Project - Optimization I

Candidate number: 10001

April 12, 2026

1

Implementation is found in the Exercise1-folder in the .zip-file. A visualization of the robot arm is in Figure 1 with the following inputs:

$$\ell = (4, 3, 2, 2), \quad \vartheta = (\pi/5, \pi/4, \pi/3, \pi/2)$$

The script also returns two vectors X and Y, which makes up the points in the plane that defines the robot-arm.

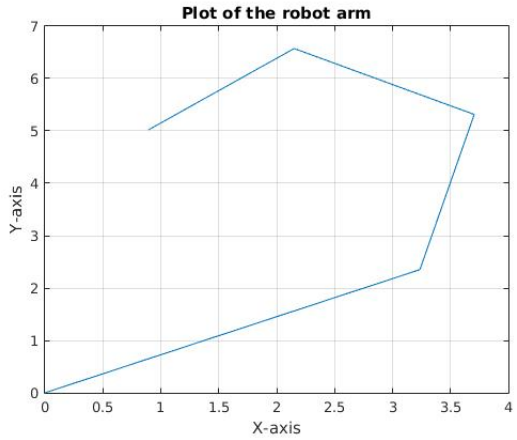

Figure 1: Demonstration of the plot of a robot arm

2

Observe that

$$d(\vartheta) = \frac{1}{2}\|F(\vartheta) - p\|^2 = \frac{1}{2}(F(\vartheta)^T F(\vartheta) - 2F(\vartheta)^T p + p^T p)$$

Let F_1 and F_2 denote the two components of F. We then calculate the i'th component of the gradient:

$$\left(\nabla d(\vartheta)\right)_i = \frac{\partial F_1}{\partial \vartheta_i}\left(F_1(\vartheta) - p_1\right) + \frac{\partial F_2}{\partial \vartheta_i}\left(F_2(\vartheta) - p_2\right),$$

where

$$\frac{\partial F_1}{\partial \vartheta_i} = \sum_{k=i}^{n} \ell_k \cos\left(\sum_{j=1}^{k} \vartheta_j\right), \qquad \frac{\partial F_2}{\partial \vartheta_i} = \sum_{k=i}^{n} \ell_k \sin\left(\sum_{j=1}^{k} \vartheta_j\right).$$

Hence, we have an expression for the i'th component for the gradient of d. Since i was chosen arbitrarily, the other components can be computed similarly.

3

Showing that $d(\vartheta)$ is non-convex requires finding some values x, y and $\alpha \in [0,1]$ for which

$$d(\alpha x + (1-\alpha)y) > \alpha d(x) + (1-\alpha)d(y) \tag{1}$$

holds. Since we only need a counterexample, we restrict ourselves to the one-dimensional case. Then we get that

$$d(\vartheta) = \frac{1}{2}\left(\ell^2 - 2p_1\ell\sin\vartheta - 2p_2\ell\cos\vartheta + p_1^2 + p_2^2\right)$$

To keep things simple as possible, we set α equal to $1/2$. This gives that

$$d\left(\frac{x+y}{2}\right) = \frac{1}{2}\left(\ell^2 - 2p_1\ell\sin\frac{x+y}{2} - 2p_2\ell\cos\frac{x+y}{2} + p_1^2 + p_2^2\right),$$

$$\frac{1}{2}d(x) = \frac{1}{4}\left(\ell^2 - 2p_1\ell\sin x - 2p_2\ell\cos x + p_1^2 + p_2^2\right),$$

$$\frac{1}{2}d(y) = \frac{1}{4}\left(\ell^2 - 2p_1\ell\sin y - 2p_2\ell\cos y + p_1^2 + p_2^2\right).$$

This reduces Inequality 1 to

$$2p_1\sin\frac{x+y}{2} + 2p_2\cos\frac{x+y}{2} < p_1\sin x + p_2\cos x + p_1\sin y + p_2\cos y \tag{2}$$

We first assume that $(p_1, p_2) \neq (0,0)$. If p_2 is positive, setting $(x,y) = (2\pi, 0)$ gives that $-2p_2 < 2p_2$. If p_2 is negative, chosing $(x,y) = (3\pi, \pi)$ gives that $2p_2 < -2p_2$. If p_2 is zero, then assume that p_1 is positive. Choose $(x,y) = (\pi/2, 5\pi/2)$ gives that $-2p_1 < 2p_1$. If p_1 is negative, choosing $(x,y) = (-\pi/2, 3\pi/2))$ gives that $2p_1 < -2p_1$. The case where $p = (0,0)$ remains, but then every term cancels and it is not possible to prove non-convexity. Hence, $d(\vartheta)$ is non-convex for all points except $(0,0)$.

