Reservoir Computing for Prediction of Beam Evolution in Particle Accelerators

Heidi Komkov
Department of Electrical & Computer Engineering
University of Maryland
College Park, MD 20740
heidib@umd.edu

Levon Dovlatyan
Department of Physics
levondov@umd.edu

Artur Perevalov
Department of Physics
pereval@umd.edu

Daniel P. Lathrop
Department of Physics and
Institute for Research in Electronics and Applied Physics

Abstract

Prediction of a charged particle beam's trajectory in a particle accelerator can be a challenge when the beam is subjected to nonlinear forces. We utilize a reservoir computer to predict the evolution of the transverse properties of a high-current electron beam as it travels through a storage ring. Using a WARP model of the University of Maryland Electron Ring, we train the reservoir using images of the beam's cross-sectional profile during the first turn to predict the second and third turns. This approach learns the dynamics of the system using only beam data in a time series, without any knowledge of the accelerator itself.

1 Introduction

Particle accelerators are used in a wide variety of research applications as well as in industry and medicine. A challenge common to all accelerators is tracking and measuring beam parameters as the beam evolves from its source, through a long distance of beam pipe, to its final target. Diagnostics along this path are only present in a limited number of fixed locations. Machine learning techniques are gaining popularity to perform inference tasks such as "virtual diagnostics," or estimation of what instruments would read in locations where they cannot be placed [1].

In a particle accelerator, a particle bunch is confined in the beam pipe by a periodic focusing system with linear restoring forces. However, nonlinearities arise from a number of sources, including nonlinear magnets, nonidealities of physical components, beam-beam effects, and Coulomb repulsion between the particles (an effect known as space charge). These effects on the particle beam combine to create a dynamical system for which no complete explicit model exists. Prediction of the evolution of a particle beam can therefore be a challenge. Reservoir computing is a machine learning method in which a recurrent neural network (the reservoir) with a single trained linear output layer is used to learn and predict the dynamics of a system. This approach was first developed by [2] as an echo state network and by [3] as a liquid state machine. In this paper we use a reservoir computer to predict the transverse beam evolution of an electron beam in the University of Maryland Electron Ring.

2 Reservoir Computing

The reservoir computer consists of an input layer, the reservoir, and an output layer. The reservoir is excited by input vector U_{n} which is coupled to the nodes within using matrix W_{in}. Each node performs a nonlinear transformation of the sum of its inputs, typically a hyperbolic tangent function.

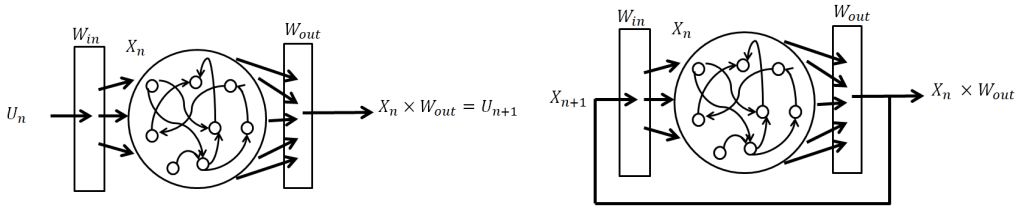


Figure 1: The reservoir computer consists of input matrix W_{in}, a recurrent neural network called the "reservoir", and a single trained linear output layer, W_{out}. The reservoir remains fixed, and only W_{out} is trained, which allows the reservoir to run quickly and at a low computational cost. The reservoir computer is trained using time series data and can be used for prediction tasks.

As it is difficult to train the weights in a recurrent neural network through backpropagation due to issues with vanishing or exploding gradients, the reservoir computer leaves the connectivity and weights in reservoir fixed after random initialization. Rather, the reservoir computer uses a single trained, linear output layer called W_{out} to map the states of all the nodes in the reservoir to the desired output, an operation achieved with a single matrix multiplication. The reservoir during training is updated using:

$$X_{n+1} = (1-\alpha)X_n + \alpha \tanh\left(X_n A^T + U_n W_{\text{in}}^T\right) \tag{1}$$

Here, X is the reservoir state, U is the input, and α is the leakage parameter, a measure of how much memory is retained in the reservoir between steps. A is the adjacency matrix, which is a randomly initialized sparse matrix with real eigenvalues. During training, the values of W_{out} are optimized using linear or logistic regression. Once the model is trained, the output is wrapped back into the input to make the model free-running.

