Numerical Methods and Software Tools for Calorimeter Design

Prof. Dr. Nicolas R. Gauger

Chair for Scientific Computing
University of Kaiserslautern-Landau (RPTU)
67663 Kaiserslautern, Germany
nicolas.gauger@scicomp.uni-kl.de

Talk at 2nd Face-2-Face Meeting of CALO5D
October 15, 2025, Bonn, Germany

Scientific Computing
University of Kaiserslautern-Landau

Software

CoDiPack – Code Differentiation Package

Fast gradient evaluation in C++ based on Expression Templates.

Algorithmic Differentiation/Discrete Adjoint Method in SU2

MeDiPack – Message Differentiation Package

AD handling for message passing interfaces.

OpDiLib – Open Multiprocessing Differentiation Library

Universal add-on for operator overloading AD tools for the automatic differentiation of OpenMP parallel codes.

Derivgrind

Automatic differentiation of compiled programs using the Valgrind framework.

Upcoming Events

no event

News

- New Project “AI Care”: Fighting cancer with AI
  7. November 2023
- Workshop on Coupled Adjoints at TU Kaiserslautern
  11. October 2022
- Release of CoDiPack 2.0
  8. September 2021
- Release of OpDiLib v1.0
  24. February 2021
- New Project SIVERT: Fighting cancer with AI
  1. October 2020
- Best Student Paper Award at AIAA Aviation 2020
  22. June 2020

github.com

eoezkaya / RoDeO Public

Notifications | Fork 6 | Star 6

Code | Issues 2 | Pull requests 1 | Actions | Projects | Security | Insights

master | 2 Branches | 1 Tags | Go to file | Code

eoezkaya some modifications in 2DQuadratic test cases readme — faee4b7 · 5 months ago — 303 Commits

Tests	some modifications in 2DQuadratic test cases readme	5 months ago
src	stable version with some new test cases	5 months ago
LICENSE	added GPL 3	4 years ago
README.md	modified readme	7 months ago

README | GPL-3.0 license

RoDeO

RoDeo (Robust Design Optimization Package) is a package for simulation based global design optimization. It is specifically designed for scientific/engineering applications, in which the objective function and constraints are evaluated by computationally expensive simulations.

Features of the RoDeO Package:

- Surrogate model based Efficient Global Optimization (EGO) strategy
- Full data-driven approach
- Easy and efficient treatment of inequality constraints
- Gradient and tangent enhaced surrogate modeling

About

A package for robust design optimization

Readme
GPL-3.0 license
Activity
6 stars
1 watching
6 forks
Report repository

Releases 1

v1.0 Latest
on Sep 28, 2023

Packages

No packages published

Contributors 4

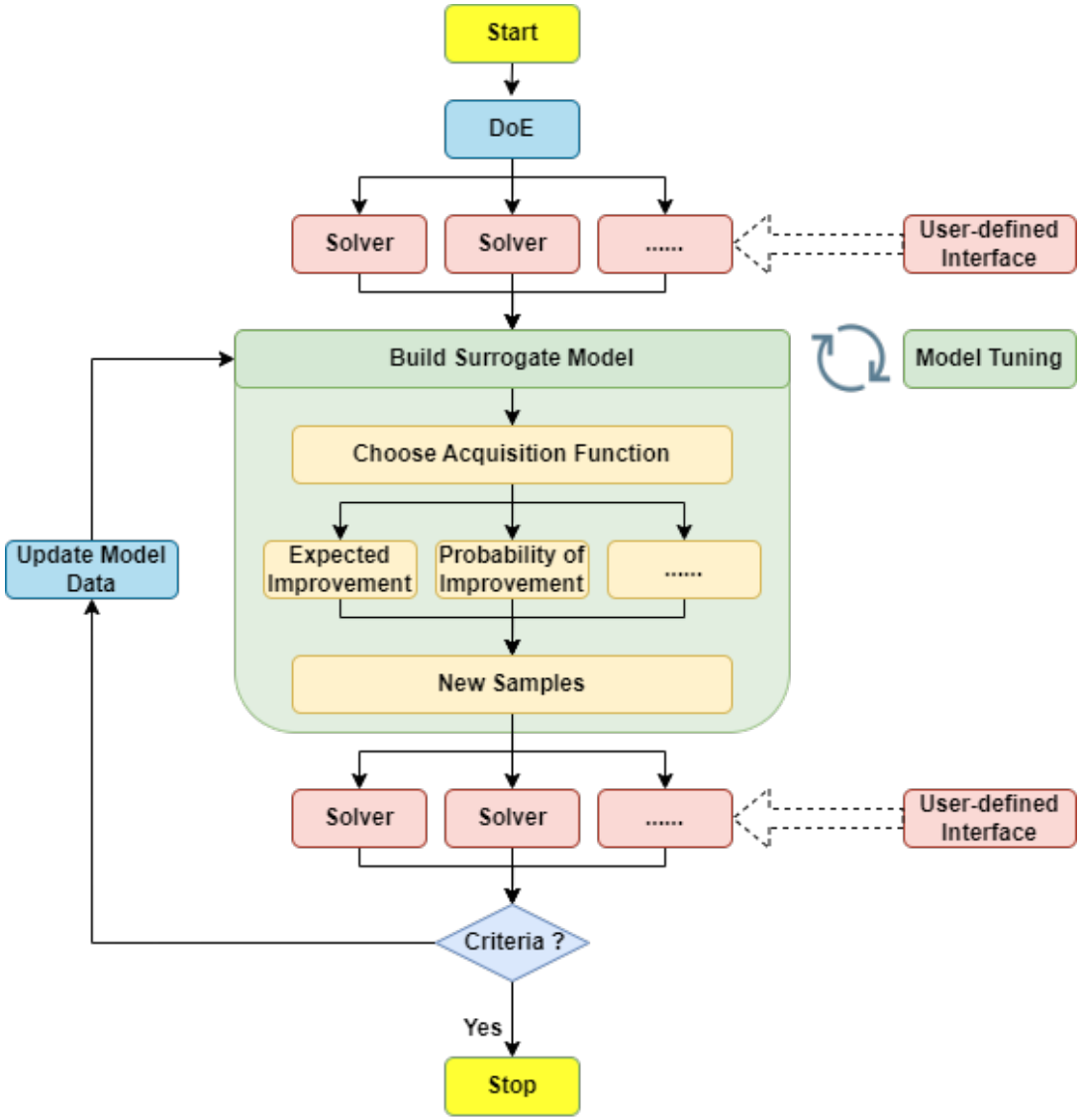
eoezkaya Emre Özkaya
Harivallabha Hari
TestSubjector Kumar Prasun

RoDeO[1]: **Ro**bust **De**sign **O**ptimization Package

- C++ code
- Test Driven Development (TDD) using Google Test library
- OpenMP parallelization
- GPLv3 license
- **Surrogate models:**
  - Using only functional values
  - Gradient enhanced
  - Tangent enhanced
- **Efficient Global Optimization (EGO)**
  - Full ML-based approach
  - Black-box usage
  - Unconstrained and constraint optimization
  - Flexible use of different surrogate models
  - User interface through configuration files

