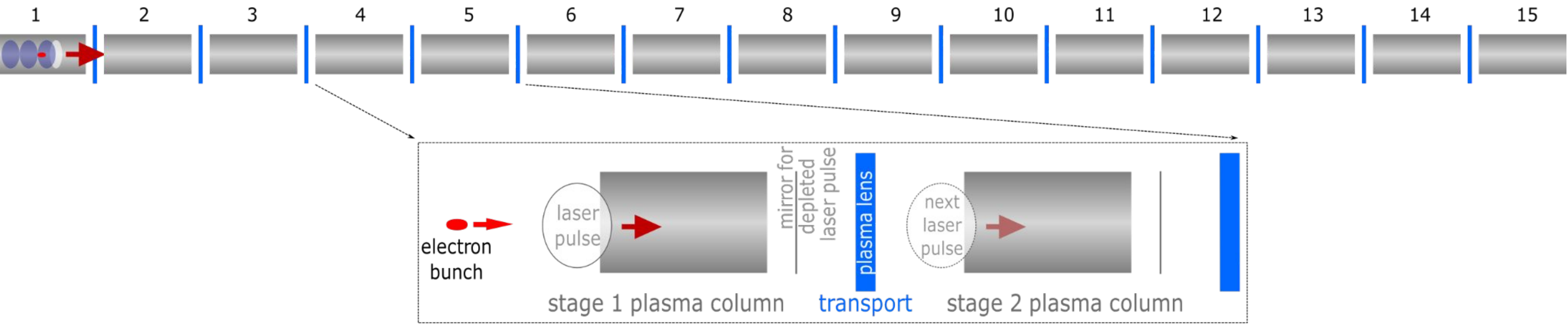
Data-Driven Modeling for Wakefield Colliders – New Capabilities for Integrated RF & Wakefield Modeling in BLAST

Axel Huebl, Ryan T. Sandberg, Remi Lehe, Chad E. Mitchell,
Marco Garten, Andrew Myers, Ji Qiang, Jean-Luc Vay



Thursday July 25th, 2024

Advanced Accelerator Concepts Workshop (AAC24) - Naperville (IL), USA

Advanced Modeling Program
BERKELEY LAB
ACCELERATOR TECHNOLOGY & APPLIED PHYSICS DIVISION
ATAP
U.S. DEPARTMENT OF ENERGY
Office of Science

Abstract (12' + 3' Q&A)

Kinetic simulations of relativistic, charged particle beams and advanced plasma accelerator elements are often performed with high-fidelity particle-in-cell simulations, some of which fill the largest GPU supercomputers. **Self-consistent modeling of wakefield accelerators for colliders includes many elements beyond plasma acceleration.** The integrated Beam, Plasma & Accelerator Simulation Toolkit (**BLAST**) provides high-performance simulation codes suitable to model different parts of a beamline on the latest and world's largest GPU supercomputers. Yet, for some workflows such as **start-to-end modeling and coupling with experimental operations** (digital twins), it is desirable to integrate and model all accelerator elements with **very fast, effective models**. Traditionally, analytical and reduced-physics models fill this role, usually at a cost of lower fidelity and/or reduced dynamics.

Here, we show that the **vast data from high-fidelity simulations** and the power of **GPU-accelerated computation** open a new opportunity to **complement traditional modeling: data-driven surrogate modeling** through **machine learning (ML)**. We present the new capabilities for fully GPU-accelerated, **in-the-loop ML workflows in BLAST** and how they complement and fill a need alongside first-principles modeling and reduced models and **pair** well with recently established out-of-the-loop machine-learning workflows (i.e., **optimization**). We demonstrate that the high-quality data from WarpX simulations can train **low-error surrogate data models**, which are seamlessly integrated into a **GPU beamline simulation using ImpactX**, with the purpose of **minimizing chromatic emittance growth during acceleration and transport in a staged laser-wakefield accelerator** of low beam charge.

Outline

Approaches to wakefield collider modeling

theoretical, first-principle & reduced physics simulation, data

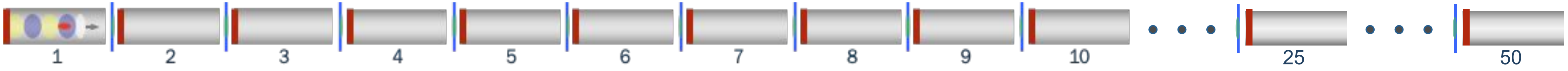
BLAST Codes for Accelerator Modeling

Exascale & ML Technology in WarpX and ImpactX

Staging of LWFA for future HEP colliders

Hybrid beamlines: plasma & transport modeling, ML integration & evaluation

Level of Realism: 50 multi-GeV Stages Modeled in 3D



first 3D simulation of a chain of 50 plasma accelerator stages

LWFA lattice:
- Plasma channels: 28cm
- Gaps: 3cm
- Plasma lens model: linear thick lens (3 mm) w/ “residence correction”

Electron beam:
- Charge: -1 fC
- Size: 0.75 μm x 0.75 μm x 0.1 μm
- Emittance: 1 mm.mrad

Analytical expression to set plasma lens strength:

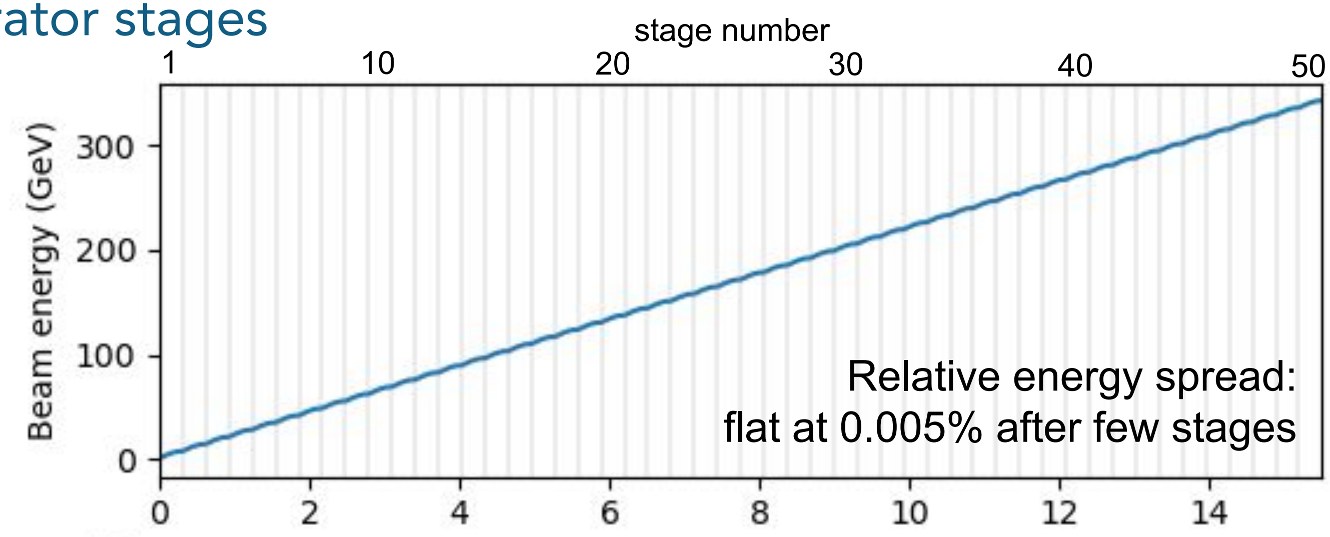
$$\frac{dB_\perp}{dr} = \frac{\langle\gamma\rangle mc^2}{e}k^2 \qquad \tan(kL_{lens}) = \frac{2\langle xx'\rangle}{k\langle x^2\rangle - \langle x'^2\rangle/k}$$

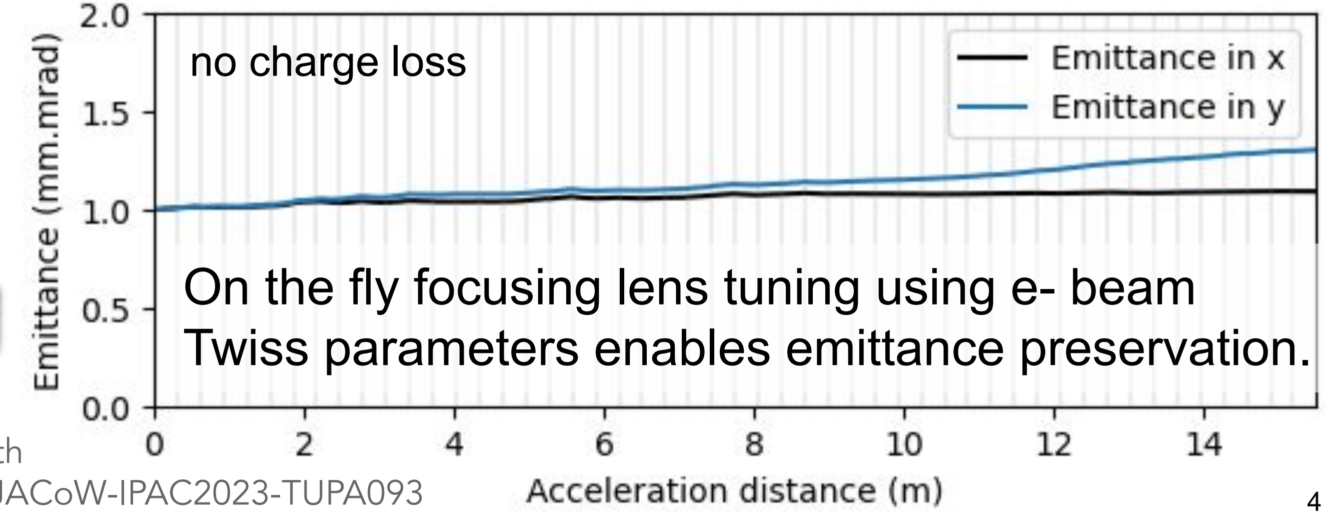
k lens strength, $\langle\cdot\rangle$ moment

Grid size/resolution:
- 128 x 128 x 17664, boosted frame
- 2 μm x 2 μm x 0.01 μm

Computer: 256 GPUs for 8h

WarpX

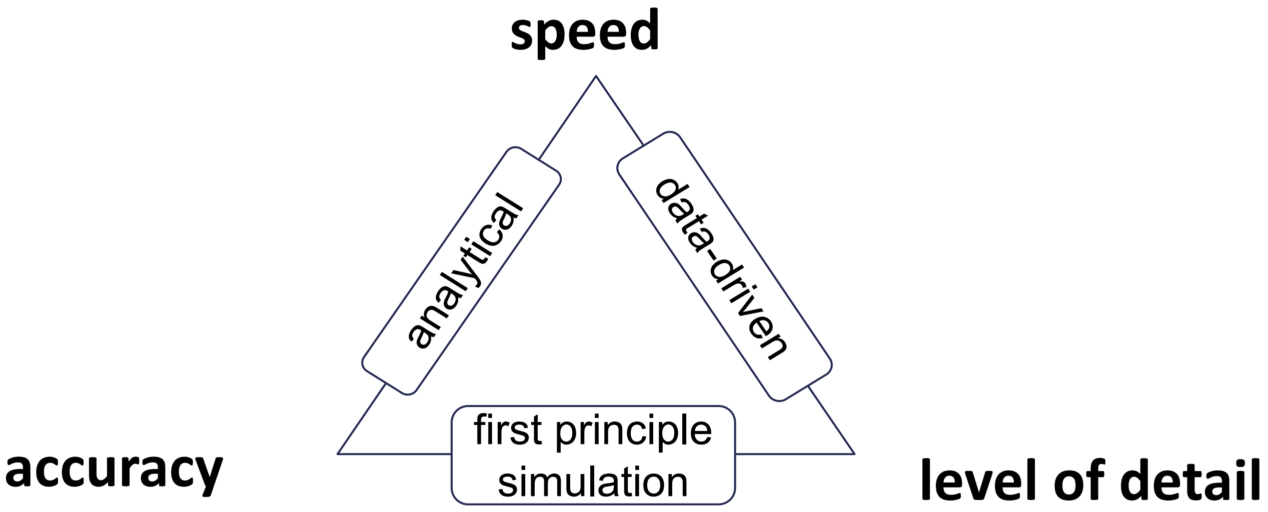




J-L Vay et al., WarpX MS FY23.1 and FY23.2 (2023); work in collaboration with A Ferran Pousa (DESY); A Ferran Pousa et al., IPAC23 (2023) DOI:10.18429/JACoW-IPAC2023-TUPA093

Approaches to Wakefield Collider Modeling

For theoretical modeling of complex, nonlinear many-body systems ***such as a collider*** we develop and evaluate models that have the following **characteristics** - and appear to do at best *two of those well*:



Is *data-driven modeling* via machine-learning surrogates *accurate and fast enough* for
- collider *design:* optimization and/or
- collider *operations:* digital twins ?

BLAST Codes for Accelerator Modeling

Exascale & ML Technology in WarpX and ImpactX

BLAST

BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

WarpX open governance

Developed by an international, multidisciplinary team

over 80 contributors, incl. from the private sector

Open Source Community Codes by Features

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

Code	Year started	Dimensionality	Independent variable	Solver	Symplectic maps	ML surrogate elements	Language	Parallel (multinode)	CPU and GPU	Multi-vendor GPU	Linac	Ring	Source	Wakefield accelerators	Beam-Beam	QED	E-cloud	IBS	CSR	Spin tracking	...
Warp	1989	2/3/RZ++	t/s/ξ	ES/EM/QS			For./Python	✓			✓	✓	✓	✓			✓		3D		...
Impact-Z	1999	3	s	ES	✓		Fortran	✓			✓	✓						✓	1D		...
Impact-T	2002	3	t	ES			Fortran	✓			✓		✓					✓			...
Marylie/IMPACT	2006	3	s	ES	✓		Fortran	✓			✓	✓									...
Posinst	2002	2	t	ES			Fortran										✓				...
BeamBeam3D	2003	2.5	t,s	ES	✓		Fortran	✓				✓			✓						...
FBPIC	2015	RZ++	t	EM			Python	✓	✓					✓						✓	...
LW3D	2018	3	t	LW			Fortran	✓											3D		...
Wake-T	2019	RZ	ξ	QS			Python				✓			✓					1D		...
WarpX	2016	1/2/3/RZ++	t	ES/EM		**	C++/Python	✓	✓	✓	*	**	✓	✓	✓	✓		**	3D	**	...
HiPACE++	2022	3	ξ	QS			C++/Python	✓	✓	✓				✓						✓	...
ImpactX	2022	3	s	ES	✓	✓	C++/Python	✓	✓	✓	✓	✓						**	1D*, ML*	**	...

* in development
** planned, seeking additional funding

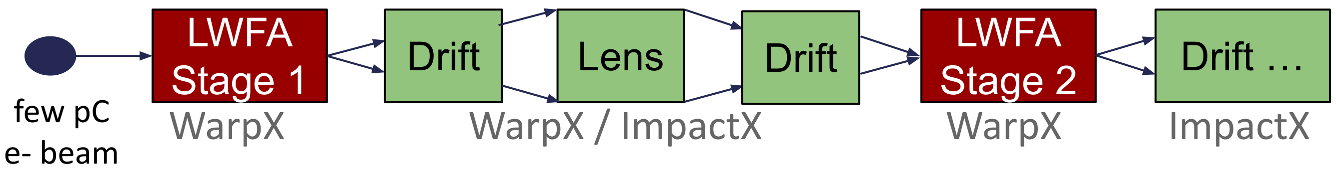
ES=Electrostatic; EM=Electromagnetic; QS=Quasistatic; LW=Lienard-Wiechert; ML=Machine Learning Model

Desktop to HPC

Integrated through Standards & Workflows

Data	openPMD
Input	PICMI Standard
Lasers	LASY
Optimize	optimas

Model Speed: for accelerator elements



Simulation time: full geometry, full physics

hrs	<sec
256 GPUs	1 GPU

Augmenting & GPU-accelerating PIC Simulations & ML Models

GPU Workflows are blazingly fast

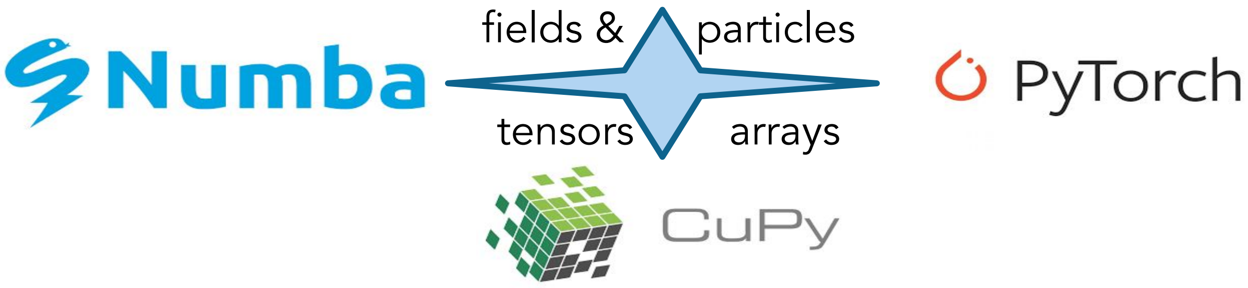
- first-principle models: PIC simulations
- data-driven models: machine learning

We augmented & accelerated on-GPU PIC simulations with on-GPU ML models!

```python
from impactx import ImpactXParIter
import torch

# loop over AMReX particle tiles
for pti in ImpactXParIter(...):
    soa = pti.soa().to_xp()  # view
    x = soa.real["x"]        # alias
    # ... y, t, py, py, pt ...

    data_arr = torch.tensor(  # SoA -> Tensor AoS
        stack([x, y, t, px, py, pt], axis=1),
        device=device,
        dtype=torch.float64,
    )
    # ... normalize data_arr ...

    with torch.no_grad():  # apply NN in-memory
        surrogate_model(data_arr)
```

Compatible ecosystem between:

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT



Persistent GPU data placement

- read+write access, no CPU transfer

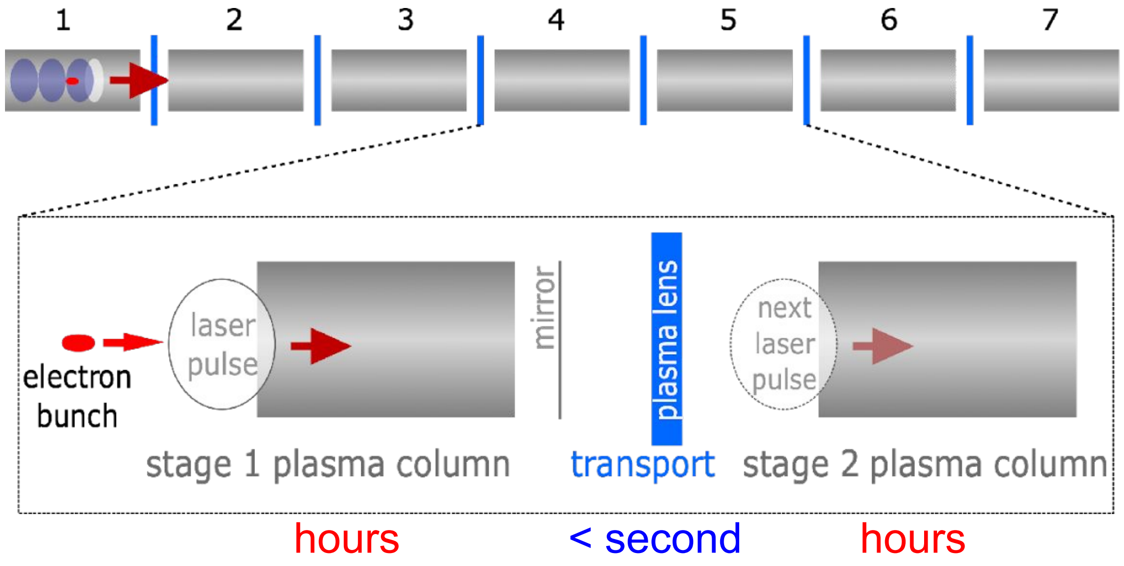
Cross-Ecosystem, In Situ Coupling:
Consortium for Python Data API Standards *data-apis.org*

