Assignment 1

Jakub Czerny - s161200
Oskar Hint - s161559
Joachim Finn Jensen - s134052

02612 Constrained Optimization

Department of Applied Mathematics and Computer Science
Danmarks Tekniske Universitet
Lyngby, March 2017

Contents

1 Assignment

1.1 Problem 1 - Quadratic Optimization

From the problem 1 it is seen that the system can be written as:

$$H = \begin{bmatrix} 6 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 4 \end{bmatrix}$$

$$g = \begin{bmatrix} -8 \\ -3 \\ -3 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

Then the general KKT system can be written as:

$$\begin{bmatrix} H & -A \\ A^T & 0 \end{bmatrix} \times \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} -g \\ b \end{bmatrix}$$

Which in this case can be written as:

$$\begin{bmatrix} 6 & 2 & 1 & -1 & 0 \\ 2 & 5 & 2 & 0 & -1 \\ 1 & 2 & 4 & -1 & -1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 8 \\ 3 \\ 3 \\ 3 \\ 0 \end{bmatrix}$$

The system is solved in MatLab by making a function that solves the problem by use of the backslash command, see the code how it was done. The result shows that:

$$x = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$

$$\lambda = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

To test the solver, random values for x and λ are generated, from these the H, g, A and b are determined. With these matrix and vectors the system are solved again. If the function finds the same x and lambda that were previously randomly chosen, the solver works. Our solver works as expected and gives the same values for x and lambda that were generated.
To see the sensitivity of the problem parameter sensitivity analysis is conducted. The general function is as followed.

$$F(x, \lambda; p) = \begin{bmatrix} \nabla_x f(x,p) - \nabla_x c(x,p)\lambda \\ c(x,p) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Where the implicit function theorem gives parameter sensitivities

$$[\nabla_x(p)\lambda(p)] = -[W_{xp} - \nabla_p c(x,p) \begin{bmatrix} W_{xx} & -\nabla_x c(x,p) \\ -\nabla_x c(x,p)^T & 0 \end{bmatrix}^{-1}$$

$$W_{xx} = \nabla^2_{xx} f(x,p) - \sum_{i \in \varepsilon} \lambda_i \nabla^2_{xx} c_i(x,p)$$

$$W_{xp} = \nabla^2_{xp} f(x,p) - \sum_{i \in \varepsilon} \lambda_i \nabla^2_{xp} c_i xp)$$

From this the sensitivity of this problem can be written as:

$$f(x,p) = \frac{1}{2} x^T H x + (g + \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix})^T x$$

$$c(x,p) = A^T x - b - \begin{bmatrix} p_5 \\ p_6 \end{bmatrix}$$

$$w_{xp} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_{xx} = H$$

$$[\nabla x(p)\lambda(p)] = -\begin{bmatrix} H & -A^T \\ -A & 0 \end{bmatrix}$$

This are added to the solver and the sensitivity found are the following:

$$\nabla x(p) = \begin{bmatrix} -0.0769 & -0.0769 & 0.0769 \\ -0.0769 & -0.0769 & 0.0769 \\ 0.0769 & 0.0769 & -0.0769 \\ 0.4615 & -0.5385 & 0.5385 \\ -0.3846 & 0.6154 & 0.3846 \end{bmatrix}$$

$$\lambda(p) = \begin{bmatrix} 0.4615 & -0.3846 \\ -0.5385 & 0.6154 \\ 0.5385 & 0.3846 \\ 2.2308 & -0.6923 \\ -0.6923 & 3.0769 \end{bmatrix}$$

These sensitivities can be verified by using the following equations:

$$x(p) \approx x(p_0) + \nabla_p x(p_0)^T (p - p_0)$$

$$\lambda(p) \approx \lambda(p_0) + \nabla_p \lambda(p_0)^T (p - p_0)$$

These are equations(1.92) and (1.93) in Numerical Methods for Constrained Optimization by John Bagterp Joergensen.
When dealing with such a quadratic problem we can always also write the dual problem. Since the Lagrangian for the primal is:

$$L(x,\lambda) = \frac{1}{2} x^T H x + g^T - \mu^T (A^T x - b)$$

The dual problem is:

$$\max_{x,\mu} -\frac{1}{2} x^T H x + b\mu^T$$

$$s.t. Hx + g - A\mu = 0$$

This maximization problem can be converted to a minimization problem by multiplying by -1. This gives the final dual problem:

$$\min_{x,\mu} \frac{1}{2} x^T H x - b\mu^T$$

$$s.t. Hx + g - A\mu = 0$$

In the dual μ is the feasible point where in the primal it is x. They are related as so that the primal optimal is less or equal to the dual optimal. But if one of them have finite optima, primal optimal equals dual optimal. In some cases the dual problem can be simpler and therefore easier to solve the primal.

1.2 Problem 2 - Equality Constrained Quadratic Optimization

The second problem that this report deals with has been defined in a following way:

$$\begin{aligned} \min_u \frac{1}{2}\sum_{i=1}^{n+1}(u_i-\bar{u})^2 \\ s.t. \quad -u_1+u_n=-d_0 \\ u_i-u_{i+1}=0 \\ u_{n-1}-u_n-u_{n+1}=0 \end{aligned} \tag{1}$$

