

Problem solving by nonlinear mathematical programming

Dirk Munro (Hamburg, Germany)

March 6, 2022

*This note is intended as a simple, general introduction to the subject of sequential approximation optimization, a general nonlinear mathematical programming paradigm. In particular, the subject is described from the solution-method's point of view, and an 'optimization problem' is taken to be a well-defined representation of a 'problem'—in flirtation with the fundamental Deutschean sense of it [1]. No novelty beyond the interpretation, description (and notation) is introduced. The list of citations would not be reasonably complete without those referenced in the **Background**, in closure of the note.*

1 Introduction

Assume we have a problem \mathcal{P} and an array of scalar decision variables \mathbf{x} . Each decision variable is assumed to be continuous*, and packed into an array of length n ; *i.e.* $\mathbf{x} \in \mathbb{R}^n$. A particular point in the 'decision space' will henceforth be represented by a particular collection of values in \mathbf{x} . What is meant herein by 'a problem \mathcal{P} '? We will assume that \mathcal{P} is programmatic (numeric) entity, which may be evaluated with an array of particular decision (input) variable values \mathbf{x} , $\mathcal{P}[\mathbf{x}]$ [†], and, in so doing, the problem returns the following information

$$\left\{ \begin{array}{l} \mathbf{c} = (c_0, c_1, \dots, c_m) \\ \partial \mathbf{c} = (\partial c_0, \partial c_1, \dots, \partial c_m) \\ \underline{\mathbf{x}} = (\underline{\mathbf{x}}, \bar{\mathbf{x}}) \end{array} \right\} = \mathcal{P}[\mathbf{x}]. \quad (1)$$

That is

- (i) an array \mathbf{c} of (scalar function) values, of length m . We will adopt the convention that the first entry c_0 is a scalar-valued objective (cost) to be *minimised*, and the following $m - 1$ entries denote constraint function values, with values of $c_j \leq 0$ indicating *feasibility*; *i.e.* adherence to constraint j , and violation otherwise;
- (ii) assuming each cost-and-constraint function is (at least once) continuously differentiable, the first-order derivatives of each, to each decision variable, $\partial \mathbf{c}$, with

$$\partial \mathbf{c}_j = \left(\frac{\partial c_j}{\partial x_1}, \frac{\partial c_j}{\partial x_2}, \dots, \frac{\partial c_j}{\partial x_n} \right) \quad \text{for } j = 0, 1, \dots, m; \quad (2)$$

- (iii) the lower $\underline{\mathbf{x}}$ and upper $\bar{\mathbf{x}}$ bounds of the decision space, respectively, defining the entire decision space $\bar{\mathbf{x}}$.

*Note that discrete-valued variables may always be represented in this form.

[†]Through-out square brackets $[\cdot]$ will denote an operation of 'to evaluate at'; and it may be used and dropped freely as is relevant to the description of a particular entity, in a particular context.

In other words, beyond the information we can collect from the quantities returned for particular evaluations $\{\mathbf{c}, \partial\mathbf{c}, \underline{\mathbf{x}}\} = \mathcal{P}[\mathbf{x}]$ we have no further information of the problem \mathcal{P} , except for what can be deduced from the upstream assumptions (and knowledge of the nature of the problem). Moreover, in general, a single evaluation of problem $\mathcal{P}[\mathbf{x}]$ is considered computationally expensive—or inconvenient—insofar as it is non-trivially costly to evaluate. That is for example simulation-based, with multiple, potentially large-scale finite-element analyses, or computationally cheap (*e.g.* reduced-order) analyses which rely on calls to licensed applications, or problem evaluations which require significant network communications.

