Hands-on CasADi course on optimal control, Yacoda

Exercise: Nonlinear programming

In this exercise, we'll be finding a minimizer of the optimization problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & x_1^2 + \tanh(x_2)^2 \\ \text{subject to} \quad & \cos(x_1 + x_2) + 0.5 = 0, \\ & \sin(x_1) + 0.5 \leq 0, \end{aligned}$$

with $x \in \mathbb{R}^2$.

1 SQP Method

Tasks:

1. For the given problem, write symbolic expressions for f, g, h from the standard form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g(x) = 0, \\ & h(x) \leq 0. \end{aligned} \tag{1}$$

   Start out with creating a vector symbol for the decision variables:

```
x = MX.sym('x',2);
```

2. Write a symbolic expression for the Lagrangian $\mathcal{L} = f + \lambda g + \nu h$. Write a CasADi `Function` to evaluate the Lagrangian at $x = \begin{bmatrix} -0.5 \\ -1.8 \end{bmatrix}, \lambda = 2, \nu = 3$. Verify that it evaluates to `0.875613`.

3. Consider the QP approximation that shares its KKT conditions for optimality with the original NLP:

$$\begin{aligned} \underset{\Delta x}{\text{minimize}} \quad & f_k + \frac{\partial f_k}{\partial x}\Delta x + \frac{1}{2}\Delta x^T \frac{\partial^2 \mathcal{L}_k}{\partial x^2}\Delta x \\ \text{subject to} \quad & g_k + \frac{\partial g_k}{\partial x}\Delta x = 0, \\ & h_k + \frac{\partial h_k}{\partial x}\Delta x \leq 0. \end{aligned} \tag{2}$$

   At a given iterate k, which parts in this equation are known? Extend the above CasADi `Function` with extra outputs to be able to compute all these parts. Use `gradient`, `hessian`, `jacobian`.

4. At the working point $x_0 = \begin{bmatrix} -0.5 \\ -1.8 \end{bmatrix}, \lambda_0 = 0, \nu_0 = 0$, numerically evaluate the function you just composed, and verify that the dimensions of all outputs are consistent with equation 2. Leave the outputs in `DM` form[1] and verify that the Hessian of the Lagrangian is equal to `[2`

[1] Do not use `full(...)`

0; 0 -0.3499].

Note that this Hessian is not positive definite. We will ignore this fact here.

5. To solve the QP, we will make use of CasADi's low-level QP interface. Its expected form is:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & G^T x + \frac{1}{2} x^T H x \\ \text{subject to} \quad & v^{\text{lba}} \leq Ax \leq v^{\text{uba}}, \end{aligned} \tag{3}$$

   Using concatenation operations `[...    ...]` and `[...;...]`, compose the matrices of Equation 3 from the numerical outputs of the previous question.

   Verify that you obtain:

```
H:
[[2, 0],
 [0, -0.3498874751520534]]
G:[-1, -0.196099]
A:
[[0.7457052121767203, 0.7457052121767203],
 [0.8775825618903728, 00]]
lba:[0.166276, -inf]
uba:[0.166276, -0.0205745]
```

6. Inspect the sparsity of H and A with `sparsity(A)`, and also inspect the spy printout with `spy(A)`.

   Explain how A seems to be sparser than H, while the numerical printout implies otherwise.

7. Using the following template to create a QP solver:

```
qp_struct = struct('a',...,'h',...);
solver = conic('solver','qrqp',qp_struct);
```

   Replace the dots with sparsities of A and H.

   What data-type is `solver`?

8. The print-representation of `solver` reveals that a lot of inputs can be given. Only a handful are relevant to us. Use the keyword-value syntax of the CasADi Function `solver` to evaluate the solver Function numerically.

   Verify that you end up with

```
dx = [-0.02344447381848419, 0.2464226943869885]
lambda = [0.3785941041969475]
nu = [0.871222132298292]
```

   Note: you may obtain higher precision in the printout with `DM.set_precision(16)`.

9. We now have a mechanism to numerically compute Δx given the numerical linearisations of the nonlinear program. Use this in a loop to perform four SQP iterations. Verify that your decision variables converge as follows:

```
x = [-0.5234444738184842, -1.553577305613012]
x = [-0.5235987687270579, -1.570711791832387]
x = [-0.5235987755982988, -1.570796324731945]
x = [-0.5235987755982988, -1.570796326794897]
```

Note: by default, the QP solver shows iterations. To hide this output, set option `print_iter` to false by passing an extra structure argument to `conic`.

2 Interior Point Method

1. Let's look at the problem's KKT conditions and their relaxation through a barrier parameter τ:

$$\begin{cases} \nabla_x \mathcal{L} = 0 \\ g = 0 \\ \nu \geq 0 \\ h \leq 0 \\ \nu_i h_i = 0 \end{cases} \Rightarrow \begin{cases} \nabla_x \mathcal{L} = 0 \\ g = 0 \\ \nu_i h_i = -\tau \end{cases} \tag{4}$$

This is a nonlinear system on which we can perform rootfinding. How many equations and how many unknowns are there in our case?

2. Relating our problem to CasADi's canonical rootfinder problem $g(x,p) = 0$, what is x, p and g? Create a rootfinder object by writing symbolic expressions for the dots:

```
rf = rootfinder('rf','newton',struct('x',...,'p',...,'g',...));
```

Verify that you obtain the following as print representation of `rf`:

```
rf:(x0[4],p)->(x[4]) Newton
```

3. Perform rootfinding with $\tau = 0.01$. Start out with $x_0 = \begin{bmatrix} -0.5 \\ -1.8 \end{bmatrix}, \lambda_0 = 0.1, \nu_0 = 0.1.$

A correct implementation will lead to

```
x = [-0.5364485647933107, -1.557946537599885];
```

4. Verify that the solution gets more accurate when you decrease τ.