The first step we took was translation of the problem into convenient matrix notation:

$$\begin{aligned} \min_x f(x)=\frac{1}{2}x'Hx+g'x \\ s.t. \quad A'x=b, \end{aligned} \tag{2}$$

which for $n=10$, $\bar{u}=0.2$ and $d_0=1$ looks as follows:

$$H=\begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}$$

$$g=\begin{bmatrix} 0.4 \\ . \\ . \\ . \\ 0.4 \end{bmatrix}$$

$$A=\begin{bmatrix} \text{-1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \text{-1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \text{-1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \text{-1} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \text{-1} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \text{-1} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \text{-1} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{-1} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{-1} & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{-1} \end{bmatrix}$$

$$b=\begin{bmatrix} \text{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

The Lagrangian can then be written as a function of n, $\bar{u}$ and d_0:

$$L=\frac{1}{2}\sum_{i=1}^{n+1}(u_i-\bar{u})^2-\sum_{i=1}^{n-2}\lambda_i(u_i-u_{i+1})-\lambda_{n-1}(-u_1+u_n+d_0)-\lambda_n(u_{n-1}-u_n-u_{n+1}). \tag{3}$$

or in matrix notation

$$L=\frac{1}{2}x'Hx+g'x-\lambda(A'x-b). \tag{4}$$

This can be further used to form the KKT first order optimality conditions

$$\begin{aligned} \nabla L=Hx+g-A\lambda=0 \\ s.t. \quad A'x-b=0, \end{aligned} \tag{5}$$

that are necessary but in this case also sufficient for the optimality. To prove that we have to show that f(x) is a convex function(because of minimization) and that active inequality constraints are convex. Since there are no inequality constraints (there is nothing to make them concave) and equality constraints are affine (linear) the latter requirement is met. Regarding the objective function, we can look at the coefficient by the $2^(nd)$ order term and check its sign. In our case by the highest order term u we see implicit 1 which means its a convex function, therefore we have proven that KKT is necessary and sufficient for the problem at hand.

Next, we implemented multiple variations of the solvers that use different matrices factorizations, procedures but also store the matrices in dense or sparse manner. All the algorithms however output the same results being

$$x = \begin{bmatrix} -0.3000 & -0.3000 & -0.3000 & -0.3000 & -0.3000 & -0.3000 & -0.3000 & -0.3000 & -0.3000 & -1.3000 & 1.0000 \end{bmatrix}^T$$

$$\lambda = \begin{bmatrix} -2.3000 & -2.2000 & -2.1000 & -2.0000 & -1.9000 & -1.8000 & -1.7000 & -1.6000 & -1.5000 & -1.4000 \end{bmatrix}^T$$

Once this step has been completed, we compared the performances of the solvers based on LU, LDL, Null-Space and Range-Space factorizations. For this purpose we measured the CPU time, so the time that the processing unit needed for finding the solution - that in case of multi-core processors may not be the same as the wall time - the actual time that the calculations took. By using the CPU time, we are able to objectively compare the performance of the algorithms and see how their pure implementations behave when varying the problem size. The obtained results are shown on Figure 1.

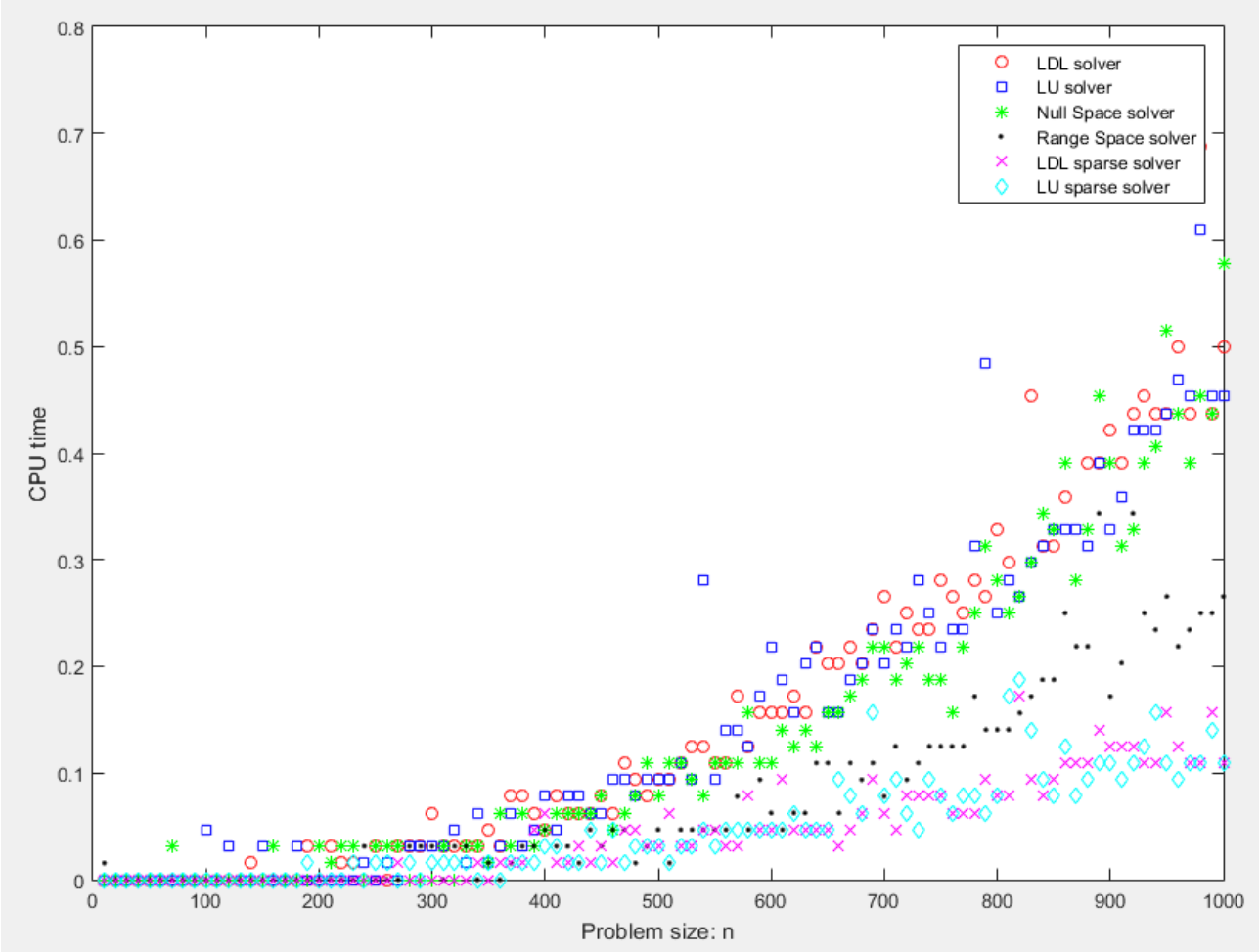

Figure 1: Comparison of multiple QP solvers

From theory we would expect the LDL-based solver to be superior over LU as it requires less operations but for tested problem sizes they perform very comparably, with LDL being more stable. Interestingly, these two solvers seems to be outperformed by Null-Space solver (at least for small problem sizes) that is using QR-factorization, which is in fact slightly more computationally expensive. Further we see that Range-Space solver

