USING NEURAL-NETWORK ANSATZ FOR THE GENERATING FUNCTION IN THE HAMILTON-JACOBI EQUATION TO OBTAIN SYMPLECTIC TRANSFER MAPS FOR ELEMENTS WITH ANY MAGNETIC FIELD CONFIGURATION*

I. Lobach†, Y. Li
Brookhaven National Laboratory, Upton, NY, USA

Abstract

We present a novel method to generate a symplectic transfer map for any beamline element defined by its magnetic vector potential, even when it is known only on a 3D grid. The method uses a neural network (NN) as an ansatz to solve the Hamilton-Jacobi (HJ) equation for the unknown generating function of the second kind. This generating function is chosen to connect the solution of a simpler system with an exact analytical solution (e.g., an ideal hard-edge quadrupole) to the system with a more complex field configuration (e.g., a quadrupole with an Enge fringe field profile). This design dramatically reduces the learning burden on the NN. The learned generating function defines the trajectories and the element's symplectic transfer map via implicit equations for the particle's position and explicit equations for its momenta. The implicit equations are typically solved to machine precision in a few iterations using Newton's method combined with automatic differentiation capability of the NN. The method's accuracy can be conveniently estimated by how well the NN solution satisfies the original Hamiltonian. We validate the method with 1D and 2D examples for drift space, hard-edge quadrupole, and quadrupole with Enge fringe field profile.

INTRODUCTION

Modern accelerator lattices increasingly rely on magnets with strong gradients, combined-function components, and pronounced 3D end effects, making accurate field modeling a central issue for long-term beam-dynamics studies. This trend is evident in recent and upcoming facilities such as NSLS-IIU [1, 2], APS-U, and ALS-U, where complex magnet geometries with large sagitta and overlapping fringe fields challenge standard fast symplectic methods [3, 4]. At the same time, realistic evaluation of dynamic aperture, beam lifetime, and injection efficiency requires stable long-term tracking with maps that remain symplectic over many turns. Direct trajectory integration through magnetic fields on 3D grids [5] can provide high fidelity, but it is often too computationally expensive for routine optimization and large-scale tracking studies. These considerations motivate the development of efficient, field-based, symplectic models for beamline elements with arbitrary magnetic field configurations. Several approaches for approximating symplectic transfer maps with neural networks have been considered before, such as Henon neural networks, Hamiltonian neural networks, SympNets [6–20]. In this paper, we propose a new generating function-based approach.

* Work supported by the U.S. Department of Energy under Contract No. DE-SC0012704, and the Field Work Proposal 2025-BNL-PS040
† ilobach@bnl.gov

METHOD DESCRIPTION

Consider the Hamiltonian system of interest, with canonical coordinates (q, p), where $q = (q_x, q_y)$ and $p = (p_x, p_y)$, and a generally complicated Hamiltonian

$$H = H(s, q, p). \tag{1}$$

Here, we assume the Frenet–Serret coordinate system, s is the reference-particle path length. Further, we introduce a second Hamiltonian system with canonical coordinates (Q, P), where $Q = (Q_x, Q_y)$ and $P = (P_x, P_y)$, and a simpler Hamiltonian

$$K = K(s, Q, P), \tag{2}$$

chosen such that its equations of motion admit an analytical solution. Hereafter, we may also refer to the systems with Hamiltonians H and K as system H and system K, respectively. For example, system K can be a linear hard-edge lattice model, and system H can be a lattice model with nonlinear fringe fields. The goal is to construct a canonical transformation from the simple system K to the more complex system H.

To this end, we postulate the existence of a generating function of the second kind,

$$G = G(s, q, P), \tag{3}$$

which maps (Q, P) to (q, p) via

$$Q = \frac{\partial G}{\partial P}, \tag{4}$$

$$p = \frac{\partial G}{\partial q}. \tag{5}$$

According to Hamiltonian mechanics, the two Hamiltonians are related by

$$\frac{\partial G}{\partial s} + H\left(s, q, \frac{\partial G}{\partial q}\right) = K\left(s, \frac{\partial G}{\partial P}, P\right), \tag{6}$$

where we have already used Eqs. (4),(5). Equation (6) is the Hamilton–Jacobi (HJ) partial differential equation that must be solved to determine $G(s, q, P)$. Note that this is the most general version [21, p. 373], while often only the case of

$K \equiv 0$ is provided [22, p. 147], or $K = K(s, Q)$ [23, p. 260]. In this paper, we propose to solve Eq. (6) directly by using a neural-network (NN) ansatz for the generating function G.

