TO: ESI group
FROM: Joshua Eckels
DATE: March 14, 2022
SUBJECT: Electric field surrogate

Summary This is a brief write-up on the details of the electric field surrogate, its limitations, and how to use it. The code is available on Github. You can download the project directory there as a .zip file and access an example usage of the surrogate in the `surrogate_example/` folder. Details on how to use the surrogate are in Section 1.5.

Contents

1 Surrogate details

1.1 Architecture

The surrogate is a simple feedforward neural network with 4 feature inputs, two fully-connected hidden layers, and one regression output, as shown in Figure 1.

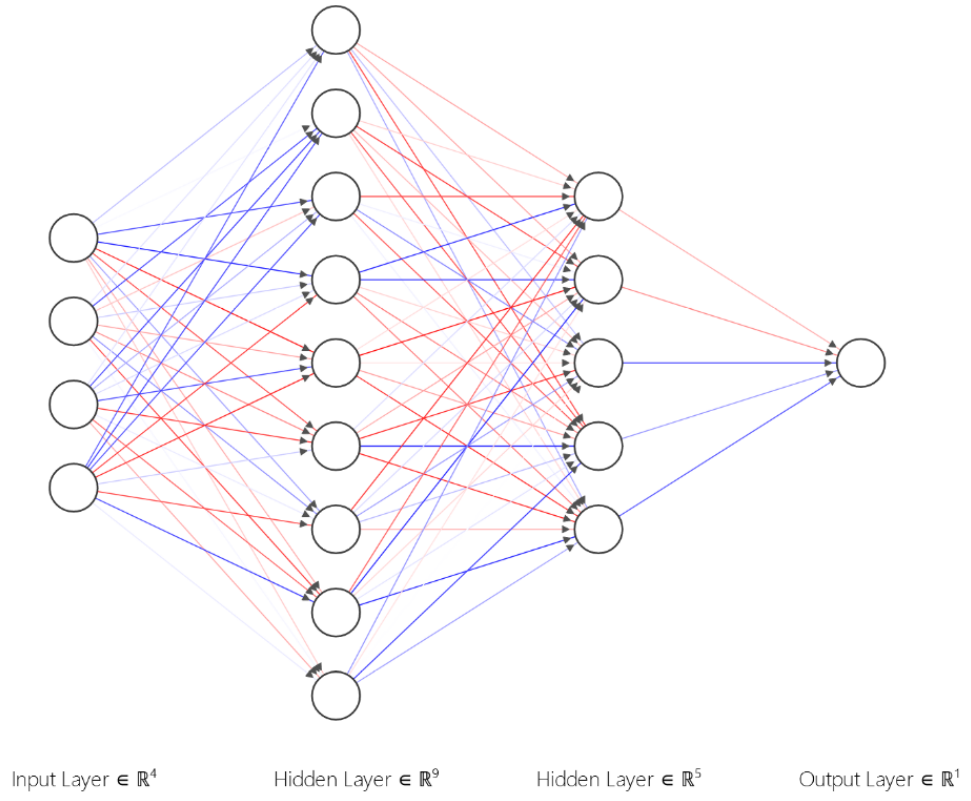


Figure 1: Fully-connected neural network (FCNN)

The activation of the first node of the first hidden layer is computed as the weighted sum of the 4 feature inputs x_i plus a scalar bias term b_1:

$$a_1 = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b_1 \tag{1}$$

Likewise, the activation of node j in the first hidden layer is

$$a_j = \sum_{i=1}^{N} w_{i,j} x_i + b_j \tag{2}$$

where $i = 1 \dots N$ are the feature inputs and $w_{i,j}$ is the weight from feature input i to node j in the first hidden layer. The activation of all N_1 nodes in the first hidden layer can then be written as the matrix product

$$\begin{bmatrix} a_1 \\ \vdots \\ a_j \\ \vdots \\ a_{N_1} \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{2,1} & w_{3,1} & w_{4,1} \\ \vdots & & & \vdots \\ w_{1,j} & w_{2,j} & w_{3,j} & w_{4,j} \\ \vdots & & & \vdots \\ w_{1,N_1} & \dots & \dots & w_{4,N_1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_j \\ \vdots \\ b_{N_1} \end{bmatrix} \tag{3}$$

More generally, the activation of all N_k nodes in the k^{th} layer is given by

$$\begin{bmatrix} a_{1,k} \\ \vdots \\ a_{j,k} \\ \vdots \\ a_{N_k,k} \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{2,1} & \dots & w_{N_{k-1},1} \\ \vdots & & & \vdots \\ w_{1,j} & w_{2,j} & \dots & w_{N_{k-1},j} \\ \vdots & & & \vdots \\ w_{1,N_k} & \dots & \dots & w_{N_{k-1},N_k} \end{bmatrix} \begin{bmatrix} a_{1,k-1} \\ a_{2,k-1} \\ \vdots \\ a_{N_{k-1},k-1} \end{bmatrix} + \begin{bmatrix} b_{1,k} \\ \vdots \\ b_{j,k} \\ \vdots \\ b_{N_k,k} \end{bmatrix} \tag{4}$$

where $a_{j,k-1}$ is the activation of the j^{th} node in the previous layer. This matrix product is typically written in shorthand as

$$\mathbf{a_k} = \mathbf{W_k a_{k-1}} + \mathbf{b_k} \tag{5}$$

where W_k is the weight matrix of the connections to the k^{th} layer, b_k is the bias vector of the k^{th} layer, and a_k is the node activations of the k^{th} layer. After evaluating Eq. (5), a nonlinear function σ is applied element-wise to a_k:

$$\mathbf{a_k} = \sigma\left(\mathbf{W_k a_{k-1}} + \mathbf{b_k}\right) \tag{6}$$

to produce the final node activations of the k^{th} layer. Each additional network layer takes the previous activations a_k as an input and applies a new weight matrix and bias vector. From a layer with N_{k-1} nodes to a layer with N_k nodes, the total number of additional trainable parameters p_k (weights and biases) is given by:

$$p_k = N_k N_{k-1} + N_k \tag{7}$$

For a FCNN with h hidden layers, the total number of trainable parameters is

$$p_T = \sum_{k=1}^{h+1} p_k \tag{8}$$

where $N_0 = N_x$ is the number of network feature inputs and $N_{h+1} = N_y$ is the number of network regression outputs. For a FCNN with h hidden layers, N_x input features, and N_y regression outputs, the regression output $y \in \mathbb{R}^{N_y}$ after a full forward pass of a given network input $x \in \mathbb{R}^{N_x}$ is given by

$$y = \mathbf{W_{h+1} a_h} + \mathbf{b_{h+1}}, \qquad \text{where} \tag{9}$$

$$\mathbf{a_h} = K_{k=1}^{h}[\sigma\left(\mathbf{W_k a_{k-1}} + \mathbf{b_k}\right)] \tag{10}$$

where the $K_{k=1}^{h}[\cdot]$ notation implies repeated function composition of the quantity in the brackets. It can be seen that there is a pure linear mapping W_{h+1} from the last hidden layer to the regression output (i.e. no nonlinear function $\sigma(\cdot)$ is applied). Also, the a_0 activations are the network inputs (i.e. $\mathbf{a_0} = x$). The surrogate presented here uses the hyperbolic tangent function as the nonlinear function in each layer, which maps inputs x to the range $[-1, 1]$:

$$\sigma(\cdot) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{11}$$

A summary of the surrogate under study is provided in Table 1.

Table 1: Electric field FCNN surrogate summary

Parameter	Symbol	Value
Number of inputs	N_x	4
Number of outputs	N_y	1
Number of hidden layers	h	2
Hidden layer sizes	N_k	$[9, 5]$
Number of trainable parameters	p_T	105

1.2 Inputs and outputs

The goal of the surrogate is to learn the mapping from emitter geometry to peak electric field. The geometry features under consideration are the tip-to-extractor distance d, radius of curvature r_c, cone half-angle α, cone height h, and radius of aperture of the emitter. The output of interest is the maximum electric field E_{max} magnitude developed near the surface of the emitter under an applied electrostatic bias voltage.

1.3 Preprocessing

Since geometry parameters are on the scale $O(1\,\mu\text{m})$ and electric field strength is on the scale $O(10\,\text{MV m}^{-1})$, it is necessary to normalize all parameters before training the network. In addition, it is advantageous to nondimensionalize all parameters before training so that the network can be applied to situations with dissimilar units or scale to the data it was trained on.

1.3.1 Nondimensionalization

All parameters with length dimension were scaled by the height of the emitter cone. The cone-half angle was expressed in dimensionless units of radians. The electric field strength was additionally scaled by the applied bias voltage V_0. Table 2 provides a summary of the nondimensionalization procedure.

Table 2: Parameter nondimensionalization

Parameter	Symbol	Units	Scaling
Tip-to-extractor distance	d	µm	$\tilde{d} = \frac{d}{h}$
Radius of curvature	r_c	µm	$\tilde{r_c} = \frac{r_c}{h}$
Cone half-angle	α	rad	$\tilde{\alpha} = \alpha$
Cone height	h	µm	$\tilde{h} = 1$
Radius of aperture	r_a	µm	$\tilde{r_a} = \frac{r_a}{h}$
Electric field strength	E_{max}	V m^{-1}	$\tilde{E}_{max} = \frac{E_{max}}{V_0/h}$

1.3.2 Normalization

After nondimensionalization, the radius of curvature and the electric field strength still covered orders of magnitude in range over the training data. To reduce this range before normalization, the common logarithm was applied to these two parameters:

$$\tilde{\tilde{r}}_c = \log_{10} \tilde{r}_c \tag{12}$$

$$\tilde{\tilde{E}}_{max} = \log_{10} \tilde{E}_{max} \tag{13}$$

All nondimensional parameters were normalized linearly to the range $[-1, 1]$ before training according to the mapping:

$$x_n = \frac{x - x_{min}}{x_{max} - x_{min}}(y_{max} - y_{min}) + y_{min} \tag{14}$$

where $[x_{min}, x_{max}]$ corresponds to the range of the parameter x and $[y_{min}, y_{max}] = [-1, 1]$ corresponds to the range of the normalized parameter x_n. After normalization, the network is only exposed to inputs and targets that fall in the range $[-1, 1]$; this aids in computational speed and the accuracy of gradient descent algorithms. This also corresponds with the choice of the tanh nonlinear function that naturally maps to the range $[-1, 1]$ in between each layer of the network. The flow of data from raw geometry inputs to dimensionalized regression outputs is shown in Figure 2.

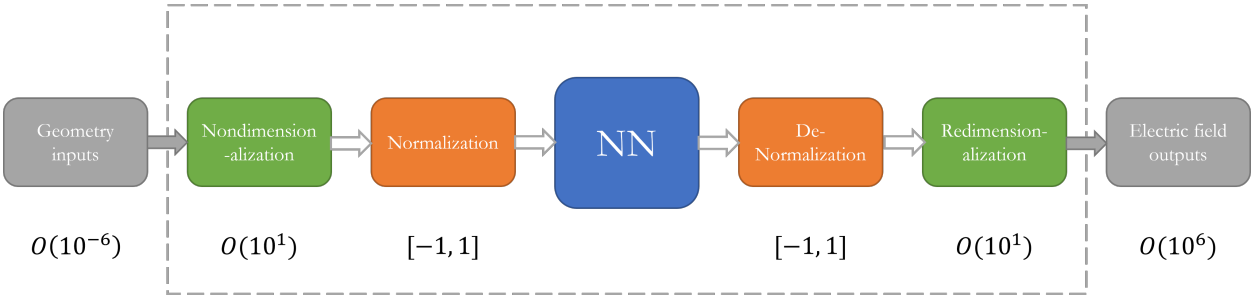


Figure 2: Flow of data from inputs to outputs

That which is in the bounding box is hidden from the user; they must only worry about passing in dimensional geometry inputs and receiving dimensional electric field outputs. Since the cone height h geometry parameter was scaled to $\tilde{h} = 1$, it is not passed to the network directly as an input; i.e. there are 4 network inputs rather than 5.

