The 34th JKPS:

Phase Space Reconstruction from Accelerator Beam Measurements Using Neural Networks and Differentiable Simulations

R. Roussel,[*] A. Edelen, C. Mayes, and D. Ratner
SLAC National Accelerator Laboratory, Menlo Park, California 94025, USA

J. P. Gonzalez-Aguilera
Department of Physics, University of Chicago, Chicago, Illinois 60637, USA

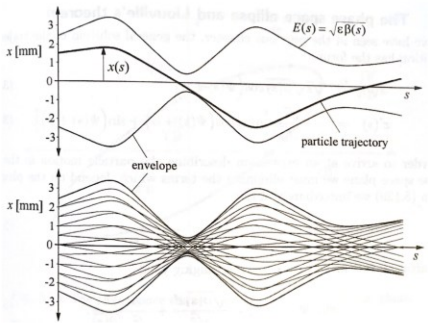
S. Kim, E. Wisniewski, and J. Power
Argonne National Laboratory, Argonne, Illinois 60439, USA

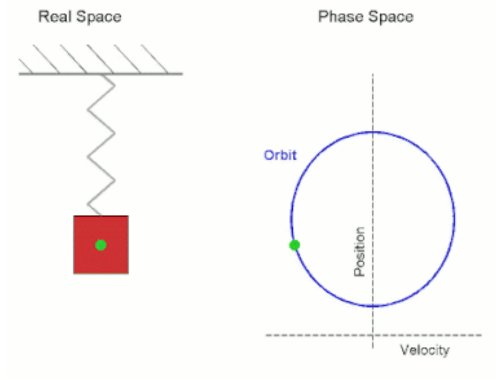
25. 10. 31

Geunwoo Kim
DANE, POSTECH

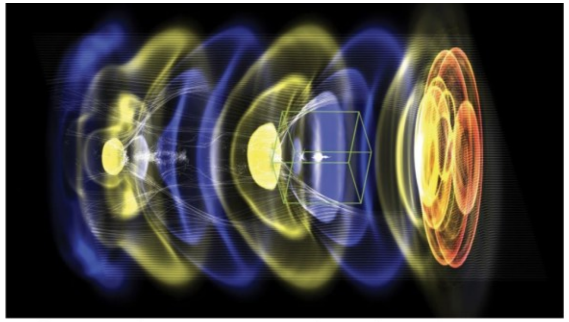
Introduction: Motivation & Goal

- **Needs:**
  - Increasingly precise control over the 6D phase space distribution is essential for the success of next-generation accelerator applications.





- **Key Applications:**
  - Examples include new operating modes at Free Electron Lasers (FELs) and novel, compact (laser-driven plasma) acceleration schemes.

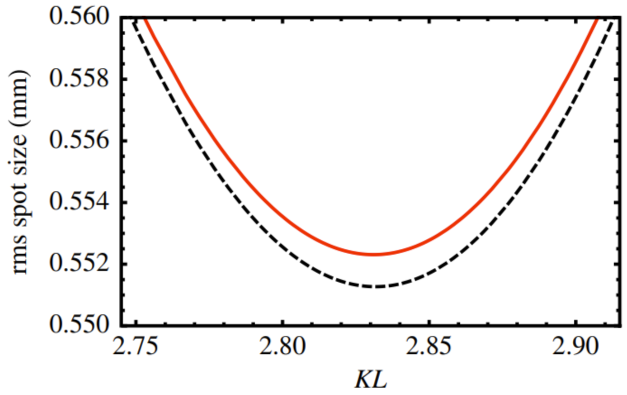


A laser-driven plasma accelerator stage: a rendering from a 3D WarpX simulation with mesh refinement (image courtesy of LBNL).

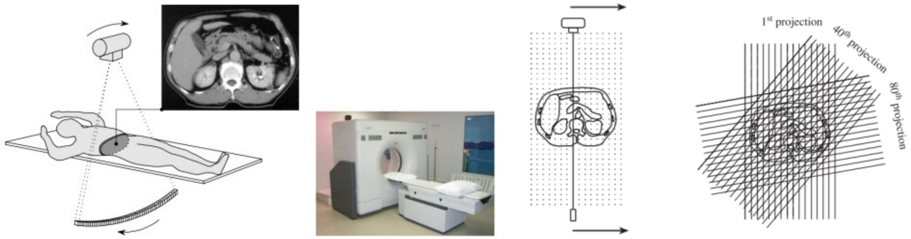
- **Core Challenge:**
  - The effectiveness of advanced beam shaping techniques relies on having accurate and detailed measurements of the full 6D phase space, which is a difficult task.

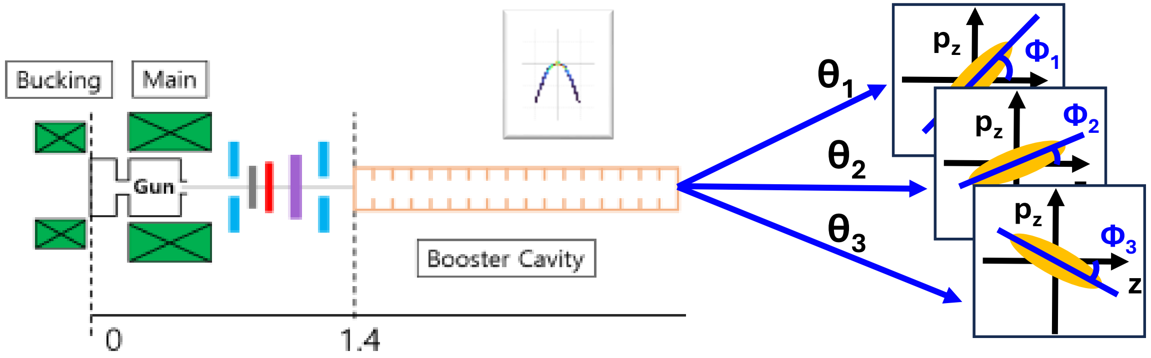
Introduction: Literature review

- **Conventional moment-based methods: quadrupole scan**
  - → discard higher order terms, can't measure the detailed beam shape.



- **Tomography (FBP, SART, MENT):**
  - → Produce accurate reconstructions, however, higher computational costs, angle constraints, and discard coupling components.