[1] https://github.com/eoezkaya/RoDeO.git

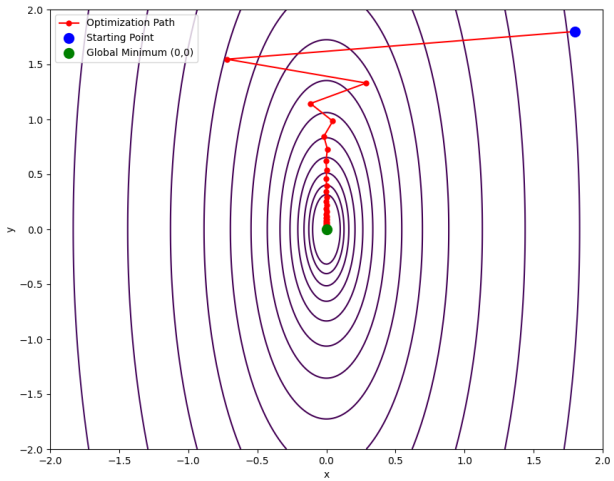
Surrogate Based Optimization Process



Motivation: A Unified Framework for Design Optimization

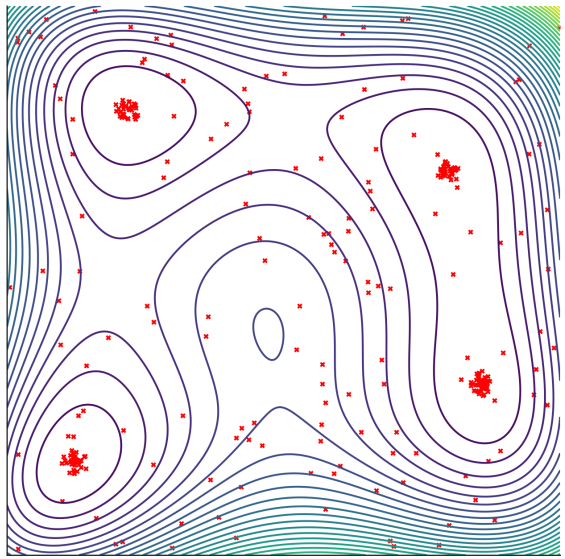
Gradient based optimization

- High-dimensional problems
- Proven fast convergence, local optimization
- Constraint treatment may be challenging
- Vulnerable to noise, initial design is important



Surrogate based optimization

- Low- to medium-dimensional problems
- Easy constraint handling
- Slow convergence but global optimization
- Robust, easy to parallelize
- No guarantee of convergence

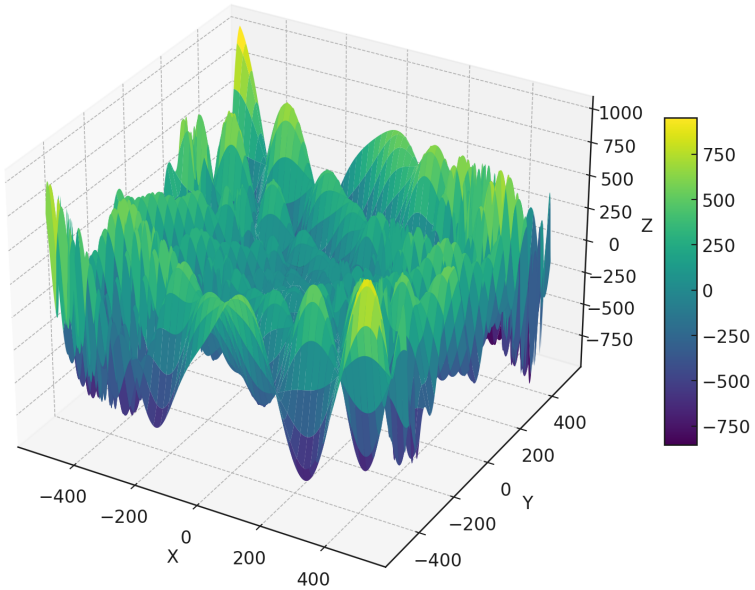


Minimization Benchmark Results

Tested Algorithms:

- Efficient Global Optimization (EGO): Default model in Rodeo
- Gradient Enhnaced EGO (GE-EGO): EGO with gradient enhancement
- ISRES: Improved Stochastic Ranking Evolution Strategy
- DIRECT: DIviding RECTangles
- CRS: Constrolled Random Search
- ESCH: Enhanced Scatter Search Algorithm for Continuous and Hybrid optimization

Eggholder Function



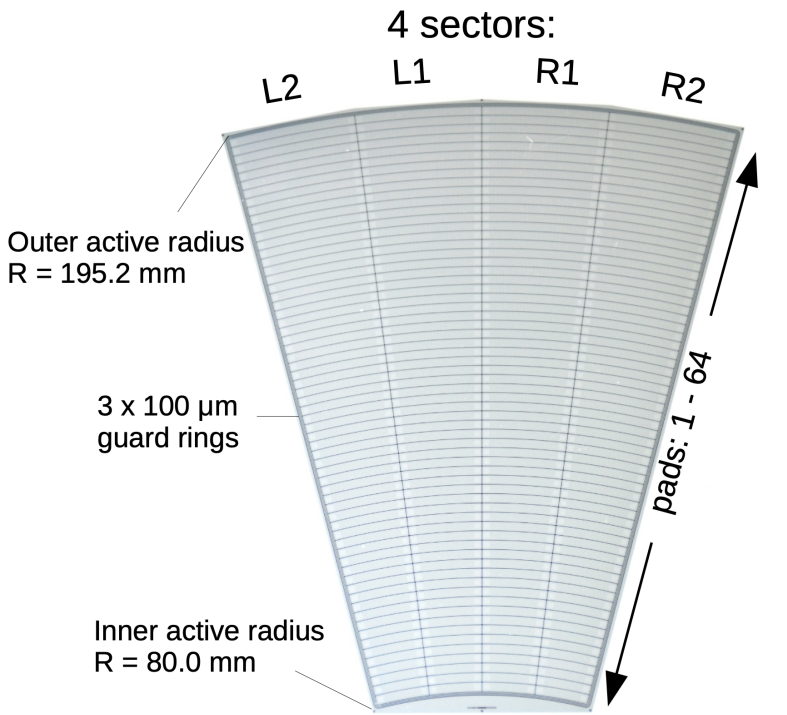
Minimization Benchmark Results[2]

Problem	Samples	Dim.	ISRES	DIRECT	ESCH	CRS	EGO	GE-EGO
Himmelblau	100	2	2.0725	0.0013	13.636	1.9713	0.0369	0.0049
Alpine02	200	5	-49.000	-80.183	-24.975	-47.418	-70.404	-87.807
Eggholder	100	2	-738.05	-931.59	-684.16	-791.35	-846.82	-853.93
Rosenbrock	100	2	0.6705	0.1111	0.5694	0.5054	0.0238	0.0126
Borehole	150	8	13.835	9.3536	12.888	10.866	7.8201	8.4296
Xor	200	9	1.8446	5.3502	3.8157	1.7023	1.4560	1.2854
Mishra03	100	2	0.0817	-0.0477	0.0702	-0.0632	-0.1037	-0.0904
Mishra05	100	2	-0.0317	-0.0543	0.2209	-0.0919	-0.0943	-0.1021
Michalewicz	300	10	-3.6881	-7.4239	-4.5648	-3.7406	-7.0055	-6.9416
Wingweight	300	10	159.61	129.78	139.81	134.43	124.11	124.65
Paviani	200	10	-18.742	-38.701	-7.5512	-16.810	-27.734	-43.146
Rastrigin	400	15	174.54	191.32	140.88	169.65	24.315	59.846
Dixon-Price	200	16	70302	2033	49920	64987	4706	67
Cola	400	17	57.767	82.356	52.570	57.512	17.977	17.340
Ackley	200	20	19.281	13.006	19.273	19.328	14.184	18.0106

[2]Lower values are always better

Case Study: LumiCal Detector Optimization

- **Current design:**
  - Silicon-tungsten sandwich calorimeter for future electron-positron linear collider experiments
  - 64 radial pads with 1.8mm pitch
  - 4 azimuthal sectors in one tile, each 7.5°
  - 12 tiles for full azimuthal coverage
- **Design parameters:** number of pads + thickness of each pad
- **Objective:** Maximize physics performance (e.g., resolution, cost)
- **Optimization:** Surrogate-based optimization of design parameters using RoDeO



In collaboration with Yan Benhammou, Tel Aviv University

AD of Geant4 – Many Challenges

Technical Challenges

- Geant4 is really big ($\sim 1\,\mathrm{M}$ lines of C++ code). AD tools *do not require rewriting*, but usually still *some manual changes* to the code. $\rightsquigarrow$ Machine-code-based AD with Derivgrind solves this aspect.
- Performance degrades, memory required to store the *tape*, etc.

