Workshop — Doktorand:innenseminar — SPP 2364

Nonlinear Optimization and Direct Optimal Control for Practitioners

Katrin Baumgärtner, Florian Messerer, Prof. Dr. Moritz Diehl

The aim of this workshop is to give you some hands on experience on methods and in particular software for numerical simulation and optimal control. The workshop exercises are based on `python`, `CasADi`, and `acados`. If you have problems following the installation instructions, don't hesitate to reach out!

Prerequisites and installation instructions

The following steps will get you started for the exercise sessions.

1. Install VSCode and Docker Desktop
2. Install the app *Dev Containers* in VSCode: Open the Extension Marketplace, search and install *Dev Containers*.
3. Download and unpack the attached zip folder *template-casadi-acados-course* from the course page.
4. Start VSCode with *template-casadi-acados-course* as the root folder:
   - A message should appear on the bottom right corner asking you to build the container. Alternatively, you can open the Command Palette of VSCode and type in *Dev Container: Rebuild and Reopen in Container*.
   - Click on it to start building the image and wait until the container runs. Depending on your CPU and internet connection, the process can take several minutes. You can check that you are running the dev container by looking to the bottom left corner where a green field should state *Dev Container: ....*

About `CasADi`

The open-source tool `CasADi`[1] 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`. One particularly powerful optimization solver interfaced to CasADi is `IPOPT`[2], which is automatically provided in the standard `CasADi` installation.

[1] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019

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

About acados

acados[3] is a software package for the efficient online solution of optimal control and estimation problems. It provides a collection of computationally efficient building blocks tailored to optimal control and estimation problems. Among others, it implements: modules for the integration of ordinary differential equations (ODE) and differential-algebraic equations (DAE), interfaces to state-of-the-art QP solvers like HPIPM, qpOASES, DAQP, qpDUNES and OSQP, condensing routines and nonlinear programming solvers based on the real-time iteration framework. The back-end of acados uses the high-performance linear algebra package BLASFEO to boost computational efficiency for small to medium scale matrices typical of embedded applications. MATLAB, Octave and python interfaces can be used to conveniently describe optimal control problems and generate self-contained C code that can be readily deployed on embedded platforms.

[3]Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, pages 147–183, Oct 2021

Workshop — Doktorand:innenseminar — SPP 2364

Part 1: Nonlinear Optimization

Katrin Baumgärtner, Florian Messerer, Prof. Dr. Moritz Diehl

The following exercise has been part of the SPP 2364 online workshop on "Nonlinear Optimization and Direct Optimal Control". If you followed the workshop and/or have some experience with `CasADi` *already, feel free to skip this exercise and continue directly with Part 2.*

If you haven't used `CasADi` *yet, take some time reading its documentation before starting with the exercise.*

Within a production process, five spheres s_i with $i = 1, \dots, 5$ shall be cut out from a quadratic plate with egde size $a = 10$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.

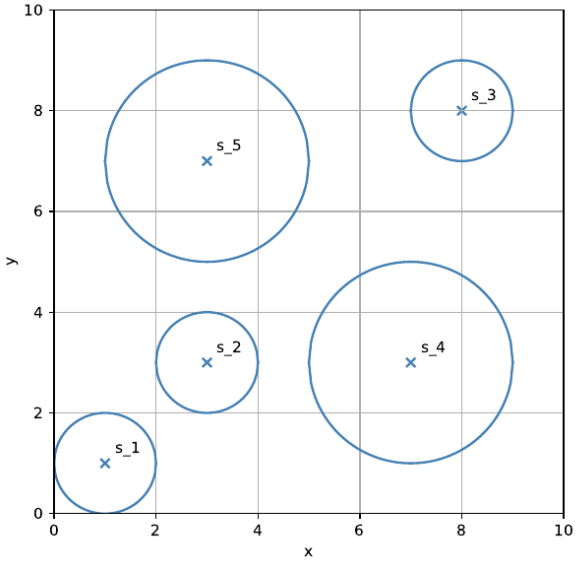


Figure 1: Graphical depiction of a possible, but suboptimal solution with $R = 1$.

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

(a) 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

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

(b) The minimum distance of the x-coordinate of any sphere 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.

$$x_i - r_i \geq 0, \quad i = 1, \ldots, 5, \tag{4}$$
$$x_i + r_i \leq a, \quad i = 1, \ldots, 5, \tag{5}$$
$$y_i - r_i \geq 0, \quad i = 1, \ldots, 5, \tag{6}$$
$$y_i + r_i \leq a, \quad i = 1, \ldots, 5, \tag{7}$$
$$\tag{8}$$

(c) The distance of the centers of two spheres must be greater or equal to the sum of both spheres' radii, which can be expressed simply by using the Pythagorean theorem as

$$(x_i - x_j)^2 + (y_i - y_j)^2 - (r_i + r_j)^2 \geq 0, \quad i = 1, \ldots, 4, \quad j = i + 1, \ldots, 5. \tag{9}$$

Tasks

1. 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 overlap 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?

2. 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.