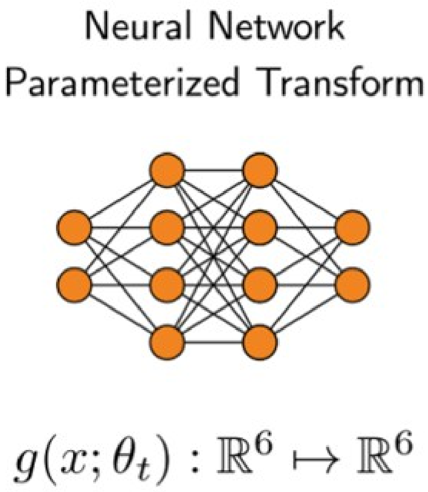


- **Machine Learning (ML) based methods:**
  - → Requires well-defined, large training datasets.
  - → A huge, time-consuming process is required (up to a few weeks)

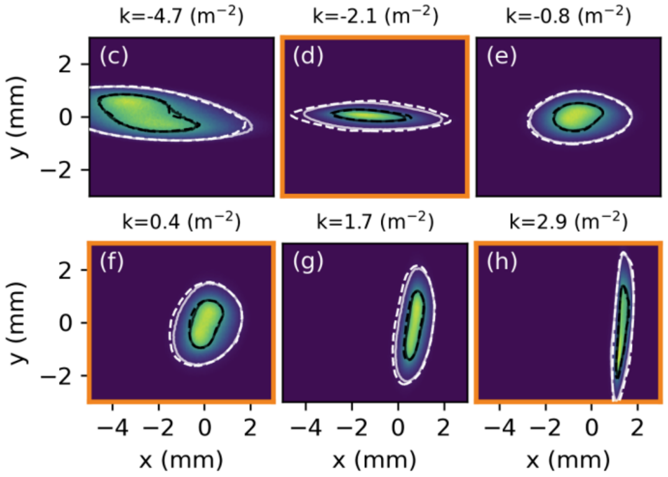
Introduction: Proposed solutions

- **New novel algorithm proposed**
  - Reconstruct phase space using differentiable simulation and neural network.
  - Utilize limited measurements from widely available accelerator components.

<Differentiable simulation>



<Neural Network>



<4D reconstructed phase space>

Background: concept of NN

Phase Space Reconstruction from Accelerator Beam Measurements Using Neural Networks and Differentiable Simulations

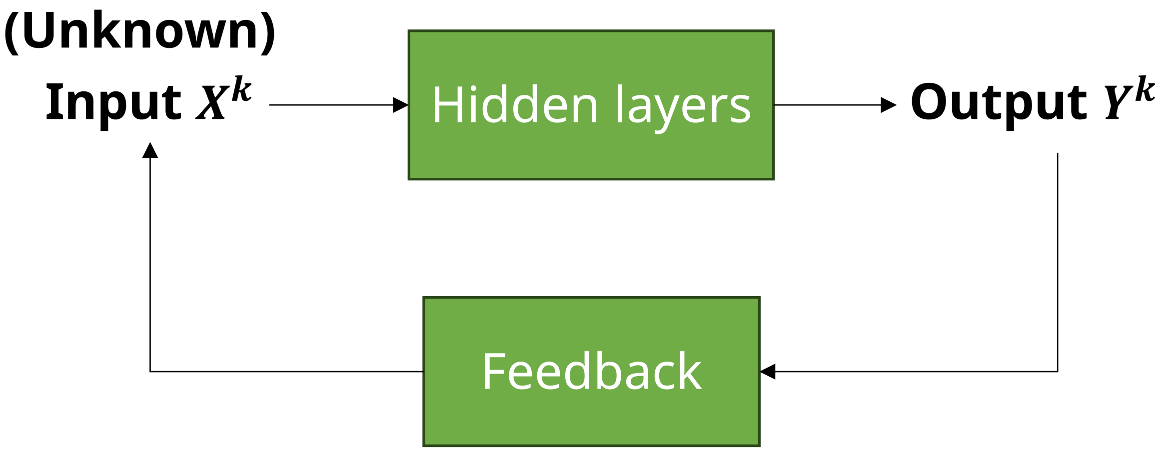
R. Roussel,* A. Edelen, C. Mayes, and D. Ratner
SLAC National Accelerator Laboratory, Menlo Park, California 94025, USA

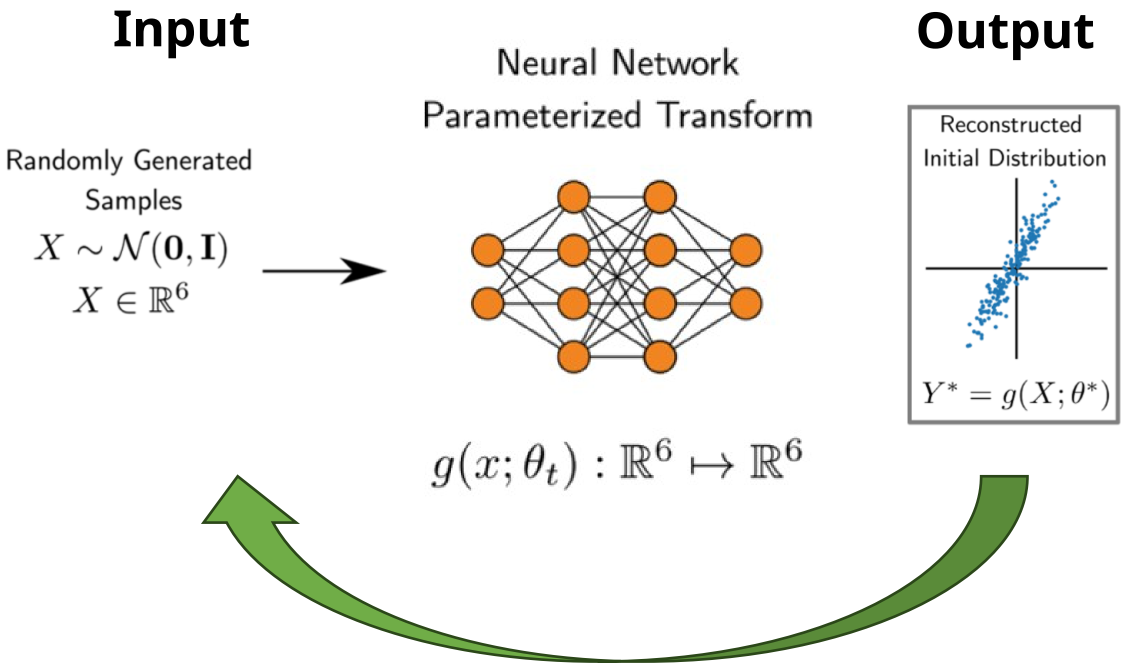
J. P. Gonzalez-Aguilera
Department of Physics, University of Chicago, Chicago, Illinois 60637, USA

S. Kim, E. Wisniewski, and J. Power
Argonne National Laboratory, Argonne, Illinois 60439, USA

Neural Network (NN) a.k.a Artificial NN (ANN)

- Well-developed open source are available: **PyTorch** and Tensorflow.

