Report - Optimization Theory and Methods – Warsaw University of Technology

Bartłomiej Guś, PW, MM, WD, MN

MATLAB Optimization Toolbox and CasADi optimization

## 1. Task

Familiarise with the Optimization Toolbox in MATLAB and CasADi optimization software. Analysis and selection of best tools to solve following optimization problem:

- optimization of a linear constrained objective function,

## 2. CasAdi and Matlab Optimization Toolbox

CasADi is a toolbox for advanced optimization and algorithmic differentiation in MATLAB. It provides a high-level interface for solving nonlinear optimization problems, including both continuous and discrete variables, and supports a wide range of problem types and solvers. The main features of CasADi include:

- Support for multiple solvers: CasADi interfaces with several popular open-source optimization solvers, allowing users to easily switch between solvers depending on the problem type and desired solution quality.
- Flexibility: CasADi allows users to define and solve optimization problems in a flexible and efficient way, with support for nonlinear constraints and mixed-integer programming.
- Interoperability: CasADi can be easily integrated with other modelling tools, such as Modelica and ACADO, to enable the use of advanced optimization techniques in a wide range of applications.

Overall, CasADi is a powerful tool for solving complex optimization problems, and it is widely used in a variety of fields, such as control systems, robotics, and machine learning.

The Optimization Toolbox for MATLAB is a set of functions for solving optimization problems, including linear and nonlinear programming. The toolbox provides a wide range of algorithms and solvers to find the optimal solution for a given problem. Some of the key features of the Optimization Toolbox include:

- Linear programming: The toolbox provides solvers for linear programming problems, including the simplex method.
- Nonlinear programming: The toolbox provides solvers for nonlinear programming problems.
- Constrained optimization: The toolbox provides solvers for optimization problems with constraints, including linear and nonlinear equality and inequality constraints.
- Multi-objective optimization: The toolbox provides solvers for multi-objective optimization problems, which are optimization problems with multiple objectives that need to be optimized simultaneously.

The Optimization Toolbox for MATLAB is designed to be easy to use, and it provides a simple and consistent interface for solving a wide range of optimization problems. It is widely used in a variety of fields, such as engineering, finance, and economics, and it is particularly useful for solving problems that involve complex, nonlinear relationships.

Both methods tested in this project are based on the 'Ipopt' solver, an open-source primal-dual interior point method for large-scale nonlinear optimization. It can be used to solve general nonlinear programming problems.

### 3. Selected functions

To test implemented algorithm, the following objective functions has been selected:

- Function A:

$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 ;$$

- Function B:

$$f(x,y) = x^2 + x * y + 2 * x + y^2$$

- Function C:

$$f(x,y) = x^4 - 3 * x^2 * y^2 + 23 * x^3 + 3 * y^2 - 2$$

### 4. Implementation

Implemented algorithm as Matlab .m function in the attached .zip file The picture below presents the block diagram of the implemented solution.

