# Project 1: direct methods in optimization with constraints

Lorenzo Vigo

Numeric Linear Algebra: $12^{th}$ November 2021

**T1: Show that the predictor steps reduces to solve a linear system with matrix $M_{KKT}$.**

As stated in the project document, predicting the minimum of $f(x)$ within the given constraints is equivalent to solving $F(z) = 0$ where $F$ is a function such that $F : \mathbb{R}^N \longrightarrow \mathbb{R}^N$, with $N = n + p + 2m$, defined by:

$$F(z) = F((x, \gamma, \lambda, s) = (Gx + g - A\gamma - C\lambda, b - A^T x, s + d - C^T x, s \odot \lambda)$$

We should note that $\odot$ notes the element-wise vector multiplication. In our case, $s$ and $\lambda$ will contain (respectively) all the $s_i$ and $k_i$ elements adequately indexed.

In order to find the solutions of $F(z) = 0$ we are suggested to apply the Newton method. This method is iterative and finds a solution following this propagation formula:

$$x_{(k+1)} = x_{(k)} - J(x_k)^{-1} F(x_k) \tag{1}$$

In this formula, $J(x_k)$ represents the Jacobian Matrix of F, which is:

$$J = \begin{pmatrix} \frac{\partial F_1}{\partial x} & \cdots & \frac{\partial F_1}{\partial s} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_4}{\partial x} & \cdots & \frac{\partial F_4}{\partial s} \end{pmatrix}$$

In fact, if we compute the value of $J$:

$$J = \begin{pmatrix} G & -A & -C & 0 \\ -A^T & 0 & 0 & 0 \\ -C^T & 0 & 0 & I \\ 0 & 0 & sI & I\lambda \end{pmatrix} = M_{KKT}$$

As we want to prove a system with matrix $M_{KKT}$ instead of its inverse, we will rearrange the terms in (1):

$$x_{(k+1)} - x_{(k)} = -M_{KKT}^{-1} F(x_k)$$
$$M_{KKT}(x_{(k+1)} - x_{(k)}) = -F(x_k)$$

We can define $\delta_z^{(k)} = x_{(k+1)} - x_{(k)}$, as it happens in our implementation. In that case:

$$M_{KKT}\delta_z^{(k)} = -F(x_k)$$

This expression is the linear system with $M_{KKT}$ that we need to solve. We are suggested to implemented a corrected Newton method, in which, after obtaining the value of $\delta_z^{(k)}$ as explained, the next iteration is computed as:

$$x_{(k+1)} = x_{(k)} + 0.95\alpha_k \delta_z^{(k)}$$

**T2. Explain the previous derivations of the different strategies and justify under which assumptions they can be applied.**

As in this case we are supposing that $p = 0$, $A$ is dimension-less. Therefore:

$$M_{KKT} = \begin{pmatrix} G & -C & 0 \\ -C^T & 0 & I \\ 0 & S & \Lambda \end{pmatrix}$$

In these strategies, the right hand vector is expressed as $-F(z) = (-r_1, -r_2, -r_3)$. We will say then that the linear system we are solving is:

$$\begin{pmatrix} G & -C & 0 \\ -C^T & 0 & I \\ 0 & S & \Lambda \end{pmatrix} \begin{pmatrix} \delta_x \\ \delta_\lambda \\ \delta_s \end{pmatrix} = \begin{pmatrix} -r_1 \\ -r_2 \\ -r_3 \end{pmatrix}$$

As an equation system, this may be expressed as:

$$\begin{cases} G\delta_x - C\delta_\lambda = -r_1 \\ \delta_s - C^T\delta_x = -r_2 \\ \Lambda\delta_s + S\delta_\lambda = -r_3 \end{cases}$$

**Strategy 1** tells us to isolate $\delta_s$ from third row, obtaining that $\delta_s = \Lambda^{-1}(-r_3 - S\delta_\lambda)$. In that case, substituting in the second row, our system is transformed to:

$$\begin{cases} G\delta_x - C\delta_\lambda = -r_1 \\ -\Lambda^{-1}r_3 - C^T\delta_x - \Lambda^{-1}S\delta_\lambda = -r_2 \end{cases}$$

