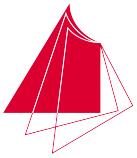

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Numerical optimization course – computer exercise

Linear and nonlinear optimization with CasADi

Prof. Dr. Moritz Diehl[1], **Prof. Dr. Angelika Altmann-Dieses**[2], **Adrian Bürger**[1,2]

[1] Systems Control and Optimization Laboratory, IMTEK, University of Freiburg

[2] Faculty of Management Science and Engineering, Karlsruhe University of Applied Sciences

1 Overview of the computer exercise

This computer exercise will provide a brief insight on how to formulate and solve linear and nonlinear programs within the symbolic optimization framework *CasADi* [1].

1.1 About CasADi

The open-source tool CasADi implements algorithmic differentiation on user-defined symbolic expressions and provides standardized interfaces to a variety of numerical routines: simulation and optimization, and solution of linear and nonlinear equations.

A key feature of these interfaces is that every user-defined CasADi function passed to a numerical solver automatically provides the necessary derivatives to this solver, without any additional user input. Often, the result of the numerical solver itself can be interpreted as a differentiable CasADi function, such that derivatives up to any order can be generated without actually differentiating the source code of the solver. Thus, concatenated and recursive calls to numerical solvers are possible and still result in differentiable CasADi functions.

CasADi is written in C++, but allows user input to be provided from either C++, Python, Octave or Matlab. When CasADi is used from the interpreter languages Python, Octave or Matlab, the user does not have any direct contact with C++; but because the internal handling of all symbolic expressions as well as the numerical computations are performed in a compiled environment, the speed of simulation or optimization computations is similar to the performance of compiled C-code. One particularly powerful optimization solver interfaced to CasADi is IPOPT, which is automatically provided in the standard CasADi installation. For more information on CasADi, please visit `http://casadi.org`.

1.2 Installing CasADi

CasADi can be installed on Windows, Linux and Mac, for detailed instructions please visit `https://github.com/casadi/casadi/wiki/InstallationInstructions`. For this exercise, we will use CasADi 3.1 from Matlab to show the solutions for the tasks given below. However, the templates and corresponding solutions will also be provided for Python in case some participants do not have access to a Matlab license.

2 Linear optimization exercise

An electricity trading company supplies customers in $n = 4$ areas, and purchases energy from $m = 3$ production sites. The contracted amount of energy delivery b_j for each customer in MWh, the production capacity a_i for each site in MWh as well as the cost for energy distribution c_{ij} from production site i to customer j in Euro/MWh are given as

$$a = \begin{pmatrix} 75.0 \\ 125.0 \\ 100.0 \end{pmatrix}, \quad b = \begin{pmatrix} 80.0 & 65.0 & 70.0 & 85.0 \end{pmatrix}, \quad C = \begin{pmatrix} 63.0 & 15.0 & 32.0 & 31.0 \\ 71.0 & 38.0 & 60.0 & 40.0 \\ 34.0 & 25.0 & 17.0 & 42.0 \end{pmatrix} \tag{1}$$

The sum of the contracted amounts of energy delivery is equal to the sum of production capacities, i. e., $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. The task now is to find an energy delivery plan in form of a transportation matrix $X = (x_{ij}) \in \mathbb{R}^{m \times n}$ that contains the energy deliveries x_{ij} from production site i to consumer j in MWh and minimizes the total transportation cost K as in

$$\underset{x_{ij}}{\text{minimize}} \quad K = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2a}$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_{ij} = a_i, \quad i = 1, \ldots, m, \tag{2b}$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1, \ldots, n, \tag{2c}$$

$$x_{ij} \geq 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n. \tag{2d}$$

This transportation problem can be formulated as a linear program in CasADi and solved using IPOPT.[1]

Tasks

a) Have a first look at the template provided for this task and at the functioning and syntax of CasADi. Identify the sections where the problem information and the symbolic optimization variables are initialized, where the several parts of problem (2) are formulated and where the `nlpsolver` class of CasADi that passes the problem to IPOPT is instantiated.