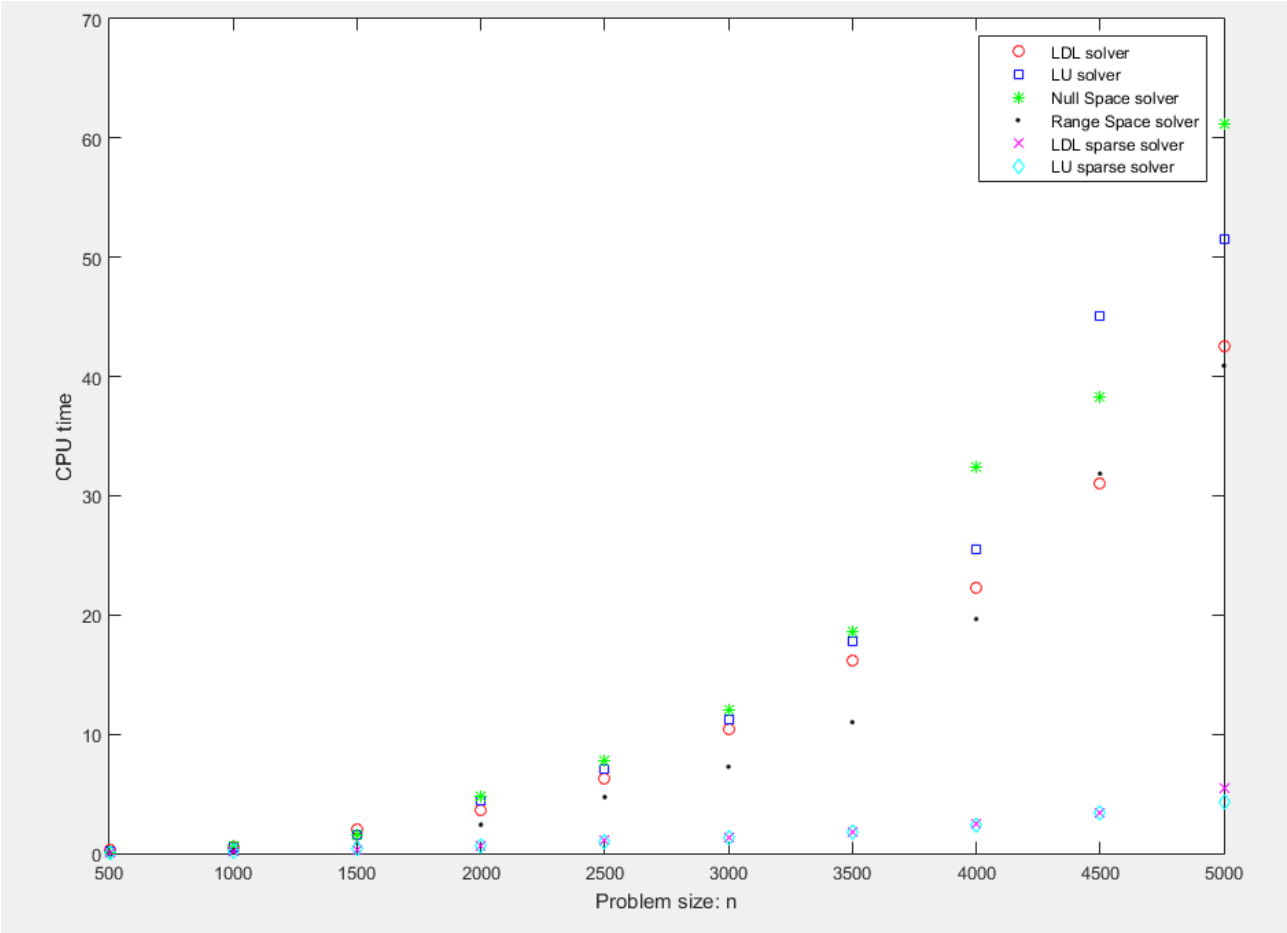

Figure 2: QP solvers with increased problem sizes

is a lot faster than previously presented methods, but we also know that it uses sparse structure for hessian matrix that is needed for Cholesky factorization if we want it to output 3 values in MatLab.

Since the results for 3 dense solvers seemed a bit surprising, but also the problem sizes were very small, we decided to increase them up to 4000 and see if our expectations would be reflected by the times that solvers took to solve the problem. The results of this test can be seen on Figure 2. As a rule of thumb, when we want to compare the runtimes of some programs/algorithms we should either repeat the experiments multiple time and take the average or as a good practice, make sure the runtime exceeds 1s and the initialization processes are not taking the majority of the runtime.

Based on the result of Range-Space method, we already suspect that other sparse-based solvers will outperform all above mentioned implementations as it uses less Cholesky factorization that is a bit less efficient than both LDL and LU. To support this claim we first investigated the sparsity pattern of KKT matrix and it looks as presented on Figure 3

This tells us that great majority of the KKT entires are 0's and therefore we could really take advantage of storing it as a sparse matrix. When we look at Figure 1 or 2, we clearly see that the variations using sparse matrices turned out to be superior. It should be also mentioned that for Range-Space solver, we make hessian (H) and matrix A sparse whereas in LDL and Lu solvers the entire KKT matrices are being converted into sparse structures and that might be the reason why there is this quite a big gap between these 3 solvers.

