The speed–robustness trade-off for first-order methods with additive gradient noise

Bryan Van Scoy
Miami University

Laurent Lessard
Northeastern University

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} \quad f(x)$$

In this talk:

- Iterative algorithms can be viewed as robust controllers.
- Algorithms can be designed, in much the same way that controllers can be designed.
- Controls and optimization!

$$x^\star \in \arg\min_{x\in\mathbb{R}^d} f(x)$$

Noisy oracle: $g(x) = \nabla f(x) + w$

- w is zero-mean and independent across queries
- $\mathbb{E}(ww^\mathsf{T}) \leq \sigma^2 I_d$ for some known σ

Use cases:

- perturb gradient for privacy
- gradient only available through noisy measurements
- risk minimization; minimize expected loss over population distribution

Gradient Descent (GD)

$$x_{k+1} = x_k - \alpha\, g(x_k)$$

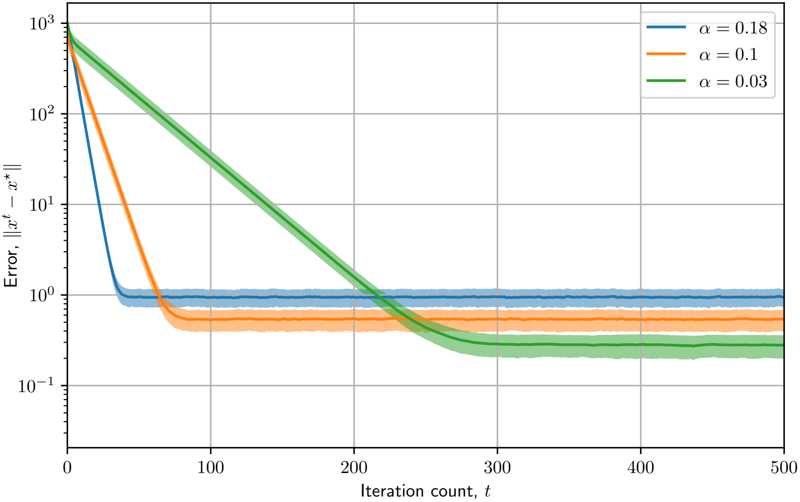
Geometric phase

- Noise is small compared to gradient
- x_k makes rapid progress toward $x^\star$

Stationary phase

- Noise is comparable to gradient
- x_k moves randomly in a ball about $x^\star$

Random quadratic function: $f(x) = \frac{1}{2}x^\mathsf{T}Qx$, $d = 10$, $\sigma = 1$
Eigenvalues satisfy $1 \le \lambda(Q) \le 10$



Acceleration

Polyak acceleration (Heavy Ball)

$$x_{k+1} = x_k - \alpha\, g(x_k) + \beta(x_k - x_{k-1})$$

Nesterov acceleration (Fast Gradient)

$$\begin{aligned} y_k &= x_k + \beta(x_k - x_{k-1}) \\ x_{k+1} &= y_k - \alpha\, g(y_k) \end{aligned}$$

- Similar geometric & stationary phases
- More parameters to tune
- Potentially better performance!

Performance metrics

Rate of convergence (ρ)

$$\|x_k - x^\star\| \le (\text{const}) \cdot \rho^k$$

Smaller ρ means faster convergence (no noise regime).

Sensitivity to noise (γ)

$$\gamma = \limsup_{N\to\infty} \sqrt{\frac{1}{N}\sum_{k=0}^{N-1} \mathbb{E}\,\|x_k - x^\star\|^2}$$

Smaller γ means more noise robustness (smaller ball).

Questions

How can we compute the rate of convergence and sensitivity to noise for a given algorithm?

Can we design algorithms that are Pareto-optimal for different function classes? What will they look like?