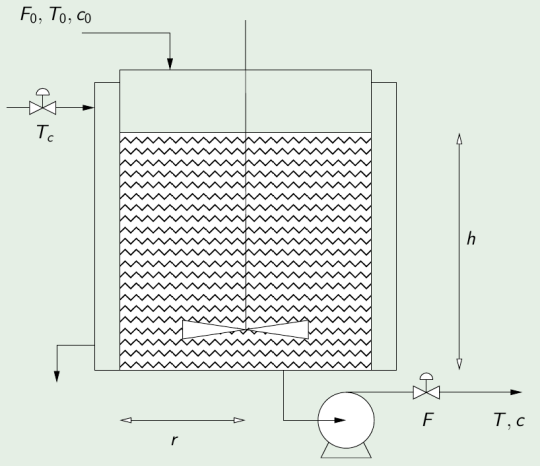
3. Check whether the gradient of the Lagrangian is indeed zero at the solution you computed.
   Hint: You might want to find out how `solver.get_function()` works.

Workshop — Doktorand:innenseminar — SPP 2364

Part 2: Numerical Simulation

Katrin Baumgärtner, Florian Messerer, Prof. Dr. Moritz Diehl

In this and all following exercises, we consider a nonlinear continuous stirred-tank reactor (CSTR). In particular, we would like to compare the performance of different integrators in terms of simulation accuracy and computation time.



An irreversible, first-order reaction $A \rightarrow B$ occurs in the liquid phase and the reactor temperature is regulated with external cooling. Mass and energy balances lead to the following nonlinear state space model:

$$\begin{aligned}
\dot{c} &= \frac{F_0(c_0 - c)}{\pi r^2 h} - k_0 \exp\left(-\frac{E}{RT}\right) c \\
\dot{T} &= \frac{F_0(T_0 - T)}{\pi r^2 h} - \frac{\Delta H}{\rho C_p} k_0 \exp\left(-\frac{E}{RT}\right) c + \frac{2U}{r \rho C_p}(T_c - T) \\
\dot{h} &= \frac{F_0 - F}{\pi r^2}
\end{aligned}$$

with states $x = (c, T, h)$ where c is the concentration of substance A, T is the reactor temperature and h is the height. The controls $u = (T_c, F)$ are the coolant liquid temperature T_c and the outlet flowrate F.

1. Complete the provided template with your own implementation of a Runge-Kutta integrator of order 4 with one step per integration interval and run the script. Do the results look reasonable? Do the results change if you perform 100 integration steps per interval?

2. Complete the provided template for using the `CasADi` interface to the integrator `CVodes`. Compare the simulated trajectories as well as the computation time with your own implementation of the RK4 integrator.

3. Change the initial state to $x_0 = (1.141, 97, 0.01)$ and try to run the simulation again. What happens? Can you explain these results by analyzing the system dynamics? What conclusions do you draw regarding the optimal control problem you would formulate to control the above system?

Workshop — Doktorand:innenseminar — SPP 2364

Part 3: Direct Optimal Control with `CasADi` (and `acados`)

Katrin Baumgärtner, Florian Messerer, Prof. Dr. Moritz Diehl

In this exercise, we implement an NMPC controller for the nonlinear continuous stirred-tank reactor (CSTR) using `CasADi` (and `acados`). We aim at tracking a reference x_{ref} and u_{ref} using a tracking cost of the form

$$l(x,u) = \frac{1}{2}\|x - x_{\text{ref}}\|_Q^2 + \frac{1}{2}\|u - u_{\text{ref}}\|_R^2, \qquad E(x) = \frac{1}{2}\|x - x_{\text{ref}}\|_P^2 \tag{10}$$

with weighting matrices Q, R and P.

1. Fill in the gaps in the provided template specifying (a) the stage cost, (b) terminal cost and (c) the state bounds (having in mind the previous exercise).

2. Formulate the discrete-time optimal control problem using direct collocation and an implicit Euler scheme. Run the script to see how your controller performs in closed-loop. How does the behaviour of the closed-loop system change if you adapt the weighting matrices Q and R?

   For a single integration step, the problem is given as:

$$\begin{aligned}
\min_{x,u,k} \quad & \sum_{n=0}^{N-1} l(x_n, u_n) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0, \\
& x_{n+1} = x_n + \Delta t\, k_n, \quad n = 0, \dots, N-1, \\
& k_n = f(x_{n+1}, u_n), \quad n = 0, \dots, N-1, \\
& u_{\min} \le u_n \le u_{\max}, \quad n = 0, \dots, N-1, \\
& x_{\min} \le x_n \le x_{\max}, \quad n = 0, \dots, N.
\end{aligned} \tag{11}$$

3. Plot the sparsity pattern of the Hessian of the Lagrangian of the optimal control problem. How many structural zeros does it include? (What are structural zeros?)
   Hint: You might want to use `solver.get_function()`.

4. Similarly, formulate a discrete-time optimal control problem using the multiple shooting formulation and where the discrete-time dynamics are given by your RK4 integrator.

$$\begin{aligned}
\min_{x,u} \quad & \sum_{n=0}^{N-1} l(x_n, u_n) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0, \\
& x_{n+1} = \phi(x_n, u_n), \quad n = 0, \dots, N-1, \\
& u_{\min} \le u_n \le u_{\max}, \quad n = 0, \dots, N-1, \\
& x_{\min} \le x_n \le x_{\max}, \quad n = 0, \dots, N,
\end{aligned} \tag{12}$$

5. How many optimization variables are there in the collocation and multiple shooting formulation? How does this number change if you change the number of integration steps?

6. (*) Complete the template for a NMPC controller using `acados` and compare your results in terms of closed-loop control performance and computation time.

7. (*) Play around with some of the controller parameters, e.g. maximum number of iterations, horizon length, etc. How is the control performance and the computation time affected by these changes?