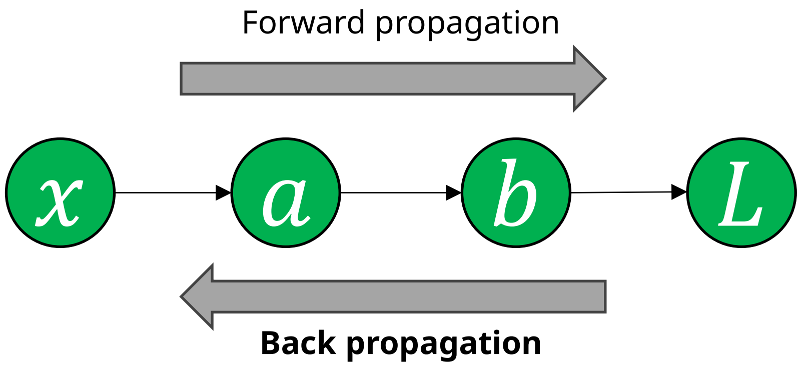




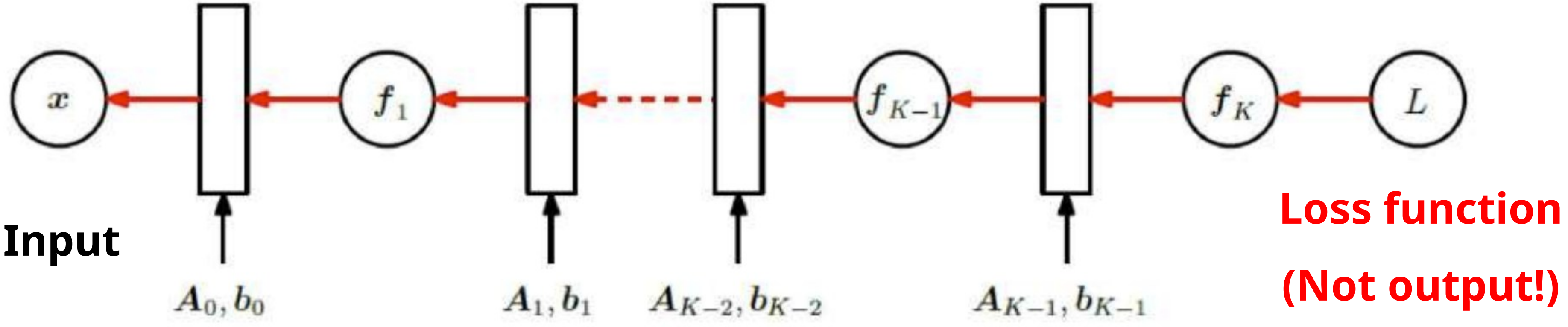
Background: Backpropagation

Basic principle of feedback in NN

- Backpropagation
- Calculate gradient by using chain rule.



