Documentation and Mathematical Foundations of the minpy Optimization Framework

Nadia Udler

October 7, 2025

1 Overview

The minpy framework implements a family of nonlocal optimization algorithms based on *potential theory*, following the ideas in Kaplinskii & Propoi ("Nonlocal Optimization Methods Based on Potential Theory"). It includes:

- A **potential-based first-order method**, which constructs a potential field from sampled points and moves particles along the gradient of this potential.
- A **second-order potential method**, which uses local curvature (Hessian) information to accelerate convergence.
- Classical methods (particle swarm, Nelder–Mead, etc.) available in the same interface for comparison.

Each method can handle multimodal and nonconvex objectives by combining global exploration and local refinement.

2 Mathematical Foundation

2.1 Potential Function Formulation

Given an objective function $f : \mathbb{R}^d \to \mathbb{R}$, we maintain a set of sample points ("particles")

$$U = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_K\}, \qquad f_i = f(\mathbf{u}_i).$$

From these samples, we construct a potential function

$$\Phi(\mathbf{x}) = \sum_{j=1}^{K} w_j \, G(\|\mathbf{x} - \mathbf{u}_j\|; \varepsilon)\,,$$

where:

- $G(r; \varepsilon)$ is a *radial kernel* that smooths the field,

- w_j are weights determined by the quality of the sample,
- $\varepsilon > 0$ controls the range of influence (annealed over iterations).

Typical choices for G include:

$$G_{\text{inv-power}}(r) = \frac{1}{(r^p + \varepsilon)}, \tag{1}$$

$$G_{\text{inv-mq}}(r) = \frac{1}{\sqrt{r^2 + \varepsilon}}, \tag{2}$$

$$G_{\text{gauss}}(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right). \tag{3}$$

2.2 Potential Gradient and Particle Movement

The "force" acting on a particle at $\mathbf{u}_i$ is given by the negative gradient of the potential:

$$\mathbf{F}_i = -\nabla_{\mathbf{x}}\Phi(\mathbf{x})\big|_{\mathbf{x}=\mathbf{u}_i} = -\sum_{j=1}^{K} w_j\, G'(r_{ij})\, \frac{\mathbf{u}_i - \mathbf{u}_j}{r_{ij}},$$

where $r_{ij} = \|\mathbf{u}_i - \mathbf{u}_j\|$ and $G'(r)$ is the radial derivative of the kernel.

The particles move along this field:

$$\mathbf{u}_i^{(t+1)} = \mathbf{u}_i^{(t)} + \eta_t\, \mathbf{F}_i^{(t)},$$

where η_t is a step size controlling the motion amplitude.

2.3 Weight Adaptation

Weights w_j emphasize points with lower function values:

$$w_j = \frac{\exp(-\beta_t(f_j - f_{\min}))}{\sum_{k=1}^{K} \exp(-\beta_t(f_k - f_{\min}))}.$$

Here β_t acts as an inverse temperature parameter that increases over iterations (annealing), focusing attraction toward promising regions.

2.4 Nonlocal–Local Transition

The potential parameter ε_t is decreased according to an annealing schedule:

$$\varepsilon_{t+1} = \rho_\varepsilon\, \varepsilon_t, \qquad 0 < \rho_\varepsilon < 1,$$

which gradually sharpens the potential field from a smooth global surface to a fine local structure.

3 Algorithm 1: Potential-Based Nonlocal Minimization

This algorithm is implemented in `minimize_potential_nonlocal()`.

Algorithm 1 Nonlocal Potential Optimization (First-Order)

```
 1: Initialize $K$ particles $\mathbf{u}_i$ within bounds $[\mathbf{l}, \mathbf{u}]$.
 2: Evaluate $f_i = f(\mathbf{u}_i)$.
 3: Set $\varepsilon \leftarrow \varepsilon_0$, $\beta \leftarrow \beta_0$.
 4: for epoch $t = 1, \dots, T$ do
 5:     Compute weights $w_j$ from $f_j$.
 6:     For each particle $i$, compute force
            $$\mathbf{F}_i = -\sum_j w_j G'(r_{ij}) \frac{\mathbf{u}_i - \mathbf{u}_j}{r_{ij}}.$$
 7:     Update positions $\mathbf{u}_i \leftarrow \mathbf{u}_i + \eta \mathbf{F}_i$.
 8:     Apply bounds and recompute $f_i$.
 9:     Anneal parameters: $\varepsilon \leftarrow \rho_\varepsilon \varepsilon, \quad \beta \leftarrow \rho_\beta \beta$.
10: end for
11: Return best particle $\mathbf{u}_* = \arg\min_i f_i$.
```

4 Algorithm 2: Second-Order Potential Optimization

To accelerate convergence near minima, we augment the previous scheme with local second-order updates (Newton or Levenberg–Marquardt type).

