Utilizing Machine Learning-optimized Piecewise Polynomial Models in Mechatronics

Hannes Waclawek [1,2] Stefan Huber [1]

[1] Josef Ressel Centre for Intelligent and Secure Industrial Automation,
Salzburg University of Applied Sciences, Salzburg, Austria

[2] Paris Lodron University Salzburg, Salzburg, Austria

February 6, 2024

Machine Learning Optimizers

Training a neural network is the iterative process of adapting weights of node connections in order to fit given training data.

- Gradients are calculated using automatic differentiation and
- weights are updated according to the gradient descent algorithm.

Optimizers

Modern ML frameworks come with a set of gradient-based optimizers potentially suitable for a wide range of optimization problems.

Trajectory Planning

There is a certain necessity for explainability when utilizing AI methods in the fields of path planning and trajectory planning.

- Movement of motors and connected kinematic chains like robotic arms
- Potential of causing harm within immediate environment
- Need for optimized and predictable movement

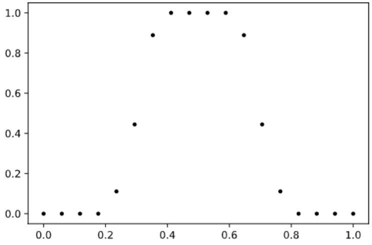
Calls for a...

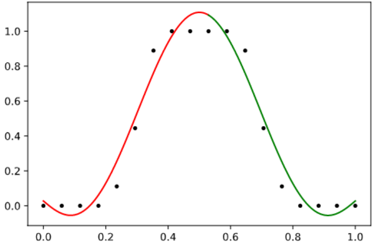
... "Predictable" model.

Motivation

Neural Networks	Trajectory Planning
$\downarrow$	$\downarrow$
Calculation of Gradients using Autodiff, Update of Weights via Gradient Descent Algorithm	Generation of Time-dependent Positional, Velocity and Acceleration Profiles
$\downarrow$	$\downarrow$
ML Frameworks APIs: Optimizers	PP Models
$\downarrow$	$\downarrow$

Utilizing Machine Learning (ML) Optimizers of Modern ML Frameworks directly for explainable Models in Trajectory Planning.





The cam position profile implicates...

- ...velocity, acceleration and jerk of the follower
- Maximizing goodness of fit: Path accuracy $\uparrow$
- Minimizing curvature: Induced forces $\downarrow$

Base Model

- Model spline s via piecewise polynomials with m polynomials of degree d
- Define individual polynomial p_i on the interval $I_i = [\xi_i, \xi_{i+1}]$ as

$$p_i(x) = \sum_{j=0}^{d} c_{i,j} x^j$$

with boundary points $\xi_0 \leq \cdots \leq \xi_m$.

Optimization criteria

- Optimize model parameters $c_{i,j}$ with respect to goodness of fit (minimize deviation to given points $x_i \in \mathbb{R}, y_i \in \mathbb{R}$)
- Achieve domain specific goals

Usage of Piecewise Polynomial Models with Orthogonal Bases

- Use Chebyshev polynomials of the first kind T_n as a basis
- Model spline s via m piecewise polynomials constructed via Chebyshev polynomials of degree up to d
- Define individual polynomial p_i on the interval $I_i = [\xi_{i-1}, \xi_i]$ as

$$p_i(x) = \sum_{j=0}^{d} c_{i,j}\ T_j(x), \text{ with}$$

$$T_n(x) = \begin{cases} 1, & \text{if } n = 0 \\ x, & \text{if } n = 1 \\ 2x\ T_{n-1}(x) - T_{n-2}(x), & \text{else.} \end{cases}$$

Gradient Descent Optimization: Loss Function

In order to establish C^k-continuity, cyclicity, periodicity and allow for curve fitting via L_2-approximation (n data points, m boundary points xi_i) we introduce the cost function

