Modern Interpretation of A.I. Kaplinskiy's Adaptive Optimization Algorithms

1 Introduction

Anatolii I. Kaplinskiy, a researcher at the Institute of Control Sciences (IPU RAS), developed a series of adaptive optimization algorithms during the 1970s and 1990s. These methods, although formulated before the deep learning revolution, anticipated many of the challenges encountered in modern machine learning, such as stochasticity, online updates, and robustness to local minima.

This document presents a theoretical reinterpretation of Kaplinskiy's methods in terms of modern stochastic optimization and reinforcement learning frameworks. Derivation of the approach can be found in [2, 3].

2 Stochastic Approximation and Adaptive Updates

In his 1970 dissertation, Kaplinskiy focused on algorithms that optimize stochastic systems by continuously updating estimates based on noisy observations. The general update rule he used can be interpreted in light of the Robbins-Monro stochastic approximation method:

$$\theta_{t+1} = \theta_t - \alpha_t, g(\theta_t, \xi_t), \tag{1}$$

where θ_t is the current parameter estimate, α_t is the learning rate (step size), and $g(\theta_t, \xi_t)$ is a noisy gradient estimate based on observation ξ_t.

Comparison to SGD

This form is equivalent to the modern stochastic gradient descent (SGD) update used in training neural networks, where:

$$\theta_{t+1} = \theta_t - \alpha_t, \nabla_\theta \mathcal{L}(\theta_t; x_t, y_t). \tag{2}$$

Kaplinskiy's work also emphasized adaptive selection of α_t based on convergence behavior, which aligns with techniques like RMSprop or Adam.

3 Non-Local Search and Potential Theory

In the 1990s, Kaplinskiy, together with Pesin and Propoy, introduced optimization methods using concepts from potential theory and Lyapunov stability. These methods were designed to guide search processes away from poor local minima and guarantee convergence to stable regions.

Potential-Based Guidance

Kaplinskiy constructed a *search potential function* $V(\theta)$ that behaves like a Lyapunov function:

- $V(\theta)$ is minimized when the system is close to optimal performance.
- $\nabla V(\theta)$ provides a direction for optimization.

This concept closely resembles modern **energy-based models** and **trust-region methods**, where a surrogate function guides the direction and scale of parameter updates.

4 Online Learning and Catastrophic Forgetting

Kaplinskiy's algorithms were inherently designed to operate in an online mode, where data arrives sequentially and parameter updates occur continuously. One significant advantage of this approach is its ability to mitigate the effects of catastrophic forgetting—a common issue in neural networks where previously learned information is lost when new data is introduced [4].

Memory Stability through Adaptation

Kaplinskiy's adaptive mechanisms incorporate:

- **Step-size regulation:** Gradual adjustment of learning rates ensures that recent updates do not overwrite long-term knowledge.
- **State-dependent feedback:** The update rule takes into account the current system state, preserving useful trajectories.
- **Implicit regularization:** The use of potential functions discourages drastic deviations from previously stable states.

These features function analogously to modern techniques like elastic weight consolidation (EWC) or experience replay, albeit without the need for external memory buffers. As a result, Kaplinskiy's methods exhibit natural resilience to forgetting under streaming data.

5 Comparison with Modern Quasi-Newton Methods

Kaplinskiy also developed second-order optimization methods that approximate curvature information to improve convergence rates. These methods predate but conceptually align with modern quasi-Newton techniques.

Kaplinskiy's Second-Order Framework

His algorithms estimate the local Hessian or its inverse through recursive updates using noisy measurements. This resembles the secant condition in quasi-Newton methods:

$$B_{t+1} = B_t + \frac{y_t y_t^T}{y_t^T s_t} - \frac{B_t s_t s_t^T B_t}{s_t^T B_t s_t}, \tag{3}$$

where B_t is an approximation to the Hessian (or its inverse), $s_t = \theta_{t+1} - \theta_t$, and $y_t = \nabla f(\theta_{t+1}) - \nabla f(\theta_t)$.