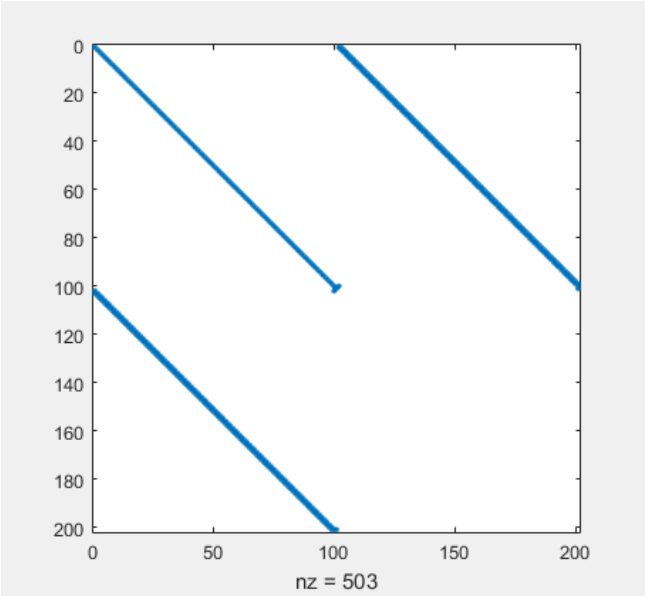

Figure 3: Sparsity pattern of KKT matrix

1.3 Problem 3 - Inequality Constrained Quadratic Programming

From page 475 in Nocedal and Wright the following quadratic program is given.

$$\begin{aligned} \min_x q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2 \\ s.t. \quad x_1 - 2x_2 + 2 >= 0, \\ -x_1 - 2x_2 + 6 >= 0, \\ -x_1 + 2x_2 + 2 >= 0, \\ x_1 >= 0, \\ x_2 >= 0. \end{aligned} \tag{6}$$

Make a contour plot of the problem.

Contour plot of the problem can be seen on figure 4. As apparent from the plot, most of the search region is cut off by the constraints, feasible region can be seen in white.

Write the KKT-conditions for this problem.

Using matrix notation the initial problem can be written as

$$\begin{aligned} \min_x f(x) = \frac{1}{2} x'Hx + g'x \\ s.t. \quad A'x \geq b, \end{aligned} \tag{7}$$

$$H = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$g = \begin{bmatrix} -2 \\ -5 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & -1 & -1 & 1 & 0 \\ -2 & -2 & 2 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} -2 \\ -6 \\ -2 \\ 0 \\ 0 \end{bmatrix}$$

Point x^* is a solution if the KKT conditions are satisfied

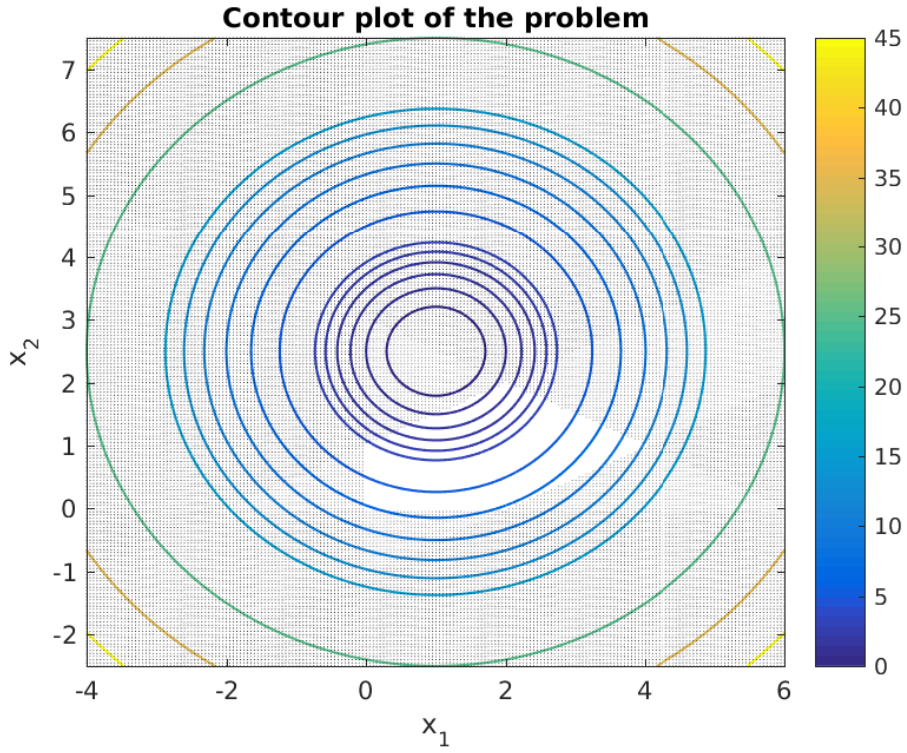

Figure 4: Contour plot with constraints

$$\begin{aligned} Hx^* + g - A\lambda &= 0 \\ a_i' x^* &= b_i \quad \text{for all i} \in A(x^*) \\ a_i' x^* &\geq b_i \quad \text{for all i} \in I \setminus A(x^*) \\ \lambda_i^* &\geq 0 \quad \text{for all i} \in A(x^*) \end{aligned} \tag{8}$$

where $A(x^*)$ is the active set at point x^*.

In other words x^* is a solution if gradient of the Lagrangian at that point is zero, active constraints hold with equality and inactive constraints hold i.e. x^* is feasible, and lambdas are non-negative meaning it is not possible to make a step that reduces the objective while still staying feasible.

Argue that the KKT-conditions are both necessary and sufficient optimality conditions.

Theorem 16.4 p.464 in N&W and its conclusion state that if x^* satisfies the KKT conditions specified above and H is positive semidefinite then x^* is the unique global solution. Since H is positive definite (identity matrix multiplied by 2) and therefore also positive semidefinite, then x^* is a global unique solution provided the KKT conditions are satisfied.

Make a function for solution of convex equality constrained QPs (see Problem 1 and Problem 2).

We used the same EqualityQPSolver function defined in problem 1. That solves the KKT system in form

$$\begin{bmatrix} H & -A \\ -A^T & 0 \end{bmatrix} \times \begin{bmatrix} x \\ \lambda \end{bmatrix} = - \begin{bmatrix} g \\ b \end{bmatrix}$$

Apply a conceptual active set algorithm to the problem. Use the iteration sequence in Figure 16.3 of Nocedal and Wright. Plot the iterations sequence in your contour plot. For each iteration (guess of working set) you should list the working set, the solution, x, and the Lagrange multipliers, λ.