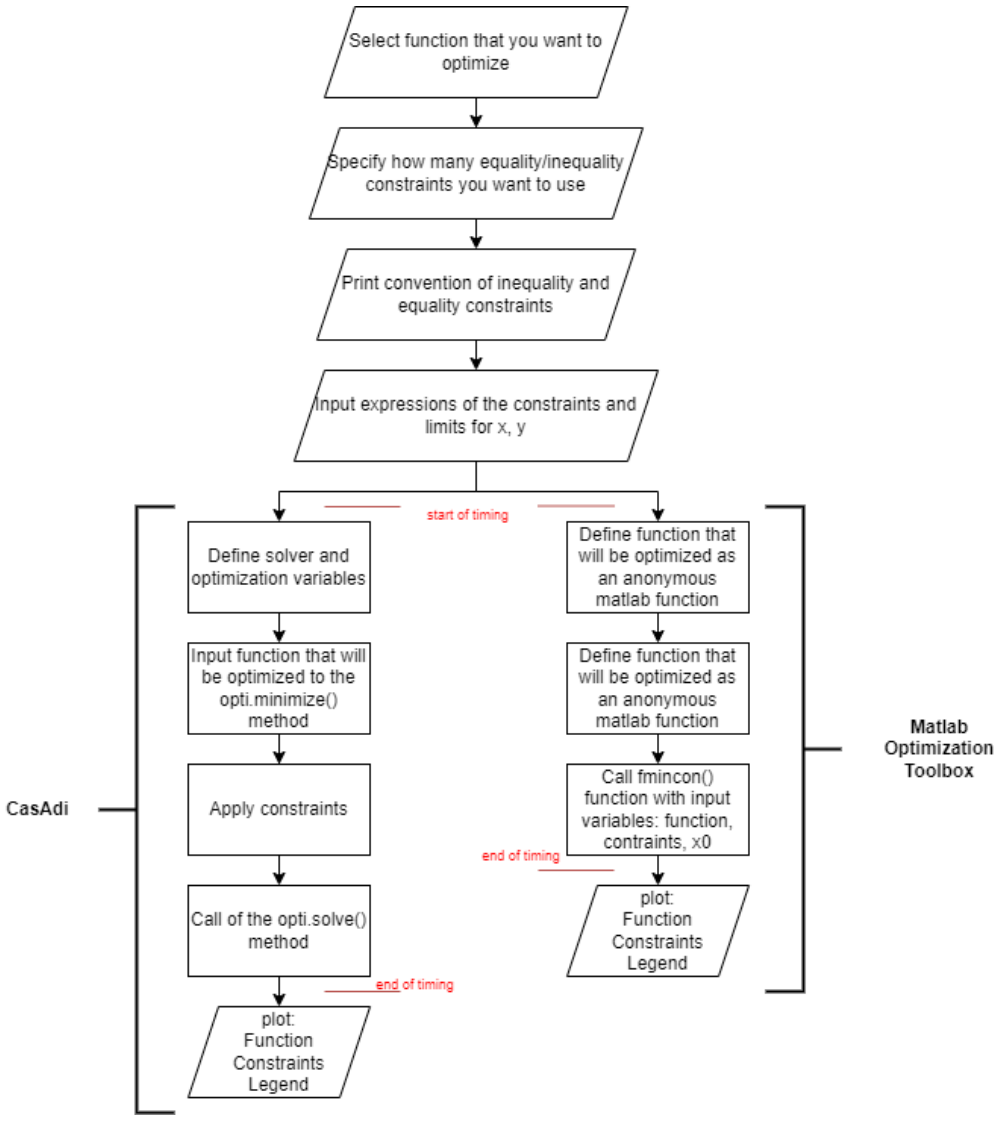


*Figure 1 Overall diagram of the implemented solution*

The built solution gives the user the option to choose the function he wants to optimize from 3 examples. The user has the option of introducing linear and non-linear constraints after defining the number of them in advance. In addition, the user enters a range of values on the axes (x, y) of the graphs - this allows drawing graphs of those

parts of the function that the user wants to analyse. The solution uses a proprietary implementation that draws contour plots of the optimized functions, along with the linear and non-linear constraints, as well as the result of the algorithm.

Optimization with CasAdi required the use of the following functions:

- *casadi.Opti()* - Object initialization
- *opti.variable()* - Definition of a scalar variable
- *Opti.minimize()* - Definition of the optimized function (function expression as a parameter)
- *Opti.subject_to()* – Definition of constraints (constraints expression as a parameters)
- *Opti.solver()* - selection of solver (solver name as a parameter)
- *Opti.solve()* - run solver