Mathematical Challenges

The derivatives we compute with AD may not be the derivatives we need for optimization.

$$(\overline{f})' = \overline{(f')} \to \mathbb{E}(f') \overset{?}{\neq} (\mathbb{E}f)'$$

$\rightsquigarrow$ This talk: Study AD for G4HepEm/HepEmShow, a code that isolates a Geant4 testcase of electromagnetic showers in a simple sandwich calorimeter.

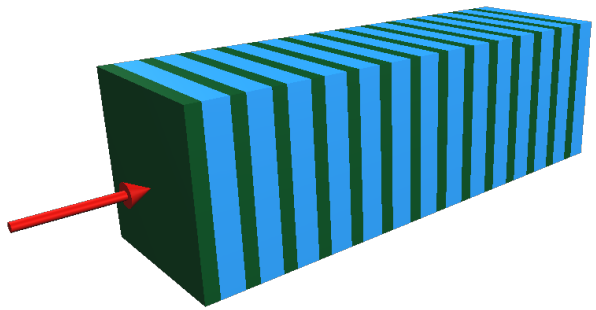
G4HepEm and HepEmShow

G4HepEm R&D

- R & D project by Mihaly Novak, Jonas Hahnfeld, Ben Morgan
- Simulation of electromagnetic showers (e^-, e^+, γ)
- Isolates the required data and functionalities from Geant4, well documented, customizable.

compact EM shower simulation
HepEmShow

- Created by Mihaly Novak
- Self-contained application using G4HepEm but not Geant4.
- Simulates electromagnetic showers in a sandwich calorimeter.

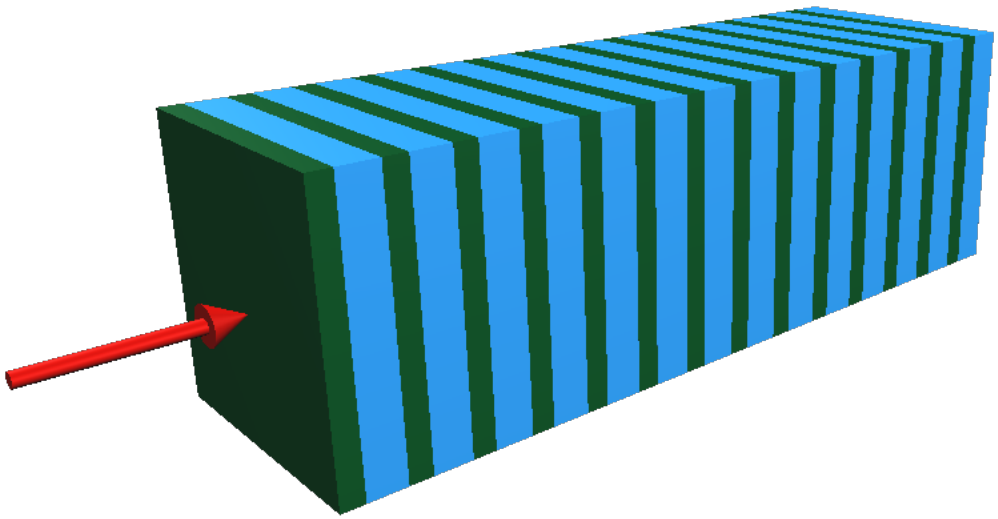


`github.com/mnovak42/g4hepem`

`github.com/mnovak42/hepemshow`

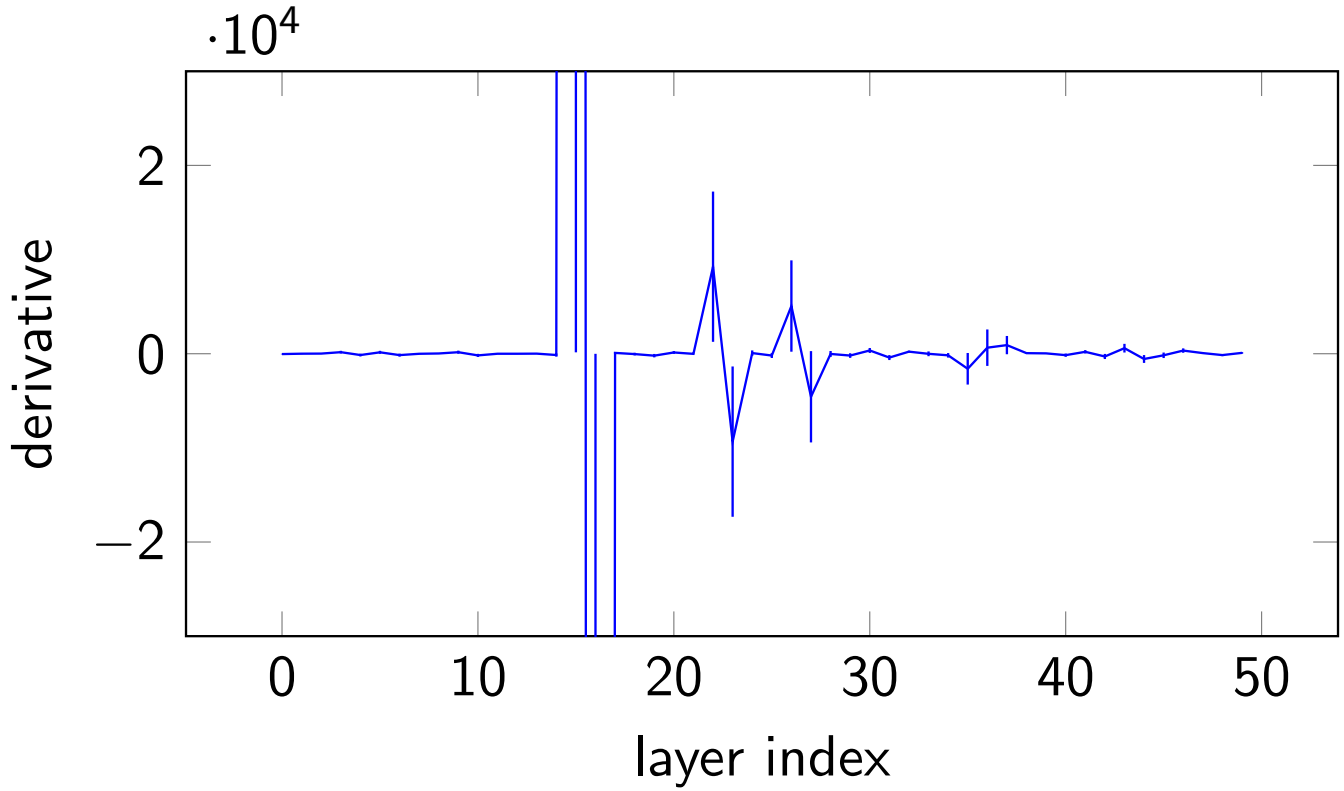
$\frac{\partial(\text{Energy Deposition})}{\partial(\text{Primary Particle Energy})}$ for the fully detailed simulation

Applied the AD tool CoDiPack.