$$X_{n+1} = (1-\alpha)X_n + \alpha \tanh\left(X_n A^T + (X_n W_{\text{out}}) W_{\text{in}}^T\right) \tag{2}$$

In effect, the reservoir is nonlinearly casting the input vector and its recent history to a high dimensional state space. The output layer is then learning which combination of those transformations produce a useful result. The recurrent nature of the reservoir causes information to circulate for some time before fading out—this is often called the echo state or short-term memory property. This temporal activity makes the reservoir particularly suited to processing time-series data, because the reservoir has the capability to create associations between successive points in time. Reservoir computing has been used to predict chaotic systems from time series data [4].

3 Accelerator Description and Simulation Setup

The **U**niversity of **M**aryland **E**lectron **R**ing (UMER) is a 10 keV storage ring designed for the study of physics of high-current electron beams. UMER consists of 18 nearly identical sections 32 cm in length, each containing the same arrangement of steering and focusing magnets as shown in Figure 2. Each section consists of dipole magnets, which steer the beam, and quadrupole magnets, which confine the beam within the pipe. Diagnostics include imaging screens at regular locations around the ring. The electron bunch current can be varied from $150\,\mu A$ to 80 mA, with larger currents corresponding to stronger nonlinear forces due to space charge.

While imaging screens placed in the beam's path are a destructive diagnostic, they provide a wealth of information about the beam's transverse characteristics. UMER has 11 phosphor screens which can be inserted into the ring to intercept the beam on its first turn. In 3 locations, an electrostatic deflector can kick the beam into a screen located to the side, allowing any turn to be visualized. More details about UMER can be found in [5] and [6].

The size of the electron beam is governed by the envelope equations:

$$x''(s) + k^2 x(s) - \frac{2K_{sp}}{[x(s) + y(s)]} - \frac{\epsilon_x^2}{x^3(s)} = 0 \tag{3}$$

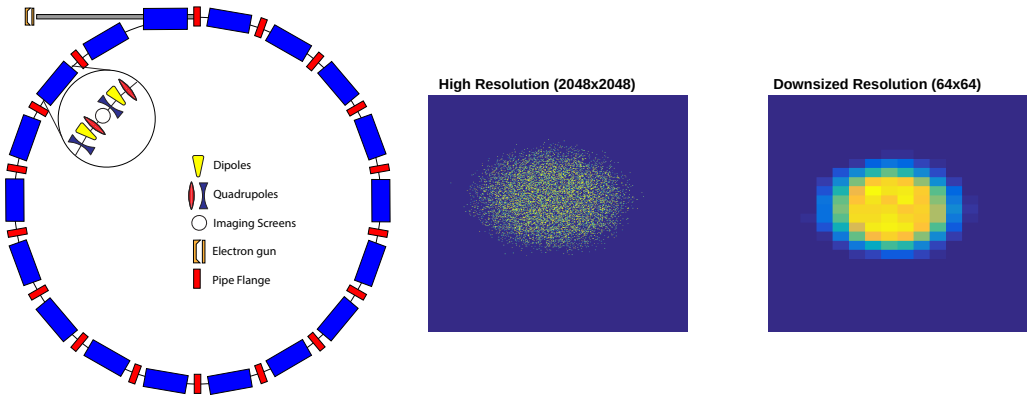
$$y''(s) + k^2 y(s) - \frac{2K_{sp}}{[x(s) + y(s)]} - \frac{\epsilon_y^2}{y^3(s)} = 0 \tag{4}$$

where k is the linear restoring force provided by quadrupoles, K_{sp} is the beam perveance (space charge) parameter, and $\epsilon_{x,y}^2$ is the emittance, a pressure-like term [7]. These are all constants set by the initial conditions of the beam and the settings of magnets in the accelerator lattice.

The beam perveance, a dimensionless parameter proportional to the beam current, determines the amount of the nonlinear Coulomb repulsion force in the system, where $K_{sp} = 0$ is a purely linear system. For the simulations in this paper a 0.6 mA beam with $K_{sp} = 9.029 \times 10^{-6}$ is used.

