Neural networks as effective surrogate models of radio-frequency quadrupole particle accelerator simulations

Joshua Villarreal, Daniel Winklehner, Daniel Koser and Janet M Conrad
Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, United States of America
* Author to whom any correspondence should be addressed.
E-mail: villaj@mit.edu

Keywords: accelerator, particle, radio-frequency quadrupole, surrogate modeling, virtual accelerator

Abstract
Radio-frequency quadrupoles (RFQs) are multi-purpose linear particle accelerators that simultaneously bunch and accelerate charged particle beams. They are ubiquitous in accelerator physics, especially as injectors to higher-energy machines, owing to their impressive efficiency. The design and optimization of these devices can be lengthy due to the need to repeatedly perform high-fidelity simulations. Several recent papers have demonstrated that machine learning can be used to build surrogate models (fast-executing replacements of computationally costly beam simulations) for order-of-magnitude computing time speedups. However, while these pilot studies are encouraging, there is room to improve their predictive accuracy. Particularly, beam summary statistics such as emittances (an important figure of merit in particle accelerator physics) have historically been challenging to predict. For the first time, we present a surrogate model trained on 200,000 samples that yields <6% mean average percent error for the predictions of all relevant beam output parameters from defining RFQ design parameters, solving the problem of poor emittance predictions by identifying and including hidden variables which were not accounted for previously. These surrogate models were made possible by using the Julia language and GPU computing; we briefly discuss both. We demonstrate the utility of surrogate modeling by performing a multi-objective optimization using our best model as a callback in the objective function to select an optimal RFQ design. We consider trade-offs in RFQ performance for various choices of Pareto-optimal design variables—common issues for any multi-objective optimization scheme. Lastly, we make recommendations for input data preparation, selection, and neural network architectures that pave the way for future development of production-capable surrogate models for RFQs and other particle accelerators.

1. Introduction

The replacement of highly accurate, but computationally costly, particle-in-cell (PIC) simulations with surrogate models (sometimes called virtual accelerators) is a topical field of increased interest [1–6]. Surrogate models use machine learning (ML) to create fast-executing virtual representations of a complex system like a particle accelerator. We can then use this surrogate model to, for instance, speed up (multi-objective) design optimization, or obtain real-time feedback during the commissioning, tuning, and running of the particle accelerator. The surrogate model is typically built from a neural network (NN) or some other statistical learning technique (like polynomial chaos expansion [2]). In the case of using the surrogate model as an autonomous tuning tool, training data can be obtained not just from simulations, but also by measurements from existing hardware [7].

The design of the isotope decay-at-rest (IsoDAR) project [8–10], a planned experiment in neutrino physics, is the primary motivation for this work. In IsoDAR, a compact particle accelerator produces a 10 mA proton beam that impinges with an energy of 60 MeV amu⁻¹ on a beryllium target surrounded by lithium-7, producing electron antineutrinos (\(\bar{\nu}_e\)) with a well-understood energy distribution through IsoDAR [10] (as...
opposed to other experiments using decay-in-flight). The $\bar{\nu}_e$ can then be measured in a nearby liquid scintillator detector (LSC) via inverse beta decay (IBD). This configuration yields unprecedented sensitivity to so-called sterile neutrinos, hypothesized new particles thought to resolve $\bar{\nu}_e$ deficits observed at experiments worldwide [11–14]. The requirements for IsoDAR are 10 mA of protons on target in a continuous wave beam at 80% duty factor to produce about $1.15 \times 10^{23} \bar{\nu}_e$ over the course of 5 years. Paired with the planned 2.3 kiloton LSC [15] in Korea, this will yield 1.67 million IBD events in the detector.

The IsoDAR particle accelerator comprises an ion source, radio-frequency quadrupole (RFQ), and a cyclotron [16, 17]. Surrogate modeling has proven invaluable to IsoDAR’s development, allowing us to demonstrate the robustness of IsoDAR’s cyclotron design [17] through uncertainty quantification [2], and to perform a small pilot study to investigate the use of surrogate models for RFQs [18].

In this paper, we expand upon work presented in [18] to build a NN-based surrogate model of an RFQ, but rethink the RFQ parametrization to account for collinear effects in the feature space, physical RFQ design constraints, and incorporate variables previously hidden from trained surrogate models. We use these insights to generate an accurate surrogate model for a 32.8 MHz RFQ covering a wide design parameter space (subject to physical design constraints). We use the highly efficient Julia programming language [19] to train our NNs with a widely cast net for hyperparameter tuning and an unusually large batch size.

The working principle of an RFQ and the generation of training data for the surrogate model were discussed in detail in [18]. We briefly summarize them in section 1.1, and section 2.1, respectively. In section 2, we discuss our methods, including data preparation and how we enhance the predictive accuracy of beam summary statistics outputs like the beam emittance (an important figure of merit for beam quality), followed by our results for training of several NNs and optimization of the RFQ in section 3. Finally, we discuss our results, general observations, and recommendations for future development of NN-powered surrogate models in sections 4 and 5. Code relevant to this project is available on GitHub [20].

1.1. The RFQ

An RFQ is a multi-purpose linear particle accelerator element able to bunch (compress along the direction of movement) and accelerate a high-current ion beam, while keeping the beam tightly focused in the transverse direction [21–23]. In an RFQ, an oscillating electric field is generated between four vanes (or rods). The arrangement for one cell can be seen in figure 1, where the length is $\beta \cdot \lambda/2$, with $\beta$ the ratio of the beam velocity to the speed of light, and $\lambda$ the wavelength corresponding to the electromagnetic wave driving the RFQ. Typical RFQs comprise tens to hundreds of such cells, each defined by three main parameters (the focusing strength $B$, the phase $\Phi$, and the modulation $m$). The IsoDAR RFQ is discussed in detail elsewhere [24, 25]. To optimize an RFQ, the parameters of each cell have to be fine-tuned to yield desirable beam output qualities like a high beam transmission. In this paper, we call the RFQ parameters design variables (DVARs) and the beam output parameters objectives (OBJs).

1.2. The Julia language

Julia is an open-source dynamically typed programming language with significant performance improvements over other languages common in scientific computing like Python and Matlab [19]. Julia’s computational efficiency is especially apparent when performing costly calculations like NN training, motivating our use of the language throughout this analysis and allowing for straightforward implementation of multi-threading and distributed computing to facilitate the completion of expensive computational tasks.

![Figure 1. A single RFQ cell with electric fields in quasi-static approximation. Left: front view (beam and z-axis point into the paper). The green arrows indicate the focusing of the charged particle beam and the red arrows the defocusing. Right: side view (beam propagates from left to right). Cell geometry parameters are shown, and the electric field is indicated by arrows. From [18].](image-url)
In addition, Julia has built-in support for GPU programming using CUDA.\(^1\)\(^2\)[26, 27], making the language an obvious choice for building and training series of NNs on both local (CPU) and remote (GPU) machines.