1.4 Training details

For a given set of inputs $x \in \mathbb{R}^{N_x}$ and newtork parameters $p \in \mathbb{R}^{N_{pT}}$, the network will return a set of regression outputs $\hat{y} \in \mathbb{R}^{N_y}$, which are approximations of the mapping $y(x) : x \to y$. For a set of N training samples $[x_i, y_i]$, $i = 1 \dots N$, where y_i is the known target value of the input x_i, the cost function of the regression predictor over the training set is expressed as the mean squared error (MSE):

$$C(p) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i(x_i; p))^2 \tag{15}$$

where the cost $C(p)$ is a function of the current network parameters $p = \{w, b\}$ for a given set of training data. In cases where $N_y > 1$, the regressor error $y_i - \hat{y}_i$ is evaluated as the MSE over N_y

dimensions.

The task of training the network is now one of optimization:

$$p^* = \underset{p}{\operatorname{argmin}}\, C(p; x, y) \tag{16}$$

where the optimal network parameters p^* are found by minimizing the cost function, given a set of training data $[x, y]$. The Adam optimizer is a popular gradient descent method for minimizing this cost function for neural networks. The Adam optimizer is in the class of stochastic gradient descent (SGD) algorithms, which use one small subset (batch) of the full training set at a time to compute gradients and update parameters. SGD typically converges faster than if the full dataset was used for each gradient step. In the surrogate studied here, the Adam optimizer was used with L_2 regularization of the network weights, the default initial learning rate of $\alpha = 0.001$, and a batch size of 2048. Trial-and-error tunings of these parameters showed that small adjustments to the batch size and the learning rate mainly impacted computational time rather than the accuracy of convergence. The Adam optimizer also keeps an element-wise moving average of the parameter gradients to adjust the "momentum" of the gradient descent algorithm adaptively. The optimizer is provided by default in the Matlab environment.

1.5 Usage

The `surrogate_example/` directory in the Github repository contains example code to load geometry samples from a text file, run inference on the geometry, and plot output electric field strength. The trained network is contained in the `esi_surrogate.onnx` file, and the `esi_surrogate.forward` Python method can be imported and used to run inference with the .onnx network file. The `forward` function is dependent on the normalization settings of the training data, which is contained in the `norm_data.mat` file. The normalization settings are also hard-coded into the `esi_surrogate` module.

Note that the `surrogate_example/` directory is a self-contained copy of code in the rest of the repository for convenient access. The original source files for training and testing the network are located in the `src/` directory, and the data should be stored in the appropriate hierarchy in the `data/` directory. Also note that the actual simulation data files are not kept in the remote Github repository because of storage requirements; I can provide these via Dropbox or some other method if need be.

2 Simulation

Matlab's electrostatic partial differential equation (PDE) simulation capability was used to generate electric field data for varying emitter geometries. The surface of the emitter cone was placed at a positive bias voltage from the extractor grid in the simulations, and a far field 0V boundary condition was applied away from the regions of interest. The simulations were performed in a 2D, axisymmetric domain. An example of the results from an electrostatic simulation are shown in Figure 3.

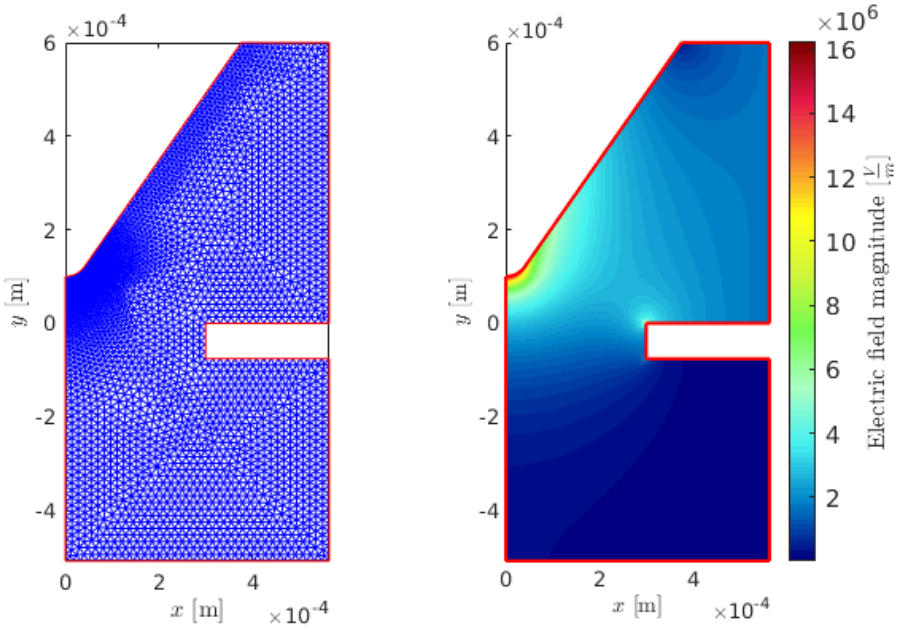
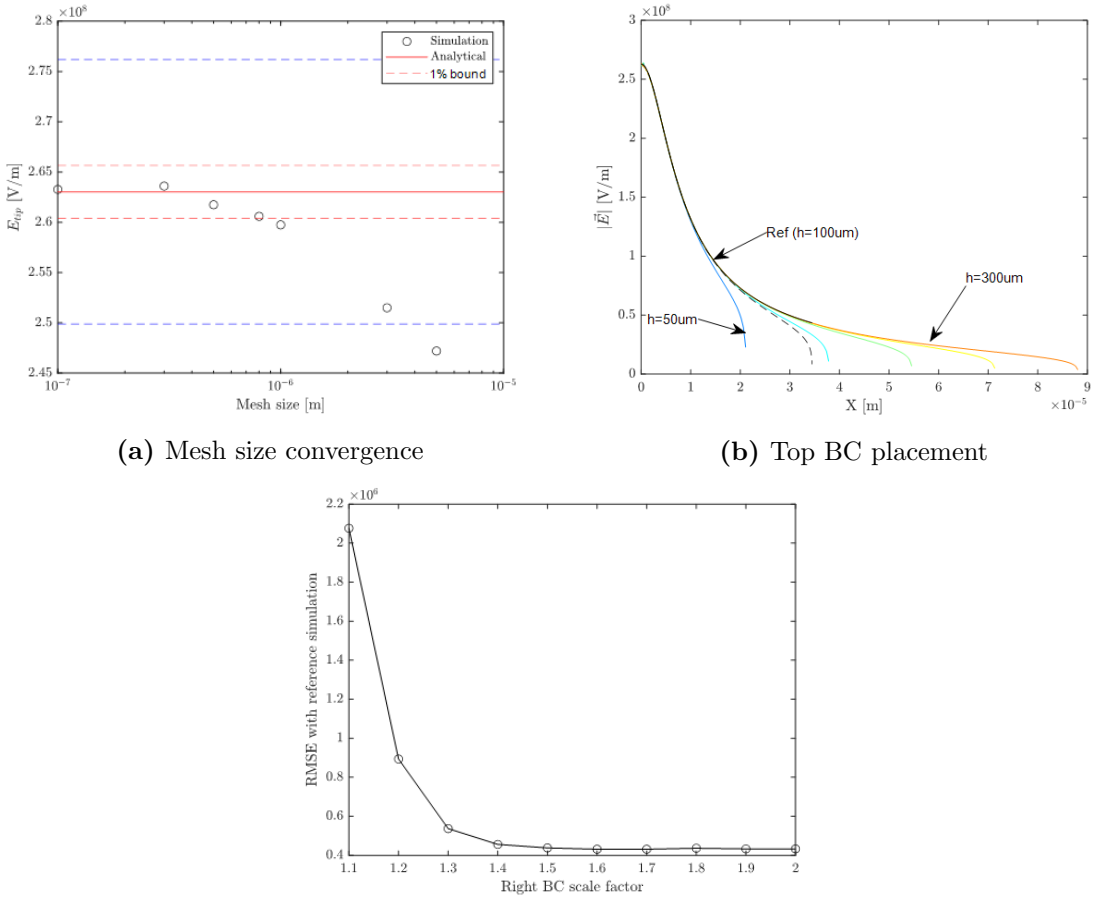