$$\frac{dL}{dx} = \frac{dL}{db}\left(\frac{db}{da}\frac{da}{dx}\right)$$

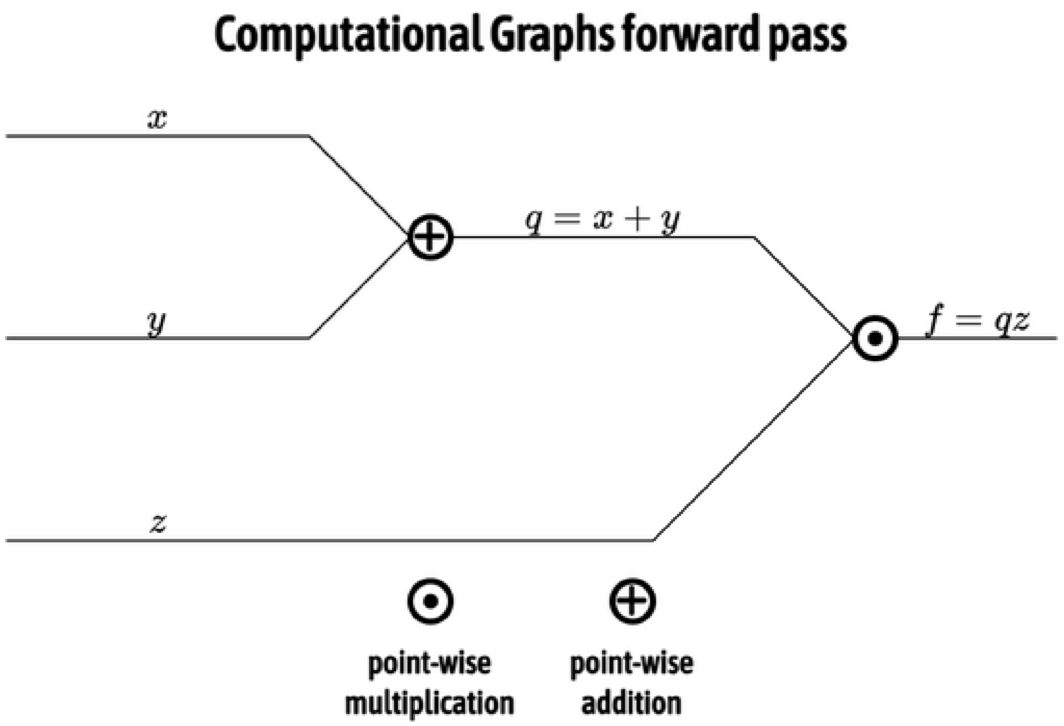


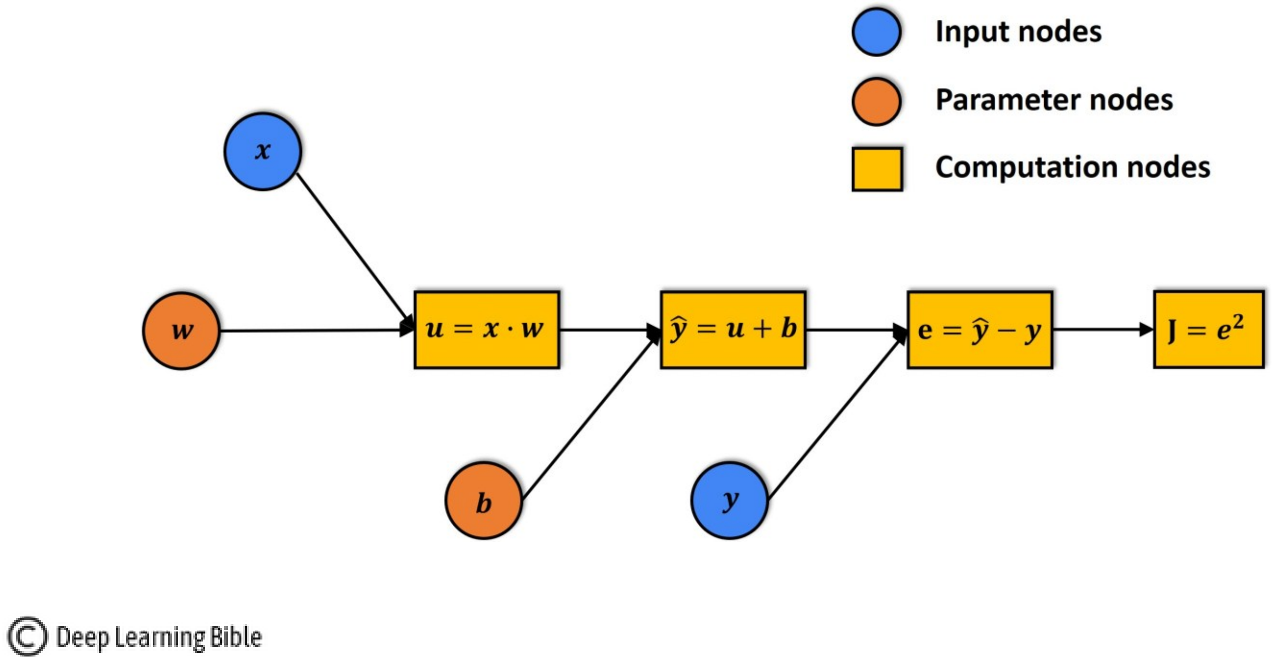
Background: Computational graphs

Basic principle of feedback in NN

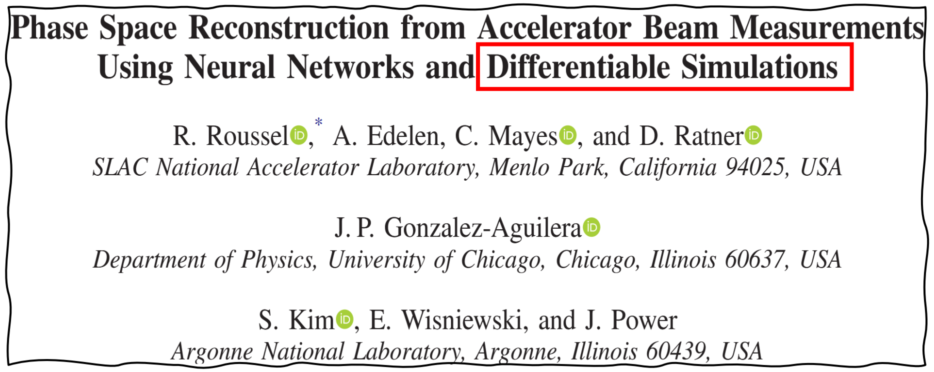
✓ <u>Computational graphs</u>

- Pytorch store the calculation form in their memory when run the forward propagation.





Background: Differentiable simulation



Phase Space Reconstruction from Accelerator Beam Measurements Using Neural Networks and Differentiable Simulations

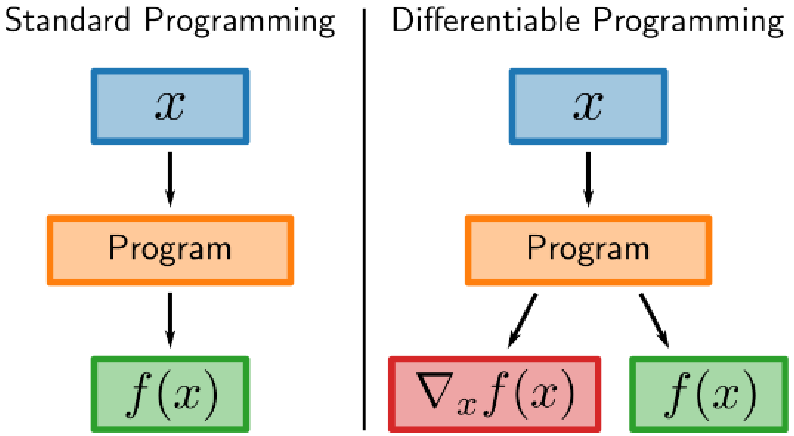
R. Roussel,* A. Edelen, C. Mayes, and D. Ratner
SLAC National Accelerator Laboratory, Menlo Park, California 94025, USA

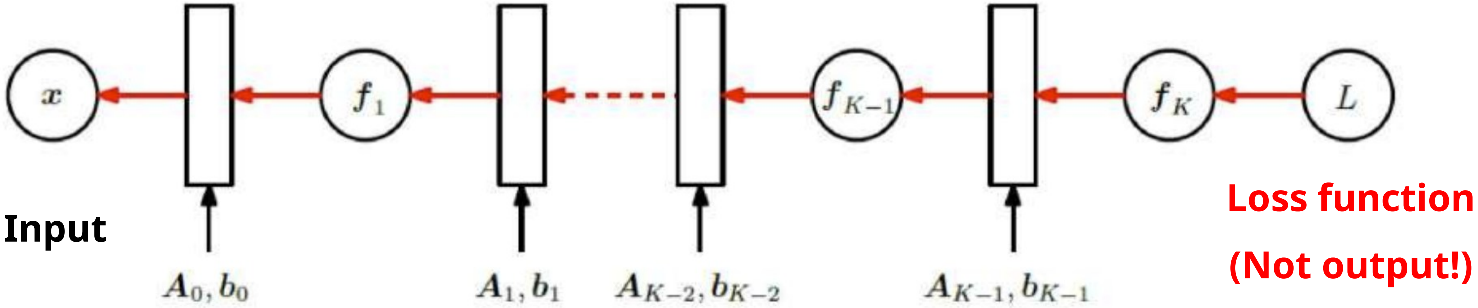
J. P. Gonzalez-Aguilera
Department of Physics, University of Chicago, Chicago, Illinois 60637, USA

S. Kim, E. Wisniewski, and J. Power
Argonne National Laboratory, Argonne, Illinois 60439, USA

What is differentiable simulation?

1. To update input $\vec{x}$, we need to know gradient $\frac{\partial L}{\partial x}$.
2. Whole process should be explained via gradient.
3. If using PyTorch, the simulation can give the gradient information to PyTorch. → ***Differentiable***





➢ **But, how to apply this concept to phase space reconstruction?**

Background: Differentiable simulation

Differentiable simulation

- Bmad-X (Library independent), Cheetah (PyTorch-based)
  - They can send gradient information to PyTorch NN model.

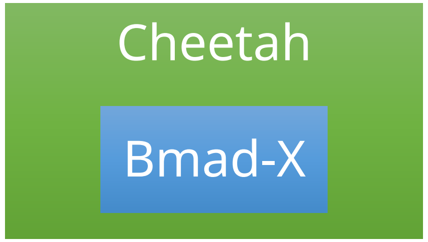
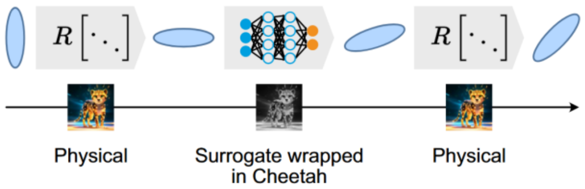


TABLE I. Step computation times of simulation codes in milliseconds.

Code	Comment	Laptop	HPC node
ASTRA	Space charge	264000.00	3605000.00
	No space charge	109000.00	183000.00
Parallel ASTRA	Space charge	39000.00	17300.00
	No space charge	16900.00	12600.00
Ocelot	Space charge	22100.00	21700.00
	No space charge	182.00	119.00
Bmad-X		40.50	74.30
Xsuite	CPU, no space charge	0.81	2.82
	GPU, no space charge	...	0.57
Cheetah	ParticleBeam	1.60	2.95
	ParticleBeam + optimization	0.79	0.72
	ParticleBeam + GPU	...	4.63
	ParticleBeam + optimization + GPU	...	0.09
	ParameterBeam	0.76	1.29
	ParameterBeam + optimization	0.02	0.04