4

To solve

$$\min d(\vartheta), \quad \text{with} \quad d(\vartheta) = \frac{1}{2}\|F(\vartheta) - p\|^2 \tag{3}$$

I have implemented an algorithm based on the BFGS-method with line search using the Wolfe-conditions and random starting value for φ. By default for all BFGS-methods in the project, I have primarily used $(c_1, c_2) = (0.1, 0.4)$ in the line search to make sure that the trust region exists. Implementation can be found in the Exercise4-folder. We consider the test input

$$n = 4, \quad \ell = (3,2,1,1), \quad p = (0,0),$$

for which the program outputs the solution

$$\vartheta = (1.5342, 3.7950, 4.5470, 0.7437).$$

Since the starting value is random, one would not expect the same output each time. The norm of the gradient is plotted against the number of iterations in Figure 2 along with the final robot arm-plot. We see in the figure that the BFGS-method uses nine iterations to approximate the solution of Equation 3 with precision close to 10^{-10}. The other test cases can be computed similarly by following the instructions given in exercise4.m in the Exercise4-folder.

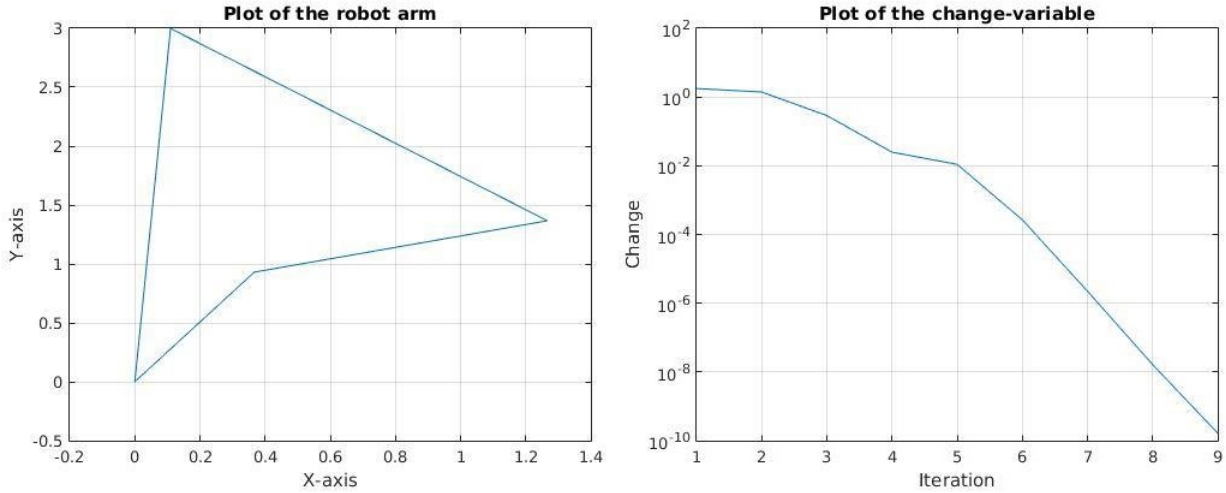

Figure 2: The left plot shows the robot-arm. The right plot shows the value of the norm of the gradient plotted against the number of iterations.

5

We see that

$$Q(\varphi;\mu) = \frac{1}{2}\varphi^T\varphi + \frac{\mu}{2}\left(F(\vartheta+\varphi)-q\right)^T\left(F(\vartheta+\varphi)-q\right).$$

Differentiating with respect to φ_i we get that

$$\frac{\partial Q}{\partial \varphi_i} = \varphi_i + \mu\frac{\partial F_1}{\partial \varphi_i}\left(F_1(\vartheta+\varphi)-q_1)\right) + \mu\frac{\partial F_2}{\partial \varphi_i}\left(F_2(\vartheta+\varphi)-q_2)\right)$$

From this I have implemented a quadratic penalty method based on the BFGS-method that uses line