In general, for arbitrary H and K, it is impossible to find global single-valued smooth solution G valid for any s, q, P. However, it is always guaranteed to exist in some finite region of s, q, P, where the canonical transformation is invertible, i.e., where [23, p. 260]

$$0 < \left|\det\left(\frac{\partial^2 G}{\partial q \partial P}\right)\right| < \infty. \tag{7}$$

In the examples provided below in this paper, this condition is satisfied in the considered s, q, P regions.

Now, let us assume we have a NN solution $G(s, q, P)$ for Eq. (6), and we know the analytical solutions for the phase-space trajectories $Q(s)$, $P(s)$ of the simpler system K. Then, at any s, we can find $q(s)$ by numerically solving the implicit equation Eq. (4) for $q(s)$. This can be done very efficiently by using the Newton method and NN auto-differentiation (provided by PyTorch [24], for example). The nth step of the Newton's method constitutes solving the following linear system for $q_x^{(n+1)}$, $q_y^{(n+1)}$,

$$Q_x = \frac{\partial G}{\partial P_x} + \frac{\partial^2 G}{\partial P_x \partial q_x}\Delta q_x^{(n)} + \frac{\partial^2 G}{\partial P_x \partial q_y}\Delta q_y^{(n)}, \tag{8}$$

$$Q_y = \frac{\partial G}{\partial P_y} + \frac{\partial^2 G}{\partial P_y \partial q_x}\Delta q_x^{(n)} + \frac{\partial^2 G}{\partial P_y \partial q_y}\Delta q_y^{(n)}. \tag{9}$$

where the derivatives of G are evaluated at $q = q^{(n)}$, and

$$\Delta q_x^{(n)} = q_x^{(n+1)} - q_x^{(n)}, \tag{10}$$

$$\Delta q_y^{(n)} = q_y^{(n+1)} - q_y^{(n)}. \tag{11}$$

Assuming that the simpler system K is relatively close in its properties to the more complex system H, one can use the initial guess $q^{(0)} = Q$ and the method will typically converge to machine precision within a few iterations. The obtained $q(s)$, $p(s)$ will be symplectic to machine precision.

The generating function method for symplectic tracking has been considered before [25, 26], but only for small steps, using Taylor expansion. Our method utilizes neural networks to find G valid for the entire s range of the considered magnetic element.

Boundary Conditions

Without loss of generality, G can be represented as

$$G(s, q, P) = G(0, q, P) + \Gamma(s, q, P), \tag{12}$$

with $\Gamma(0, q, P) \equiv 0$. There is some gauge freedom for the generating function G. Namely, we can choose any Dirichlet boundary condition at $s = 0$, $G(0, q, P)$, while preserving the same phase-space trajectories for q, p. A very convenient choice, especially in the case where q, p are close to Q, P, is the following,

$$G(s, q, P) = q_x \cdot P_x + q_y \cdot P_y + \Gamma(s, q, P). \tag{13}$$

Indeed, one can verify using Eqs. (4) and (5), that this allows one to seamlessly connect Q, P to q, p at $s = 0$ in the following way [26],

$$Q_{x0} = q_{x0}, \quad Q_{y0} = q_{y0}, \quad P_{x0} = p_{x0}, \quad P_{y0} = p_{y0}, \tag{14}$$

thus turning $Q(s), P(s)$ into fully known functions of s and initial q_0, p_0, where $q_0 = (q_{x0}, q_{y0})$, $p_0 = (p_{x0}, p_{y0})$, and the subscript "0" refers to the values at $s = 0$.

Remark about an Alternative Approach

We want to briefly mention an alternative approach to finding G which will be more thoroughly discussed in a separate paper. By considering dG using the chain rule, the Hamiltonian equations of motion for q and p, and Eq. (6), one can show that along the phase-space trajectory

$$\frac{dG}{ds} = \left(p_x \frac{\partial H}{\partial p_x} + p_y \frac{\partial H}{\partial p_y} - H\right) - \left(Q_x \frac{\partial K}{\partial Q_x} + Q_y \frac{\partial K}{\partial Q_y} - K\right) \equiv \mathscr{L}_{HK}(s), \tag{15}$$

where the first term coincides with the Lagrangian of the system H. This is why we chose the notation $\mathscr{L}_{HK}(s)$ meaning the Lagrangian of system H "shifted" by system K. To find G in the gauge corresponding to Eqs. (13) and (14), one simply needs to compute it in the following way

$$G(s) = q_{x0} \cdot p_{x0} + q_{y0} \cdot p_{y0} + \int_0^s \mathscr{L}_{HK}(s')\, ds'. \tag{16}$$

