Ian David Lockhart Bogle and Michael Fairweather (Editors), Proceedings of the 22nd European Symposium on Computer Aided Process Engineering, 17 - 20 June 2012, London.

Run-to-Run MPC Tuning via Gradient Descent

Gene A. Bunin, Fernando Fraire Tirado, Grégory François, and Dominique Bonvin
Laboratoire d'Automatique, École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland.

Abstract

A gradient-descent method for the run-to-run tuning of MPC controllers is proposed. It is shown that, with an assumption on process repeatability, the MPC tuning parameters may be brought to a locally optimal set. SISO and MIMO examples illustrate the characteristics of the proposed approach.

Keywords: Model predictive control, Run-to-run adaptation, Gradient descent.

1. MPC Formulation and Tuning

The success of model predictive control (MPC) is well documented. With the ability to explicitly handle constraints, to "look ahead", and to calculate an adequate model-based control action even with a very rough linear model of the dynamics, its use has now expanded into many diverse fields and applications (Qin and Badgwell (2003)).

MPC tuning, however, remains somewhat *ad hoc*. Though there now exists a fair body of literature for good heuristic tuning choices, which can allow an operator to tune the MPC offline before applying it to the real process (Garriga and Soroush (2010)), the obtained parameters will only be nominally optimal and, when the model uncertainty is significant, may be unable to yield satisfactory performance in practice. A particular case where this problem can be solved in a general, algorithmic manner is the one of batch processes, where the MPC controller may be asked to track the same reference profile many times – for example, when maintaining a cooling profile in a crystallizer (Shen et al. (1999)). In this paper, we propose to solve this run-to-run (or "batch-to-batch") problem via gradient-descent optimization, noting that a simpler realization of what is essentially the same idea but with a single tuning parameter may be found in the work of Magni et al. (2009).

Any MPC controller requires a dynamical model of the system, $[\hat{\mathbf{y}}_{k+1},...,\hat{\mathbf{y}}_{k+n}] = f(\mathbf{u}_k, ..., \mathbf{u}_{k+n-1})$, that is able to predict how the outputs $\mathbf{y} \in \mathbb{R}^{n_y \times 1}$ will evolve when driven by the inputs $\mathbf{u} \in \mathbb{R}^{n_u \times 1}$ over some discrete prediction interval $[k+1, k+n]$[1].

The majority of MPC schemes use this model to calculate the optimal control action at the current iteration k, $\mathbf{u}_k^*$, by solving the following problem over the constrained set $\mathscr{U}$:

$$\begin{aligned} \underset{\mathbf{u}_k,...,\mathbf{u}_{k+n-1}}{\text{minimize}} \quad & \sum_{i=k+1}^{k+n} \|\mathbf{Q}(\mathbf{y}_{set,i} - \hat{\mathbf{y}}_i - \mathbf{d}_k)\| + \sum_{i=k}^{k+n-1} \|\mathbf{R}(\mathbf{u}_i - \mathbf{u}_{i-1})\| \\ \text{subject to} \quad & \mathbf{u}_k, ..., \mathbf{u}_{k+m-1} \in \mathscr{U}; \; \mathbf{u}_{k+m+j} = \mathbf{u}_{k+m-1}, \forall j = 0, ..., n-m-1 \end{aligned} \tag{1}$$

where $\mathbf{Q} \in \mathbb{R}^{n_y \times n_y}, \mathbf{R} \in \mathbb{R}^{n_u \times n_u}$ are diagonal weighting matrices, m is the control horizon, and $\mathbf{d}_k$ is the output bias at the time instant k, which is estimated via the filtering law, $\mathbf{d}_k = \mathbf{K}(\mathbf{y}_k - \hat{\mathbf{y}}_k) + (\mathbf{I}_{n_y} - \mathbf{K})\mathbf{d}_{k-1}$, with $\mathbf{K} \in \mathbb{R}^{n_y \times n_y}$ a diagonal matrix of bias filters.

[1] $(\hat{\cdot})$ denotes a model-based prediction, while $(\cdot)$ denotes a measured value.

When constant setpoints $\mathbf{y}_{set,c}$ are to be tracked, the setpoint $\mathbf{y}_{set,i}$ may be defined via a reference trajectory, $\mathbf{y}_{set,i} = \left(\mathbf{I}_{n_y} - e^{-\mathbf{B}^{-1}i}\right)\mathbf{y}_{set,c}$, parameterized via the diagonal matrix $\mathbf{B} \in \mathbb{R}^{n_y \times n_y}$, to add a degree of robustness. We note that m, n, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{K}$, and $\mathbf{B}$ all constitute tuning parameters.