Staging of LWFA for future HEP colliders

Hybrid beamlines: plasma & transport modeling, ML integration & evaluation

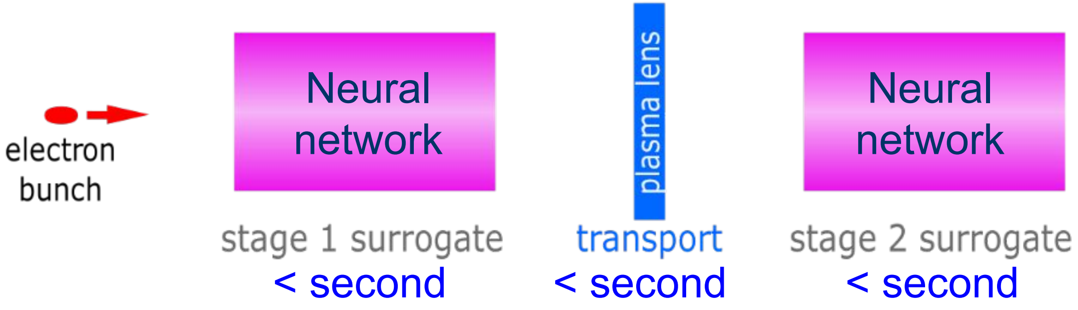
LPA surrogate models bridge runtime gap

Our goal is to find better transport: Combine Plasma & RF Accelerator Elements for start-to-end modeling



- *high-quality plasma simulation* can be expensive
  - 15 stages: **1,316 A100 GPUhrs**
  - 3D electromagnetic, fully kinetic, 128x128x35328 cells
- *optimization challenge*
  - usually 1000s of runs (derivative-free)
  - repeated evaluation *in 3D* would be ***very expensive***
- approach: **specialized replacements**
  - *in situ* coupling of **ImpactX** simulation with **data-driven surrogates**
  - train surrogate models from **high-quality WarpX data**

tightly-coupled LPA-*neural networks* inside **ImpactX**



Related works (CPU): Edelen et al. (2020), Djordjevic et al. (2021), Koser et al. (2022), Badiali et al. (2022)
Concurrent work (1 GPU, differentiable), NN model integration: J Kaiser et al., Phys. Rev. Accel. Beams 27, 054601, May 28th (2024)

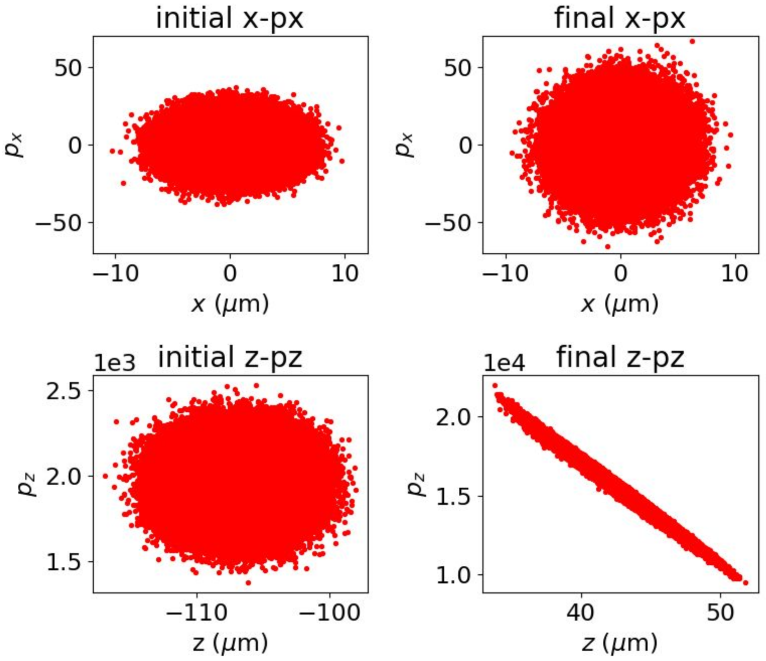
Surrogate models learn initial ⇨ final phase space map from LPA stage data generated by a high-fidelity WarpX simulation

Surrogate model: Generic Transport Map

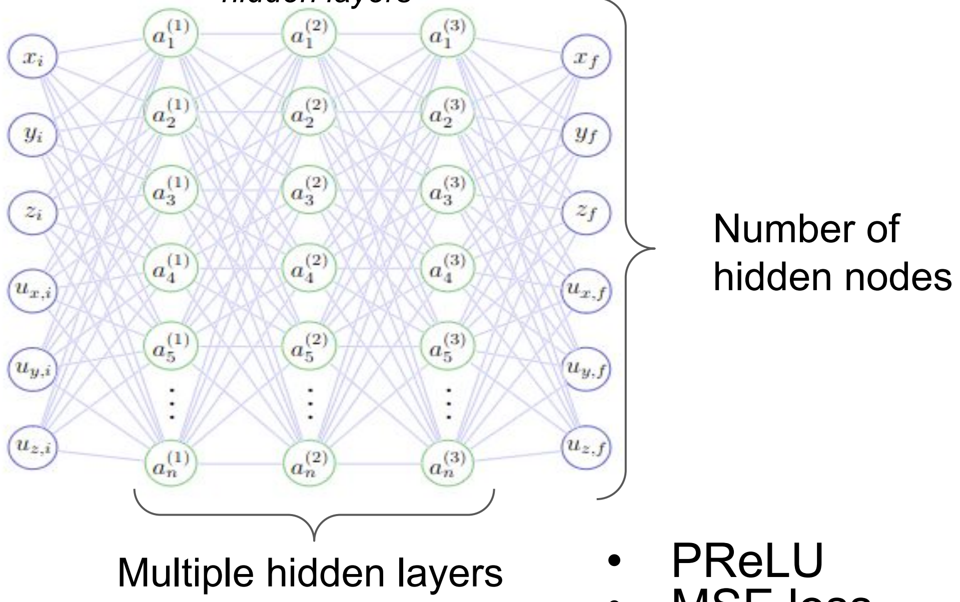
Initial → final phase space

$$\begin{pmatrix} x \\ y \\ z \\ p_x \\ p_y \\ p_z \end{pmatrix}_i \qquad f : \mathbb{R}^6 \rightarrow \mathbb{R}^6$$

example: stage 1 training data



Example of neural network with three hidden layers



- PReLU
- MSE loss
- Adam optimizer

Stages 1-3:	5 hidden layers, 900 nodes per layer
Stages 4-15:	3 hidden layers, 700 nodes per layer

RT Sandberg et al., WEPA101, IPAC23 (2023)

Surrogate models learn initial ⇨ final phase space map from LPA stage data generated by a high-fidelity WarpX simulation

Surrogate model: Generic Transport Map

$$\begin{pmatrix} x \\ y \\ z \\ p_x \\ p_y \\ p_z \end{pmatrix}_i$$

Initial → final phase space

$$f : \mathbb{R}^6 \rightarrow \mathbb{R}^6$$

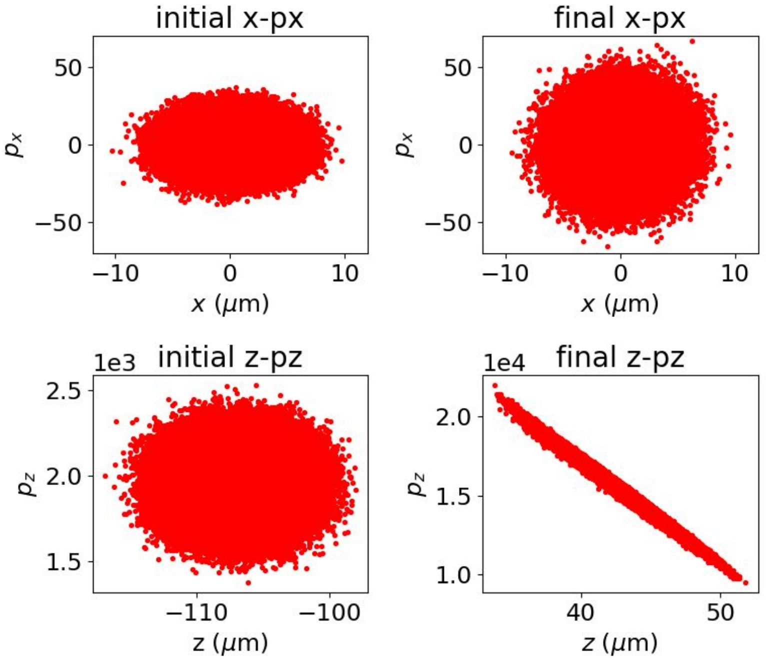
supports beams with

✔arbitrary profiles

✔chromatic effects

✗ collective effects

example: stage 1 training data

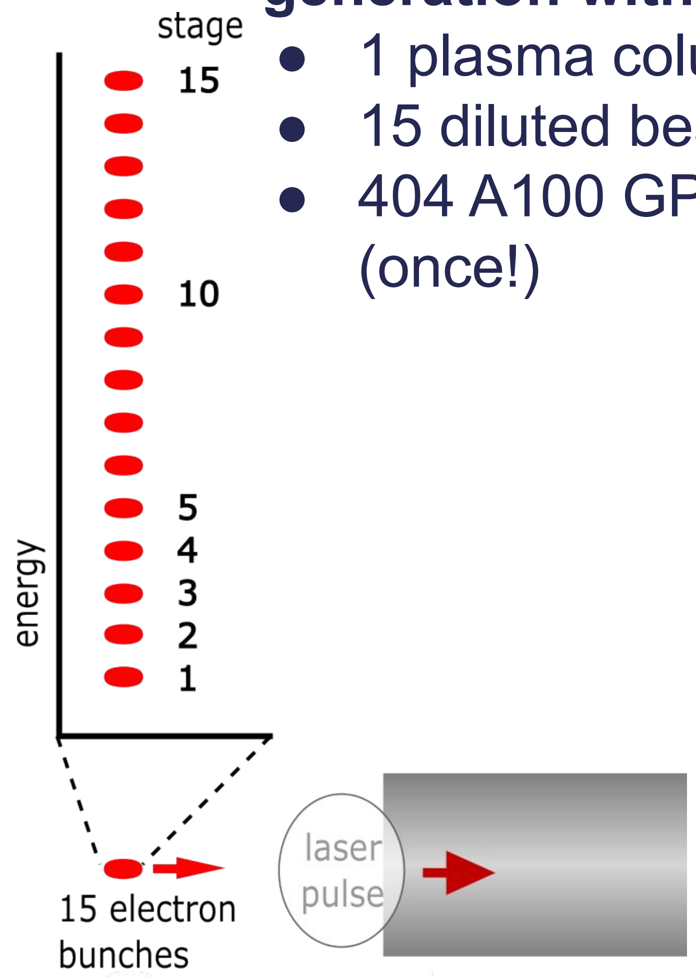


Notes:

- *intentional* choice
- very easy to modify models from Python
- *ideal ground for ML model development*

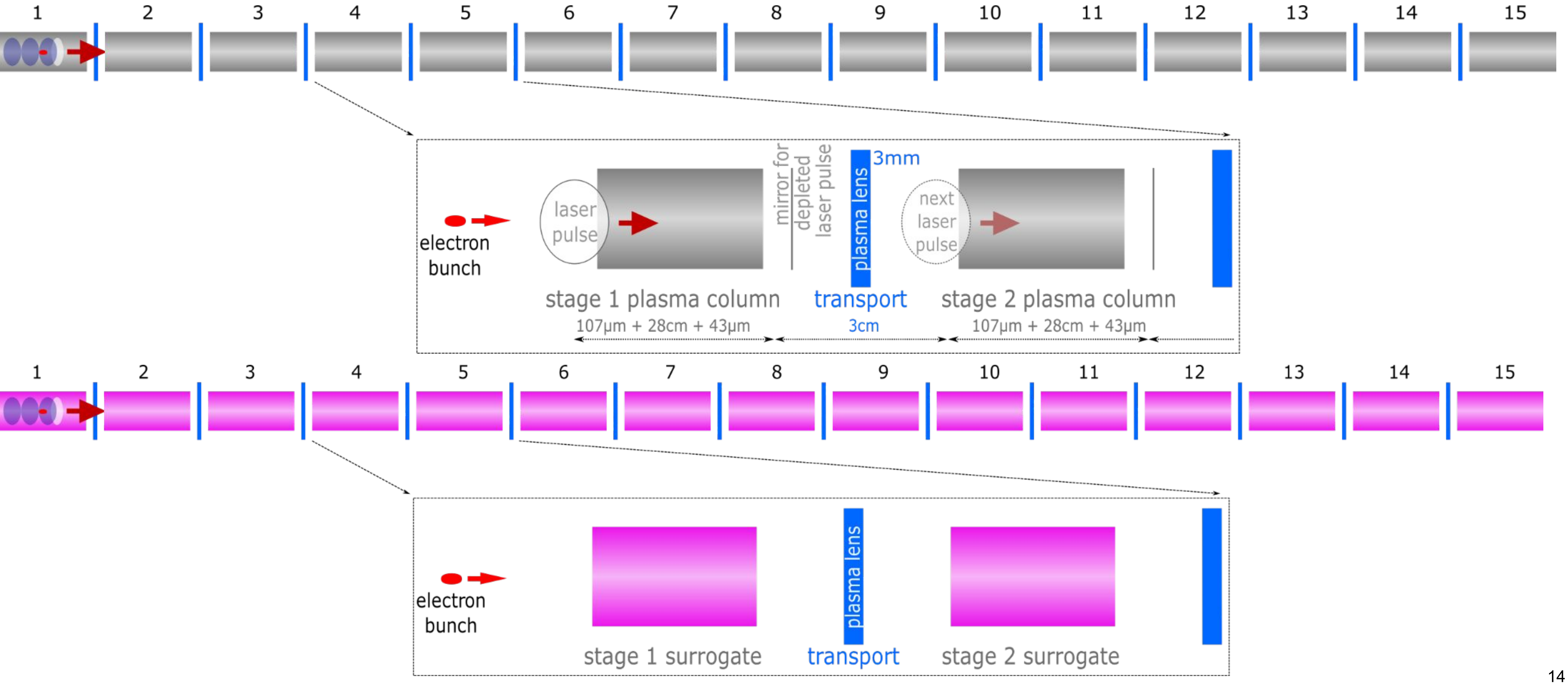
Training Data generation with WarpX

- 1 plasma column
- 15 diluted beams
- 404 A100 GPUhrs (once!)

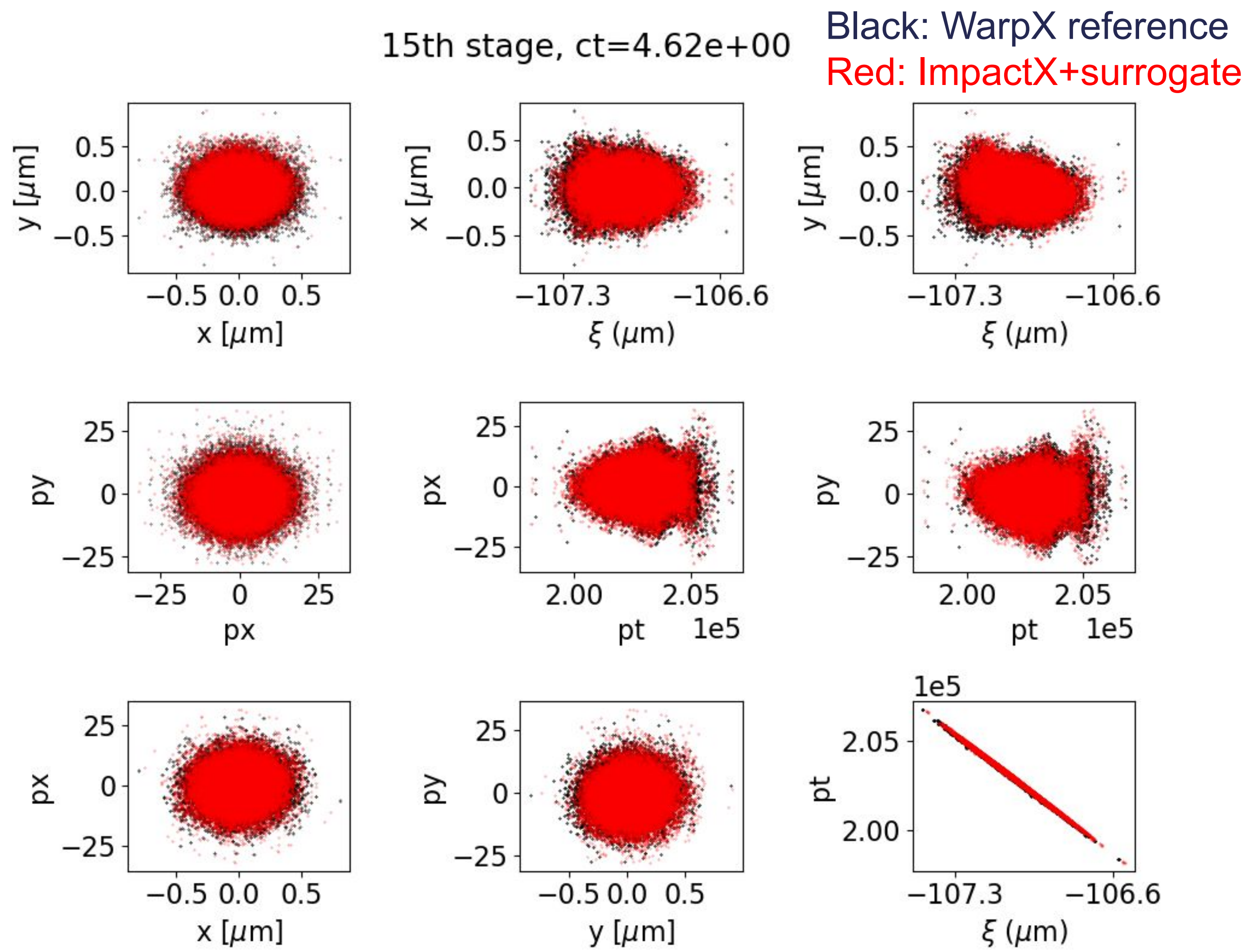


RT Sandberg et al., WEPA101, IPAC23 (2023)

Evaluation: Synthesis of ImpactX and WarpX-trained surrogate models



ImpactX+WarpX surrogate agrees with WarpX reference after 15 stages

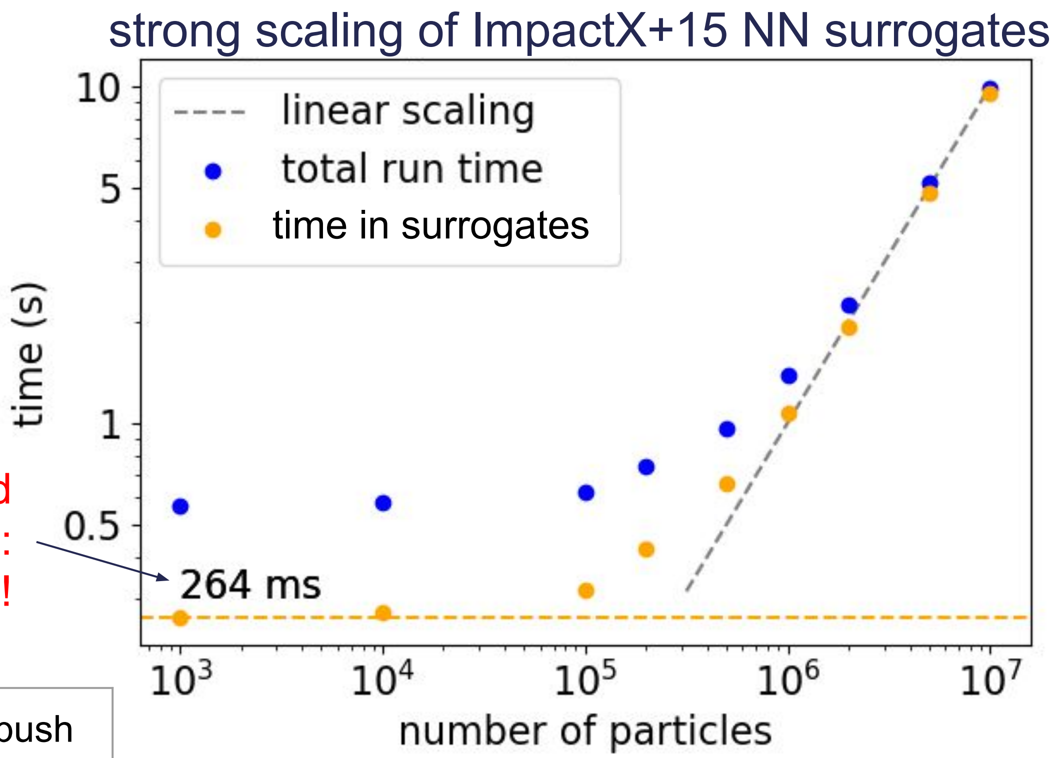


Relative errors in beam moments

	stage 1	stage 2	stage 15
σ_x	0.12%	1.8%	3.2%
σ_{px}	0.54%	2.1%	2.8%
ε_x	0.43%	0.38%	0.39%
σ_y	0.03%	1.5%	1.2%
σ_{py}	0.3%	1.9%	3.2%
ε_y	0.3%	0.44%	2.1%

Modeling + ML Inference are fully GPU accelerated, approaches linear strong scaling in number of particles

Note: NN inference needs significant memory. Surrogates of 15 stages did fit into 80GB A100 GPU memory.



