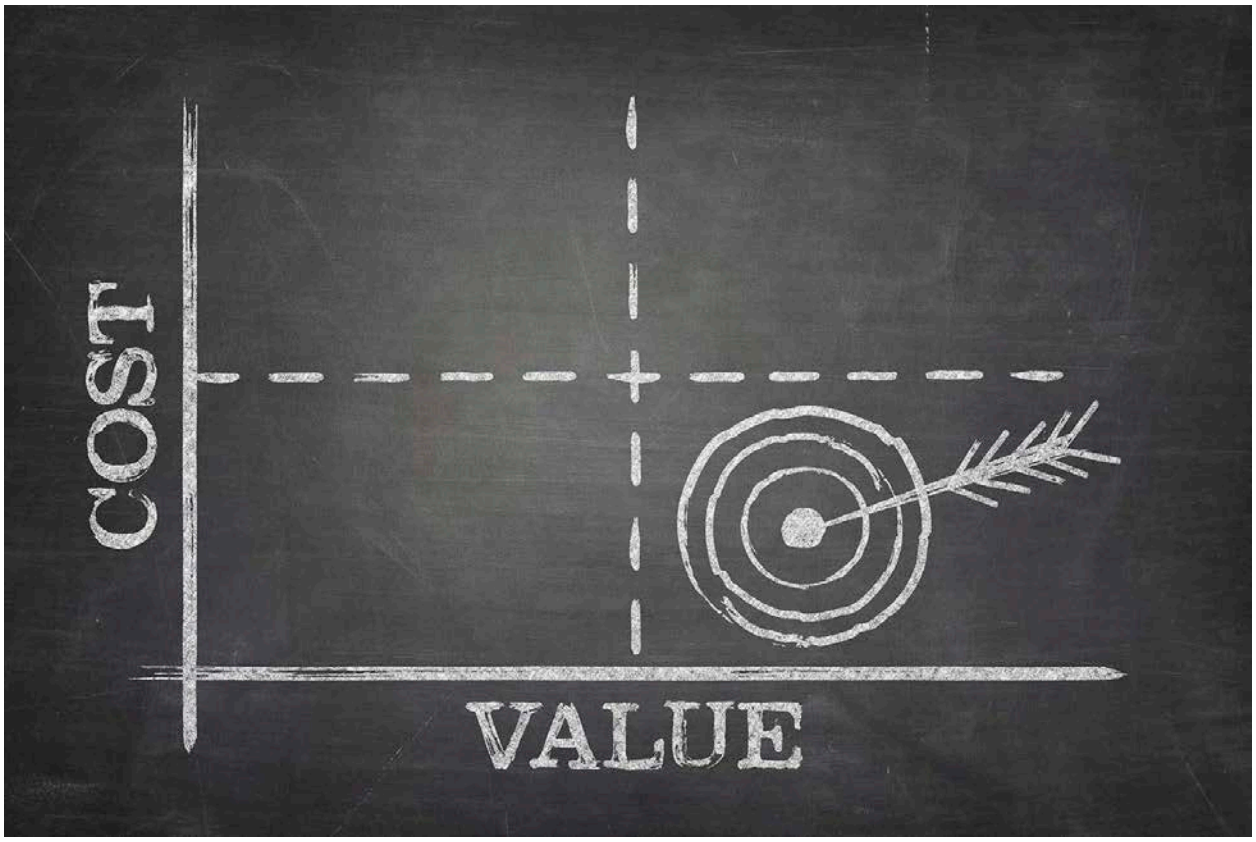

White Paper

OptiRamp® Real-Time Optimization

Optimization Methodology for Real-Time Control

Vadim Shapiro
Dmitriy Khots, Ph.D.

Table of Contents

Introduction

The Real-Time Optimization (RTO) Submodule refers to a set of models and algorithms that continually analyze and adjust operating conditions of a process with the purpose of maximizing its economic efficiency. The key to successful optimization is the presence of robust process models. The *OptiRamp* Model Construction Submodule provides an array of such models that are custom-built across all process components. These models describe dependencies between process attributes and are represented by mathematical functions that can be used as optimization criteria as well as constraints.

RTO is dynamic in nature—optimization occurs continuously and optimal operating mode settings can be found "online." The system is scalable; however, given very large-scale problems (tens of thousands of attributes), it does allow for "off-line" optimization, where optimal solutions are pre-computed during batch processing.

Optimal solutions are found numerically by using a portfolio of standard optimization techniques. An overview of this portfolio is provided.

The current RTO system also allows for several uncontrolled variables, which consist of inlet gas flow and ambient conditions in the following discussion.

