WIKIPEDIA

Sequential quadratic programming

Sequential quadratic programming (**SQP**) is an iterative method for constrained nonlinear optimization which may be considered a quasi-Newton method. SQP methods are used on mathematical problems for which the objective function and the constraints are twice continuously differentiable, but not necessarily convex.

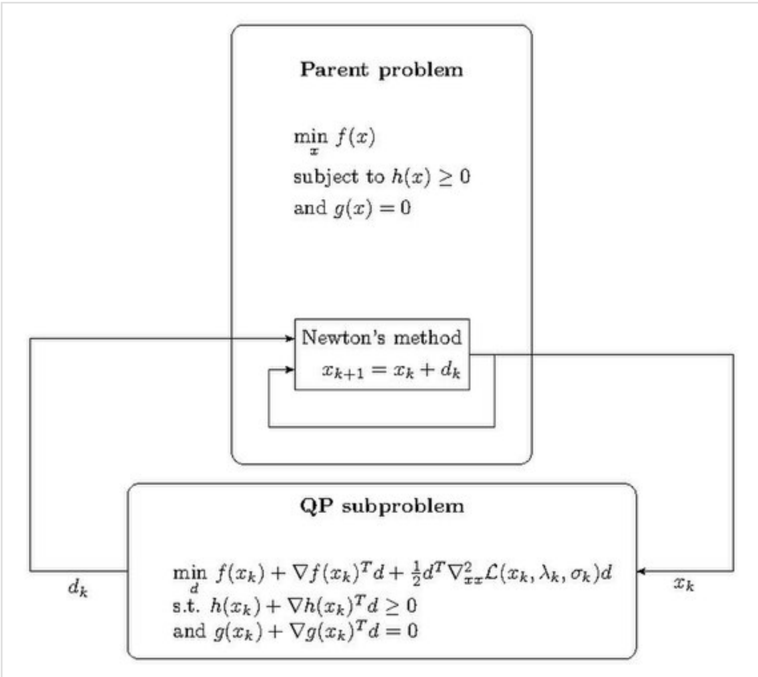
SQP methods solve a sequence of optimization subproblems, each of which optimizes a quadratic model of the objective subject to a linearization of the constraints. If the problem is unconstrained, then the method reduces to Newton's method for finding a point where the gradient of the objective vanishes. If the problem has only equality constraints, then the method is equivalent to applying Newton's method to the first-order optimality conditions, or Karush–Kuhn–Tucker conditions, of the problem.

Algorithm basics

Consider a nonlinear programming problem of the form:

$$\begin{array}{rl} \min\limits_{x} & f(x) \\ \text{subject to} & h(x) \geq 0 \\ & g(x) = 0. \end{array}$$

The Lagrangian for this problem is[1]



Overview schematic illustrating the basic SQP algorithm

$$\mathcal{L}(x, \lambda, \sigma) = f(x) - \lambda h(x) - \sigma g(x),$$

where λ and σ are Lagrange multipliers.

The standard Newton's Method searches for the solution $\nabla \mathcal{L}(x, \lambda, \sigma) = 0$ by iterating the following equation, where ∇^2_{xx} denotes the Hessian matrix:

$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \\ \sigma_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ \lambda_k \\ \sigma_k \end{bmatrix} - \underbrace{\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & \nabla h & \nabla g \\ \nabla h^T & 0 & 0 \\ \nabla g^T & 0 & 0 \end{bmatrix}^{-1}}_{\nabla^2 \mathcal{L}} \underbrace{\begin{bmatrix} \nabla f + \lambda_k \nabla h + \sigma_k \nabla g \\ h \\ g \end{bmatrix}}_{\nabla \mathcal{L}}.$$

However, because the matrix $\nabla^2 \mathcal{L}$ is generally singular (and therefore non-invertible), the Newton step $d_k = \left(\nabla^2_{xx}\mathcal{L}\right)^{-1} \nabla \mathcal{L}$ cannot be calculated directly. Instead the basic sequential quadratic programming algorithm defines an appropriate search direction d_k at an iterate $(x_k, \lambda_k, \sigma_k)$, as a solution to the quadratic programming subproblem

$$\begin{array}{rl} \min\limits_{d} & f(x_k) + \nabla f(x_k)^T d + \tfrac{1}{2} d^T \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k, \sigma_k) d \\ \mathrm{s.t.} & h(x_k) + \nabla h(x_k)^T d \geq 0 \\ & g(x_k) + \nabla g(x_k)^T d = 0. \end{array}$$

where the quadratic form is formed with the Hessian of the Lagrangian. Note that the term $f(x_k)$ in the expression above may be left out for the minimization problem, since it is constant under the $\min\limits_{d}$ operator.