b) Complete the template using the information given in the problem description above and run the script. What is the minimal cost for transportation? How has the delivery been organized?

c) Assume now that we have an oversupply of produced energy, i. e., $\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$, with

$$a = \begin{pmatrix} 105 \\ 125 \\ 100 \end{pmatrix}. \tag{3}$$

For finding a new cost-optimal transportation plan, we can still formulate an optimization problem similar to (2) if we introduce an additional, virtual customer whose contracted amount of energy is

[1] Please note that we use the nonlinear program solver IPOPT [2] only for convenience here. Using a linear or quadratic program solver, the problem could be solved faster than by IPOPT.

$b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$ and therefore exactly consumes the oversupply. This virtual consumer can be supplied by all producers without invoking any cost.

Implement the changes accordingly, and look at the resulting cost and transportation plan. How did the cost change? Which production sites deliver the virtual customer, and are therefore not able to distribute all energy they produced?

3 Nonlinear optimization exercise

Within a production process, five spheres s_i with $i = 1, \ldots, 5$ shall be cut out from a quadratic plate with egde size $a = 10\,\text{cm}$. Three of those spheres shall be of radius R and two of radius $2R$. The objective is to maximize the radius R.

The center of each sphere s_i can be expressed in Cartesian coordinates (x_i, y_i) on the plate, and are to be optimized in addition to the radius R. The spheres may not lie outside of the plate or overlap each other. To ensure this, the minimum distance between the centers of all spheres from each other as well as the edges of the plate must enter the constraints of the optimization problem. A depiction of a possible but suboptimal solution with $R = 1$ is given in Figure 1.

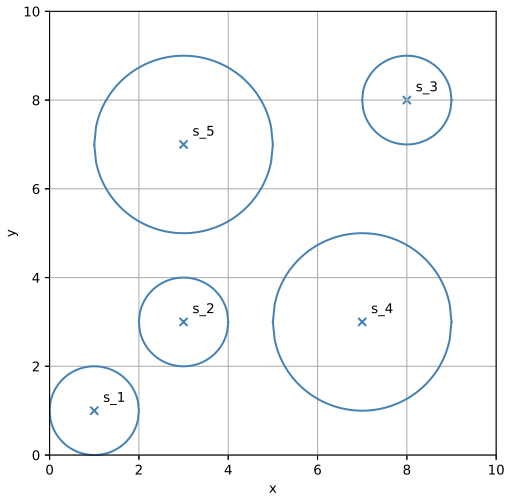

Figure 1: Graphical depiction of a possible, but suboptimal solution for Task 2 with $R = 1$.

The problem can be formulated as a nonlinear program in CasADi and solved using IPOPT, where the following sets of constraints must enters the optimization problem:

1. The radii of two of the spheres must be twice a big as the radii of the three other spheres, and must therefore fulfill the condition

$$r_i = R, \quad i = 1, \ldots, 3, \tag{4}$$
$$r_j = 2R, \quad j = 4, \ldots, 5. \tag{5}$$

2. The minimum distance of a sphere's x-coordinate from the left egde and the right edge of the plate must be greater or equal than its radius r_i, the same must hold for the distance of the

y-coordinate from the top edge and bottom edge of the plate:

$$x_i - r_i \geq 0, \quad i = 1, \ldots, 5, \tag{6}$$
$$x_i + r_i \leq a, \quad i = 1, \ldots, 5, \tag{7}$$
$$y_i - r_i \geq 0, \quad i = 1, \ldots, 5, \tag{8}$$
$$y_i + r_i \leq a, \quad i = 1, \ldots, 5. \tag{9}$$

3. The distance of two sphere's centers must be greater or equal to the sum of both sphere's radii, which can be expressed simply by using the Pythagorean theorem as

$$(x_i - x_j)^2 + (y_i - y_j)^2 - (r_i + r_j)^2 \geq 0 \quad i = 1, \ldots, 5, \quad i < j \leq 5. \tag{10}$$

Tasks

a) Complete the template provided for this task with the information given above and run the script. On success, you should see a plot that depict the positioning of the spheres on the plate, and they should neither interlap nor lie outside the plate. How big is R if you use the initial guesses for the circles coordinates that are already contained in the template?

b) Looking at the plot, could you think of a distribution for the spheres that might lead to even bigger values for R? Try setting different initial guesses for the spheres' center coordinates, and write down your best solution for R.