The results here are from a physics-based simulation of UMER made in WARP, a 3D particle-in-cell code for modeling plasmas and high current particle beams [8]. Simulations are initialized with $10,000$ particles in a semi-Gaussian initial beam distribution, and are checked to converge with increasing particle number.

Images of the transverse particle distribution at 288 evenly spaced locations around the ring are extracted from WARP. While WARP tracks every particle individually, the images are purposely downsampled to 15x15 pixels as shown in Figure 2 (b) for the sake of the reservoir speed. Each pixel represents a physical size of about 400 microns. Pixel values are normalized to be between 0 and 1. Images are generated at a close enough spacing to resolve centroid and envelope oscillations around the ring.



(a) (b)

Figure 2: (a) A Diagram of the UMER accelerator. (b) (left) A high resolution simulation image. (right) A downsampled image used for predictions.

In some accelerators, only the first turn can be visualized with a phosphor screen. For that reason, we use only the first turn for training. A 4000-node reservoir in MATLAB is trained on the first 288 images, which is the full first turn around the ring, and generates predicted images with equal spacing in the next two turns, as shown in Figure 3. To prevent overfitting, 2% noise is added to training data. Results are generally observed to degrade with fewer nodes and not to improve with more nodes. The value of each pixel is presented to a subset of nodes in the reservoir determined by randomly initialized matrix W_{in}. For each image in the series, the output of the reservoir is trained to match the subsequent image to learn the dynamics of the particle beam. The reservoir is then allowed to run freely for prediction. The output layer is trained using least squares optimization.

The reservoir is sensitive to a variety of hyperparameters, including input strength (the magnitude of values in W_{in}), spectral radius (the size of the largest eigenvalue of A), α (the memory parameter), regularization parameter (a parameter that determines the amount of error allowed during training), reservoir size, and the particular choice of adjacency matrix A. Reservoir size must be large in

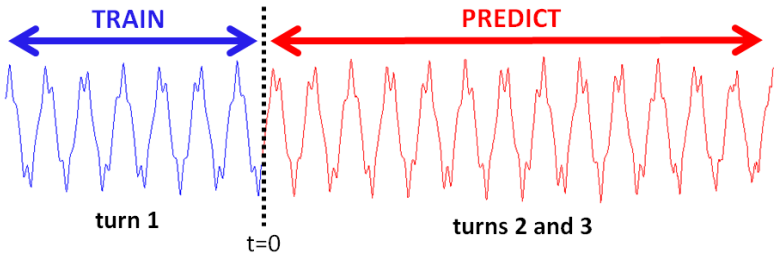


Figure 3: Training is done on the first turn, and the subsequent two turns are predicted.

comparison to dimensionality of input data, but there is no known way to optimize A other than through trial and error. After A is fixed, the other hyperparameters are manually tuned until a satisfactory result is obtained. The reservoir is gemerated and the result is computed in under a minute on a PC.

4 Results

To flatten data from the image series, four parameters are extracted from the predicted images: the centroid motion and beam size in both vertical and horizontal planes. These are the first and second central moments of the images, $(\mu_{01}, \mu_{10}, \mu_{20}, \mu_{02})$, and are calculated using the following equation:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y) \tag{5}$$

where $I(x, y)$ is the pixel intensity, x, y are the pixel locations, and p, q are the 2D moment orders.

Centroid motion evolves in a semi-periodic fashion dictated by the magnetic focusing lattice. The beam envelope, which evolves according to equations (3,4), is subject to substantial nonlinearities, and is aperiodic. Cross moments such as μ_{11} can appear if there are rotational offsets in magnets, which cause x and y motion to couple, but for these initial studies there is no skew in the magnets.