The active set algorithm iteration sequence can be seen in table 1 and on figure 5. Column x_k represents the position at the start of an iteration. Column $\alpha_k * p_k'$ represents the scaled step made in corresponding iteration.

Table 1: Active set iterations

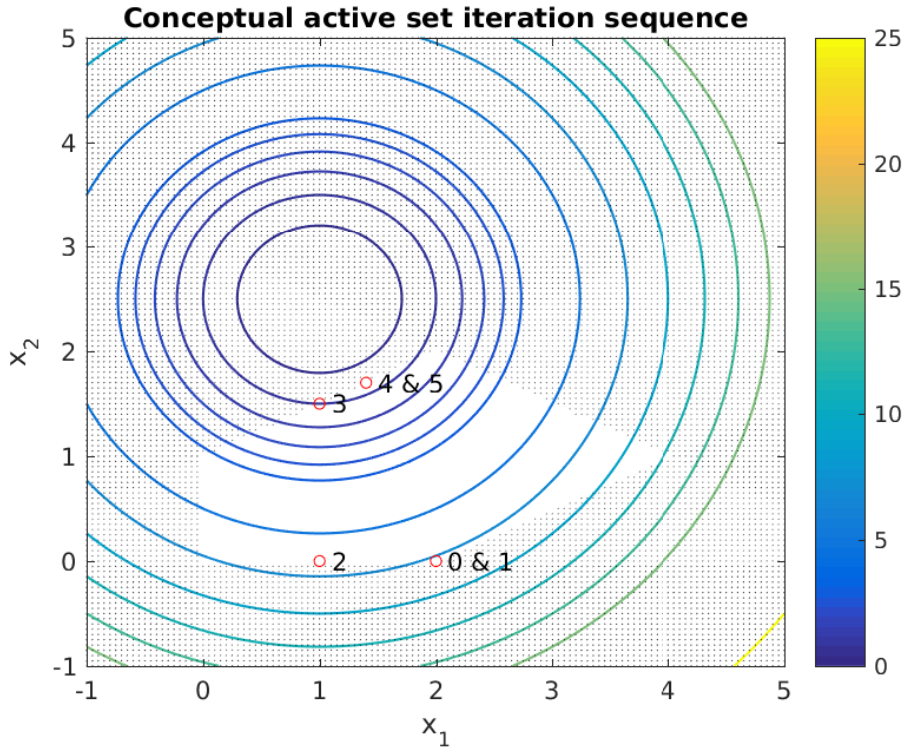

Figure 5: Conceptual active set iteration sequence

iteration	W	x'_k	$\alpha_k * p'_k$	$\hat{\lambda}'$
1	3,5	[2, 0]	[0, 0]	[-2, -1]
2	5	[2, 0]	[-1, 0]	[-5]
3	$\emptyset$	[1, 0]	[0, 1.5]	[]
4	1	[1, 1.5]	[0.4, 0.2]	[0.8]
5	1	[1.4, 1.7]	[0, 0]	[0.8]

With the first iteration, a constraint is removed from the working set (step 0, remove negative λ constraint), with the second iteration a step is made and a constraint is removed (step not 0, remove negative λ constraint), with the third iteration a step is made and a blocking constraint is added to working set. With the fourth iteration a step is made and with the fifth iteration the algorithm stops (zero step, no negative λ).

Comment on the Lagrange multipliers at each iteration.

Negative multipliers indicate that moving away from the corresponding constraint and towards the feasible region lowers the objective. The multipliers in first iteration are both negative, looking at the plot this makes sense. The starting point is at the intersection of two constraints that would not be part of the optimal active set, we could remove either for next iteration.

The Lagrange multiplier for the second iteration is -5, so moving away from this constraint likely yields a large decrease in the objective. Looking at the plot that is indeed the case, in fact, the step away from this constraint is in exactly opposite direction to the gradient of the objective function, i.e. the step is in direction of maximal rate of descent for the unconstrained problem. Since iteration 3 has empty working set, it obviously does not have any Lagrange multipliers associated with it.

With iteration 4 we move along a constraint that when looking at the plot seems to be part of the optimal working set so we expect a positive Lagrange multiplier. The multiplier is indeed positive and since at iteration 5 the step is 0 and the single λ is still the same 0.8 (i.e. positive) the algorithm stops.

Explain the active set method for convex QPs listed on p. 472 in N&W.

We start with an initial feasible point and an active set corresponding to that point. (Linear programming problem)

In each iteration we solve an equality constrained QP sub-problem using the current working set to find a step p_k that would move us from the initial point for this iteration x_k to the minimizer $x_k + p_k$ corresponding to the working set W_k.

If p_k is nonzero and making the step retains feasibility in regards to the original (inequality constrained) problem, we set $x_{k+1} = x_k + p_k$ and continue to next iteration.

If p_k is nonzero but making the step does not retain feasibility, then there exists at least one constraint currently not in the working set that essentially blocks us from making the full step. In that case we make the