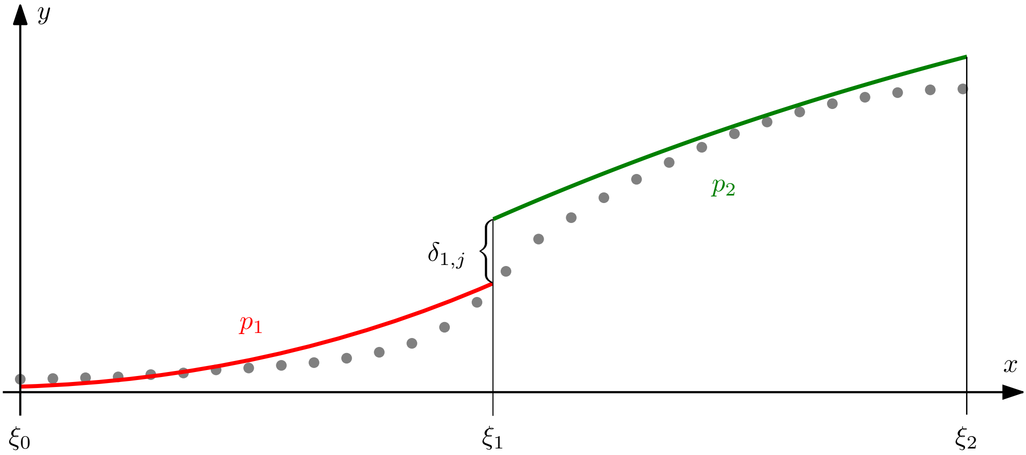
$$\ell = \alpha \ell_{\text{CK}} + (1 - \alpha)\ell_2$$

with $\ell_2 = \frac{1}{n}\sum_i |f(x_i) - y_i|^2$ and $\ell_{\text{CK}} = \sum_i D(\xi_i)$.

C^k-loss $D(x)$ is retrieved by summing up discontinuities at boundary points ξ_i across all derivatives relevant for C^k-continuity as

$\ell_{\text{CK}} = \frac{1}{m-1}\sum_{i=1}^{m-1}\sum_{j=0}^{k}\left(\frac{\Delta_{i,j}}{r_k}\right)^2$ with $\Delta_{i,j} = p_{i+1}^{(j)}(\xi_i) - p_i^{(j)}(\xi_i)$ and $r_k = \frac{d!}{(d-k)!}$.