The metadata of the optimization performed with CasAdi is displayed in the console as shown in the illustration below.

```
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr
   0 1.7000000e+002 6.00e+000 6.43e-001  -1.0 0.00e+000    -  0.00e+000 0.00e·
   1 1.4032696e+002 3.17e+000 9.83e+001  -1.0 2.10e+000   2.0 1.00e+000 4.71e·
   2 1.1932390e+002 2.66e-015 1.65e+002  -1.0 3.75e+000   1.5 6.25e-002 1.00e·
   3 4.0483725e+001 1.78e-015 5.95e+001  -1.0 8.79e-001    -  1.00e+000 6.89e·
   4 4.0004808e+001 1.78e-015 9.03e+001  -1.0 3.63e-001    -  1.00e+000 1.67e·
   5 4.0286307e+001 8.88e-016 4.81e-004  -1.0 3.57e-003    -  1.00e+000 1.00e·
   6 4.0004676e+001 0.00e+000 4.81e-004  -2.5 3.57e-003    -  1.00e+000 1.00e·
   7 4.0000149e+001 0.00e+000 1.25e-007  -3.8 5.75e-005    -  1.00e+000 1.00e·
   8 3.9999999e+001 0.00e+000 1.36e-010  -5.7 1.89e-006    -  1.00e+000 1.00e·
   9 3.9999998e+001 0.00e+000 6.04e-014  -8.6 2.34e-008    -  1.00e+000 1.00e·

Number of Iterations....: 9

                                   (scaled)                 (unscaled)
Objective...............:  3.9999997642506031e+001   3.9999997642506031e+001
Dual infeasibility......:  6.0396132539608516e-014   6.0396132539608516e-014
Constraint violation....:  0.0000000000000000e+000   0.0000000000000000e+000
Complementarity.........:  2.5059934962951819e-009   2.5059934962951819e-009
Overall NLP error.......:  1.0618617096724847e-009   2.5059934962951819e-009


Number of objective function evaluations             = 10
Number of objective gradient evaluations             = 10
Number of equality constraint evaluations            = 10
Number of inequality constraint evaluations          = 10
Number of equality constraint Jacobian evaluations   = 10
Number of inequality constraint Jacobian evaluations = 10
Number of Lagrangian Hessian evaluations             = 9
Total CPU secs in IPOPT (w/o function evaluations)   =      0.028
Total CPU secs in NLP function evaluations           =      0.000

EXIT: Optimal Solution Found.
      solver  :   t_proc      (avg)   t_wall      (avg)    n_eval
       nlp_f  |        0 (       0)        0 (       0)        10
       nlp_g  |        0 (       0)        0 (       0)        10
  nlp_grad_f  |        0 (       0)        0 (       0)        11
  nlp_hess_l  |        0 (       0)        0 (       0)         9
   nlp_jac_g  |        0 (       0)        0 (       0)        11
       total  |  29.00ms ( 29.00ms)  28.92ms ( 28.92ms)         1
```

*Figure 2 Example metadata returned by CasAdi during optimization process*

Optimization with Matlab Optimization Toolbox required the use of the following functions:

- Anonymous function as a definition of optimized function
- Fmincon() - nonlinear programming solver with the following parameters:
    - *Fun* – function that will be optimized
    - *x0* – Initial point
    - *Inequality_constraints* (matrix of A)
    - *Value_inequality_contraints* (vector of B)
    - *Equality_constraints* (matrix of A)
    - *Value_equality_constraints* (vector of B)

  As the output of *fmincon()* function we get:

- *x* – values of independent variables for minimum
- *fval* – value of function in the found minimum point
- *exitflag* – variable containing information about the termination status of the function. E.g. 1 - minimum found; -2 - minimum not found
- *Output* - structure with metadata of the optimization shown on the picture below
- *Lambda* - Lagrange multipliers at the solution
- *Grad* – gradient at the solution
- *Hessian* – Approximate Hessian

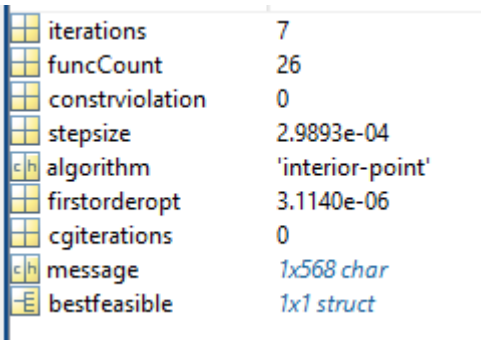


*Figure 3 Example metadata of optimization contained in the output structure*

## 5. Implementation test

### a. Function A

The following domain constraints are assumed for function A:

Constraints:

- $2 * x + 3 * y = 6$
- $x + y \leq 1$

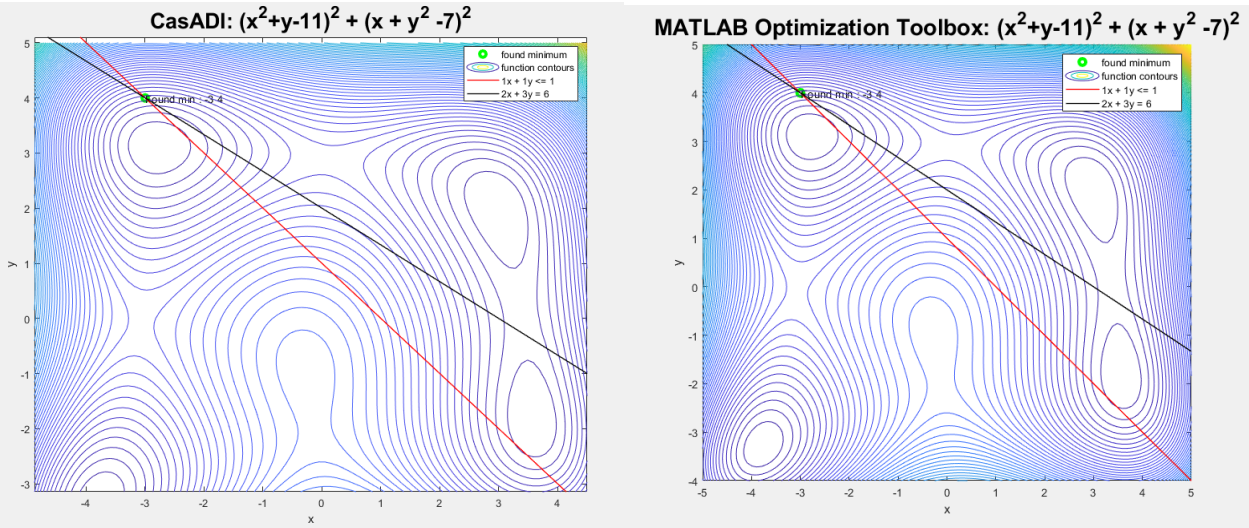


*Figure4 4 From left: CasAdi and MATLAB Optimization Toolbox optimization results for function A*

*Table 1 From left: CasAdi and MATLAB Optimization Toolbox optimization results for function B*

| | Number of iterations | Overall evaluation time [s] |
|---|---|---|
| **CasAdi** | 9 | 0.129245 |
| **Matlab Optimization Toolbox** | 8 | 0.660439 |

### b. Function B

Constraints:

- $0.5 * x + y \leq 1$

- $x \leq 1$
- $-x + y \leq 0$
- $y \leq -1.5$

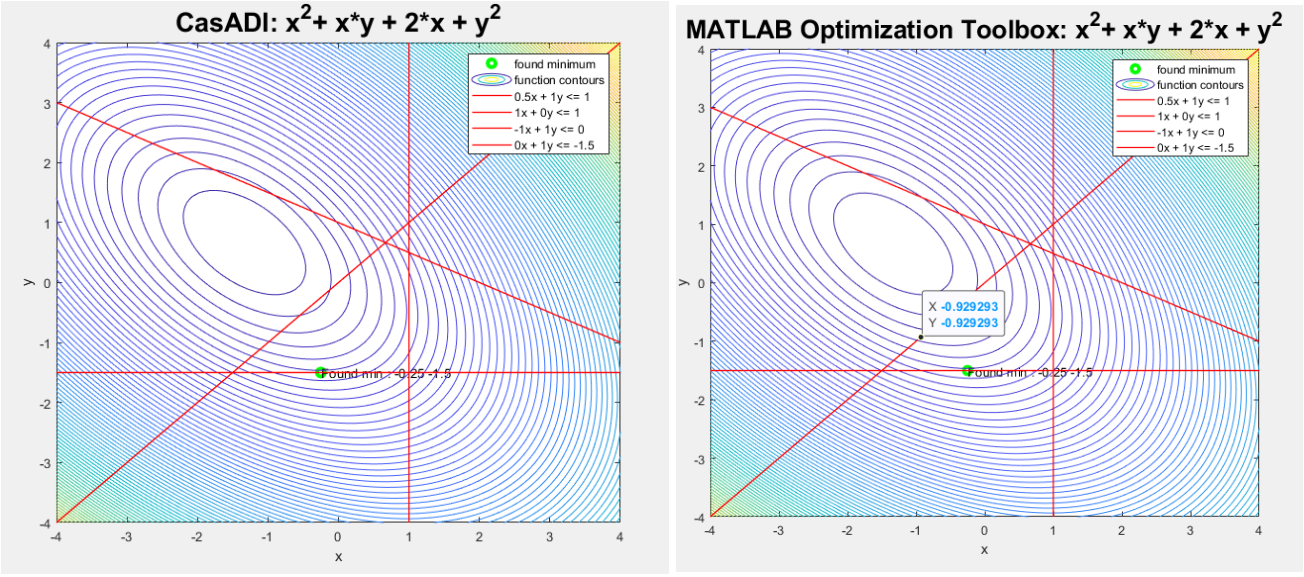


*Figure 5 From left: CasAdi and MATLAB Optimization Toolbox optimization results for function B*

Comparing execution time and number of iterations between CasAdi and Matlab Optimization Toolbox.