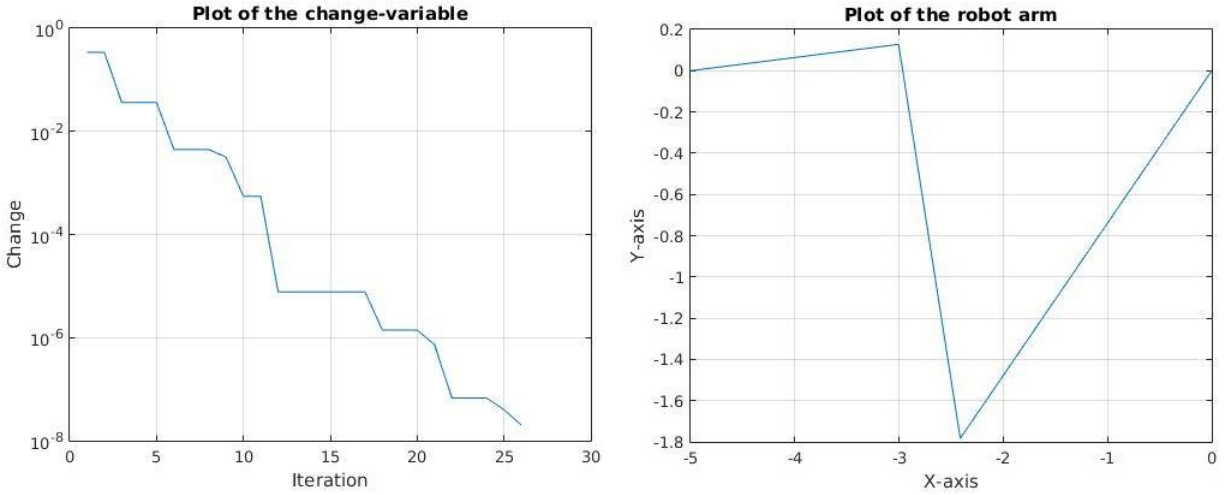

Figure 3: The left plot shows the robot-arm. The right plot shows the value of the norm of the gradient plotted against the number of iterations.

search satisfying the Wolfe conditions. Implementation is contained in the Exercise5-folder. For each iteration of the penalty method I scale μ and τ by two and one half respectively. The line search seems in some cases to return small values for α or converge to some fixed value, for which the reason is unclear. The script takes this however in consideration for stability, but it will still on the other hand run forever in some cases for μ-variable becomes big (around 10^8). This is expected as the penalty method is known to

become inaccurate for values μ where $1/\mu^2 \approx \epsilon$, where ϵ is the machine epsilon which equals $2.2204 \cdot 10^{-16}$ in MATLAB. We now consider the last test case given in the project:

$$n = 3, \quad \ell = (3, 2, 2), \quad \vartheta = (0, 0, 0), \quad q = (-5, 0).$$

The implementation outputs the approximate solution $\varphi = (3.7780, 4.3762, 1.3346)$. The corresponding convergence plot and plot of the robot arm is given in Figure 3. We have set the μ and the τ-variable to shrink by a factor of two in each iteration of the quadratic penalty method, so we expect the change-variable to decrease meanly by a factor of two in each iteration, which the plot verifies. It is important to mention that the plot only shows the iterations for the quadratic penalty method and when we rescale μ and τ. The convergence plot in Figure 2 (without the penalty method) therefore tells more about the convergence of the method.

6

We start by assuming that $\varphi = \varphi^{(i)} = \varphi^{(j)}$ for all i, j and that all $\varphi^{(i)}$ are solutions of (2) in the project. Then we have that

$$d(\varphi) = \frac{1}{2}\|\varphi\|^2$$

is minimized and the following is true:

$$F(\vartheta + \varphi) = q$$

We use that all $\varphi^{(i)}$ are equal and put this into (3) in the project, which leads us to the equivalent equations as in (2):

$$\frac{1}{2}\|\sum_{i=1}^{s}\frac{1}{s}\varphi^{(i)}\|^2 = \frac{1}{2}\|\frac{1}{s}\sum_{i=1}^{s}\varphi\|^2 = \frac{1}{2}\|\varphi\|^2$$

$$F\left(\vartheta + \frac{1}{s}\sum_{i=1}^{s}\varphi^{(i)}\right) = F\left(\vartheta + \varphi\right) = q$$

Since this is the minimization of (2), the first way is proved.

