ISBN: 978-3-95450-208-0 doi:10.18429/JACoW-IPAC2019-MOPGW026

# TRANSFER LINE OPTICS DESIGN USING MACHINE LEARNING TECHNIQUES*

D. M. Vilsmeier†, Goethe Universität Frankfurt, Frankfurt, Germany
M. Bai, M. Sapinski, GSI Helmholtzzentrum für Schwerionenforschung GmbH, Darmstadt, Germany

*Abstract*

Optimization of transfer line optics is essential for delivering high quality beams to the experimental areas. This type of optimization is usually done by hand and relies on the experience of operators. The nature of this task is repetitive though highly complex. Besides optimizing the beam quality at the experiments this task is often accompanied by secondary objectives or requirements such as keeping the beam losses below an acceptable threshold. In the past years Deep Learning algorithms have experienced a rapid development and gave rise to various advanced software implementations which allow for straightforward usage of corresponding techniques, such as automatic differentiation and gradient backpropagation. We investigate the applicability and performance of these techniques in the field of transfer line optics optimization, specifically for the HADES beamline at GSI, in form of gradient-based differentiable simulators. We test our setup on results obtained from MADX simulations and compare our findings to different gradient-free optimization methods. Successfully employing such methods relieves operators from the tedious optimization tasks.

## INTRODUCTION

HADES beam line is one of the high-energy transfer lines linking SIS18 synchrotron with experimental areas. This beamline is about 160 meters long and contains 21 main quadrupoles and 2 active dipoles. Most of the beamline is straight while the two 7 degree dipoles towards the end of the line are used to guide the beam to the experiment. HADES experiment is very demanding concerning beam stability and minimization of beam losses. For that purpose it operates an additional beam quality monitoring system [1] which includes a *start* detector right in front of the target and a *veto* detector which is located 7 m downstream of the target. The veto detector is used to verify that the beam doesn't hit the vacuum chamber in front of the beam dump which would create additional background on the detectors. For a detailed description of the beamline please consider [2].

## DIFFERENTIABLE SIMULATOR

The extensive and rapid development of machine learning and specifically deep learning algorithms during the past years gave rise to various advanced software packages implementing required techniques such as automatic differentiation [3] and gradient backpropagation [4]. This opens the possibility for implementing simulations with tractable, analytically exact gradients with respect to simulation parameters and hence providing a natural optimization setting. The usage of such differentiable simulators has become a recent research interest in various areas of science [5–9].

Typically simulations, and particle tracking in particular, involve a sequence of well-defined, elementary operations that are applied in order to transform some initial state. This eventually leads to the desired output, denoted as final state, while intermediate states are typically discarded. By tracking all the involved operations and arranging them into a graph-like structure, where edges represent the data at the various stages of computation and nodes represent operations on these states, one can trace back how the final state, as well as any intermediate state, emerged from the preceding operations and their parameters. In case all involved operations are differentiable as well, one can compute the gradient of any of the states with respect to the preceding operations' parameters using the chain rule in form of the backpropagation algorithm [4]. In the scope of simulations the operations hereby represent the simulated system and are parametrized accordingly. Computing the gradient of user-defined metrics, e.g. the mean-squared error deviation between a simulated and measured final state, with respect to the model's parameters allows for finding gradient-based parameter updates that minimize the particular metric. Keeping track of the involved operations as well as performing the backpropagation of associated gradients is a tedious and complex task however various deep learning software packages, such as Tensorflow [10] or PyTorch [11], implement this functionality in a general way (typically used for updating the parameters of neural networks in order to minimize the deviation of predicted and expected output).

The MADX simulation tool for example can be used to perform particle tracking in accelerators in order to compute various quantities of interest, such as losses along a beamline and beam sizes at target locations. Gradient-free optimization procedures can use these results in order to update their recommendations about optimal parameter settings. These simulations can also be combined with gradient-approximating procedures (numerical differentiation) however estimating the gradients requires multiple forward passes of the data through the simulator and the gradient estimation can be susceptible to limited statistics in the data. The phase-space evolution in particle tracking along a beamline (or accelerator lattice in general) is given by a well-defined sequence of operations, given by the solutions to Hamilton equations, for example in form of transfer matrices and corresponding matrix products. These operations encode the various parameters of the accelerator and since they are differentiable they are suitable for usage in an automatic

* Work supported by HGS-HIRe
† d.vilsmeier@gsi.de

differentiation setting, coded in one of the aforementioned software packages. This allows for tracing back the simulation results to the parameters of interest, e.g. quadrupole gradient strengths, and to compute the corresponding analytically exact gradients for minimizing user-defined target metrics, e.g. losses along beamline or beam spot size at experiments.