Equation (16) can be easily incorporated into various integrators, e.g., symplectic 4th order Gauss-Legendre (GL4) integrator [27]. Then, one can use GL4 to numerically integrate many (e.g., millions) phase space trajectories. Thanks to Eq. (15), at all trajectory points, one will know the value of G and its derivatives, because of Eqs. (4), (5), and (6). Such data can be very efficiently used to find a NN fit for G. In fact, the fit is not limited to NNs. One could also consider, e.g., B-spline regression. The only important requirement is that the fit has smooth derivatives up to the second order.

DRIFT AND QUADRUPOLE EXAMPLES

First, to develop confidence in the proposed method, we test it using trivial examples of a drift element and an ideal thick quadrupole, where the exact analytical solutions are well known. In these two examples we put $K = 0$. Therefore, Q and P become constants along phase-space trajectories,

$$Q(s) = q_0, \qquad P(s) = p_0. \tag{17}$$

Further, since there is no x–y coupling, the generating function separates into two contributions,

$$G(s, q, P) = G_x(s, q_x, P_x) + G_y(s, q_y, P_y), \tag{18}$$

Therefore, for brevity, we will consider a 1D case in this section for the x-axis. We will consider the following Hamiltonian,

$$H(q_x, p_x) = \frac{p_x^2}{2} + \frac{k\, q_x^2}{2}, \tag{19}$$

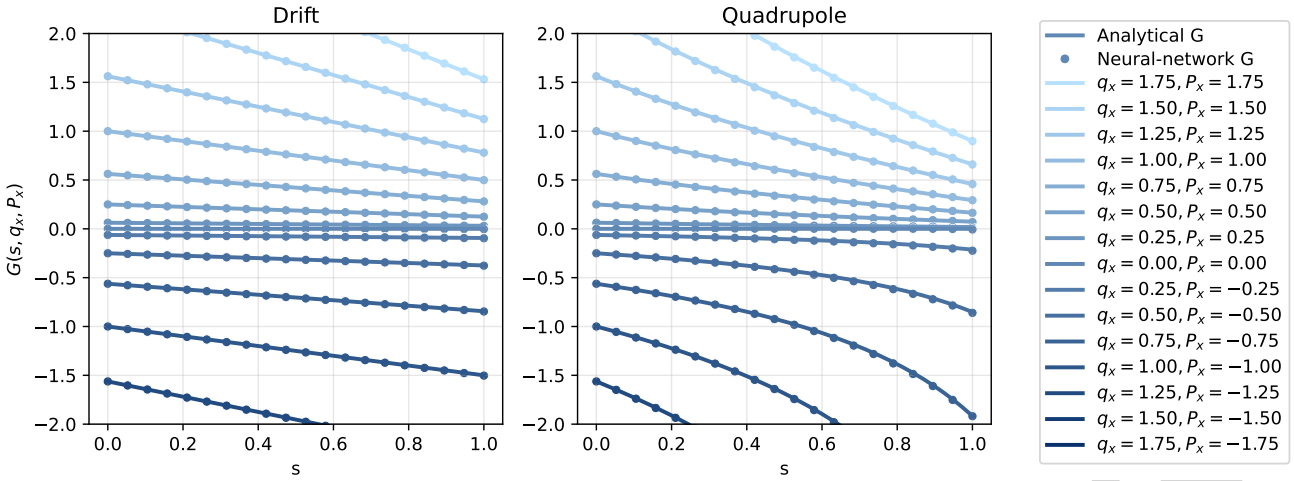

Figure 1: Comparison of the exact analytical generating function (solid lines) and the NN-learned generating function (markers) for a drift element and a thick focusing quadrupole element.

where the paraxial approximation is assumed, k is the quadrupole geometric strength, and we have put the total particle's momentum $p_{\text{tot}} = 1$ for simplicity. For the illustrative examples in this paper, we use dimensionless variables throughout. The drift-space Hamiltonian can be obtained from Eq. (19) at $k = 0$.

Both drift-element and ideal-quadrupole cases can be described by a linear symplectic transfer map,

$$\begin{pmatrix} q_x(s) \\ p_x(s) \end{pmatrix} = \begin{pmatrix} a(s) & b(s) \\ c(s) & d(s) \end{pmatrix} \begin{pmatrix} Q_x \\ P_x \end{pmatrix}, \tag{20}$$

keeping in mind that $Q_x = q_{x0}$, $P_x = p_{x0}$.