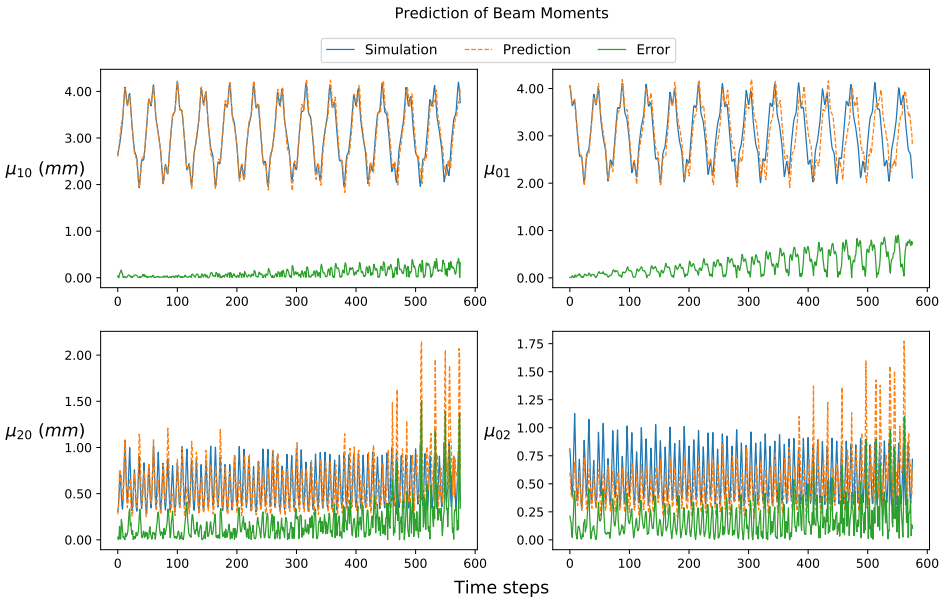


Figure 4: Plots of four beam parameters: horizontal and vertical centroid position (μ_{10} , μ_{01}) and beam size (μ_{20} , μ_{02}), from simulation and from reservoir prediction. The error is measured as the absolute difference between the simulation and prediction values.

Error	Turn 1	Turn 2
μ_{10}	1.6%	5.0%
μ_{01}	5.4%	14.3%
μ_{20}	21.1%	37.7%
μ_{02}	37.5%	45.0%

Table 1: Average errors calculated between moments from simulations and moments from reservoir predictions, in the first and second predicted turns.

The results, summarized in Figure 4, show good agreement between prediction and experiment, particularly in tracking the relatively regular centroid motion. The beam envelope is subject to more nonlinear forces, and the reservoir's error is relatively higher, yet the general character of the time series is captured, with the prediction closely matching the periodicity of the data from simulation. Error percentages for each turn are summarized in table 1. These promising initial studies using simulated data will be followed with physical experiments using UMER. We plan to use measured data as the initial conditions in a reservoir computer trained with simulated data and compare the reservoir computer's forecast to real measurements at available beam imaging locations.

5 Conclusion

Reservoir computing is a machine learning technique that enables the prediction of behavior of dynamical systems, such as the transverse profile of a particle beam inside an accelerator. We have shown promising initial results for prediction of transverse electron beam parameters using a reservoir computer in MATLAB, allowing beam properties to be inferred in locations where they cannot be physically measured. Time series forecasting using reservoir computing could be utilized in active control scenarios, in particle accelerators or other systems. Experimental tests on the University of Maryland Electron Ring are planned.

6 Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1840340. Work is also supported by Department of Energy - High Energy Physics award No. DE-SC0010301.

References

[1] A. Edelen, et al., "Opportunities in Machine Learning for Particle Accelerators.," arXiv:1811.03172[physics], Nov. 2018.

[2] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an Erratum note 1.," Bonn, Germany, 2010.

[3] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: a new framework for neural computation based on perturbations.," Neural Comput., vol. 14, no. 11, pp. 2531–60, 2002.

[4] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach.," Phys. Rev. Lett., vol. 120, no. 2, pp. 024102, 2018.

[5] R.A. Kishek et al., "The University of Maryland Electron Ring Program.," Nucl. Instrum. Methods Phys. Res., vol 733, pp. 233-237, 2014.

[6] L. Dovlatyan, T.M. Antonsen, B.L. Beaudoin, I. Haber, D.B. Matthew, and K.J. Ruisard, "Preliminary Lattice Studies for the Single-Invariant Optics Experiment at the University of Maryland", in Proc. IPAC'19, Melbourne, Australia, May 2019, pp. 367-370. 2019.

[7] Reiser, M., OShea, P. (2008). Theory and design of charged particle beams. Weinheim: Wiley-VCH.

[8] A. Friedman et al., "Computational Methods in the Warp Code Framework for Kinetic Simulations of Particle Beams and Plasmas.," IEEE Trans. Plasma Sci., vol 42, pp. 01321, 2014.