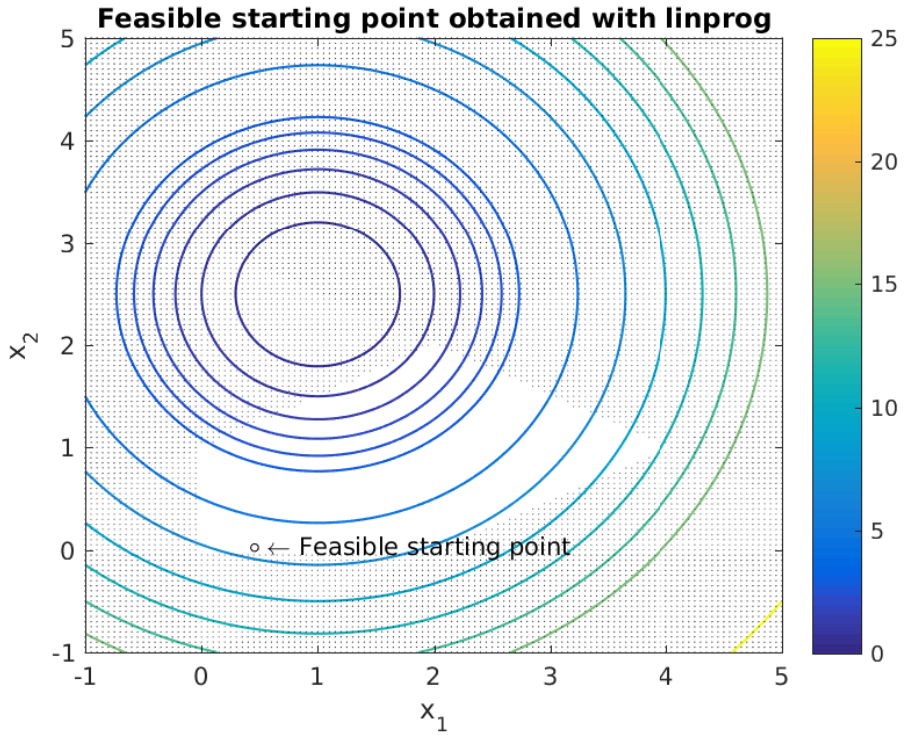

Figure 6: Feasible starting point

largest possible step in direction of p_k while maintaining feasibility, add one of the blocking constraints to the working set, and continue to next iteration.

If p_k is zero we compute the Lagrange multipliers $\hat{\lambda}$ corresponding to the current working set. If all these multipliers are non-negative then $x^* = x_k$ is a solution to the original problem. Using $\hat{\lambda}$ and setting all multipliers corresponding to constraints not in the working set to 0 yields the multipliers for the solution λ^*. Feasibility is guaranteed since the algorithm starts with a feasible point and only allows for feasible steps.

If one of the obtained Lagrange multipliers $\hat{\lambda}$ is negative then we can move away from the corresponding constraint while reducing the objective function, this means we will remove that constraint from the working set and continue to next iteration.

Use linprog to compute a feasible point to a QP. Apply and test this procedure to the problem in Example 16.4.

We used linprog with empty coefficient vector f (equivalent to f(x) = 0) and the inequality constraints specified previously. Figure 6 displays the feasible starting point (0.4568, 0.0227) we obtained.

Although the point might look close to a constraint on the figure, the lambdas obtained from linprog are all effectively zero and thus the initial working set is empty.

Implement the algorithm in p. 472 and test it it for the problem in Example 16.4. Print information for every iteration of the algorithm (i.e. the point x_k , the working set W_k , etc) and list that in your report.

Running the implemented algorithm with the same initial conditions as in table 1 yields the table 2. The obvious difference is that our implemented algorithm runs for 6 iterations compared to the previous conceptual algorithm that ran for 5 iterations. The difference comes from the fact that the algorithm specified in N&W (Algorithm 16.3) and therefore also our implementation does not remove constraints with negative Lagrange multipliers from the working set when the computed step p_k was nonzero. Statements in our implementation could be restructured if smaller number of iterations is desired. Note that $alpha_k$ and p'_k are separated into different columns in table 2 but otherwise the tables are identical and thus our implementation correct for the particular starting conditions.

Table 2: Active set test with initial point (2, 0)

iteration	W	x'_k	p'_k	$\hat{\lambda}'$	α_k
1	3,5	[2, 0]	[0, 0]	[-2, -1]	-
2	5	[2, 0]	[-1, 0]	[-5]	1
3	5	[1, 0]	[0, 0]	[-5]	-
4	$\emptyset$	[1, 0]	[0, 2.5]	[]	0.6
5	1	[1, 1.5]	[0.4, 0.2]	[0.8]	1
6	1	[1.4, 1.7]	[0, 0]	[0.8]	-

Test your active set algorithm for the problem in Example 16.4 using (16.47) and (16.48) in N&W. Do the same using quadprog.

We tested our implementation as well as quadprog on inputs corresponding to the big M method with various feasible and infeasible starting points and verified that the our implementation and quadprog produce the same solutions.

1.4 Problem 4 - Markowitz Portfolio Optimization

For a given return, R, formulate Markowitz' Portfolio optimization problem as a quadratic program.

The classical Markowitz objective function is given by

$$w^T \Sigma w - q * R^T w \tag{9}$$

where w is the vector of weights corresponding to proportion of the portfolio invested in each asset, Σ is the covariance matrix of the assets, q is a scaling parameter called risk-tolerance factor and R is a vector of expected returns for the securities. Risk-tolerance of 0 corresponds to just minimizing the risk, i.e. the second part of the objective function can be left out. Changing the notation, this can be written as

$$\frac{1}{2} x' H x \tag{10}$$

There are two inequality constraints, one to specify the minimum required return and one to ensure that each of the portfolio weights is non-negative. If short-selling is allowed the later of these constraints can be removed. There is also a single equality constraint that ensures the weights of assets in the portfolio sum up to one, this makes the assumption that all of the available money is invested in this market. The quadratic program is thus

$$\begin{aligned} \min_x f(x) &= \frac{1}{2} x' H x \\ s.t. \quad Ax &\leq b \\ -x_i &\leq 0 \\ A_{eq} x &= 1 \end{aligned} \tag{11}$$

$$H = 2 \times \begin{bmatrix} 2.30 & 0.93 & 0.62 & 0.74 & -0.23 \\ 0.93 & 1.40 & 0.22 & 0.56 & 0.26 \\ 0.62 & 0.22 & 1.80 & 0.78 & -0.27 \\ 0.74 & 0.56 & 0.78 & 3.40 & -0.56 \\ -0.23 & 0.26 & -0.27 & -0.56 & 2.60 \end{bmatrix}$$

$$A = \begin{bmatrix} -15.10 & -12.50 & -14.70 & -9.02 & -17.68 \end{bmatrix}$$

$$A_{eq} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$b = -10$$

One could also include the inequality constraints on weights x_i in the A matrix by means of a diagonal matrix where diagonal elements are all -1. In that case a vector of zeros would need to be appended to b as well. Additionally, the equality constraint could also be included in the same A matrix by adding rows of negative ones and positive ones to the matrix and corresponding -1 and 1 to the b vector.