Cheetah: Fast simulation for linear beam dynamics

(d) Integrate modular neural network surrogates



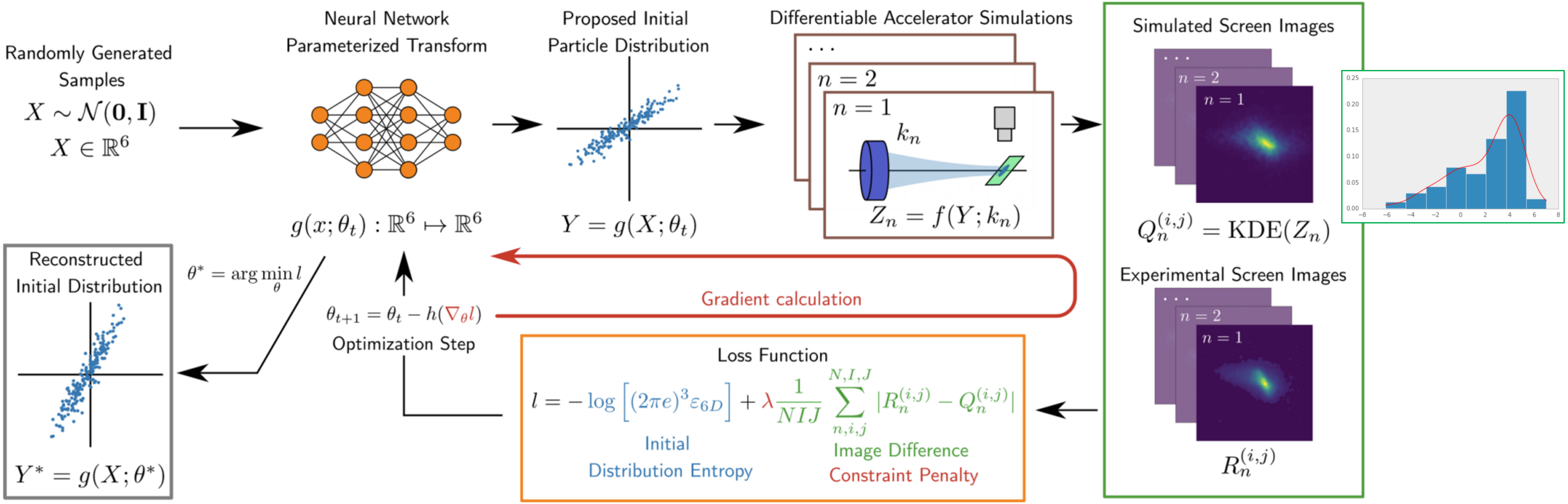
Standard simulation

- In contrast, standard simulation (ASTRA, ELEGANT, MAD-X, Xsuite, *etc.)* can't do that.
- *They use the external library* which is considered blackbox model because PyTorch can't interpret its code.

Method: Overview

Determine difference	Estimated Image	Particle Tracking Simulation	Proposed k^{th} beam distribution

$$\frac{d\boldsymbol{L}}{d\boldsymbol{\theta}_k} = \boxed{\frac{\partial L}{\partial \mathrm{Image}_{\mathrm{sim}}}} \times \boxed{\frac{\partial \mathrm{Image}_{\mathrm{sim}}}{\partial \mathrm{Particle}_{\mathrm{out}}}} \times \boxed{\frac{\partial \mathrm{Particle}_{\mathrm{out}}}{\partial \mathrm{Optics}} \times \frac{\partial \mathrm{Optics}}{\partial \mathrm{Particle}_{\mathrm{in}}}} \times \frac{\partial \mathrm{Particle}_{\mathrm{in}}}{\partial \theta_k}$$



Method: KDF

Determine Loss gradient | Estimated Image | Particle Tracking Simulation | Proposed k^th beam distribution

$$\frac{dL}{d\theta_k} = \frac{\partial L}{\partial \mathrm{Image_{diff}}} \times \frac{\partial \mathrm{Image_{diff}}}{\partial \mathrm{Particle_{out}}} \times \frac{\partial \mathrm{Particle_{out}}}{\partial \mathrm{Optics}} \times \frac{\partial \mathrm{Optics}}{\partial \mathrm{Particle_{in}}} \times \frac{\partial \mathrm{Particle_{in}}}{\partial \theta_k}$$

Kernel Density Function (KDF)

- Make the histogram to smooth(=differentiable) function

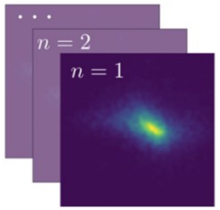
$$l = -\log[(2\pi e)^3 \varepsilon_{6D}] + \lambda \frac{1}{NIJ} \Sigma_{n,i,j}^{N,I,J} \left| R_n^{(i,j)} - Q_n^{(i,j)} \right|$$

Distributed Entropy   Image Difference

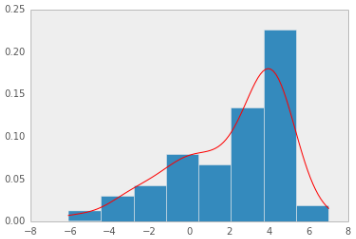
Take 1 image at a certain k

20 in total

Simulated Screen Images



$Q_n^{(i,j)} = \mathrm{KDE}(Z_n)$

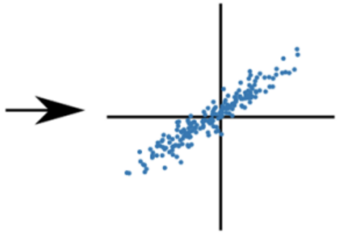


$$\frac{dL}{d\theta_k}$$

Neural Network Parameterized Transform

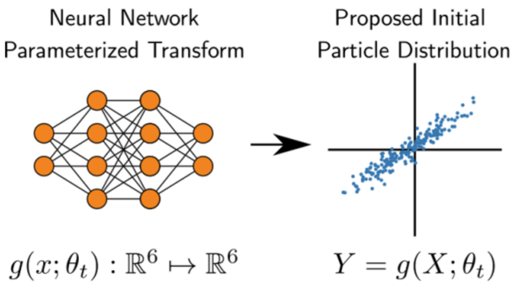
$g(x; \theta_t) : \mathbb{R}^6 \mapsto \mathbb{R}^6$

Proposed Initial Particle Distribution



$Y = g(X; \theta_t)$

Method: Snapshot ensemble data



Ensemble data taking method

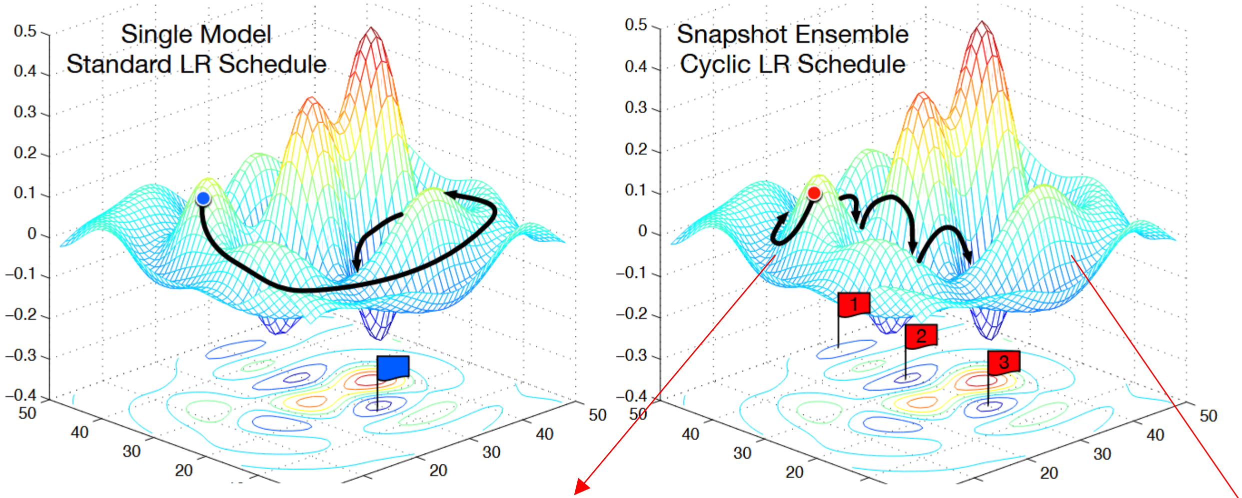
- Get multiple data from single epoch (iteration set).

