



School of Electronic Engineering

CB54: Machine Learning Algorithms for EM Wave Scattering Problems

Appendix D: Project Design & Implementation

Anthony James McElwee

ID Number: 20211330

August 2023

MEng in Electronic and Computer Engineering

Supervised by Dr Conor Brennan

Contents

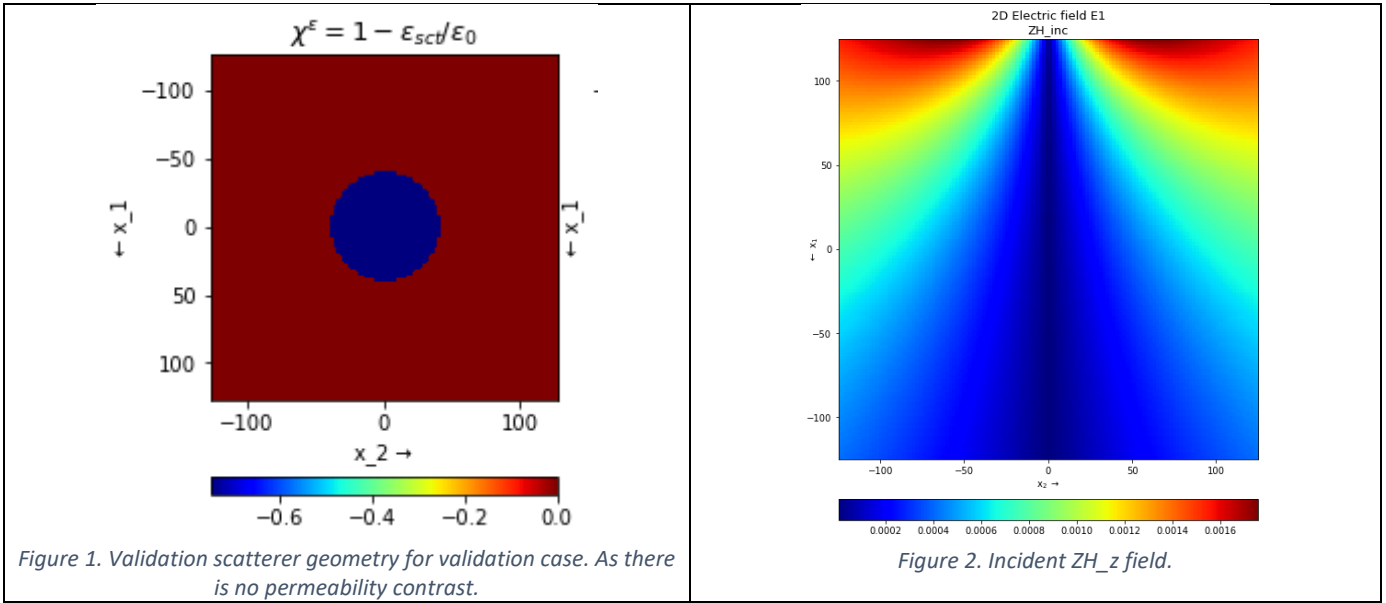
Project Design & Implementation.....	3
Code Validation.....	3
Model Architecture Description	5
SovlerEMF2 Infusion	8
Bibliography	8