RTO Capabilities

The *OptiRamp* Model Construction Submodule creates both steady-state and dynamic models that describe the interdependencies between various process variables. Steady-state models describe relationships between process variables, irrespective of time. Dynamic models can predict future behavior of a process variable based on today's independent variable values and according to a pre-defined horizon.

The RTO Submodule uses these models, such as performance curves, in conjunction with power curve functions to run optimization tasks and meet either performance or economic objectives.

RTO uses both the steady-state dependencies and dynamic information produced by the *OptiRamp* Model Construction Submodule to predict how the process will respond to changes in each independent variable. It is then able to calculate future moves that will maintain the operation at specified targets.

In particular, at any point of time, RTO uses current process variable values and predicted future values to find the optimal path of the control actions. The algorithm then continuously recalculates itself to ensure that only optimal solutions are executed.

The RTO Submodule is used to increase production and compute the operating mode in accordance to the selected optimization criteria. The objective function (optimization criteria) can be chosen to do the following:

- Maximize production
- Minimize energy cost
- Minimize fuel gas flow
- Maximize profit

Whenever uncontrolled variables appear in the models produced by the *OptiRamp* Model Construction Submodule, RTO ensures that optimal values for the optimization criteria are found for every value (or an acceptable range of values) of the uncontrolled variables.

The optimization process takes place at the station/plant levels.

Optimization Methods

The RTO Submodule is equipped with a variety of deterministic and stochastic optimization techniques that may be manually or automatically selected based on the objective function and constraints.

The objective function, as well as most of constraints, are provided by the *OptiRamp* Model Construction Submodule. Most of the optimization techniques employed are used to find the global optimal point, where the term “global” is considered to be over the domain defined by the constraints.

The example in Figure 1 shows differences in global minimums because of constraints. This example is also a good demonstration of common optimization pitfalls (local minimums) that RTO avoids.

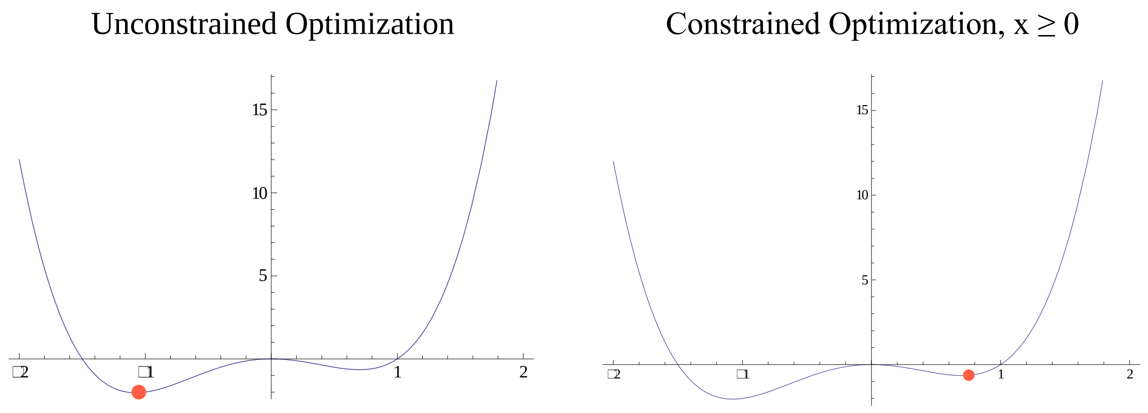

Figure 1. Global minimums with constraints

The main advantage of the objective functions produced by the *OptiRamp* Model Construction Submodule is the fact that they can be described by polynomials of the form $y = f(x_1 \ldots x_n)$, where $y, x_1, \ldots, x_n$ are process variables. This form makes the objective functions smooth (continuously differentiable); thus, first- and second-order conditions can be used to determine the optimal points.

In particular, the optimization process goal is to find vectors $(x_1^{opt} \ldots\ldots x_n^{opt})$ such that $\frac{df}{dx_j}(x_1^{opt} \ldots\ldots x_n^{opt}) = 0$ for $j = 1 \ldots\ldots n$ (vectors satisfying the first-order conditions, i.e., critical points). The critical points can be identified as the minimum or maximum depending on the determinant of the matrix of second derivatives of $y = f(x_1 \ldots x_n)$, called:

Hessian in an unconstrained case

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \ldots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \ldots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \ldots & \ldots & \ldots & \ldots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \ldots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

bordered Hessian in an constrained case

$$H(f,g) = \begin{pmatrix} 0 & \frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2} & \ldots & \frac{\partial g}{\partial x_n} \\ \frac{\partial g}{\partial x_1} & \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \ldots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial g}{\partial x_2} & \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \ldots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \frac{\partial g}{\partial x_n} & \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \ldots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

where g is a set of m constraints. This form fits well with RTO because constraints produced by the *OptiRamp* Model Construction Submodule are also polynomials.