Outline

- Algorithms as dynamical systems
- Quadratic functions
- Smooth strongly convex functions
- Polyak–Łojasiewicz and smooth functions

3-parameter family (α, β, η)

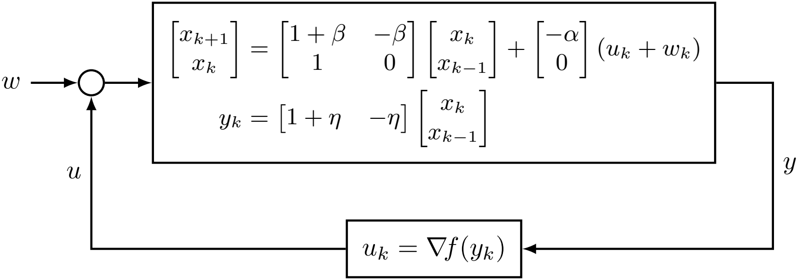
$$x_{k+1} = x_k - \alpha\, g\big(x_k + \eta(x_k - x_{k-1})\big) + \beta(x_k - x_{k-1})$$

Special cases:

- recovers Gradient descent when $\beta = 0$ and $\eta = 0$
- recovers Polyak acceleration when $\eta = 0$
- recovers Nesterov acceleration when $\beta = \eta$

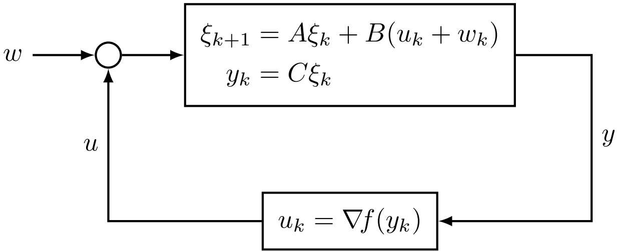
Dynamical system interpretation

$$x_{k+1} = x_k - \alpha\, g\big(x_k + \eta(x_k - x_{k-1})\big) + \beta(x_k - x_{k-1})$$



Dynamical system interpretation

$$x_{k+1} = x_k - \alpha\, g\big(x_k + \eta(x_k - x_{k-1})\big) + \beta(x_k - x_{k-1})$$



- Analysis applies to general algorithms (A, B, C)
- Design 3-parameter algorithms (α, β, η)

Quadratic functions

- functions of the form $f(x) = \frac{1}{2}(x - x^\star)^\mathsf{T} Q (x - x^\star)$
- each eigenvalue of Q is in the closed interval $[m, L]$

Heavy Ball (HB) achieves fastest possible rate when used with tuning

$$\alpha = \frac{4}{(\sqrt{L}+\sqrt{m})^2} \qquad \beta = \left(\frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}\right)^2 \qquad \eta = 0$$

$$
\begin{aligned}
\xi_{k+1} &= A\xi_k + B(u_k + w_k) \\
y_k &= C\xi_k \\
u_k &= \nabla f(y_k)
\end{aligned}
$$

Closed-loop map:

$$
\xi_{k+1} = (A + BQC)\xi_k + Bw_k
$$

- the rate of convergence is the spectral radius of $A + BQC$
- the sensitivity to noise is the $\mathcal{H}_2$-norm of the system

Quadratic performance

- **Rate:**

$$\rho = \sup_{q \in [m,L]} \rho(A + qBC)$$

- **Sensitivity:** if $\rho < 1$, then

$$\gamma = \sigma\sqrt{d} \sup_{q \in [m,L]} \sqrt{B^\mathsf{T} P_q B}$$

where P_q is the solution to the matrix equation

$$(A + qBC)^\mathsf{T} P_q (A + qBC) - P_q + C^\mathsf{T} C = 0$$

Both $\rho(A + qBC)$ and P_q are nonconvex functions of q in general.

