Exercises for Lecture Course on Numerical Optimal Control (NOC)
Albert-Ludwigs-Universität Freiburg – Summer Term 2022

Exercise 2: Nonlinear Optimization and Newton-type Methods

Prof. Dr. Moritz Diehl, Florian Messerer, Andrea Zanelli, Dimitris Kouzoupis, Yizhen Wang

In this exercise we will start using solvers for nonlinear and nonconvex optimization problems and we will implement a simple Newton-type algorithm for unconstrained problems.

1. **The Rosenbrock problem.** Consider the following unconstrained optimization problem:

$$\min_{x,y} \quad f(x,y) := (1-x)^2 + 100(y-x^2)^2. \tag{1}$$

Such a problem is commonly referred to as Rosenbrock problem. Have a look at the script provided with this exercise that formulates (1) using CasADi and solves it with the solver for nonlinear nonconvex optimization problems IPOPT. In this exercise we will implement a simple Newton-type algorithm that can be used to solve such a problem.

(a) Compute on paper the gradient of f and its Hessian.

(b) Implement two functions that take as input arguments x and y and return $\nabla f(x,y)$ and $\nabla^2 f(x,y)$ respectively.

(c) To simplify notation we introduce $w = (x,y)$. We now want to numerically solve the optimization problem by finding a point w^* at which $\nabla f(w^*) = 0$. Implement the following Newton-type method:

$$w_{k+1} = w_k - M_k^{-1} \nabla f(w_k), \tag{2}$$

where $M \approx \nabla^2 f(w^k)$ is an approximation of the exact Hessian. Test your implementation with two different Hessian approximations: i) $M_k = \rho I_2$, with $I_2 \in \mathbb{R}^{2\times 2}$ the identity matrix, for different values of $\rho \in \mathbb{R}_{++}$ and ii) $M_k = \nabla^2 f(w_k)$. Initialize the iterates at $w_0 = (1,\ 1.1)^T$ and run the algorithm for 1000 iterations. Plot the iterates in the x-y space. When using the fixed Hessian approximation, does the algorithm converge for $\rho = 100$? And for $\rho = 500$?

(d) Use now CasADi to compute the gradient and Hessian of f and use it in your implementation of the Newton method.

Hint: once you have created a CasADi expression, you can compute its Jacobian and Hessian calling the CasADi functions `jacobian` *and* `hessian`:

```
% MATLAB
import casadi.*
x = MX.sym('x',2,1);
expr = sin(x(1))*x(2);
j_expr = jacobian(expr,x);
J = Function('J', {x}, {j_expr});
% hessian() returns only hessian
% expression.
h_expr = hessian(expr, x);
H = Function('H', {x}, {h_expr});
```

```
# Python
import casadi as ca
x = ca.MX.sym('x',2,1)
expr = ca.sin(x[0])*x[1]
j_expr = ca.jacobian(expr, x)
J = ca.Function('J', [x], [j_expr])
# hessian() returns gradient
# expression as the second value.
h_expr, __ = ca.hessian(expr, x)
H = ca.Function('H', [x], [h_expr])
```

2. **A simple dynamic optimization problem.** Consider the problem of finding the optimal way of throwing two balls from different locations such that their distance after a fixed time T is minimized. The dynamics of the system taken into account can be modeled by the following differential equation:

$$\begin{aligned}
\dot{p}_{1y} &= v_{1y}, & \dot{p}_{2y} &= v_{2y} \\
\dot{p}_{1z} &= v_{1z}, & \dot{p}_{2z} &= v_{2z} \\
\dot{v}_{1y} &= -(v_{1y} - w) \left\| v_1 - [w,\, 0]^T \right\| d_1, & \dot{v}_{2y} &= -(v_{2y} - w) \left\| v_2 - [w,\, 0]^T \right\| d_2 \\
\dot{v}_{1z} &= -v_{1z} \left\| v_1 - [w,\, 0]^T \right\| d_1 - g, & \dot{v}_{2z} &= -v_{2z} \left\| v_2 - [w,\, 0]^T \right\| d_2 - g,
\end{aligned}$$

where p_{iy} and p_{iz} represent the y and z coordinate of the i-th ball respectively and v_{iy} and v_{iz} the components of its velocity. The two balls are subject to drag forces with drag coefficients d_1 and d_2, side wind w and gravitational acceleration g. In order to achieve the desired goal, we formulate the following optimization problem:

$$\begin{aligned}
\min_{v_{\text{start}}} \quad & \|p_1(T) - p_2(T)\|_2^2 && \text{(3a)} \\
\text{s.t.} \quad & p_{1z}(T) \geq 0, \quad p_{2z}(T) \geq 0, && \text{(3b)} \\
& \|v_1\|_2^2 \leq \bar{v}^2, \quad \|v_2\|_2^2 \leq \bar{v}^2, && \text{(3c)}
\end{aligned}$$

where $v_{\text{start}} := [v_{1y}(0),\, v_{1z}(0),\, v_{2y}(0),\, v_{2z}(0)]^T$ are the decision variables and $p(T)$ is the output of an RK4 integrator that discretizes the dynamics of the system. Additional constraints have been added to the formulation that represent the requirement that the balls have to be above the ground at time T (notice that, due to the dynamics of the system this implies that the balls are above the ground at every time $t \in [0,\, T]$).

(a) A template MATLAB/Python function that takes the initial velocities of the balls as an input and returns the final position at time T is provided with this exercise. This function can be used both with numerical and CasADi symbolic inputs. Complete the provided template and use it to generate a CasADi expression for $p(T)$. Use $N = 100$ equidistant intermediate steps and $T = 0.5\,\text{s}$. Set $d_1 = 0.1\,\text{m}^{-1}$, $d_2 = 0.5\,\text{m}^{-1}$ and $w = 2\,\text{m/s}$.

(b) Using CasADi, formulate the described dynamic optimization problem (3) and solve it using IPOPT. Fix $\bar{v} = 15\,\text{m/s}$ and $p_1(0) = [0,\, 0]^T$, $p_2(0) = [10,\, 0]^T$. Once you have solved the optimization problem, simulate the system for the optimal initial velocities and plot the resulting trajectories in space.

Hint: you can have a look at the constrained Rosenbrock example provided with this exercise to learn how to formulate constrained problems in CasADi.

(c) [**Bonus**] Consider the case where there is no drag ($d_1 = d_2 = 0\,\text{m}^{-1}$). What kind of optimization problem does (3) become?

(d) [**Bonus**] Change (3) to an unconstrained problem by removing (3b) and (3c). Set $\|v_1\|_2 = \bar{v}$ and $\|v_2\|_2 = \bar{v}$ and reformulate (3) such that the angles $\alpha_1 := \arccos(v_{1y}(0)/\,\|v_1\|)$ and $\alpha_2 := \arccos(-v_{2y}(0)/\,\|v_2\|)$ are the only decision variables. In this way a two-dimensional dynamic optimization problem is obtained. Use the Newton-type method implemented at point 1.b to solve this problem.