2. Proposed Method for Run-to-Run MPC Tuning

2.1. Optimal MPC Performance as a Static Optimization Problem

Given a process that uses an MPC controller to meet certain criteria during operation, we would like to vary the MPC tuning parameters between each run of the process in an intelligent manner so as to improve the controller's performance. In order to do so, we are required to make the following assumption.

Assumption 1 (Repeatability)

For a set of MPC tuning parameters $\theta \in \mathbb{R}^{n_\theta}$, the obtained MPC performance $P_j(\theta)$ for any given run j will be represented by a deterministic, run-independent function $P_d(\theta)$ and an additive stochastic, run-dependent element δ_j:

$$P_j(\theta) = P_d(\theta) + \delta_j \tag{2}$$

For the proposed method to be applicable, it is sufficient for $P_d(\theta)$ to exist, and for its effects to overwhelm those of δ (i.e. runs with identical θ should yield very similar results). Practically, the deterministic part corresponds to the familiar, but analytically unknown, relations between the tuning parameters and MPC behavior, while the stochastic part corresponds to measurement noise and disturbances that are specific to a run.

The following metric for $P_d(\theta)$ is proposed to quantify "performance" based on recorded input and output data for a specific run, expressed as (unknown) functions of θ[2]:

$$P_d(\theta) = \sum_{i=1}^{k_f} \|\mathbf{Q}_P(\mathbf{y}_{set,i} - \mathbf{y}_i(\theta))\| + \sum_{i=0}^{k_f-1} \|\mathbf{W}_P\mathbf{u}_i(\theta)\| + \sum_{i=0}^{k_f-1} \|\mathbf{R}_P(\mathbf{u}_i(\theta) - \mathbf{u}_{i-1}(\theta))\| \tag{3}$$

with k_f being the value of the counter at the end of the run, and $\mathbf{Q}_P \in \mathbb{R}^{n_y \times n_y}, \mathbf{W}_P, \mathbf{R}_P \in \mathbb{R}^{n_u \times n_u}$ being diagonal weighting matrices (let $\mathbf{u}_{-1}$ denote the starting inputs). With $\mathbf{Q}_P$, $\mathbf{W}_P$, and $\mathbf{R}_P$, one judges performance based on the MPC's ability to track without using excessive resources or aggressive control action.

The choice of these matrices should not be arbitrary, and may stem from the simulated performance for the nominally tuned MPC. Let $(\bar{\cdot})$ denote values obtained in simulation. For each diagonal value $q_P \in \mathbf{Q}_P, w_P \in \mathbf{W}_P, r_P \in \mathbf{R}_P$, one may then define, based on the corresponding inputs and outputs, $q_P = 1/\Sigma_{i=1}^{k_f} \|y_{set,i} - \bar{y}_i\|, w_P = 1/\Sigma_{i=0}^{k_f-1} \|\bar{u}_i\|, r_P = 1/\Sigma_{i=0}^{k_f-1} \|\bar{u}_i - \bar{u}_{i-1}\|$. In this manner, the "good" qualitative performance found in simulation may be quantified.

Finally, by limiting the tuning parameters to lie in some defined set Θ, we may now formulate the problem of run-to-run MPC tuning as a static optimization problem where we seek to minimize the deterministic part of the performance index:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & P_d(\theta) \\ \text{subject to} \quad & \theta \in \Theta \end{aligned} \tag{4}$$

[2] In the case of a reference trajectory, the $\mathbf{y}_{set}$ in (3) refers to the original, unfiltered setpoint.

2.2. Gradient-Descent Optimization Algorithm

The gradient-descent method may be used to bring $P_d(\theta)$ to a locally optimal set θ^*, provided that a good approximation of the gradient $\nabla P_d(\theta)$ may be obtained for all $\theta \in \Theta$. The proposed algorithm starts by using the nominal model gradient $\nabla \hat{P}_d(\theta)$ to achieve fast improvement without requiring extra runs to estimate the gradient. When no more progress is possible with $\nabla \hat{P}_d(\theta)$, $\nabla P_d(\theta)$ is estimated and used to "fine tune" the solution.