By searching for $G_x(s, q, P)$ with a quadratic form as the ansatz, one can find that

$$G_x(s, q_x, P_x) = \frac{c(s)}{2\,a(s)} q_x^2 + \frac{1}{a(s)} q_x \cdot P_x - \frac{b(s)}{2\,a(s)} P_x^2, \tag{21}$$

For a drift space, the transfer matrix is

$$M_{\text{drift}}(s) = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix}, \tag{22}$$

and the generating function is

$$G_x^{(\text{drift})} = q_x \cdot P_x - \frac{s}{2} P_x^2. \tag{23}$$

Let us only consider the focusing quadrupole polarity. Within an ideal thick focusing quadrupole, the transfer matrix is

$$M_{\text{quad}}(s) = \begin{pmatrix} \cos(\sqrt{k}\,s) & \frac{1}{\sqrt{k}} \sin(\sqrt{k}\,s) \\ -\sqrt{k} \sin(\sqrt{k}\,s) & \cos(\sqrt{k}\,s) \end{pmatrix} \tag{24}$$

and the generating function is

$$G_x^{(\text{quad})} = \frac{q_x \cdot P_x}{\cos(\sqrt{k}\,s)} - \frac{\tan(\sqrt{k}\,s)}{2\sqrt{k}} \left(k\, q_x^2 + P_x^2\right). \tag{25}$$

For the proposed NN-based method, we used the following ansatz for the generating function

$$G_x(s, q_x, P_x) = q_x \cdot P_x + s \cdot f(s, q_x, P_x), \tag{26}$$

where $f(s, q_x, P_x)$ is a fully-connected multi-layer perceptron, consisting of an input layer for s, q_x, P_x followed by four hidden layers of width 32 with tanh activation functions, and a final scalar output layer. We are using the factor s in front of f in Eq. (26) to ensure the convenient properties from Eqs. (13) and (14) by construction. Other choices for the factor in front of f are possible, e.g., $1 - e^{-s}$, see [17]. During the training, we use the Mean Squared Error (MSE) loss function describing the differences between the left and right-hand sides of Eq. (6) at the random sampling points s, q, P,

$$L = \frac{1}{N} \sum_{\substack{s,q,P \\ \text{random} \\ \text{samples}}} \left| \frac{\partial G}{\partial s} + H\left(s, q, \frac{\partial G}{\partial q}\right) - K\left(s, \frac{\partial G}{\partial P}, P\right) \right|^2, \tag{27}$$

where in the specific examples of the drift element and the thick focusing quadrupole, we use $K \equiv 0$, and q, P are one-dimensional, q_x, P_x. For random sampling in Eq. (27) we used uniform distributions for s, q_x, P_x with the ranges $(0, 1)$, $(-5, 5)$, $(-5, 5)$, respectively. The considered quadrupole's geometric strength was $k = 1.0$. We used 40 000 randomly sampled batches to train the models. The batch size N was 32 768 . The learning rate was 10^{-3}. Adam optimizer from PyTorch was used. The derivatives of G required for the loss function in Eq. (27) were computed using the auto-differentiation capability of PyTorch.

For both the drift element and the quadrupole, the learned NN representation of the generating function G agrees very well with the exact analytical expression. Please see Fig. 1, where we compared G as functions of s for a variety of q_x, P_x value pairs. The NN-based G predictions are represented by markers only to help visually distinguish them from the exact analytical results. We only compare the results for

G in this section, trajectory reconstruction from G will be illustrated in the next section.

QUADRUPOLE WITH ENGE PROFILE

In this section, we consider a more practical example of a quadrupole with Enge field profile [28] as the more complex system H, while the simpler system K with analytical solutions is a quadrupole with a constant geometric strength k_{eff} equal to the average geometric strength of the Enge-profile quadrupole. We consider two transverse dimensions q_x and q_y in the paraxial approximation and such vector potential gauge, where $A_y \equiv 0$. Then, according to Ref. [4], up to the 4th order,

$$A_x = -\frac{xy^2}{2} k\, F'(s), \tag{28}$$

$$A_y = 0, \tag{29}$$

$$A_s = -\frac{x^2 - y^2}{2} k\, F(s) + \frac{x^4 - 6x^2y^2 + y^4}{48} k\, F''(s), \tag{30}$$

