Efficient Natural Gradient Descent Methods for Large-Scale PDE-Based Optimization Problems

Based on Levon Nurbekyan, Wanzhou Lei, and Yunan Yang

SIAM J. Sci. Comput., 2023

Background

- PDE-constrained optimization arises in inverse problems and ML-based PDE solvers.

$$\min_{\theta} \; f(\rho(\theta)), \quad \text{s.t.} \;\; \mathcal{F}(\rho(\theta), \theta) = 0 \;\; \text{(PDE model)}$$

- Notation:
  - θ: parameter vector (e.g., discretized coefficients, NN weights).
  - $\rho(\theta)$: state variable determined by PDE model $\mathcal{F}$.
  - $f(\rho(\theta))$: objective/loss measuring mismatch between model output and observed data.

Background

- Standard Gradient Descent (GD):

$$\dot{\theta} = -\nabla_\theta f(\rho(\theta))$$

  Efficient but can be slow and easily trapped in local minima.

- Natural Gradient Descent (NGD):

$$\dot{\theta} = -G(\theta)^{-1}\nabla_\theta f(\rho(\theta)), \qquad G_{ij}(\theta) = \left\langle \partial^g_{\theta_i}\rho,\ \partial^g_{\theta_j}\rho \right\rangle_g$$

  Performs steepest descent in the state manifold $(\mathcal{M}, g)$, but computing $G(\theta)^{-1}$ is expensive in large-scale PDEs.

- Goal: Design efficient NGD schemes that scale to high-dimensional PDE optimization.

Motivation / Challenges

- **Problem:** Direct NGD is infeasible in PDE optimization (huge $G(\theta)$, no explicit inverse).
- **Observation:** NGD can be reformulated as a *least-squares problem*:

$$\eta^{nat} = \arg\min_{\eta} \left\| \nabla_\rho^g f + J\eta \right\|_g^2, \quad J = [\partial_{\theta_1}^g \rho, \ldots, \partial_{\theta_p}^g \rho].$$

- **Derivation (sketch):**
  - By chain rule: $\nabla_\theta f = J^\top \nabla_\rho^g f$.
  - Metric pullback: $G(\theta) = J^\top J$.
  - NGD system: $G(\theta)\eta = -\nabla_\theta f \iff$ normal equations of the LS problem above.
- **Key Challenge:** Efficiently solving this LS system at scale.

How to Solve the LS Problem

Recall: NGD reduces to solving the least-squares system

$$\eta^{nat} = \arg\min_{\eta} \|\nabla_\rho^g f + J\eta\|_g^2, \quad J = [\partial_{\theta_1}^g \rho, \ldots, \partial_{\theta_p}^g \rho].$$

- **Case 1: J explicit (moderate size).**
  - J is a matrix we can store explicitly.
  - Classical linear algebra tools apply: QR, SVD.
  - Low-rank structure $\Rightarrow$ efficient approximation.
- **Case 2: J implicit (PDE constraints).**
  - In PDE problems, parameter dimension can be 10^5-10^9.
  - J is too large to form or store explicitly.
  - What we *can* compute efficiently:
    - Jv: perturb parameter θ in direction v, solve PDE $\Rightarrow$ state variation.
    - $J^\top v$: via **adjoint-state method**, one additional PDE solve.
  - Iterative solvers (CG, LSQR) only require Jv and $J^\top v$, so they are the natural choice.

Algorithms for Computing NGD

NGD Algorithms (Explicit vs Implicit Jacobian)

Goal: Compute NGD direction η_L^{nat} in cases 1 and 2.

Algorithm 3.1 (Explicit J)

1. Compute $Y = LJ$.
2. Perform QR factorization: $[Q, R] = \mathrm{qr}(Y)$.
3. Compute NGD direction

$$\eta_L^{nat} = -R^{-1} Q^\top (L^\top)^\dagger \, \partial_\rho f.$$

Algorithm 3.2 (Operator form of G_L)

1. Given constraint h, solve $\partial_\rho h \, \gamma = -\partial_\theta h \, \eta$.
2. Solve $\partial_\rho h^\top \lambda = L^\top L \gamma$.
3. Evaluate $-\partial_\theta h^\top \lambda = G_L \eta$.