Figure 3: 2D axisymmetric electrostatic simulation of an emitter

The mesh was tuned adaptively during data generation to ensure the presence of enough grid points at the tip of the emitter, despite large changes in length dimension. The simulation procedure was validated using the results from a simulated hyperboloid geometry against the analytical Martinez-Sanchez solution. The convergence of the simulation to the analytical value was confirmed with respect to mesh size and boundary condition (BC) placements, as shown in Figure 4.



(a) Mesh size convergence

(b) Top BC placement

(c) Right BC placement

Figure 4: Convergence studies on the electrostatic simulation

3 Datasets

3.1 Feasible performance-based design space

The sampling procedure in [1] was used to generate a dataset of 16,836 emitter geometries. The geometries in this design space satisfy a set of basic, practical geometry constraints (e.g. cone height is not smaller than the radius of curvature) and an additional set of thruster performance requirements, such as minimum thrust, specific impulse, efficiency, etc. The 1D and 2D marginals of this design space are shown in Figure 5.

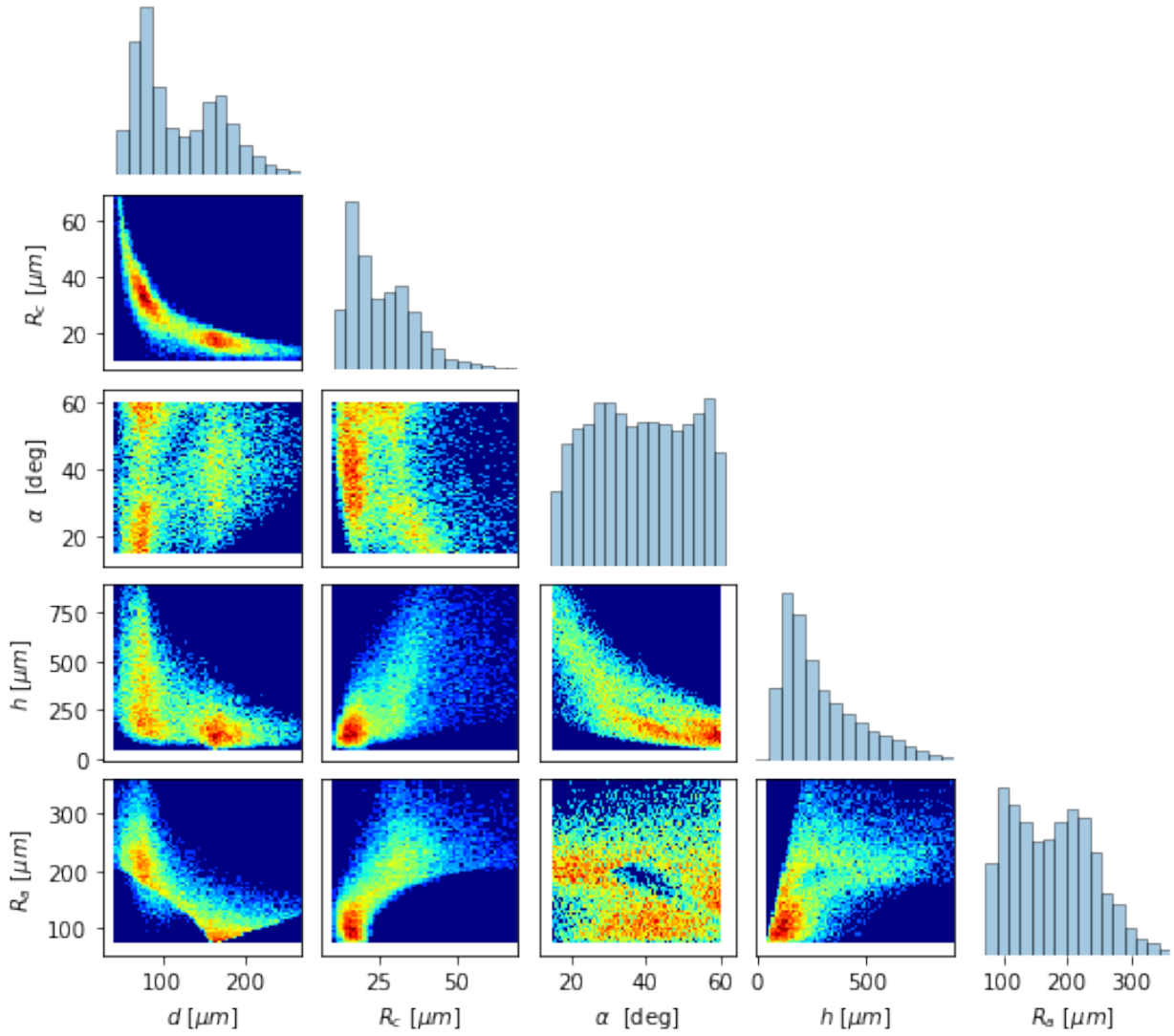


Figure 5: Feasible performance-based design space 1D and 2D marginals

3.2 General geometry-based design space

One limitation of the feasible design space is that the performance requirements are estimated using the Martinez-Sanchez approximation to the electric field strength, which the reader will note is the very same model that this surrogate is meant to replace. In order to realize the much larger design space that exists outside of these performance constraints, another set of $16,000$ geometry samples were collected using the same geometry constraints, but with the performance constraints lifted. This (much) larger design space is shown in the 1D and 2D marginals in Figure 6.

