

Control and dynamical system tools for optimization

Integral control for (seemingly) diverse problems

Prof. Ivano Notarnicola

Dept. of Electrical, Electronic, Information Engineering
Alma Mater Studiorum Università di Bologna (Italy)
ivano.notarnicola@unibo.it

ENSTA • Institut Polytechnique de Paris (France)
09 February 2026

Step response of a system

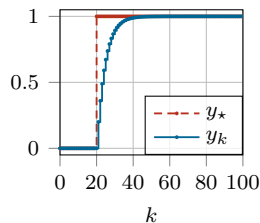
Consider a discrete-time linear system

$$x_{k+1} = 0.8x_k + 0.2u_k$$

$$y_k = x_k$$

and choose $u_k = 0$ for all $0 \leq k < 20$ and then $u_k = 1 = y_\star$ for all $k \geq 20$

Remark. This is called the *open-loop* step response of the system



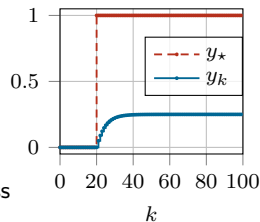
Consider a slightly modified discrete-time linear system

$$x_{k+1} = 0.8x_k + 0.05u_k$$

$$y_k = x_k$$

and choose $u_k = 0$ for all $0 \leq k < 20$ and then $u_k = 1 = y_\star$ for all $k \geq 20$

Remark. Perfect tracking is no longer achieved, the *open-loop* control lacks robustness



Proportional control

A *closed-loop* control input u_k can be designed to improve the output y_k by using also the instantaneous value y_k

Consider a discrete-time linear system

$$x_{k+1} = 0.8x_k + 0.2u_k$$

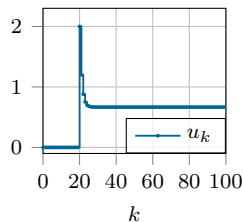
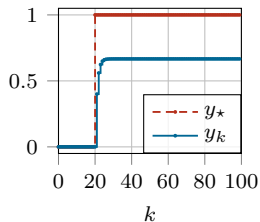
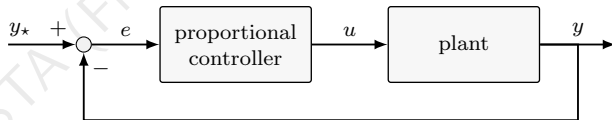
$$y_k = x_k$$

and choose the input u_k by elaborating $e_k = y_k - y_*$ as

$$u_k = -K_P e_k$$

where $K_P = 0.2$ and the reference $y_* = 1$

Remark. Faster, but not precise



Proportional-integral control

Integral control is a useful technique to generate a closed-loop control input u that enables the output y_k to exactly track a *constant reference* y_* , in a robust manner

Consider a discrete-time linear system

$$x_{k+1} = 0.8x_k + 0.2u_k$$

$$y_k = x_k$$

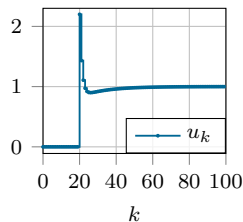
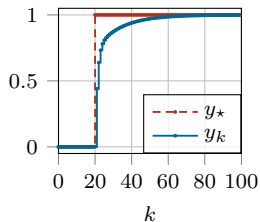
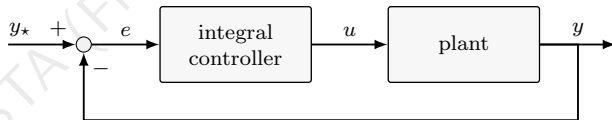
and choose the input u_k by elaborating $e_k = y_k - y_*$ as

$$\xi_{k+1} = \xi_k + e_k$$

$$u_k = K_I \xi_k + K_P e_k$$

where $K_I = 0.2$, $K_P = 2$, and the reference $y_* = 1$

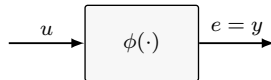
Remark. Fast and precise



Integral control for algebraic maps

Consider the algebraic (static, possibly nonlinear) map $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$

$$u \mapsto e = \phi(u)$$



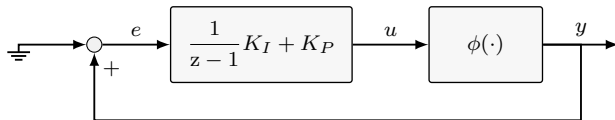
Goal. Steer the error $e = y$ to zero using a (dynamic) feedback controller

$$\xi_{k+1} = \xi_k + e_k$$

$$u_k = K_I \xi_k + K_P e_k$$

where K_I and K_P are the *integral* and *proportional* gains

As long as the interconnection is stable (bounded states), the it must necessarily be $\lim_{k \rightarrow \infty} e_k = 0$