ImpactX with WarpX-trained surrogates: 10 GPU sec for 15 stages

ImpactX with WarpX-trained surrogates: 2-4 simulations / second!

10^7 particles	Time (ms)	% of push
Stage 15 Push	495	100
Inference	477	96.4
Data Preparation	18	3.6

10^3 particles	Time (ms)	% of push
Stage 15 Push	2.77	100
Inference	0.77	27.8
Data Preparation	2.00	72.2

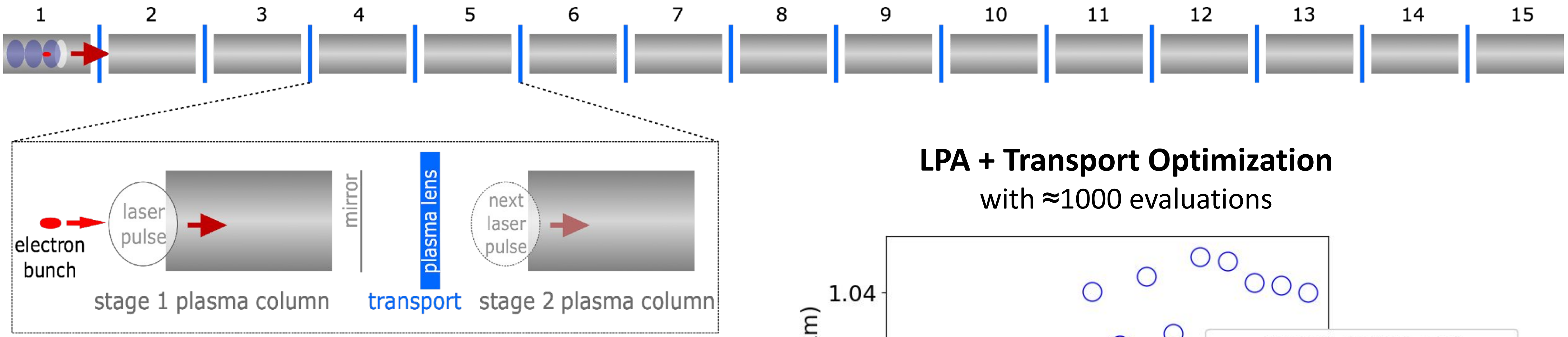
GPU inference time: 63ns / particle / stage

ImpactX tracking >1M particles

Rapid Optimization with Surrogates: Results Transfer to 3D WarpX

Central BLAST Code Interoperability: Combine Plasma & RF Accelerator Elements for start-to-end modeling

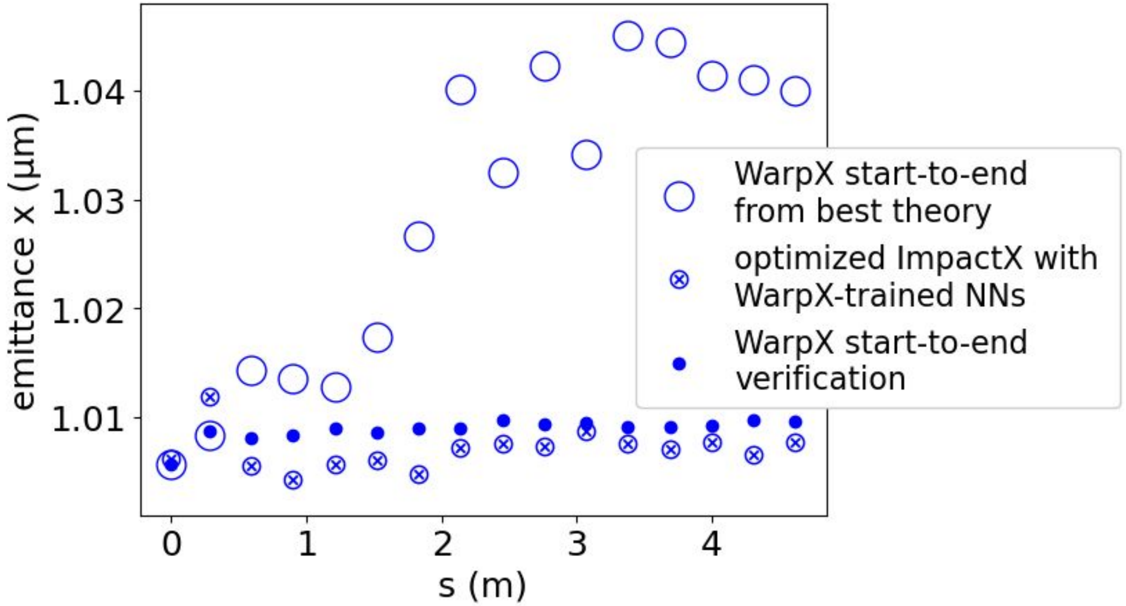
high-quality, first-principle ***WarpX data*** used for ***ImpactX*** ML surrogate training



tightly-coupled LPA-*neural networks* inside **ImpactX**

LPA + Transport Optimization

with ≈1000 evaluations



≈752x estimated cost savings with in-the-loop ML optimization workflow

Previously (Estimate)

1500 GPU hours simulation
x 1000 iterations

+ 1500 GPU hours validation simulation

= 1 501 500 GPU hours

Optimization with in-the-loop ML surrogate model

450 GPU hours training simulation
+ 3 GPU hours PyTorch training
x 15 stages
+ 10 GPU seconds ImpactX+NN
x 1000 iterations
+ 1500 GPU hours validation simulation

= 1 998 GPU hours

In-the-loop Machine Learning Surrogates Beyond Single-Particle Tracking Maps

- **$R^6 \rightarrow R^6$ surrogate**: intentional choice, for the detailed study of **chromatic effects**
  - high level of detail, *arbitrary* low-charge phase spaces, conserves the *phase* of each particle
  - *drop-in* replacement for single-particle, first-principle models

Examples to **include collective effects** in ML surrogates:

- 🔨 **double down**: trajectory + collective beam parameters $R^{6+m} \rightarrow R^{6+m}$
  - how: expose additionally *m* collective beam parameters to ML model for various beam charges
  - note: very costly learning phase, unless constrained (e.g., only change 1D current profile)
- 📽️ **project**: learn & predict phase spaces
  - how: learn & predict selected 2D phase spaces for various beam charges
  - note: less detailed; resampling loses phase, e.g., for tune calculations in rings
  - e.g., Emma et al, PRAB 21, 112802 (2018); Edelen et al., TUPS72, IPAC24 (2024)
- 🌱 **simplify**: work with beam moments and simpler distributions
  - how: learn & predict *only* collective beam parameters, learn simpler distributions (e.g., KV)
  - note: little detail; resampling loses phase, e.g., for tune calculations in rings
  - e.g., Edelen et al., PRAB 23, 044601 (2020); Garcia-Cardona & Scheinker, PRAB 27, 024601 (2024)

These and your own ML ideas can now easily be implemented (Python) & studied in BLAST codes WarpX/ImpactX - see our documentation and detailed examples on how to get started 🚀

In-the-loop Machine Learning Surrogates Beyond Single-Particle Tracking Maps

ImpactX
latest

Search docs

Code of Conduct
Acknowledge ImpactX

INSTALLATION
Users
Developers
HPC

USAGE
Run ImpactX
Examples
Single Particle Dynamics
Space Charge
Beam Distributions
Lattice Design & Optimization
Virtual Test Stands
Cyclotron
The "bare" linear lattice of the Fermilab IOTA storage ring
The full nonlinear lattice of the Fermilab IOTA storage ring
Positron Channel
15 Stage Laser-Plasma Accelerator Surrogate
Run
Analyze
Visualize

Examples / 15 Stage Laser-Plasma Accelerator Surrogate — Edit on GitHub

15 Stage Laser-Plasma Accelerator Surrogate

This example models an electron beam accelerated through fifteen stages of laser-plasma accelerators with ideal plasma lenses providing the focusing between stages. For more details, see:

- Sandberg R T, Lehe R, Mitchell C E, Garten M, Myers A, Qiang J, Vay J-L and Huebl A. **Synthesizing Particle-in-Cell Simulations Through Learning and GPU Computing for Hybrid Particle Accelerator Beamlines.** Proc. of Platform for Advanced Scientific Computing (PASC'24), *submitted*, 2024. arXiv:2402.17248
- Sandberg R T, Lehe R, Mitchell C E, Garten M, Qiang J, Vay J-L and Huebl A. **Hybrid Beamline Element ML-Training for Surrogates in the ImpactX Beam-Dynamics Code.** 14th International Particle Accelerator Conference (IPAC'23), WEPA101, 2023. DOI:10.18429/JACoW-IPAC2023-WEPA101

A schematic with more information can be seen in the figure below:

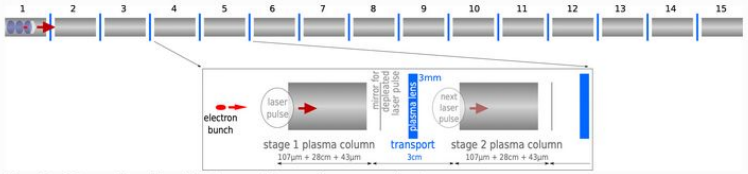


Fig. 10 *Schematic of the 15 stages of laser-plasma accelerators.*

The laser-plasma accelerator elements are modeled with neural networks as surrogates. These networks are trained beforehand. In this example, pre-trained neural networks are downloaded from a Zenodo archive and saved in the `models` directory. For more about how these neural network surrogate models were created, see this description of a workflow for training neural networks from WarpX simulation data.

Developers
HPC

USAGE
Run WarpX
Examples
Parameters: Python (PICMI)
Parameters: Inputs File
Workflows
Extend a Simulation with Python
Domain Decomposition
Visualizing a distribution mapping
Debugging the code
Generate QED lookup tables using the standalone tool
Plot timestep duration
Predicting the Number of Guard Cells for PSATD Simulations
Archiving
Training a Surrogate Model from WarpX Data
Data Cleaning
Create Normalized Dataset
Neural Network Structure
Train and Save Neural Network
Evaluate
Optimizing with Optimas
FAQ

DATA ANALYSIS
Output formats

Workflows / Training a Surrogate Model from WarpX Data — Edit on GitHub

Training a Surrogate Model from WarpX Data

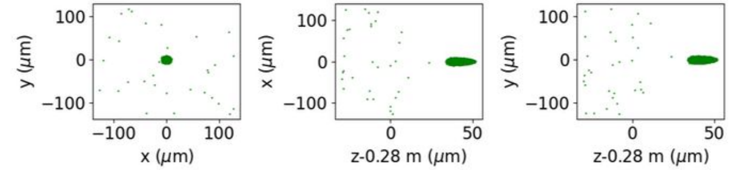
Suppose we have a WarpX simulation that we wish to replace with a neural network surrogate model. For example, a simulation determined by the following input script

Python Input for Training Simulation

In this section we walk through a workflow for data processing and model training, using data from this input script as an example. The simulation output is stored in an online Zenodo archive, in the `lab_particle_diags` directory. In the example scripts provided here, the data is downloaded from the Zenodo archive, properly formatted, and used to train a neural network. This workflow was developed and first presented in Sandberg *et al.* [1], Sandberg *et al.* [2]. It assumes you have an up-to-date environment with PyTorch and openPMD.

Data Cleaning

It is important to inspect the data for artifacts, to check that input/output data make sense. If we plot the final phase space of the particle beam, shown in Fig. 18. we see outlying particles. Looking closer at the z-pz space, we see that some particles were not trapped in the accelerating region of the wake and have much less energy than the rest of the beam.

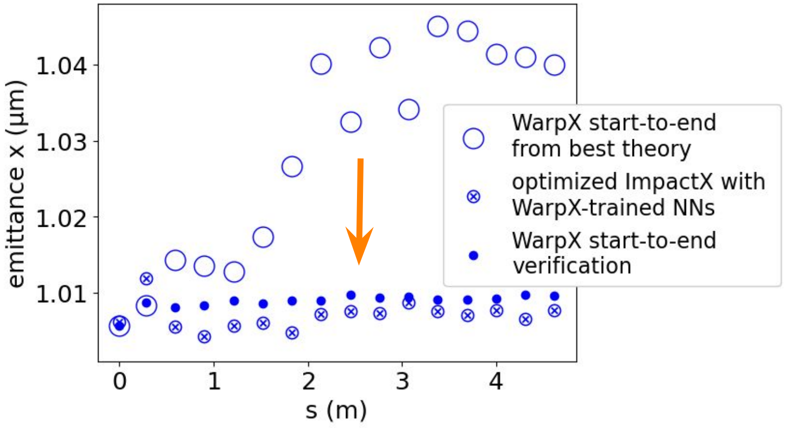


These and your own ML ideas can now easily be implemented (Python) & studied in BLAST codes WarpX/ImpactX - see our documentation and detailed examples on how to get started 🚀

Summary

A fast, high-fidelity, data-driven LPA staging workflow with ImpactX+surrogate models

- ***neural network surrogates*** reproduce unloaded LPA simulations with % level error
- runs in ***seconds*** – optimization workflow gets ***$\mathcal{O}$(1000) speedup***
- best ImpactX+surrogate transport parameters ***readily transfer*** to 3D WarpX simulations
  - **emittance significantly improved** ● for **15 stages** to ○ prior best results



Established data-driven methods in BLAST codes WarpX & ImpactX

- kinetic codes & in situ ML elements: easy to **test & study** new data models
- fully **accelerated** (GPU or CPU), fully **documented**
- vibrant, friendly & helpful **open source community** - we invite you to join

bring your own lattice & ML model

Follow-up work on achromatic transport already underway in collaboration with C Lindstrøm et al., Oslo (2024).

Thank you for your attention!
Try it yourself:

open source initiative®

ECP-WarpX/WarpX
ECP-WarpX/impactX
AMReX-Codes/pyamrex

Paper: R. Sandberg et al.,
PASC24 *Best Paper (2024)*
🔗 DOI:10.1145/3659914.3659937

Documented example links:
🔗 WarpX ML training from openPMD
🔗 ML Surrogates in ImpactX

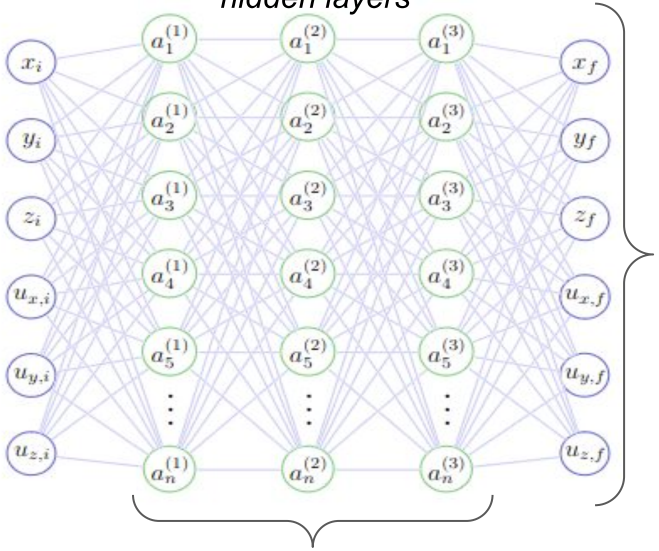
We acknowledge the **ImpactX, WarpX, AMReX** and **pyAMReX** open source communities for their invaluable contributions. In addition, we acknowledge and thank **Alexander Sinn** and **Thierry Antoun** for their contributions to particle data structures. We thank Carl Schroeder, Eric Esarey, Carlo Benedetti and Davide Terzani for discussions. This work was supported by the **Laboratory Directed Research and Development Program** of **Lawrence Berkeley National Laboratory** under U.S. Department of Energy Contract No. DE-AC02-05CH11231. This material is based upon work supported by the **U.S. Department of Energy, Office of Science, Office of High Energy Physics, General Accelerator R&D (GARD)**, under contract number DE-AC02-05CH11231. This material is based upon work supported by the **CAMPA collaboration**, a project of the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics, **Scientific Discovery through Advanced Computing (SciDAC)** program. This research was supported by the **Exascale Computing Project** (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative. This research used resources of the **National Energy Research Scientific Computing Center**, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award HEP-ERCAP0023719.