Going back to the matrix representation, we will see that:

$$\begin{pmatrix} G & -C \\ -C^T & -\Lambda^{-1}S \end{pmatrix} \begin{pmatrix} \delta_x \\ \delta_\lambda \end{pmatrix} = \begin{pmatrix} -r_1 \\ -r_2 + \Lambda^{-1} r_3 \end{pmatrix}$$

This is equivalent to the matrix expression given in the project statement. We need this matrix to be positive definite and symmetric in order to apply $LDL^T$ factorization and solve the system. As $G$ is symmetric and $-\Lambda^{-1}S$ is a diagonal block, it is clear that our new reduced $M_{KKT}$ is symmetric. To show that it is positive definite, we should apply Schur Complements. [1]

Meanwhile, **Strategy 2** invites us to isolate $\delta_s$ in the second row instead, and then substitute into the third row. In that way:

$$\begin{cases} G\delta_x - \quad C\delta_\lambda = -r_1 \\ -r_2\Lambda - \Lambda C^T\delta_x + S\delta_\lambda - = -r_3 \end{cases}$$

This is equivalent to the next step stated in the project's statement. After that, we need to perform substitution once again. We will finally get that:

$$(G + CS^{-1}\Lambda C^T)\delta_x = -r_1 + CS^{-1}(-r_3 + \Lambda r_2)$$

In order to perform Cholesky, we need to see that $(G + CS^{-1}\Lambda C^T)$ is symmetric. In fact:

$$(G + CS^{-1}\Lambda C^T)^T = G^T + (C^T)^T\Lambda^T S^{-T}C^T = G + C\Lambda S^{-1}C = G + CS^{-1}\Lambda C^T$$

This is true because $G$ is symmetric and $\Lambda$ and $S$ are diagonal. In order to see that it is positive definite we need to check that for any suitable $x$, $x^T Gx + x^T CS^{-1}\lambda C^T x > 0$.

As $G$ is positive definite, we only need to check that $x^T CS^{-1}\lambda C^T x > 0$. In our example, $S^{-1}\lambda$ is a diagonal matrix with only positive numbers. Also, due to the structure of $C$, if we define the concatenation of $x$ and $-x$ as $x' = (x, -x)$, the previous expression can be written as it follows:

$$x'^T S^{-1}\lambda x'$$

This expression is positive as it results in:

$$\sum_{i=0}^{n} k_i (x'_i)^2$$

It should be noted that, by definition, $k_i$ is always positive. Therefore, the matrix is also positive definite and Cholesky factorization is applicable.

**T3. Isolate $\delta_s$ from the 4th row of $M_{KKT}$ and substitute into the 3rd row. Justify that this procedure leads to a linear system with a symmetric matrix.**

Let's consider the equations of the general case:

$$\begin{cases} G\delta_x - A\delta_\gamma - C\delta_\lambda = -r_L \\ -A^T\delta_x = -r_A \\ \delta_s - C^T\delta_x = -r_C \\ \Lambda\delta_s + S\delta_\lambda = -r_S \end{cases}$$

By taking the steps suggested by the exercise, we get that $\delta_s = -\Lambda^{-1}(r_S + S\delta_\lambda)$. Substituting into the 3rd row, we get that:

$$C^T\delta_x - \Lambda^{-1}(r_S + S\delta_\lambda) = r_C$$
$$C^T\delta_x - \Lambda^{-1}S\delta_\lambda = r_C + \Lambda^{-1}r_S$$

Passing it into matrix form, we get that:

$$\begin{pmatrix} G & -A & -C \\ -A^T & 0 & 0 \\ -C^T & 0 & -S\Lambda^{-1} \end{pmatrix} \begin{pmatrix} \delta_x \\ \delta_\lambda \\ \delta_s \end{pmatrix} = \begin{pmatrix} -r_L \\ -r_A \\ -r_C + \Lambda^{-1}r_S \end{pmatrix}$$

As we are applying $LDL^T$ factorization, we need to check once again whether this matrix is symmetric and positive definite or not. It is clear that it is symmetric, knowing that $S$ and $\Lambda$ are diagonal. We could apply Schur Complements in order to prove that it is positive definite, as we stated previously.

**The memory of the project should also contain information about the number of iterations performed, the precision of the results, the condition number of the matrices, etc, and also about any difficulties in the implementation you have found, improvements, other checks, etc.**

The most evident difficulty we have encountered while implementing the project is the confusion caused by the different values of the right-hand vector along all the strategies. We have found it hard to identify how it should be constructed for each one of the algorithms.