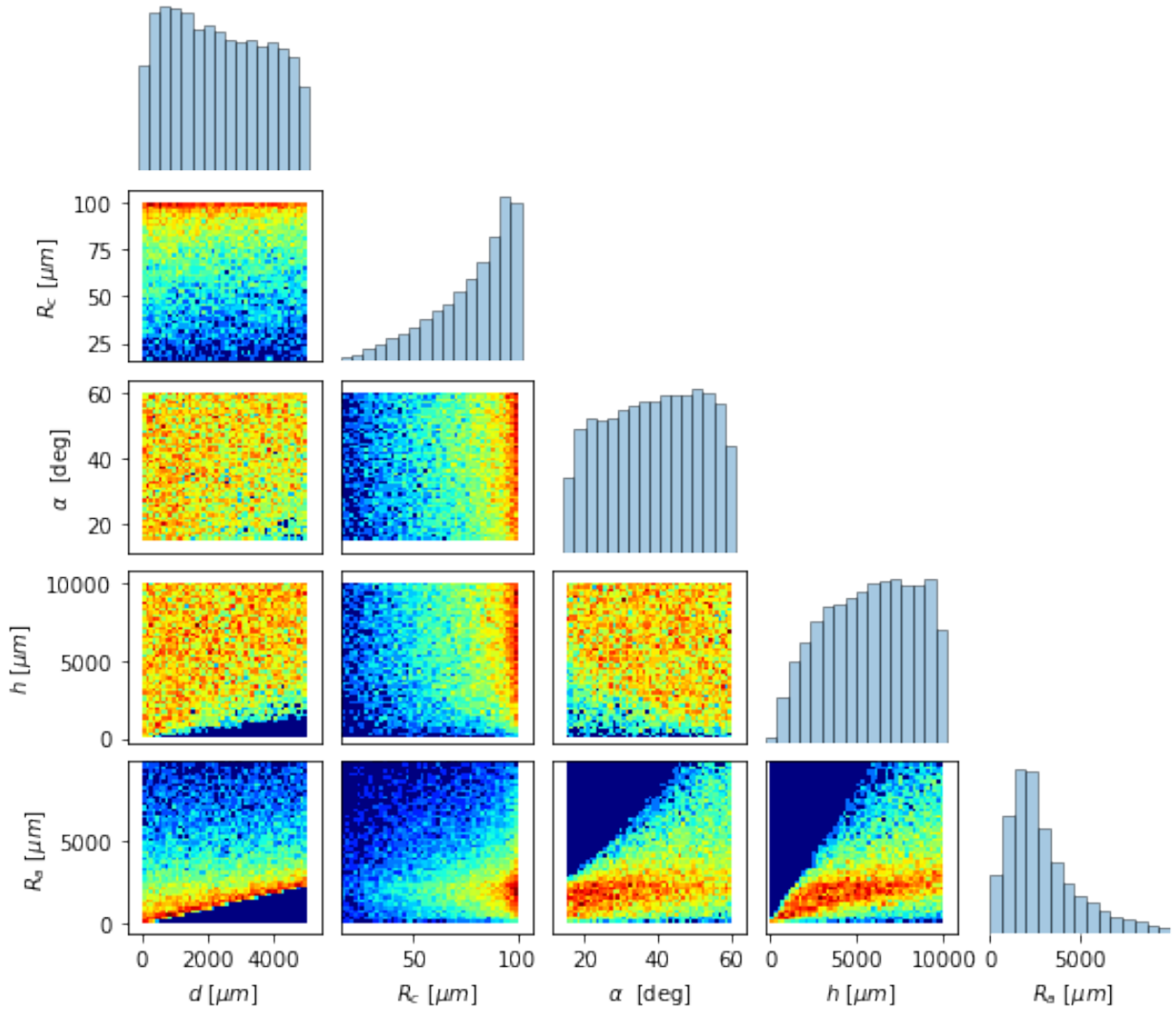


Figure 6: General geometry-based design space 1D and 2D marginals

Figure 6 shows the extent to which the feasible design space is constricted by the performance requirements. For example, the cone height in the feasible design space is in the range $h \in [0, 800]$µm while it is in the range $h \in [0, 10,000]$µm in the general design space; a 10-fold reduction in scale.

3.3 Dataset summary

It is expected that geometries that meet true performance constraints (i.e. those based on the real electric field strength and not the Martinez-Sanchez model) will still lie relatively close to the feasible design space. So the task of training the surrogate is treated as one of learning the feasible design space fully, and then augmenting its domain to the full geometry-based design space. In this study, the surrogate was trained on both datasets simultaneously. The performance of the surrogate was evaluated relative to each space separately (it is noted that the feasible space is a subset of the general space). A summary of the datasets is provided in Table 3.

Table 3: Dataset summary

Dataset	Size
Feasible performance-based	16,836
General geometry-based	16,000
Total	32,836

The full training dataset was randomly split 80/20% into training and validation sets, respectively.

4 Performance

4.1 Feasible design space

Two test sets, both of size 2000, were generated independently from the training data; one from the feasible design space and one from the general design space. The samples covered the full range of the design spaces so that the performance of the surrogate on the test sets is representative of its performance over the full space. For a given geometry sample x_i, the relative percent error of the electric field strength predictor $\hat{y}_i$ is evaluated as:

$$\text{Relative \% error} = \frac{|y_i - \hat{y}_i|}{y_i} \times 100\% \tag{17}$$

where y_i is the known true value of the electric field strength. The distribution of the relative percent errors of the surrogate over the feasible design space test set is shown in Figure 7. The distribution is normalized to an empirical probability density function (PDF).