Backup Slides

Hyperparameter tuning indicated that relatively simple neural networks were sufficiently accurate

Model of a single stage

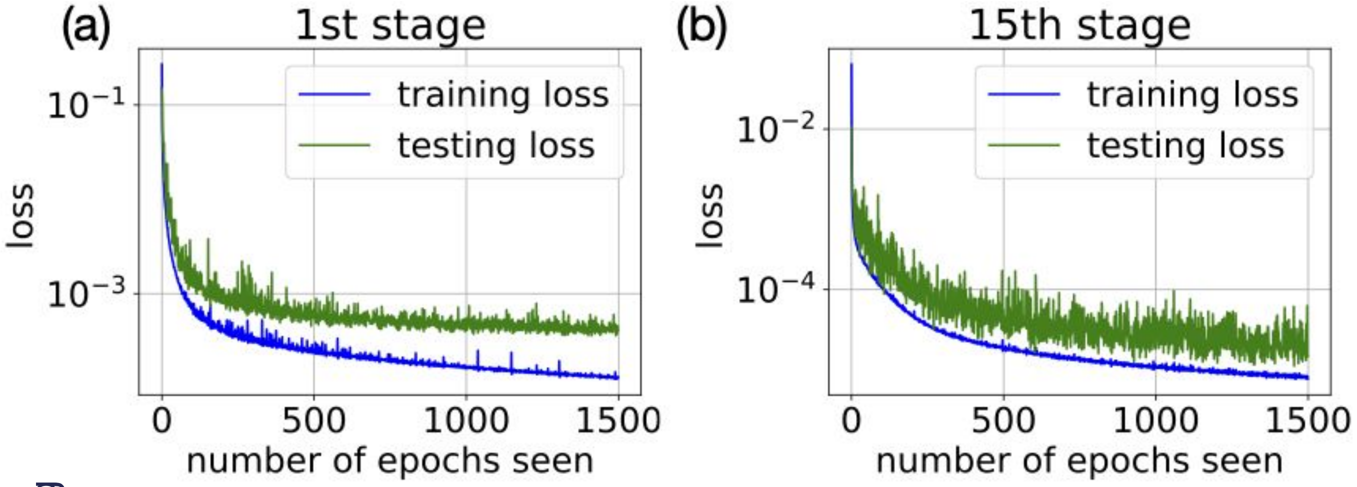
Example of neural network with three hidden layers



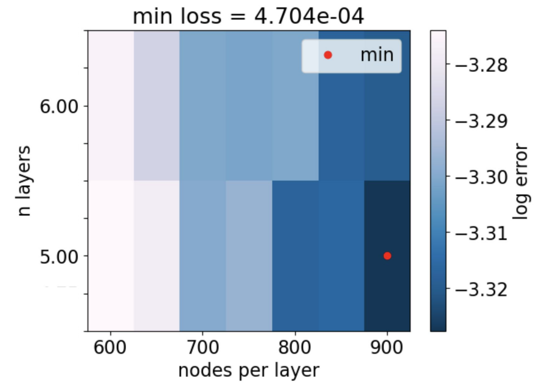
Number of hidden nodes

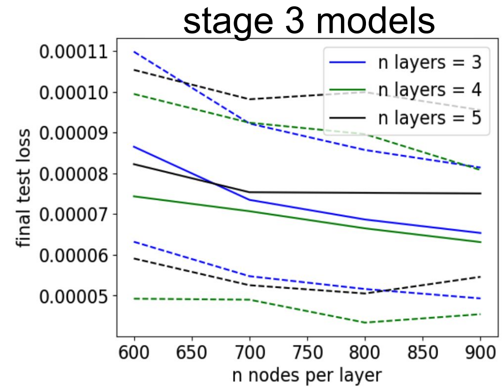
Multiple hidden layers

implemented in PyTorch
- PReLU
- MSE loss
- Adam optimizer



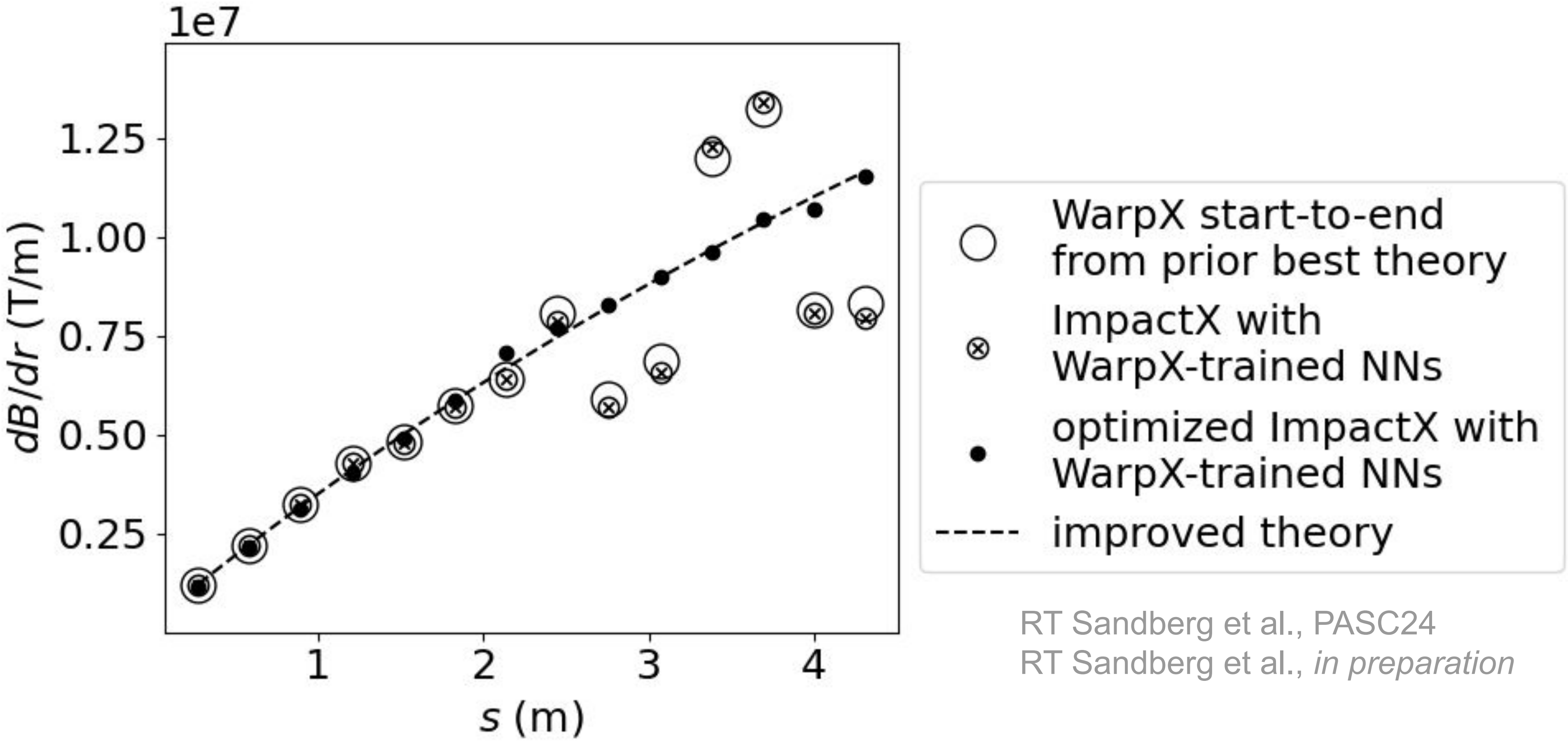
stage 1 hyperparameter tuning





Stages 1-3:	5 hidden layers, 900 nodes per layer
Stages 4-15:	3 hidden layers, 700 nodes per layer

Synthesized Simulation with Optimized Lenses Enabled Development of an Improved Analytical Theory



RT Sandberg et al., PASC24
RT Sandberg et al., *in preparation*

- before: ***analytically***-motivated in situ tuning of lens strength
- now: ***automated*** tuning of ***multiple*** lens parameters
- enables: development & validation of ***new theoretical*** models

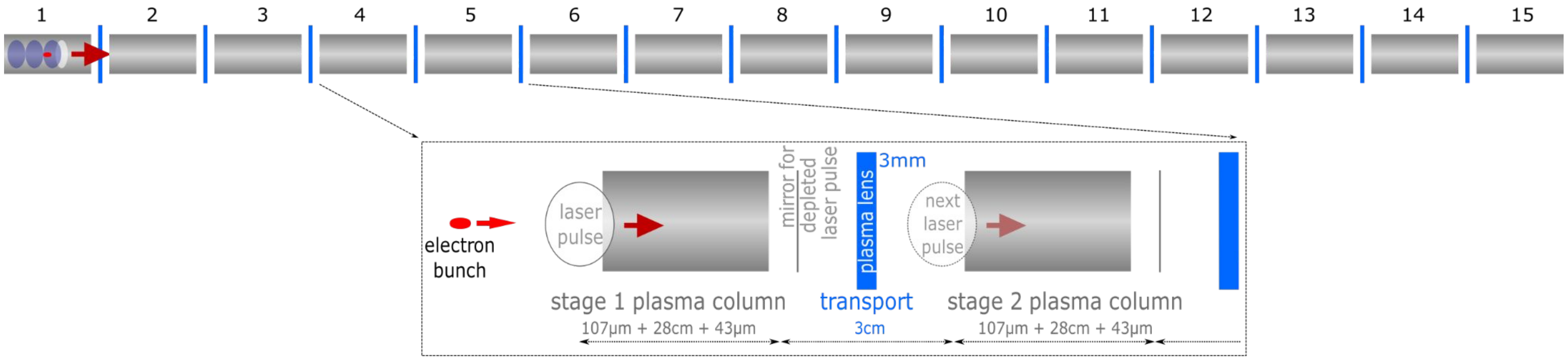
Challenge Problem: Rapid Design Optimization of Inter-Stage Transport

Goal: improve **beam quality** (emittance) after many (15) LPA stages

- focus beam to matching conditions of subsequent stage
- transport complex beams, e.g., with energy spread (chromaticity) without degrading beam quality (emittance, particle loss, energy spread) …

Task: find best interstage **transport parameters** including **chromatic effects**

- transport: plasma lens for beam focusing
- **two parameters** per lens: lens strength and position

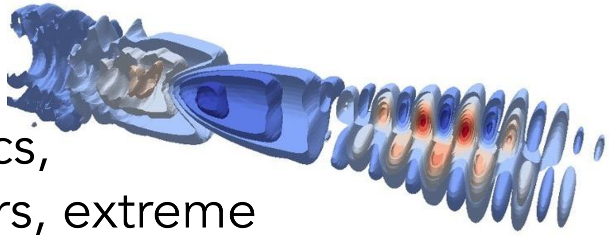


Follow-up work on achromatic transport already underway in collaboration with C Lindstrøm et al., Oslo (2024)

WarpX is a Community Exascale Particle-in-Cell Code

EXASCALE COMPUTING PROJECT

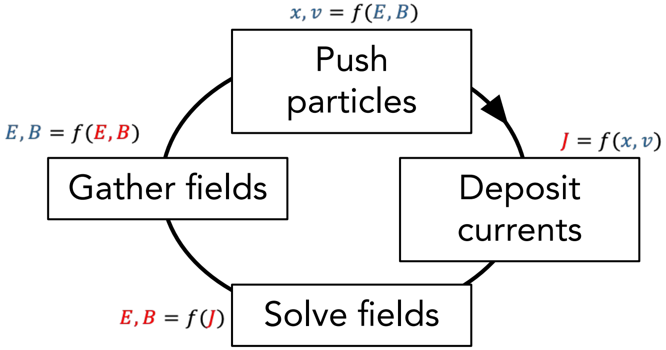
Applications

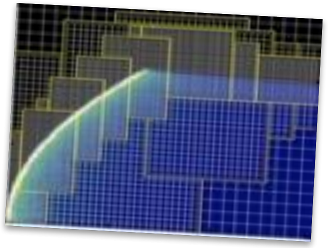


laser-plasma physics,
particle accelerators, extreme
light sources, fusion devices & plasmas, …

Exascale Particle-in-Cell Code

- electromagnetic or electro/magnetostatic
- PIC-fluid hybrid
- time integration: explicit, implicit





International Contributors incl. private sector

BERKELEY LAB · cea PARIS-SACLAY · DESY. · LLNL · loa · CERN · MODERN ELECTRON · tae TECHNOLOGIES · UR LLE · SLAC · AVALANCHE

Award–Winning Code & Science

Portable, Multi-Level Parallelization

- MPI: 3D MR decomposition
  - dynamic load balancing
- GPU: CUDA, HIP and SYCL
- CPU: OpenMP

mac OS

Desktop to HPC

Scalable & Standardized

- Python APIs, openPMD data
- In situ processing
- Open community ecosystem

openPMD · LASY · AMReX · PICMI Standard

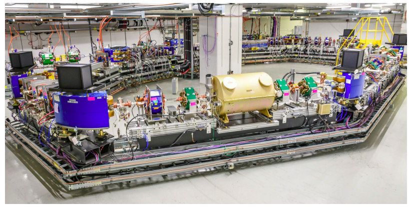
BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

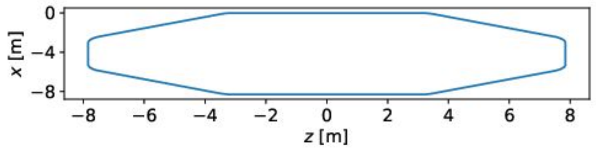
J-L Vay et al., NIMA 909.12 (2018)
L Fedeli, A Huebl et al., SC22, DOI:10.1109/SC41404.2022.00008 (2022)

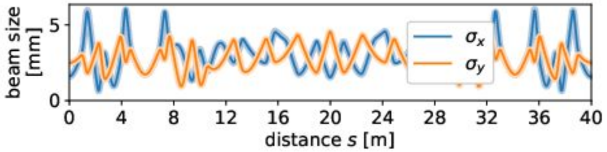
ImpactX: We leverage WarpX Technology for RF Accelerator Modeling

Beam-Dynamics in Linacs, Rings, Colliders

- intense beams, long-term dynamics
- HEP science: FNAL complex evolution, FCC-ee, FCC-hh, muon collider
- **s-based, electrostatic**
  - relative to a reference particle
  - elements: <u>symplectic maps</u>





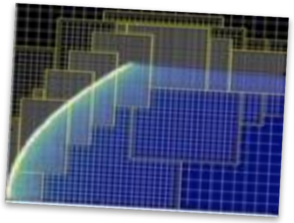


Advanced Numerics

symplectic, based on IMPACT-Z, space charge,
soon: radiative effects (CSR & ISR)

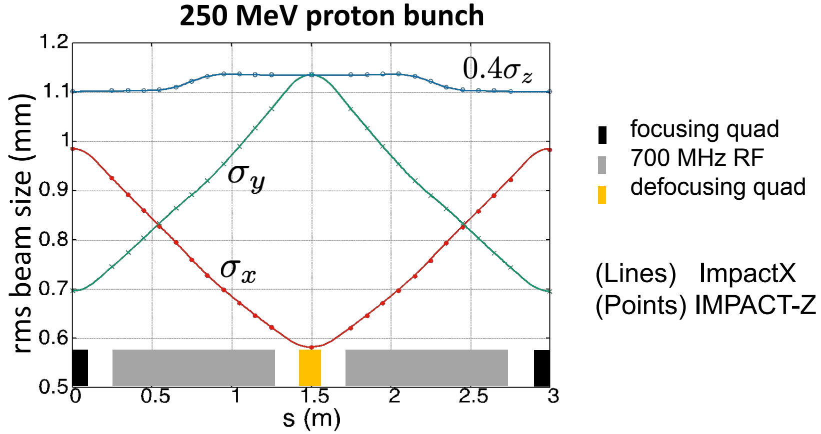
Triple Acceleration Approach

- GPU support
- Adaptive Mesh Refinement
- AI/ML & Data Driven Models



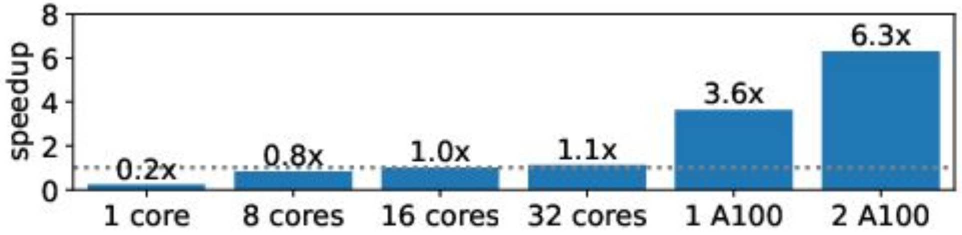
Benchmarks & Validations

- 86 continuously run benchmarks
- code-to-code comparisons



Performance

- order-of-magnitude perf.↗ from GPUs



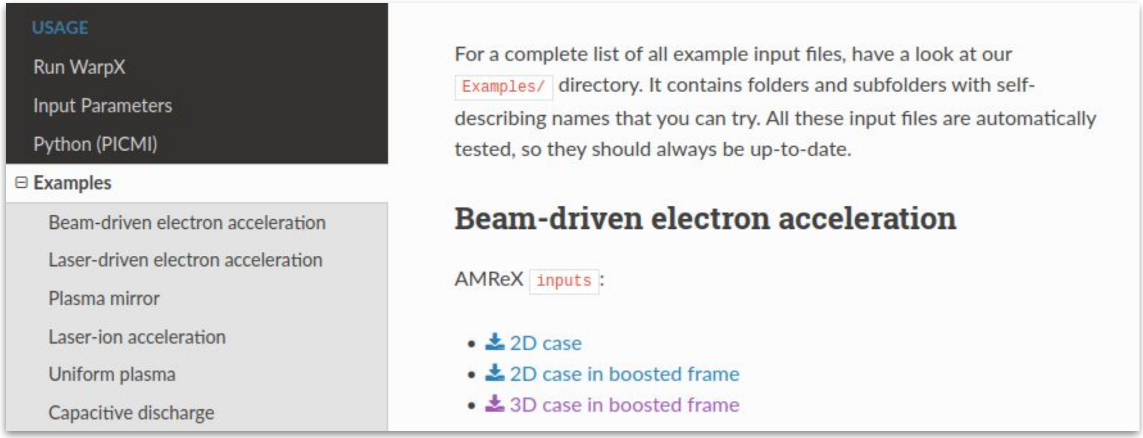
C Mitchell et al., HB2023, THBP44 and TUA2I2 (2023); A Huebl et al., NAPAC22 and AAC22 (2022); J Qiang et al., PRSTAB (2006); RD Ryne et al., ICAP2006 ICAP2006 (2006)

BERKELEY LAB LDRD SciDAC Scientific Discovery through Advanced Computing

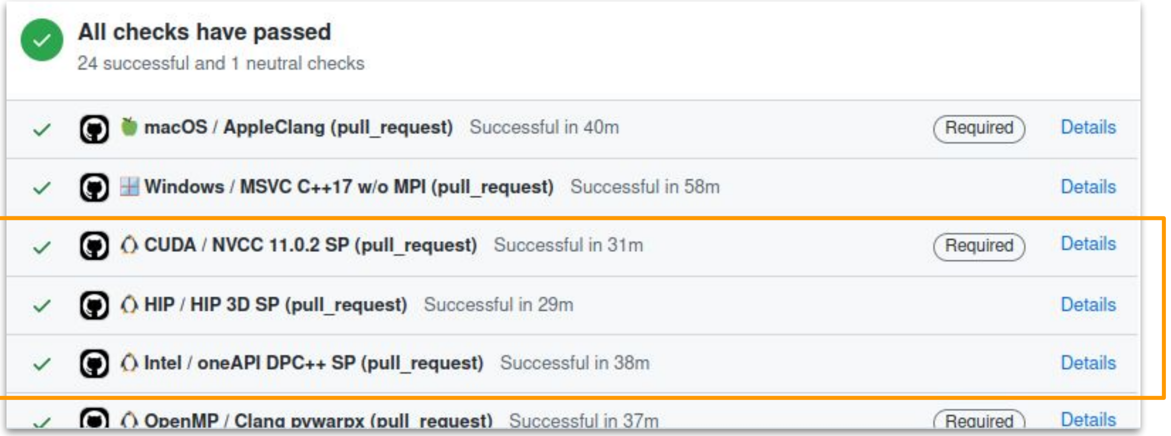
We Develop Openly with the Community

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

Online Documentation:
warpx|hipace|impactx.readthedocs.io



Open-Source Development & Benchmarks:
github.com/ECP-WarpX



230 physics benchmarks *run on every code change* of WarpX
34 physics benchmarks for ImpactX

Rapid and easy installation on any platform:

conda install
-c conda-forge warpx

spack install warpx
spack install
py-warpx

cmake -S . -B build
cmake --build build --target
install

python3 -m pip install .

brew tap ecp-warpx/warpx
brew install warpx

module load warpx
module load py-warpx

Modular Software Architecture

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

Python: Modules, PICMI interface, Workflows

WarpX
full PIC,
LPA/LPI

ImpactX
accelerator lattice design

…

HiPACE++
quasi-static,
PWFA

ARTEMIS
microelectronics

ML Frameworks
PyTorch,
Tensorflow, …

pyAMReX

PICSAR
QED Modules

ABLASTR: shared PIC

AMReX

Containers, Communication,
Portability, Utilities

openPMD
diagnostics

Math

FFTs,
lin. alg.