## OPTIMIZING SPOT SIZE AT TARGET

We implemented particle tracking in form of six-dimensional thick lens tracking by considering linear optics, as a sequence of differentiable matrix products using the PyTorch package [11]. The gradient strengths of the 21 quadrupoles along the beamline were varied and optimized in order to reach the following goals: minimize beam loss along beamline, minimize beam size at start detector and keep beam size below threshold at veto detector. The corresponding metrics were realized as follows. The loss value per particle at each lattice element is computed as $\lambda = x^2/a_x^2 + y^2/a_y^2$ if $\lambda \geq 1$ and 0 otherwise, where $(a_x, a_y)$ is the aperture of the element. This loss value is not bounded from above and hence lacks a direct physical interpretation but since the objective of minimizing this loss value coincides with particles staying within the aperture it serves the purpose. Actually it has the desirable effect that for greater loss values the corresponding injected gradient will be larger as well and hence allows for faster convergence. We note that the aforementioned definition by cases of $\lambda$ is not differentiable around $\lambda = 1$ w.r.t. particle position $(x, y)$ however the implementation actually only considers particles for which $\lambda \geq 1$ (i.e. not injecting a gradient for particles with $\lambda < 1$). Particles that are lost at an element, i.e. particles for which $\lambda \geq 1$, are excluded from further tracking and do not contribute to metrics' values at subsequent elements. Beam sizes at target and veto detector are computed as standard deviations of the spatial distributions of particles that were not lost before.

The present version of the algorithm targets the beam spot size at the relevant locations directly and hence doesn't require an explicit notion of optics functions such as beta functions or dispersion in order to achieve this goal. Nevertheless if during operation the dispersion at target is desired to be specifically small for higher beam stability, this can be included by minimizing the correlation between particle momentum and position at the dispersion-free locations (though not directly optimized for in the present study).

We further like to point out that for the present study we have considered optimization via particle tracking but the same principle of differentiable simulations can be used in conjunction with optics functions directly, by propagating these functions according to the lattice parameters along the beamline using appropriate transfer matrices, and then optimizing for the desired values in a similar manner. Either way computes the specified metrics as well as their dependence on the relevant lattice parameters and hence allows for minimization by computing the corresponding gradients.

A common concern with gradient descent algorithms is that they do not necessarily converge to a global minimum but might as well converge to a local minimum instead. However during the past years various countermeasures such as varying starting points and cyclic learning rate schedules have proven successful in the scope of deep learning [12,13], the former of which can be trivially parallelized, i.e. not adding a constraint on compute time.

For the present study we used Adam optimizer [14] with learning rates varying between $1 \times 10^{-3}$ and $1 \times 10^{-5}$. The individual metrics were weighted with factors ranging from 0.5 to 2.0 and then added up to yield the overall optimization target (metric). Several restarts with different learning rates and weight factors have been performed. The results of the optimization procedure are summarized in Table 1 and shown in Figures 1, 2. We have only considered solutions that do not require a change in polarity of the quadrupole magnets.

Table 1: Results of the Optimization Using Differentiable Simulator (DS) Compared to Two Gradient-Free Optimizers, Evolutionary Algorithm (EA) and Particle Swarm Optimizer (PSO) (Units In Meter Except Where Otherwise Indicated)

| | **Loss** | **Target** | | | | **Veto** | |
|---|---|---|---|---|---|---|---|
| | (%) | $\beta_x$ | $\beta_y$ | $D_x$ | $D_y$ | $\beta_x$ | $\beta_y$ |
| **DS** | 0.30 | 0.25 | 0.23 | −0.02 | −0.10 | 200 | 209 |
| **EA** | 0.27 | 0.36 | 0.30 | 0.31 | −0.11 | 180 | 174 |
| **PSO** | 0.05 | 0.25 | 0.23 | 0.44 | 0.02 | 198 | 210 |

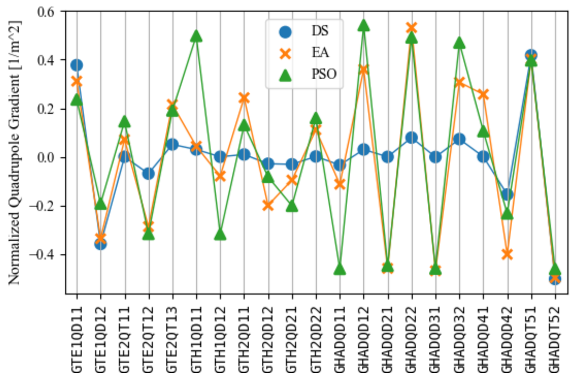

Figure 1: Comparison of resulting normalized quadrupole gradients for the different optimization methods.