Also, we were able to generalize the main workflow of all strategies into one only method (called KKT_solver_step) by passing the matrix and vector construction and the system solving routines as parameters. Nevertheless, we came across computation issues during the step "4. Corrector substep" in exercise C2. Luckily, we could identify that multiplying by $-1$ some of the terms was enough to amend the issue, which led to the existence of the parameter called sign. It would have been desirable to spot the cause of this incident, instead of adding this quick fix to the code.

To conclude, we will add that $LDL^T$ strategy should have been implemented using Numpy's Linear Algebra package's solve_triangular method to solve the system after obtaining the $LDL^T$ decomposition. This approach would raise issues in exercise C6 as it seems that one of the involved matrices is diagonal by blocks rather than purely diagonal, which is not accepted by the solve_triangular method. In order to be able to solve the system, we chose to use the general solve method instead, which is much slower, leading exercise C6 to take way too long. A desirable improvement in future work is to explore alternatives to solve the system quicker.

Finally, we will show graphs containing information about the number of iterations, the precision of the results and the condition number of the matrices in each exercise in terms of a given dimension. It should be noted that the piece of code that generated these graphs is commented and kept apart from the main execution of the script.

**Exercise C2**

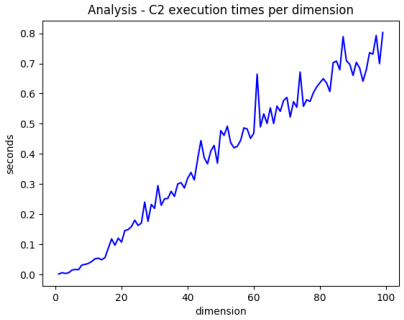


Figure 1: Execution times of exercise C2 depending on the dimension

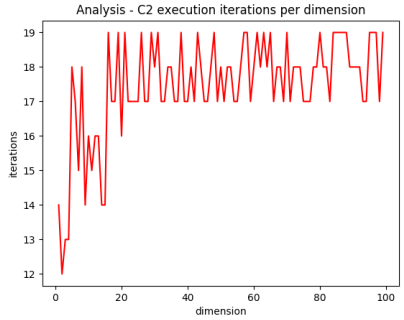


Figure 2: Iterations performed in exercise C2 depending on the dimension

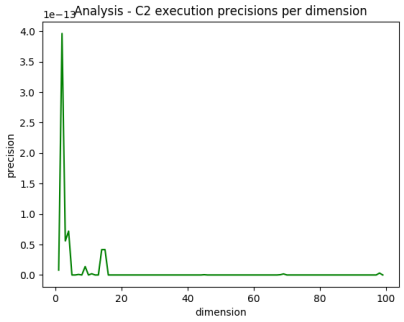


Figure 3: Precision obtained in exercise C2 depending on the dimension


Figure 4: Condition number in exercise C2 depending on the dimension

For exercise C2, execution times increase along with dimension, but always stay under 1 second. The number of iterations is 15 on average for a low dimensionality (under 20), while it varies from 17 to 19 for higher dimensions. We can guarantee precision up to the $12^{th}$ decimal number and condition number's value is always around 24 (computed with 2-norm).

**Exercise C4, Strategy 1**

For strategy 1 in exercise C4, execution times also increase along with dimension, but now reaching up to 3.5 seconds. This effect may be caused by the fact that we are using the solve method, as mentioned before. The behavior of the number of iterations remains more or less similar to the one in exercise C2. We can also guarantee precision up to the $12^{th}$ decimal number in this exercise. This time, condition number is (almost) always really low. For this reason, we may say that in this strategy the KKT matrix is well conditioned.