2. Methodology

2.1. Data generation

In this study, we reuse the dataset from [18], consisting of 217 292 samples, each representing a randomly configured RFQ with corresponding output beam parameters obtained from beam dynamics simulations using the well-established code PARMTEQM [28]. The beam dynamics characteristics of an RFQ are fully described by a set of three main cell parameters \((B_n, \Phi_n, m_n)\) for each RFQ cell \(n\), resulting in a total of \(3n\) DV ARs. While historically the LANL four-section procedure (FSP) has been used as a design strategy since the very first proof-of-principle RFQs at LANL at the end of the 1970s [22], over the years, modified approaches have evolved to improve RFQ performance for specific applications. In the FSP, the RFQ is divided into four sections with dedicated functions: a radial matcher, a shaper, a gentle buncher and an accelerator, with the overall aim of smoothly matching the beam to the structure, bunching it, and accelerating it with as little losses as possible.

As a first approach for the IsoDAR RFQ, a baseline design was created via an adaptation of the FSP framework, but with special characteristics: (1) in addition to the typical linear increase of the synchronous phase \(\Phi\) and the modulation \(m\) in the shaper (called the ‘linear shaper’), a section of exponential increase is introduced (called the ‘exponential shaper’); (2) in the gentle buncher, not only the modulation but also the synchronous phase is ramped up exponentially (called the ‘exponential buncher’); and (3) the accelerator section is omitted due to the IsoDAR RFQ being intended for use as a dedicated pre-buncher.

For a ML based design optimization, a reduction of the number of the initially \(3n\) DV ARs was pursued and thus a parametrization according to figure 2 was performed. This allowed to capture all crucial characteristics of the design functions while reducing the number of DV ARs to 14. While DVAR1 corresponds to the value of the (constant) focusing strength and DVAR2 and DVAR3 set the lengths of the linear and exponential shaper, the total slope and smoothness of the shaping/bunching effect are characterized by DVAR4–DVAR13. DVAR14 specifies the design energy at the RFQ exit, which determines the length of the RFQ.

Besides the 14 DV ARs, the dataset from [18] contains 6 OBJs: beam transmission, output energy, RFQ length, and three beam emittances (one longitudinal, two transverse to the beamline). This data is summarized in table 1.

2.2. Data preprocessing

Unlike [18], we perform transformations on the data set, as some variables have values that directly affect values of others. In particular, for some upper bounds \(u_i\) and buffers \(\delta_i\), \(i \in \{3, 5, 9, 12, 13\}\), after a subset of the 14 DV ARs for the \(j\)th sample are drawn from their respective uniform distributions, the rest are acquired via:

\[
\begin{align*}
\text{DVAR3}_j & \sim \text{Uniform}(\text{DVAR2}_j + \delta_3, u_3) \\
\text{DVAR5}_j & \sim \text{Uniform}(\text{DVAR4}_j + \delta_5, u_5) \\
\text{DVAR9}_j & \sim \text{Uniform}(\text{DVAR8}_j + \delta_9, u_9) \\
\text{DVAR12}_j & \sim \text{Uniform}(\text{DVAR5}_j + \delta_{12}, u_{12}) \\
\text{DVAR13}_j & \sim \text{Uniform}(\text{DVAR9}_j + \delta_{13}, u_{13})
\end{align*}
\]

with:

- \(\delta_3\) the minimum allowable length of the exponential shaper,
- \(\delta_5\) the minimum allowable modulation increase in the exponential shaper,
- \(\delta_9\) the minimum allowable phase increase in the exponential shaper,
- \(\delta_{12}\) the minimum allowable modulation increase in the exponential buncher, and
- \(\delta_{13}\) the minimum allowable phase increase in the exponential buncher.

Values of the \(\delta_i\) and \(u_i\) can be inferred from table 1. Having these buffers ensures that physically realistic RFQ sections are created. For instance, if DVAR3 \(\sim \text{Uniform}(\text{DVAR2}_j, u_3)\), it is possible that a randomly generated RFQ would have an exponential shaper component of zero length, which is nonphysical for the family of accelerators considered in this study.
Figure 2. The three main cell parameters (focusing function $B$, phase $\Phi$, and modulation $m$), parametrized along the length of the RFQ by fourteen DV ARs (DV AR14, not shown, is the design energy determining the overall length). From [18].

Table 1. Summary of design variables (DV ARs) and objectives (OBJs) used for surrogate model training. *In uniformly drawing random RFQ design variables, certain relationships between these parameters must be satisfied to ensure the RFQ is physical. This is further discussed in section 2.2.

| Label | Variable | Lower bound | Upper bound | Physical meaning |
|-------|----------|-------------|-------------|------------------|
| DV AR1 | $B_{\text{max}}$ [1] | 8.5 | 12.0 | (Constant) focusing strength |
| DV AR2 | $m_{X1}$ [cm] | 5 | 140 | End position of linear shaper |
| DV AR3 | $m_{X2}$ [cm] | $DVAR2 + 10$ | 160 | End position of exponential shaper |
| DV AR4 | $m_{Y1}$ [1] | 1.005 | 1.7 | Modulation at end of linear shaper |
| DV AR5 | $m_{Y2}$ [1] | $DVAR4 + 0.05$ | 1.85 | Modulation at end of exponential shaper |
| DV AR6 | $m_{\tau 1}$ [cm] | 1 | 500 | Exp. param. for modulation in exp. shaper |
| DV AR7 | $m_{\tau 2}$ [cm] | 1 | 500 | Exp. param. for modulation in exp. buncher |
| DV AR8 | $\Phi_{Y1}$ [deg] | $-89.95$ | $-30$ | Synchronous phase at end of lin. shaper |
| DV AR9 | $\Phi_{Y2}$ [deg] | $DVAR8 + 2.5$ | $-25$ | Synchronous phase at end of exp. shaper |
| DV AR10 | $\Phi_{\tau 1}$ [cm] | 1 | 500 | Exp. param. for sync. phase in exp. shaper |
| DV AR11 | $\Phi_{\tau 2}$ [cm] | 1 | 500 | Exp. param. for sync. phase in exp. buncher |
| DV AR12 | $m_{\text{Y3 ref}}$ [1] | 1 | 500 | Modulation at end of exp. buncher |
| DV AR13 | $\Phi_{\text{Y3 ref}}$ [deg] | $DVAR9 + 2.5$ | $-20$ | Synchronous phase at end of exp. buncher |
| DV AR14 | $E_{\text{ref}}$ [MeV] | 0.055 | 0.075 | Design energy (at exit) |

| OBJ1 | Transmission [%] |
| OBJ2 | Output energy [MeV] ($E_{\text{out}}$) |
| OBJ3 | RFQ length [cm] |
| OBJ4 | Longitudinal emittance [MeV deg] ($\epsilon_{\text{long}}$) |
| OBJ5 | $x$-emittance [cm mrad] ($\epsilon_x$) |
| OBJ6 | $y$-emittance [cm mrad] ($\epsilon_y$) |

As a result, DV ARs like DV AR3 are not uniformly distributed. We can introduce a variable $DVAR3'$ that is uniformly distributed according to the following transformation of DV AR2, DV AR3, and constants:

$$DVAR3' = \frac{DVAR3 - (DVAR2 + \delta_3)}{\delta_3}.$$

And likewise for the remaining pairs of features. A side-by-side comparison of distributions of DV AR3 and the transformed DV AR3' is shown in figure A1 in appendix A.