where we put the particle's charge $e = -1$ and the particle's total momentum $p_{\text{tot}} = 1$ for simplicity; k is the peak geometric strength observed in the center of the quadrupole, $F(s)$ is the Enge profile, normalized to unity in the center of the quadrupole, $F'(s)$ and $F''(s)$ are the first and second-order derivatives with respect to s, which are readily obtained analytically for the Enge profile. In this example, we considered

$$F(s) = \frac{1 + e^{-5}}{1 + e^{-5+50\,(s-0.5)^2}}. \tag{31}$$

The chosen profile is shown in Fig. 2 as the black solid line; only its shape is shown, not its actual vertical scale. We use $k = 1.0$, and the corresponding k_{eff} is

$$k_{\text{eff}} = \int_0^1 k\, F(s) ds = 0.6240, \tag{32}$$

which was used in the Hamiltonian of the simpler system K,

$$K = \frac{1}{2}\left(P_x^2 + P_y^2\right) + \frac{k_{\text{eff}}}{2}\left(Q_x^2 - Q_y^2\right), \tag{33}$$

while the Hamiltonian of the more complex system with Enge field profile was [26, 29]

$$H = \frac{1}{2}\left(\left(p_x - A_x\right)^2 + \left(p_y - A_y\right)^2\right) - A_s, \tag{34}$$

with the vector potential components given by Eqs. (28–30). An ansatz similar to Eq. (26) was used, but extended to two transverse dimensions,

$$G_{\text{enge}} = q_x \cdot P_x + q_y \cdot P_y + s \cdot f_{\text{enge}}(s, q_x, q_y, P_x, P_y), \tag{35}$$

where the neural network f_{enge} was a fully-connected multilayer perceptron with 2 hidden layers with the width of 256, using tanh activation throughout. The loss function was constructed in the same way as in Eq. (27). We used variable batch size starting from 32 768 and rising to ≈ 100 000 during the training. We used variable learning rate starting from 10^{-3} and dropping to 10^{-4}. The training was divided into epochs, each epoch included 10^7 random samples of s, q_x, q_y, P_x, P_y. We used uniform distributions with ranges $(0, 1)$ for s and $(-0.65, 0.65)$ for q_x, q_y, P_x, and P_y. The main training phase included ≈ 3500 epochs with PyTorch's Adam optimizer. After that, we also used PyTorch's LBFGS optimizer with ≈ 2 million samples (on a uniform grid) for another 100 epochs.

After obtaining the NN representation of G_{enge}, we were able to find q_x and q_y by solving Eq. (4) with the Newton's method [Eqs. (8),(9)], and, then, p_x and p_y using Eq. (5). As the initial guess for the Newton's method, we used Q_x and Q_y from the simpler hard-edge model with constant k_{eff} spanning from $s = 0$ to $s = 1$. The resulting trajectories are symplectic to machine precision, when the Newton's method converges to machine precision. A few such trajectories are shown in Fig. 2 (markers). The required number of Newton iterations varied from 6 to 14. The resulting phase-space trajectories are quite close to the trajectories computed using the 4th order Runge-Kutta method [30] with 600 integration steps (solid lines). The analytical trajectories for the simpler hard-edge model with k_{eff} are shown as the dashed lines.

Hamilton-Jacobi Residuals

In general, the obtained NN generating function $\tilde{G}$ may not precisely correspond to the system H. However, by design, it precisely corresponds to some Hamiltonian system with a Hamiltonian $\tilde{H}$ that is close to H. The difference $\tilde{H} - H$ serves as a convenient fundamental measure of how well the NN-learned $\tilde{G}$ describes the real system. $\tilde{G}$ does not provide an explicit expression for $\tilde{H}$. To find the value of $\tilde{H}(s, q, p)$ at some s, q, p, in general, one has to solve the following system of implicit equations for P^*,

$$p = \frac{\partial \tilde{G}}{\partial q}(s, q, P^*). \tag{36}$$

This can be one or more equations, depending on the number of considered dimensions. Then, knowing P^*, one can compute

$$\tilde{H}(s, q, p) = K(s, \left.\frac{\partial \tilde{G}}{\partial P}\right|_{P=P^*}, P^*) - \frac{\partial \tilde{G}}{\partial s}(s, q, P^*). \tag{37}$$

However, if one does not need to compute $\tilde{H}$ at specific q and p, the residual $\tilde{H} - H$ can be computed at certain specified q and P explicitly, without the need to solve the implicit equation Eq. (36). Indeed, at certain specified q and P, the residuals $\tilde{H} - H$ coincide with the terms in the sum in the loss function in Eq. (27). Therefore, we arrive at the conclusion that the loss function in Eq. (27) is not only a measure for solving the partial differential equation Eq. (6), but also a fundamental measure describing the deviation of the NN-learned Hamiltonian $\tilde{H}$ from the true Hamiltonian H.