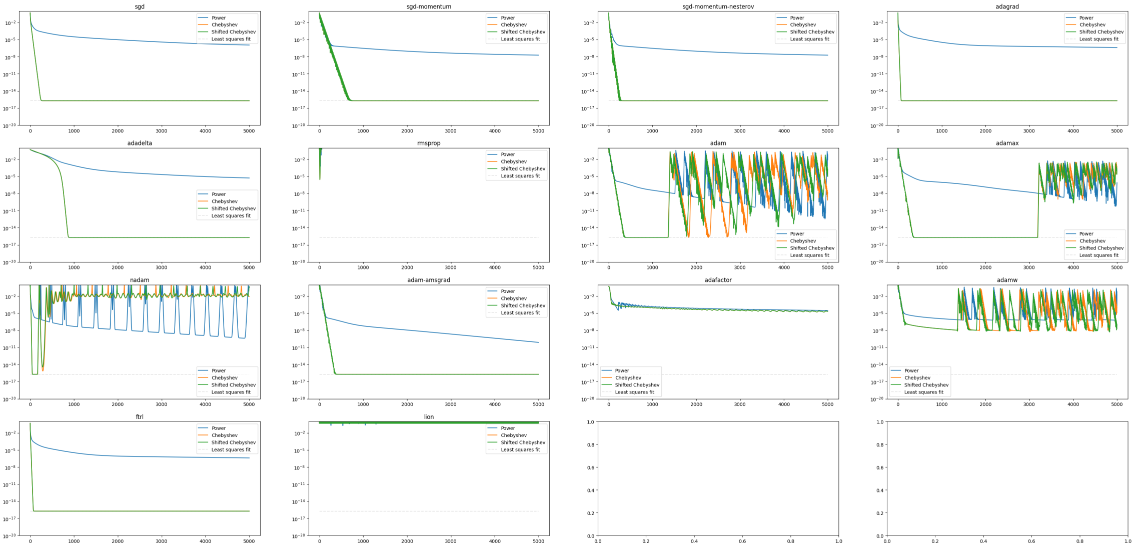
Base Model



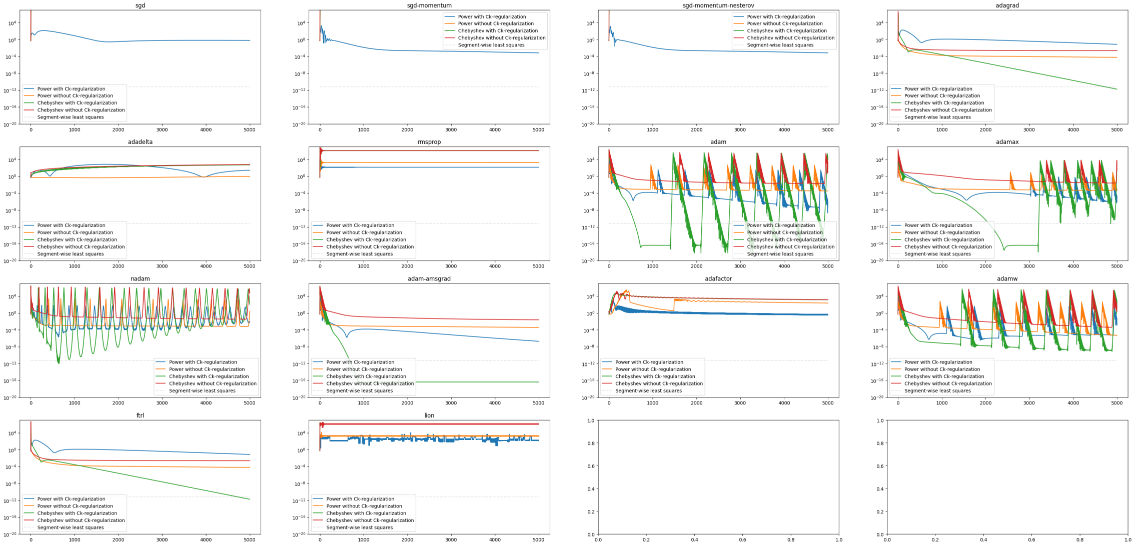
Gradient Descent Optimization in TensorFlow

```python
# Training Loop
for e in range(epochs):
    # GradientTape Context
    with tf.GradientTape(persistent=True) as tape:
        loss_ck = tf.multiply(calculate_ck_loss(), alpha)
        loss_l2 = tf.multiply(calculate_l2_loss(), 1-alpha)
        total_loss = tf.add(loss_ck, loss_l2)
    # Calculate Gradients
    gradients = tape.gradient(total_loss, pp.coeffs)
    # Apply Gradients
    optimizer.apply_gradients(zip(gradients, pp.coeffs))
    # Check for Early Stopping
    if early_stop():
        revert_to_best_epoch()
        return
```

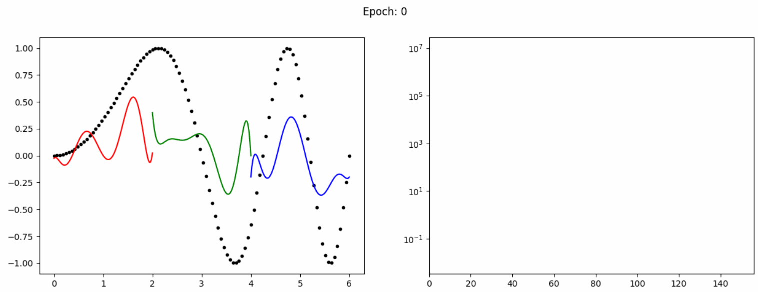
Investigate Convergence for available TensorFlow Optimizers - ℓ_2 only