*Table 2 Comparasion of execution time and number of iterations between CasAdi and Matlab Optimization Toolbox*

| | Number of iterations | Overall evaluation time [s] |
|---|---|---|
| **CasAdi** | 5 | 0.126476 |
| **Matlab Optimization Toolbox** | 6 | 0.365013 |

**c. Function C**

Constraints:

- $x = 0.3$
- $y = 0.2$

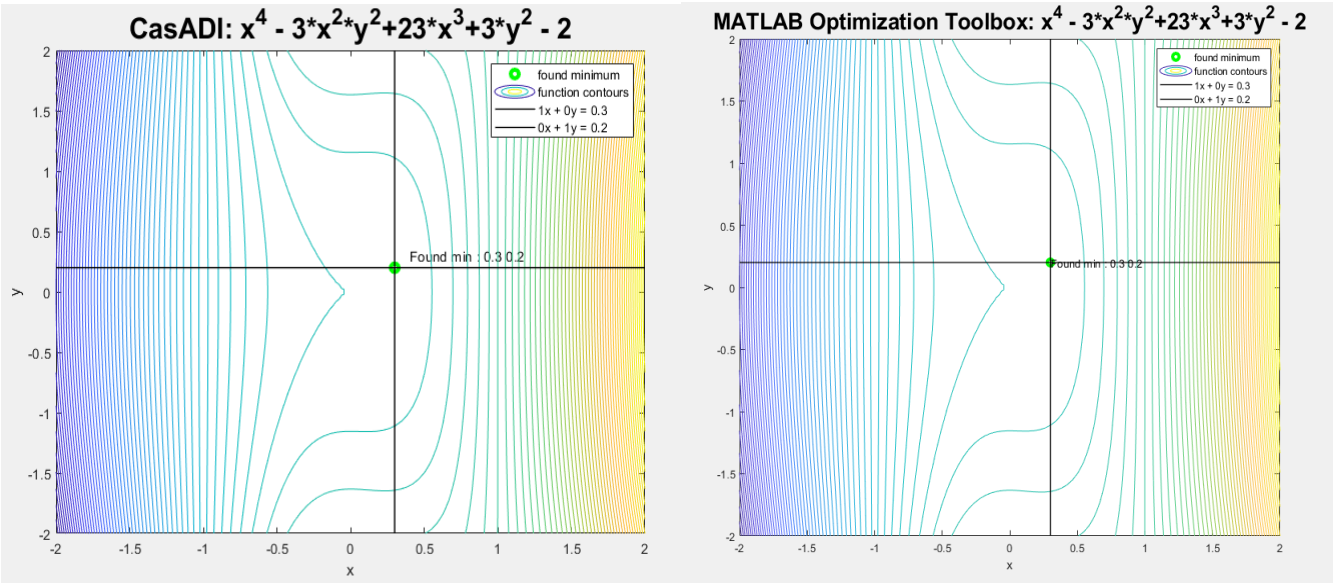


*Figure 6 From left: CasAdi and MATLAB Optimization Toolbox optimization results for function C*

*Table 3 Comparasion of execution time and number of iterations between CasAdi and Matlab Optimization Toolbox*

| | Number of iterations | Overall evaluation time [s] |
|---|---|---|

| | | |
|---|---|---|
| **CasAdi** | 1 | 0.114885 |
| **Matlab Optimization Toolbox** | 3 | 0.233951 |

## 6. Conclusion

Based on the optimization results of the 3 proposed functions, it can be concluded that the algorithm was implemented correctly. Implemented functions that draw contour plots of functions along with constraints make it easy to analyze the results and correctness of the solution. Furthermore, according to results the overall time of calculation and number of iterations is better for CasADi toolbox. Graph plots showing minimum of given functions looks similar for both implementations what may be caused by usage of same solver for both methods (solver used: *ipopt*). Since it is difficult to measure the optimization process itself with Matlab Optimization Toolbox, it is difficult to draw conclusions about the execution time of the optimization itself (CasAdi, on the other hand, reports the execution time). Therefore, time measurement was used from the beginning of the process initialization, that is, from the definition of the function, the assignment of equality and inequality constraints, until the end of the process. Based on these measurements, it can be concluded that the optimization using CasAdi is faster, but as mentioned, this applies to the entire process of calling the function.