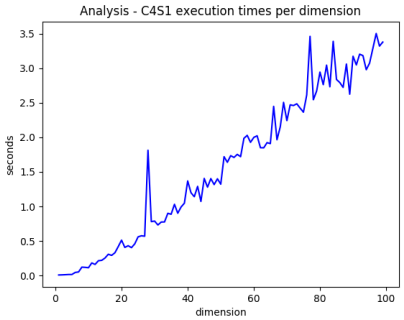


Figure 5: Execution times of exercise C4, strat 1 depending on the dimension

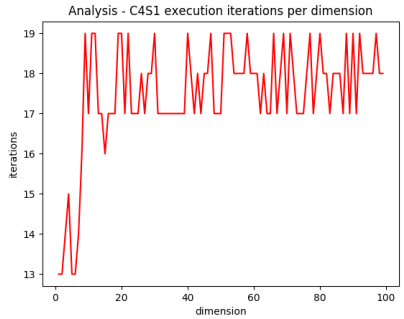


Figure 6: Iterations performed in exercise C4, strat 1 depending on the dimension

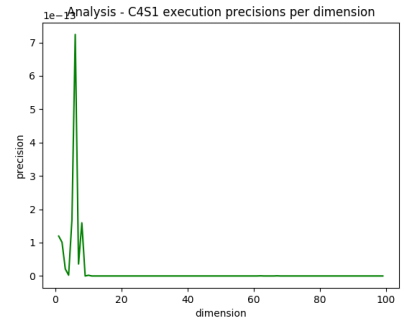


Figure 7: Precision obtained in exercise C4, strat 1 depending on the dimension

**Exercise C4, Strategy 2**

For strategy 2 in exercise C4, execution times also increase along with dimension, but this time on a slower rate. Also, the execution time generally

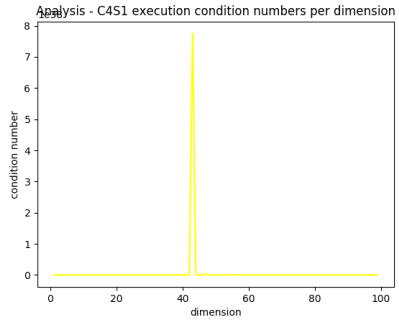


Figure 8: Condition number in exercise C4, strat 1 depending on the dimension

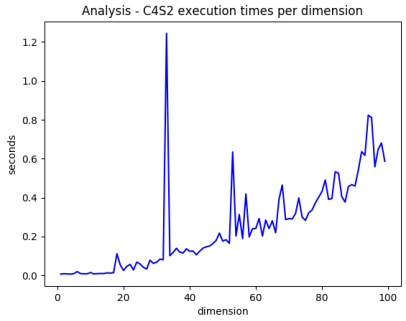


Figure 9: Execution times of exercise C4, strat 2 depending on the dimension

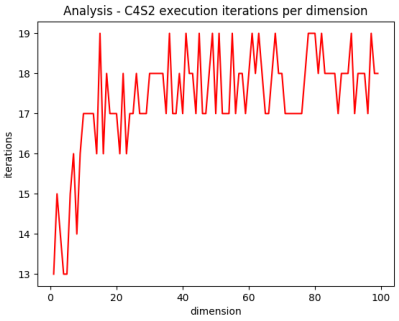


Figure 10: Iterations performed in exercise C4, strat 2 depending on the dimension

stays under 1 second. The behavior of the number of iterations remains more or less similar to the found in the past two exercises. We can once again guarantee precision up to the $12^{th}$ decimal number. This time, condition number is always

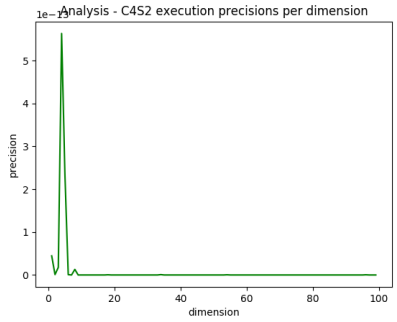


Figure 11: Precision obtained in exercise C4, strat 2 depending on the dimension

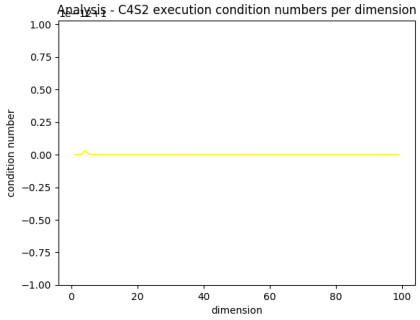


Figure 12: Condition number in exercise C4, strat 2 depending on the dimension

really low and therefore, we can claim that the KKT matrix is well conditioned.

# References

[1] Jean Gallier, *The Schur Complement and Symmetric Positive Semidefinite (and Definite) Matrices*. University of Pennsylvania, 2019.