$\frac{\partial\,(\text{Energy Deposition})}{\partial\,(\text{Primary Particle Energy})}$ for the fully detailed simulation

Applied the AD tool CoDiPack.

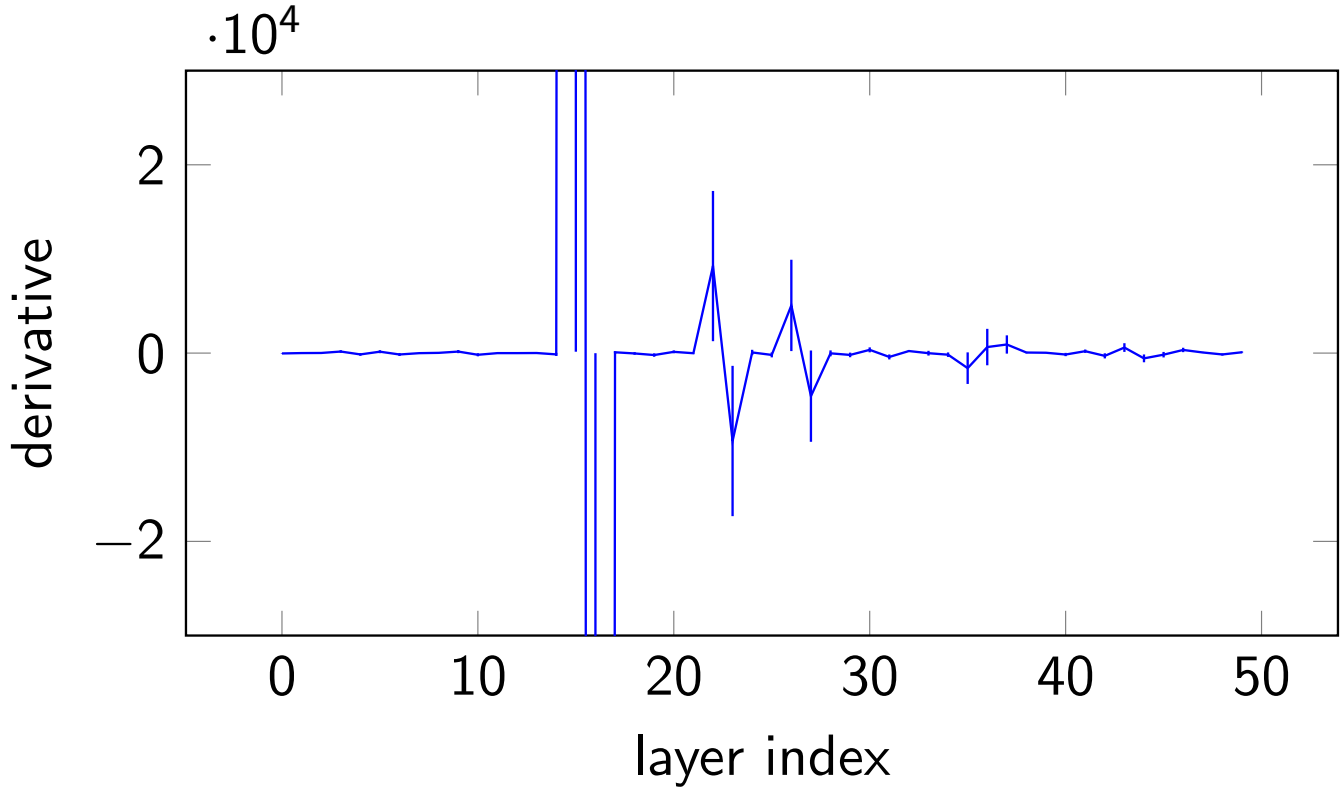


AD at 10 GeV

Look at the vertical axis range. These noisy derivatives can hardly be useful.

$\frac{\partial(\text{Energy Deposition})}{\partial(\text{Primary Particle Energy})}$ for the fully detailed simulation

Applied the AD tool CoDiPack.

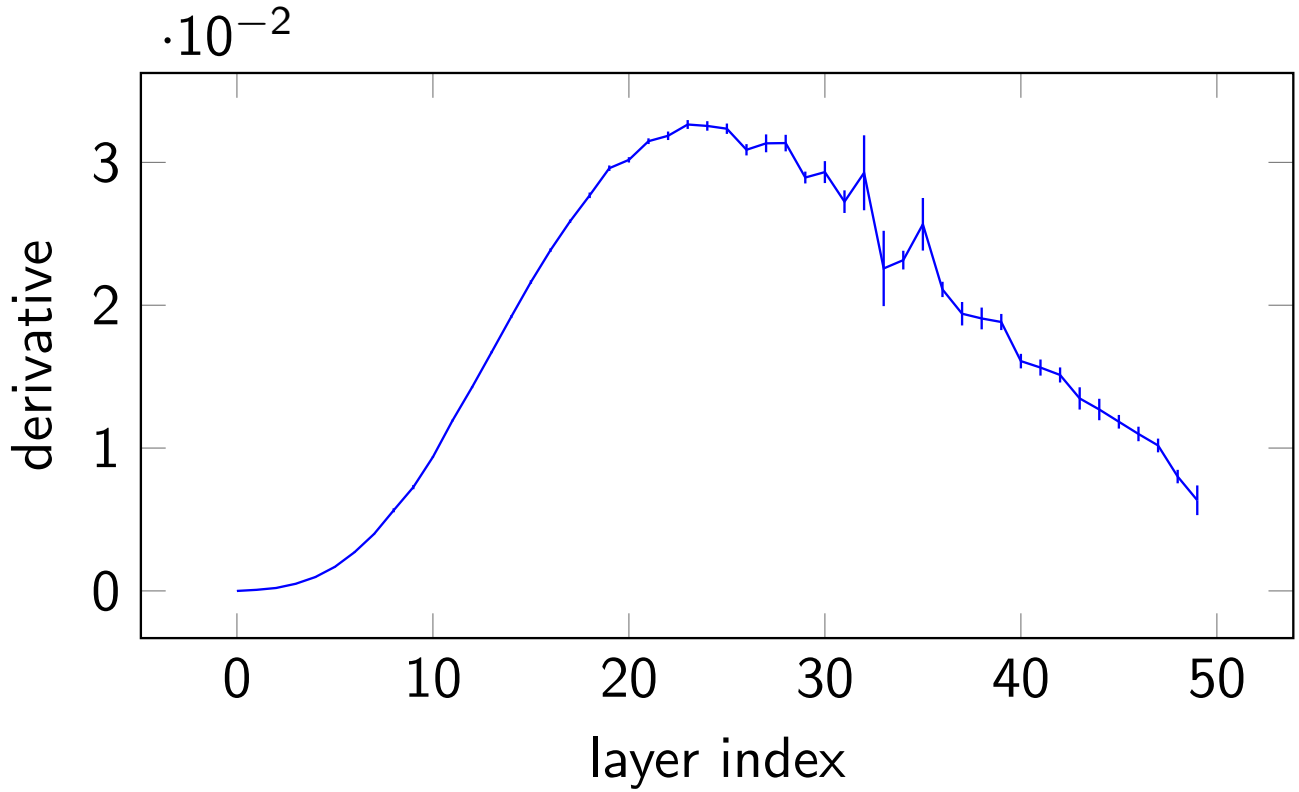


AD at 10 GeV

Look at the vertical axis range. These noisy derivatives can hardly be useful.

⇝ **Deactivate multiple scattering** in the simulation.