### *Comparison with Gradient-Free Optimizers*

Gradient-free optimization procedures such as evolutionary algorithms and particle swarm optimization offer an alternative and flexible solution for complex optimization problems by effectively combining exploration and exploitation of the relevant parameter space through nature-inspired processes [15, 16]. Here we compare the final optics as computed by the Differentiable Simulator (DS) with two gradient-free optimizers, namely the (1+1) evolutionary algorithm [17] using fast mutation rates [18] (EA) and par-

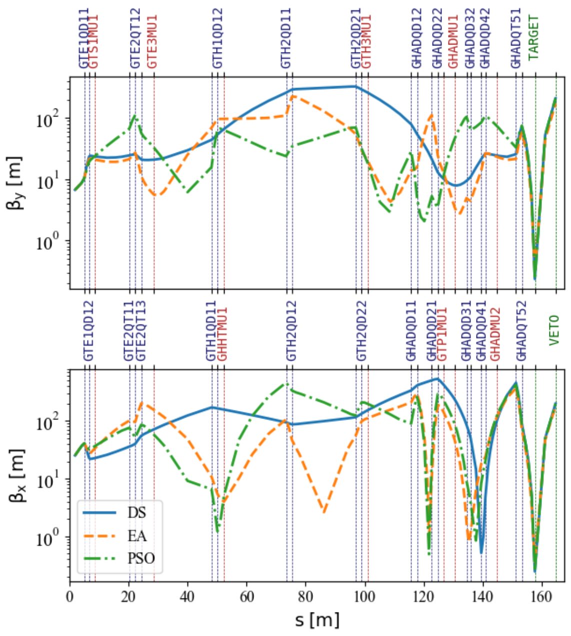

Figure 2: Comparison of resulting beta functions for the different optimization methods.

ticle swarm optimization (PSO), both implemented in the *nevergrad* software package [19]. These methods were configured to optimize directly for optics functions using the following target values: $loss \leq 0.5\,\%$, $\beta_{x,target} \leq 0.12\,\mathrm{m}$, $\beta_{y,target} \leq 0.06\,\mathrm{m}$, $D_{x,target} \leq 0.1\,\mathrm{m}$, $D_{y,target} \leq 0.1\,\mathrm{m}$, $\beta_{x,veto} \leq 90\,\mathrm{m}$, $\beta_{y,veto} \leq 45\,\mathrm{m}$. Scaling factors for the individual metrics have been varied between 1.0 and 2.0 and the best result was considered. The overall metric is computed as the sum of individual metrics scaled with the respective factors. Figure 1 shows the resulting quadrupole gradients, Fig. 2 shows the resulting beta functions and Table 1 gives an overview of the results. We note that the application of other gradient-free optimization methods to a similar task has been further explored in [20].

## INFERENCE OF MODEL ERRORS

An important advantage of the differentiable simulator is its versatility with respect to the involved parameters as well as the considered metrics. This allows for using the same method of differentiable particle tracking for inferring model errors, such as errors in quadrupole gradients, by using the trajectory response matrix (TRM) as the target metric. This matrix is created from observing the orbit response of the beam to changes in steerer magnets. Here we optimized for model errors by minimizing the squared difference between measured and simulated response matrix. We considered quadrupole gradient errors as well as screen calibration errors of up to 5%. Since the beam is extracted off-center the quadrupole gradient errors affect the resulting trajectory. The response matrix consists of 11 kickers and 2 dipoles used for steering the beam and 7 screens and grids, making a total of 30 effective pairs (i.e. having at least one quadrupole between them). The algorithm optimized for the 21 gradient errors as well as the 7 screen calibration errors in order to match the simulated and measured TRM. For the present study a test setup using MADX simulation representing the measurements was used in order to assess the effectiveness of the method. For that purpose a dedicated software package for interfacing MADX from Python has been created [21]. We found that the algorithm manages to successfully restore the actual orbit by inferring the relevant errors as shown in Fig. 3.

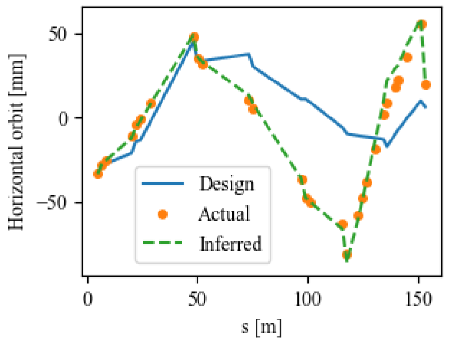

Figure 3: Design orbit without gradient errors, actual orbit resulting from errors and orbit inferred by the differentiable simulator.