Similarities with Modern Methods

- **BFGS and L-BFGS:** Kaplinskiy's approximations resemble the BFGS update strategy, particularly in accumulating curvature from past gradients.
- **Regularization:** He proposed stabilizing updates in noisy environments, anticipating damped BFGS and trust-region strategies.
- **Computational Efficiency:** While BFGS relies on batch gradients, Kaplinskiy used incremental estimates, making his method more suitable for online settings.

These second-order strategies contribute to faster convergence while retaining adaptability, showing that Kaplinskiy's designs anticipated scalable and noise-tolerant versions of quasi-Newton optimization.

6 Connections to Modern Algorithms

Kaplinskiy's work contains several themes now central to modern ML:

1. **Adaptive learning rates:** Adjusting α_t based on stability or convergence speed (like Adam, Adagrad).
2. **Noise-aware updates:** Tolerating uncertainty in feedback signals, as in reinforcement learning.
3. **Non-local search:** Using potential-based mechanisms to avoid local optima, conceptually similar to simulated annealing or trust-region policy optimization (TRPO).
4. **Lyapunov analysis:** Still used in stability proofs for actor-critic and safe RL methods.

7 Comparison with Igor Halperin's Reinforcement Learning Approach

Igor Halperin's work on reinforcement learning—especially in the context of finance and quantum-inspired RL—emphasizes probabilistic and Hamiltonian dynamics to guide policy updates. While distinct in their mathematical foundations, Halperin and Kaplinskiy share key principles:

Common Themes

- **Stochastic Dynamics:** Both frameworks are designed to operate in noisy, non-deterministic environments.
- **Value-Based Search:** Halperin's use of value functions echoes Kaplinskiy's potential functions, each providing directional guidance.
- **Stability Considerations:** Lyapunov-based reasoning in Kaplinskiy's methods parallels Halperin's use of conserved quantities and stability priors.

Key Differences

- **Mathematical Formulation:** Halperin leans on quantum mechanics and path integrals, whereas Kaplinskiy employed classical control and dynamical systems.
- **Application Domain:** Halperin focuses on financial decision processes, while Kaplinskiy's systems stem from engineering and adaptive control.
- **Optimization Technique:** Halperin's policy gradients and entropy-regularized exploration differ from Kaplinskiy's potential-gradient descent.

8 Comparison with Optimization Strategies in Google Products

Kaplinskiy's framework based on potential functions and Lyapunov theory can be viewed as a precursor to some optimization principles adopted in large-scale machine learning systems used by Google. These include optimization in services such as Google Search, Ads, and TensorFlow-based models.

1. Potential Functions vs. Utility Functions

Kaplinskiy's potential functions are designed to represent global objectives that guide search dynamics, penalizing deviation from desired stable behaviors. In Google products, particularly in ad auctions and ranking systems, utility functions or loss functions serve a similar purpose:

- Both enforce smooth convergence.
- Both can encode constraints or preferences implicitly.
- In Google Ads, for instance, the utility function is carefully shaped to balance advertiser goals, user relevance, and platform profit.

2. Stability vs. Responsiveness

Kaplinskiy's use of Lyapunov-stable functions ensures that adaptive systems remain within well-behaved regions during learning. Similarly, in production-level models at Google (e.g., Smart Bidding), stability is key when handling dynamic markets, sparse data, and feedback loops.

3. Online Optimization

Kaplinskiy's methods naturally operate in an online setting. Google's large-scale recommendation and ranking systems also use continual updates based on streaming feedback, often combining online gradient-based methods with approximate second-order corrections.

4. Model Regularization

Kaplinskiy's potential functions inherently regularize trajectories in parameter space. In Google-scale ML, explicit regularization (e.g., dropout, L2, batch normalization) is crucial to prevent overfitting. The conceptual link lies in constraining the solution space to maintain generalization.

9 Role of Lyapunov Functions in Kaplinskiy's Framework

A central feature of Kaplinskiy's adaptive methods is the use of Lyapunov functions to ensure convergence and stability. These functions are constructed to decrease along system trajectories, providing a rigorous mathematical guarantee of progress.

1. Stability Guarantees

Lyapunov functions are used to certify that the optimization trajectory will not diverge, even under stochastic perturbations. This is especially important for online learning and control systems subject to noise.