4 (If you want to do more:) Dynamic optimization exercise

This additional exercise will give you an insight on how to solve Optimal Control Problems (OCPs) with CasADi. For this, we will implement an adapted version of the BERGMAN model of the human glucose insulin system [3], and compute the optimal insulin infusion rate of a (simplified) artificial pancreas for a type 1 diabetic patient to react on a meal disturbance of the blood glucose level, according to [4], [5], [6]. The adapted BERGMAN model is an Ordinary Differential Equation (ODE) of the form

$$\begin{aligned}
\dot{G}(t) &= -P_1 \cdot (G(t) - G_{\text{init}}) - (X(t) - X_{\text{init}}) \cdot G(t) + u_{\text{meal}}(t) \\
\dot{X}(t) &= -P_2 \cdot (X(t) - X_{\text{init}}) + P_3 \cdot (I(t) - I_{\text{init}}) \\
\dot{I}(t) &= -n \cdot I(t) + \frac{D(t)}{V_1} \\
\dot{D}(t) &= u_{\text{icr}}(t),
\end{aligned} \tag{11}$$

where

- $G(t)$ describes the plasma glucose concentration in $\frac{\text{mmol}}{\text{L}}$ at a time t,
- $X(t)$ is a proportional to $I(t)$ in remote compartment $\frac{\text{mU}}{\text{L}}$,
- $I(t)$ describes the plasma insulin concentration in $\frac{\text{mU}}{\text{L}}$ above the basal value, and
- $D(t)$ describes the insulin infusion rate in $\frac{\text{mU}}{\text{min}}$ [4], [6].

The parameter values for a type 1 diabetic are given in [5] as

- $P_1 = 0.028753\,\text{min}^{-1}$,
- $P_2 = 0.028344\,\text{min}^{-1}$,

- $P_3 = 5.035 \cdot 10^{-5}\,\frac{\mathrm{mU}}{\mathrm{L}}$,
- $n = 5.0/54.0\,\mathrm{min}^{-1}$ and
- $V_1 = 12.0\,\mathrm{L}$.

The control $u_{\mathrm{icr}}(t)$ describes the change of the insulin infusion rate in $\frac{\mathrm{mU}}{\mathrm{min}\cdot\mathrm{s}}$. Choosing the values for this control in an optimal way will later allow us to optimally control the insulin infusion rate with regard to a specified objective. The control $u_{\mathrm{meal}}(t)$ on the other hand, which is the meal disturbance of the plasma glucose concentration G, we can not optimize, but assume it to take values from the meal disturbance function [5]

$$u_{\mathrm{meal}}(t) = 3 \cdot e^{0.05 \cdot t}\,\frac{\mathrm{mmol}}{\mathrm{L}}. \tag{12}$$

For the problem formulation in CasADi, we set up a state vector x that contains all states as follows

$$x(t) = \begin{pmatrix} G(t) \\ X(t) \\ I(t) \\ D(t) \end{pmatrix}, \tag{13}$$

and for the simulation we assume the initial values for the states to be [6]

$$x_0 = \begin{pmatrix} G_0 \\ X_0 \\ I_0 \\ D_0 \end{pmatrix} = \begin{pmatrix} 4.5\,\frac{\mathrm{mmol}}{\mathrm{L}} \\ 15.0\,\frac{\mathrm{mU}}{\mathrm{L}} \\ 15.0\,\frac{\mathrm{mU}}{\mathrm{L}} \\ 0.0\,\frac{\mathrm{mU}}{\mathrm{min}} \end{pmatrix}. \tag{14}$$