In Fig. 3, we plot MSE($\tilde{H} - H$) computed at 200 different values of s. At each s slice, we used a uniform 4D grid for q_x, q_y, P_x, P_y with 20 grid points along each variable, from

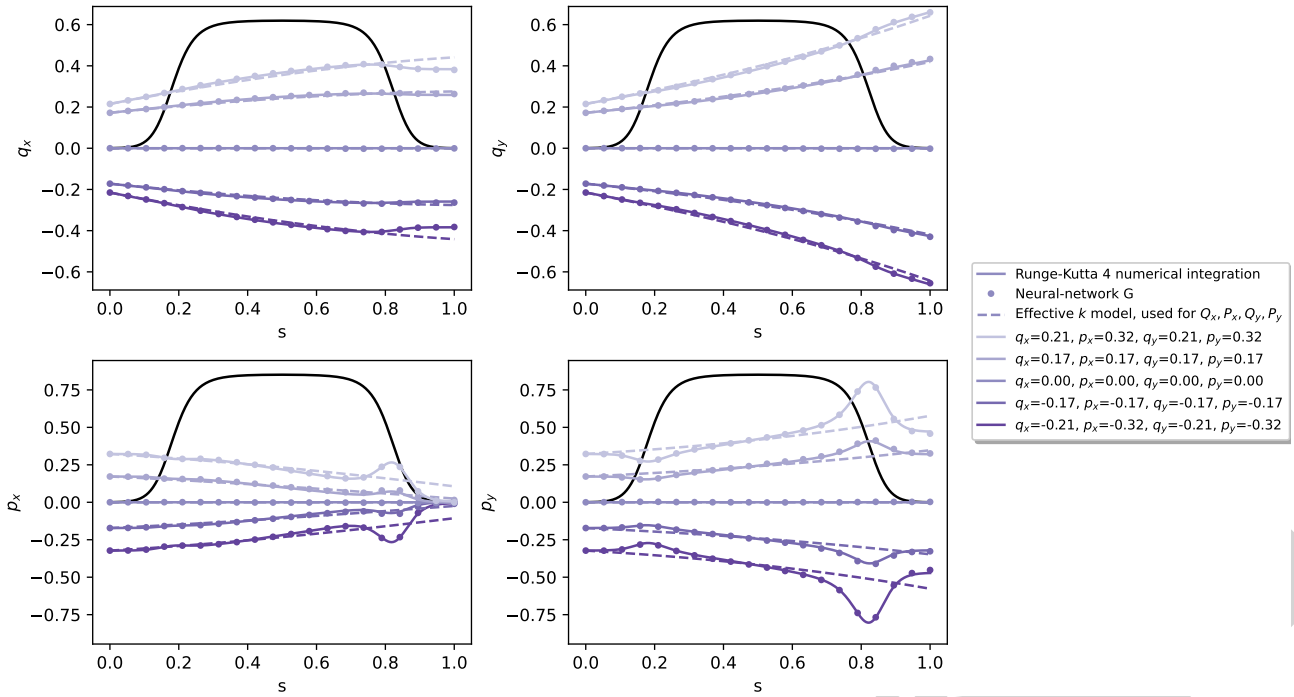

Figure 2: Comparison of the phase space trajectories in a quadrupole with Enge field profile computed by 4th order Runge-Kutta method (solid lines) and the trajectories obtained from the NN generating function G (markers). The dashed lines represent the analytical trajectories for the hard-edge quadrupole model with constant effective geometric strength k_{eff} spanning from $s = 0$ to $s = 1$. The black solid lines represent the Enge profile shape.

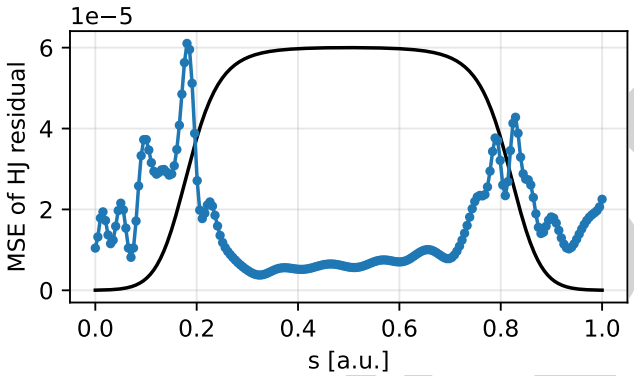

Figure 3: Mean Squared Error of Hamilton-Jacobi residuals $\tilde{H} - H$, computed at 200 values of s for the example of a quadrupole with Enge field profile.