$\frac{\partial\,(\text{Energy Deposition})}{\partial\,(\text{Primary Particle Energy})}$ with simplifications

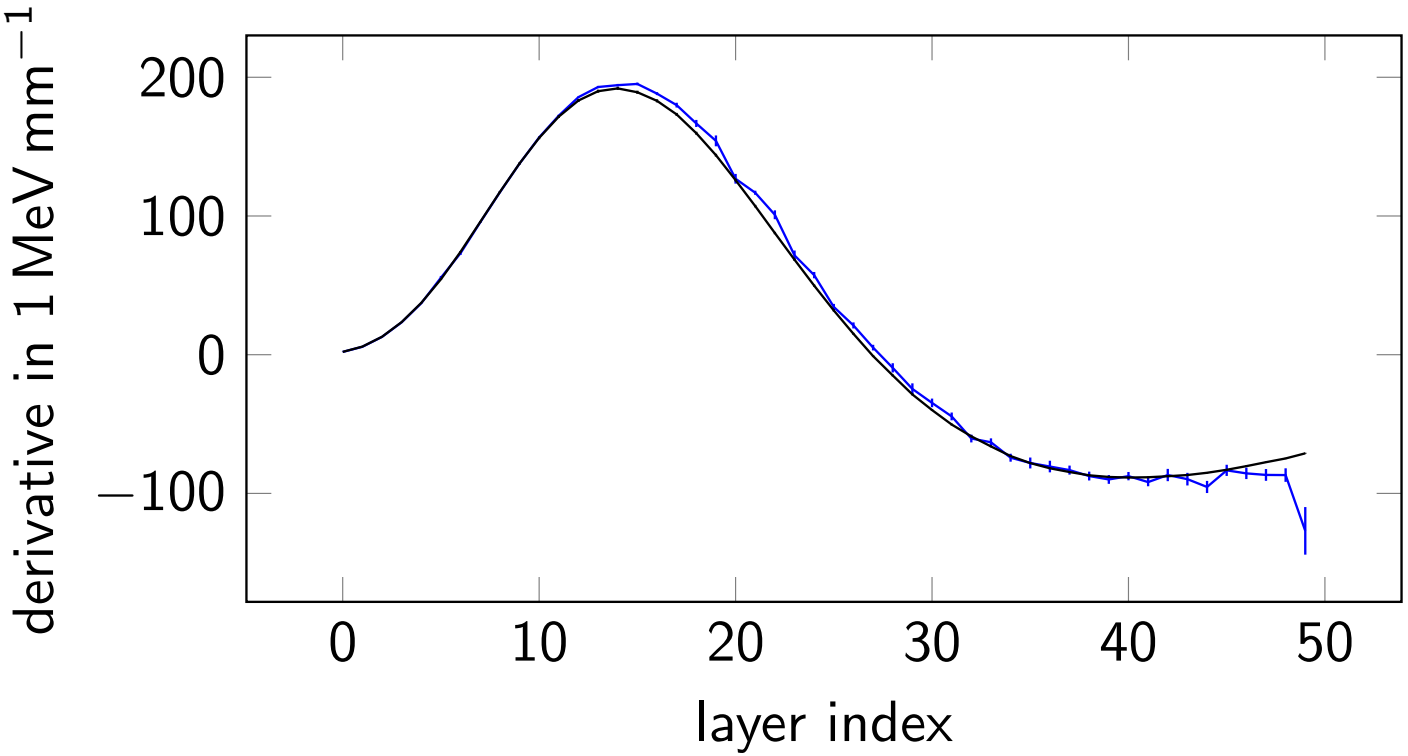


AD at 10 GeV

multiple scattering disabled

This looks much better!

$\frac{\partial\,(\text{Energy Deposition})}{\partial\,(\text{Absorber Thickness})}$ with simplifications



AD at 2.3 mm
Diff.quot. at
2.29. . . 2.31 mm
multiple scattering and
fluctuation disabled

Good agreement as well.

Gradient-Based Optimization Problem

Given a

- primary energy e and
- absorber thickness a,

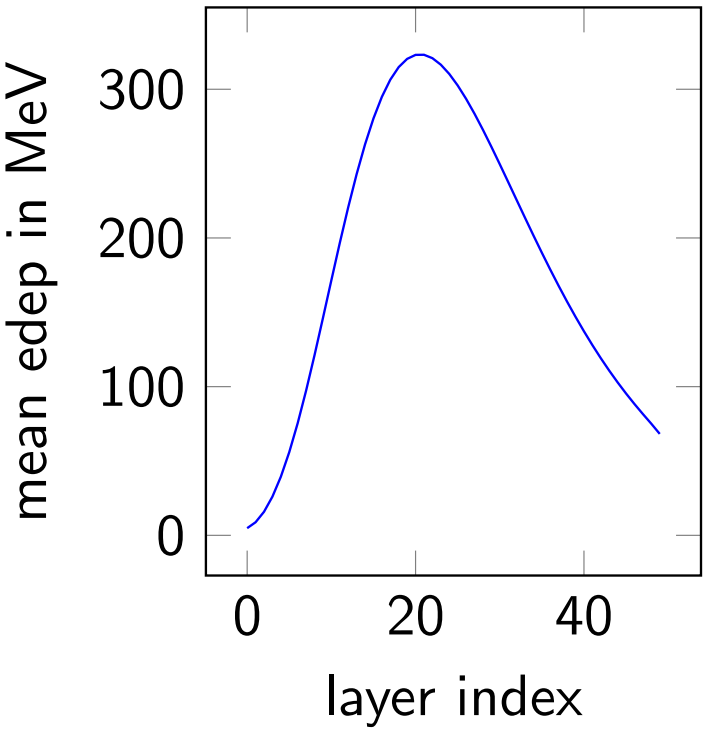
HepEmShow computes the resulting mean edep distribution $f_i(e, a)$ in the layers $i = 0, \ldots, 49$.

Given $f(e^*, a^*)$ (e. g. measured), can we infer e^*, a^*?

