Optimization Using Feedback Control

Did you know that optimization problems can be solved online using feedback control? Consider the task of minimizing a scalar cost function J, or equivalently, maximizing the profit −J. Typically, J represents an economic quantity with units such as [$/s]. We assume that we can influence J through a manipulated input variable u, and that we may also have access to measurements y. Furthermore, we assume the system is static (i.e., without dynamics), so we can write J=J(u).

Our objective is to minimize J(u) in real time using feedback control. There are three main approaches to doing this: purely data-based, model-based, and self-optimizing control (offline model based). In addition, there are hybrid methods and combinations. Actually, all three methods can be combined.

---

1. Purely Data-Based Feedback Optimization (ESC)
Also known as extremum seeking control (ESC), hill-climbing, greedy search, or perturb-and-observe.

This method relies entirely on measurements of the cost J (which may not always be available) and does not use a process model. It is simple, but generally slow, especially when gradient estimation is involved. In classical ESC, sinusoidal perturbations are used to estimate the gradient, but this is not always an efficient method.

A simpler and often more intuitive version is the **perturb-and-observe** algorithm (commonly used, for example, to optimize the position of wind turbines):

- **Step 1**: Apply a perturbation Δu to the input:
  $u(k)=u(k-1)+\text{sign}(k)\cdot \Delta u$
  where k is the current time sample and sign(k) alternates between +1 and -1.
- **Step 2**: Observe the cost J(k+1). If J(k+1)<J(k), then the system is moving in the right direction so keep the same sign; otherwise, reverse the direction (change sign(k)). Repeat from Step 1.

This method has two main tuning parameters:

- The **step size** Δu: Larger steps speed up convergence but cause larger oscillations around the optimum.
- The **sampling time** T_s: This must be long enough to let the system respond. A rule of thumb is $T_s \approx 3\tau$, where τ is the system's time constant.

This method can be combined with faster model-based optimization methods like RTO by letting u be the **bias** for the RTO-gradient. This hybrid approach is often called **modifier adaptation**.

---

2. Standard Model-Based Optimization (RTO)
Real-Time Optimization (RTO) uses a detailed nonlinear model and does not require that the cost J is measured.

The system model is used to compute the input u(k) that minimizes the cost J. Measurements y are used online to update selected model parameters (e.g., efficiencies). RTO minimizes the cost J while explicitly handling constraints.

---

3. Self-Optimizing Control (SOC)
Here, the model is used offline to find good "self-optimizing" variables c. The online implementation uses a simple PID controller to keep c and its setpoint c_s.

The goal of SOC is to find a **"magic variable"** c to control—ideally one where a constant setpoint c_s leads to near-optimal performance despite disturbances. This allows the optimization to be embedded into the fast control layer.

In its simplest form, c=y is a single measurement. The variable c is chosen such that its setpoint c_s is **insensitive to disturbances**, yet c is **sensitive to input changes** (i.e., has a large gain from u to c).

More generally, we can use combinations of measurements: c=Hy where H is a matrix. Ideally, this corresponds to the **gradient**: $c=J_u$. See the paper by Bernardino and Skogestad (2024) for methods to estimate this gradient.

3C. Self-Optimizing Control with Constraints

SOC can be extended to handle constraints by incorporating Lagrange multipliers into the control strategy. In this approach, the **multiplier acts as a manipulated variable in an upper slow control layer**. Constraint violation can be avoided using **override logic** (Dirza and Skogestad, 2024). In essence, this is a clever trick where a PI controller is used to iteratively solve a set of equations numerically.

Combinations

These approaches can be combined:

- **1 + 2**: ESC (slow) sends bias for gradient J_u (modifier adaptation) to RTO. The bias means that the RTO-gradient will not be zero. This is to correct for model errors, unmeasured disturbances and measurement errors in the RTO-layer.
- **1 + 3**: ESC (slow) updates setpoints to SOC (fast).
- **2 + 3**: RTO (slow) updates setpoints to SOC (fast).
- **1 + 2 + 3**: ESC (slow) sends bias for J_u to RTO (faster), which updates setpoints to the SOC layer (fastest).

There needs to be a time scale separation between the layers, typically about 10.

References

* Optimal switching of MPC cost function for changing active constraints. LF Bernardino, S Skogestad. Journal of Process Control 142, 103298. 2024

* Optimal measurement-based cost gradient estimate for feedback real-time optimization LF Bernardino, S Skogestad. Computers & Chemical Engineering, 108815 2024

* Primal-dual feedback-optimizing control with override for real-time optimization R Dirza, S Skogestad. Journal of Process Control 138, 103208 3 2024

* Decentralized control using selectors for optimal steady-state operation with changing active constraints LF Bernardino, S Skogestad. Journal of Process Control 137, 103194 5 2024