Linear averaging

Each agent $i = 1, \dots, N$ in a network, modeled as a graph, performs

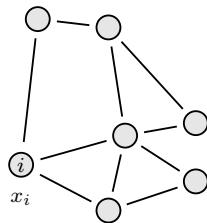
$$x_{i,k+1} = \sum_{j \in N_i} w_{ij} x_{j,k}, \quad x_{i,0} \in \mathbb{R}$$

where N_i is the set of neighbors of i (including itself) and $w_{ij} \geq 0$ are mixing weights

Setting $x = (x_1, \dots, x_N)$, the network dynamics is

$$x_{k+1} = W x_k, \quad x_0 \in \mathbb{R}^N$$

where $W \in \mathbb{R}^{N \times N}$ is a sparse, doubly-stochastic matrix



It is an autonomous linear system with a marginally stable pole in $z = 1$, whose (free) evolution is

$$\lim_{k \rightarrow \infty} x_k = \lim_{k \rightarrow \infty} W^k x_0 = \underbrace{\frac{1}{N} \mathbf{1} \mathbf{1}^\top}_{:= J} x_0$$

Remark. The motion asymptotically aligns with the eigenvector of the dominant (constant) mode

Consensus (discrete-time) algorithm

The *consensus problem* among N agents amounts to computing $u \in \mathbb{R}^N$ such that $Lu = 0_N$, where L is the Laplacian matrix $L \in \mathbb{R}^{N \times N}$ of the communication (connected undirected aperiodic) graph

The Laplacian mixing is modeled by the following algebraic (linear) map

$$u \mapsto e = Lu$$

If the error $e = 0_N$, then it must be $u \in \text{span } \mathbf{1}$ (consensus)

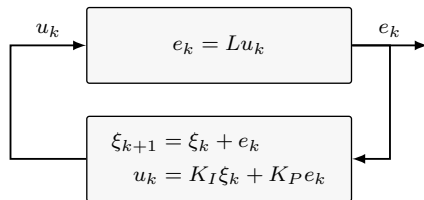
Consider a *discrete-time PI* controller

$$\xi_{k+1} = \xi_k + e_k$$

$$u_k = K_I \xi_k + K_P e_k$$

Setting $K_I = -\alpha I_N$, with a sufficiently small $\alpha > 0$, and $K_P = 0$ yields

$$\begin{aligned} \xi_{k+1} &= \xi_k + L(-\alpha \xi_k) \\ &= \underbrace{(I - \alpha L)}_W \xi_k \end{aligned}$$



Laplacian (continuous-time) averaging

The Laplacian mixing is modeled by the following algebraic (linear) map

$$u \longmapsto e = Lu$$

Consider a *continuous-time PI* controller

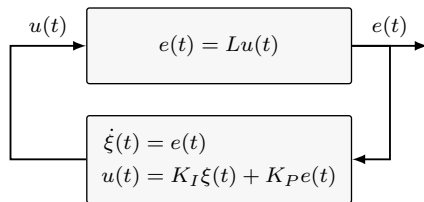
$$\dot{\xi}(t) = e(t)$$

$$u(t) = K_I \xi(t) + K_P e(t)$$

Setting $K_I = -I_N$ and $K_P = 0$ yields

$$\dot{\xi}(t) = -L\xi(t)$$

$$\dot{\xi}_i(t) = - \sum_{j \in \mathbb{N}_i} a_{ij} (\xi_i(t) - \xi_j(t))$$



Remark. The discrete-time consensus algorithm is the *Forward-Euler discretization* of the Laplacian averaging dynamics with *sampling time* $\alpha > 0$

The gradient method as a controlled nonlinearity

Unconstrained optimization amounts to computing $u \in \mathbb{R}^n$ such that $\nabla f(u) = 0_n$

The gradient operator is an algebraic nonlinear map

$$u \mapsto e = \nabla f(u)$$

If the error $e = 0_n$, then u must be a stationary point of f

Consider the discrete-time PI controller

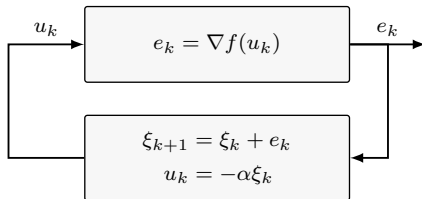
$$\begin{aligned}\xi_{k+1} &= \xi_k + e_k \\ u_k &= K_I \xi_k + K_P e_k\end{aligned}$$

Setting $K_I = -\alpha I_n$, with a sufficiently small $\alpha > 0$, and $K_P = 0$ yields

$$\xi_{k+1} = \xi_k + \nabla f(-\alpha \xi_k)$$

Changing coordinates via $\xi \mapsto x := -\alpha \xi$, the gradient method is recovered