$\rightsquigarrow$ For the loss function

$$L(e, a) = \|f(e, a) - f(e^*, a^*)\|_2^2,$$

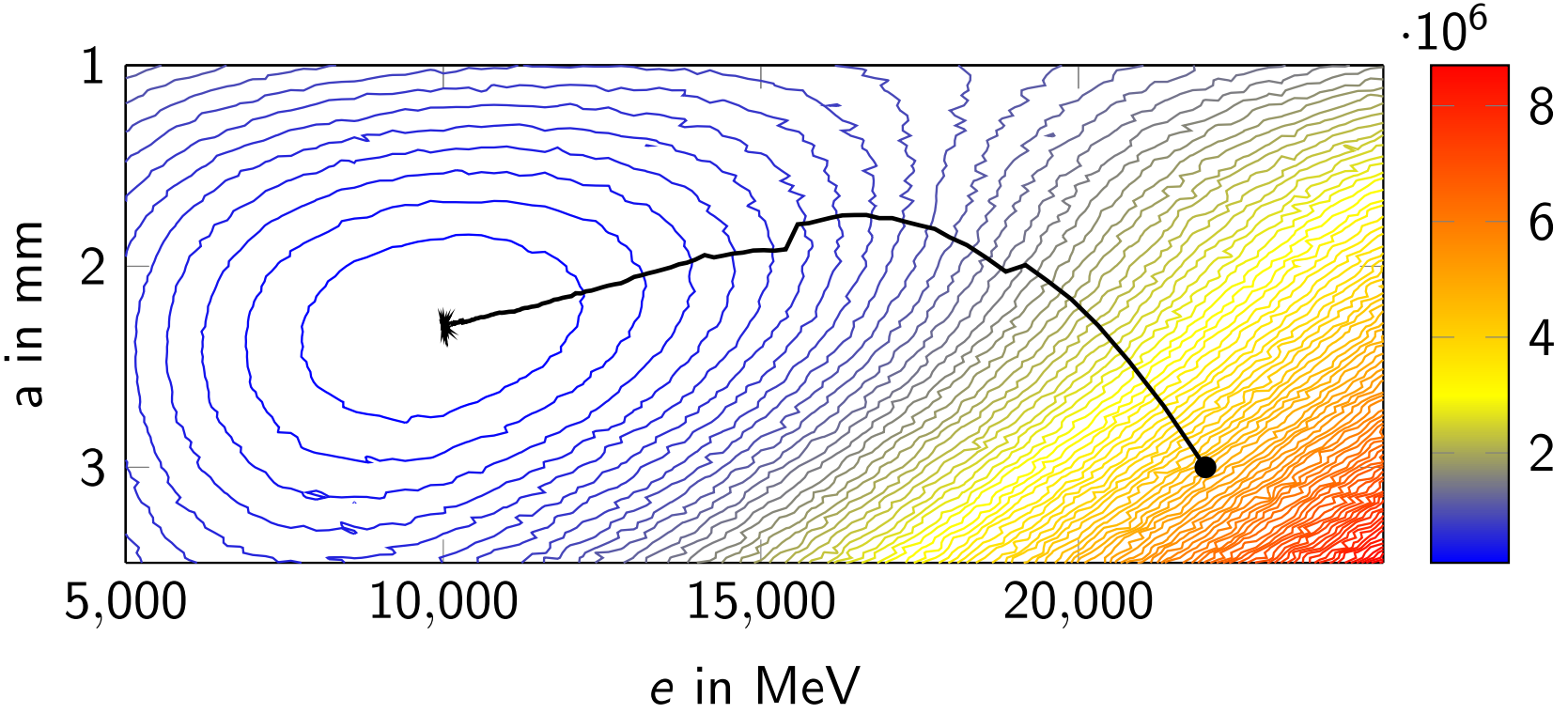
find $\min_{e,a} L(e, a)$!



Gradient-Based Optimization Results

Gradient descent with fixed step-sizes λ_e, λ_a:

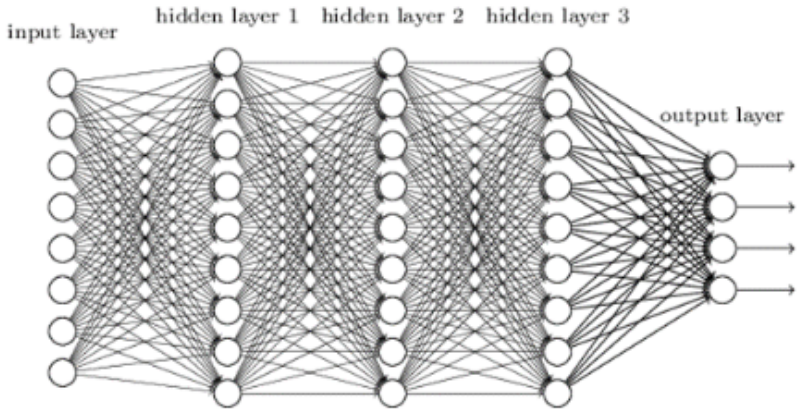
$$e_{i+1} = e_i - \lambda_e \cdot \frac{\partial L}{\partial e}(e_i, a_i)$$
$$a_{i+1} = a_i - \lambda_a \cdot \frac{\partial L}{\partial a}(e_i, a_i)$$



350 gradient descent steps with $\lambda_e = 1$, $\lambda_a = 10^{-7}\,\mathrm{mm^2\,MeV^{-2}}$, 1000 events per iteration. Starting from $e_0 = 22\,000\,\mathrm{MeV}$, $a_0 = 3\,\mathrm{mm}$, converging to the minimizer $e^* = 10\,000\,\mathrm{MeV}$, $a^* = 2.3\,\mathrm{mm}$.

Deep Residual Learning

$y_{\text{data}} \rightarrow$



$\text{Loss}(y^N, c_{\text{data}})$

- **ResNet[2] propagation:**

$$y^0 = y_{data},\ y^{n+1} = y^n + F(W^n y^n + b^n),\ \forall n = 0, 1, \ldots, N-1 \quad (*)$$

- **Learning problem:**

$$\min_{W^n, b^n} \ Loss(y^N, c_{data}) \quad \text{such that} \quad (*)$$

[2]He et. al.:Deep Residual Learning for Image Recognition, IEEE Conf. on CVPR, 2016

Residual Networks Meet Dynamical Systems[3]

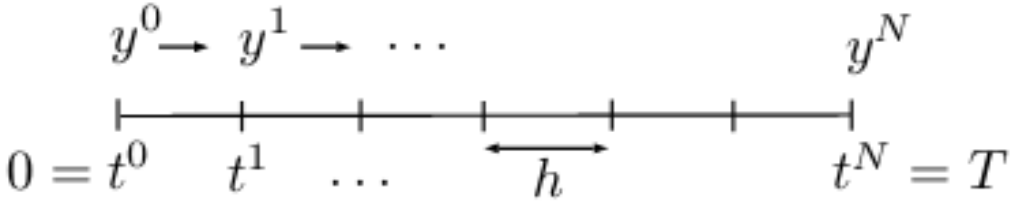
- **Forward ResNet propagation:**

$$y^{n+1} = y^n + hF(W^n y^n + b^n),\ \forall n = 0, 1, \ldots, N-1 \Leftrightarrow$$

$$\underbrace{\frac{y^{n+1} - y^n}{h}}_{\to \frac{dy(t)}{dt}} = F(W^n \underbrace{y^n}_{y(t)} + b^n)$$

- **Explicit Euler discretization of the ODE:**

$$\frac{dy(t)}{dt} = F(W(t)y(t) + b(t)),\ t \in (0, T) \quad \text{with} \quad y^0 = y_{data}$$



[3]Weinan E: A Proposal on Machine Learning via Dynamical Systems, Comm. Math. Stats., 2017

“Backpropagation = Discrete Adjoint”

- ResNet backpropagation

$$\bar{y}^N = \partial_{y^N} \text{Loss}(y^N, c_{\text{data}})$$
$$\bar{y}^n = \bar{y}^{n+1} + h \partial_y F(W^n y^n + b^n)^T \bar{y}^{n+1} \quad \forall\, n = N-1, \dots, 0$$

- Forward Euler discretization of **adjoint** dynamical system:

$$\frac{\mathrm{d}\bar{y}(t)}{\mathrm{d}t} = \partial_y F\left(W(t)y(t) + b(t)\right)^T \bar{y}(t) \quad \forall\, t \in (0, T)$$

with $\bar{y}(T) = \partial_y \text{Loss}(y(T), c_{\text{data}})^T$

Deep Learning Meets Optimal Control

ResNet Training $\Leftrightarrow$ Optimal Control Problem

$$\begin{aligned} \min_{W(t),b(t)} \quad & \text{Loss}(y(T), c_{\text{data}}) \\ \text{subject to} \quad & \frac{\mathrm{d}y(t)}{\mathrm{d}t} = F(W(t)y(t) + b(t)) \quad \forall t \in (0, T) \\ & y(0) = y_{\text{data}} \end{aligned}$$

- Network architecture design based on numerical discretization
- Regularization
- Optimization techniques

Multigrid for Forward- and Backpropagation

- **forward progapation** $\Leftrightarrow$ parallel multigrid for $\boldsymbol{y} = (y^0, \dots, y^N)$

$$\begin{aligned} y^0 &= y_{\text{data}} \\ y^{n+1} &= \Phi(y^n, \theta^n) \quad \forall\, n = 0, \dots, N-1 \end{aligned}$$

$$\Leftrightarrow \quad \boldsymbol{y}_{k+1} = H(\boldsymbol{y}_k, \boldsymbol{\theta}) \quad k = 1, 2, \dots$$