Together, the SQP algorithm starts by first choosing the initial iterate $(x_0, \lambda_0, \sigma_0)$, then calculating $\nabla^2 \mathcal{L}(x_0, \lambda_0, \sigma_0)$ and $\nabla \mathcal{L}(x_0, \lambda_0, \sigma_0)$. Then the QP subproblem is built and solved to find the Newton step direction d_0 which is used to update the parent problem iterate using $\left[x_{k+1}, \lambda_{k+1}, \sigma_{k+1}\right]^T = \left[x_k, \lambda_k, \sigma_k\right]^T + d_k$. This process is repeated for $k = 0, 1, 2, \ldots$ until the parent problem satisfies a convergence test.

Practical implementations

Practical implementations of the SQP algorithm are significantly more complex than its basic version above. To adapt SQP for real-world applications, the following challenges must be addressed:

- The possibility of an infeasible QP subproblem.
- QP subproblem yielding an bad step: one that either fails to reduce the target or increases constraints violation.
- Breakdown of iterations due to significant deviation of the target/constraints from their quadratic/linear models.

To overcome these challenges, various strategies are typically employed:

- Use of merit functions, which assess progress towards a constrained solution, or filter methods.
- Trust region or line search methods to manage deviations between the quadratic model and the actual target.
- Special feasibility restoration phases to handle infeasible subproblems, or the use of L1-penalized subproblems to gradually decrease infeasibility

These strategies can be combined in numerous ways, resulting in a diverse range of SQP methods.

Alternative approaches

- Sequential linear programming
- Sequential linear-quadratic programming
- Augmented Lagrangian method

Implementations

SQP methods have been implemented in well known numerical environments such as MATLAB and GNU Octave. There also exist numerous software libraries, including open source:

- SciPy (de facto standard for scientific Python) has scipy.optimize.minimize(method='SLSQP') solver.
- NLopt (https://nlopt.readthedocs.io/en/latest/) (C/C++ implementation, with numerous interfaces including Julia, Python, R, MATLAB/Octave), implemented by Dieter Kraft as part of a package for optimal control, and modified by S. G. Johnson.[2][3]
- ALGLIB SQP solver (C++, C#, Java, Python API)

and commercial

- LabVIEW
- KNITRO[4] (C, C++, C#, Java, Python, Julia, Fortran)
- NPSOL (Fortran)
- SNOPT (Fortran)
- NLPQL (Fortran)
- MATLAB
- SuanShu (http://www.numericalmethod.com/javadoc/suanshu/com/numericalmethod/suanshu/optimization/multivariate/constrained/convex/sdp/pathfollowing/package-frame.html) (Java)

See also

- Newton's method
- Secant method
- Model Predictive Control

Notes

1. Jorge Nocedal and Stephen J. Wright (2006). *Numerical Optimization* (http://www.ece.northwestern.edu/~nocedal/book/num-opt.html). Springer. ISBN 978-0-387-30303-1.
2. Kraft, Dieter (Sep 1994). "Algorithm 733: TOMP–Fortran modules for optimal control calculations" (https://github.com/scipy/scipy/blob/master/scipy/optimize/slsqp/slsqp_optmz.f). *ACM Transactions on Mathematical Software*. **20** (3): 262–281. CiteSeerX 10.1.1.512.2567 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.512.2567). doi:10.1145/192115.192124 (https://doi.org/10.1145%2F192115.192124).

S2CID 16077051 (https://api.semanticscholar.org/CorpusID:16077051). Retrieved 1 February 2019.

3. "NLopt Algorithms: SLSQP" (https://nlopt.readthedocs.io/en/latest/NLopt_Algorithms/#slsqp). *Read the Docs*. July 1988. Retrieved 1 February 2019.
4. KNITRO User Guide: Algorithms (https://www.artelys.com/tools/knitro_doc/2_userGuide/algorithms.html)

References

- Bonnans, J. Frédéric; Gilbert, J. Charles; Lemaréchal, Claude; Sagastizábal, Claudia A. (2006). *Numerical optimization: Theoretical and practical aspects* (https://www.springer.com/mathematics/applications/book/978-3-540-35445-1). Universitext (Second revised ed. of translation of 1997 French ed.). Berlin: Springer-Verlag. pp. xiv+490. doi:10.1007/978-3-540-35447-5 (https://doi.org/10.1007%2F978-3-540-35447-5). ISBN 978-3-540-35445-1. MR 2265882 (https://mathscinet.ams.org/mathscinet-getitem?mr=2265882).
- Jorge Nocedal and Stephen J. Wright (2006). *Numerical Optimization* (http://www.ece.northwestern.edu/~nocedal/book/num-opt.html). Springer. ISBN 978-0-387-30303-1.

External links

- Sequential Quadratic Programming at NEOS guide (https://neos-guide.org/guide/algorithms/sqp/)