CUDA, OpenMP, SYCL, HIP

MPI

mac OS

Desktop
to
HPC

GPU-accelerated Synthesis: PIC Simulations & ML Models

Demonstrated profits from GPUs

- *first-principle models:*
  Particle-in-Cell simulations
- *data-driven models:*
  neural network training & inference

Approach

- Creation of a *compatible ecosystem*
- C++ core, Python control/glue
- pure C++ Python bindings: pybind11

Implementation Goals

- **augment & accelerate** *on-GPU* PIC simulations with *on-GPU* ML models
- support many **HPC C++ compilers**
- **rapid ML model design** "plug-and-play"

W Jakob et al., pybind11 – Seamless operability between C++11 and Python (2017)
A Huebl et al., pyAMReX: GPU-Enabled, Zero-Copy AMReX Python Bindings including AI/ML (2023)
A Myers et al., AMReX and pyAMReX: Looking Beyond ECP, under review, arXiv:2403.12179 (2024)

Augmenting & GPU-accelerating PIC Simulations & ML Models

Embracing Emerging API Standards

- here: __cuda_array_interface__

```
{
  'shape': (1,),
  'typestr': '<f8',
  'descr': [('', '<f8')],
  'stream': 1,
  'version': 3,
  'strides': None,
  'data': (136661631501920, False)
}
```

- more general: DLPack

Cross-Ecosystem, In Situ Coupling
Consortium for Python Data API Standards data-apis.org

Compute example

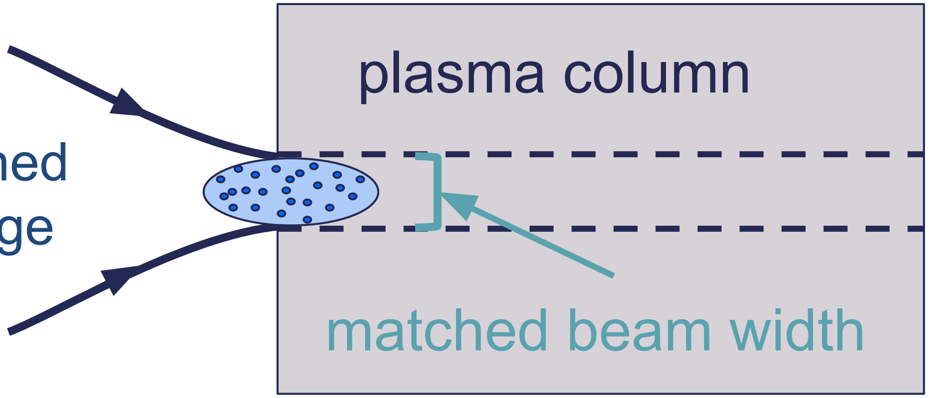
- data shared as views, stays on device
- enables in-memory updates

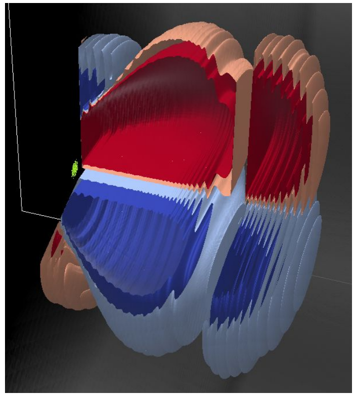
```python
from impactx import ImpactXParIter
import torch

# loop over AMReX particle tiles
for pti in ImpactXParIter(...):
    soa = pti.soa().to_xp()  # view
    x = soa.real["x"]        # alias
    # ... y, t, py, py, pt ...

    data_arr = torch.tensor(  # SoA -> Tensor AoS
        stack([x, y, t, px, py, pt], axis=1),
        device=device,
        dtype=torch.float64,
    )
    # ... normalize data_arr ...

    with torch.no_grad():  # apply NN in-memory
        surrogate_model(data_arr)
```

A Huebl et al., pyAMReX: GPU-Enabled, Zero-Copy AMReX Python Bindings including AI/ML (2023)
A Myers et al., AMReX and pyAMReX: Looking Beyond ECP, under review, arXiv:2403.12179 (2024)

A key challenge to particle accelerator design: suppress emittance growth

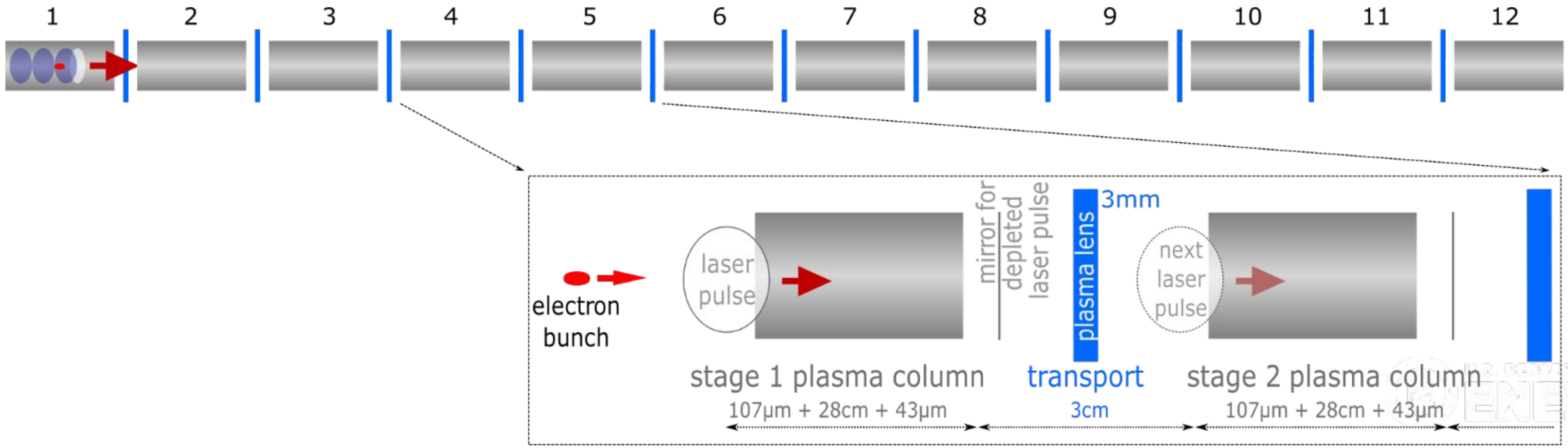
- Within plasma stage
  - emittance preserved if beam width is matched to *transverse focusing forces* in plasma stage





- Transport between stages
  - focus beam to matching conditions of subsequent stage
  - transport complex beams, e.g., with energy spread (chromaticity) without degrading beam quality (emittance, particle loss, energy spread) …

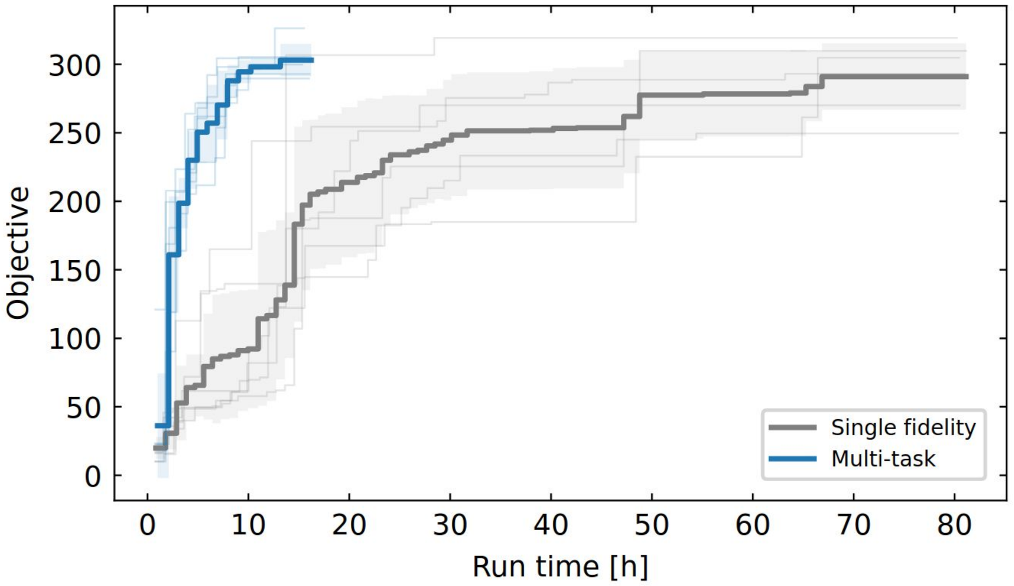
- Demonstrator problem: control emittance growth through 15 stages



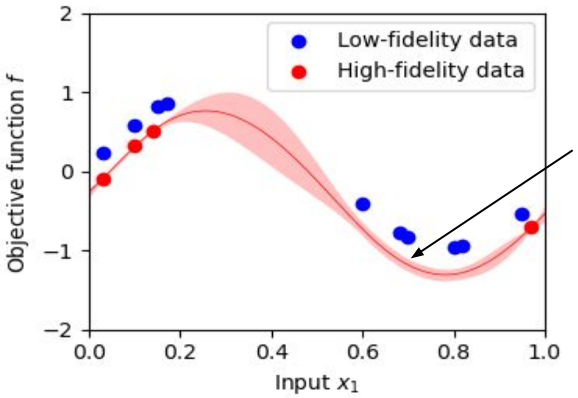
ML-Guided Optimization: Automate Scans & Design Workflows

Design Optimization:

- ML **finds optima rapidly**, e.g. *Gaussian Processes*, *Bayesian Optimization*
- **Multi-Fidelity** (think: multi-resolution): Learn trends from fast simulations and add precision with large costly sims

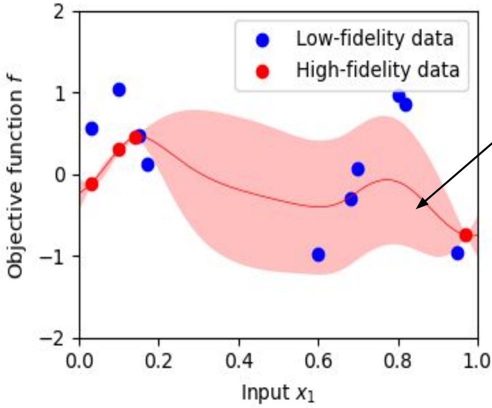


Strongly-correlated case:



Low uncertainty, despite the absence of high-fidelity data

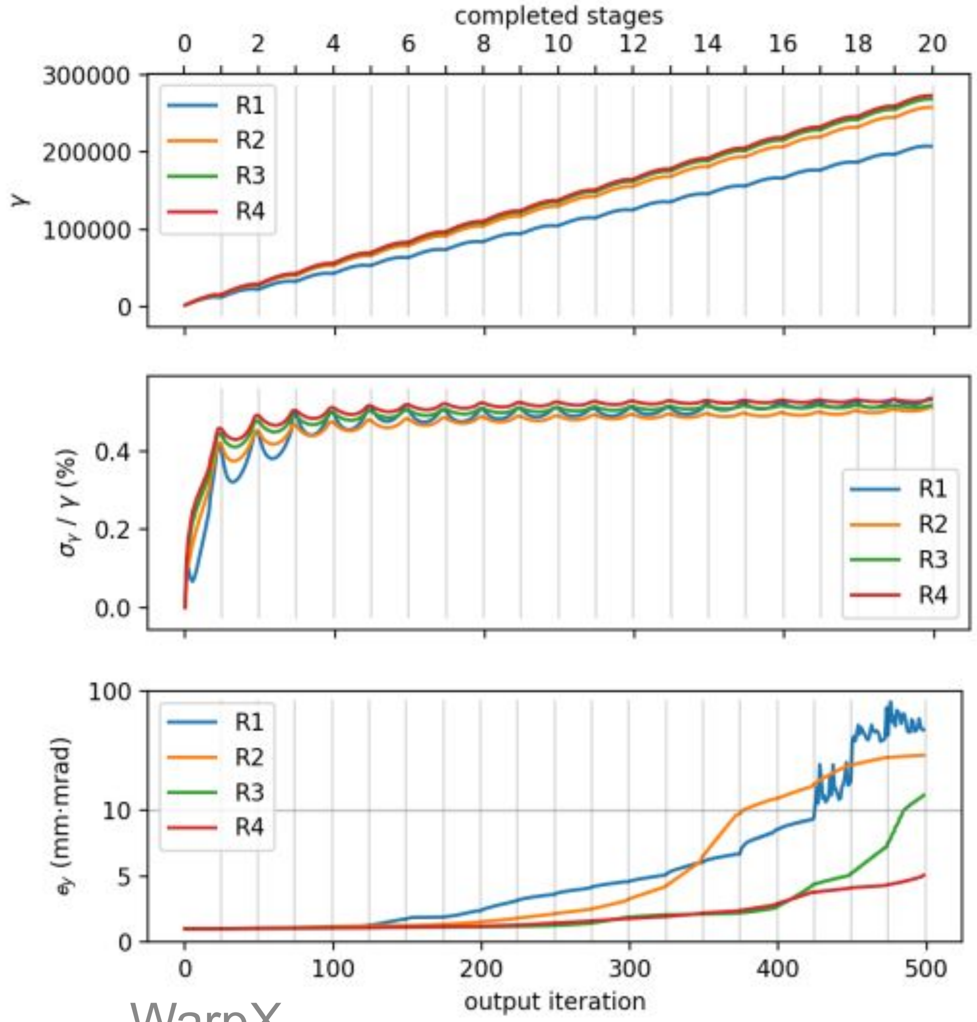
Un-correlated case:



High uncertainty; low-fidelity data is ignored

Bonilla et al., NIPS, (2007); R. Lehe et al., APS DPP (2022); Á. Ferran Pousa et al., IPAC22 (2022) & PRAB (2023)

libEnsemble: Design Optimization with Reduced Models



WarpX

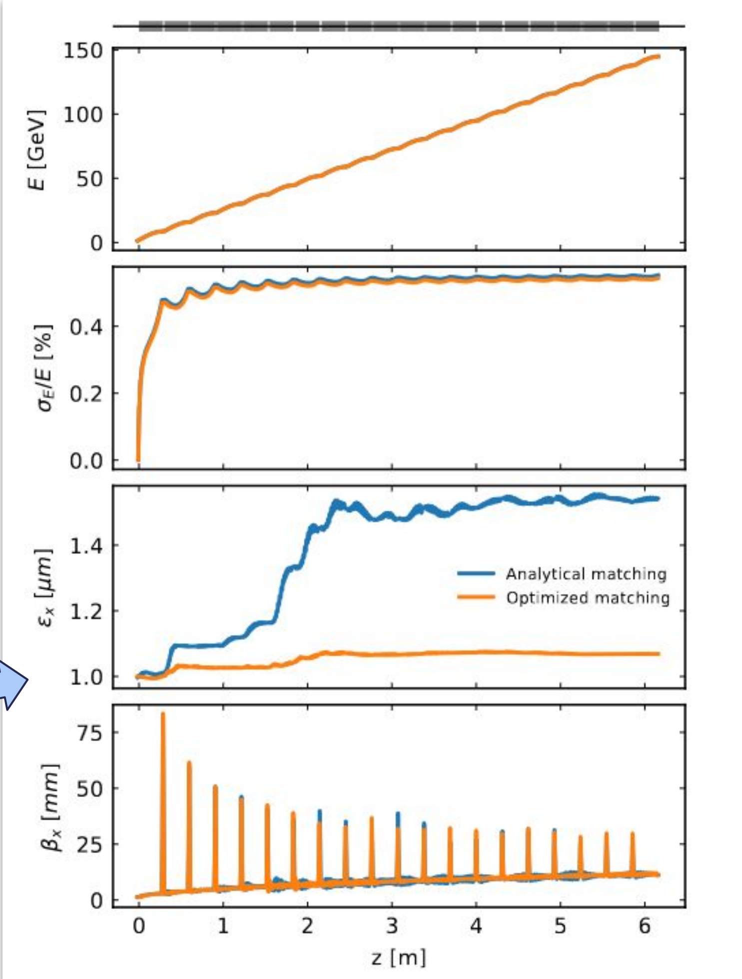
Staged LPA

Beam Emittance Preservation

3. converge
   3D
4. optimize

1. optimize
   low-D, redu.

2. inform
   3D

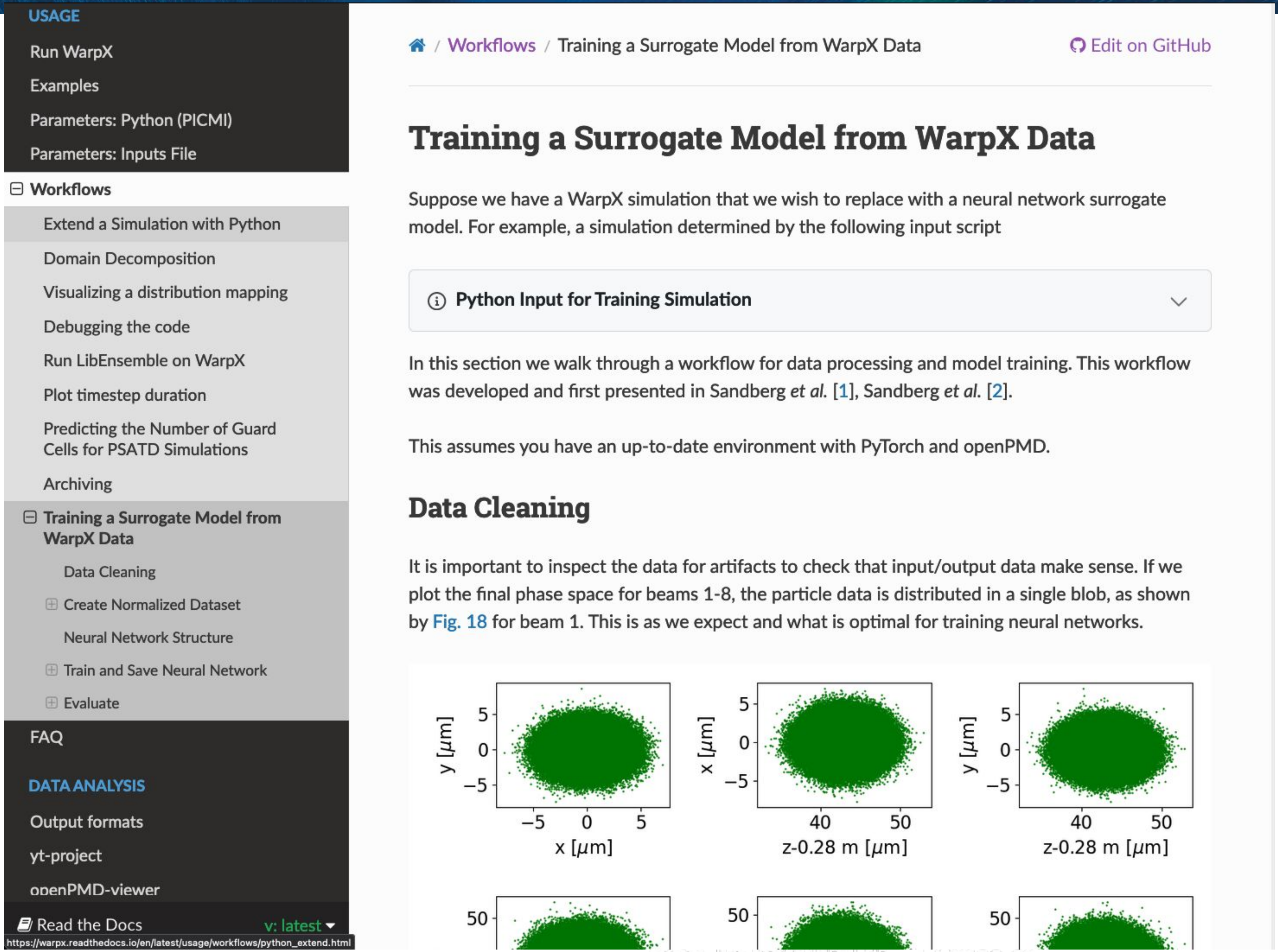


Wake-T, libEnsemble

J.-L. Vay et al., ECP WarpX MS FY23.1; A. Ferran Pousa et al., IPAC23, DOI:10.18429/JACoW-IPAC2023-TUPA093 (2023)

Functional examples of cleaning and training can be found on-line

https://warpx.readthedocs.io/en/latest/usage/workflows/ml_dataset_training.html



USAGE
Run WarpX
Examples
Parameters: Python (PICMI)
Parameters: Inputs File
Workflows
Extend a Simulation with Python
Domain Decomposition
Visualizing a distribution mapping
Debugging the code
Run LibEnsemble on WarpX
Plot timestep duration
Predicting the Number of Guard Cells for PSATD Simulations
Archiving
Training a Surrogate Model from WarpX Data
Data Cleaning
Create Normalized Dataset
Neural Network Structure
Train and Save Neural Network
Evaluate
FAQ
DATA ANALYSIS
Output formats
yt-project
openPMD-viewer
Read the Docs v: latest
https://warpx.readthedocs.io/en/latest/usage/workflows/python_extend.html

/ Workflows / Training a Surrogate Model from WarpX Data — Edit on GitHub

Training a Surrogate Model from WarpX Data

Suppose we have a WarpX simulation that we wish to replace with a neural network surrogate model. For example, a simulation determined by the following input script