- **backpropagation** $\Leftrightarrow$ parallel multigrid for $\bar{\boldsymbol{y}} = (\bar{y}^N, \dots, \bar{y}^0)$

$$\begin{aligned} \bar{y}^N &= \partial_{y^N} \text{Loss}(y^N, c_{\text{data}}) \\ \bar{y}^n &= \partial_y \Phi(y^n, \theta^n)^T \bar{y}^{n+1} \quad \forall\, n = N-1, \dots, 0 \end{aligned}$$

$$\Leftrightarrow \quad \bar{\boldsymbol{y}}_{k+1} = H(\bar{\boldsymbol{y}}_k, \boldsymbol{y}, \boldsymbol{\theta}) \quad k = 1, 2, \dots$$

Layer-Parallel Training for Deep Residual Networks: XBraid[4]

https://github.com/xbraid

- Open-source software library
- Flexible user-interface for existing time-stepping codes
- Non-intrusive time-parallezation
- Outer derivatives are implemented by SciComp directly in XBraid
- Inner derivatives by coupling AD tools, especially CoDiPack

my_Step:	$y^{n+1} = \Phi(y^n, \theta^n)$
my_Objective:	$J(y^n, u)$
my_Clone:	$v^n = y^n$
my_Sum:	$y^n = \alpha v^n + \beta y^n$
my_SpatialNorm:	$\|y^n\|$
...	

[4] S. Günther, L. Ruthotto, J. B. Schroder, E. C. Cyr, and N. R. Gauger, Simultaneous Layer-Parallel Training for Deep Residual Networks, SIAM J. Math Data Sci., 2020

Layer-Parallel Training

- Iterative updates for $\boldsymbol{\theta}$:
  1. Layer-parallel forward propagation:

$$\text{For } k = 1, 2, \ldots : \quad \boldsymbol{y}_{k+1} = H(\boldsymbol{y}_k, \boldsymbol{\theta}) \quad \overset{k\to\infty}{\longrightarrow} \quad \boldsymbol{y}_*$$

$$\Rightarrow \text{Loss}(y_*^N, c_{\text{data}})$$

  2. Layer-parallel backpropagation:

$$\text{For } k = 1, 2, \ldots : \quad \bar{\boldsymbol{y}}_{k+1} = H(\bar{\boldsymbol{y}}_k, \boldsymbol{y}_*, \boldsymbol{\theta}) \quad \overset{k\to\infty}{\longrightarrow} \quad \bar{\boldsymbol{y}}_*$$

$$\Rightarrow \nabla_{\theta^n}\text{Loss} = \partial_{\theta^n}\Phi(y_*^n, \theta^n)^T \bar{y}_*^n \quad \forall\, n$$

  3. Network parameter update:

$$\theta^n \leftarrow \theta^n - \alpha \nabla_{\theta^n}\text{Loss} \quad \forall\, n$$

Early Stopping of Multigrid Iterations

- Iterative updates for $\boldsymbol{\theta}$:

1. Layer-parallel forward propagation:

$$\cancel{\text{For } k = 1, 2, \ldots :} \quad \boldsymbol{y}_{k+1} = H(\boldsymbol{y}_k, \boldsymbol{\theta}) \quad \cancel{k \to \infty \Rightarrow \boldsymbol{y}_*}$$

$$\Rightarrow \text{Loss}(y_k^N, c_{\text{data}})$$

2. Layer-parallel backpropagation:

$$\cancel{\text{For } k = 1, 2, \ldots :} \quad \bar{\boldsymbol{y}}_{k+1} = H(\bar{\boldsymbol{y}}_k, \boldsymbol{y}_{k+1}, \boldsymbol{\theta}) \quad \cancel{k \to \infty \Rightarrow \bar{\boldsymbol{y}}_*}$$

$$\Rightarrow \nabla_{\theta^n} \text{Loss}_{k+1} = \partial_{\theta^n} \Phi(y_{k+1}^n, \theta^n)^T \bar{y}_{k+1}^n \quad \forall\, n$$

3. Network parameter **update**:

$$\theta^n \leftarrow \theta^n - \alpha \nabla_{\theta^n} \text{Loss}_{k+1} \quad \forall\, n$$

One-shot Optimization

- Convergence theory: One-shot method[5]

$$\theta^n \leftarrow \theta^n - \alpha H_k^{-1} \left(\frac{\mathrm{dLoss}_k}{\mathrm{d}\theta^n} \right)$$

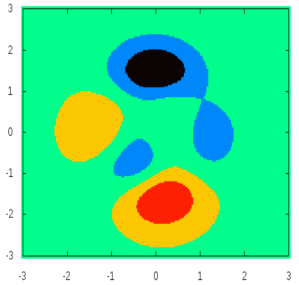
$$\text{with} \quad H \approx \partial^2_{\boldsymbol{\theta}} \left(L + \alpha \| \boldsymbol{y} - H(\boldsymbol{y}, \boldsymbol{\theta}) \|^2 \right)$$

- Numerically: limited memory BFGS approximation, Identity, ...

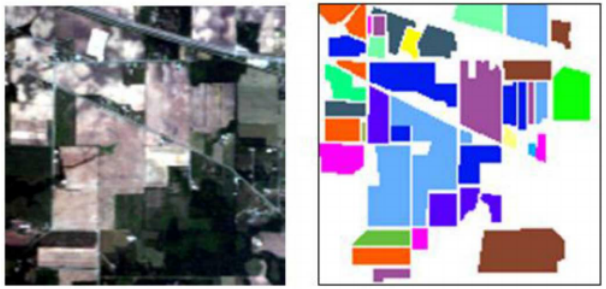
[5] [Griewank, Hamdi (2011)], [Gauger, Schulz et al. (2008)], [Blommert (2016)]

Test Cases

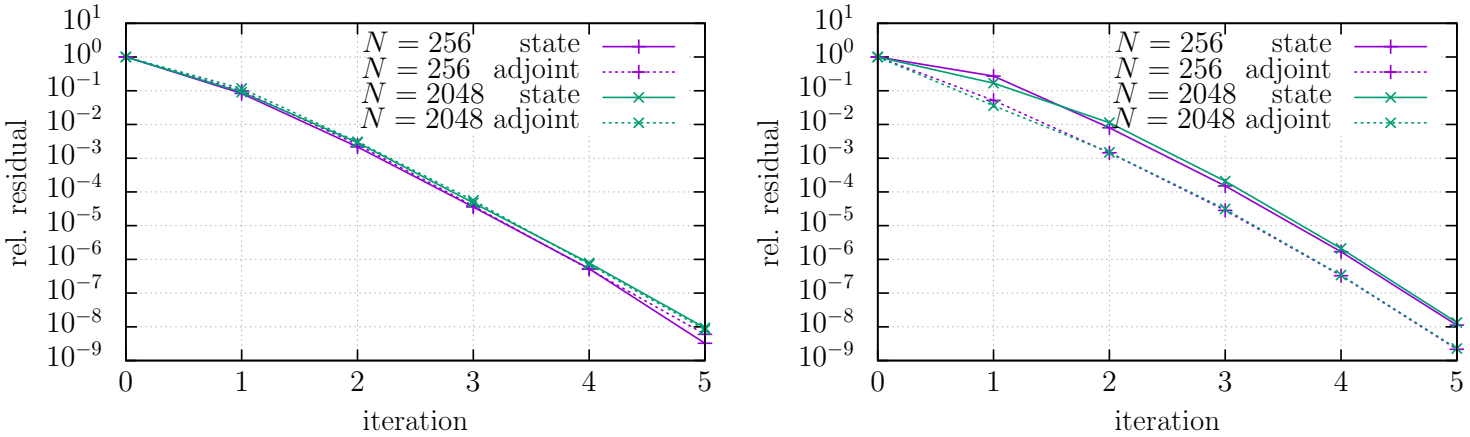
1. Level set classification:
   5000 grid points in $[-3, 3]^2$, 5 classes (level sets)



