Machine-learning accelerated particle in-cell simulations

R. Kube, R.M. Churchill, B. Sturdevant, CS Chang
Princeton Plasma Physics Laboratory, 100 Stellarator Rd, Princeton NJ 08540 USA

PPPL PRINCETON PLASMA PHYSICS LABORATORY

Motivation

Particle-in-cell (PIC) codes are one of the workhorses for numerically exploring plasma dynamics across a vast parameter space. Charge and energy-conserving implicit PIC formulations have been developed and opened the path for accurate and efficient kinetic plasma simulations [1]. By discretizing the PIC equations in an implicit manner, they bypass stability constraints that limit the grid-size or time-step size in explicit formulations. This comes at the cost of increased computational cost. For each time step, a demanding system of non-linear equations needs to be solved iteratively. Here we explore how machine learning can be used to accelerate the convergence of the Jacobian-free Newton Krylov solver that is often used in such PIC simulations.

Implicit PIC formulation

We are considering a collisionless, unmagnetized and fully ionized plasma consisting of electrons and ions, p in {e,i} in a 1-dimensional domain z in [0;L_z], descretized on N_z equally sized cells.

$$\frac{x_{\mathrm{p}}^{n+1} - x_{\mathrm{p}}^{n}}{\triangle t} = v_{\mathrm{p}}^{n+1/2}$$

$$\frac{v_{\mathrm{p}}^{n+1} - v_{\mathrm{p}}^{n}}{\triangle t} = \frac{q_{\mathrm{p}}}{m_{\mathrm{p}}} \mathrm{SM}\left[\boldsymbol{E}^{n+1/2}\right]\left(x_{\mathrm{p}}^{n+1/2}\right)$$

$$\epsilon_0 \frac{E_i^{n+1} - E_i^{n}}{\triangle t} + \frac{q_{\mathrm{p}}}{m_{\mathrm{p}}} \mathrm{SM}\left[\bar{j}\right]_i^{n+1/2} = \langle \bar{j} \rangle^{n+1/2}$$

An implicit Crank-Nicolson formulation is chosen for the time evolution. Recognizing that x_p,v_p = x_p(E) and v_p(E), the residuals of the equations of motions are formulated as

$$\boldsymbol{G}\left(\boldsymbol{E}, f\left(\boldsymbol{E}\right)\right) = 0$$

The non-linear PIC residual can in principle be solved using Newton's method. In each iteration k this linear system of type Ax=b has to be solved. Can this solve be accelerate using ML?

$$\left.\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{E}}\right|^{k} \delta \boldsymbol{E}^{k} = -\boldsymbol{G}\left(\boldsymbol{E}^{k}\right)$$

[1] G. Chen, L. Chacon, D.C. Barnes Journ. Comp. Phys 230 7018-7036 (2011)

Machine learning approaches

Data-driven GMRES[2] augments GMRES' Krylov space with a set of n_{aug} vectors. For this, one projects A and b out of span(v_1,...,v_{naug}) and solves the remaining problem with standard GMRES.

$$x_i = \underset{x \in \mathcal{V} + \mathcal{K}_i(\widetilde{A}, \widetilde{b})}{\operatorname{argmin}} ||Ax - b||^2$$

Use that $\boldsymbol{x} = \sum_{j=1}^{n_{\mathrm{aug}}} \alpha_j \boldsymbol{v}_j + \widetilde{\boldsymbol{x}} \quad \widetilde{\boldsymbol{x}} \perp \mathrm{span}\left(\mathcal{V}\right)$ and insert into Ax=b to solve for alpha

$$\begin{bmatrix} \langle A\boldsymbol{v}_1, A\boldsymbol{v}_1\rangle & \langle A\boldsymbol{v}_1, A\boldsymbol{v}_2\rangle & \dots & \langle A\boldsymbol{v}_1, A\boldsymbol{v}_{n_{\mathrm{aug}}}\rangle \\ \langle A\boldsymbol{v}_2, A\boldsymbol{v}_1\rangle & \langle A\boldsymbol{v}_2, A\boldsymbol{v}_2\rangle & \dots & \langle A\boldsymbol{v}_2, A\boldsymbol{v}_{n_{\mathrm{aug}}}\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle A\boldsymbol{v}_{n_{\mathrm{aug}}}, A\boldsymbol{v}_1\rangle & \langle A\boldsymbol{v}_{n_{\mathrm{aug}}}, A\boldsymbol{v}_2\rangle & \dots & \langle A\boldsymbol{v}_{n_{\mathrm{aug}}}, A\boldsymbol{v}_{n_{\mathrm{aug}}}\rangle \end{bmatrix} * \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n_{\mathrm{aug}}} \end{bmatrix} = \begin{bmatrix} \langle A\boldsymbol{v}_1, b\rangle \\ \langle A\boldsymbol{v}_2, b\rangle \\ \vdots \\ \langle A\boldsymbol{v}_{n_{\mathrm{aug}}}, b\rangle \end{bmatrix}$$

Finally, solve $A\widetilde{x} = b - \sum_{j=1}^{n_{\mathrm{aug}}} \alpha_j A v_j$ using GMRES and reconstruct **x**.

