Interior Point Method for Nonlinear Optimization

Nonlinear optimization (also called *nonlinear programming*, NLP) is present in a plethora of applications across various fields such as finance, engineering and operational research, so it is important to have the right solver for your needs. We are happy to present the latest addition to our nonlinear optimization solvers offering: a solver based on an **interior point method (IPM)** for large-scale problems. The new solver does not supersede the existing ones; rather it complements them as the underlying algorithms are fundamentally different. Some key features, along with advice on what solver to choose, are the subject of this mini-article.

Introduction to the algorithms

Here we consider large-scale nonlinear optimization problems, i.e., problems with nonlinear objective and/or nonlinear constraints which are sufficiently smooth with hundreds to hundreds of thousands of variables. The general formulation can also include linear and box constraints as stated below:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & l_g \leq g(x) \leq u_g \\ & l_B \leq Bx \leq u_B \\ & l_x \leq x \leq u_x \end{aligned}$$

There are **two main approaches** for solving NLP. It is good to understand their very distinct features. The easiest for a comparison is to look at how the inequality constraints are treated and how the solver approaches the optimal solution (the progress of the optimality measures: optimality, feasibility, complementarity).

Inequality constraints are the hard part of the optimization because of their "twofold nature". If the optimal solution satisfies strictly the inequality, i.e., the optimal point is in the interior of the constraint, the inequality constraint doesn't influence the result and could be removed from the model. On the other hand, if the inequality is satisfied as an equality (is *active* at the solution), the constraint must be present and could be treated as an equality from the very beginning.

Most of the existing solvers (such as *e04vh, e04uc, e04us*) in the NAG Library are based on the **active-set sequential quadratic programming method** (or just **SQP**). Such a solver needs to solve at each iteration a quadratic approximation of the original problem and it tries to estimate which constraints needs to be kept (are active) and which can be ignored. A practical consequence is that the algorithm partly "walks along the boundary" of the feasible region given by the constraints. The iterates are thus early on feasible w.r.t. all linear constraints (and a local linearization of the nonlinear constraints) which is preserved through the iterations. The complementarity is satisfied by default and once the active set is determined correctly and optimality is within the tolerance, the solver finishes. The number of iterations might be high but each is relatively cheap. See [1] for more details.

In contrast, an **interior point method** generates iterations that avoid the boundary defined by the inequality constraints. As the solver progresses the iterates are allowed to get closer and closer to the boundary and converge to the optimal solution which might lie on the boundary. From the practical point of view, IPM typically requires only tens of iterations. Each iteration consists of solving a large

linear system of equations taking into account all variables and constraints so each iteration is fairly expensive. All three optimality measures are reduced simultaneously.

The new solver is available as *e04st* within the *NAG Optimization Modelling Suite*, a new suite of routines which allows an easier way of setting up and solving the problems. The algorithmic details can be found in [2].

Key features of the solvers and their differences

In many cases it is difficult to predict which of the algorithms will behave better on a particular problem, however, some initial guidance can be given as follows:

IPM advantages	*SQP advantages*
• Can exploit 2nd derivatives and its structure • Efficient on unconstrained or loosely constrained problems • Efficient also for (both convex and nonconvex) quadratic problems (QP) • Better use of multi-core architecture (*SMP library only) • New interface, easier to use	• Stay feasible w.r.t. linear constraints through most of the iterations • Very efficient for highly constrained problems • Better results on pathological problems in our experience • Generally requires less function evaluations (efficient for problems with expensive function evaluations) • Requires first derivatives but can work only with function values • Can capitalize on a good initial point • Allows warm starts (good for a sequence of similar problems) • Infeasibility detection

Advice on the choice of the solver

Unless some of the specific features are required which are offered only by one algorithm, the initial decision should be based on the *availability of the derivatives* of the problem and the *number of constraints* (for example, expressed as a ratio between the numbers of variables and the sum of the number of linear and nonlinear constraints). Readiness of exact 2nd derivatives is a clear advantage for IPM so unless the number of constraints is close to the number of variables, IPM will probably work better. Similarly, if a large-scale problem has relatively only a few constraints (e.g., <40%) IPM might be more successful, especially as the problem gets bigger. On the other hand, if no derivatives are available, either our SQP or a specialized algorithm from the Library (see Derivative Free Optimization, DFO) needs to be used. With more and more constraints SQP might be faster. For problems which don't fall in either of the categories, it is not easy to anticipate which solver will work better and some experimentations might be required.

References

[1] Gill P E, Murray W and Saunders M A (2005) SNOPT: An SQP Algorithm for Large-scale Constrained Optimization *SIAM Review* **47(1)** 99-131

[2] Wächter A and Biegler L T (2006) On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming *Mathematical Programming* **106(1)** 25-57