Correlation matrices for the 14 DV ARs before and after the described pre-processing transformation of necessary DV ARs are shown in figure 3. We call these transformed DV ARs decorrelated, since all correlations have been removed. Each feature (after the decorrelating step) was later scaled to have minimum $-1$ and maximum $+1$.

To assess the impact of decorrelating data on NN training and performance, we trained two identical NNs on both correlated and decorrelated scaled datasets. We observed no significant difference in the training times or prediction accuracy between these two NNs; this is to be expected, NNs are, in general, over-parameterized so decorrelation of features is not generally a necessary step. Eventually, however, the top-performing surrogate model in this study will serve as an important component of a design optimization. It is easier in a prebuilt Bayesian optimization package to enforce the relationships between these DV ARs listed in equations (1) by working with decorrelated features; we can perform a multi-OBJ design optimization under no additional constraints (besides the constraint that the scaled, decorrelated features should have minima $-1$ and maxima $1$). All NNs we present in the remainder of this study are
Figure 3. Correlation matrices for raw and decorrelated feature data (DVARs).

Figure 4. Overlaid distributions of the difference in transverse emittances for samples having odd and even cell numbers. The clear separation of these distributions indicates the inclusion of the number of cells is vital to the prediction of transverse emittance.

trained on the uncorrelated data, and we recommend decorrelating the DVARs as a best practice moving forward, particularly when built surrogate models will be plugged into any kind of optimization scheme.

2.3. Number of RFQ cells as a previously hidden variable
The models reported in [18] and early NNs built in this study have one thing in common: predictions of transverse emittances (OBJ5 and OBJ6) were significantly less accurate than those of the other OBJ variables. In [7, 17] the authors also observe this for a cyclotron and the LCLS-II injector beam line, making the failure to accurately predict RMS emittance a problem worth exploring. In appendix B, we show that for surrogate models similar to those presented in [18], patterns in the true and surrogate-model-predicted emittances indicate some underlying hidden variable should be included to resolve the asymmetry between the two transverse directions.

To this end, we extend the dataset from [18] to include a fifteenth feature used for training the surrogate models produced in this work; namely, whether the number of RFQ cells is even or odd. As discussed in section 1.1, across any particular RFQ cell, the beam is simultaneously focused in one of the two transverse directions while being defocused in the other. The number of RFQ cells (specifically whether the number of RFQ cells is even or odd, which we denote as the ‘parity’ of the RFQ in this work) is essential in resolving any asymmetries in the beam characteristics across the two transverse directions, such as the two transverse emittances $\epsilon_x$ and $\epsilon_y$. Histograms of $\epsilon_x - \epsilon_y$ for simulated RFQs with odd and even numbers of cells are shown in figure 4, further elucidating this concept. We added an extra column to the features: a binary variable denoting whether the total number of RFQ cells is odd or even.

This ‘15th DVAR’ can be obtained from the other 14 DVARs in a fully deterministic way, and this calculation becomes another preprocessing step of our data. The algorithm that we devised in Python is based on the work of Kapchinskii and Teplyakov [21] as well as Crandall and Wangler [23, 29–32], calculating an ideal particle’s energy gain for each RFQ cell, based on the parametrization of cell parameters shown in figure 2. Each preceding cell thus determines the position along the z-axis at which to evaluate the
parametrization curves for the next cell. We tested the algorithm on all data in the training and validation sets and obtained a >83% success rate in predicting even or odd number of cells (cf figure 5). The code for this routine is available in the compute_cellnumber/compute_cellnumber.py file on our GitHub [20]. The ≈17% miss rate can be attributed to the fact that our Python script uses approximations for each full cell, while Pari, an RFQ design code used in sequence with PARMTEQM, integrates over the cell numerically by slicing it. Much slower, but equally legitimate way of calculating DVAR15 during preprocessing would be to perform a system call to Pari.exe and to read the corresponding output file. This would predict the parity of the cell number with 100% success. This is how the training set was generated. For the optimizer, we chose the fast Python-based algorithm as it is not restricted to Windows platforms and not export-controlled in the USA (i.e. only allowed to be used without restrictions within the USA).

It is important to note that the inclusion of this additional DVAR does not impact the integrity of the surrogate model, nor does it significantly increase our surrogate model’s computational overhead. The algorithm is fast-executing and the 15th DVAR follows from the other 14 DVARs fully deterministically. All NNs are trained on correct parity information; the approximation algorithm is only used during the optimization phase. During the optimization process, the use of the Python algorithm, which is incorrect ≈17% of the time, may lead to slightly longer time-to-solution and a larger error associated with the emittance results, however, this error is comparable with the NN MAPEs for the emittance in general. Furthermore, each point in the Pareto-front is ultimately calculated by a call to the surrogate model. We observed empirically that artificially swapping the parity from +1 to −1 has the effect of (approximately) swapping the predictions of $\epsilon_x$ and $\epsilon_y$ (i.e. moving from one branch to the other in figure B1), while keeping predictions on the other 4 OBJs constant. Since the distribution of even and odd in the incorrectly calculated parities is about equal, the Pareto front will look approximately the same. Finally, the chosen Pareto-optimal point after optimization will have to be corroborated by the original simulation software, where incorrect parity predictions will be evident. The designer can modify the machine accordingly. Even with a 17% miss rate in predicting the parity of an RFQ, we still achieve substantial performance improvements over previous efforts to predict emittances.

2.4. Surrogate model reporting
NN performance is evaluated by computing $R^2$ scores and mean absolute percent errors (MAPEs). We cite the ‘aggregate $R^2$’ score, the average of the $R^2$ scores for each of the six response variables, to report the total accuracy of the model. This does not, however, give any insight into predictive accuracies of each of the six OBJs separately: we use MAPEs on the unscaled data to handle this task.