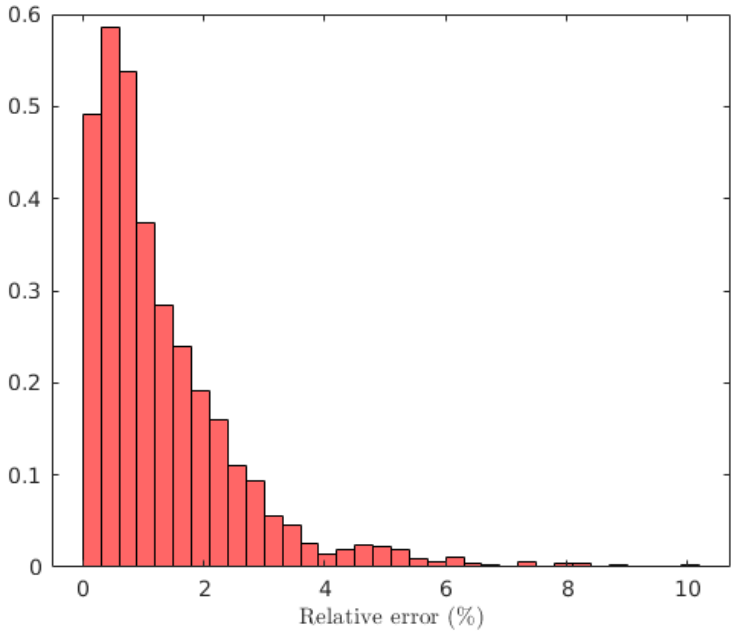


Figure 7: Relative percent error of the surrogate over the feasible design space test set

For comparison, the relative percent error of the Martinez-Sanchez approximation over the same test set is shown in Figure 8. This distribution is also normalized to an empirical PDF.

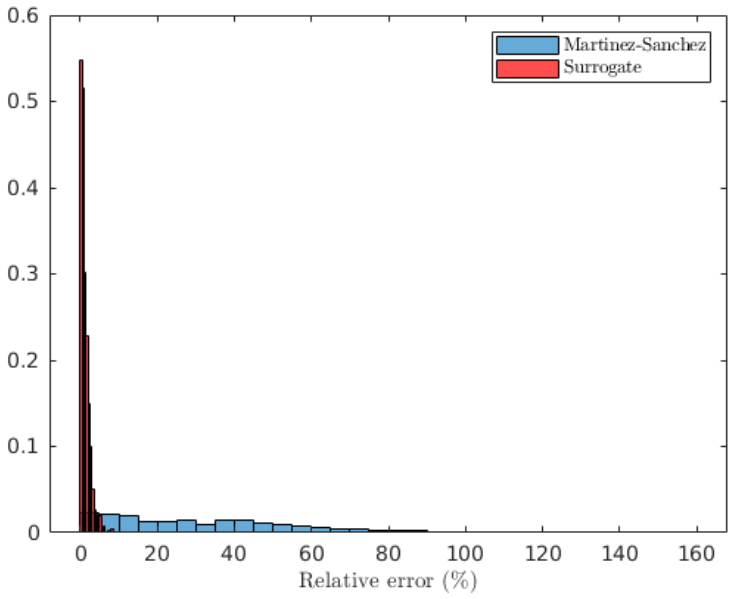


Figure 8: Relative percent error of the surrogate compared to the Martinez-Sanchez solution

The root mean squared error (RMSE) over the full feasible test set ($i = 1 \dots N$) is evaluated as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{18}$$

The statistics of these distributions are provided in Table 4, along with the RMSE in electric field prediction over the test set.

Table 4: Distribution of relative percent errors over feasible design space test set

Statistic	**Martinez-Sanchez**	**Surrogate**
Mean	33.7 %	1.30%
Median	28.7 %	0.93 %
Minimum	1.3×10^{-3} %	0.1×10^{-3} %
Maximum	158 %	10 %
RMSE	8.6×10^{6} V m^{-1}	0.5×10^{6} V m^{-1}

4.2 General design space

The same procedure as in the previous section was used to evaluate the performance of the surrogate over the general design space test set. The distribution of the relative percent errors of the surrogate

over the general test set is shown in Figure 9.

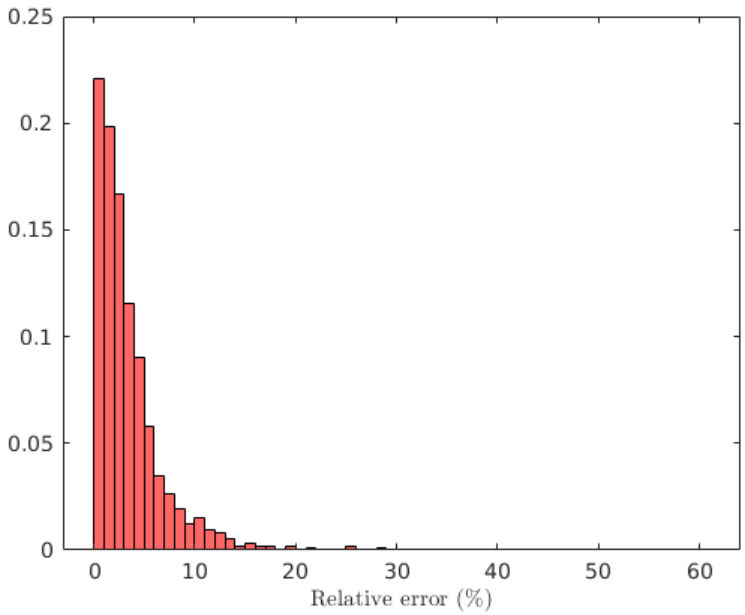


Figure 9: Relative percent error of the surrogate over the general design space test set

The relative percent error of the Martinez-Sanchez approximation over the same test set is shown in Figure 10.