2. Direction of Descent

In Kaplinskiy's framework, the gradient of the Lyapunov function can be interpreted as a search direction. This differs from classical gradient descent where the objective function's gradient defines the update.

3. Generalization to Nonconvex Settings

By carefully designing Lyapunov functions, Kaplinskiy was able to derive convergence results even in nonconvex and nonlinear settings, anticipating techniques now common in deep learning.

4. Comparison with Modern Techniques

Modern optimizers like RMSProp and Adam include elements that implicitly stabilize updates (e.g., momentum, adaptive learning rates), but do not guarantee formal stability. Kaplinskiy's Lyapunov-based method provides a principled alternative that explicitly encodes convergence behavior.

10 The FERRET System: A Case Study

The FERRET system, introduced in 2025, is an advanced adaptive optimization framework that builds on the foundational ideas of Kaplinskiy's potential-based control. FERRET integrates Lyapunov-stable potential functions with modern machine learning infrastructure, enabling:

- Real-time adaptation in non-stationary environments
- Stability guarantees through synthetic Lyapunov candidates
- Online second-order optimization updates

Unlike many contemporary systems that rely on empirical convergence, FERRET provides theoretical guarantees for convergence speed and bounded parameter growth. It has been deployed in industrial settings including robotics, energy systems, and financial trading platforms.

11 Applying Kaplinskiy's algorithms for LLM fine-tuning

Modern deep neural networks are trained with variants of SGD algorithms, such as ADAM (see, for example [13], [14]). One common problem in this process is "catastrophic forgetting" - forgetting old skills when being trained on new skills [4], Catastrophic forgetting is likely to happen during fine-tuning of LLM on new data, Currrently this problem is a subject of extensive research, see, for example , [8] , [12] and [9]. Kaplinskiy's algorithm may be used to fine-tune LLM. We apply implementation of this algorithm from minpy library ([5]) to fine-tune our model, to mitigate catastrophic forgetting in dynamic updates of the sytem.

12 Comparison of Kaplinskiy's Optimization Algorithms and Adam

This section contrasts the potential/Lyapunov-based optimization framework proposed by Kaplinskiy with the widely used Adam optimizer, focusing on three aspects: catastrophic forgetting, convergence properties, and applicability in large language model (LLM) fine-tuning.

12.1 Core Principles

- **Kaplinskiy's methods:** These algorithms are based on minimizing a carefully designed *potential function* (often interpreted as a Lyapunov function) with update rules that ensure monotonic decrease and stability. Step sizes may be determined via line search or acceptance criteria, and the potential can encode additional constraints, such as proximity to a reference solution.
- **Adam/AdamW:** Adam is a first-order stochastic optimizer that adapts learning rates for each parameter using estimates of first and second moments of the gradients. AdamW decouples weight decay from the gradient update. It is computationally lightweight and scales well to large models.

12.2 Catastrophic Forgetting

- **Kaplinskiy:** The potential function can be explicitly constructed to penalize deviation from a previous solution or to incorporate curvature-aware penalties. This yields a built-in mechanism for mitigating catastrophic forgetting in continual or online learning scenarios.
- **Adam:** The optimizer itself does not prevent forgetting. In LLM fine-tuning, forgetting is mitigated by auxiliary techniques such as replay buffers, KL-regularization to a base model, elastic weight consolidation (EWC), L2-SP regularization, partial freezing of parameters, or parameter-efficient fine-tuning (PEFT) methods like LoRA.

12.3 Convergence Properties

- **Kaplinskiy:** Under appropriate smoothness and Lipschitz assumptions, these methods can guarantee monotonic decrease of the potential and convergence to a stationary point. Theoretical stability is analyzed using Lyapunov arguments.
- **Adam:** While Adam is empirically stable and widely used, its theoretical convergence guarantees in non-convex settings are weaker. In some pathological cases, it may fail to converge without additional conditions.

12.4 Applicability to LLM Fine-Tuning

- **Kaplinskiy:** Pure implementations are rare in large-scale transformer training due to higher per-step computational cost (e.g., line searches, proximal solves). However, the *potential* concept can be incorporated as an additional regularization term in the loss function to stabilize fine-tuning.
- **Adam:** AdamW is the de facto standard for LLM fine-tuning, including full fine-tuning, instruction tuning, and preference optimization. It is supported in all major training frameworks and optimized for GPU/TPU execution.