Algorithm 3.3 (Implicit J, PDE-scale)

1. Solve $(\partial_\rho h)^\top \lambda = \partial_\rho f$, obtain λ.
2. Compute parameter gradient $\partial_\theta f = -\partial_\theta h^\top \lambda$.
3. Obtain operator action $G_L \eta$ via Algorithm 3.2.
4. Solve $G_L \eta_L^{nat} = -\partial_\theta f$ using Conjugate Gradient.

NGD under Different Metrics

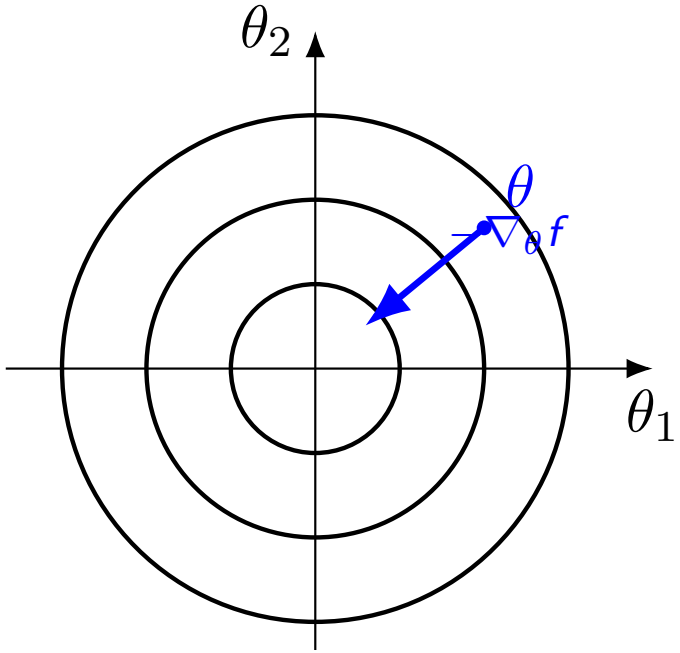
Metric	Formula (gradient in ρ-space)
L^2	$\nabla_\rho^g f = \nabla_\rho f$
Sobolev H^s	$\nabla_\rho^g f = (I - \Delta)^s \nabla_\rho f$
Fisher–Rao	$\nabla_\rho^g f = \rho \nabla_\rho f$
Wasserstein W_2	$\nabla_\rho^g f = \nabla \nabla_\rho f$

- L^2: standard Euclidean gradient.
- Sobolev H^s: smooths the update, reduces oscillations.
- Fisher–Rao: scale-invariant, common in statistics/ML.
- Wasserstein W_2: linked to optimal transport, mass movement.

Takeaway: The LS reformulation applies to all metrics, making NGD a unified framework across different geometries.

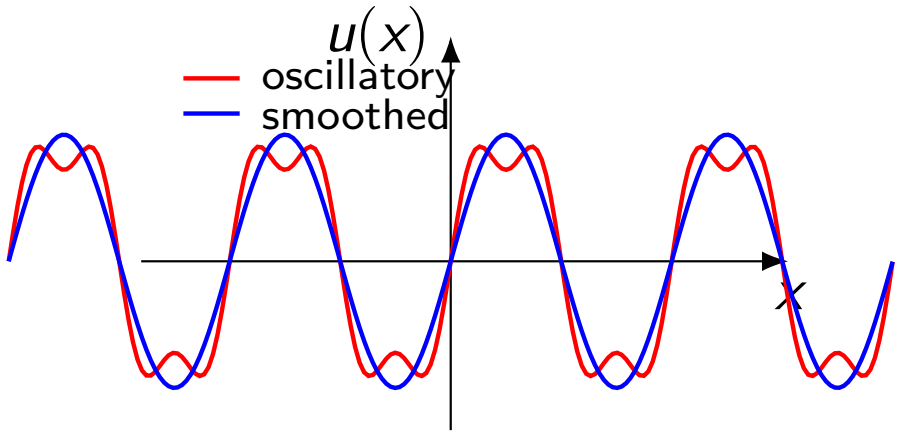
Geometric Intuition of Metrics

L^2 (Euclidean)



Level sets are circles (flat geometry). Steepest descent is the usual Euclidean negative gradient. LS uses standard $\|\cdot\|_2$ (Frobenius) norm.