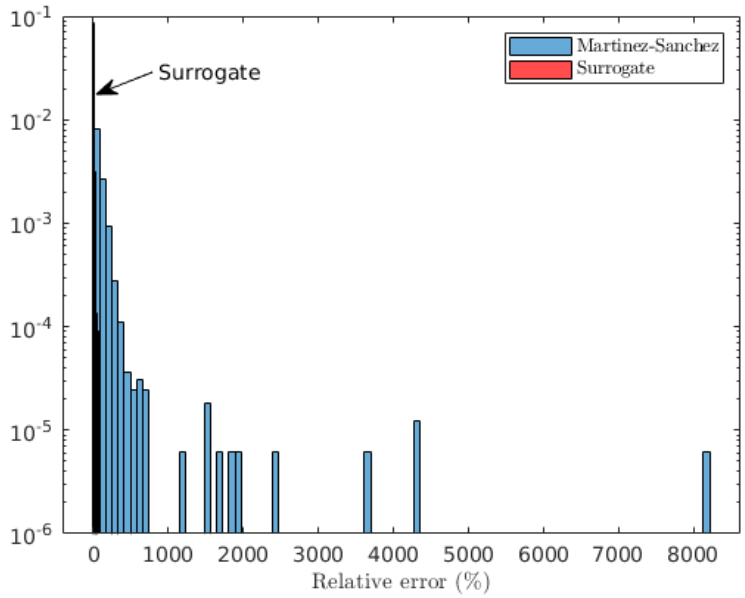


Figure 10: Relative percent error of the surrogate compared to the Martinez-Sanchez solution

The statistics of these distributions are provided in Table 5, along with the RMSE in electric field prediction over the general design space test set.

Table 5: Distribution of relative percent errors over general design space test set

Statistic	**Martinez-Sanchez**	**Surrogate**
Mean	92 %	3.56%
Median	46 %	2.46 %
Minimum	26×10^{-3} %	0.02×10^{-3} %
Maximum	8200 %	60 %
RMSE	24×10^{6} V m^{-1}	0.5×10^{6} V m^{-1}

The geometry of the emitter that caused the maximum error in the Martinez-Sanchez (MS) approximation is shown highlighted in the 1D geometry marginals in Figure 11.

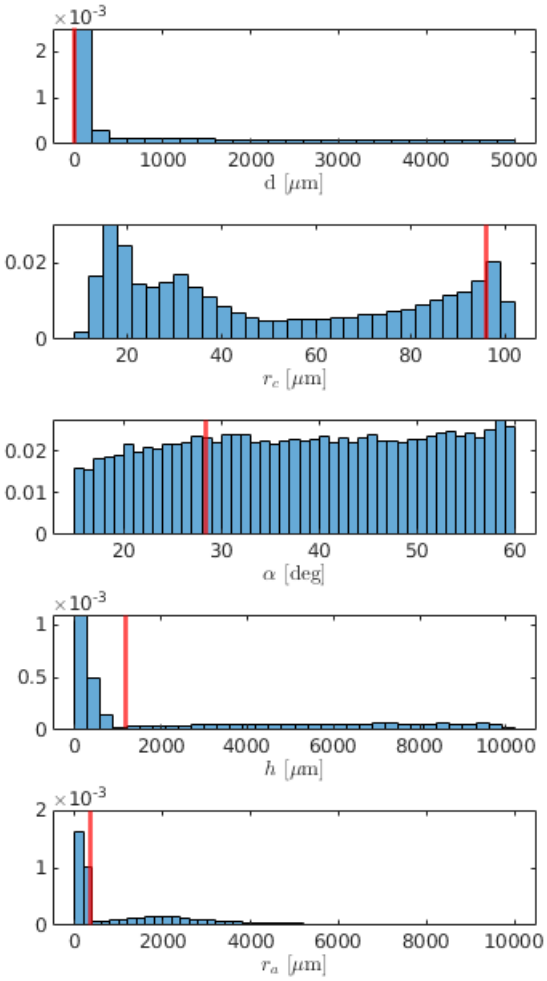


Figure 11: MS Worst-performing geometry location in the general design space

The marginals in Figure 11 are computed over the full training dataset (i.e. feasible and general design space combined). The simulation field solution for this case is shown in Figure 12.

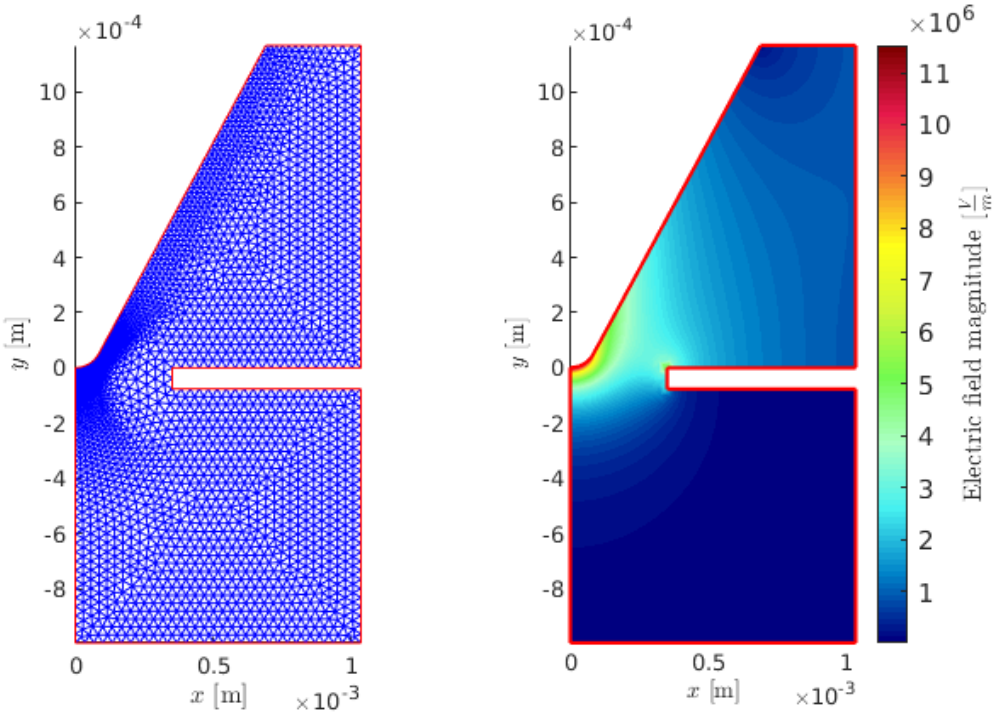


Figure 12: MS Worst-performing geometry simulation field solution

The emitter for this particular geometry is very close to the plane of the extractor grid, and the radius of the aperture is such that the emitter is nearly sticking through the aperture. It makes since then that the assumptions of the MS approximation are not met due to the absence of the extractor grid near the tip of the emitter. The MS solution is correspondingly very inaccurate relative to the true electric field. In comparison, the surrogate had a relative error of 3.0% for this case; it was trained to handle cases like this indistinguishably.