What is the minimal and maximal possible return in this financial market?

Maximal expected return is attained if the entire portfolio is just the single security with the highest return, which in this case is 17.68%. Similarly, assuming short-sales are not allowed, the minimal expected return is 9.02%. Assuming shorting with no additional costs is allowed, the minimal possible expected return is -17.68%.

Use quadprog to find a portfolio with return, R = 10.0, and minimal risk. What is the optimal portfolio and what is the risk (variance)?

Setting the minimum required return to 10 yields the following portfolio weights and variance.

$$x' = \begin{bmatrix} 0.0883 & 0.2509 & 0.2824 & 0.1038 & 0.2748 \end{bmatrix}$$

$$\sigma^2 = 0.6249$$

An important note is that the return of this portfolio is actually R = 14.4129. This means that there is no efficient portfolio with return 10. This portfolio is the minimum variance portfolio. One could also change the return constraint to an equality instead of inequality but in that case the portfolios yielded by the optimization routine would only be efficient if the return was set to above 14.4129. If exactly 10 percent expected return is desired, one should invest a portion of the money in some efficient portfolio (specifically, the portfolio which maximizes Sharpe ratio i.e. the tangency portfolio) and the rest into a zero return zero risk asset, that is withhold from investing it at all.

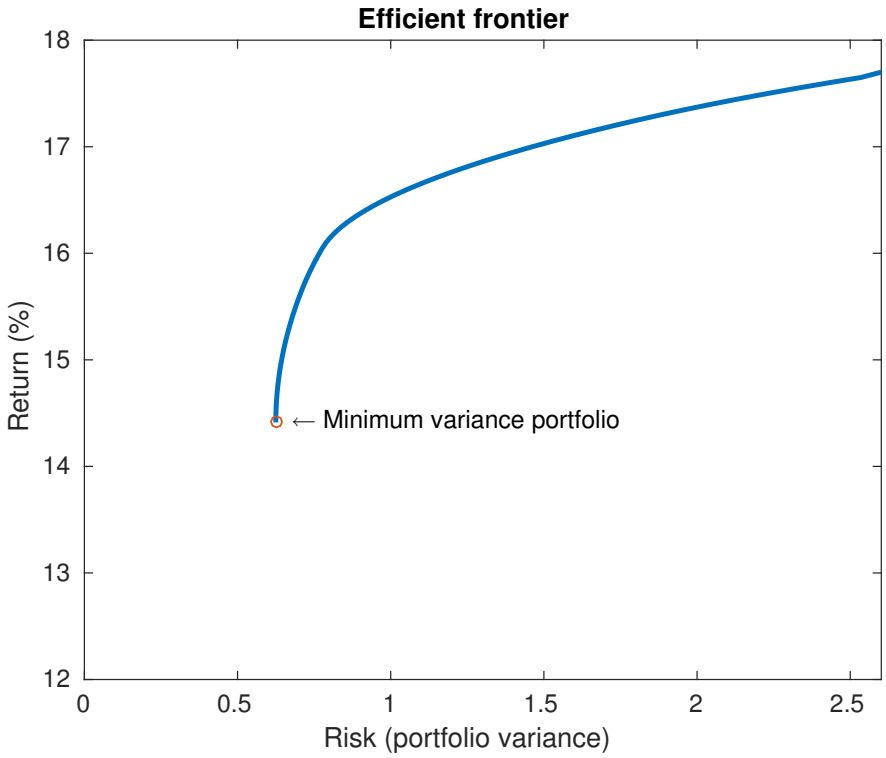

Figure 7: Efficient frontier

Compute the efficient frontier, i.e. the risk as function of the return. Plot the efficient frontier as well as the optimal portfolio as function of return.

Figure 7 shows the efficient frontier as well as the minimum variance portfolio obtained in the previous step.

As can be seen from the plot initially the return increases greatly as we take on slightly more risk than the minimum variance portfolio but above around 16.5% return the effect slows down as the efficient portfolio approaches the security with the largest return and the risk reduction from diversification is lost.

In the following we add a risk free security to the financial market. It has return $r_f = 2.0$. What is the new covariance matrix and return vector?

Since there is no uncertainty in the return of the risk free asset and it is not related to the performance of the rest of the market, a row and column of zeros should be added to the covariance matrix.

$$H = 2 \times \begin{bmatrix} 2.30 & 0.93 & 0.62 & 0.74 & -0.23 & 0 \\ 0.93 & 1.40 & 0.22 & 0.56 & 0.26 & 0 \\ 0.62 & 0.22 & 1.80 & 0.78 & -0.27 & 0 \\ 0.74 & 0.56 & 0.78 & 3.40 & -0.56 & 0 \\ -0.23 & 0.26 & -0.27 & -0.56 & 2.60 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$returns' = \begin{bmatrix} 15.10 & 12.50 & 14.70 & 9.02 & 17.68 & 2.0 \end{bmatrix}$$

Compute the efficient frontier, plot it as well as the (return,risk) coordinates of all the securities. Comment on the effect of a risk free security. Plot the optimal portfolio as function of return.

Figure 1.5 shows the two efficient frontiers as well as each of the 5 risky securities. With the risk free security one can have an efficient portfolio with arbitrary level of risk (up to 2.60 - the risk for largest return asset). Risk zero with the entire portfolio invested in the risk free asset and as a larger return is desired the proportion of portfolio in risk free security gradually decreases and the proportion in risky assets rises. Above around 15% return the two efficient frontiers are essentially the same since to achieve these returns virtually nothing can be invested in the risk free security.