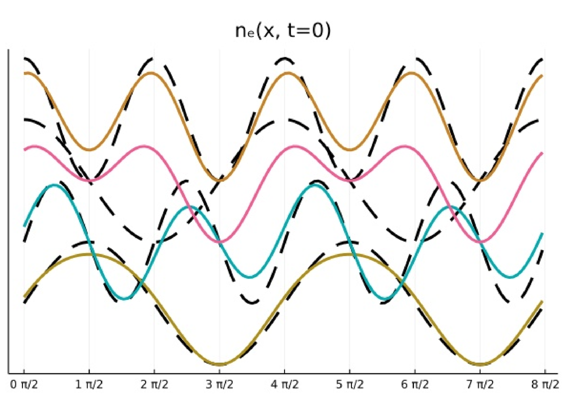
Determining v_i's to accelerate first Newton iteration (k=1) using by training neural network to map {n_e(x), E(x), IC} -> {v_1, ..., v_N} by minimizing:

$$\mathcal{L}_{\mathrm{proj}} = \min_{x \in \mathcal{V}^k} \frac{\|x - x^{(m)}\|_2}{\|x^{(m)}\|_2} + \lambda \left|\mathrm{triu}\left(VV^{\dagger}, 1\right)\right|$$

where $x^{(m)}$ is the converged solution of a simulation time-step. Here m=1...M indices the solutions $\delta \boldsymbol{E}^{k=1}$ for a set of simulations.

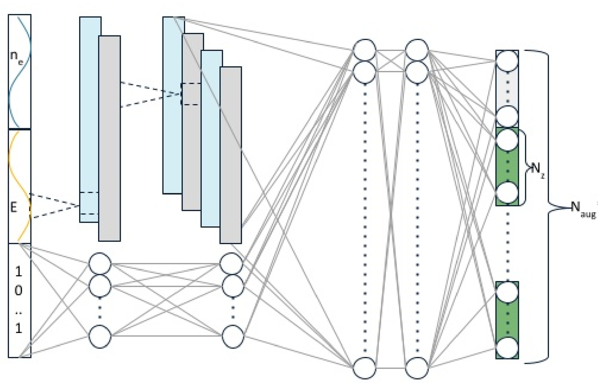
[2] R. Trivedi et al. Nature Scientific Reports 9, 19728 (2019)

Training and inference

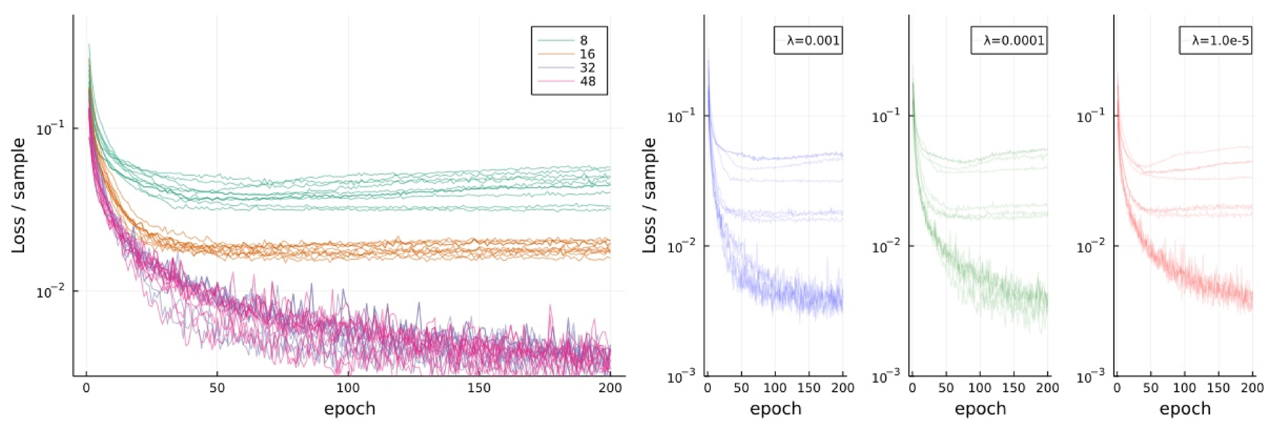


Training data $\delta \boldsymbol{E}^{k=1}$ is taken from simulations with dt=1.0, T_{end}=30.0 and n_e(z,t=0) = A f(kz) where A={1,2,5,10}*10^{-2}, f=sin, cos, k=1,2