Python Input for Training Simulation

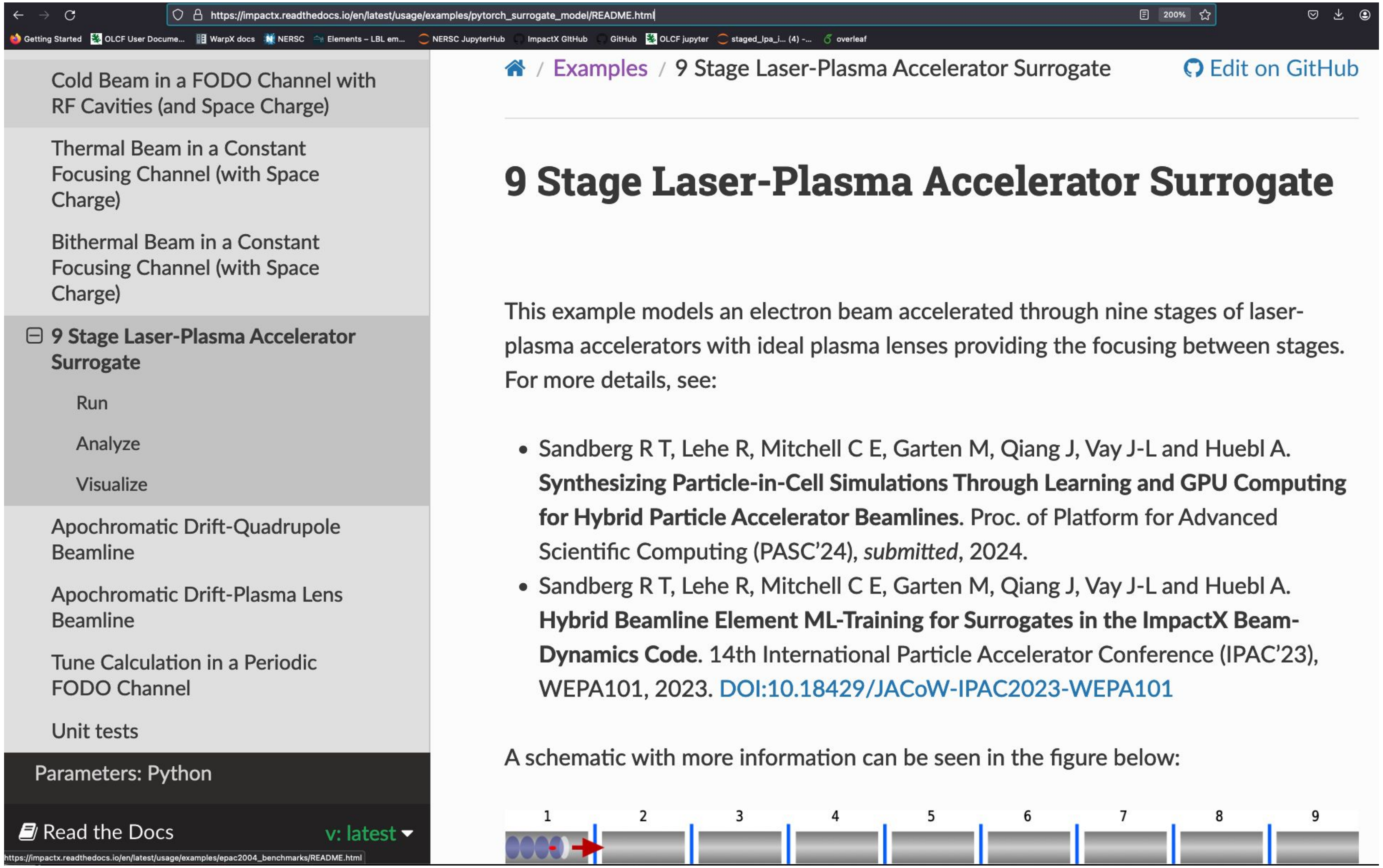
In this section we walk through a workflow for data processing and model training. This workflow was developed and first presented in Sandberg *et al.* [1], Sandberg *et al.* [2].

This assumes you have an up-to-date environment with PyTorch and openPMD.

Data Cleaning

It is important to inspect the data for artifacts to check that input/output data make sense. If we plot the final phase space for beams 1-8, the particle data is distributed in a single blob, as shown by Fig. 18 for beam 1. This is as we expect and what is optimal for training neural networks.

Functional examples of surrogates in ImpactX can also be found on-line

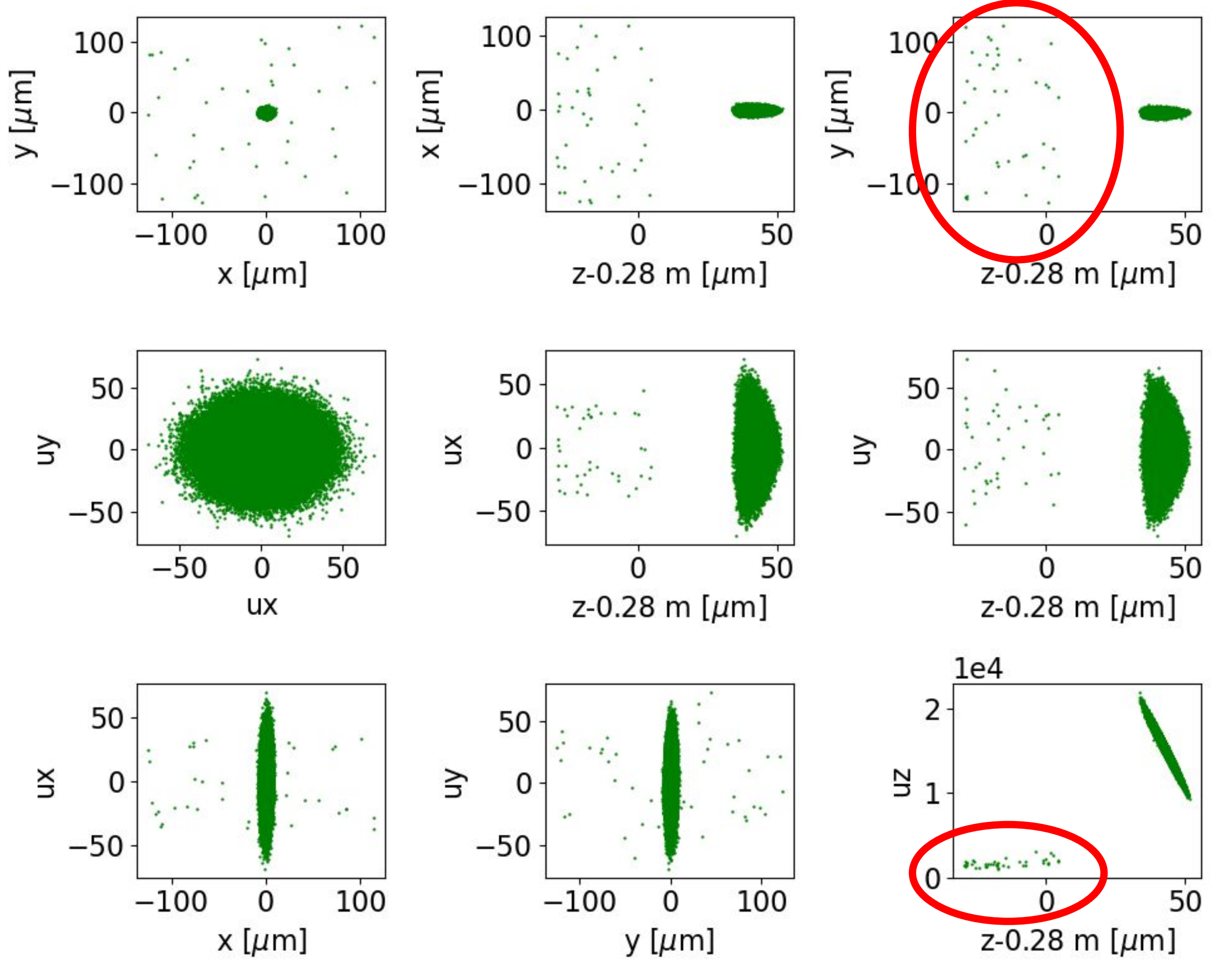


https://impactx.readthedocs.io/en/latest/usage/examples/pytorch_surrogate_model/README.html

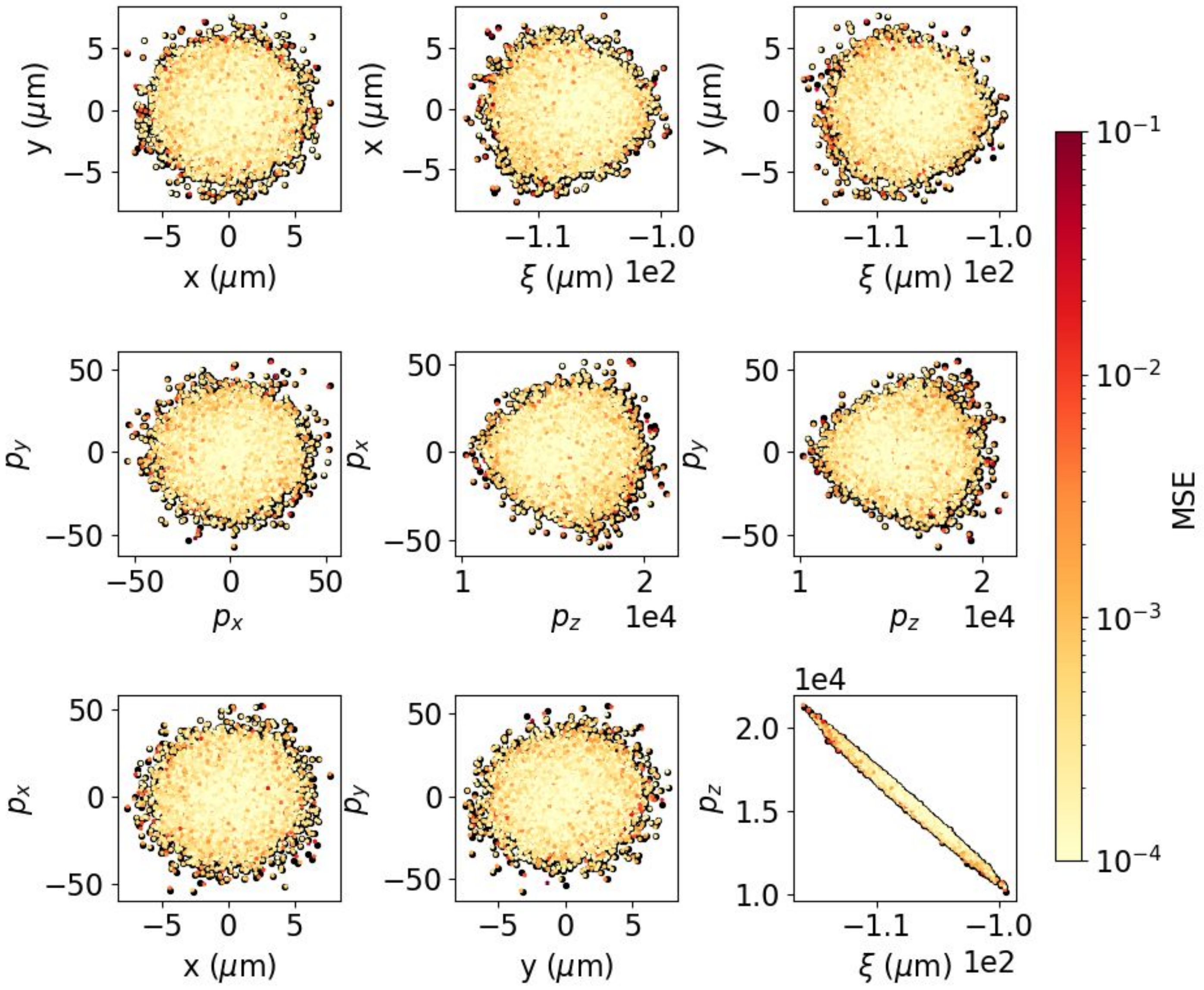
Data preparation and cleaning

- Remove clear outliers
- 70/30% train/test split
- Normalize by training bunch mea

Beam at Stage 1 end



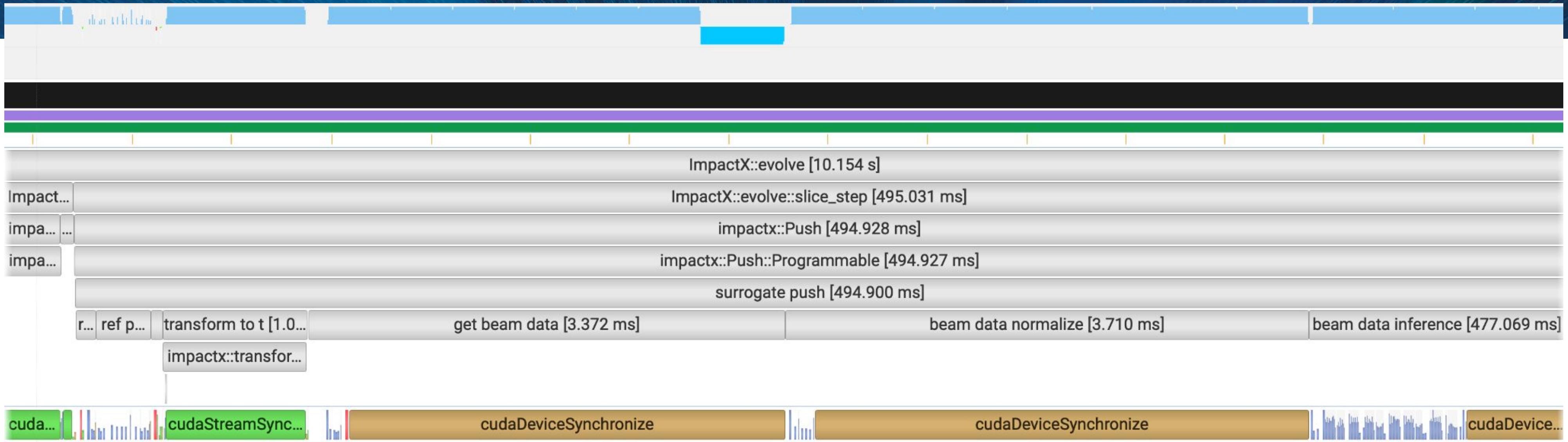
Model learns training data very well

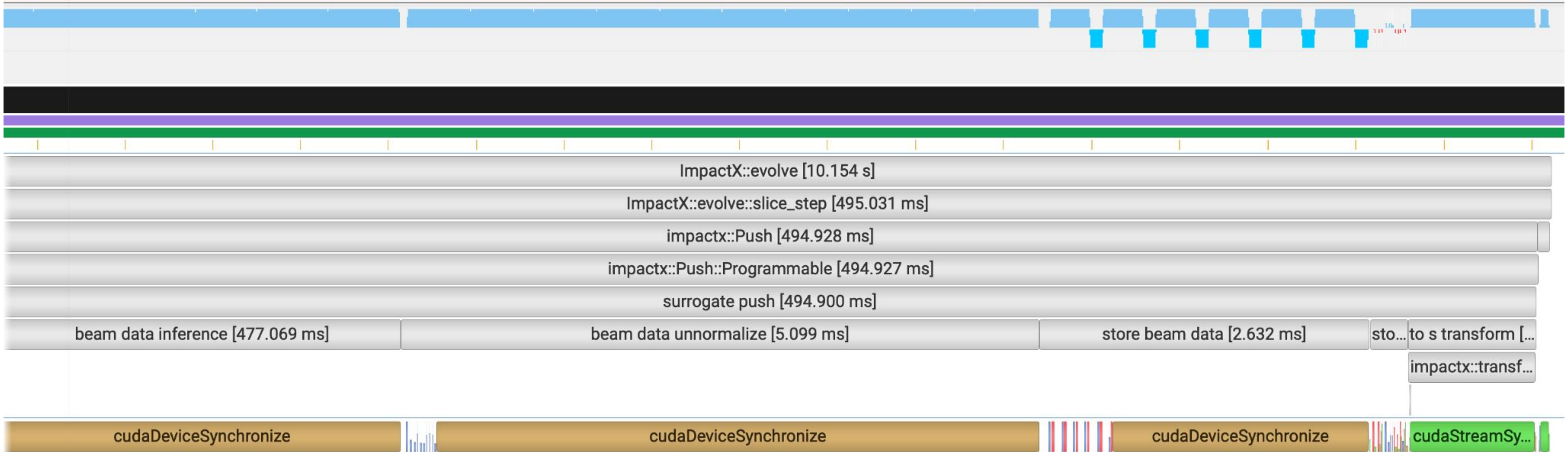


Stage 1
Black dots: training beam
Colored dots: predicted beam

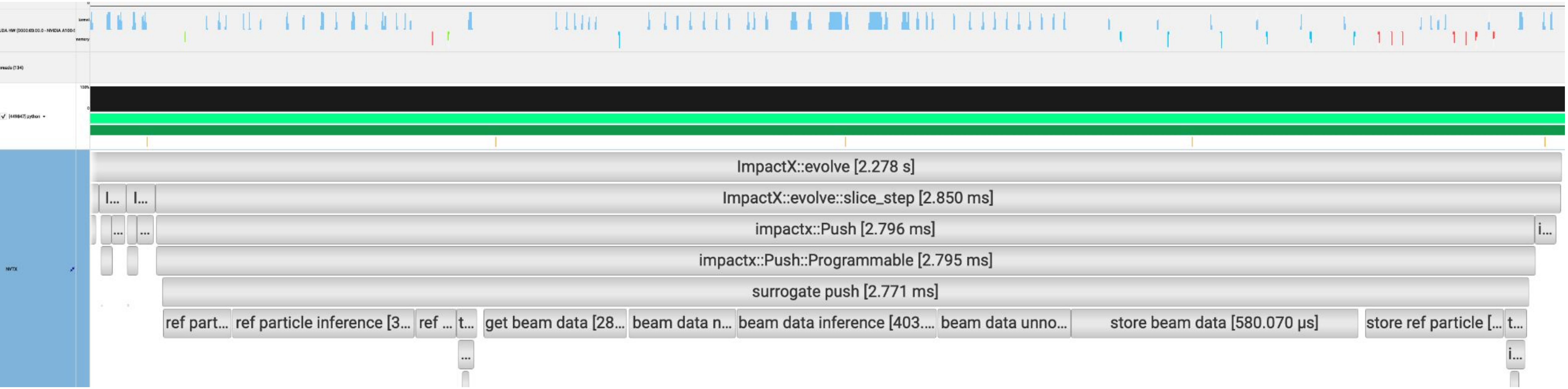
Training data: 1M particles / beam
Training time: 2-2.2 hrs on 1 GPU

10 million particles





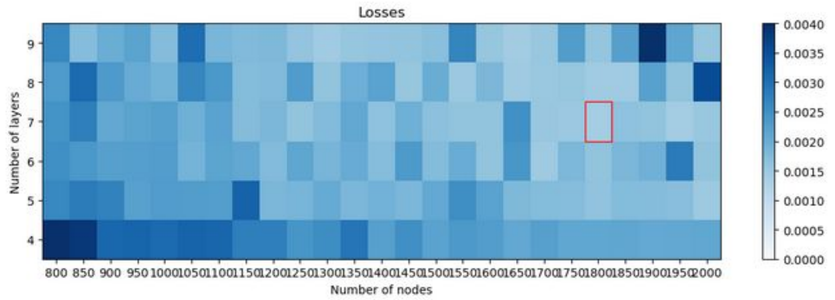
1000 particles

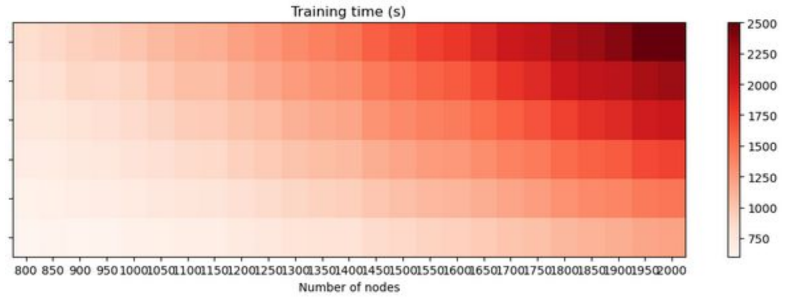


Workflows: Surrogates - NN Hyperparam Tuning [Ryan, Juliette]