Algorithm 1 (Run-to-Run MPC Parameter Tuning)

1. Initialize: Tune the MPC controller offline and obtain θ_0, $\mathbf{Q}_P$, $\mathbf{W}_P$, and $\mathbf{R}_P$. Initialize $\theta_h := \theta_0$. Let H denote a logical switch that determines whether or not the true process gradient should be estimated, and set $H := 0$ (model-based gradient).
2. Gradient Definition: If $H = 0$, $\nabla P(\theta_h) := \nabla \hat{P}_d(\theta_h)$. If $H = 1$, $\nabla P(\theta_h) := \nabla P_d(\theta_h)$.
3. Line Search (Algorithm 2): Solve approximately, with $t_L \geq 0$,
$$t_L^* = \arg\underset{t_L}{\text{minimize}} \quad P_d\left(\theta_h - t_L \nabla P(\theta_h)\right) \tag{5}$$
If $t_L^* = 0$ and $H = 0$, set $\nabla P(\theta_h) := -\nabla \hat{P}_d(\theta_h)$ and redo the line search. If $t_L^* = 0$ again, set $H := 1$ and return to Step 2.
4. Update: $\theta_{h+1} := \theta_h - t_L^* \nabla P(\theta_h)$.
5. Projection: If $\theta_{h+1} \notin \Theta$
$$\begin{aligned} \theta_{h+1} := \arg\underset{\theta}{\text{minimize}} \quad & \|\theta_{h+1} - \theta\| \\ \text{subject to} \quad & \theta \in \Theta \end{aligned} \tag{6}$$
6. Termination: If $\|\theta_{h+1} - \theta_h\| < \varepsilon$, then terminate. Else, set $h := h + 1$ and return to Step 2.

A practical run-to-run adaptation should be able to obtain significant improvement quickly and have mostly monotonic improvement from run to run. The above algorithm is believed to achieve those goals. Ideally, the model gradient captures the main relations between the parameters and the performance to provide a good descent direction. If this direction is false and one of ascent, one may simply reverse it to go in a descent direction. If both yield ascent directions, the usefulness of the model gradient is exhausted and one must switch to gradient estimation to find the locally optimal set.

The line search is designed to achieve fast improvement without requiring too many iterations. Because the gradient may be very local, the search begins with very small steps and, while improvement is noted, doubles the step size until the maximum allowable change in parameters per run is reached (this requires S iterations, with S set by the user). Multiples of this step are then applied. The search terminates as soon as an increase in the function value is noted, and takes its previous value as the optimal step:

Algorithm 2 (Line Search)

1. Initialize: $\theta_l := \theta_h$. Set $M := 1$, where M is a multiplier used to augment the step size following observed improvement. Use $\Delta\theta_{max}$, the maximum allowable change in θ from run to run, to define $t_{L,max}$ as
$$t_{L,max} = \{\sup t_L : -\Delta\theta_{max} \leq -t_L \nabla P(\theta_h) \leq \Delta\theta_{max}\} \tag{7}$$
2. Step: $\theta_{l+1} := \theta_l - M\, t_{L,max} \nabla P(\theta_h)/2^{S-1}$.
3. Projection: If $\theta_{l+1} \notin \Theta$
$$\begin{aligned} \theta_{l+1} := \arg\underset{\theta}{\text{minimize}} \quad & \|\theta_{l+1} - \theta\| \\ \text{subject to} \quad & \theta \in \Theta \end{aligned} \tag{8}$$
4. Termination: If $P_d(\theta_{l+1}) \leq P_d(\theta_l)$ and $M \leq 2^{S-1}$, set $M := 2M$, $l := l + 1$, and return to Step 2. If $P_d(\theta_{l+1}) \leq P_d(\theta_l)$ and $M > 2^{S-1}$, set $M := M + 2^{S-1}$, $l := l + 1$, and return to Step 2. Else, set t_L^* as corresponding to θ_l and terminate.

As an example, consider $t_{L,max} = 8$ and $S = 4$. The sequence of step sizes would then be 1,2,4,8,16,24... .

3. Illustrative Examples

We illustrate the proposed method on both a SISO and a MIMO example with $\theta := [\tilde{m}, \tilde{n}, \mathbf{q}, \mathbf{r}, \mathbf{b}, \mathbf{k}]$, with $(\tilde{\cdot})$ denoting a scaling and $\mathbf{q}, \mathbf{r}, \mathbf{b}, \mathbf{k}$ denoting the diagonals of $\mathbf{Q}, \mathbf{R}, \mathbf{B}$, $\mathbf{K}$, respectively. Both are set to be noise free ($\delta = 0$), with the gradients being estimated via two perturbations (in opposite directions) of size $\Delta\theta_e := [0.01, 0.01, \mathbf{0.02}, \mathbf{0.02}, \mathbf{0.04}, \mathbf{0.02}]$ for each parameter. A single perturbation in the feasible direction is applied when a parameter is at its boundary.