Quadratic performance of 3-parameter algorithms

- **Rate:**

$$\rho = \max_{q \in \{m,L\}} \begin{cases} \sqrt{\beta - \alpha\eta q} & \text{if } \Delta < 0 \\ \frac{1}{2}\left(|\beta + 1 - \alpha q - \alpha\eta q| + \sqrt{\Delta}\right) & \text{if } \Delta \geq 0 \end{cases}$$

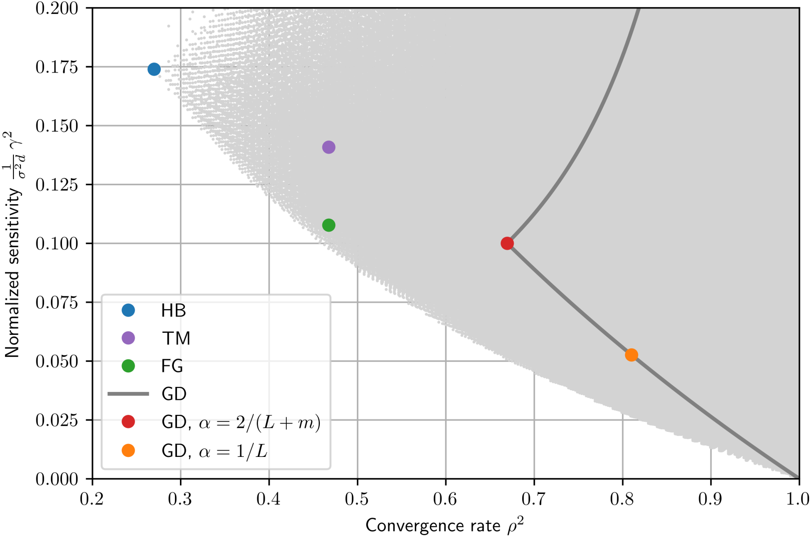
$$\text{where } \Delta := (\beta + 1 - \alpha q - \alpha\eta q)^2 - 4(\beta - \alpha\eta q)$$

- **Sensitivity:** if $\rho < 1$, then

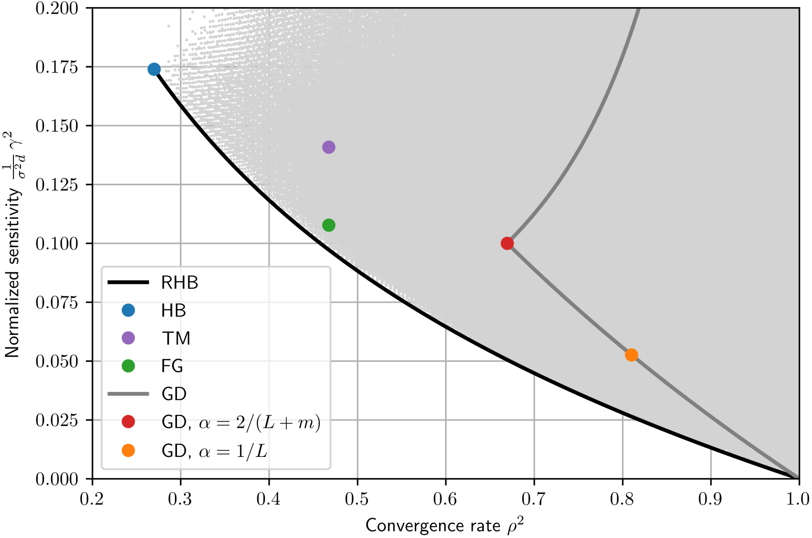
$$\gamma = \sigma\sqrt{d} \max_{q \in \{m,L\}} \sqrt{\frac{\alpha(1 + \beta + (1 + 2\eta)\alpha\eta q)}{q(1 - \beta + \alpha\eta q)(2 + 2\beta - (1 + 2\eta)\alpha q)}}$$

Both are easy to evaluate and analyze!

(ρ, γ) tradeoff for quadratics with $m = 1$ and $L = 10$



(ρ, γ) tradeoff for quadratics with $m = 1$ and $L = 10$



Robust Heavy Ball (RHB)

RHB is the 3-parameter algorithm parameterized by $r \in \left[\frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}, 1\right)$

$$\alpha = \tfrac{1}{m}(1-r)^2 \qquad \beta = r^2 \qquad \eta = 0$$

Setting $r = \frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}$ recovers ordinary Heavy Ball.