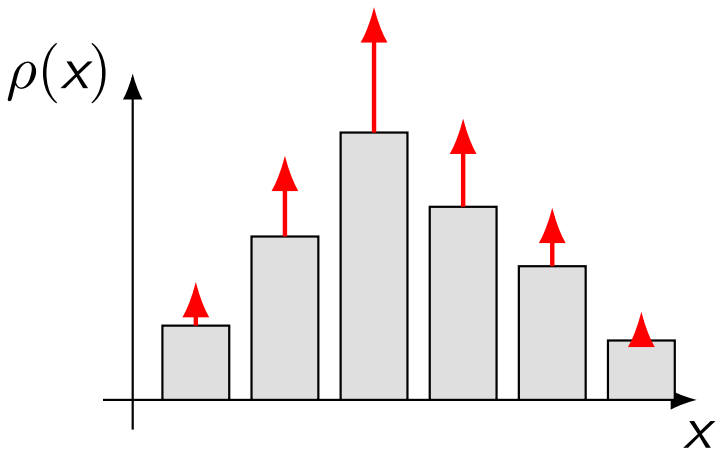
Sobolev H^s



H^s metric weights high frequencies via $(I-\Delta)^s$: updates are smoothed, penalizing oscillations in the state.

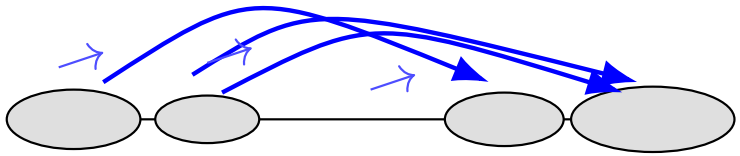
Geometric Intuition of Metrics

Fisher–Rao



Inner product $\langle u, v\rangle_{\mathrm{FR}} = \int \frac{uv}{\rho}\, dx \Rightarrow$ effective gradient $\nabla^g_\rho f = \rho\, \nabla_\rho f$: regions with larger ρ are weighted more (scale-invariant).

Wasserstein W_2



Geometry of mass transport: tangent vectors are velocity fields v with $-\nabla\cdot(\rho v) = \zeta$. The Riemannian gradient takes the form $\nabla^g_\rho f = \nabla(\nabla_\rho f)$; updates move mass along v.

Extensions: Damped NGD

- **Problem:** Information matrix G_L can be
  - rank-deficient or ill-conditioned,
  - leading to unstable or extreme NGD updates.

- **Solution:** Add damping for stability:

$$G_\lambda = \lambda I + G_L, \quad \lambda > 0.$$

  - Ensures positive definiteness.
  - Avoids extreme updates, improves numerical robustness.

- **Connection:** Levenberg–Marquardt method = damped Gauss–Newton = L^2 NGD in this framework.
- **Update rule:** In mixed (θ, ρ) metric space:

$$\theta^{l+1} = \arg\min_\theta \left\{ f(\rho(\theta)) + \frac{\lambda d_\theta(\theta,\theta^l)^2 + d_\rho(\rho(\theta),\rho(\theta^l))^2}{2\tau} \right\}.$$

Extensions: Damped NGD

- **Generalization:** Use another ρ-space metric for regularization instead of θ-space.

$$\theta^{l+1} = \arg\min_{\theta} \left\{ f(\rho(\theta)) + \frac{\lambda d_{\rho_1}(\rho(\theta),\rho(\theta^l))^2 + d_{\rho_2}(\rho(\theta),\rho(\theta^l))^2}{2\tau} \right\}.$$

- Here:
  - d_{ρ_2} = main natural gradient metric,
  - d_{ρ_1} = regularizing metric.

- **Interpretation:** H^1 NGD damped by L^2 NGD (regularization strength set by λ).
- **Takeaway:** Damping provides a flexible stabilization mechanism, unifying NGD with classical methods such as Gauss–Newton and Levenberg–Marquardt.