2.5. Training on GPUs
As part of our efforts to fully explore the Julia language, and to exploit the significant speedup coming from massive parallelization of matrix algebra, we migrated NN training to GPUs. Here we used two NVIDIA A30 Tensor Core GPU cards with 24 GB of VRAM each, combined with Intel Xeon(R) Silver 4310 12-core CPUs

1 Feeding NNs the raw number of cells, rather than the binary parity variable, experimentally proved difficult for the NNs to use when determining the sign of $\epsilon_x - \epsilon_y$. We suspect that this is due to the fact that the number of cells in the virtual RFQs in this study is $\sim O(100)$, so performing the uniform-minimum-and-maximum transformation degrades the ability of the network to resolve whether the number of cells was originally odd or even.
### Table 2. Values chosen for initial hyperparameter scan in Julia.

| Hyperparameter       | Value(s) scanned               |
|----------------------|--------------------------------|
| NN depth             | 4, 5, 6                        |
| NN width             | 50, 75, 100                    |
| Batch size           | 1024                           |
| Learning rate        | 0.001                          |
| Activation function  | Sigmoid                        |
| Loss function        | Mean squared error             |

(we assigned one thread per GPU) on the MIT ‘SubMIT’ cluster [33] for training ‘RFQNet1’ and ‘RFQNet2’ (see below). We used CUDA version 11.6. While we do not present a GPU benchmark, we can point out that the speedup over CPU training was significant.

### 2.6. Implementation of NN training and hyperparameter scans

Operating on the full decorrelated dataset, we use Julia to implement grid search scans over NN architecture hyperparameters; namely, width and depth. We hold batch size, learning rate, and activation functions constant for this study but plan to expand future searches to these hyperparameters, as well. All NNs were trained using mean squared error as the loss function. Scanned NN hyperparameters are summarized in Table 2. NN training was performed primarily using the Flux.jl package, an open-source Julia library used for training deep ML models [34, 35]. NNs of different architectures were trained in parallel to reduce script runtime. Results from [18] indicate that NNs trained with the largest batch sizes were desirable. While larger batch sizes may be preferred in principle to show NNs more data per training step, they ultimately increase total training time [36]. The use of Julia allows us to feasibly consider training NNs with larger batch sizes in a more reasonable amount of time. Hence, we choose to hold batch size constant at 1024, outside of the range of that explored by [18]. While it is difficult to interpret why the surrogate models developed in this study seemed to prefer larger batch sizes, we suspect that because we are sampling the 14-dimensional feature space uniformly, each training step sees a more representative sample of the training data, allowing our networks to generalize better. Also, to fully leverage the parallelization capabilities of GPUs used for training these NNs, larger batch sizes are computationally preferred.

Data was split into training and test sets (of proportions 80% and 20%, respectively). The test set was withheld from any analysis for the entirety of the scan, and was only used for computing the final test-set MAPEs shown in Table 3. We used the MLUtils.jl package ([37]) to perform 5-fold cross-validation, wherein each NN was trained on an 80% subsample of the training data, with the remaining 20% used as a validation dataset to estimate out-of-sample model performance. NNs were not trained on this smaller subset of data. The validation sets used in each fold of the cross-validation procedure were disjoint.

Each NN was constructed using a sigmoid activation function. The Flux.jl ADAM ([38]) optimizer was used to perform stochastic gradient descent in NN training, using the default learning rate of 0.1% unless otherwise specified. All NNs were trained for 2500 epochs, which appeared experimentally to be a reasonable threshold for convergence (one in which improvements in predictive accuracy began to plateau while preserving agreement between the validation and training set prediction errors). The results of this hyperparameter scan are reported in section 3.1, and the best performing of these networks is referred to as ‘RFQNet1’. This network is summarized in Table 4.

In prototyping of the surrogate models developed in this study, we also added dropout regularization to trained NNs and tested different combinations of dropout rates and learning rates. The inclusion of dropout regularization, in general, negatively impacted the validation-set predictive accuracy of the networks in the worst case, while different OBJs preferred different combinations of dropout and learning rates, in the best case. None of the tested networks outperformed RFQNet1 and RFQNet2 presented in this work.

### 2.7. Additional design-motivated data preprocessing

The RFQ dataset used both in this study as well as in [18] was generated by performing basic random sampling of the 14 design parameters within their respective specified value ranges, and determining the OBJ variables by simulating beam traversing the RFQ using a physics engine. However, generating RFQs in this way can result in RFQs that are especially undesirable and even unrealistic. Many of the virtual RFQs were determined to have an alarmingly low beam transmission of 60% or less. In the case that an accelerator engineer would like to use a surrogate model to optimize their RFQ design, it is reasonable to assume their RFQ’s beam transmission will be much higher. We therefore perform the same hyperparameter scan outlined in section 2.6 on data culled to have simulated beam transmissions of at least 60%. The best-performing NN
of this hyperparameter scan is referred to as ‘RFQNet2’. RFQNet2’s structure is compared to that of RFQNet1 in table 4.

Due to RFQs being complex machines with many cells, and parameters like transmission depending on the interplay between the DV ARs rather than being smooth monotonically changing functions, it is not possible to determine ab-initio which DV ARs need to be restricted (and where, as there are many local minima) to limit transmission to >60%. Thus, we cut based on the OBJ rather than the DV ARs. Chosing a value of 60%, which is still far away from the desired transmissions of RFQs, gives the surrogate model enough margin for us to be able to perform optimizations.

2.8. RFQ design optimization

To demonstrate the utility of such ML-powered surrogate models, we used RFQNet2 as the surrogate model for the beam dynamics simulator to run an optimization of the design parameters of an RFQ like the one proposed in the IsoDAR experiment. Such an RFQ has the following optimal properties:

- Maximal beam transmission
- Minimal difference between output and target (70 KeV) energies
- Minimal RFQ length
- Minimal longitudinal emittance
- Minimal transverse emittance

Finding solutions to such a multidimensional optimization problem motivates the use of Bayesian optimization. The goal of such a procedure is to find the set of RFQs that are Pareto-optimal, where all points in the Pareto front are optimal in the sense that improvements in some OBJs come at the cost of others. To this end, we make use of the package Surrogates.jl, as part of the open-source Scientific ML Initiative of Julia [39, 40]. This package allows us to use our surrogate model as part of the acquisition function in the optimization algorithm to select and evaluate data points in the feature space. As the algorithm selects each 14-dimensional data point to test for optimality, we take the intermediate step of employing the procedure outlined in section 2.3 to determine whether the number of RFQ cells will be odd or even to extend the feature vector with a fifteenth entry. To narrow the Pareto-optimal solution space to a set of RFQ design parameters feasible for the goals of IsoDAR’s RFQ, we search for points in the Pareto set with beam transmission of at least 90% and as low as possible transverse beam emittance ($\epsilon_x, \epsilon_y \leq 0.04 \text{ cm mrad}$). Since longitudinal and transverse beam emittances are correlated, we expect that the longitudinal beam emittance will similarly be low.