Further, we define a control vector u that contains all controls as

$$u(t) = \begin{pmatrix} u_{\mathrm{icr}}(t) \\ u_{\mathrm{meal}}(t) \end{pmatrix}. \tag{15}$$

We want to formulate and solve an optimal control problem for (11) so that the change of the insulin infusion rate u_{icr} is controlled in a way that the plasma glucose concentration G and the insulin infusion rate D are pushed towards their reference values $G_{\mathrm{ref}} = 5.0\,\frac{\mathrm{mmol}}{\mathrm{L}}$ and $D_{\mathrm{ref}} = 13.0\,\frac{\mathrm{mU}}{\mathrm{min}}$, respectively. For this, a quadratic objective function is used.

Over the whole control horizon, the insulin infusion rate D has to stay between its minimum value $D_{\mathrm{min}} = 5.0\,\frac{\mathrm{mU}}{\mathrm{min}}$ and it's maximum value $D_{\mathrm{max}} = 20.0\,\frac{\mathrm{mU}}{\mathrm{min}}$. All other states display concentration values, and therefore must not become negative. While the change of the insulin infusion rate u_{icr} is not bounded, the meal disturbance u_{meal} is exactly constrained to the meal disturbance function (12). These conditions lead to the following OCP formulation in continuous time:

$$\begin{aligned}
\underset{u_{\mathrm{icr}}(\cdot)}{\text{minimize}} \quad & \int_{t_0}^{t_{\mathrm{end}}} \left((G(t) - G_{\mathrm{ref}})^2 + (D(t) - D_{\mathrm{ref}})^2 \right) \mathrm{d}t \\
\text{subjectto} \quad & (11) && t \in [t_0, t_{\mathrm{end}}] \\
& (12) && t \in [t_0, t_{\mathrm{end}}] \\
& x_{\mathrm{min}} \leq x(t) \leq x_{\mathrm{max}} && t \in [t_0, t_{\mathrm{end}}] \\
& x(0) = x_0
\end{aligned} \tag{16}$$

with state bounds defined as

$$x_{\min} = \begin{pmatrix} G_{\min} \\ X_{\min} \\ I_{\min} \\ D_{\min} \end{pmatrix} = \begin{pmatrix} 0.0\,\frac{\mathrm{mmol}}{\mathrm{L}} \\ 0.0\,\frac{\mathrm{mU}}{\mathrm{L}} \\ 0.0\,\frac{\mathrm{mU}}{\mathrm{L}} \\ 5.0\,\frac{\mathrm{mU}}{\mathrm{min}} \end{pmatrix} \tag{17}$$

and

$$x_{\max} = \begin{pmatrix} G_{\max} \\ X_{\max} \\ I_{\max} \\ D_{\max} \end{pmatrix} = \begin{pmatrix} \infty\,\frac{\mathrm{mmol}}{\mathrm{L}} \\ \infty\,\frac{\mathrm{mU}}{\mathrm{L}} \\ \infty\,\frac{\mathrm{mU}}{\mathrm{L}} \\ 20.0\,\frac{\mathrm{mU}}{\mathrm{min}} \end{pmatrix}. \tag{18}$$

This OCP shall now be transformed to a Nonlinear Program (NLP) using direct multiple shooting, which then can be handed to CasADi's `nlpsol` function that can automatically generate the necessary derivatives using algorithmic differentiation and pass the optimization problem to a NLP solver interfaced by CasADi, which by default is IPOPT.

For this, the continuous time problem first needs to be discretized, so we divide the time horizon $[t_0, t_{\mathrm{end}}]$ with $t_0 = 0\,\mathrm{s}$ and $t_{\mathrm{end}} = 200.0\,\mathrm{s}$ into $N = 250$ control intervals. Further, we need to introduce optimization variables s for the discrete states and q for the discrete controls. Since $x \in \mathbb{R}^{n_\mathrm{x}}$ and $u \in \mathbb{R}^{n_\mathrm{u}}$, we need to introduce a total $s \in \mathbb{R}^{n_\mathrm{x} \times (N+1)}$ and a total $q \in \mathbb{R}^{n_\mathrm{x} \times N}$.