The parameter r is the convergence rate on quadratics and the sensitivity is

$$\gamma \quad = \quad \frac{\sigma\sqrt{d}}{m}\sqrt{\frac{1-r^4}{(1+r)^4}}$$

RHB appears to be Pareto-optimal (no formal proof).

Smooth and strongly convex functions

Differentiable functions for which:

a) $f(y) - \frac{m}{2}\|y\|^2$ is a convex function of y

b) $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ for all $x, y \in \mathbb{R}^d$

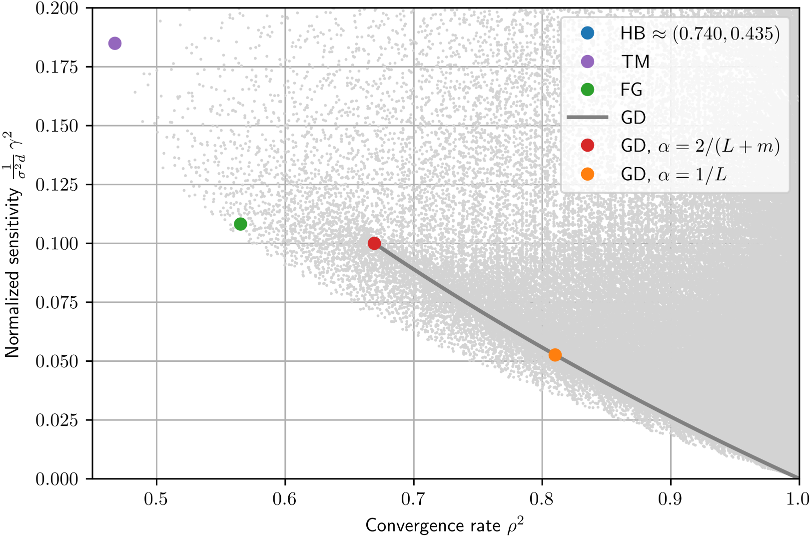
- **Triple Momentum** (TM) achieves fastest possible rate

$$\alpha = \frac{\sqrt{L}-\sqrt{m}}{L^{3/2}} \qquad \beta = \frac{(\sqrt{L}-\sqrt{m})^2}{L+\sqrt{mL}} \qquad \eta = \frac{(\sqrt{L}-\sqrt{m})^2}{2L-m+\sqrt{mL}}$$

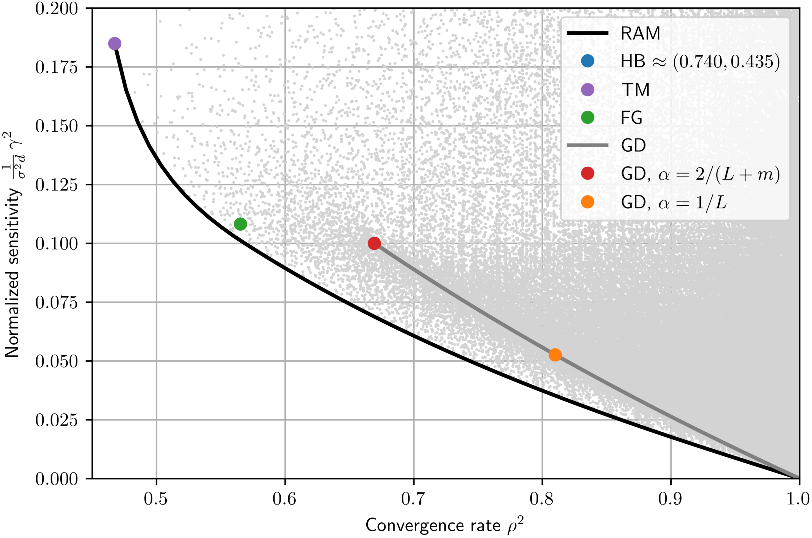
- **Fast Gradient** (FG) is a popular choice

$$\alpha = \frac{1}{L} \qquad \beta = \frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}} \qquad \eta = \frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}$$