An MPC controller with $\mathscr{U} = \mathbb{R}^{n_u}$ is programmed as outlined in Section 1, with a squared 2-norm used in the objective. $q \in \mathbf{q}$, $r \in \mathbf{r}$, $b \in \mathbf{b}$, $K \in \mathbf{k}$ denote individual components. A constant bias is assumed in simulation to help nominally tune $\mathbf{K}$ (0.3 for y in the SISO case, and 0.3 and 0.6 for y_1 and y_2 in the MIMO case, respectively). The projection step is simplified to: 1) rounding m and n to their nearest integers, 2) setting $\theta := \theta^U = [2.00, 2.00, \mathbf{1.00}, \mathbf{1.00}, \mathbf{2.00}, \mathbf{1.00}]$ or $\theta := \theta^L = [0.02, 0.02, \mathbf{0.10}, \mathbf{0.10}, \mathbf{0.10}, \mathbf{0.10}]$ as necessary when upper and lower bounds on the parameters are violated, and 3) setting $m = n := (m+n)/2$ when $m > n$. A 1-norm is used to define $P_d(\theta)$. To relax the rounding error on m and n, large values are chosen and scaled by a factor of 100 to give $\tilde{m}$ and $\tilde{n}$ that are comparable in size to the other parameters. Both examples involve 20-min batches where constant setpoints are to be tracked, with an MPC action every 6 s. A 2-norm termination criterion with the threshold $\varepsilon = 10^{-3}$ is used. For the line search, $\Delta\theta_{max} := [0.10, 0.10, \mathbf{0.10}, \mathbf{0.10}, \mathbf{0.20}, \mathbf{0.10}]$ with $S := 4$.

The following transfer functions describe the simulated "real" process, $g(s)$ and $G(s)$, and the corresponding models, $\hat{g}(s)$ and $\hat{G}(s)$:

$$\begin{aligned}
&\hat{g}(s) = \frac{0.8s+2.2}{1.8s^2+0.9s+8}, \quad g(s) = \frac{0.9s+2.1}{0.01s^3+2s^2+s+6.5} \\
&\hat{G}(s) = \begin{bmatrix} \frac{1}{2s+3} & \frac{s+2}{2s^2+s+5} & \frac{-s+2}{s^3+2s^2+s+1} \\ \frac{9s+1}{s^2+s+1} & \frac{s-2}{s+1} & \frac{2}{s+5} \end{bmatrix}, \quad G(s) = \begin{bmatrix} \frac{1.5}{s+4} & \frac{s+2}{2s^2+1.2s+6} & \frac{s+2}{s^3+2s^2+s+1} \\ \frac{8s+0.8}{1.7s^2+1.2s+0.8} & \frac{s-2}{2s+1} & \frac{2}{s+4} \end{bmatrix}
\end{aligned} \tag{9}$$

Fig. 1 illustrates the control performance, with Table 1 providing the parameter information.

Table 1. Parameter values for the SISO and MIMO examples.

SISO	$\tilde{m}$	$\tilde{n}$	q	r	b	K	MIMO	$\tilde{m}$	$\tilde{n}$	$\mathbf{q}$	$\mathbf{r}$	$\mathbf{b}$	$\mathbf{k}$
θ_0	.70	1.00	.70	.39	1.20	.43	θ_0	.70	1.00	.50,.60	.20,.50,.50	1.20,.30	.50,.20
θ^*	.70	1.00	1.00	.10	1.35	1.00	θ^*	.71	1.23	.10,.75	.10,.99,1.00	.98,.27	1.00,1.00
$\frac{dP_d}{d\theta}\vert_{\theta^*}$	.00	.00	-.15	1.51	-.01	-.31	$\frac{dP_d}{d\theta}\vert_{\theta^*}$	.21	.03	.59,-.02	1.07,-.04,-.10	.02,.00	-.04,-.12

