Control Methods for Distributed Optimization

Integral control for (seemingly) diverse problems

Prof. Ivano Notarnicola

Dept. of Electrical, Electronic, Information Eng.
Università di Bologna
ivano.notarnicola@unibo.it

Prof. Ruggero Carli

Dept. of Information Engineering
Università di Padova
ruggero.carli@unipd.it

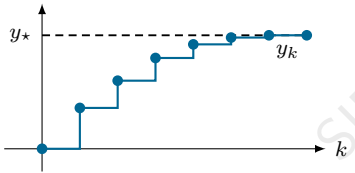
SIDRA Ph.D. Summer School
July, 10-12 2025 • Bertinoro, Italy

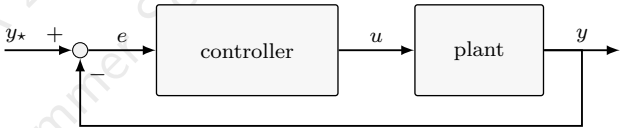
Lecture outline

- Refresher on integral control (in discrete-time and state-space)
- Application to various problems:
  - consensus algorithm
  - gradient method (and its acceleration)
  - (parallel) consensus optimization

Integral control

Integral control is useful to ensuring the ability to generate a control input u in closed-loop that enables the output y to (exactly) track a given *constant reference* $y_\star$, in a robust manner

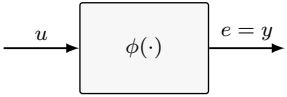




Integral control for algebraic maps

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

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

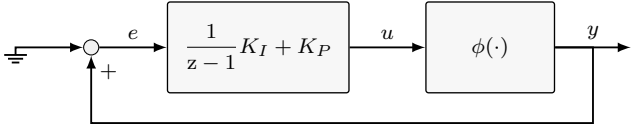


Goal. Steer the error $e = y$ to zero using a (dynamic) feedback controller

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

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

If the interconnection is stable, then $\lim_{k\to\infty} e_k = 0$



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 aperiodic) graph

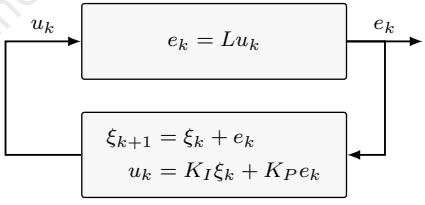
The Laplacian mixing is modeled by the following algebraic (linear) map

$$u \longmapsto e = Lu$$

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

Consider a *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

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

Laplacian (continuous-time) averaging

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

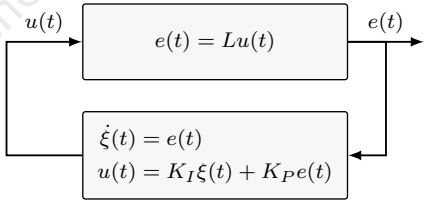
$$u \longmapsto e = Lu$$

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

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



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 $\nabla f(u) = 0_n$

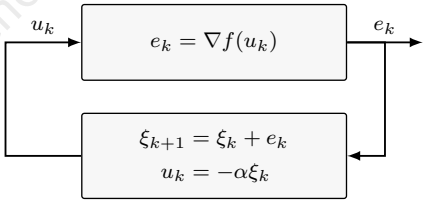
The gradient operator is an algebraic nonlinear map

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

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 $\xi \longmapsto x := -\alpha \xi$, the gradient method is recovered

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

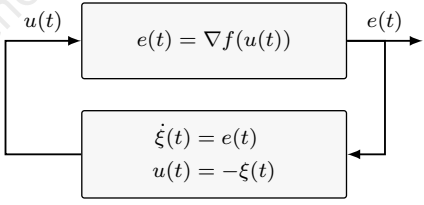
The gradient flow

The gradient operator is an algebraic nonlinear map

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

Consider the continuous-time PI controller

$$\begin{aligned} \dot{\xi}(t) &= e(t) \\ u(t) &= K_I \xi(t) + K_P e(t) \end{aligned}$$



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

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

Changing coordinates via $\xi \longmapsto x := -\alpha\xi$ 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* $\alpha > 0$

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

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

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

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

Changing coordinates via $\xi \longmapsto x \coloneqq -\alpha\xi$ yields

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

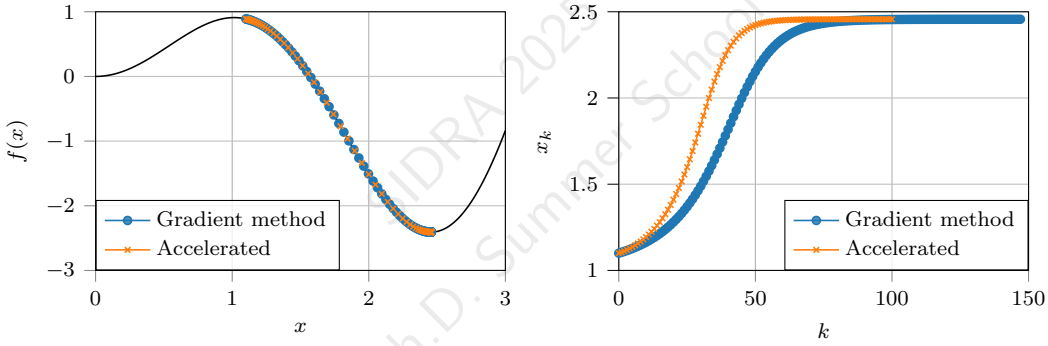
which coincides with the so-called *proximal minimization method*