## CONCLUSION & OUTLOOK

We have created a novel implementation of particle tracking which is based on the paradigms of dataflow and differentiable programming by leveraging corresponding techniques from existing deep learning software packages. Simulation results emerge as traceable quantities and hence allow for differentiation with respect to simulation parameters. This provides a natural setting for gradient-based optimization of model parameters which we have demonstrated at the example of beamline optics optimization by tuning quadrupole gradient strengths in order to fulfill multiple objectives such as minimizing beam loss and beam spot size at target locations. The method is versatile with respect to the included parameters as well as the final simulation metrics which we have shown by taking the example of quadrupole gradient error inference by considering trajectory response matrix deviation. For the present study we have considered particle tracking with linear optics in the scope of transfer lines. However the method is not limited to any of these specifications and it can be translated for usage with non-linear optics, evolution of optics functions or application to circular accelerators, each of which we plan to investigate in the future. Another advantage of the presented method is the option to be run on GPU or TPU hardware without much overhead from the user's perspective, allowing for a potential performance increase through massive parallelization.

## ACKNOWLEDGEMENT

The authors would like to acknowledge the support by GSI staff as well as by other colleagues.

## REFERENCES

[1] A. Rost *et al.*, "Performance of the CVD Diamond Based Beam Quality Monitoring System in the HADES Experiment at GSI", presented at the 10th Int. Particle Accelerator Conf. (IPAC'19), Melbourne, Australia, May 2019, paper WEPGW019, this conference.

[2] M. Sapinski *et al.*, "Upgrade of GSI HADES Beamline in Preparation for High Intensity Runs", in *Proc. 8th Int. Particle Accelerator Conf. (IPAC'17)*, Copenhagen, Denmark, May 2017, pp. 2214–2216. doi:10.18429/JACoW-IPAC2017-TUPVA060

[3] Atilim G. Baydin, Barak Pearlmutter, Alexey A. Radul and Jeffrey Siskind, "Automatic differentiation in machine learning: a survey", *Journal of Machine Learning Research* 18, 2018.

[4] Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, 2016.

[5] Connor Schenck and Dieter Fox, "SPNets: Differentiable Fluid Dynamics for Deep Neural Networks", *2nd Conference on Robot Learning*, 2018.

[6] Filipe de A. Belbute-Peres, Kevin A. Smith, Kelsey R. Allen, Joshua B. Tenenbaum and J. Zico Kolter, "End-to-End Differentiable Physics for Learning and Control", *32nd Conference on Neural Information Processing Systems*, 2018.

[7] John Ingraham, Adam Riesselman, Chris Sander and Debora Marks, "Learning Protein Structure with a Differentiable Simulator", *International Conference on Learning Representations*, 2019.

[8] Jonas Degrave, Michiel Hermans, Joni Dambre and Francis Wyffels, "A Differentiable Physics Engine for Deep Learning in Robotics", *Frontiers in Neurorobotics* 13, 2019.

[9] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus and Wojciech Matusik, "ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics", *International Conference on Robotics and Automation*, 2019.

[10] Martín Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems", *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, 2016, pp. 265-283.

[11] Adam Paszke *et al.*, "Automatic differentiation in PyTorch", *Proceedings of NIPS*, 2017.

[12] I. Loshchilov and F. Hutter, "SGDR - Stochastic Gradient Descent with Warm Restarts", *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[13] G. Huang *et al.*, "Snapshot Ensembles: Train 1, Get M for free", *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[14] Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization", *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[15] Thomas Bäck and Hans-Paul Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization", *Evolutionary Computation* 1, 1993.

[16] M. Zambrano-Bigiarini, M. Clerc and R. Rojas, "Standard Particle Swarm Optimisation 2011 at CEC-2013: A baseline for future PSO improvements", *IEEE Congress on Evolutionary Computation*, 2013.

[17] Pavel A. Borisovsky and Anton V. Eremeev, "A Study on Performance of the (1+1)-Evolutionary Algorithm", *Foundations of Genetic Algorithms*, 2003.

[18] B. Doerr, H. Le, R. Makhmara and T. Nguyen, "Fast Genetic Algorithms", *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2017.

[19] J. Rapin and O. Teytaud, "Nevergrad - A gradient-free optimization platform", version 0.2.0, https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[20] S. Appel and S. Reimann, "Beam Line Optimization Using a Derivative-Free Algorithm", presented at the 10th Int. Particle Accelerator Conf. (IPAC'19), Melbourne, Australia, May 2019, paper WEPMP005, this conference.

[21] D. Vilsmeier, "Madplot: A Python Toolbox for Interfacing MADX", version 0.2.5, https://pypi.org/project/madplot, 2018.