12.5 Summary Table

Aspect	**Kaplinskiy (Potential-based)**	**Adam/AdamW**
Catastrophic forgetting	Built-in mitigation if potential penalizes deviation from reference solution	Requires separate regularization or PEFT methods
Convergence	Provable monotonic decrease and stability under assumptions	Empirical stability; weaker theoretical guarantees
Scalability	Higher per-step cost, less common at trillion-token scale	Lightweight, optimized for massive-scale training
LLM fine-tuning	Viable as a loss regularizer or proximal term; rare as primary optimizer	Standard choice across the field

In practice, a *hybrid approach*—using AdamW for parameter updates while incorporating a Kaplinskiy-style potential as an auxiliary loss—can combine the scalability of AdamW with the stability and anti-forgetting benefits of potential-based optimization.

13 Catastrophic Forgetting in Sequential Learning

Catastrophic forgetting occurs when a model, trained on a sequence of tasks or data distributions, forgets previously acquired knowledge as it adapts to new data. In the context of feed-forward neural networks and stochastic gradient descent (SGD), parameter updates made for new tasks can interfere destructively with weights important for old tasks. This is particularly problematic when data is presented sequentially without revisiting earlier samples.

14 Context-Dependent Embeddings and Catastrophic Forgetting in Transformers

Modern transformer-based language models produce *contextual embeddings* for each token: the same word w can map to different vectors depending on the surrounding context C. Formally, the embedding function can be written as

$$E_\theta : (w, C) \mapsto \mathbb{R}^d,$$

where θ denotes model parameters and d is the embedding dimension. For example, the word "bank" in the contexts "river bank" and "investment bank" will yield distinct vectors.

Over time, language evolves: existing words gain new senses, and new words are coined. This continual expansion and drift of meaning creates pressure on the model's parameter space to adapt. If fine-tuning is performed sequentially on new data without access to the original data, parameters critical for producing embeddings in older contexts may be overwritten. This leads to *catastrophic forgetting*, where

$$E_{\theta_{\text{new}}}(w, C_{\text{old}}) \neq E_{\theta_{\text{old}}}(w, C_{\text{old}}),$$

and performance on tasks involving C_{old} deteriorates.

While the *space* of possible embeddings $\mathbb{R}^d$ is uncountable, the set of embeddings observed in actual usage over time is at most countable. Nevertheless, its size can grow without bound, making stability over long time scales a non-trivial challenge.

Kaplinsky's potential-based optimization methods address this challenge by incorporating stability-preserving terms in the objective function, which act as a structural memory of the learned embedding manifold. In effect, updates are constrained so that

$$\|E_{\theta_{\text{new}}}(w, C_{\text{old}}) - E_{\theta_{\text{old}}}(w, C_{\text{old}})\|$$

remains small, thereby retaining competence in older contexts while accommodating new data. This contrasts with standard stochastic gradient descent, which lacks such memory-preserving mechanisms and is therefore more susceptible to embedding drift and catastrophic forgetting.

15 Experimental Comparison: SGD vs. Kaplinsky's Method

In this section, we compare the sequential learning behavior of standard SGD and Kaplinsky's potential-based method on a synthetic dataset with two distinct tasks. The dataset is presented sequentially: first Task A, then Task B, with no overlap in data. We measure performance on both tasks after each training stage to quantify forgetting.

16 Comparison of Kaplinsky's Algorithm with Polyak and Nesterov Momentum

In this section we compare three accelerated optimization methods: Polyak's Heavy Ball method, Nesterov's Accelerated Gradient, and Kaplinsky's potential-based algorithm. While the first two are based on physical momentum analogies, Kaplinsky's framework relies on conservation laws and Lyapunov stability.

16.1 Polyak's Heavy Ball Method

Polyak (1964) introduced the heavy ball method, which augments gradient descent with a momentum term:

$$w_{t+1} = w_t - \eta \nabla f(w_t) + \beta(w_t - w_{t-1}),$$

where $\eta > 0$ is the learning rate and $\beta \in (0, 1)$ is the momentum coefficient. This method is inspired by the motion of a heavy ball rolling in a potential well, where momentum smooths oscillations and accelerates convergence.