(ρ, γ) tradeoff for strongly convex functions with $m = 1$ and $L = 10$



(ρ, γ) tradeoff for strongly convex functions with $m = 1$ and $L = 10$



Robust Accelerated Method (RAM)

RAM is the 3-parameter algorithm parameterized by $r \in \left[1 - \sqrt{\frac{m}{L}}, 1\right)$

$$\alpha = \frac{(1+r)(1-r)^2}{m} \qquad \beta = r\,\frac{L\,(1-r+2r^2) - m\,(1+r)}{(L-m)(3-r)}$$
$$\eta = r\,\frac{L\,(1-r^2) - m\,(1+2r-r^2)}{(L-m)(3-r)(1-r^2)}$$

Setting $r = 1 - \sqrt{m/L}$ recovers Triple Momentum.

The parameter r is the rate of convergence on strongly convex functions.

RAM appears to be *nearly* Pareto-optimal (no expression for γ).

Polyak–Łojasiewicz (PL) functions

Differentiable functions for which:

a) $\frac{1}{2}\|\nabla f(x)\|^2 \geq m\,(f(x) - f^\star)$ for all $x \in \mathbb{R}^d$

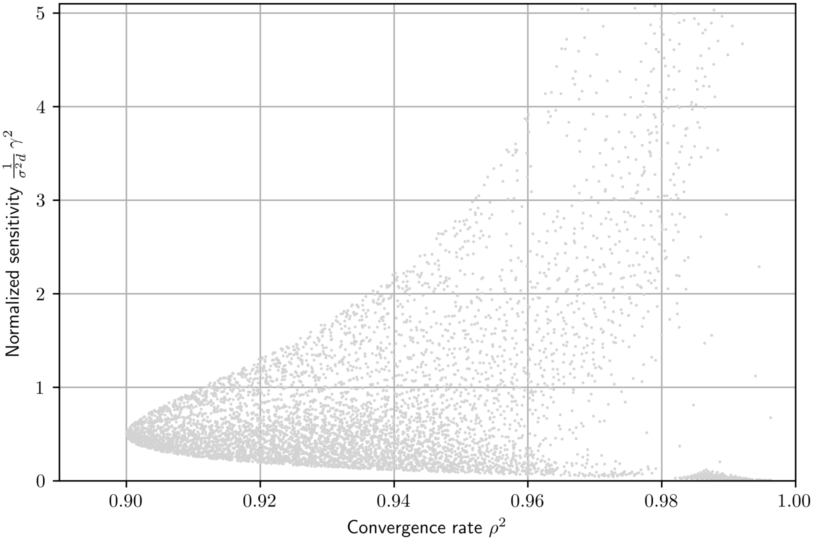
b) $f(y) \leq f(x) + \nabla f(x)^\mathsf{T}(y - x) + \frac{L}{2}\|y - x\|^2$ for all $x, y \in \mathbb{R}^d$

Gradient Descent (GD) converges when there is no noise.

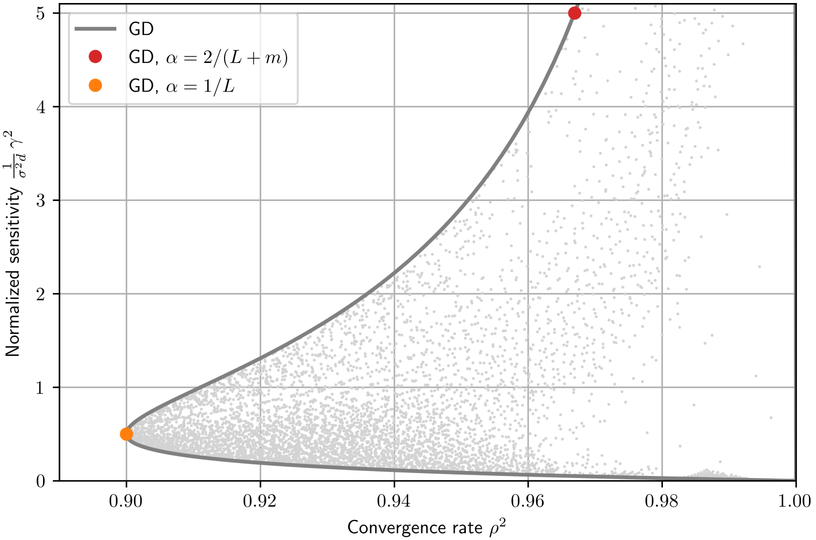
$$\alpha = \frac{1}{L} \qquad \rho = \sqrt{1 - \frac{m}{L}}$$