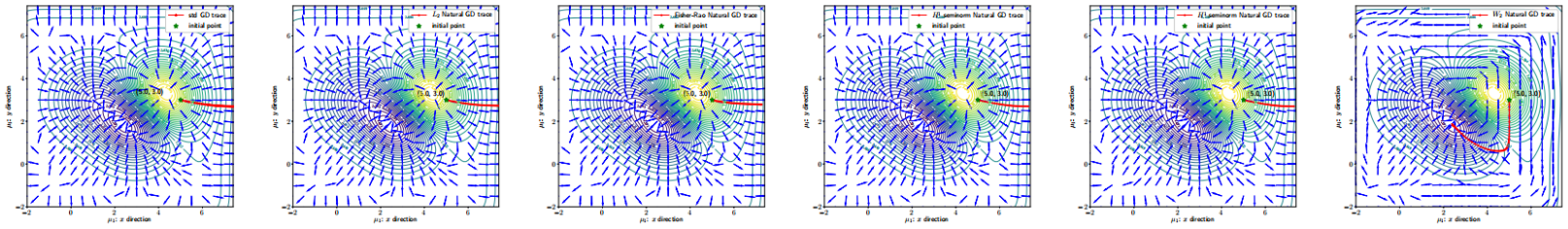
Experimental Setup

- **Tasks:** PDE-constrained optimization problems
  - Gaussian mixture inversion
  - Physics-Informed Neural Networks (PINNs)
  - Full Waveform Inversion (FWI)
- **Methods compared:**
  - Gradient Descent (GD)
  - Natural Gradient Descent (NGD) with metrics:

$$L^2, \quad H^1, \quad H^{-1}, \quad \text{Fisher-Rao (FR)}, \quad W_2$$

- **Evaluation criteria:**
  - Convergence path and trajectory (visualization)
  - Loss decay vs iterations and wall-clock time
  - Reconstruction quality (SSIM for FWI, error vs ground truth)
- **Implementation:** NGD directions computed via LS formulation with explicit/implicit Jacobian.

Gaussian Mixture Example

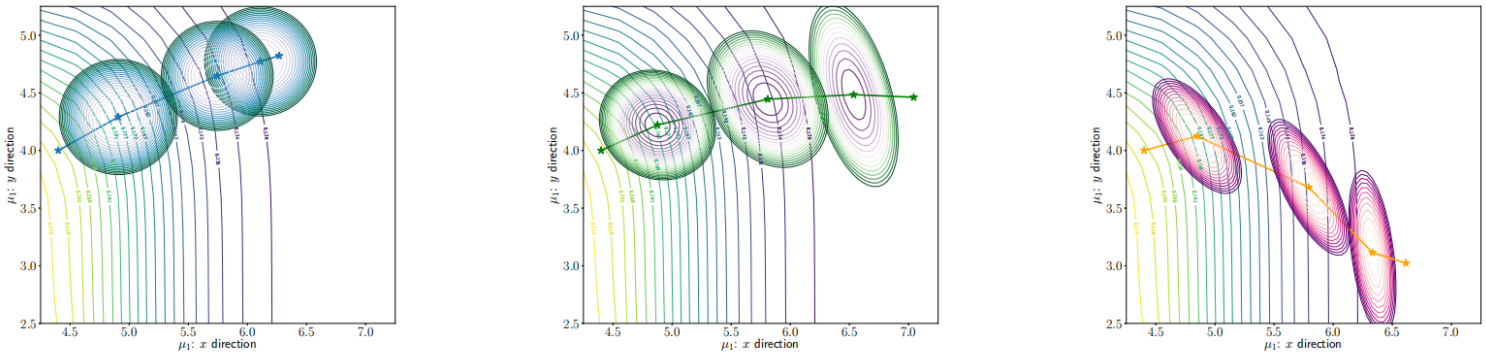


(a) GD (b) L^2 NGD (c) FR NGD (d) H^1 NGD (e) H^{-1} NGD (f) W_2 NGD

FIG. 2. *Gaussian mixture example: Level sets, vector fields, and convergent paths using GD and different NGD methods to invert* μ_1. *All algorithms start from initial guess* $(5, 3)$.