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$



Proportional-integral control: the proximal minimization method

If a more general (discrete-time) PI controller is adopted ($K_P = -\alpha I_n \neq 0$), then

$$\xi_{k+1} = \xi_k + e_k$$

$$u_k = -\alpha \xi_k - \alpha e_k = -\alpha \xi_{k+1}$$

The resulting closed-loop dynamics has the following *implicit update*

$$\xi_{k+1} = \xi_k + \nabla f(-\alpha \xi_{k+1})$$

Changing coordinates via $\xi \mapsto x := -\alpha \xi$ yields

$$x_{k+1} = x_k - \alpha \nabla f(x_{k+1})$$

which coincides with the so-called *proximal minimization method*

$$x_{k+1} = \operatorname{argmin}_x f(x) + \frac{1}{2\alpha} \|x - x_k\|^2$$

The gradient flow

The gradient operator is an algebraic nonlinear map

$$u \mapsto e = \nabla f(u)$$

Consider the continuous-time PI controller

$$\dot{\xi}(t) = e(t)$$

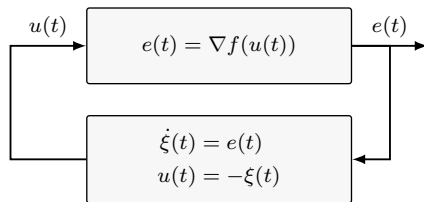
$$u(t) = K_I \xi(t) + K_P e(t)$$

Setting $K_I = -I_n$ and $K_P = 0$ yields

$$\dot{\xi}(t) = \nabla f(-\xi(t))$$

Changing coordinates via $\xi \mapsto x := -\xi$ recovers the *gradient flow*

$$\dot{x}(t) = -\nabla f(x(t))$$



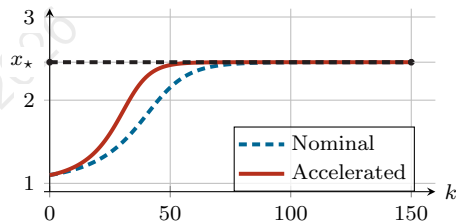
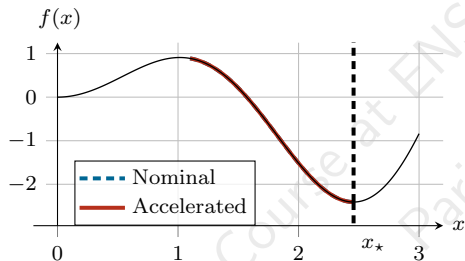
Remark. The gradient method is the *Forward-Euler discretization* of the gradient flow with *sampling time* $\alpha > 0$

Remark. The PMM is the *Backward-Euler discretization* of the gradient flow, which works for all $\alpha > 0$

Accelerating the gradient method

Consider the (nonconvex) function $f(x) = x \sin(2x)$ and compare

- the nominal gradient method (in blue)
- an accelerated instance (in red)



Remark. They really look like “step responses”...

The heavy-ball method

Idea. The nominal gradient method can be enhanced by elaborating on the closed-loop *performances*

The *heavy-ball method* is described by the following update

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1})$$

where $\beta \in (0, 1)$ is the so-called *momentum* parameter

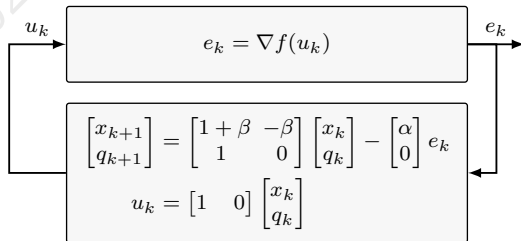
The state-space realization is a second-order dynamics given by

$$x_{k+1} = (1 + \beta)x_k - \beta q_k - \alpha e_k$$

$$q_{k+1} = x_k$$

$$u_k = x_k$$

in feedback with $e_k = \nabla f(u_k)$



Remark. What about the zero-pole map of the linear subsystem?