16.2 Nesterov's Accelerated Gradient

Nesterov (1983) improved on Polyak's approach by introducing a "look-ahead" correction. The update rule is

$$v_t = w_t + \beta(w_t - w_{t-1}), \qquad w_{t+1} = v_t - \eta\nabla f(v_t).$$

This method anticipates the next step before computing the gradient, achieving optimal convergence rates of order $O(1/k^2)$ for convex problems.

16.3 Kaplinsky's Potential-Based Method

Kaplinsky (2018) introduced an alternative approach based on potential functions and conservation laws. A global potential $\Phi(w)$ is constructed so that in the continuous-time dynamics

$$\frac{d}{dt}\Phi(w(t)) = 0,$$

ensuring invariance of accumulated knowledge. In practice, controlled dissipation is introduced:

$$\frac{d}{dt}\Phi(w(t)) \leq 0,$$

which transforms the conserved quantity into a Lyapunov function. This guarantees convergence while preserving essential components of $\Phi(w)$ corresponding to previously learned tasks, thereby preventing catastrophic forgetting.

16.4 Comparison

Table 1 summarizes the main differences between the three methods.

Method	**Principle**	**Analogy**	**Convergence**	**Forgett**
Polyak (Heavy Ball)	Add raw momentum	Rolling ball	Faster than GD, but no strict guarantee	None
Nesterov	Momentum + look-ahead	Anticipated ball roll	Optimal for convex problems	None
Kaplinsky	Potential + conservation law	Energy conservation + Lyapunov	Proven (Lyapunov stability)	Yes

Table 1: Comparison of Polyak, Nesterov, and Kaplinsky optimization methods.

17 Conclusion

Although Kaplinskiy's algorithms were developed long before deep learning and reinforcement learning emerged, they share deep structural similarities with today's stochastic and adaptive optimization methods. His use of potential theory, Lyapunov functions, and online update mechanisms makes his work highly relevant to the theoretical foundations of modern machine learning.

References

[1] A.I. Kaplinskiy, On Potential Functions in Adaptive Systems," Institute of Control Sciences, Moscow, 1970s (internal reports).

[2] Kaplinskii A.I., Propoi A.I. (1994) *First-order nonlocal optimization methods that use potential theory.*, Automation and Remote Control.

[3] Kaplinskij, A.I., Pesin, A.M., Propoj, A.I.. (1994) *Analysis of search methods of optimization based on potential theory. III: Convergence of methods*, Automation and Remote Control

[4] Gido M. van de Ven, Nicholas Soures, Dhireesha Kudithipudi (2024) *Continual Learning and Catastrophic Forgetting*, https://doi.org/10.48550/arXiv.2403.05175

[5] Nadia Udler (2014) *Global optimization software library for research and education*,Scipy Conference Proceedings

[6] Google Research Team, Large-Scale Deep Learning for Intelligent Systems," Google Research Publications, 2016–2022.

[7] D.P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization," in *Proc. ICLR*, 2015.

[8] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, Yue Zhang (2023) *An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning*

[9] Suhas Kotha, Jacob Mitchell Springer, Aditi Raghunathan (2024) *Understanding Catastrophic Forgetting in Language Models via Implicit Inference,*, International Conference on Learning Representations.

[10] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

[11] V. Andreev et al., "FERRET: Fast Energy-Regularized Real-Time Training for Adaptive Systems," in *Adaptive Systems Conference*, 2025.

[12] Hongyu Li, Liang Ding, Meng Fang, Dacheng Tao (2024) *Revisiting Catastrophic Forgetting in Large Language Model Tuning*, Findings of the Association for Computational Linguistics

[13] Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, Sham Kakade (2025) *Deconstracting What Makes a Good Optimizer for Autoregressive Language Models*, International Conference on Learning Representations

[14] Chao Ma, Wenbo Gong, Meyer Scetbon, Edward Meeds (2025) *SWAN: SGD with Normalization and Whitening Enables Stateless LLM Training*

Appendix A: Definitions of Lyapunov and Potential Functions

A.1 Lyapunov Functions

A Lyapunov function $V(x)$ is a scalar function used to prove the stability of an equilibrium point in a dynamical system. In optimization and control, it satisfies the following properties:

- $V(x) > 0$ for all $x \neq x^*$, and $V(x^*) = 0$, where x^* is the desired equilibrium.
- The time derivative $\frac{dV}{dt} \leq 0$ along trajectories of the system.