Investigate Convergence for available TensorFlow Optimizers - ℓ

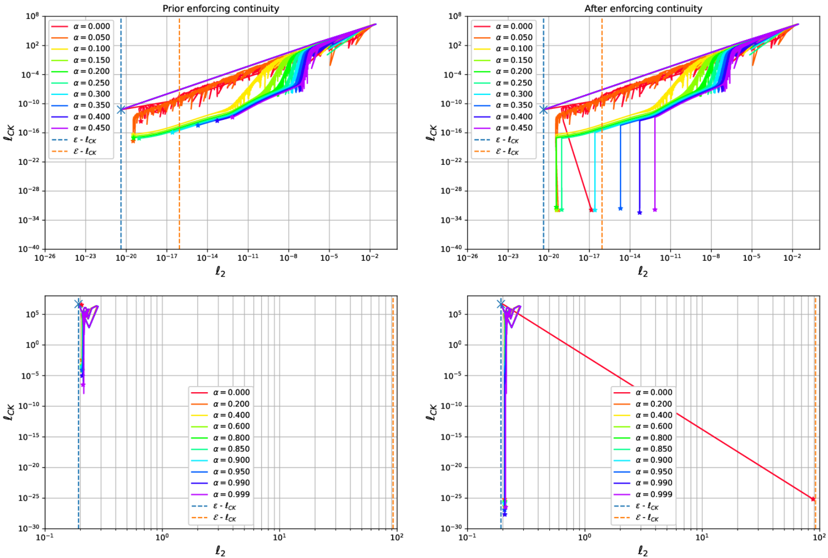


"ℓ's dance"



Epoch: 0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
1
2
3
4
5
6
10^7
10^5
10^3
10^1
10^{-1}
0
20
40
60
80
100
120
140

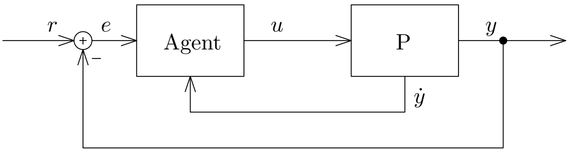
ℓ_{CK} over ℓ_2

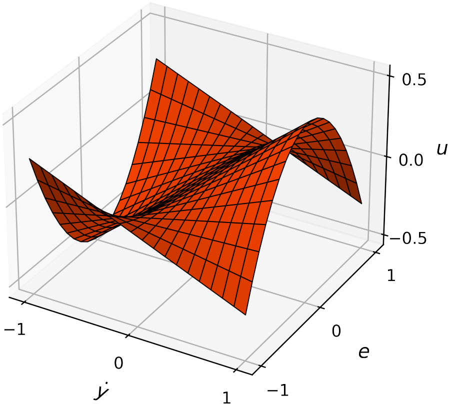


Testbed

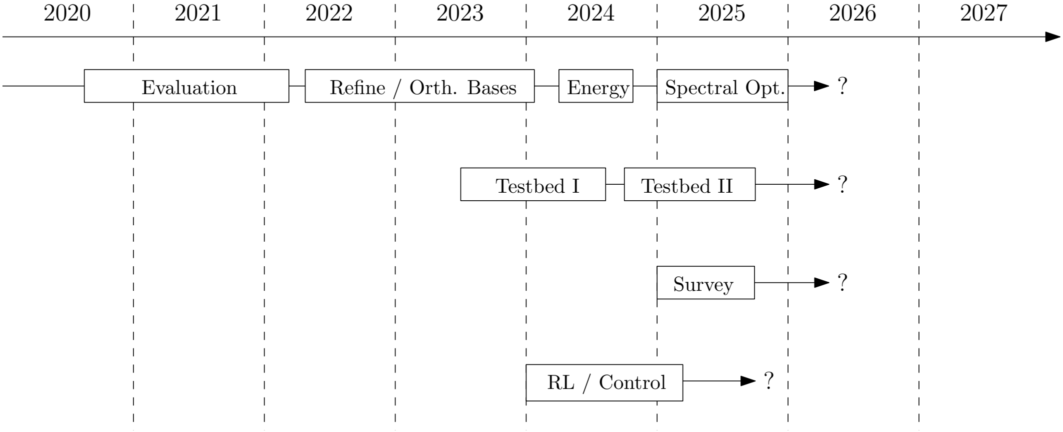
- Based on Lucas-Nülle servo machine test system

Link to Reinforcement Learning





Outlook


2020
2021
2022
2023
2024
2025
2026
2027
Evaluation
Refine / Orth. Bases
Energy
Spectral Opt.
?
Testbed I
Testbed II
?
Survey
?
RL / Control
?