How to find a candidate[‡] optimum solution \mathbf{x}^* of problem \mathcal{P} ? Well, we can start by sampling the decision space $\underline{\mathbf{x}}$, to find an \mathbf{x} with a reasonable cost $c_0[\mathbf{x}]$, while all constraints are feasible $c_j[\mathbf{x}] \leq 0, \forall j > 0$. Then, we can ask the question: can this sample decision \mathbf{x} be improved if we adjust the evaluation in the decision space by a particular amount $\mathbf{x} + \Delta\mathbf{x}$? If there is a $\Delta\mathbf{x}$ for which $c_0[\mathbf{x}] > c_0[\mathbf{x} + \Delta\mathbf{x}]$, while all constraints remain feasible $c_j[\mathbf{x} + \Delta\mathbf{x}] \leq 0, \forall j > 0$, then the sample \mathbf{x} we started with was indeed not optimal[§], and we should adjust the decision to $\mathbf{x} + \Delta\mathbf{x}$.

How to determine what $\Delta\mathbf{x}$ should be? Keep in mind, we have assumed we are working with an expensive-to-evaluate problem \mathcal{P} , which only returns the information $\{\mathbf{c}, \partial\mathbf{c}, \underline{\mathbf{x}}\} = \mathcal{P}[\mathbf{x}]$ upon evaluation at \mathbf{x} . In practical terms, we will take this to mean that we want to compute a good change in the decision space $\Delta\mathbf{x}$, without re-evaluating problem \mathcal{P} .

Let us, in hope[¶], assume that the actual cost-and-constraint functions \mathbf{c} are (near) linear over the entire decision space $\underline{\mathbf{x}}$. That is, an analytic linear approximation of the cost-and-constraint functions \mathbf{c} , in terms of $\Delta\mathbf{x}$, is constructed

$$\mathbf{v}[\Delta\mathbf{x}] = \mathbf{c} + \partial\mathbf{c} \cdot \Delta\mathbf{x} , \quad (3)$$

with first derivatives following accordingly

$$\partial\mathbf{v}[\Delta\mathbf{x}] = \partial\mathbf{c} . \quad (4)$$

In general, the first-order approximation of the cost-and-constraint functions \mathbf{v} is only equal^{||} to the actual cost-and-constraint functions \mathbf{c} at \mathbf{x} —i.e., with $\Delta\mathbf{x} = \mathbf{0}$ —and representative in a infinitely small region around \mathbf{x} . In the light of this, we introduce a lower $\Delta\underline{\mathbf{x}}$ and upper $\Delta\bar{\mathbf{x}}$ bound on the change $\Delta\mathbf{x}$ we allow ourselves in the decision space $\underline{\mathbf{x}}$. That is, we have arrived at a subproblem

$$\left\{ \begin{array}{l} \mathbf{v} = (v_0, v_1, \dots, v_m) \\ \partial\mathbf{v} = (\partial v_0, \partial v_1, \dots, \partial v_m) \\ \Delta\underline{\mathbf{x}} = (\Delta\underline{\mathbf{x}}, \Delta\bar{\mathbf{x}}) \end{array} \right\} = \mathcal{S}[\Delta\mathbf{x}] , \quad (5)$$

[‡]In general the problem is assumed to be multi-modal. That is, it may have many local optima, in addition to a global optimum. The topic is discussed, and a solution method is proposed, in Reference [2].

[§]Not even locally.

[¶]or desperation.

^{||}Equal in zero- and first-order information.

which is, importantly, unlike problem \mathcal{P} , very cheap to evaluate for different values of $\Delta\mathbf{x}$, because an analytic approximation \mathbf{v} has been constructed. In general we say that a solution to problem \mathcal{S} —finding that $\Delta\mathbf{x}$ in $\Delta\bar{\mathbf{x}}$ for which v_0 is a minimum, while all $v_j \leq 0, \forall j > 0$ —is computable in polynomial time. Moreover, in practice, polynomial time solution methods are readily available.

