TEMPO Spring School: Theory and Numerics for Nonlinear Model Predictive Control

Exercise 3: Gauss-Newton SQP

J. Andersson M. Diehl J. Rawlings M. Zanon

University of Freiburg, March 27, 2015

Gauss-Newton sequential quadratic programming (SQP)

In the exercises so far, we solved the NLPs with IPOPT. IPOPT is a popular open-source primal-dual interior point code employing so-called filter line-search to ensure global convergence. Other NLP solvers that can be used from CasADi include SNOPT, WORHP and KNITRO. In the following, we will write our own simple NLP solver implementing sequential quadratic programming (SQP).

Starting from a given initial guess for the primal and dual variables $(x^{(0)}, \lambda_g^{(0)})$, SQP solves the NLP by iteratively computing local convex quadratic approximations of the NLP at the current iterate $(x^{(k)}, \lambda_g^{(k)})$ and solving them by using a quadratic programming (QP) solver. For an NLP of the form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & \underline{x} \le x \le \overline{x}, \quad \underline{g} \le g(x) \le \overline{g}, \end{aligned} \tag{1}$$

these quadratic approximations take the form:

$$\begin{aligned} \underset{\Delta x}{\text{minimize}} \quad & \tfrac{1}{2}\, \Delta x^\intercal\, \nabla_x^2 \mathcal{L}(x^{(k)}, \lambda_g^{(k)}, \lambda_x^{(k)})\, \Delta x + \nabla_x f(x^{(k)})^\intercal\, \Delta x \\ \text{subject to} \quad & \underline{x} - x^{(k)} \le \Delta x \le \overline{x} - x^{(k)}, \\ & \underline{g} - g(x^{(k)}) \le \frac{\partial g}{\partial x}(x^{(k)})\, \Delta x \le \overline{g} - g(x^{(k)}), \end{aligned} \tag{2}$$

where $\mathcal{L}(x, \lambda_g, \lambda_x) = f(x) + \lambda_g^\intercal\, g(x) + \lambda_x^\intercal\, x$ is the so-called Lagrangian function. By solving this QP, we get the (primal) step $\Delta x^{(k)} := x^{(k+1)} - x^{(k)}$ as well as the Lagrange multipliers $\lambda_g^{(k+1)}$ and $\lambda_x^{(k+1)}$.

For NLPs with a least-squares objective function $f(x) = \frac{1}{2}\, \|R(x)\|_2^2$, it is often a good idea to use the so-called *Gauss-Newton* method, which uses the following approximation of the Hessian of the Lagrangian:

$$\nabla_x^2 \mathcal{L}\left(x^{(k)}, \lambda_g^{(k)}, \lambda_x^{(k)}\right) \approx \frac{\partial R}{\partial x}\left(x^{(k)}\right)^\intercal\, \frac{\partial R}{\partial x}\left(x^{(k)}\right).$$

Note that for least-squares objective functions, the gradient of the objective function is given by $\nabla_x f(x) = \frac{\partial R}{\partial x}\left(x^{(k)}\right)^\intercal\, R\left(x^{(k)}\right)$, independently of the Hessian approximation.

The Gauss-Newton method has two major advantages:

- The quadratic approximation will always be a convex optimization problem, for which efficient algorithms and software exist.
- Since the approximation of $\nabla_x^2 \mathcal{L}$ does not depend on the Lagrange multipliers explicitly, we do not need to consider it at all during the iterations and we can work only with the primal iterates $x^{(k)}$.

Tasks:

3.1 Note that the objective function of the Rosenbrock problem from Exercise 1 can be put in a least-squares form. Identify $R(x)$ such that $f(x) = \frac{1}{2}\|R(x)\|_2^2$ (on paper, no coding needed).

3.2 Implement a CasADi `SXFunction` $g : \mathbb{R}^3 \to \mathbb{R}$ that calculates the equality constraint function $g(x)$. Evaluate it numerically and confirm that you get the expected result.

3.3 Use algorithmic differentiation in CasADi to calculate the Jacobian $\frac{\partial g}{\partial x}$. Evaluate it numerically and confirm that you get the expected result.

3.4 Derive the Gauss-Newton QP subproblem (2) (on paper, no coding needed).

3.5 Implement a full-step Gauss-Newton method with 10 iterations to solve the Rosenbrock problem, using $x^{(0)} := [2.5, 3.0, 0.75]^\intercal$.

QP solvers that are available in CasADi include qpOASES (mainly for dense QPs), CPLEX and OOQP. Out of these, only qpOASES is distributed with CasADi – others must be installed separately. You can also use an NLP solver such as IPOPT to solve the QP.

CasADi uses the following formulation of QPs:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2} x^\intercal H\, x + g^\intercal\, x \\ \text{subject to} & \underline{x} \le x \le \overline{x}, \quad \underline{a} \le A\, x \le \overline{a}, \end{array} \tag{3}$$

QP solvers in CasADi are allocated for a fixed sparsity pattern of H and A:

```
qp = qpStruct(h=H.sparsity(),a=A.sparsity())
solver = QpSolver("nlp.ipopt",qp)    # Allocate solver (IPOPT)
#solver = QpSolver("qpoases",qp)     # Allocate solver (qpOASES)
solver.init()                        # Initialize the solver
```

and can then be "evaluated" to solve QPs as follows:

```
solver.setInput(H,"h")
solver.setInput(g,"g")
solver.setInput(A,"a")
solver.setInput(lbx,"lbx")
solver.setInput(ubx,"ubx")
solver.setInput(lba,"lba")
solver.setInput(uba,"uba")
solver.evaluate()
print solver.getOutput("x")
```

3.6 **Extra**: At each SQP iteration, print the constraint violation (primal infeasibility) as well as gradient of the Lagrangian (dual infeasibility). These are given by

$$\sum_{i=0}^{n_x} \left(\max(0, x_i^{(k)} - \overline{x}_i) + \max(0, \underline{x}_i - x_i^{(k)})\right) + \sum_{i=0}^{n_g} \left(\max(0, g_i(x^{(k)}) - \overline{g}_i) + \max(0, \underline{g}_i - g_i(x^{(k)}))\right)$$

and

$$\frac{\partial R}{\partial x}\left(x^{(k)}\right)^\intercal R\left(x^{(k)}\right) + \frac{\partial g}{\partial x}\left(x^{(k)}\right)^\intercal \lambda_g + \lambda_x$$

respecively.