The “realm” of an accelerated gradient method

The poles of the linear part of the heavy-ball method are the state matrix eigenvalues, namely the roots of

$$(\lambda - 1 - \beta)\lambda + \beta = 0 \quad \Rightarrow \quad \begin{aligned} \lambda_1 &= 1 \\ \lambda_2 &= \beta \end{aligned} \quad (\text{stable poles})$$

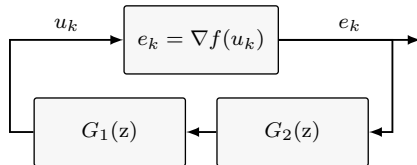
As for the zeros, the state-space realization reveals a zero at $z = 0$

Computing the Z-transform of the state-space equations gives

$$\begin{aligned} zX(z) &= (1 + \beta)X(z) - \beta Q(z) - \alpha E(z) \\ zQ(z) &= X(z) = U(z) \end{aligned} \quad \Rightarrow \quad \begin{aligned} Q(z) &= \frac{1}{z}U(z) \\ (z - 1 - \beta)U(z) &= -\beta \frac{1}{z}U(z) - \alpha E(z) \end{aligned}$$

Therefore, it can always be represented as

$$G(z) = \frac{U(z)}{E(z)} = -\frac{\alpha}{z - 1 - \beta + \beta \frac{1}{z}} = \underbrace{\frac{z}{z - \beta}}_{G_1(z)} \underbrace{\frac{-\alpha}{z - 1}}_{G_2(z)}$$



namely, an integrator (the gradient method) in series with a *lag compensator*

Nesterov's method

The *Nesterov's method* is described by the following updates

$$\begin{aligned}\eta_k &= x_k + \beta(x_k - x_{k-1}) \\ x_{k+1} &= \eta_k - \alpha \nabla f(\eta_k)\end{aligned}$$

where η_k extrapolates based on the current iterate x_k and the previous one x_{k-1} , using a momentum parameter $\beta > 0$. The descent step is then performed based on η_k

The state-space realization is a second-order dynamics given by

$$\begin{aligned}x_{k+1} &= (1 + \beta)x_k - \beta q_k - \alpha e_k \\ q_{k+1} &= x_k \\ u_k &= (1 + \beta)x_k - \beta q_k\end{aligned}$$

in feedback with $e_k = \nabla f(u_k)$

Remark. How does this differ from the heavy-ball method?

Transfer function of Nesterov's method

Consider the open-loop ($n = 1$, SISO) system

$$\begin{bmatrix} x_{k+1} \\ q_{k+1} \end{bmatrix} = \begin{bmatrix} 1 + \beta & -\beta \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ q_k \end{bmatrix} - \begin{bmatrix} \alpha \\ 0 \end{bmatrix} e_k$$
$$u_k = \begin{bmatrix} 1 + \beta & -\beta \end{bmatrix} \begin{bmatrix} x_k \\ q_k \end{bmatrix}$$

The transfer function from e_k to u_k is

$$\begin{aligned} G(z) &= \frac{U(z)}{-E(z)} = \begin{bmatrix} 1 + \beta & -\beta \end{bmatrix} \begin{bmatrix} z - (1 + \beta) & \beta \\ -1 & z \end{bmatrix}^{-1} \begin{bmatrix} -\alpha \\ 0 \end{bmatrix} \\ &= -\alpha \frac{(1 + \beta)z - \beta}{z^2 - (1 + \beta)z + \beta} \\ &= \underbrace{\frac{(1 + \beta)z - \beta}{z - \beta}}_{G_1(z)} \underbrace{\frac{-\alpha}{z - 1}}_{G_2(z)} \end{aligned}$$

Remark. Nesterov's method has the same poles as the heavy ball method, but the zero is at $z = \frac{\beta}{1 + \beta}$

Analysis of the Nesterov's method

Applying the two loop transformations to the Nesterov's method for strongly convex smooth problems gives

$$\tilde{x}_{k+1} = (1 - \alpha\mu)(1 + \beta)\tilde{x}_k - (1 - \alpha\mu)\beta\tilde{q}_k - \alpha\hat{u}_k$$

$$\tilde{q}_{k+1} = \tilde{x}_k$$

$$\hat{y}_k = (1 + \beta)\tilde{x}_k - \beta\tilde{q}_k - \frac{1}{L - \mu}\hat{u}_k$$

The transfer function from $-\hat{u}_k$ to $\hat{y}_k$

$$\begin{aligned} G_{cl}(z) &= \begin{bmatrix} 1 + \beta & -\beta \end{bmatrix} \begin{bmatrix} z - (1 - \alpha\mu)(1 + \beta) & (1 - \alpha\mu)\beta \\ -1 & z \end{bmatrix}^{-1} \begin{bmatrix} \alpha \\ 0 \end{bmatrix} + \frac{1}{L - \mu} \\ &= \alpha \frac{(1 + \beta)z - \beta}{z^2 - (1 - \alpha\mu)(1 + \beta)z + (1 - \alpha\mu)\beta} + \frac{1}{L - \mu} \\ &= \frac{1}{L - \mu} \frac{z^2 - (1 - \alpha L)(1 + \beta)z + (1 - \alpha L)\beta}{z^2 - (1 - \alpha\mu)(1 + \beta)z + (1 - \alpha\mu)\beta} \end{aligned}$$

in negative feedback with a passive map