We may thus, upon solving subproblem \mathcal{S} , update the decision to $\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$; repeat the evaluation of problem \mathcal{P} at the new values of the decision variables $\{\mathbf{c}, \partial\mathbf{c}, \bar{\mathbf{x}}\} = \mathcal{P}[\mathbf{x}]$, and repeat the construction and solution of subproblem \mathcal{S} . The process may be repeated until there is no change in the decision $\Delta\mathbf{x}$ which improves the solution, and we may reasonably deduce that we have arrived at a candidate solution $\mathbf{x} \rightarrow \mathbf{x}^*$ of problem \mathcal{P} .

In general, however, linear approximations \mathbf{v} of the cost-and-constraint functions \mathbf{c} may result in changes $\Delta\mathbf{x}$ in the decision space which violates the constraints $c_j[\mathbf{x} + \Delta\mathbf{x}] > 0 \forall j > 0$, increases the cost function $c_0[\mathbf{x} + \Delta\mathbf{x}]$, very restrictive allowable decision changes $\Delta\bar{\mathbf{x}}$ resulting in excessive, expensive evaluations of problem $\mathcal{P}[\mathbf{x}]$, and hence complete failure of the procedure to converge to a candidate solution \mathbf{x}^* of problem \mathcal{P} .

What can we do?

2 Decision space transformations

Let us assume that we notice, by observation, that the cost-and-constraint functions \mathbf{c} have a particular proportional relationship to decisions \mathbf{x} in the decision space $\bar{\mathbf{x}}$. For example, we might notice** that the cost-and-constraint functions \mathbf{c} have a reciprocal-like relation to our decision variables, *i.e.* $\mathbf{c} \sim 1/\mathbf{x}$. Can we exploit this to improve the change $\Delta\mathbf{x}$ in the decision space we attempt to make? Yes, we can imagine applying an analytic transformation (mapping) to the decision space $\mathbf{y} = \mathbf{y}[\mathbf{x}]$, and formulating the approximate cost-and-constraint functions \mathbf{v} in terms of it

$$\mathbf{v} = \mathbf{c} + \partial_{\mathbf{y}}\mathbf{c} \cdot \Delta\mathbf{y} . \quad (6)$$

That is, we hope that the transformation of the decision space from $\mathbf{x} \rightarrow \mathbf{y}$ has worked to ‘linearise’ the cost-and-constraint functions \mathbf{c} . Rewriting Eq. (6) in terms of the first-order information supplied by problem \mathcal{P} , we retrieve

$$\mathbf{v} = \mathbf{c} + \partial\mathbf{c} \cdot \partial_{\mathbf{x}}^{-1}\mathbf{y} \cdot \Delta\mathbf{y} , \quad (7)$$

with the derivative of the analytic mapping $\mathbf{y} = \mathbf{y}[\mathbf{x}]$ easy to compute

$$\partial_{\mathbf{x}}^{-1}\mathbf{y} = 1 / \frac{\partial\mathbf{y}}{\partial\mathbf{x}} . \quad (8)$$

Note that $\partial_{\mathbf{x}}^{-1}\mathbf{y}$ is an $n \times n$ square matrix (although in practice, typically, only diagonal terms are utilised).

**and/or know, based on knowledge on the nature of the problem \mathcal{P} .

How does this help us? Consider again the example of the reciprocal transformation $\mathbf{y} = 1/\mathbf{x}$. In this case

$$\partial_{\mathbf{x}}^{-1}\mathbf{y} = -\mathbf{x}_{\mathbf{I}}^2, \quad (9)$$

with \mathbf{I} indicating that the array is cast along the diagonal of the correspondingly sized square identity matrix. Hence

$$\mathbf{v} = \mathbf{c} + \partial\mathbf{c} \cdot (-\mathbf{x}_{\mathbf{I}}^2) \cdot \Delta\mathbf{y}, \quad (10)$$

and the approximate cost-and-constraint functions \mathbf{v} may be re-written in terms of the original decision space \mathbf{x} and the change $\Delta\mathbf{x}$, which yields

$$\mathbf{v} = \mathbf{c} + \partial\mathbf{c} \cdot (-\mathbf{x}_{\mathbf{I}}^2) \cdot \left(\frac{1}{\mathbf{x} + \Delta\mathbf{x}} - \frac{1}{\mathbf{x}} \right), \quad (11)$$

and, is the same as

$$\mathbf{v} = \mathbf{c} + \partial\mathbf{c} \cdot \left(\frac{\mathbf{x}}{\mathbf{x} + \Delta\mathbf{x}} \right)_{\mathbf{I}} \cdot \Delta\mathbf{x}. \quad (12)$$