Iteratively, we need to add the multiple shooting continuity constraints to the NLP formulation. As integrator $r(\cdot)$, we use CVODES of SUNDIALS [7], which is shipped with the CasADi package. Also, we need to add the constraints for the upper and lower bounds of s and q to the NLP that result from the upper and lower bounds of the states and controls, respectively.

With CasADi's `nlpsol` function, only inequality constraints can be used to formulate NLPs, so equality constraints need to be reformulated accordingly, i. e.

$$x(0) = x_0 \tag{19}$$

needs to be formulated as

$$x_0 \leq x(0) \leq x_0. \tag{20}$$

These reformulations lead to the NLP

$$\begin{aligned}
\underset{s(\cdot), q(\cdot)}{\text{minimize}} \quad & \underbrace{\sum_{k=0}^{N} (s_\mathrm{k} - s_\mathrm{ref})^\mathrm{T} (s_\mathrm{k} - s_\mathrm{ref})}_{=f} \\
\text{subjectto} \quad & \underbrace{0}_{=g_{\min}} \leq \underbrace{s_{\mathrm{k}+1} - r(s_\mathrm{k}, q_\mathrm{k})}_{=g} \leq \underbrace{0}_{=g_{\max}}, \qquad k = 0, \ldots, N-1 \\
& x_{\min} \leq s_\mathrm{k} \leq x_{\max} \qquad k = 0, \ldots, N-1 \\
& q_{\min,\mathrm{k}} \leq q_\mathrm{k} \leq q_{\max,\mathrm{k}} \qquad k = 0, \ldots, N-1 \\
& x_0 \leq s_0 \leq x_0
\end{aligned} \tag{21}$$

with

$$s_{\mathrm{ref}} = \begin{pmatrix} G_{\mathrm{ref}} \\ 0 \\ 0 \\ D_{\mathrm{ref}} \end{pmatrix} \tag{22}$$

and

$$q_{\mathrm{min,k}} = \begin{pmatrix} -\infty \\ 3 \cdot e^{0.05 \cdot k\Delta t} \end{pmatrix}, \tag{23}$$

$$q_{\mathrm{max,k}} = \begin{pmatrix} \infty \\ 3 \cdot e^{0.05 \cdot k\Delta t} \end{pmatrix}. \tag{24}$$

As initial values for the optimization variables s and q, the corresponding values of states and controls from an ex ante simulation wit $u_{\mathrm{icr}}(t) = 0.0\,\frac{\mathrm{mU}}{\mathrm{min \cdot s}}$ can be used.

Tasks

a) With the information given in this section, complete the template file provided for this task.

References

[1] Andersson, Joel: *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.

[2] Wächter, Andreas; Biegler, Lorenz T.: *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*. Mathematical Programming, 106(1):25–57, 2006.

[3] Bergman, R. N.; Philips, L. S.; Cobelli, C.: *Physiological evaluation of factors controlling glucose tolerance in man*. Journal of Clinical Investigation, 68:1456–1467, 1981.

[4] Lynch, S. M.; Bequette, B. W.: *Estimation-based model predictive control of blood glucose in type I diabetics: a simulation study*. In *Bioengineering Conference, 2001. Proceedings of the IEEE 27th Annual Northeast*, pages 79–80, 2001.

[5] Lynch, S. M.; Bequette, B. W.: *Model predictive control of blood glucose in type I diabetics using subcutaneous glucose measurements*. In *American Control Conference, 2002. Proceedings of the 2002*, volume 5, pages 4039–4043 vol.5, 2002.

[6] Modelon AB: *blood_glucose_opt.py*. http://www.jmodelica.org/api-docs/pyjmi/EXAMPLE_blood_glucose_opt.html, visited on November 30, 2015.

[7] Lawrence Livermore National Laboratory: *SUNDIALS: SUite of Nonlinear and DIfferential/ALgebraic Equation Solvers*. https://computation.llnl.gov/projects/sundials, visited on April 11, 2017.