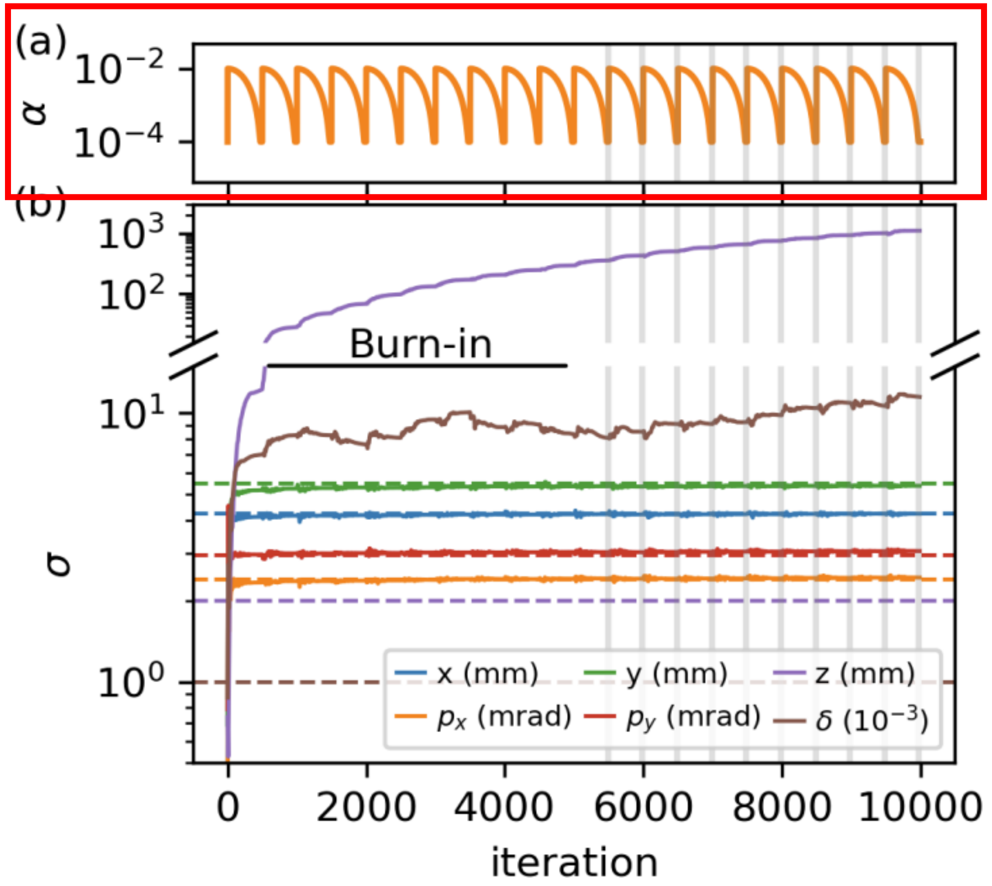
SNAPSHOT ENSEMBLES: TRAIN 1, GET M FOR FREE

Gao Huang*, Yixuan Li*, Geoff Pleiss
Cornell University
{gh349, yl2363}@cornell.edu, geoff@cs.cornell.edu

Zhuang Liu
Tsinghua University
liuzhuangthu@gmail.com

John E. Hopcroft, Kilian Q. Weinberger
Cornell University
jeh@cs.cornell.edu, kqw4@cornell.edu





Moving step is determined by learning rate, α.

Contour is drawn by loss function $\partial L/\partial\theta$.

Method: burn-in and minor components

Evolution of the proposal distribution

- **Burn-in:** Neglect early estimated results
- $l = -\log[(2\pi e)^3 \varepsilon_{6D}] + \lambda \frac{1}{NIJ} \Sigma_{n,i,j}^{N,I,J} \left| R_n^{(i,j)} - Q_n^{(i,j)} \right|$

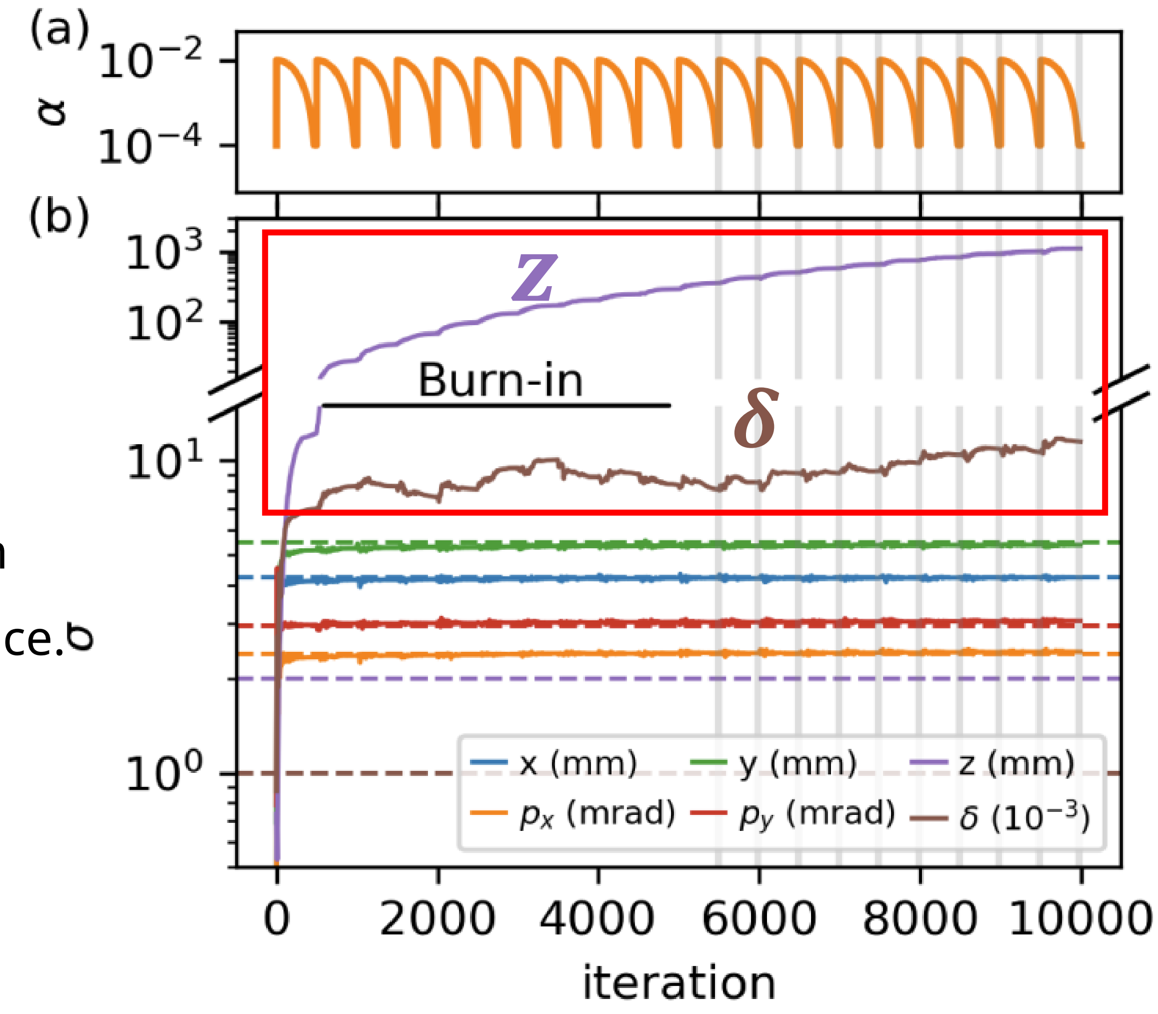
Why z and δ increase w.r.t iteration?