Notice how the term in brackets (\dots) has introduced an analytically available nonlinearity (or ‘curvature’) in the approximate cost-and-constraint functions \mathbf{v} , with respect to $\Delta\mathbf{x}$. Keep in mind, the evaluation of problem $\mathcal{P}[\mathbf{x}]$ is constant while we evaluate (and solve) subproblem \mathcal{S} —which remains computationally cheap—to compute the change in the decision space $\Delta\mathbf{x}$.

Similarly, for example, the decision space transformation may be generalised to the form $\mathbf{y} = \mathbf{x}_{\mathbf{I}}^p$, with p any real number. In this case

$$\partial_{\mathbf{x}}^{-1}\mathbf{y} = \frac{1}{p}\mathbf{x}_{\mathbf{I}}^{1-p}, \quad (13)$$

and hence

$$\mathbf{v}[\Delta\mathbf{x}] = \mathbf{c} + \partial\mathbf{c} \cdot \frac{1}{p}\mathbf{x}_{\mathbf{I}}^{1-p} \cdot ((\mathbf{x} + \Delta\mathbf{x})^p - \mathbf{x}^p), \quad (14)$$

which is easily evaluated for different values of $\Delta\mathbf{x}$, with any manner of estimate for p . The first derivatives of the approximate cost-and-constraint functions \mathbf{v} follow readily

$$\partial\mathbf{v}[\Delta\mathbf{x}] = \partial\mathbf{c} \cdot (\mathbf{x}_{\mathbf{I}}^{1-p}) \cdot (\mathbf{x} + \Delta\mathbf{x})^{p-1}, \quad (15)$$

with the first-order information of problem \mathcal{P} at the current evaluation point \mathbf{x} , $\partial\mathbf{c}$, maintained

$$\partial\mathbf{v}[\mathbf{0}] = \partial\mathbf{c}. \quad (16)$$

3 Taylor series expansions

The analytic nonlinearity introduced by the decision space transformation is conveniently unpacked by Taylor series expansion ν , of the transformed cost-and-constraints \mathbf{v} , in terms of $\Delta\mathbf{x}$ (away from the current decision point \mathbf{x} , where $\Delta\mathbf{x} = \mathbf{0}$)

$$\nu[\Delta\mathbf{x}] = \mathbf{v}[\mathbf{0}] + \frac{1}{1!}\partial\mathbf{v}[\mathbf{0}] \cdot \Delta\mathbf{x} + \frac{1}{2!}\partial^2\mathbf{v}[\mathbf{0}] \cdot \Delta\mathbf{x}^2 + \frac{1}{3!}\partial^3\mathbf{v}[\mathbf{0}] \cdot \Delta\mathbf{x}^3 + \dots \quad (17)$$

For sake of simplicity, let us truncate after the first nonlinear (2nd) term

$$\boldsymbol{\nu}[\Delta \mathbf{x}] = \mathbf{v}[\mathbf{0}] + \partial \mathbf{v}[\mathbf{0}] \cdot \Delta \mathbf{x} + \frac{1}{2} \partial^2 \mathbf{v}[\mathbf{0}] \cdot \Delta \mathbf{x}^2, \quad (18)$$

noting that the second-derivative of the transformed cost-and-constraints \mathbf{v} is readily available

$$\partial^2 \mathbf{v}[\Delta \mathbf{x}] = \partial \mathbf{c} \cdot (p-1) \mathbf{x}_{\mathbf{I}}^{1-p} \cdot (\mathbf{x} + \Delta \mathbf{x})^{p-2}, \quad (19)$$

and, at the current decision point $\Delta \mathbf{x} = \mathbf{0}$, it reduces to

$$\partial^2 \mathbf{v}[\mathbf{0}] = \partial \mathbf{c} \cdot \left(\frac{p-1}{\mathbf{x}} \right)_{\mathbf{I}}. \quad (20)$$