In large-dimensional problems, analytical computations of these concepts become rather cumbersome; thus, the RTO uses numerical methods.

The RTO Submodule utilizes various computational techniques to find global optima. The technique choice depends on the optimization problem.

Integer Programming (Branch and Bound)

At the station/plant level, the load sharing problem is solved by either unit shutdown or start-up. The criteria for optimal load sharing is based on comparing the objective function value at the current point in time to its value computed using the static model after the planned change in loads. The optimization process takes place at the overall process, operating unit, and station/plant levels.

Certain optimization tasks with binary control variables (unit start-up/shutdown) can be solved using an integer programming method called branch and bound.

The idea is to find the minimum of a function $y = f(x_1 ... x_n)$, e.g., cost, where x is located over a range of possible solutions (feasible region). The first step in the branch-and-bound technique is branching—covering the feasible region with two or more smaller subsets so that the global minimum of $f(x_1 ... x_n)$ over the entire feasible region is the minimum of $f(x_1 ... x_n)$ over each subset. The next step is bounding, where upper and lower bounds for $\min f(x_1 ... x_n)$ are computed. The primary concept of the branch-and-bound algorithm is that if the lower bound over a subset is greater than the upper bound of another subset, then the first subset can be removed from analysis. The algorithm continues recursively eliminating subsets until the entire feasible region consists of a single element.

Linear Programming (Simplex Method)

When the *OptiRamp* Model Construction Submodule produces linear objective functions as well as constraints, the system uses a linear programming technique developed by George Dantzig—the simplex method.

A linear optimization problem can be typically stated as shown in equation (1):

$$\text{maximize } c^T x \text{ subject to } Ax \le b,\ x \ge 0, \tag{1}$$

where $x = (x_1, ..., x_n)$ are the *n* process variables; $c = (c_1, ..., c_n)$ is a vector of linear coefficients, i.e., the linear form that is optimized; *A* is a rectangular $p \times n$ matrix; and $b = (b_1, ..., b_n)$ are the linear constraints.

The simplex method uses the augmented form of the linear programming problem in equation (1), where inequalities are replaced by equalities. The problem is then transformed to equation (2)

$$\begin{bmatrix} 1 & -c^T & 0 \\ 0 & A & I \end{bmatrix} \begin{bmatrix} Z \\ x \\ x_s \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix} \tag{2}$$

$$x, x_s \geq 0,$$

where $x_s = \left(x_{s,1}, \ldots, x_{s,p}\right)$ is the vector of slack variables used in the augmentation process, i.e., the distances between the point and linear constraints *A*. The objective function obtains a value *q* on a hyperplane $c^T x = q$, and the maximum *q* occurs when the last hyperplane intersects the feasible region at the vertex of the simplex (a polytope with *N+1* vertices and *N* edges).

The following methods are used for nonlinear optimization problems.

Quasi-Newton Method

The quasi-Newton method is based on Newton's well-known method of optimization. The goal is to locally approximate the objective function by a second-order Taylor series. Given the function $y = f(x)$, where x is an *n*-dimensional vector, the Taylor series (for each iteration) is defined by equation (3):

$$f(x_k + \Delta x) \approx f\left(x_k\right) + \nabla f\left(x_k\right)\Delta x + \frac{1}{2}\Delta x^T B \Delta x, \tag{3}$$

where ∇f is the gradient and *B* is an approximation of the Hessian. The gradient of the approximation is

$$\nabla f(x_k + \Delta x) \approx \nabla f\left(x_k\right) + B\Delta x$$

The first-order condition then provides Newton's method step:

$$\Delta x = -B^{-1}\nabla f\left(x_k\right)$$

The next step of the algorithm determines *B*, such that

$$\nabla f\left(x_k + \Delta x\right) = \nabla f\left(x_k\right) + B\Delta x$$

Note that in a one-dimensional setting, this algorithm is equivalent to the secant method; however, in a multi-dimensional environment, various calculation methods are used to determine *B*. The system uses the symmetric rank-one (SR1) update method shown in equation (4):

$$B_{k+1} = B_k + \frac{\left(y_k - B_k \Delta x_k\right)\left(y_k - B_k \Delta x_k\right)^T}{\left(y_k - B_k \Delta x_k\right)^T \Delta x_k}, \qquad (4)$$

where $\Delta x_k = -\alpha_k B_k^{-1} \nabla\left(x_k\right)$, α satisfies Wolfe conditions, $x_{k+1} = x_k + \Delta x_k$, and $y_k = \nabla f\left(x_{k+1}\right) - \nabla f\left(x_k\right)$.