4.1 Gradient and Hessian Approximation

For a point $\mathbf{x}$, we approximate:

$$\nabla f(\mathbf{x})_i \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h},$$

and

$$H_{ij}(\mathbf{x}) \approx \frac{f(\mathbf{x} + h(\mathbf{e}_i + \mathbf{e}_j)) - f(\mathbf{x} + h(\mathbf{e}_i - \mathbf{e}_j)) - f(\mathbf{x} - h(\mathbf{e}_i - \mathbf{e}_j)) + f(\mathbf{x} - h(\mathbf{e}_i + \mathbf{e}_j))}{4h^2}.$$

4.2 Damped Newton Step

The Newton update is computed as:

$$\Delta\mathbf{x} = -(H + \lambda I)^{-1} \nabla f,$$

where $\lambda > 0$ is a damping factor. If the step fails to reduce f, λ is increased; otherwise it is decreased.

This corresponds to `minimize_potential_second_order()` in the Python code.

5 Connection to Classical Optimization Methods

5.1 Relation to Momentum Methods

In first-order gradient methods with momentum:

$$\mathbf{v}_{t+1} = \mu \mathbf{v}_t - \eta \nabla f(\mathbf{x}_t), \qquad \mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1}.$$

Algorithm 2 Potential + Second-Order Refinement

```
1: Run potential-based updates as in Algorithm 1.
2: Select $N_{\text{top}}$ best particles for local refinement.
3: for each selected particle $\mathbf{u}_i$ do
4:    for Newton iteration $k = 1, \dots, N_{\text{newton}}$ do
5:       Compute gradient $\mathbf{g}$ and Hessian $H$.
6:       Solve $(H + \lambda I)\Delta\mathbf{x} = -\mathbf{g}$.
7:       Line-search along $\Delta\mathbf{x}$ to ensure decrease in $f$.
8:       If improvement $< \tau$, increase $\lambda$.
9:    end for
10: end for
11: Return updated best particle.
```

Kaplinskii's potential methods can be viewed as a *discretized, collective* momentum system, where multiple trajectories share information through a smoothed potential rather than direct gradient terms.

5.2 Relation to Particle Swarm Optimization

The potential force term $\mathbf{F}_i$ plays the same role as the attraction toward global and local best in PSO, but arises from a physically motivated potential rather than heuristic coefficients.

5.3 Relation to Newton and Quasi-Newton Methods

The second-order method approximates the local curvature of the objective directly, yielding steps similar to Newton or Levenberg–Marquardt updates:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - (H_t + \lambda I)^{-1} \nabla f_t.$$

In practice, this allows quadratic convergence once the algorithm enters a local basin of attraction.

6 Implementation Structure in Python

6.1 Class `Minimization`

- **Attributes:**
  - `u` : array of shape (K, d), particle coordinates.
  - `fu` : array of objective values.
  - `lb, ub` : lower and upper bounds.
  - `dim` : problem dimension.
- **Methods:**
  - `get_f(x)`: evaluates the objective.
  - `sift()`: resamples poor particles.

– `minimize_potential_nonlocal()`: first-order method.

– `minimize_potential_second_order()`: second-order refinement.

6.2 Usage Example

```
def ackley(x):
    x = np.asarray(x)
    d = len(x)
    term1 = -20 * np.exp(-0.2 * np.sqrt(np.sum(x**2)/d))
    term2 = -np.exp(np.sum(np.cos(2*np.pi*x))/d)
    return term1 + term2 + 20 + np.e

m = Minimization(ackley, X0=np.zeros(2), K=60, lb=[-5,-5], ub=[5,5])
fbest, xbest = m.minimize_potential_second_order(verbose=True)
```

7 Parameter Summary

Parameter	Description	Typical Range
ε_0	Initial potential smoothing	1–5
$\varepsilon_{\text{final}}$	Final smoothing scale	10^{-4}
β_0	Initial inverse temperature	0.5
β_{final}	Final inverse temperature	20–50
η	Step size	0.1–0.5
K	Number of particles	30–200
λ	Damping for Newton step	10^{-4}–10^{-2}

8 Convergence Considerations

Under mild smoothness conditions, the potential-based method converges toward regions where $\nabla f \approx 0$, and the second-order refinement ensures rapid local convergence near stationary points.

Potential-based algorithms exhibit good robustness in multimodal landscapes due to the nonlocal interactions among particles, which prevent premature convergence to poor minima.

9 References

1. Kaplinskii, I., & Propoi, A. (1994). *Nonlocal Optimization Methods Based on Potential Theory.*

2. Vidyasagar, M. (2005). *Theory of Learning and Generalization: With Applications to Neural Networks and Control Systems.* Springer.

3. Nesterov, Y. (2004). *Introductory Lectures on Convex Optimization: A Basic Course.* Kluwer.