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

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

Accelerated gradient method

Consider the function $f(x) = x\sin(2x)$ and compare the nominal and the accelerated gradient methods



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 ARMA model

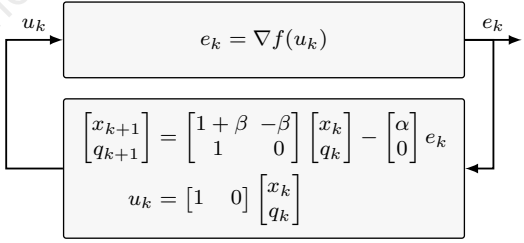
$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1})$$

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

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

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

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



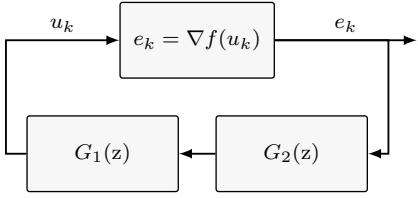
Remark. What about the zero-pole map of the linear subsystem?

The realm of an accelerated gradient method

The linear part of the heavy-ball dynamics has a zero at $\mathrm{z} = 0$ and two poles at $\mathrm{z} = 1$ and $\mathrm{z} = \beta$ (stable)

Therefore, it can always be represented as an integrator (the gradient method) in series with a *lag compensator*

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



Nesterov's method

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

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

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

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

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

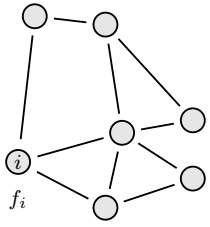
Remark. How does this differ from the heavy-ball method?

Consensus optimization (recall)

A *consensus optimization* problem is

$$\min_{\mathrm{x}\in\mathbb{R}} \sum_{i=1}^{N} f_i(\mathrm{x})$$

where each $f_i : \mathbb{R} \to \mathbb{R}$ is strongly convex and has Lipschitz continuous gradient



Let $x = (x_1, \ldots, x_N) \in \mathbb{R}^N$ and define $f : \mathbb{R}^N \to \mathbb{R}$ as

$$f(x) \coloneqq \sum_{i=1}^{N} f_i(x_i)$$

The optimal solution $x_\star = \mathbf{1}\mathrm{x}_\star$, with $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^N$, must satisfy

$$\begin{aligned} &\mathbf{1}^\top \nabla f(x_\star) = 0 \\ &(I_N - J)x_\star = 0_N \end{aligned} \qquad \Longleftrightarrow \qquad \exists \lambda_\star \in \mathbb{R}^N \text{ s.t. } \nabla f(x_\star) + (I - J)\lambda_\star = 0_N$$

where $J \coloneqq \frac{1}{N}\mathbf{1}\mathbf{1}^\top$

Parallel implementation of the gradient method for consensus optimization

The consensus optimization problem can be expressed as

$$\min_{\mathrm{x}\in\mathbb{R}} \sum_{i=1}^{N} f_i(\mathrm{x}) \qquad \Longleftrightarrow \qquad \begin{array}{l} \min\limits_{x\in\mathbb{R}^N} f(x) \\ \text{subj. to } (I-J)x = 0_N \end{array}$$

The *KKT operator* ($u^1 \in \mathbb{R}^N$ is the primal variable, $u^2 \in \mathbb{R}^N$ is the Lagrange multiplier) is given by

$$u = \begin{bmatrix} u^1 \\ u^2 \end{bmatrix} \longmapsto e = \begin{bmatrix} e^1 \\ e^2 \end{bmatrix} = \begin{bmatrix} \nabla f(u^1) + (I-J)u^2 \\ (I-J)u^1 \end{bmatrix} =: \phi(u)$$

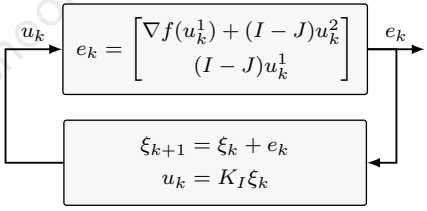
Remark. If $e^1 = 0_N$, then it must be $\mathbf{1}^\top \nabla f(u^1) = 0$

Solving the constrained optimization problem amounts to computing u such that $e = \phi(u) = 0_{2N}$

Parallel gradient algorithm for consensus optimization

Applying a pure integral action, with gain $K_I := -\begin{bmatrix} \alpha I_N & \\ & \beta I_{N} \end{bmatrix}$, to the KKT operator yields

$$\xi_{k+1} = \xi_k + \begin{bmatrix} \nabla f(-\alpha\xi_k^1) - \beta(I-J)\xi_k^2 \\ -\alpha(I-J)\xi_k^1 \end{bmatrix}$$



Changing the coordinates as $\begin{bmatrix} \xi^1 \\ \xi^2 \end{bmatrix} \longmapsto \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} := \begin{bmatrix} -\alpha\xi^1 \\ -\beta\xi^2 \end{bmatrix}$, yields

$$x_{k+1}^1 = x_k^1 - \alpha\nabla f(x_k^1) - \alpha(I-J)x_k^2$$
$$x_{k+1}^2 = x_k^2 - \beta(I-J)x_k^1$$

Remark. In optimization, it is called the a *primal-dual algorithm* with a Lagrangian function given by

$$L(x,\lambda) := f(x) + \lambda^\top(I-J)x$$

Remark. It is a *parallel algorithm*: the local gradient steps are fully decentralized, while the terms involving the operator J require a centralized node to compute the averages