Now suppose $(\varphi^{(i)})_{i=1}^{s}$ is a solution of (3). The idea of the proof is to show that the quadratic sum of the i'th coordinates is minimized when all of the i'th coordinates are equal. Since an identical argument can be done on all other coordinates, it will follow that $\varphi^{(i)} = \varphi^{(j)}$ for all $1 \leq i, j \leq s$. First define $x_i \in \mathbb{R}$ to be the mean of the i'th coordinates:

$$x_i = \frac{1}{s}\sum_{j=1}^{s}\varphi_i^{(j)} \tag{4}$$

We want to minimize the expression

$$\min \frac{1}{s}\|\varphi_i\|^2 = \frac{1}{s}\sum_{j=1}^{s}\left(\varphi_i^{(j)}\right)^2 \tag{5}$$

with respect to the constraint in Equation 4. Let $\{a_i^{(j)}\}_{j=1}^{s}$ be the number $\{\varphi_i^{(j)}\}_{j=1}^{s}$ differs from x_i respectively. Then we have that

$$x_i = \frac{1}{s}\sum_{j=1}^{s}\varphi_i^{(j)} = \frac{1}{s}\sum_{j=1}^{s}(x_i + a_i^{(j)}) = x_i + \frac{1}{s}\sum_{j=1}^{s}a_i^{(j)}, \qquad a_i^{(j)} \in \mathbb{R}.$$

It follows that

$$\sum_{j=1}^{s}a_i^{(j)} = sx_i - sx_i = 0$$

Then we can reduce Equation 5:

$$\frac{1}{s}\sum_{j=1}^{s}\left(\varphi_i^{(j)}\right)^2 = \frac{1}{s}\sum_{j=1}^{s}(x_i - a_i^{(j)})^2 = x_i^2 - \frac{2x_i\sum_{j=1}^{s}a_i^{(j)}}{s} + \frac{1}{s}\sum_{j=1}^{s}\left(a_i^{(j)}\right)^2$$

$$= x_i^2 + \frac{1}{s}\sum_{j=1}^{s}\left(a_i^{(j)}\right)^2$$

With x_i being fixed, this is clearly minimized when all $a_i^{(j)}$ equals zero, which shows that all the i'th coordinates equals x_i. Since i was chosen arbitrarily, this will holds for all the other coordinates. It follows that all the vectors $\left(\varphi^{(i)}\right)_{i=1}^{s}$ are equal. We have from Equation (3) in the project that

$$F(\vartheta + \frac{1}{s}\sum_{i}^{s}\varphi^{(i)}) = F(\vartheta + \frac{1}{s}\sum_{i}^{s}\varphi) = F(\vartheta + \varphi),$$

so each φ-vector is a solution of (2) in the project. Hence, the other way is proved.

7

We have that

$$D((\varphi^{(i)})_{i=1}^{s}) = \frac{1}{2}\|F(\vartheta + \frac{1}{s}\sum_{i=1}^{s}\varphi^{(i)}) - q\|^2$$

Then we see that

$$\frac{\partial D}{\partial \varphi_m^{(j)}} = \frac{\partial F_1}{\partial \varphi_m^{(j)}}(F_1 - q_1) + \frac{\partial F_2}{\partial \varphi_m^{(j)}}(F_2 - q_2) \tag{6}$$

Observe that

$$F_1(\vartheta + (\varphi^{(i)})_{j=1}^{s}) = \sum_{j=1}^{n}\ell_j\left(\cos\left(\sum_{l=1}^{j}\left(\vartheta_l + \frac{1}{s}\sum_{k=1}^{s}\varphi_l^{(k)}\right)\right)\right)$$

$$F_2(\vartheta + (\varphi^{(i)})_{j=1}^{s}) = \sum_{j=1}^{n}\ell_j\left(\sin\left(\sum_{l=1}^{j}\left(\vartheta_l + \frac{1}{s}\sum_{k=1}^{s}\varphi_l^{(k)}\right)\right)\right)$$

Then we have for $1 \leq i \leq s$, $1 \leq m \leq n$, that

$$\frac{\partial F_1}{\partial \varphi_m^{(i)}} = -\frac{1}{s}\sum_{j=m}^{n}\ell_j\left(\sin\left(\sum_{l=1}^{j}\left(\vartheta_l + \frac{1}{s}\sum_{k=1}^{s}\varphi_l^{(k)}\right)\right)\right)$$

$$\frac{\partial F_2}{\partial \varphi_m^{(i)}} = \frac{1}{s}\sum_{j=m}^{n}\ell_j\left(\cos\left(\sum_{l=1}^{j}\left(\vartheta_l + \frac{1}{s}\sum_{k=1}^{s}\varphi_l^{(k)}\right)\right)\right).$$

Hence, we have found the (m,i)'th component of the gradient. As there is no restrictions for our choice of m and i, the other components can be computed similarly. Hence, we have calculated the gradient.