What is the minimal risk and optimal portfolio giving a return of R = 15.00. Plot this point in your optimal portfolio as function of return as well as on the efficient frontier diagram

The weights and variance for optimal portfolio with R=15.00 are the following.

$$x'_{P_{15}} = \begin{bmatrix} 0.1655 & 0.1365 & 0.3115 & 0.0266 & 0.3352 & 0.0247 \end{bmatrix}$$

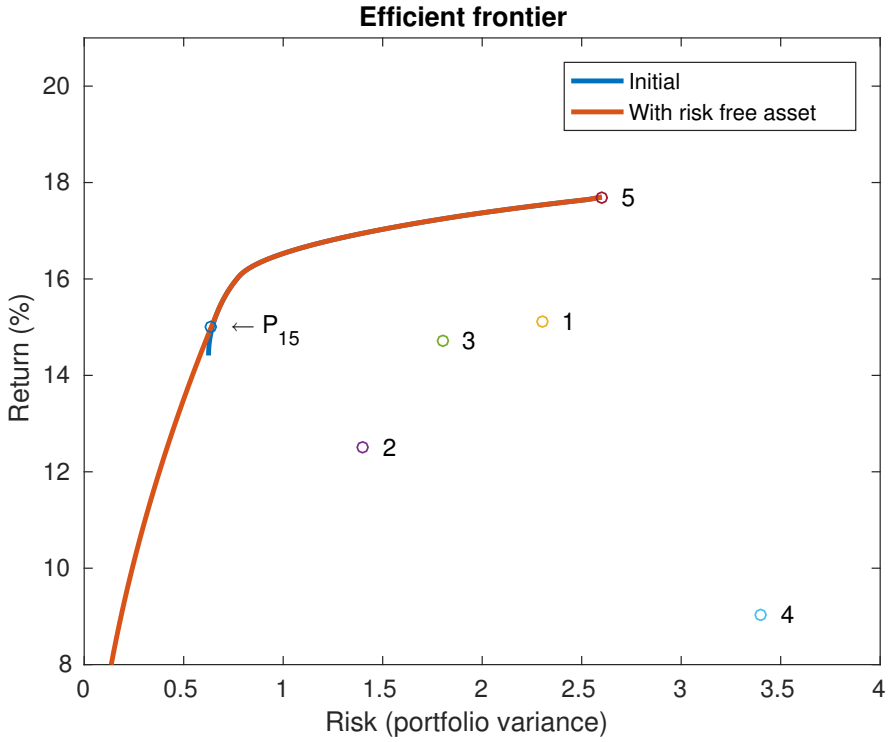

Figure 8: Efficient frontier with risk free asset

$$\sigma^2_{P_{15}} = 0.6249$$

This portfolio can be seen on figure 1.5 (P_{15})

1.5 Problem 5 - Interior-Point Algorithm for Convex Quadratic Programming

One of the procedures for solving a convex quadratic program is an interior-point algorithm. To start with, we use the following standard formulation of the problem:

$$\begin{aligned} \min_x f(x) = \frac{1}{2}x'Hx + g'x \\ s.t. \quad A'x = b \\ C'x \geq d. \end{aligned} \tag{12}$$

From this system, we can derive a Lagrangian by introducing slack variables for inequalities and further its gradient that form following KKT optimality conditions:

$$\begin{aligned} Hx + g - \sum_i a_i\lambda_i = 0, \quad i \in E \cup I \\ a_i'x + b_i = 0, \quad i \in E \\ c_i'x + d_i - s_i = 0, \quad i \in I \\ s_i\lambda_i = 0, \quad i \in I \\ \lambda_i \geq 0, \quad i \in I \\ s_i \geq 0, \quad i \in I \end{aligned} \tag{13}$$

This algorithm requires multiple iterations to find the solution and starts the procedure from a feasible solution that in fact is already non trivial. That is due to the fact that non-negativity constraints are causing the most problem so by starting within feasible area we can just make sure never to leave it and violate them. Once we have the starting point, we use Newton method to find the best direction along which we should move in order to improve the objective function. However, to make a step we also need to know how far we can move before we violate one of the constraints therefore we perform a line search along the found direction. Then we repeat this simple procedure until we reached optimality. In essence this is the key idea behind the interior-point algorithm, but if we follow described procedure we will realize that its not very effective and suffers from serious limitations. The problem is that the algorithm gets very close to the boundary very quick and therefore the steps made from there bring very little improvement before violating the constraints.

Because of that, we implement its modified version which for each iteration performs two steps that are called centering and affine. Before making the actual step toward the solution we try to get to the center in between the constrains what is supposed to ensure that the step we make is not excessively small as in case of naive version. That is followed by affine step, that simply moves along the optimal direction from a given point. Although this may sound like quite a bit more of work, it can be implemented efficiently, so that one iteration is not much more computationally expensive as the iteration of a naive procedure. This is because the most demanding part of an iteration is factorization of KKT matrix which has to be done for naive as well as modified version. In case of the latter, the factorization can be shared between the two steps, therefore even though it requires more calculations, the most demanding part is the same thus the total performance of a single iteration is not much worse, whereas the convergence rate is much better.

What is H, g, A, C, b, and d for the Markowitz Portfolio Optimization Problem with R = 15 and the presence of a risk-free security?

H is the covariance matrix with risk free security added as defined in Problem 4 section 5 and g is a 6-element columnvector of zeros since we are only minimizing risk. A is a 6-element columnvector of ones, b is 1. This ensures that portfolio weights sum to 1. C is a 6x7 matrix with the first column being the returns and the rest of the columns form a 6x6 identity matrix, d is 7-element column vector with first element 15 and rest zeros. This ensures that expected return is at least 15 and all weights are non-negative, i.e. no short selling allowed.

The variables defined in the end of MatLab file problem4.m.

Next, we tried our own interior-point algorithm on this problem. As an initial starting point we used

$$x' = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

, that we know to be a feasible point (weights sum to 1, all weights are non-negative and the return for asset 5 is 17.68 ¿= 15). For $R = 15$ we obtained following portfolio:

$$x' = \begin{bmatrix} 0.1546 & 0.1273 & 0.2516 & 0.0354 & 0.4201 & 0.0110 \end{bmatrix}$$

that is slightly different than the solution we got with quadprog built-in function since our implementation had some issues converging. The algorithm seemed to hover around the true minimum. Due to that the efficient frontier would look slightly different than on figure .