−0.65 to +0.65. The resulting plotted MSE deviations in Fig. 3 are generally small (a characteristic scale for comparison is ∼ 1). Slight spikes are observed in the fringe field regions.

CONCLUSIONS AND OUTLOOK

We presented a method to construct symplectic transfer maps for beamline elements with arbitrary magnetic-field configurations by representing the generating function of the second kind with a NN ansatz and solving the corresponding Hamilton–Jacobi equation. The approach is especially attractive when a nearby simpler Hamiltonian with known analytical solution can be used as a reference system, because this choice substantially reduces the complexity of the learned correction. Once the generating function is known, particle trajectories and the corresponding transfer map can be recovered by solving implicit equations for the coordinates and then evaluating explicit expressions for the momenta, thereby preserving symplecticity to machine precision when Newton's method converges to that level.

The method was validated in progressively more demanding examples. For the drift element and the ideal thick quadrupole, the learned generating function reproduced the exact analytical result with high accuracy. For a quadrupole with an Enge fringe-field profile, the NN generating function, combined with a hard-edge quadrupole reference model, yielded phase-space trajectories that are in close agreement with direct Runge-Kutta integration, while retaining the important advantage of a symplectic map-based description. The residual of the Hamilton–Jacobi equation also provides a natural and convenient metric for monitoring the quality of the learned solution.

The presented method can be used even when the vector potential is available only on a 3D field grid, which is often the case in realistic magnets in modern and future accelerators. In a separate paper, we will consider more efficient training strategies, including the trajectory-based construction, outlined in the Method Description section, and apply it to study long-term beam dynamics in NSLS-II Upgrade lattice and improve nonlinear lattice optimization speed.

ACKNOWLEDGEMENTS

The authors are grateful to Yue Hao, Qi Tang, and Victor Smaluk for fruitful discussions on the subject.

REFERENCES

[1] M. Song and T. Shaftan, "Design study of a low emittance complex bend achromat lattice", *Phys. Rev. Accel. Beams*, vol. 27, no. 6, p. 061601, Jun. 2024. doi:10.1103/PhysRevAccelBeams.27.061601

[2] V. Smaluk *et al.*, "Novel magnet lattice for the high-brightness upgrade of NSLS-II", *J. Phys.: Conf. Ser.*, vol. 3010, no. 1, p. 012036, May 2025. doi:10.1088/1742-6596/3010/1/012036

[3] C. E. Mitchell and A. J. Dragt, "Accurate transfer maps for realistic beam-line elements: straight elements", *Phys. Rev. ST Accel. Beams*, vol. 13, no. 6, p. 064001, Jun. 2010. doi:10.1103/PhysRevSTAB.13.064001

[4] R. Lindberg and M. Borland, "Fringe field maps for symplectic models of general cartesian dipoles", *Phys. Rev. Accel. Beams*, vol. 26, no. 11, p. 114001, Nov. 2023. doi:10.1103/PhysRevAccelBeams.26.114001

[5] A. W. Chao, M. Tigner, H. Weise, and F. Zimmermann, *Handbook of accelerator physics and engineering*. Singapore: World Scientific, 2023. doi:10.1142/13229

[6] D. Turaev, "Polynomial approximations of symplectic dynamics and richness of chaos in non-hyperbolic area-preserving maps", *Nonlinearity*, vol. 16, no. 1, pp. 123–135, 2003. doi:10.1088/0951-7715/16/1/308

[7] J. W. Burby, Q. Tang, and R. Maulik, "Fast neural Poincaré maps for toroidal magnetic fields", *Plasma Phys. Control. Fusion*, vol. 63, no. 2, p. 024001, Dec. 2020. doi:10.1088/1361-6587/abcbaa

[8] P. Jin, Z. Zhang, A. Zhu, Y. Tang, and G. E. Karniadakis, "SympNets: intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems", *Neural Netw.*, vol. 132, pp. 166–179, 2020. doi:10.1016/j.neunet.2020.08.017

[9] C.-K. Huang *et al.*, "Symplectic neural surrogate models for beam dynamics", *J. Phys.: Conf. Ser.*, vol. 2687, no. 6, p. 062026, Jan. 2024. doi:10.1088/1742-6596/2687/6/062026