- **Observation:**
  - GD: slow convergence, oscillatory path.
  - NGD: faster and smoother trajectories.
  - Different metrics yield different geometry of descent:
    - L^2: standard Euclidean updates.
    - FR: scale-invariant steps.
    - H^1: smooths oscillations in updates.
    - H^{-1}: emphasizes large-scale structure.
    - W_2: mass-transport interpretation, very different path.

Local Quadratic Models

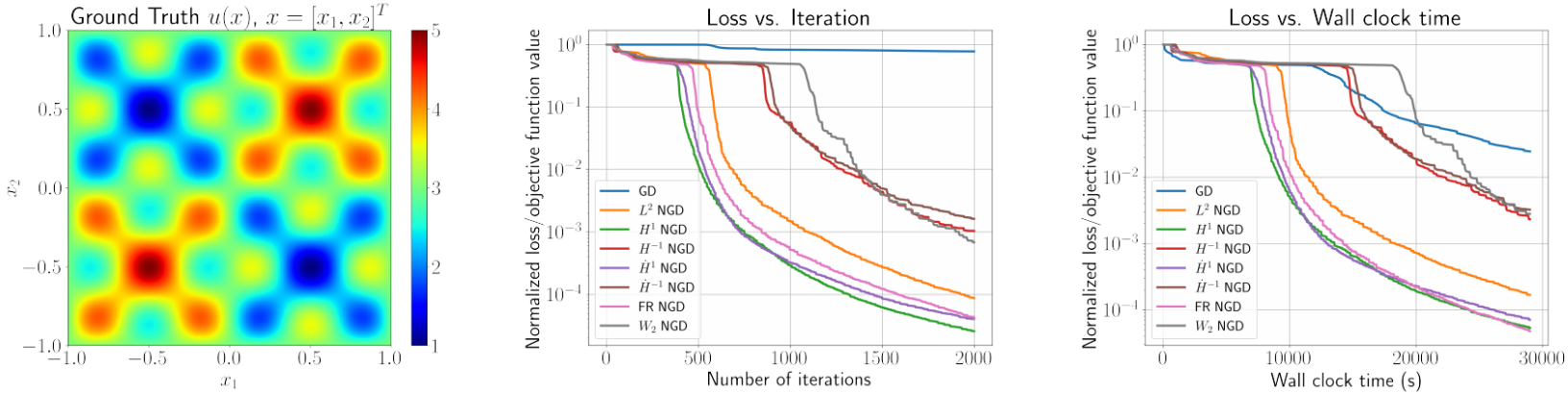


(a) Standard gradient descent (b) L^2 natural gradient (c) W_2 natural gradient

FIG. 3. *The local quadratic models of GD, L^2 NGD, and W_2 NGD in the first several iterations.*

- Compare GD, L^2 NGD, and W_2 NGD in the first iterations.
- **GD:** isotropic quadratic model, ignores geometry.
- L^2 **NGD:** improves conditioning, balanced ellipses.
- W_2 **NGD:** anisotropic ellipses capturing transport geometry.
- **Takeaway:** NGD changes local landscape, leading to better-conditioned descent directions.