Variable: QM strength k

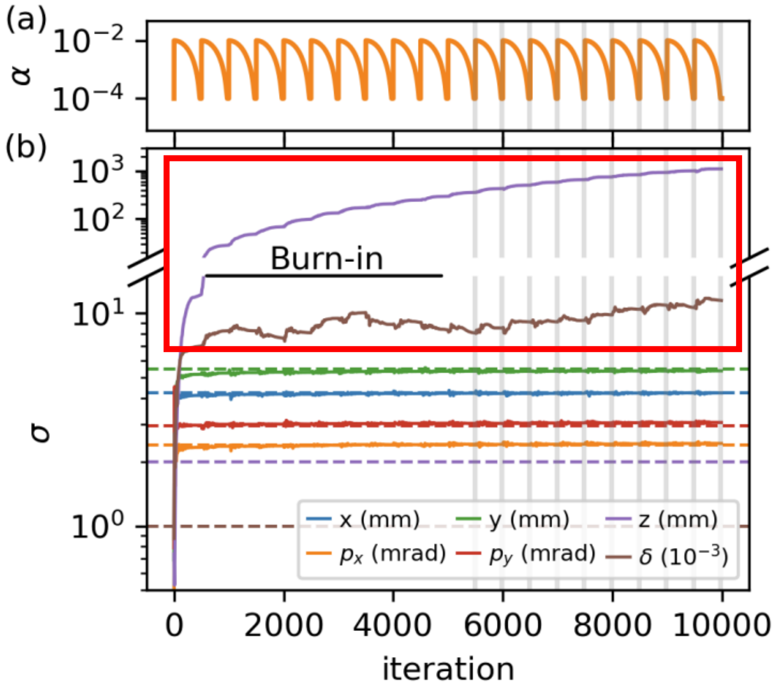
- To reduce l, algorithm tends to increase entropy term up to intersection point with the image difference.

- $\varepsilon_z = \sqrt{\langle z^2 \rangle \langle \delta^2 \rangle - \langle z\delta \rangle^2} \leftrightarrow z$ is independent of k.

  → **continuously increase (Entropy increase)**
- δ, energy spread, affects to focal point f.



Method: δ compensation

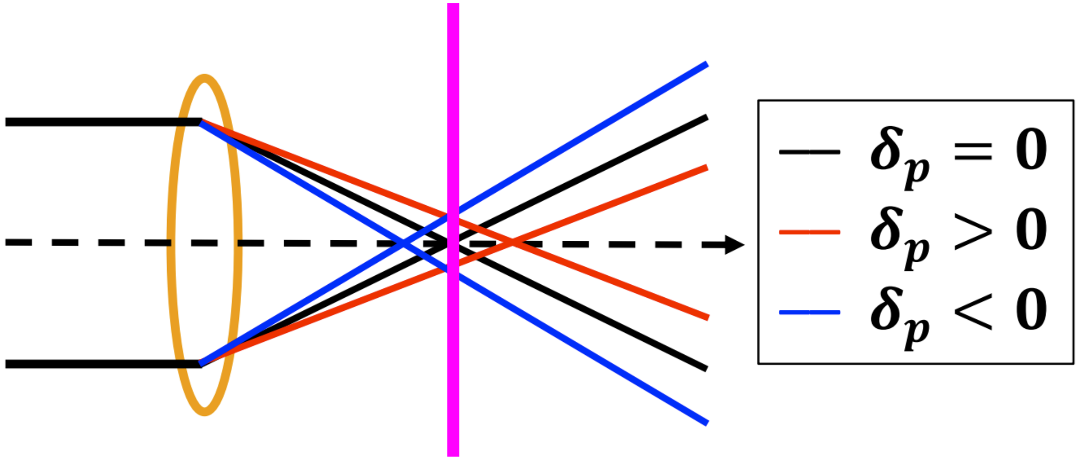


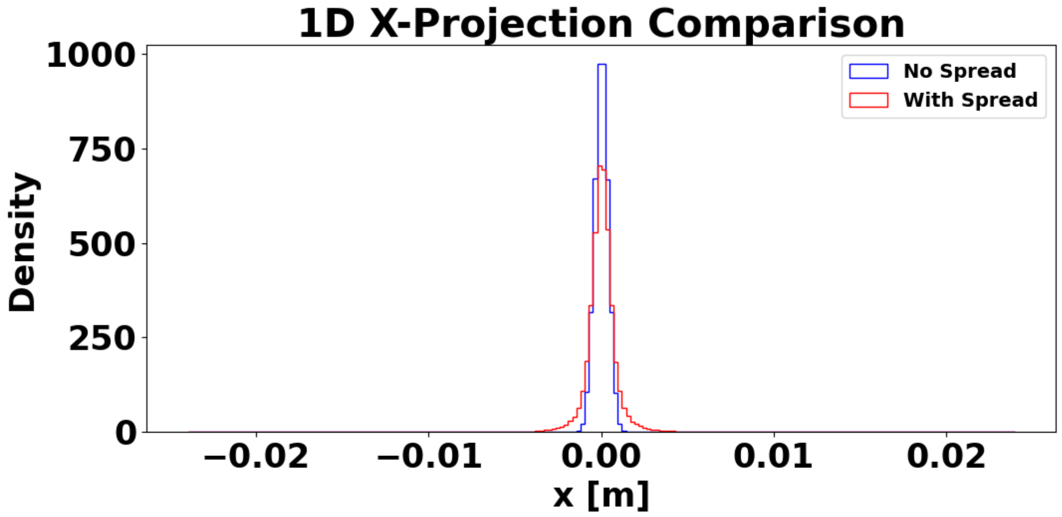
Effect of δ on transverse beam size

$$\delta = \frac{p - p_0}{p_0}$$

- $\boldsymbol{k(\delta)} = \frac{k_0}{1+\boldsymbol{\delta}} : \delta \propto k^{-1}$
- $f = \frac{x}{-\Delta\theta_x} = \frac{y}{\Delta\theta_y} = \frac{B\rho}{\boldsymbol{k(\delta)}l} : \delta \propto f$
- Therefore, δ must be compensated for in every optimization.