Project Design & Implementation

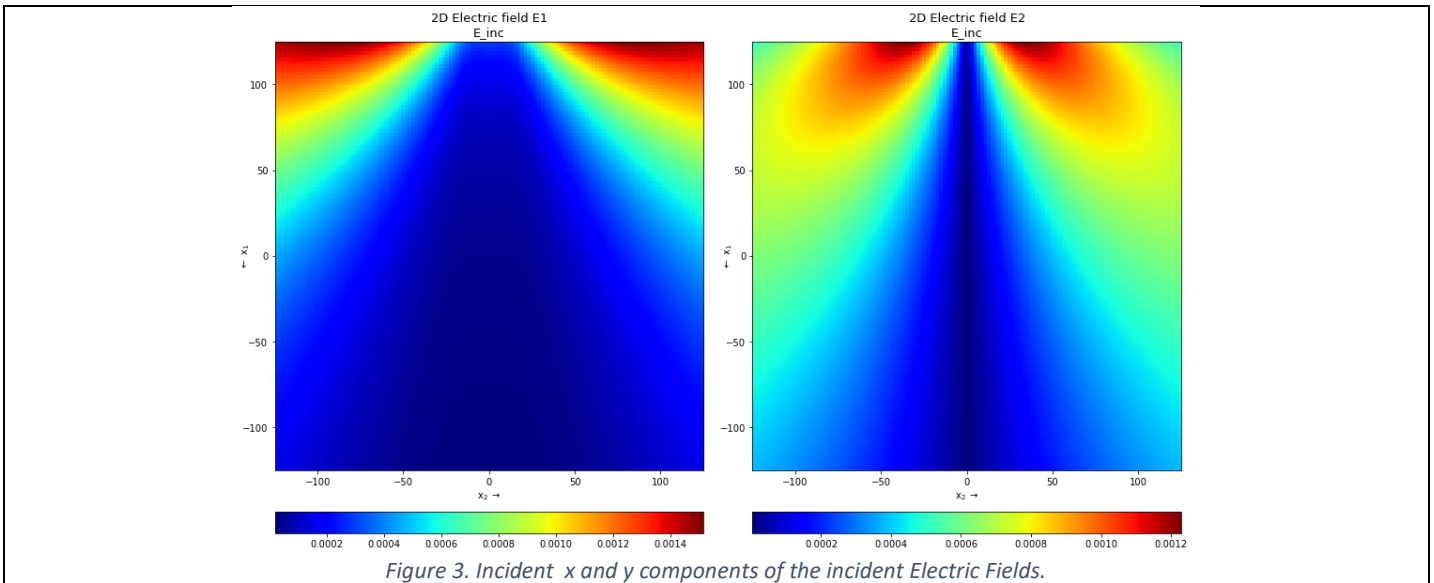
Code Validation

The Python code was validated in the same manner as the MATLAB code was validated in [1]. This code is called using the function EmsctCircle in the file custom_functions_EM.py found in the folder lib found in the project GitHub repository “<https://github.com/spookworm/CB54>”. Details of the validation code and lengthy derivation are covered in Section 3.A.1.3 of [1] and the main points with plots are briefly described below. In the code ForwardBICGSTABFFTweE.py code, by changing the variable “validation” to equal “True” the validation code will be reproduced. Ensure that the other two variables “guess_validation_answer” and “guess_model” are both set to ‘False’.

- The parameters of the simulations are loaded using the initEM function with the custom “bessel” argument enabled to indicate that the programme returns the base scatterer geometry.

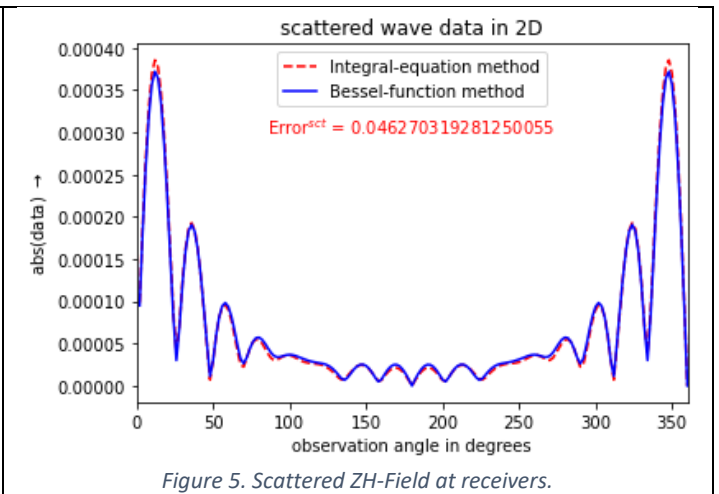
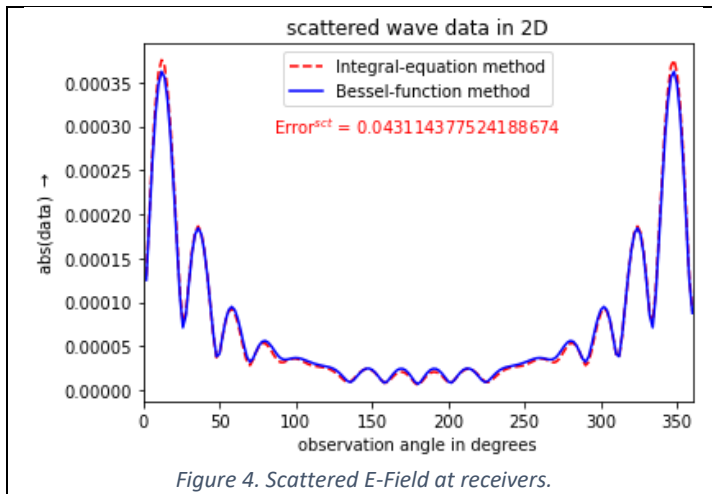


- The receiver and source emitter have their co-ordinates transformed from cartesian to polar form.