3. Results

3.1. Results of the two preliminary hyperparameter scans

Loss curves for the full-dataset training of the 9 different NN architectures are shown in figure 6, top, and the validation set aggregate $R^2$ scores are summarized in table 3, top. The NN with the highest validation set aggregate $R^2$ is the NN with depth 5 and width 100, which we select over the $6 \times 100$ network due to the lower cross-fold standard deviation. This $5 \times 100$ NN is referred to as RFQNet1. Complete validation set MAPEs for all NNs generated in this scan are summarized in table C1 in appendix C.

As discussed in section 2.7, limiting the dataset used in training and testing NNs to samples with a transmission of at least 60% is physically well-motivated. Loss curves for each of the NNs trained on the data subset having transmission $\geq 60\%$ are shown in figure 6, bottom, and the validation-set aggregate $R^2$ scores are summarized in table 3, bottom. The depth 6 and width 100 NN had the highest aggregate $R^2$ score, and is referred to as RFQNet2. The complete validation set MAPEs for similar NNs scanned are summarized in table C2 in appendix C.

The test set MAPEs for both RFQNet1 and RFQNet2 are compared to the best-performing model from [18] in table 5. Checks for overfitting to the data are shown by reporting the training and validation set MAPEs for each of the 5 folds performed in the cross validation for choosing the optimal architectures of the NNs, figure 7. The MAPEs for the training and validation sets are generally close enough that we can conclude that neither model is overfit to the training data. Since RFQNet2’s MAPEs for each of the 6 OBJs are equal to or lower than those of RFQNet1, RFQNet2 should be chosen over RFQNet1 for use as a surrogate model in a design optimization.

3.2. Design optimization of IsoDAR’s RFQ

As mentioned in section 2.8, we use RFQNet2 as a surrogate model for the multi-OBJ Bayesian Optimization of an RFQ suitable for IsoDAR’s needs. This model was a depth 6, width 100 NN trained on data having to
Figure 6. Training set loss curves (mean squared error) for scanned neural network architectures (table 2), trained on the complete dataset (top) and samples having beam transmission $\geq 60\%$ (bottom). The solid line in the center of each curve represents the cross-fold loss mean, and one standard deviation is shaded above and below.

Table 3. Aggregated validation-set $R^2$ scores for each set of hyperparameters (table 2), for NNs trained on the complete dataset (top) and samples having transmission $\geq 60\%$ (bottom). Each $R^2$ score is computed across all objective variables, but is not representative of the prediction accuracy of individual objectives; MAPEs for each OBJ for each NN architecture are shown in appendix C. Small apparent differences in validation-set $R^2$ scores correspond to a few percentage-point differences in MAPEs.

|  | Depth 4 | Depth 5 | Depth 6 |
|---|---|---|---|
| Width 50 | 0.9884 ± 0.000311 | 0.9887 ± 0.000217 | 0.9893 ± 0.000127 |
| Width 75 | 0.9900 ± 0.000169 | 0.9904 ± 0.0000864 | 0.9907 ± 0.000145 |
| Width 100 | 0.9905 ± 0.0000763 | 0.9908 ± 0.0000698 | 0.9908 ± 0.00017 |

|  | Depth 4 | Depth 5 | Depth 6 |
|---|---|---|---|
| Width 50 | 0.9901 ± 0.0002 | 0.9902 ± 0.000174 | 0.9905 ± 0.00026 |
| Width 75 | 0.9912 ± 0.000129 | 0.9915 ± 0.000178 | 0.9918 ± 0.000217 |
| Width 100 | 0.9918 ± 0.0000719 | 0.9917 ± 0.000152 | 0.9920 ± 0.0000623 |

Table 4. Summary of RFQNet1 and RFQNet2, whose architectures where chosen by the cross-validation results of a hyperparameter scan over NN depth and width. $R^2$ scores correspond to those reported in table 3.

|  | Depth | Width | Validation-set $R^2$ | Additional specifications |
|---|---|---|---|---|
| RFQNet1 | 5 | 100 | 0.9908 ± 0.0000698 | Used complete training dataset |
| RFQNet2 | 6 | 100 | 0.9920 ± 0.0000623 | Subset of data having transmission $\geq 60\%$ |

beam transmissions of $\geq 60\%$. 2D projections of the 6D Pareto front are plotted in figure 8. Of the hundreds of points computed to be Pareto-optimal, we narrow down the possible designs to ones having transmissions of at least 95% with radial and longitudinal beam emittances below 0.04 cm-mrad and 0.04 MeV °, respectively. We then weigh these remaining options according to their overall performance in energy matching and minimal RFQ length. The optimum’s DVARs and corresponding beam output parameters from this study are summarized in table 6.
Table 5. Test set MAPEs for each of the 6 objective variables studied, compared to previous results in [18]. Specifications of RFQNet1 and RFQNet2 are given in table 4. These NNs were chosen via the hyperparameter scans summarized in table 3, top and bottom, respectively.

| Label | Objective variable | RFQNet1 | RFQNet2 | Reference [18] |
|-------|--------------------|---------|---------|----------------|
| OBJ1  | Transmission [%]   | 1.5%    | 0.97%   | 2.4%           |
| OBJ2  | $E_{out}$ [MeV]    | 1.8%    | 1.8%    | 1.9%           |
| OBJ3  | RFQ length [cm]    | 1.3%    | 1.3%    | 2.0%           |
| OBJ4  | $\epsilon_{long}$ [MeV-deg] | 6.9%    | 5.8%    | 8.2%           |
| OBJ5  | $\epsilon_x$ [cm-mrad] | 4.8%    | 4.1%    | 12.8%          |
| OBJ6  | $\epsilon_y$ [cm-mrad] | 4.8%    | 4.0%    | 12.5%          |

Figure 7. Comparison of mean absolute percent errors (MAPEs) on each of the 5 performed cross-validation folds on training and validation datasets for RFQNet1 (top) and RFQNet2 (bottom). RFQNet1/RFQNet2 are the optimal networks whose defining characteristics are summarized in table 4. Nearly equivalent MAPEs for both training and validation sets point to models that are unlikely to be overfit to the training data.