Nelder-Mead Simplex Method

The Nelder-Mead simplex method (not to be confused with Dantzig's simplex method in linear programming) is typically used to find the global minimum of a nonlinear real-valued function. The method works by first constructing an initial symplex (convex hull with *N+1* vertices), measuring the values of the objective function at each vertex, transforming (reflecting, expanding, or contracting) the simplex with the purpose of decreasing the values of the function at the vertices, and terminating when no significant change in the function values is observed. Since derivatives are not calculated, the Nelder-Mead simplex method belongs to the directed search family of optimization techniques.

The initial simplex is constructed by randomly generated *N+1* vertices around a given initial point. Typically, the simplex is chosen to be right-angled according to the formula along the coordinate axes $x_i = x_{initial} + l_i e_i$, where l_i is the edge length in the direction of the unit vector e_i.

Each iteration of the algorithm consists of three main steps:

1. Find vertices where the function is largest (worst), next largest, and smallest (best).
2. Calculate the centroid of the simplex area opposite the worst vertex.
3. Transform the simplex.

Simplex transformation is the core of the Nelder-Mead algorithm. The goal is to replace the worst vertex using the following consecutive transformations:

1. Reflection: $x_{\text{reflected}} = x_{\text{centroid}} + \alpha\left(x_{\text{centroid}} - x_{\text{worst}}\right)$
2. Expansion: $x_{\text{expanded}} = x_{\text{centroid}} + \gamma\left(x_{\text{reflected}} - x_{\text{centroid}}\right)$
3. Contraction: $x_{\text{contracted}} = x_{\text{centroid}} + \beta\left(x_{\text{reflected}} - x_{\text{centroid}}\right)$ or $x_{\text{contracted}} = x_{\text{centroid}} + \beta\left(x_{\text{worst}} - x_{\text{centroid}}\right)$
4. Shrinkage (completely new vertices are generated): $x_{\text{j}} = x_{\text{i}} + \delta\left(x_{\text{j}} - x_{\text{i}}\right)$, where $\alpha, \beta, \gamma, \delta$ are parameters.

Each transformation step can be terminated if better function values are found on the resulting simplex. There are three main termination criteria in the algorithm (so that it converges in finite time).

1. Domain Convergence—process terminates whenever the current simplex is sufficiently small, i.e., distances between vertices are less than $\varepsilon_1 > 0$.
2. Function Convergence—process terminates whenever function values on all vertices are similar, i.e., the distance between the function evaluated on vertices is less than $\varepsilon_2 > 0$.
3. No Convergence—process terminates whenever the number of iterations exceeds a pre-defined parameter M.

The algorithm terminates whenever any of three criteria is met.

Lagrange Multiplier

The Lagrange multiplier method was introduced to transform a constrained optimization problem to an unconstrained one. For an optimization problem where the goal is to maximize $f(x)$ subject to $g_k(x) = c_k$, where $x = (x_1 ... x_n)$, the *OptiRamp* Model Construction Submodule ensures that continuous first partial derivatives exist for both $f(x)$ and $g_k(x)$ and that gradients of g_k are not zero in the domain. The Lagrangian Λ is defined in equation (5):

$$\Lambda(x, \lambda) = f(x) + \sum_k \lambda_k g_k(x), \tag{5}$$

where λ is a vector with independent variable elements λ_k (known as Lagrange multipliers). Note that the objective functions and all of the constraints are the critical points of the Lagrangian: $\nabla_x \Lambda = 0$ if and only if $\nabla_x f = -\sum_k \lambda_k \nabla_x g_k$, where ∇_x is the gradient with respect to each element of x instead of all x_i, $i = 1, ..., n$. Furthermore, $\nabla_\lambda \Lambda = 0$ also implies that $g_k = 0$.

A Lagrange multiplier can be interpreted as the rate of change of the objective function with respect to the constraint, as is seen from equation (6):

$$\lambda_k = \frac{\partial \Lambda}{\partial g_k} \tag{6}$$

Once the Lagrangian is calculated, its minimum can be found using the non-linear optimization methods described.

About Statistics & Control, Inc.

S&C—an engineering consulting and technology company headquartered in West Des Moines, IA—solves complex challenges for customers through its unique technology and its highly seasoned team of professionals. The company has a global portfolio spanning the energy, oil and gas, utility, and digital oil field industry sectors. S&C provides clients with turbomachinery control solutions that easily integrate with the existing system as well as *OptiRamp*® solutions, which focus on process and power analytics to optimize processes and, in turn, reduce costs and increase reliability. S&C also provides consulting, dynamic system studies, modeling, automation, training and OTS, and support services.

Statistics & Control, Inc.
4401 Westown Pkwy, Suite 124
West Des Moines, IA 50266 USA
Phone: 1.515.267.8700
Fax: 1.515.267.8701