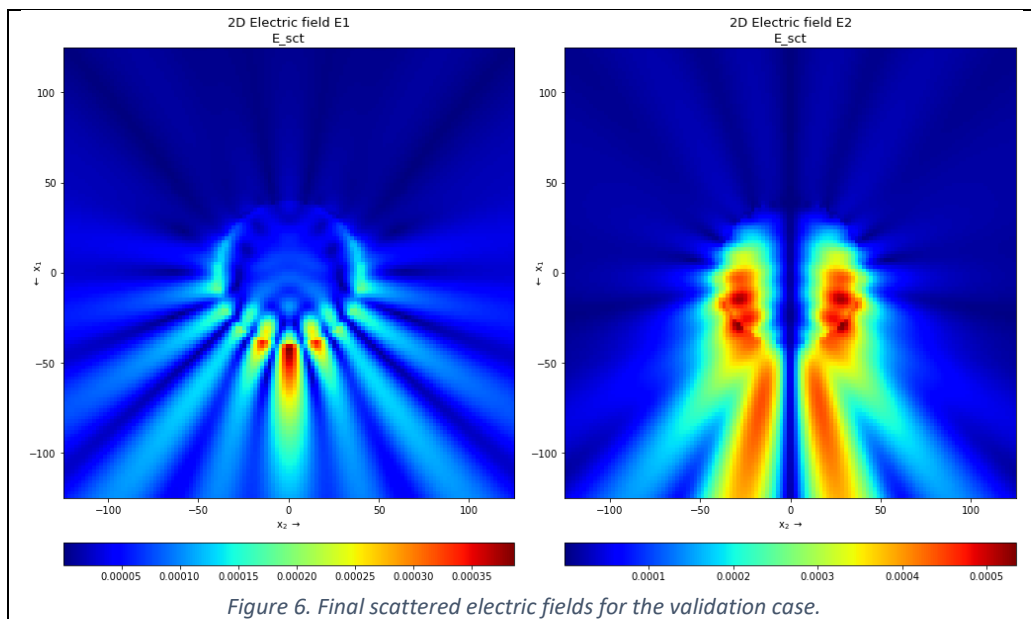


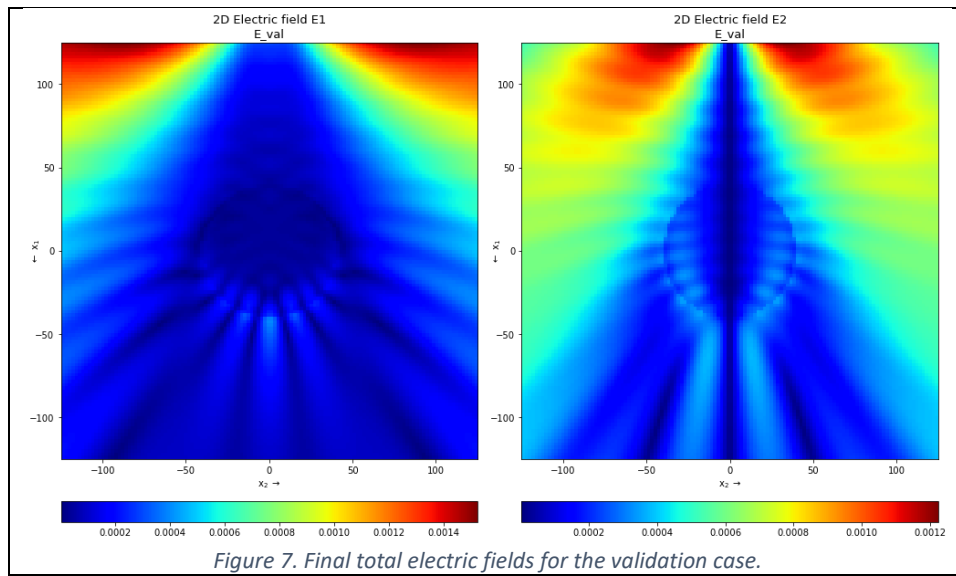
- The Bessel-Function series are calculated in a manner to reduce the required algebraic steps using a for loop with a higher number of terms originally found in [1] due to experimentation with initEM parameters during the development process. Coefficients A and B are used in determining the reflected electric field at the receivers as denoted in equations (3.A.15-16) in [1].

