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +1.00e+00              -1.00e+01
                    +1.00e+00

    1          1     +1.00e+00   0.00e+00    -1.00e+01    1.63e+01       1     0.00e+00
                    +1.00e+00


x =

     1
     1


no_its =

     1


normg =

   16.2635
```

```
------------------------------------------------------------------------------------------
Minimizing Styblinski-Tang function, min at (-2.904, ..., -2.904)
Initial point [1, 1]
Using BFGS method
Executing minimization of function @(x)1/2*(x(1)^4-16*x(1)^2+5*x(1)+x(2)^4-16*x(2)^2+5*x(2))
Starting at point [1, 1],
Using BFGS method.

outer it.   iteration     x       step size     f(x)     norm(grad)  ls iters   lambda
   init              +1.00e+00              -1.00e+01
                    +1.00e+00

    1          1     +1.00e+00   0.00e+00    -1.00e+01    1.63e+01       1     0.00e+00
                    +1.00e+00


x =

     1
     1


no_its =

     1


normg =

   16.2635

diary off
```