Screen location





Results: Simulation

Test algorithm with simulation

It allows the estimation of nonlinear effect, despite the simulation relying only on linear calculation.

$$\boldsymbol{b_{out}} = f_{linear}\left(\boldsymbol{b_{in}} \middle| l_Q, k_Q\right) + \Delta\Sigma_{SC}(\boldsymbol{b_{in}}|l_Q, k_Q)$$

Simulation **Neural Network**

"It reduce the complexity of the function modeled by the NN" (J. Kaiser, PRAB 054601, 2024).

TABLE I. Predicted emittances compared to true values.

Parameter	Ground truth	rms prediction	Reconstruction	Unit
ε_x	2.00	2.47	2.00 ± 0.01	mm-mrad
ε_y	11.45	14.10	10.84 ± 0.04	mm-mrad
ε_{4D}	18.51	34.83[a]	17.34 ± 0.08	mm^2-$mrad^2$

[a]Assumes x-y phase space independence.

Quadrupole scan

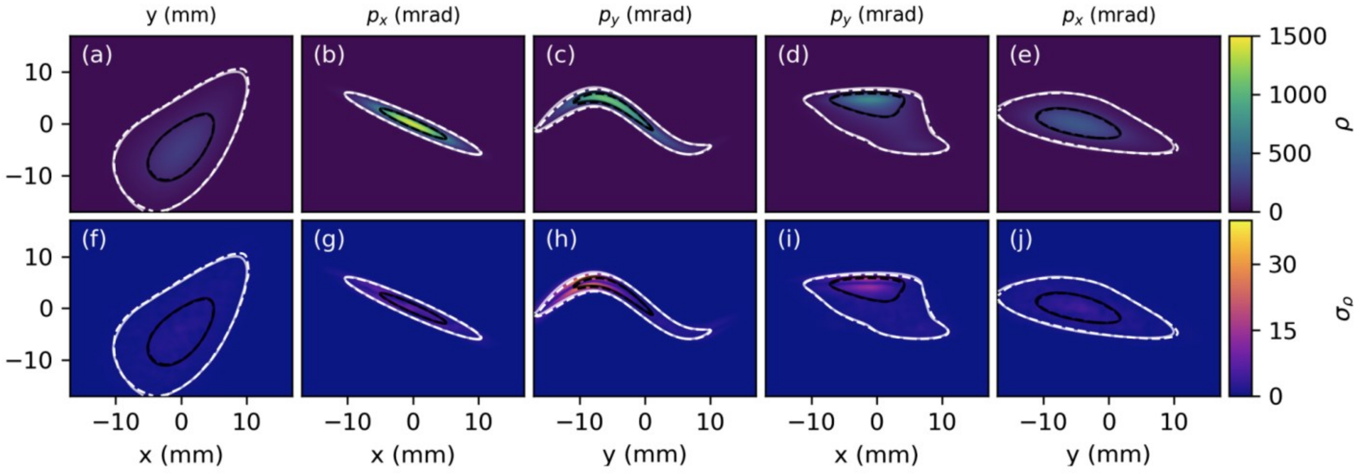


FIG. 2. Comparisons between the synthetic and reconstructed beam probability distributions using our method. (a)–(e) Plots of the mean predicted phase space density projections in 4D transverse phase space. Contours that denote the 50th (black) and 95th (white) percentiles of the synthetic ground truth (dashed) and reconstructed (solid) distributions. (f)–(j) Plots of the predicted phase space density uncertainty.

Results: Experimental

Proof-of-principle experimental @ Argonne Wakefield Accelerator (AWA)

- Well-matched image with nonlinear effect (ex. Space charge)

TABLE II. Predicted emittances from experimental data.

Parameter	rms prediction	Reconstruction	Unit
$\varepsilon_{x,n}$	4.18 ± 0.71	4.23 ± 0.02	mm-mrad
$\varepsilon_{y,n}$	3.65 ± 0.36	3.42 ± 0.02	mm-mrad

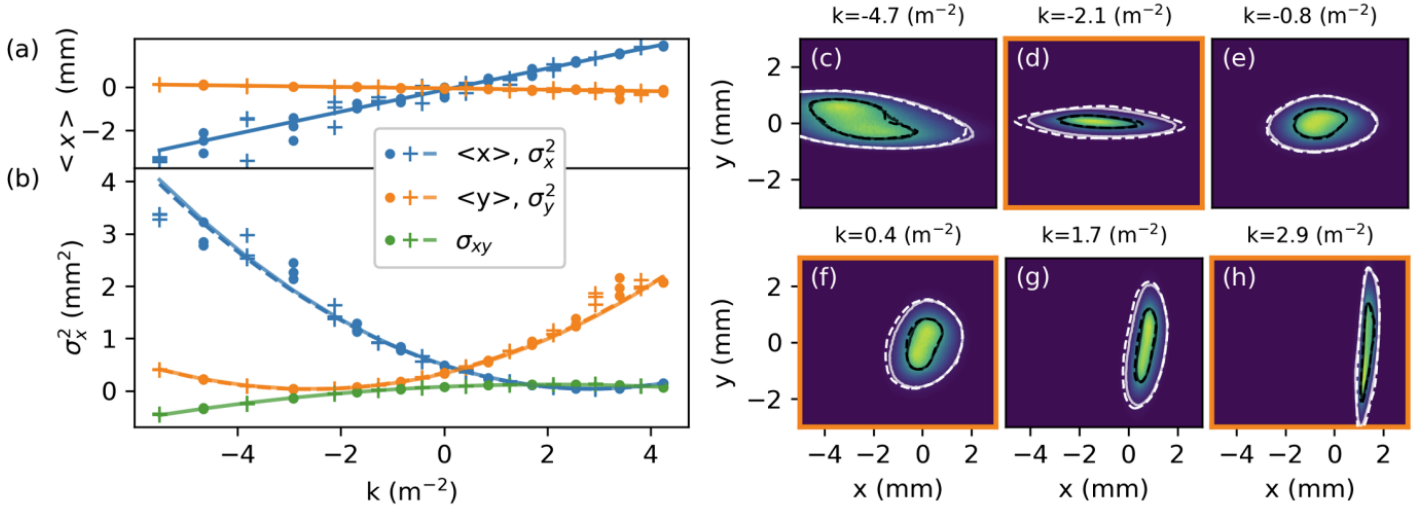


FIG. 4. Reconstruction results from experimental measurements at AWA. Comparison between measured and predicted beam centroids (a) and second-order beam moments (b) on the diagnostic screen as a function of geometric quadrupole focusing strength (k). Points denote training samples and crosses denote test samples. Dashed line shows second order polynomial fit of training data and solid line shows predictions from image-based phase space reconstruction. We also compare (c)–(h) screen images and reconstructed predictions for a subset of quadrupole strengths. Contours denote the 50th (black) and 95th (white) percentiles of the measured (dashed) and predicted (solid) screen distributions. Orange borders denote test samples.

Discussion

Key achievement

- Novel Framework: Differentiable simulation & Neural Network
- Standard equipment: Only use single QM and screen.

Limitations

- Incomplete uncertainly quantification.
- There is limitation to measure δ accurately.
- High memory consumption.

Future works

- Enhanced uncertainty modeling
- Memory optimization
- Full 6D phase space characterization using TDS and dipole.