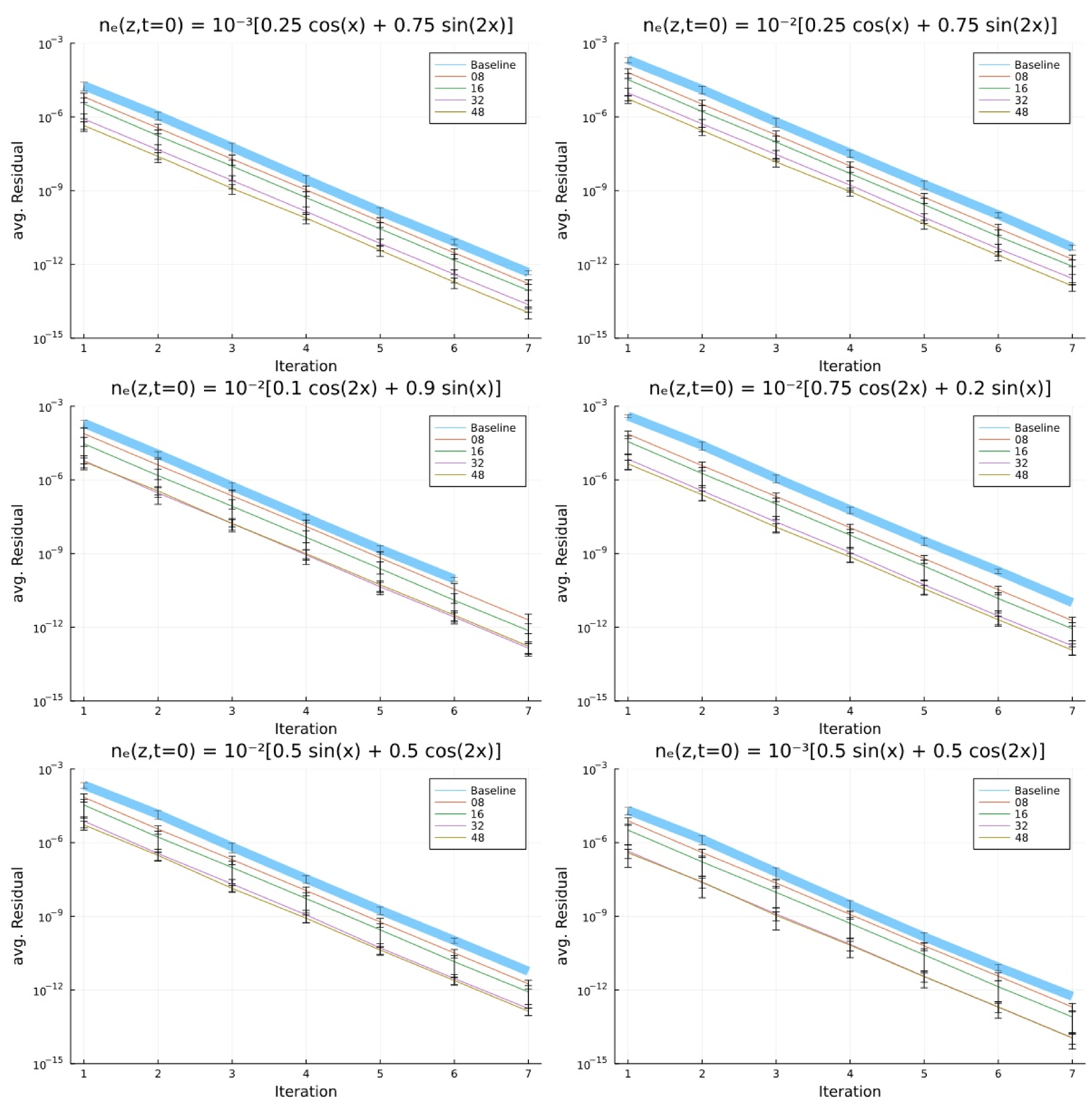
Inference is performed on simulations with initial profiles given by colored lines.



Profile data are fed forward through 2 convolutional layers
One-hot encoding of simulation inital conditions is fed through MLP
Final output layer are v_i's of size N_z * n_{aug}



Most effective method to minimize the loss is to increase n_{aug}.
Orthogonality regularization is most effective for n_{aug}=32, 48.
Only little sensitivity to other hyperparameters such as activation function, number and shape of layers etc.



Number of GMRES iterations required to reduce residual below same threshold reduced by 1-2
This is robustely produced in simulations with unseen initial profiles
Large reduction for n_{aug}=8,16,32 and increasing n_{aug} over 32 has little effect

Conclusion and future work

We explored how to a linear iterative solve used for implicit PIC simulations can be accelerated using machine learning. For this we trained a neural network to augment the Krylov space with a set of vectors which approximate the solution. The network is trained on a set of 20 simulations and inference was run over 6 simulations. The augmented GMRES solve consistently requires about 1-2 iterations less to converge to the same residual as the baseline case. Future work will address how to modify GMRES so that the expensive projection steps can be skipped [3].

[3] R.B Morgan, J Matrix Anal. Appl. 16, 4, 1154 (1995)