(Karimi, Nutini, Schmidt. 2016)

(ρ, γ) tradeoff for PL functions with $m = 1$ and $L = 10$

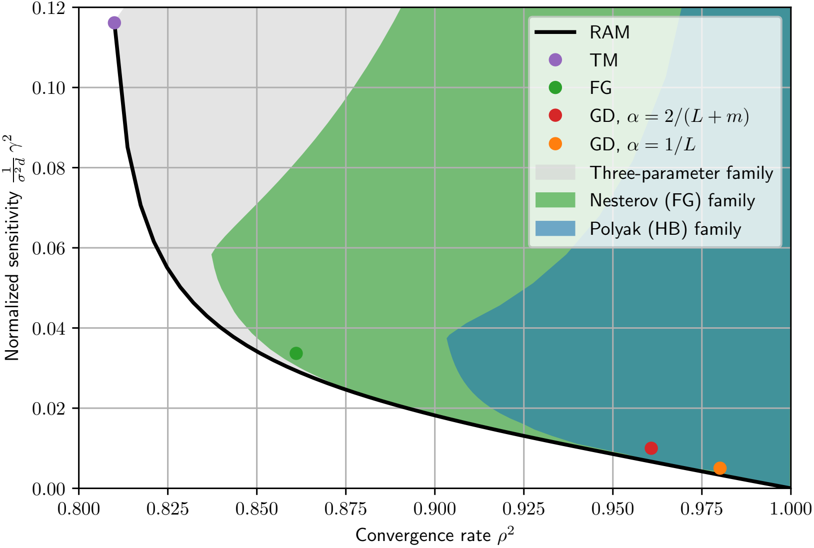


(ρ, γ) tradeoff for PL functions with $m = 1$ and $L = 10$



Our algorithms use all three parameters (α, β, η). What if we use only Polyak or only Nesterov acceleration?

Nesterov and Polyak coverage for strongly convex with $m = 1$ and $L = 100$



Analysis techniques

How do we analyze other function classes?

Issue: cannot parameterize other function classes (e.g., strongly convex)

Lyapunov approach

- search for functions whose existence provides upper bounds on ρ and γ
- use interpolation conditions to list valid inequalities
- use S-lemma to formulate as a semidefinite program
- use lifting technique to tighten bounds

Design challenges

- Not as straightforward as quadratic case because we do not have an explicit function $(\alpha, \beta, \eta) \mapsto (\rho, \gamma)$.
- In principle, solution is a *semialgebraic set*.
- Optimality conditions yield polynomials of degree > 200 that are not solvable analytically.

Challenge is to find algorithms that:

- Have relatively simple algebraic expressions. Avoid numerical solutions if possible.
- Are as close to being optimal as possible.

General strategy

a) Use numerical solver (e.g. Nelder–Mead) to find locally optimal (α, β, η), e.g. fix ρ and minimize γ.

b) Write LMI as polynomial optimization problem: convert semidefinite constraints into determinant inequalities.

c) Substitute numerical solution to find active constraints and dual variables. At optimality, matrices in LMI will drop rank.

d) Look for analytic solution to system of active constraints. Might require trying different elimination orderings.

Thank you!

- Preprint available: https://arxiv.org/abs/2109.05059
- Slides available: https://vanscoy.github.io
- Funding acknowledgement: NSF 1750162, 1936648