The geometry of the emitter that caused the maximum error in the surrogate is shown highlighted in the 1D geometry marginals in Figure 13.

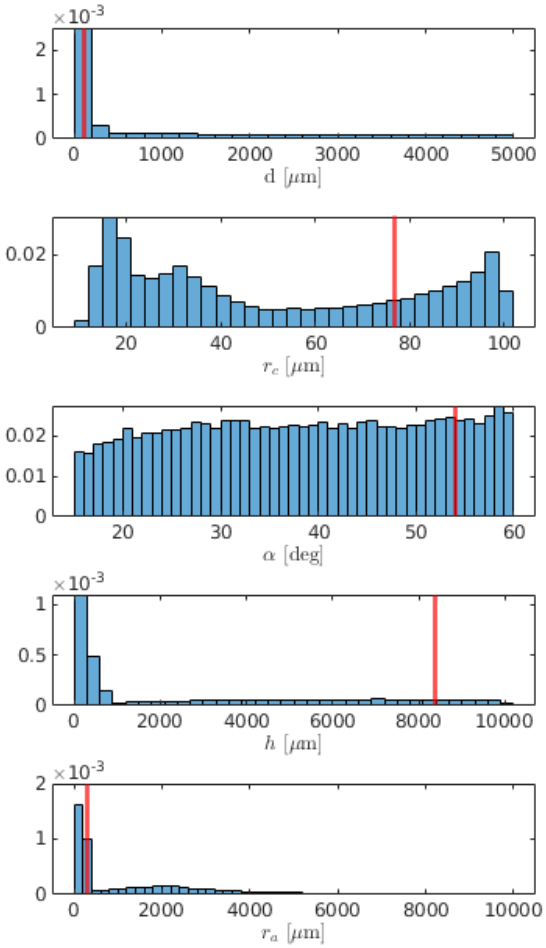


Figure 13: Surrogate worst-performing geometry location in the general design space

The simulation field solution for this case is shown in Figure 14.

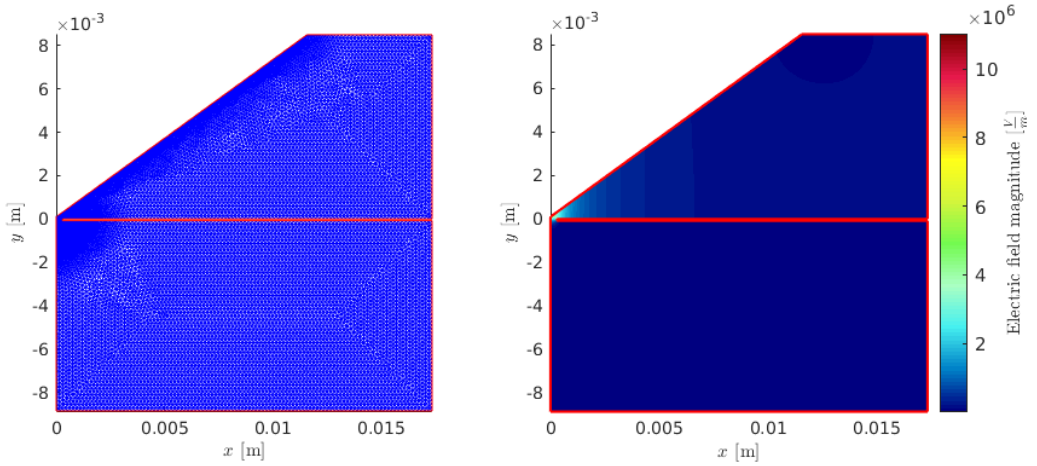


Figure 14: Surrogate worst-performing geometry simulation field solution

The emitter is noted by a sharp radius of curvature relative to the large height of the cone. As the tip radius approaches a sharper and sharper peak, the electric field strength at the tip exhibits exponentially growing behavior. The electric field is much more sensitive to small changes in geometry in this regime than it is for "more blunt" emitters. The surrogate was trained on relatively fewer geometries in this regime to be able to accurately characterize this sensitive peaking behavior. The relative error of the MS solution for this geometry was 44%, compared to the surrogate's 60%. Even at its worst performance, the surrogate is still on the same order of magnitude of error as the MS solution.

4.3 Conclusion

For the overwhelming majority of both design spaces considered here, the surrogate performs better not only more often, but to a higher degree of accuracy than the MS solution. The surrogate applies in all cases even where the assumptions of the MS solution are not met, such as for large aperture radii. The surrogate shows a $< 10\%$ relative error over the full feasible design space, and an average 3.5% relative error over the full general design space. The cases where the surrogate performs poorly are impractical designs, such as really tall and wide emitters with sharp tips, and even in these cases, its predictions are on the same order of magnitude of error as the MS solution.

The surrogate is applicable for peak electric field prediction within these error margins to any emitter geometry that meets the basic geometry constraints outlined in the sampling procedure in [1].

It is briefly noted here that the surrogate has also shown robust performance in predicting the vector component electric field distribution along the surface of the emitter, and not only just the peak electric field magnitude. Should the full electric field distribution (magnitude and direction) be useful in the future, it is possible to retrain the surrogate for this task, although I will note that this is a more difficult problem when applied to the general design space. It might be necessary to train two networks: one to learn the distribution over a single emitter, and the other to predict the parameters of the first network based on a given geometry input.

References

[1] A. Gorodetsky, C. B. Whittaker, A. Szulman, and B. Jorns, "Robust design of electrospray emitters," in *AIAA Propulsion and Energy 2021 Forum*, 2021, p. 3422.