In the SISO case, the method works "as planned", in that a good direction is found with the model gradient, and the majority of the improvement is gained in the first 10 batches, with the switch to the estimated gradient occurring after 20 batches. However, the resulting improvement is relatively small as the process is already in a relatively optimal region. The MIMO scenario illustrates a less ideal case, where the model gradient proves almost useless and gradient estimation begins after just 6 batches. For 11 parameters, this requires a total of 22 perturbations before significant improvement is achieved through the line search. The theoretically slow convergence of the gradient-descent method (Boyd and Vanderberghe (2008)) is also witnessed, as nearly 2,500 batches are needed to fully converge. Practically, however, this is not a drawback – in Table 2 one sees that the majority of optimality gains are achieved with significantly fewer batches. In both cases, it is possible to verify, in a somewhat cursory manner, that the necessary conditions of optimality hold, as the derivatives with respect to the parameters that are not at their constraints are approximately 0.

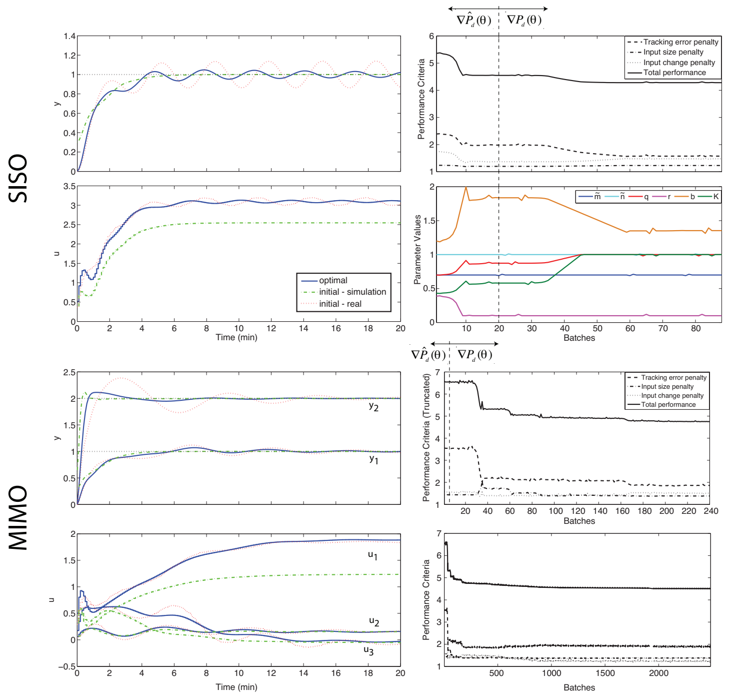

Figure 1. Simulated MPC performance for the SISO and MIMO cases. The performance for the initial and optimal θ is given on the left, with the evolution of the performance index over batches given on the right. For the MIMO case, a truncated view is also given. A vertical dashed line indicates the batch number where gradient estimation begins.

Table 2. Optimality gains with respect to number of runs, which comprises the runs needed for both the line search and gradient estimation.

SISO	0	9	45	88	MIMO	0	34	61	87	168	1000	2484
$\frac{P_d(\theta_0)-P_d(\theta_k)}{P_d(\theta_0)-P_d(\theta^*)}\cdot 100\%$	0	75.0	96.3	100	$\frac{P_d(\theta_0)-P_d(\theta_k)}{P_d(\theta_0)-P_d(\theta^*)}\cdot 100\%$	0	60.9	73.8	79.4	86.2	97.6	100

For processes where batches are particularly costly, an uninformative model gradient may be compensated for by working with only a subset of the parameters, by lowering the number of perturbations used for the estimation step, or by using more advanced gradient estimation techniques. This is particularly pertinent to MIMO processes, where the dimensionality of the problem is naturally higher.

References

Boyd, S., Vanderberghe, L., 2008. Convex Optimization. Cambridge University Press.

Garriga, J., Soroush, M., 2010. Model predictive control tuning methods: A review. Ind. Eng. Chem. Res. 49, 3505–3515.

Magni, L., Forgione, M., Toffanin, C., Man, C., Kovatchev, B., De Nicolao, G., Cobelli, C., 2009. Run-to-run tuning of model predictive control for type 1 diabetes subjects: In silico trial. J. of Diabetes Science and Technology 3 (5), 1091–1098.

Qin, S. J., Badgwell, T. A., 2003. A survey of industrial model predictive control technology. Contr. Eng. Practice 11, 733–764.

Shen, J., Chiu, M., Wang, Q., 1999. A comparative study of model-based control techniques for batch crystallization process. Journal of Chemical Engineering of Japan 32 (4), 456–464.