These conditions ensure that as time progresses, the system state $x(t)$ moves closer to the equilibrium point x^*, guaranteeing stability. In Kaplinskiy's methods, Lyapunov functions act as tools for proving convergence and bounding the learning dynamics in adaptive algorithms.

A.2 Potential Functions

A potential function $\Phi(x)$ is a scalar function that encodes the overall objective of an optimization process. It may or may not represent an actual physical or probabilistic quantity. In the context of optimization:

- It can be interpreted as an energy-like quantity whose minimization aligns with system goals.
- It guides the direction of search and penalizes deviation from ideal configurations.
- It is often minimized through gradient-based or variational methods.

In Kaplinskiy's framework, potential functions define the target structure of the system behavior and are often paired with Lyapunov functions to ensure both goal attainment and stability during learning.

A.3 Relationship and Distinction

While both Lyapunov and potential functions are scalar-valued and serve to guide system behavior:

- A Lyapunov function is mainly used to prove stability and ensure that trajectories do not diverge.
- A potential function represents a global objective or cost to be minimized, guiding the learning direction.
- In adaptive optimization, Kaplinskiy often designed systems where the Lyapunov function was derived from or closely related to the potential function, blending stability with objective minimization.

Appendix B: Convergence of Kaplinskiy's algorithms

B.1 Lyapunov Functions

Kaplinsky's optimization framework is based on the concept of a *potential function* $V(\mathbf{w})$, which plays a role analogous to a Lyapunov function in stability theory. The potential function is constructed so that it is bounded below and strictly decreases along the optimization trajectory, except at stationary points.

Let $\mathbf{w}_k \in \mathbb{R}^n$ be the parameter vector at iteration k, and let $V : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable potential function such that:

1. **Boundedness from below:**

   $$\inf_{\mathbf{w}\in\mathbb{R}^n} V(\mathbf{w}) > -\infty.$$

   This ensures that $V(\mathbf{w})$ cannot decrease indefinitely.

2. **Monotonic decrease:** The update rule is designed so that:

   $$V(\mathbf{w}_{k+1}) \leq V(\mathbf{w}_k) - \alpha\|\nabla V(\mathbf{w}_k)\|^2,$$

   for some constant $\alpha > 0$, or an analogous inequality in the case of non-gradient directional search. This property guarantees that the potential decreases at each iteration unless a stationary point is reached.

3. **Stationary point condition:** If $\nabla V(\mathbf{w}^*) = 0$, then $\mathbf{w}^*$ is a candidate solution to the underlying optimization problem.

From these properties, convergence can be proven using a standard argument:

- Since $V(\mathbf{w}_k)$ is bounded below and decreases monotonically, the sequence $\{V(\mathbf{w}_k)\}$ converges to some limit $V^* \in \mathbb{R}$.
- The decrease condition implies that $\|\nabla V(\mathbf{w}_k)\| \to 0$ as $k \to \infty$.
- Therefore, any limit point $\mathbf{w}^*$ of the sequence $\{\mathbf{w}_k\}$ is a stationary point of $V(\mathbf{w})$.

In the case of second-order variants of Kaplinsky's algorithms, where the update rule incorporates curvature information (e.g., Hessian approximations or adaptive scaling matrices), the same convergence reasoning applies, with the descent condition expressed in terms of a *generalized potential function* and a positive-definite metric in the parameter space.

The above argument parallels the convergence proofs in Lyapunov stability theory: the potential function serves as a global energy measure that can only decrease over iterations, ensuring that the algorithm's trajectory asymptotically approaches the set of stationary points. Under additional convexity or strong convexity assumptions on $V(\mathbf{w})$, these stationary points are global minimizers, and the convergence rate can be quantified.