8

Consider the quadratic penaly method on the function

$$Q((\varphi^{(i)})_{i=1}^{s};\mu) = \frac{1}{2s}\sum_{i=1}^{s}\left\|\varphi^{(i)}\right\|^2 + \frac{\mu}{2}\left\|F\left(\vartheta + \frac{1}{s}\sum_{i=1}^{s}\varphi^{(i)}\right) - q\right\|^2$$

Differentiating with respect to the j'th component of the i'th φ-vector yields that

$$\frac{\partial Q}{\partial \varphi_j^{(i)}} = \frac{\varphi_j^{(i)}}{s} + \mu\frac{\partial D}{\partial \varphi_j^{(i)}} = \frac{\varphi_j^{(i)}}{s} + \mu\frac{\partial F_1}{\partial \varphi_j^{(i)}}(F_1 - q_1) + \mu\frac{\partial F_2}{\partial \varphi_j^{(i)}}(F_2 - q_2).$$

The implementation is found in the Exercise8-folder. Algorithmically the implementation is almost identical as in Exercise 5, but it takes in consideration several φ-vectors. We must therefore use the reshape-function in MATLAB to make turn φ into one large vector for it to compile with the previous code. Consider the input

$$n = 3, \quad s = 20, \quad \ell = (3,2,2), \quad \vartheta = (\pi/2, \pi/2, \pi/2), \quad q = (2,1),$$

all the φ-vectors is approximately equal to $(4.0550, 1.5588, 3.17777)$, which results in the total approximate solution $(5.6158, 3.1296, 4.7485)$. The plot and the convergence is found in Figure 4 To verify that all

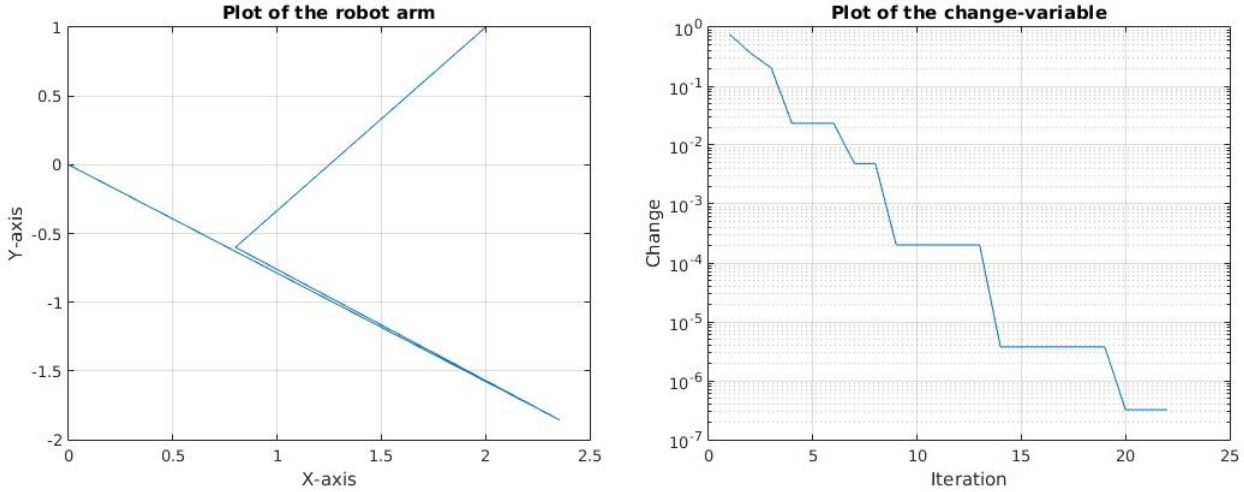

Figure 4: The left plot shows the final step of the robot arm for the given input. The right plot shows the value for the variable change for each step the tau-variable shrinks 22 times.

the vectors $\varphi^{(i)}$, $1 \leq i \leq s$ equals each other numerically, consider the mean of all the coordinates as a reference vector. Subtract this vector to all of the φ-vectors and calculate the total sum of the norms of the resulting vectors. The result for each iteration is shown in Figure 5, and the total norm of our vectors ends up having a relative error below 10^{-10} in the last iteration. The other test cases yields similar results and can be tested by following the instructions given in exercise8.m. This verifies the result in Exercise 6.