It is important to note that the boundaries and relationships between the first 14 DVARS were chosen to ensure physical realizability of an RFQ. Because these were folded into the optimization procedure (via the decorrelation procedure described in section 2.2), the optimal RFQ is indeed a physically realistic design.
Figure 8. The RFQ design optimal for IsoDAR's use case was computed by running a multi-objective Bayesian optimization scheme using the best-performing NN as the surrogate model for the algorithm's acquisition function. 2D projections of the 6D Pareto front are shown above, with the optimal design configuration (see Table 6) marked with an orange star.

Table 6. Optimal design configuration and corresponding beam summary statistics of RFQ for IsoDAR's use case. This RFQ's design was created by running a multi-objective Bayesian optimization using the best-performing NN, RFQNet2, as a surrogate model for the beam dynamics simulation. For the optimal set of design variables shown, RFQNet2's predictions for the beam summary parameters are shown alongside PARMTEQM's simulation results.

| Label | Design variable | Optimal value |
|-------|-----------------|---------------|
| DVAR1 | Bmax [1]        | 9.1289        |
| DVAR2 | mX1 [cm]        | 46.1325       |
| DVAR3 | mX2 [cm]        | 92.6490       |
| DVAR4 | mY1 [1]         | 1.0430        |
| DVAR5 | mY2 [1]         | 1.4656        |
| DVAR6 | mtau1 [cm]      | 425.9281      |
| DVAR7 | mtau2 [cm]      | 160.8364      |
| DVAR8 | PhiY1 [deg]     | -82.9247      |
| DVAR9 | PhiY2 [deg]     | -68.7336      |
| DVAR10| Phitau1 [cm]    | 199.8206      |
| DVAR11| Phitau2 [cm]    | 176.4324      |
| DVAR12| mY3ref [1]      | 1.6405        |
| DVAR13| PhiY3ref [deg]  | -56.4804      |
| DVAR14| Eref [MeV]      | 0.0723        |

| Label | Objective variable | Predicted value | Confirmed value (% error) |
|-------|---------------------|-----------------|---------------------------|
| OBJ1  | Beam transmission [%]| 96.4862         | 97.14 (0.67%)             |
| OBJ2  | Output energy [MeV]| 0.0737          | 0.0750 (0.27%)            |
| OBJ3  | RFQ length [cm]    | 166.5253        | 163.56 (1.8%)             |
| OBJ4  | Longitudinal emittance [MeV deg]| 0.0374 | 0.0431 (13%) |
| OBJ5  | x-emittance [cm mrad]| 0.0366 | 0.03369 (8.6%) |
| OBJ6  | y-emittance [cm mrad]| 0.0305 | 0.02792 (9.2%) |
4. Discussion

In this work, we, for the first time, build a surrogate model (RFQNet2) for an RFQ-throughgoing beam simulator that can achieve <6% MAPE on predictions of the outgoing beam dynamics. The performance of this surrogate model by OBJ studied can be seen in table 5. Significant improvements in prediction accuracy were achieved by training a series of NNs with an unusually large batch size of 1024, and by unveiling a previously hidden variable: the number of RFQ cells, which is computed deterministically from the other 14 DVARs. The test-set MAPEs are improved over previous work for each of the OBJs studied; in the case of transverse emittances, the MAPEs are a third of those seen in [18]. However, we should note that there is no one-size-fits-all NN that achieves the lowest possible MAPE for all OBJs studied; this is emphasized by the boldfaced MAPEs shown in the tables in appendix C.

4.1. Time costs of surrogate models and their development

The development of surrogate models, in this study, is meant to (1) reduce the computational burden that running high-fidelity simulations places on optimization routines; and (2) facilitate quasi-real-time feedback for accelerator operators during tuning and running a machine. The timescale of fine-tuning an RFQ can quickly balloon; a single run of PARMTEQM to compute the summary statistics of a beam traversing an RFQ of arbitrary design takes, conservatively, 30 s, while the surrogate models RFQNet1 and RFQNet2 execute in just a fraction of a second.

We must point out, however, that the end-to-end development of the surrogate models presented do not, on their own, beat PARMTEQM at saving time for a single optimization. Data generation for the training and test samples took about a week of continuous running on a single workstation. Performing hyperparameter scans and cross-validation took another day of runtime. A single optimization run took about 1000 function calls to the surrogate model.

However, considering the lengthy process of designing an accelerator system (typically consisting of more elements than the RFQ) and iterations on design choices, it is not realistic to assume a single optimization run will yield the desired machine. Rather, tens of optimizations will have to be performed with ever changing output energies, beam sizes, etc to match the RFQ to the previous and following acceleration stages.

Finally, an avenue of future study will be to understand the lower limit of feasible training sample sizes. Generation of the training data used in this study served as the main bottleneck, and a circumstance in which fewer training samples could be used to build surrogate models with comparable predictive accuracy may, eventually, prove faster than PARMTEQM.

The prototyping of an accurate surrogate model demonstrated in this study paves the way for future work using high-fidelity simulation codes based on the PIC method, like OPAL [41] and WARP [42]. As opposed to PARMTEQM, these codes can accurately calculate the behaviour of space-charge dominated beams, albeit at much higher computation time (it can take hours to compute a beam traversing a single RFQ). The replacement of these softwares with fast-executing surrogate models would result in orders of magnitude improvement in runtime [1]. Moreover, even though PARMTEQM is much faster than these higher-fidelity codes, it would still be hard to use it in a real-time commissioning tool due to its ≈30 s runtime, while RFQNet1 or RFQNet2 are, by comparison, virtually instantaneous. One can envision a program that allows an accelerator scientist to make small adjustments to certain controllable RFQ parameters (such as the target energy or the electromagnetic fields responsible for beam focusing, shaping, and bunching) and see, in almost real-time, a description of the outgoing beam. Such a program could find use during the running of an accelerator, where rapid computations are necessary to ensure the successful operation of the RFQ, and would further justify the need for the development of surrogate models similar to those presented here.

4.2. Recommendations

The significant improvements in MAPE for top-performing NNs in this work over [18] lead us recommend the following avenues for future work:

(i) **Explore different NN architectures for different OBJs.** Different OBJ variables preferred different NN architectures (indicated explicitly in the out-of-sample MAPEs for each OBJ in appendix C). We can feasibly run hyperparameter scans for NNs aiming to predict one (or perhaps some subset of) OBJs for additional boosts in performance. We are especially interested in implementing such models for the poorest-performing OBJ, namely, longitudinal emittance.