[10] E. G. Drimalas, F. Fraschetti, C. Huang, and Q. Tang, "Symplectic neural network and its application to charged particle dynamics in electromagnetic fields", *Phys. Plasmas*, vol. 32, no. 10, p. 103901, 2025. doi:10.1063/5.0283551

[11] A. Ivanov and I. Agapov, "Physics-based deep neural networks for beam dynamics in charged particle accelerators", *Phys. Rev. Accel. Beams*, vol. 23, no. 7, p. 074601, Jul. 2020. doi:10.1103/PhysRevAccelBeams.23.074601

[12] J. Wan, J. Qiang, and Y. Hao, "Symplectic machine learning model for fast simulation of space-charge effects", *Phys. Rev. Accel. Beams*, vol. 28, no. 7, p. 074602, Jul. 2025. doi:10.1103/bhpv-bcqk

[13] D. Di Croce, M. Giovannozzi, C. E. Montanari, T. Pieloni, S. Redaelli, and F. F. Van der Veken, "Assessing the performance of deep learning predictions for dynamic aperture of a hadron circular particle accelerator", *Instruments*, vol. 8, no. 4, p. 50, 2024. doi:10.3390/instruments8040050

[14] M. Rautela, A. Williams, and A. Scheinker, "A conditional latent autoregressive recurrent model for generation and forecasting of beam dynamics in particle accelerators", *Sci. Rep.*, vol. 14, no. 1, p. 18157, 2024. doi:10.1038/s41598-024-68944-0

[15] V. Duruisseaux, J. W. Burby, and Q. Tang, "Approximation of nearly-periodic symplectic maps via structure-preserving neural networks", *Sci. Rep.*, vol. 13, no. 1, p. 8351, 2023. doi:10.1038/s41598-023-34862-w

[16] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks", *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[17] M. Mattheakis, D. Sondak, A. S. Dogra, and P. Protopapas, "Hamiltonian neural networks for solving equations of motion", *Phys. Rev. E*, vol. 105, no. 6, p. 065305, Jun. 2022. doi:10.1103/PhysRevE.105.065305

[18] A. Edelen *et al.*, "Opportunities in machine learning for particle accelerators", *arXiv*, Nov. 2018. doi:10.48550/arXiv.1811.03172

[19] A. J. Linot, J. W. Burby, Q. Tang, P. Balaprakash, M. D. Graham, and R. Maulik, "Stabilized neural ordinary differential equations for long-time forecasting of dynamical systems", *J. Comput. Phys.*, vol. 474, p. 111838, 2023. doi:10.1016/j.jcp.2022.111838

[20] C. Garcia-Cardona and A. Scheinker, "Machine learning surrogate for charged particle beam dynamics with space charge based on a recurrent neural network with aleatoric uncertainty", *Phys. Rev. Accel. Beams*, vol. 27, no. 2, p. 024601, Feb. 2024. doi:10.1103/PhysRevAccelBeams.27.024601

[21] H. Goldstein, C. P. Poole, and J. L. Safko, *Classical mechanics*. Boston, MA, USA: Addison-Wesley, 2001.

[22] L. D. Landau and E. M. Lifshitz, *Mechanics*. Oxford, UK: Butterworth-Heinemann, 1976.

[23] V. I. Arnold, *Mathematical methods of classical mechanics*. New York, NY, USA: Springer-Verlag, 1989.

[24] A. Paszke *et al.*, "PyTorch: an imperative style, high-performance deep learning library", *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[25] J. Bahrdt and G. Wuestefeld, "A Taylor-expanded generating function for particle motion in arbitrary magnetic fields", in *Proc. EPAC'92*, Berlin, Germany, Mar. 1992, pp. 670–673.

[26] J. Li *et al.*, "Symplectic tracking through curved three-dimensional fields by a method of generating functions", *Phys. Rev. Accel. Beams*, vol. 28, no. 9, p. 094001, Sep. 2025. doi:10.1103/sgl1-17y8

[27] J.-M. Sanz-Serna and M.-P. Calvo, *Numerical Hamiltonian problems*. Mineola, NY, USA: Courier Dover Publications, 2018.

[28] H. A. Enge, "Deflecting magnets", in *Focusing of Charged Particles*, A. Septier, Ed. New York, NY, USA: Academic Press, 1967, pp. 203–264.

[29] S. Y. Lee, *Accelerator physics*. Singapore: World Scientific, 2012.

[30] J. C. Butcher, *Numerical methods for ordinary differential equations*. Chichester, UK: John Wiley & Sons, Jul. 2016. doi:10.1002/9781119121534