What was accomplished

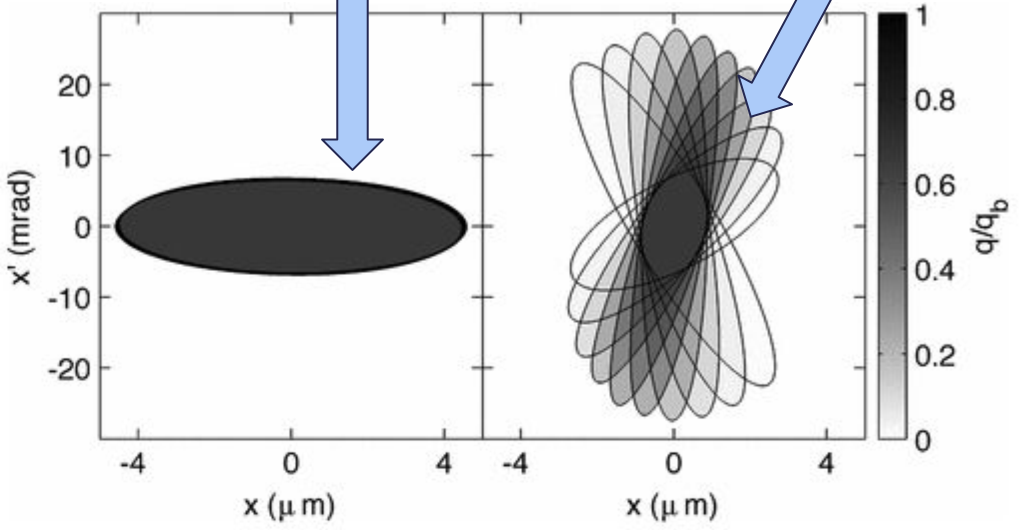
- Incorporated dropout layers
- Manage NN training with Ray Tune
- Used torch.compile (but it didn't speed things up)
- Continue the learning rate scan
- Learn that Ray Tune is the tool we want
- **Speedup > 2x** & smaller, less-noisy final loss-function with tf32 & PReLu

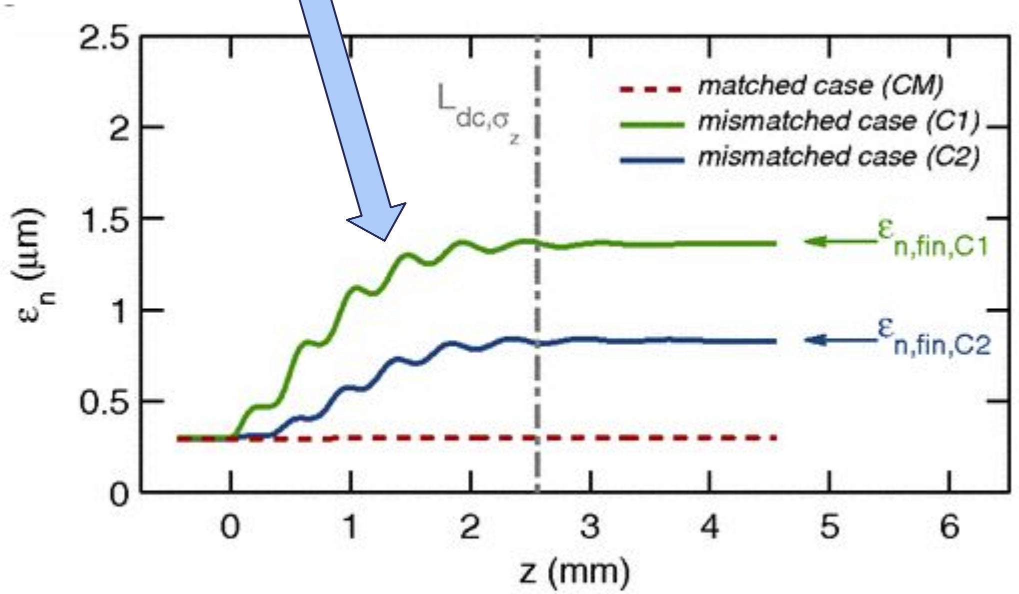




Emittance: area of phase space ellipse

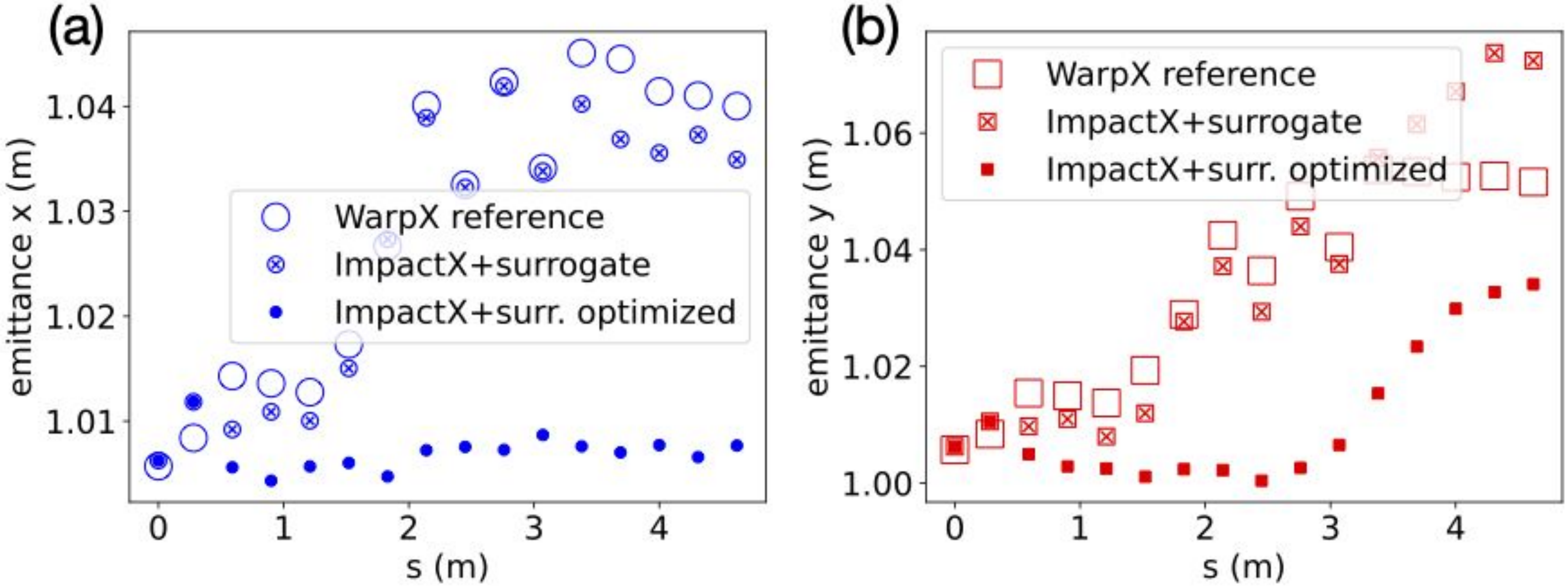
Emittance increases if beam is not matched and “smeared out”





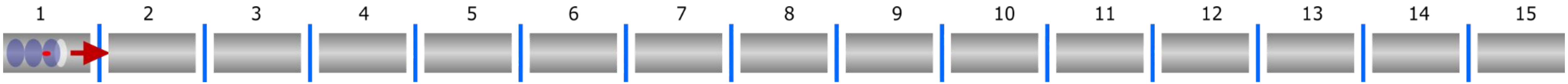
Example usage: Find lens strengths that minimize x emittance

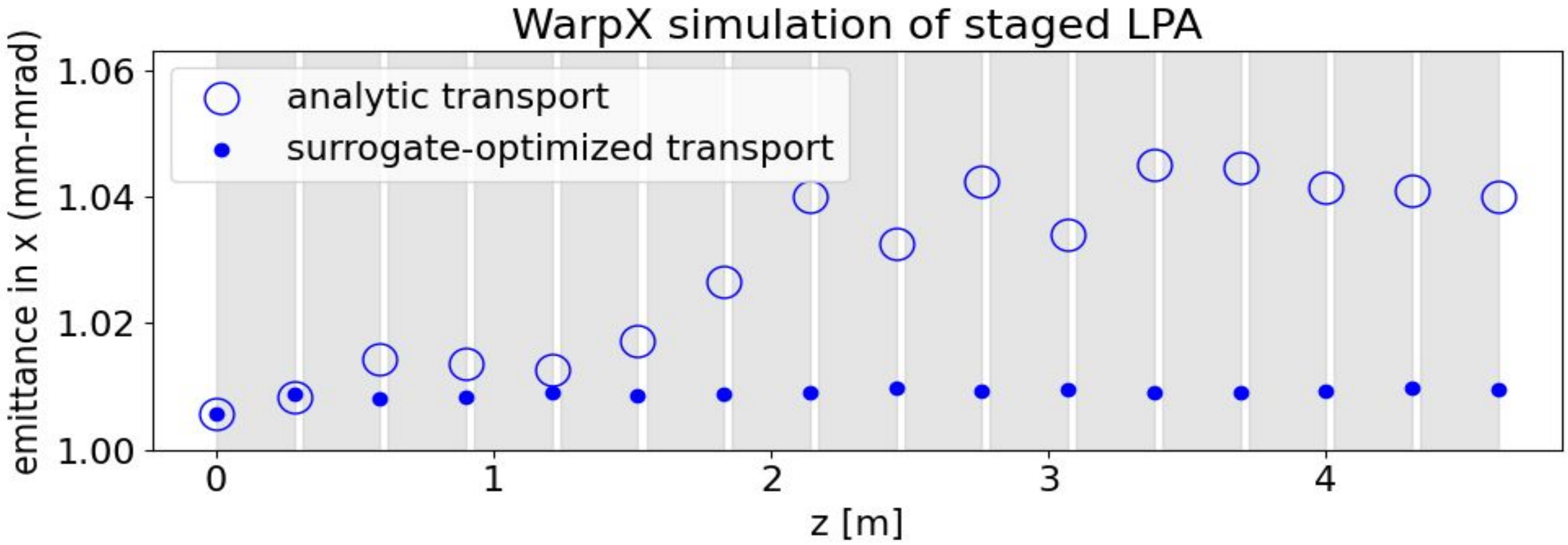
- Stage-by-stage optimization of transport parameters
- Emittance in x is kept constant, emittance growth in y reduced



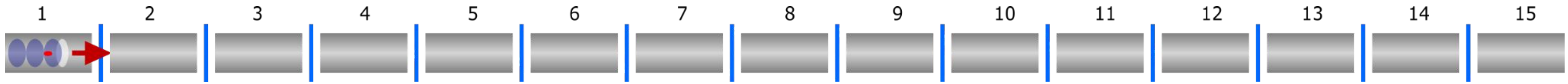
scipy.optimize.minimize with Nelder-Mead (simplex) optimization

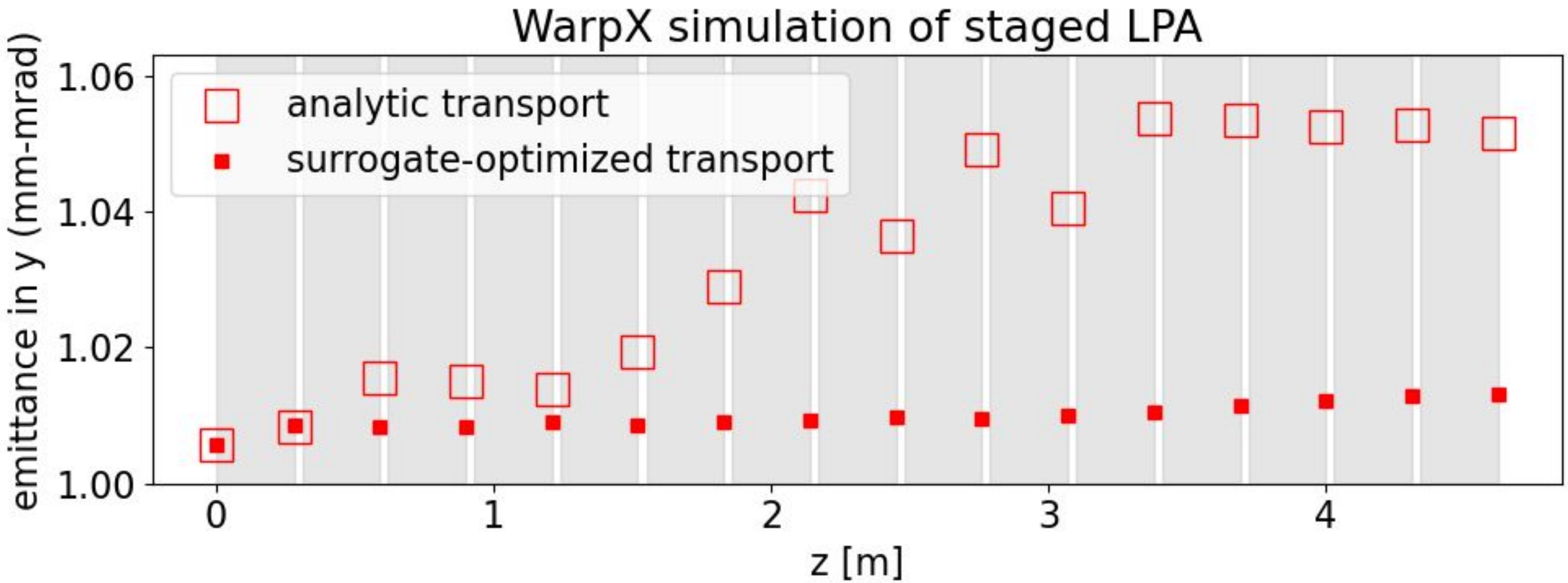
Close the loop: use ImpactX+WarpX-optimized transport to improve transport in WarpX





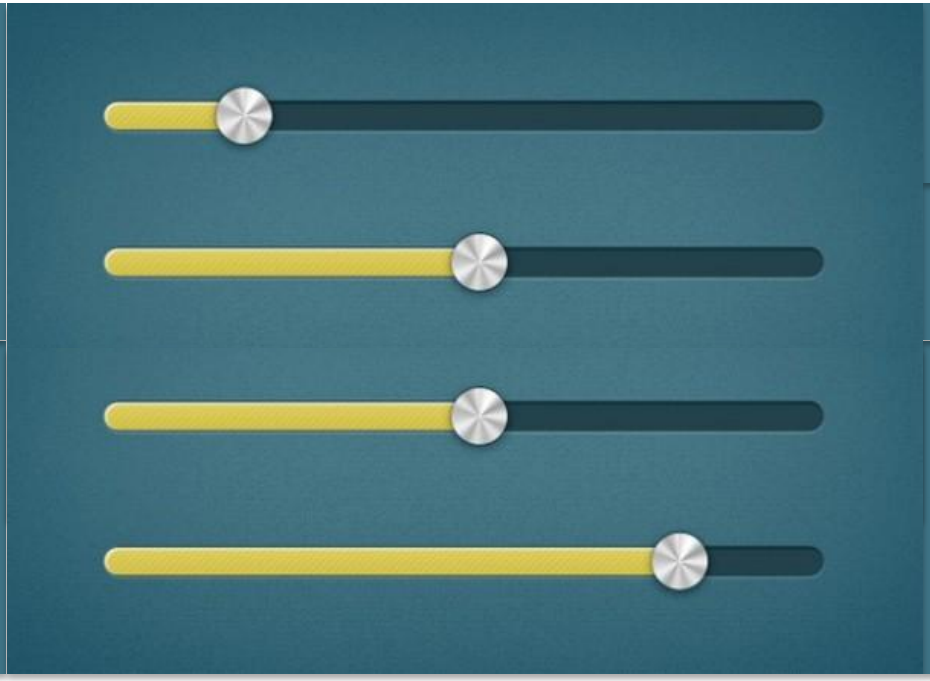
Close the loop: use ImpactX+WarpX-optimized transport to improve transport in WarpX





Toward an integrated ecosystem of codes with on-the-fly tunability

Speed				Fidelity
Fast & as accurate as possible	Reduced physics		Full physics	**Accurate** & as fast as possible
	Reduced models		First principles	
	1D-1V		3D-3V	
	Low resolution		High resolution	

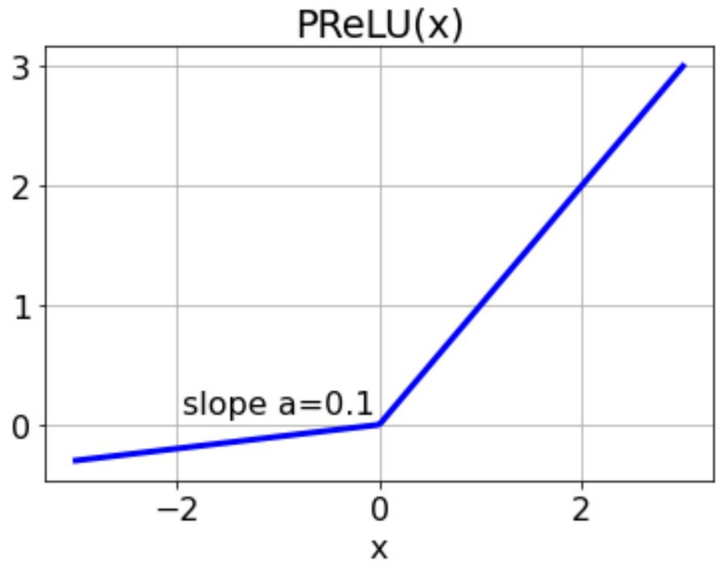


e.g., optimization & operations

e.g., exploration, training data

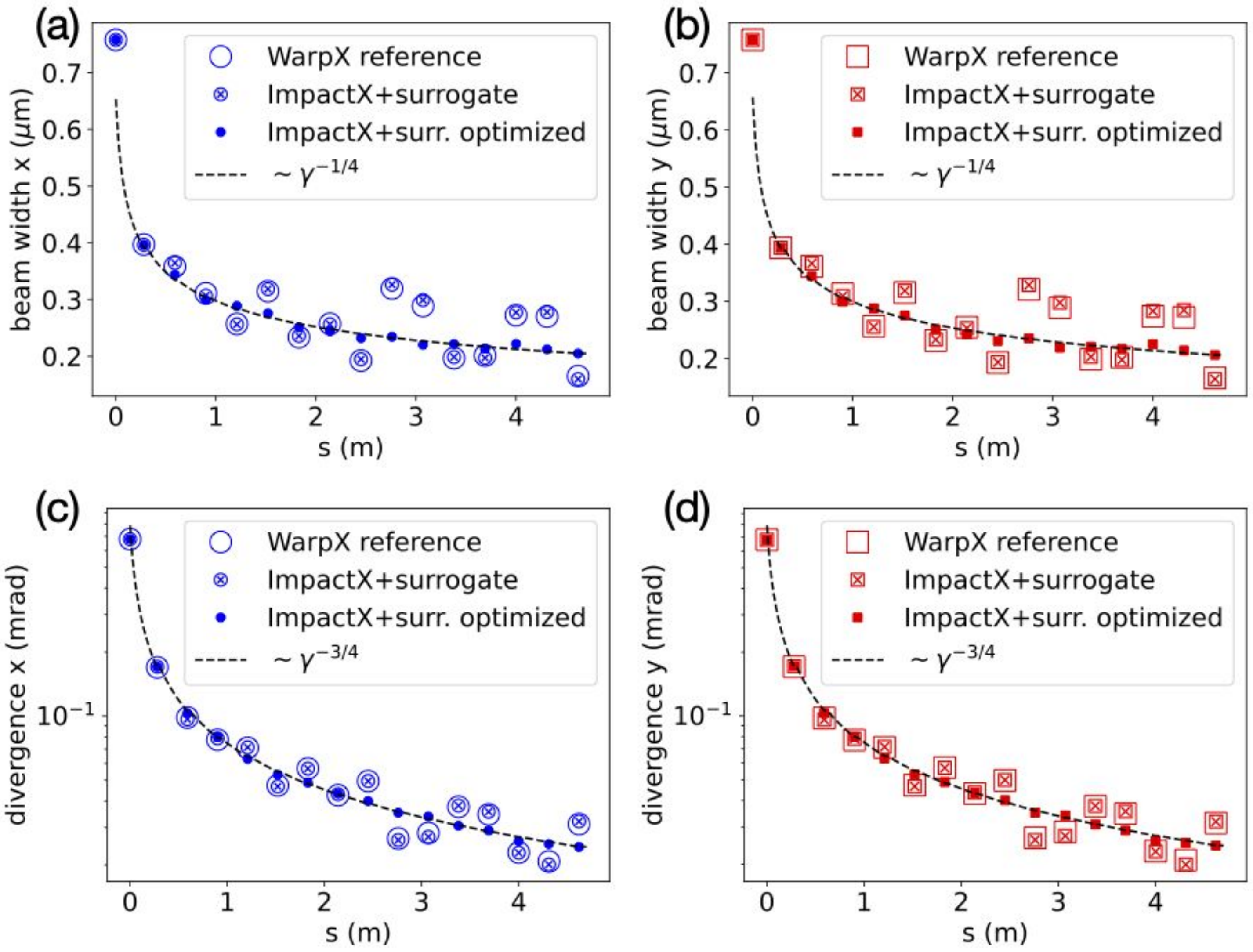
Ecosystem of codes
- share models & data between codes
- works best when standardized