(ii) **Limit datasets to physically realistic scenarios.** A large boost in performance of the NN was due in part to the rejection of data samples with simulated beam transmission below 60%. In the real-world commissioning of an RFQ, beam transmissions below 60% are unrealistic, and we expect that the quality of the simulation of softwares like PARMTEQM would deteriorate in these circumstances.
While the DVAR boundaries were set as large as practically physically applicable when generating the initial training dataset to cover a possible large number of (even unreasonable) RFQ configurations, ensuring physically realistic scenarios enhances general data quality and predictive accuracy. (iii) **Explore surrogate model architectures beyond fully-connected NNs and summary statistic inputs.** We emphasize that multivariate regressive tasks are not limited to just fully-connected NNs, or that the only way to develop an RFQ surrogate model is by feeding the exact same 14 DVARs that we reference in this study. We are especially interested in the use of Residual NNs (ResNets) for improving the predictive accuracy of the general multivariate regression task, as well as the use of convolutional neural networks that can be trained on pixelated ‘pictures’ of the throughgoing beam, giving a more granular picture of the underlying physics. Recurrent neural networks may also be useful in propagating a beam iteratively through some number of RFQ cells.

### 5. Conclusion

We presented an in-depth investigation of surrogate modeling for RFQ linear particle accelerators. We especially scrutinized input data preparation, and unearthed hidden variables to address the predictive shortcomings of previous works. Correlations between the DVARs did not impact the quality of the model, though we still recommend decorrelation especially when performing Bayesian optimizations necessitating complex relationships between feature variables.

Choosing physically sensible limits and removal of nonphysical data points is essential. For example, we omitted all data with RFQ transmission <60% from NN training, as these data points are undesirable (and sometimes even nonphysical) in the context of particle accelerator engineering.

In order to resolve asymmetries between response variables that are otherwise symmetric, such as the two transverse emittances, it is essential to carefully prepare data to include some variable having this distinguishing power. For example, determining whether the emittance in the x or y directions were higher was only possible if we included whether the number of RFQ cells was odd or even.

With these considerations in mind, our best-performing model was a depth 6, width 100 NN trained on a \( \approx 160{,}000 \) subsample of >200,000 point dataset having 14 DVARs, one additional deterministic feature, and 6 OBJs, which performs exceptionally well compared to previous efforts of a similar kind (we reproduce all relevant OBJs with a MAPE of <6%). To demonstrate the utility of this surrogate model, we performed a multi-OBJ optimization of the IsoDAR RFQ, and confirmed the optimized design using PARMTEQM (the simulation software used to generate the training data).

Our next steps are to test network architectures other than deep fully connected networks and to build surrogate models for other types of particle accelerators such as the IsoDAR cyclotron. These are important steps forward to creating a system of surrogate models that can speed up the development and real-time tuning of a multitude of particle accelerator experiments.

**Data availability statement**

No new data were created or analysed in this study.

**Acknowledgments**

We would like to thank Andreas Adelmann for valuable input on the topic of surrogate modeling of particle accelerators. We are grateful to the broader Julia community, whose expertise and responsiveness helped make this work possible. This work was supported by NSF Grants PHY-1505858 and PHY-1626069. J V was supported by the School of Science at Massachusetts Institute of Technology. D W was supported by funding from the Bose Foundation and the Heising-Simons Foundation.
Appendix A. Histograms of the transformed design variable

Figure A1. Histograms of the unscaled (left) and transformed (right) DVAR3. The transformation is outlined in equation (2).

Appendix B. Joint distributions of true and predicted x- and y-emittances for early neural networks

In early versions of this work, we had trained a series of neural networks for which we did not specify the parity of the number of RFQ cells as a fifteenth DVAR. The predicted RMS transverse emittances of the best-performing of these neural networks are shown as a joint distribution to compare to the true underlying joint distribution.

Figure B1. Joint distributions of true x and y emittances (OBJ5 and OBJ6, respectively) for sample training and test sets. From the inherent x, y symmetry of RFQs, we expect approximate symmetry about the $\epsilon_y = \epsilon_x$ line.
Appendix C. Out-of-sample MAPEs for scanned neural networks

Since the validation-set $R^2$ of each of the 9 NN architectures presented in table 3 do not account for the predictive accuracies of each of the 6 OBJ variables separately, we report the MAPE for each OBJ for each neural network architecture scanned, as the cross-fold average $\pm$ standard deviation. No one neural network observed the lowest validation set MAPE for all OBJ variables, highlighting the conclusion that different neural network architectures may be better equipped to predict different OBJs.
Table C2. Cross-fold average ± standard deviation validation-set MAPEs by objective variable, for each of the 9 NN architectures explored in the initial hyperparameter scan (table 2), trained on samples having transmissions ≥60%. RFQNet2 was selected to be the depth 6, width 100 neural network based on scores reported in table 3, bottom, though different objectives preferred different NN architectures. The lowest MAPEs for each objective are shown in bold.

|       | Depth 4 | Depth 5 | Depth 6 |
|-------|---------|---------|---------|
| OBJ1  |         |         |         |
| Width 50 | 0.01212 ± 0.000647 | 0.0107 ± 0.000246 | **0.009651 ± 0.000243** |
| Width 75 | 0.0150 ± 0.000234  | 0.01054 ± 0.000316 | 0.009738 ± 0.000283 |
| Width 100 | 0.01238 ± 0.000222 | 0.01035 ± 0.000462 | 0.009979 ± 0.000444 |
| OBJ2  |         |         |         |
| Width 50 | 0.01801 ± 0.000287 | **0.01786 ± 0.000076** | 0.01793 ± 0.0000767 |
| Width 75 | 0.01787 ± 0.0000562 | 0.01792 ± 0.000131 | 0.01801 ± 0.0000828 |
| Width 100 | 0.01799 ± 0.00017  | 0.01793 ± 0.000221 | 0.01797 ± 0.0000958 |
| OBJ3  |         |         |         |
| Width 50 | 0.01479 ± 0.00132  | 0.01304 ± 0.00027  | **0.01227 ± 0.000135** |
| Width 75 | 0.01385 ± 0.000213 | 0.01291 ± 0.00032  | 0.01270 ± 0.000252 |
| Width 100 | 0.01467 ± 0.000487 | 0.01295 ± 0.000215 | 0.01289 ± 0.000336 |
| OBJ4  |         |         |         |
| Width 50 | 0.06922 ± 0.0017   | 0.06335 ± 0.00356  | 0.05802 ± 0.00209 |
| Width 75 | 0.07061 ± 0.000696 | 0.06256 ± 0.00365  | **0.05704 ± 0.000965** |
| Width 100 | 0.06896 ± 0.0014   | 0.0598 ± 0.0014    | 0.05864 ± 0.00353  |
| OBJ5  |         |         |         |
| Width 50 | 0.05445 ± 0.00153  | 0.04895 ± 0.00241  | 0.04367 ± 0.000593 |
| Width 75 | 0.05528 ± 0.0027   | 0.04704 ± 0.00243  | 0.04236 ± 0.000757 |
| Width 100 | 0.05177 ± 0.00119  | 0.04313 ± 0.000659 | **0.04110 ± 0.000769** |
| OBJ6  |         |         |         |
| Width 50 | 0.05438 ± 0.00151  | 0.04818 ± 0.00143  | 0.04310 ± 0.000202 |
| Width 75 | 0.05398 ± 0.00156  | 0.04596 ± 0.000912 | 0.04257 ± 0.00107  |
| Width 100 | 0.05159 ± 0.00109  | 0.04305 ± 0.000586 | **0.04095 ± 0.000797** |