2. Hyperspectral image segmentation for
   Indian Pines data set:
   145×145 pixels each with 224 spectral
   reflectance bands, 16 classes (land cover)



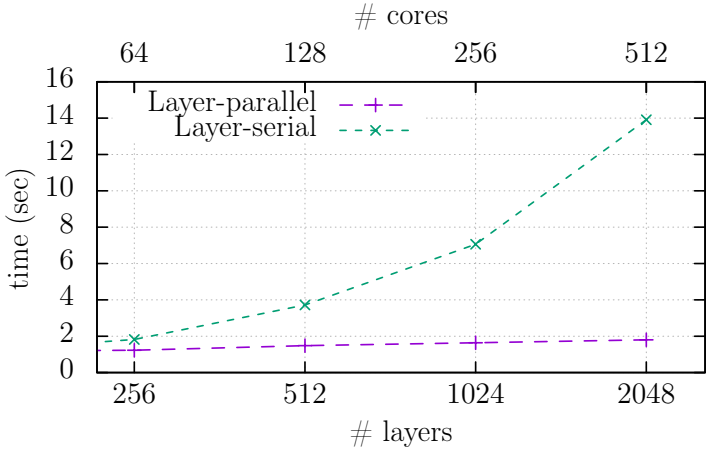
Multigrid Convergence



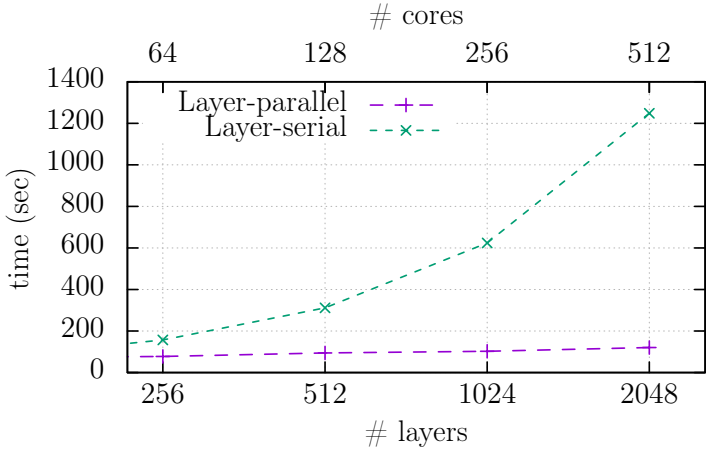
- Fast multigrid convergence, independent of the number of layers

Layer-parallel Multigrid Performance

Weak scaling for one gradient evaluation
4 layers per core



8x

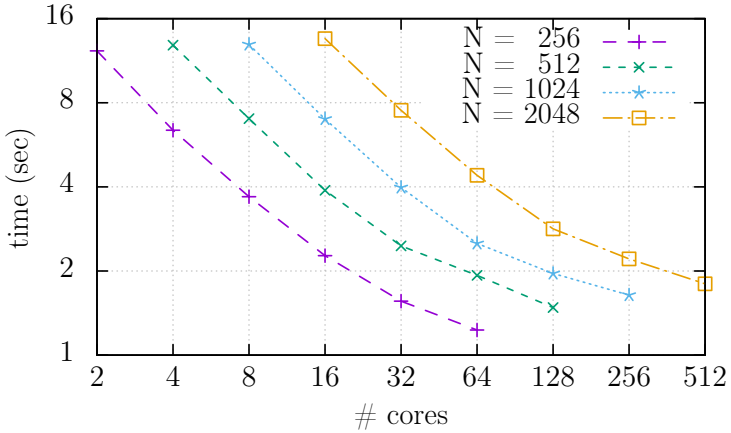


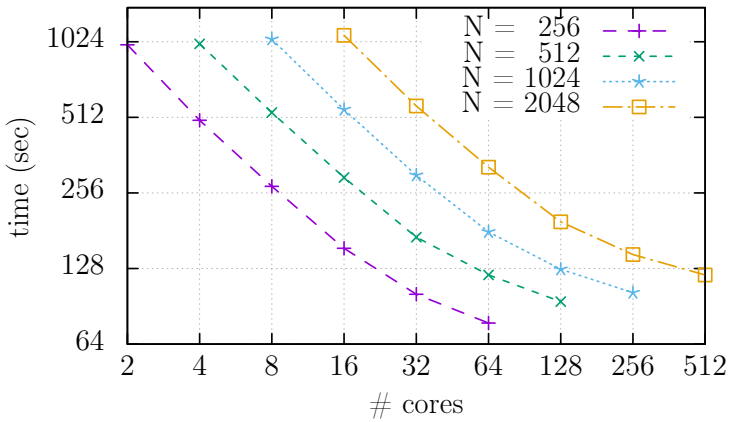
10x

- Layer-parallel multigrid yields constant runtimes for increasing problem sizes and comptational resources.

Layer-parallel Multigrid Performance

Strong scaling

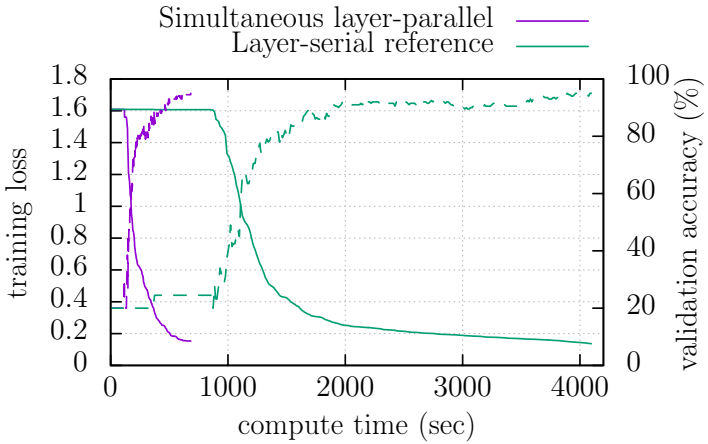




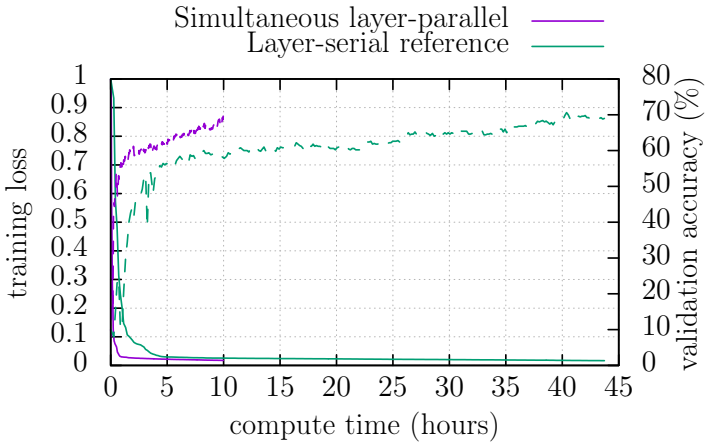
- Cross over point: $\approx$ 16 cores

Simultaneous Training

2 multigrid cycles per optimization iteration



Level sets
$N = 1024$



Hyperspectral images
$N = 512$

Thank you very much !

Questions ?