PINN Results

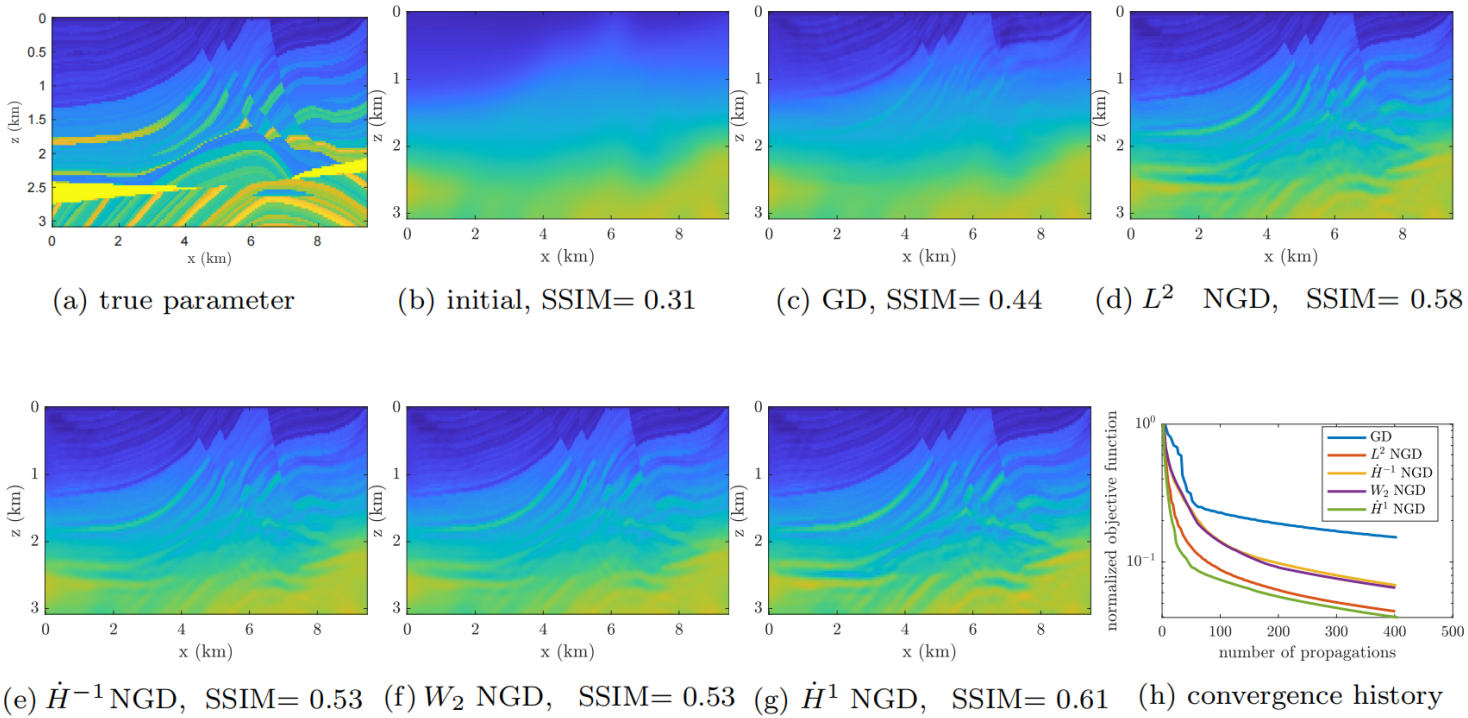


(a) True solution (b) Loss decay vs. iteration number (c) Loss decay vs. wall clock time

FIG. 4. (a) *PINN example true solution;* (b) *loss function value decay in terms of the number of iterations;* (c) *loss function value decay in terms of the wall clock time.*

- Physics-Informed Neural Network (PINN) example.
- **Observations:**
  - NGD accelerates convergence significantly compared to GD.
  - Sobolev H^1 and Wasserstein W_2 metrics yield smoother, more stable decay.
  - Consistent improvements both in iteration count and actual runtime.

Full Waveform Inversion (FWI) Results



(a) true parameter (b) initial, SSIM= 0.31 (c) GD, SSIM= 0.44 (d) L^2 NGD, SSIM= 0.58

(e) $\dot{H}^{-1}$ NGD, SSIM= 0.53 (f) W_2 NGD, SSIM= 0.53 (g) $\dot{H}^1$ NGD, SSIM= 0.61 (h) convergence history

FIG. 5. *FWI example:* (a) *ground truth;* (b) *initial guess;* (c)–(g) *inversion results using GD and NGDs based on the* L^2, $\dot{H}^{-1}$, W_2, *and* $\dot{H}^1$ *metrics after* 400 *PDE solves;* (h) *the history of the objective function decay versus the number of propagations/PDE solves. SSIM denotes the structural similarity index measure compared with* (a). *A bigger value means better similarity.*

- **Observations:**
  - NGD methods yield reconstructions closer to ground truth.
  - H^1 NGD achieves best similarity (SSIM = 0.61).
  - Convergence: NGDs consistently faster than GD.