ORCID iDs

Joshua Villarreal https://orcid.org/0000-0001-9690-1310
Daniel Winklehner https://orcid.org/0000-0002-0715-6310
Daniel Koser https://orcid.org/0009-0005-3062-0360
Janet M Conrad https://orcid.org/0000-0002-6393-0438

References

[1] Edelen A, Neveu N, Frey M, Huber Y, Mayes C and Adelmann A 2020 Phys. Rev. Accel. Beams 23 044601
[2] Adelmann A 2019 SIMA/ASA J. Uncertain. Quantification 7 383–416
[3] Vay J L, Huebl A, Lehe R, Cook N M, England R J, Niedermayer U, Piot P, Tsung F and Winklehner D 2021 J. Instrum. 16 T10003
[4] Sagan D et al 2021 J. Instrum. 16 T10002
[5] Bellotti R, Boiger R and Adelmann A 2021 Information 12 351
[6] Adelmann A et al 2022 New directions for surrogate models and differentiable programming for high energy physics detector simulation (arXiv:2203.08806 [hep-ex, physics/hep-ph, physics:hep-ex])
[7] Gupta L, Edelen A, Neveu N, Mishra A, Mayes C and Kim Y K 2021 Mach. Learn.: Sci. Technol. 2 045025
[8] Bungau A et al 2012 Phys. Rev. Lett. 109 141802
[9] Alonso J R et al 2022 J. Instrum. 17 09042
[10] Alonso J R et al 2022 Phys. Rev. D 105 052009
[11] Mention G, Fechner M, Lasserre T, Mueller T A, Lhuillier D, Cribier M and Letourneau A 2011 Phys. Rev. D 83 073006
[12] An F P et al (Daya Bay Collaboration) 2016 Phys. Rev. Lett. 116 061801
[13] Abramenko P et al (MicroBooNE Collaboration) 2022 Phys. Rev. Lett. 128 241801
[14] Barinov V et al 2022 Phys. Rev. C 105 065502
[15] Seo S H et al 2023 Physics potential of a few kiloton scale neutrino detector at a deep underground lab in Korea (arXiv:2309.13435 [hep-ex, physics/hep-ph])
[16] Winklehner D et al 2018 Nucl. Instrum. Methods Phys. Res. A 907 231–43
[17] Winklehner D, Adelmann A, Conrad J M, Mayani S, Muralikrishnan S, Schoen D and Yampolskaya M 2021 arXiv:2103.09352 [physics]
[18] Koser D, Waites L, Winklehner D, Frey M, Adelmann A and Conrad J 2022 Front. Phys. 10 875889
[19] Bezanson J, Edelman A, Karpinski S and Shah V 2017 SIAM Rev. 59 65–98
[20] Villarreal J and Winklehner D 2022 rfq-nn (available at: https://github.com/joshvillarreal/rfq-nn)
[21] Kapchinskii I M and Teplyakov V A 1970 Instrum. Exp. Tech. (USSR) (Engl. Transl.) (available at: www.slac.stanford.edu/pubs/slactrans/trans01/slac-trans-0099.pdf)
[22] Crandall K, Stokes R and Wangler T 1979 RF quadrupole beam dynamics design studies Proc. LINAC1979
[23] Wangler T P 2008 RF Linear Accelerators (Wiley)
[24] Höltermann H et al 2021 Technical design of an RFQ injector for the IsoDAR cyclotron Proc. 12th Int. Particle Accelerator Conf. vol IPAC2021 p 3
[25] Kosor D, Conrad J, Podlech H, Ratzinger U, Schuett M and Winklehner D 2021 Thermal analysis of a compact split-coaxial CW RFQ for the IsoDAR RFQ-DIP Proc. 12th Int. Particle Accelerator Conf. vol IPAC2021 p 3
[26] Besard T, Foket C and De Sutter B 2018 IEEE Trans. Parallel Distrib. Syst. 30 827–41
[27] Besard T, Churavy V, Edelman A and De Sutter B 2019 Adv. Eng. Softw. 132 29–46
[28] Billen J, Crandall K and Young I. 2010 Phase and radial motion in a transverse electric quadrupole -manufacturing (available at: www.osti.gov/biblio/1375180)
[29] Crandall K R, Crandall C, Park S and Mexico N 1994 Proc. 1994 Int. Linac Conf. (Tsukuba, Japan) (available at: https://accelconf.web.cern.ch/l94/papers/mo-69.pdf)
[30] Crandall K R and Wangler T P 1988 AIP Conf. Proc. 177 22–28
[31] Crandall K R 1984 RFQ radial matching sections and fringe fields Tech. Rep. LA-UR-84-1281 CONF-840529-23 Los Alamos National Lab., NM (USA) (available at: www.osti.gov/biblio/7136994)
[32] Crandall K R, Stokes R H and Wangler T P 1979 RF quadrupole beam dynamics design studies Tech. Rep. LA-UR-79-2499 CONF-790927-17 Los Alamos Scientific Lab., NM (USA) (available at: www.osti.gov/biblio/5831205)
[33] subMIT – getting physics things done at MIT 2022 (available at: https://submit.mit.edu/)
[34] Innes M, Saba E, Fischer K, Gandhi D, Rudolosso M C, Joy N M, Karmali T, Pal A and Shah V 2018 arXiv:1811.01457
[35] Innes M 2018 J. Open Source Softw. 3 25
[36] Golmant N, Vemuri N, Yao Z, Feinberg V, Gholami A, Rothauge K, Mahoney M W and Gonzalez J 2018 arXiv:1811.12941
[37] MLUtils: utilities and abstractions for machine learning tasks 2022 (available at: https://github.com/JuliaML/MLUtils.jl)
[38] Kingma D P and Ba J 2014 CoRR arXiv:1412.6980
[39] Rackauckas C and Nie Q 2017 J. Open Res. Softw. 5 15
[40] Rackauckas C, Ma Y, Martensen J, Warner C, Zubov K, Supekar R, Skinner D and Ramadhan A 2020 arXiv:2001.04385
[41] Adelmann A et al 2019 OPAL a versatile tool for charged particle accelerator simulations (arXiv:1905.06654 [physics])
[42] Vay J L, Grote D P, Cohen R H and Friedman A 2012 Comput. Sci. Discov. 5 014019