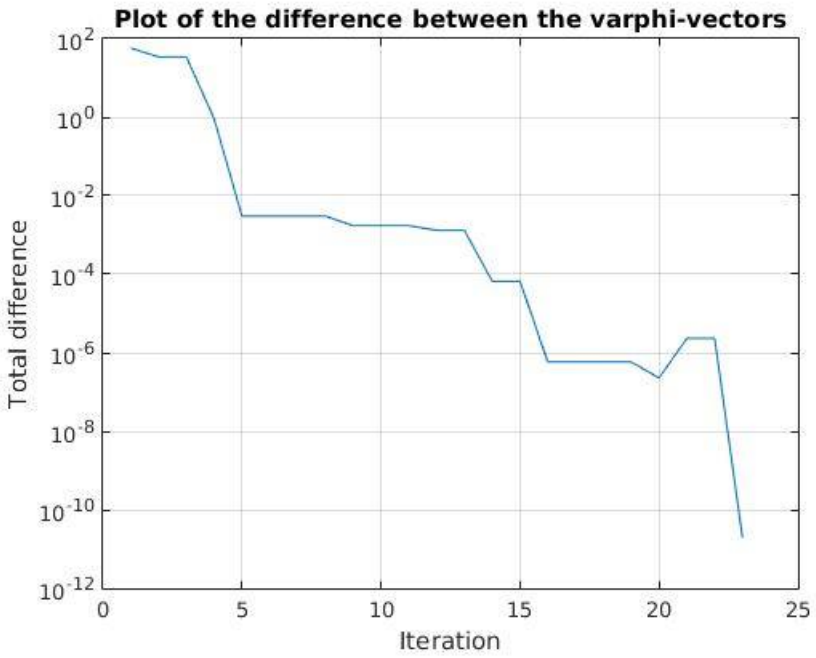

Figure 5: This plots measures the difference between the φ-vectors for each iteration in Exercise 8. This is done by calculating the mean φ-vector and then subtracting this to each one of them. Then I sum up the norm for each resulting vector.

9

Let $Lb((\varphi^{(i)})_{i=1}^{s}; \mu, \beta)$ be defined as in the project. The Jacobian of Lb can be calculated as follows:

$$\frac{\partial Lb}{\partial \varphi_a^{(b)}} = \frac{\partial}{\partial \varphi_a^{(b)}} \sum_{k=1}^{s} \sum_{j=1}^{n} \left[\log\left(\frac{3\pi}{4} + \vartheta_j + \frac{1}{s} \sum_{i=1}^{k} \varphi_j^{(i)} \right) + \log\left(\frac{3\pi}{4} - \vartheta_j - \frac{1}{s} \sum_{i=1}^{k} \varphi_j^{(i)} \right) \right]$$

We are interested in the terms that doesn't disappear under differentiation. Then the conditions that $k \geq b$ and $j = a$, must be fulfilled simultaneously. Cancelling the terms that equals zero results in the following expression

$$\begin{aligned} \frac{\partial Lb}{\partial \varphi_a^{(b)}} &= \frac{\partial}{\partial \varphi_a^{(b)}} \sum_{k=b}^{s} \left[\log\left(\frac{3\pi}{4} + \vartheta_a + \frac{1}{s} \sum_{i=1}^{k} \varphi_a^{(i)} \right) + \log\left(\frac{3\pi}{4} - \vartheta_a - \frac{1}{s} \sum_{i=1}^{k} \varphi_a^{(i)} \right) \right] \\ &= \frac{1}{s} \sum_{k=b}^{s} \left[\frac{1}{\left(\frac{3\pi}{4} + \vartheta_a + \frac{1}{s} \sum_{i=1}^{k} \varphi_a^{(i)} \right)} - \frac{1}{\left(\frac{3\pi}{4} - \vartheta_a - \frac{1}{s} \sum_{i=1}^{k} \varphi_a^{(i)} \right)} \right] \end{aligned}$$

For the implementation I have used a random starting point, relying heavily on the implementation in Exercise 8, but taken the logarithmic barrier-term in consideration. The value β is set to equal the inverse of μ as μ increases. The line search seems to fail for which the alpha-value converges towards some fixed value, or becomes very small. If α gets very small, y_k and s_k becomes too big MATLAB fails to approximate the Hessian in the BFGS-method. This was taken in consideration in Exercise 5 and Exercise 8 by either putting α, s_k or y_k to have random components between zero and one, but this is no longer possible as the step size must be chosen carefully. I did not manage to safeguard against this. The attempted implementation is in the Exercise9-folder.