This notion of ‘approximated approximations’ (approximation of decision space mappings) was first introduced, in this manner, by Groenwold *et al.* [3]. In significant practical terms, the 2nd-order Taylor expansion allows for a completely flexible and general interface with common polynomial time (subproblem) solvers, which are typically fed with the first and second-order information of subproblem $\mathcal{S}^{\dagger\dagger}$.

We can therefore redefine subproblem \mathcal{S} as ...

with \mathbf{v} the approximations (footnote; open to interpretation whether this is defined by subproblem \mathcal{S} , or indeed, if this is estimated from the outside).

Quadratically constrained quadratic program. Historically, dual solution methods are available, given particular kinds of approximations / decision space transformations. Cone program solution methods.

4 Quadratic programs and the necessary conditions

Introduce as simplification to arrive at common

[4]

Solution methods for the form

in this way, the notion of decision space transformations become obsolete to some extent, in practical terms. Although the notion of it may be used to estimate the second order information...

in rough drafting If we use only diagonal information. Ensure it is positive definite. and formulate the hessian of the lagrangian ...

Then we arrive at a convenient computational ‘kernel’: the quadratic program (QP)^{‡‡}. In simple terms, polynomial time solution methods for QPs are often the cheapest, fastest,

^{††}Even if linear programming is applied to solve the subproblem \mathcal{S} , the (variable) first-order information $\partial \boldsymbol{\nu}[\Delta \mathbf{x}] = \partial \mathbf{v}[\mathbf{0}] + \partial^2 \mathbf{v}[\mathbf{0}] \cdot \Delta \mathbf{x}$ is supplied, iteratively.

^{‡‡}Conceptually, the step-wise solving of QP problems in general, non-linear programming, can be seen as equivalent to the way in which a linear system-solve is the ‘kernel’ in nonlinear structural analysis. That is the ‘kernel’ of the Newton-Raphson method.

simplest to implement, interface, and most readily available (to an average computational engineer), and ... for subproblem \mathcal{S} , given some straight-forward conditions. See for example the recently released implementation by Boyd and collaborators [4].

TO BE COMPLETED

Eventually we arrive rather naturally at the necessary conditions: How far can we go, before we have to evaluate the problem again..? Keeping in mind, that we can not afford to evaluate the problem an excessive number of times... etc. etc. Move limits and estimates of nonlinear information (surrogate functions/problems, second order estimates).... upon convergence / when we can stop, we get something for free: the necessary conditions are satisfied / we have arrived at the necessary conditions in a rather practical way. Done.

Background

The introduction to the ‘Elements of Structural Optimization’ by Haftka, Gürdal and Kamat [5]. The review of sensitivity analysis in structural optimization by van Keulen, Haftka and Kim [6]. More to follow.

References

- [1] D. Deutsch. *The Fabric of Reality*.
- [2] An interpretation of Bayesian global optimization as proposed by Snyman and Fatti. https://github.com/dirkmunro89/SOAPs/blob/main/bayopt/bay_20220104.pdf. Accessed: 2022-02-27.
- [3] A.A. Groenwold, L.F.P. Etman, and D.W. Wood. Approximated approximations for SAO. *Structural and Multidisciplinary Optimization*, 41(1):39–56, 2010.
- [4] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.
- [5] R.T. Haftka, Z. Gürdal, and M.P. Kamat. *Elements of structural optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.
- [6] F. Van Keulen, R.T. Haftka, and N.H. Kim. Review of options for structural design sensitivity analysis. Part 1: Linear systems. *Computer methods in applied mechanics and engineering*, 194(30):3213–3243, 2005.