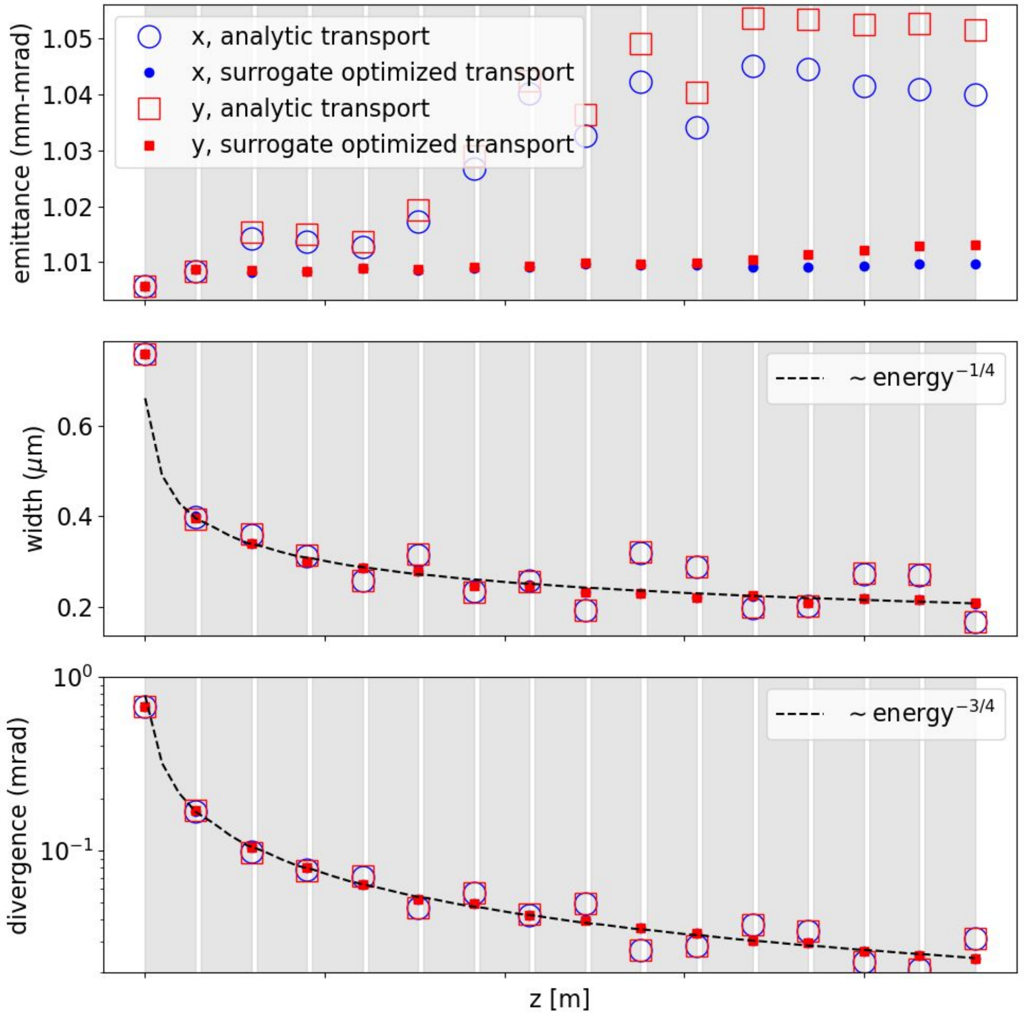
PReLU activation function



Optimal lens strengths improve beam match

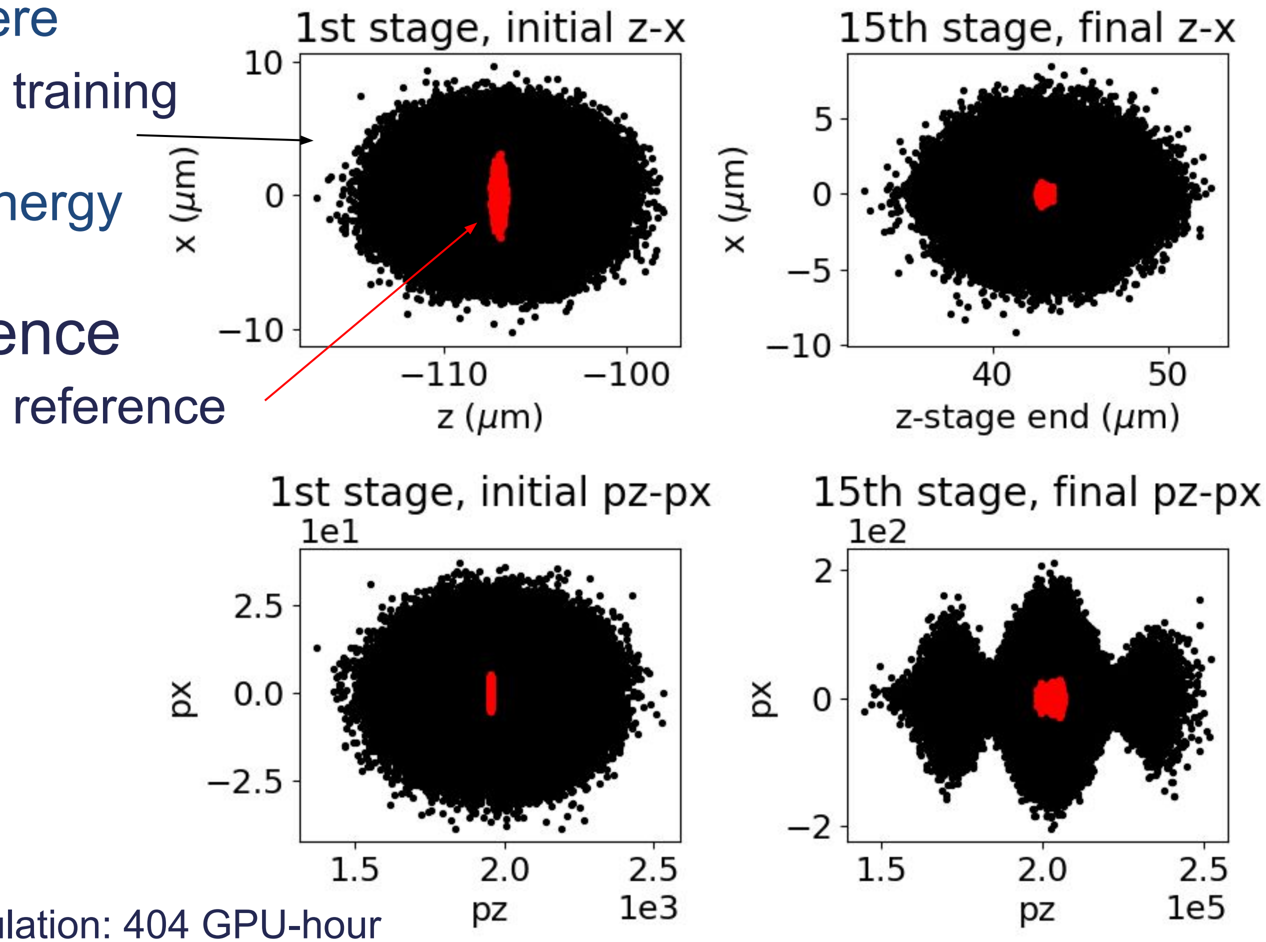
- Beam is better matched
  - Optimized beam width, divergence fit theory
- Recall: objective is emittance i
  - Optimizer “learns” to find better m
    In order to improve emittance




x, analytic transport
x, surrogate optimized transport
y, analytic transport
y, surrogate optimized transport
emittance (mm-mrad)
1.05
1.04
1.03
1.02
1.01
~ energy$^{-1/4}$
width (μm)
0.6
0.4
0.2
~ energy$^{-3/4}$
divergence (mrad)
10^0
10^{-1}
z [m]

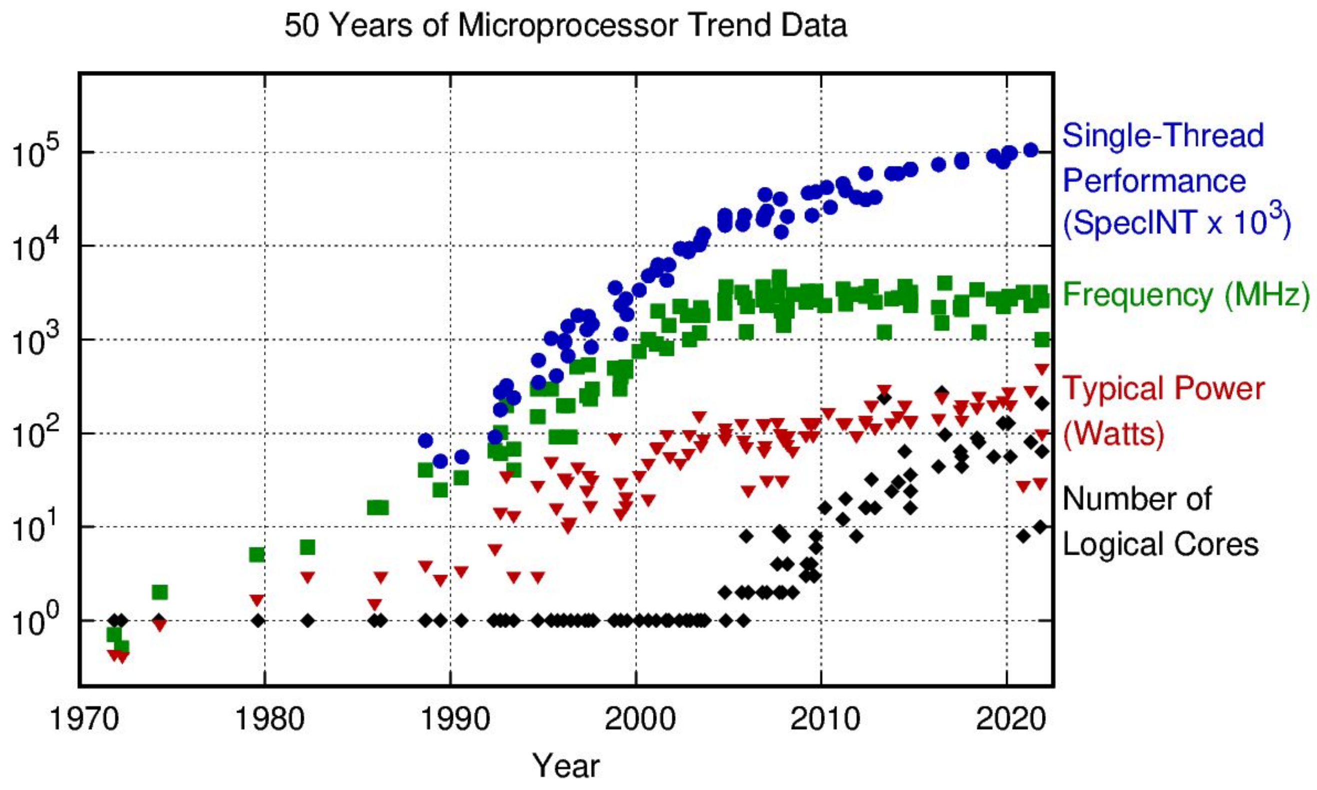
A high-fidelity WarpX simulation provides training data

- Single simulation / single stage
  - Low space charge – beams do not interfere
- 1 electron beam / stage
  - identical except in energy
  - beam *i* mean energy = expected mean energy reference beam at stage *i*
- Training beam ~ 3-5x larger than reference
  - Larger region of phase space
    - More general
    - Harder to learn
  - Smaller region of phase space
    - Could miss region of interest
    - Less general
    - More efficient training

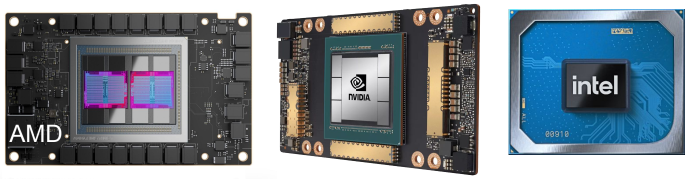


Training simulation: 404 GPU-hour WarpX simulation on Perlmutter

Power-Limits Seeded a Cambrian Explosion of Compute Architectures



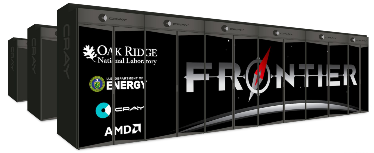
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp



CPUs

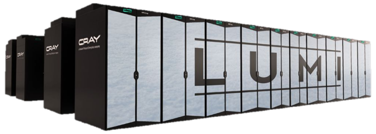
GPUs

Top 500



Frontier (USA): 1.2 EFlops
- AMD GPUs

Fugaku (Japan): 0.44 EFlops
- Fujitsu ARM CPUs



Lumi (Finland): 0.3 EFlops
- AMD GPUs

Leonardo (Italy): 0.24 EFlops
- Nvidia GPUs

Summit (USA): 0.15 EFlops
- Nvidia GPUs

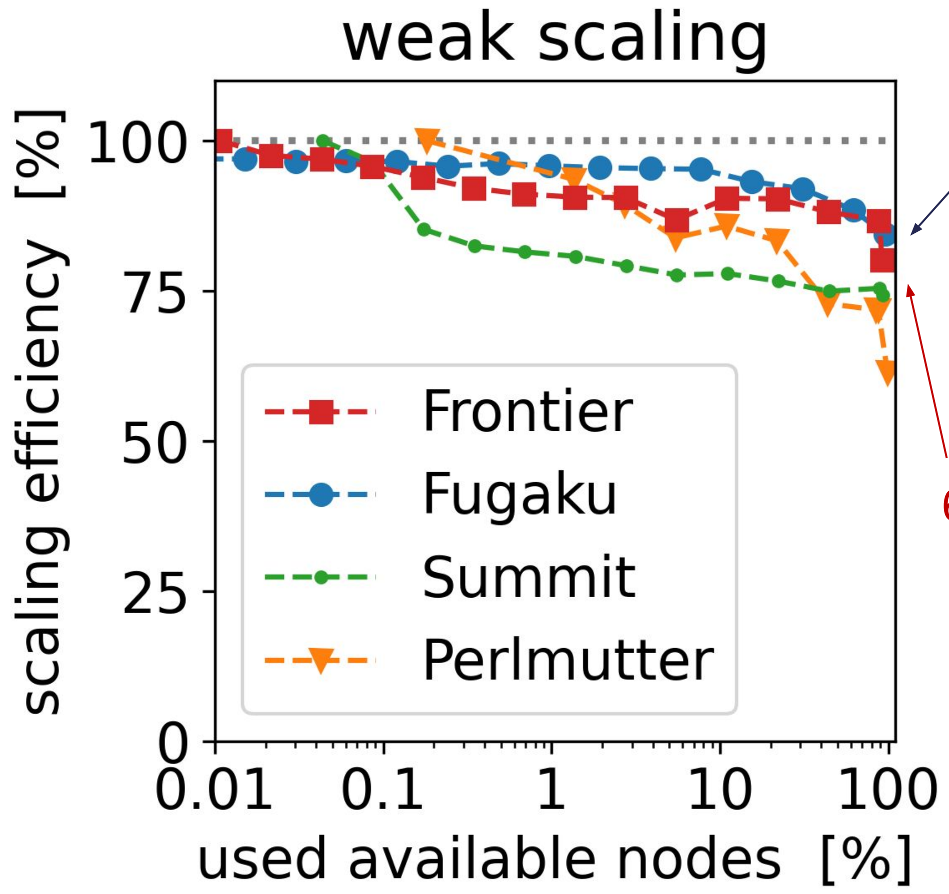
Upcoming (under acceptance testing)

Aurora (USA): ~2 EFlops
- Intel GPUs

WarpX is now 500x More Performant than its Pre-Exascale Baseline

April-July 2022: WarpX on **world's largest HPCs**

L. Fedeli, A. Huebl et al., *Gordon Bell Prize Winner* in SC'22, 2022



7,299,072 CPU Cores

68,608 GPUs of *First Exascale* Machine

Note: Perlmutter & Frontier in pre-acceptance at the time!

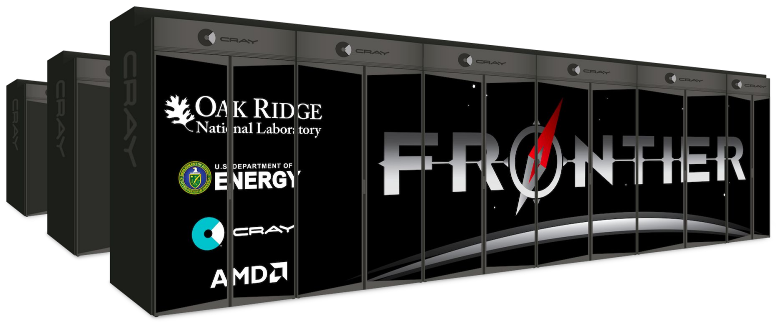


Figure-of-Merit: weighted updates / sec

Date	Code	Machine	N_c/Node	Nodes	FOM
3/19	Warp	Cori	0.4e7	6 625	2.2e10
3/19	WarpX	Cori	0.4e7	6 625	1.0e11
6/19	WarpX	Summit	2.8e7	1 000	7.8e11
9/19	WarpX	Summit	2.3e7	2 560	6.8e11
1/20	WarpX	Summit	2.3e7	2 560	1.0e12
2/20	WarpX	Summit	2.5e7	4 263	1.2e12
6/20	WarpX	Summit	2.0e7	4 263	1.4e12
7/20	WarpX	Summit	2.0e8	4 263	2.5e12
3/21	WarpX	Summit	2.0e8	4 263	2.9e12
6/21	WarpX	Summit	2.0e8	4 263	2.7e12
7/21	WarpX	Perlmutter	2.7e8	960	1.1e12
12/21	WarpX	Summit	2.0e8	4 263	3.3e12
4/22	WarpX	Perlmutter	4.0e8	928	1.0e12
4/22	WarpX	Perlmutter†	4.0e8	928	1.4e12
4/22	WarpX	Summit	2.0e8	4 263	3.4e12
4/22	WarpX	Fugaku†	3.1e6	98 304	8.1e12
6/22	WarpX	Perlmutter	4.4e8	1 088	1.0e12
7/22	WarpX	Fugaku	3.1e6	98 304	2.2e12
7/22	WarpX	Fugaku†	3.1e6	152 064	9.3e12
7/22	WarpX	Frontier	8.1e8	8 576	1.1e13

110x

500x

EXASCALE COMPUTING PROJECT