- The reflected electric field and incident fields are transformed from polar co-ordinates back into cartesian co-ordinates.
- The absolute difference between the magnitudes of the analytical Bessel-Function Approach and the fields produced by the Contrast-Integral Approach is calculated and printed to the plotted diagrams below.

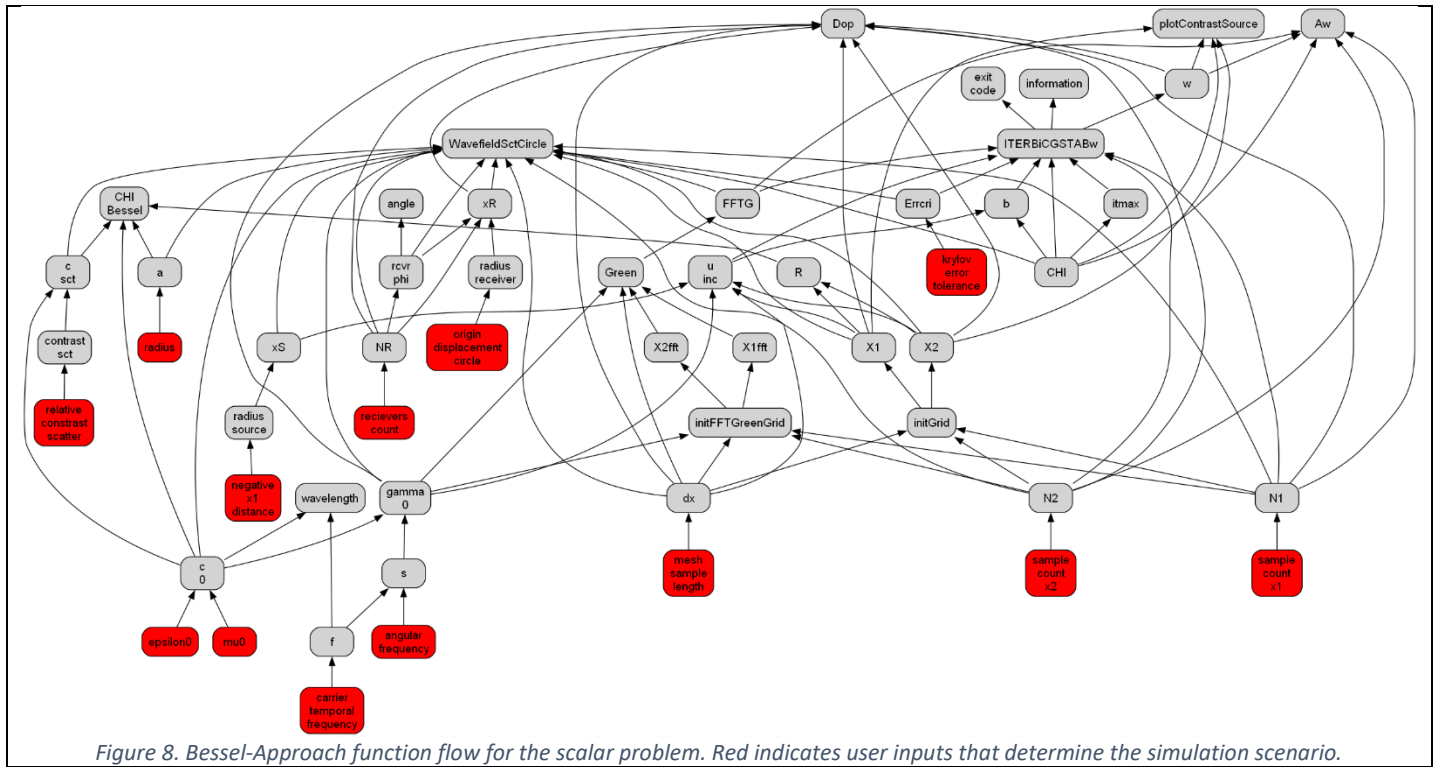


- The validation used in [1] is that of a 2D electric dipole line source so that comparison can be made more easily with the 3D case. Aside from validating the solver against an analytical solution, by reproducing this MATLAB code in Python, the code itself is validated for the input parameters.





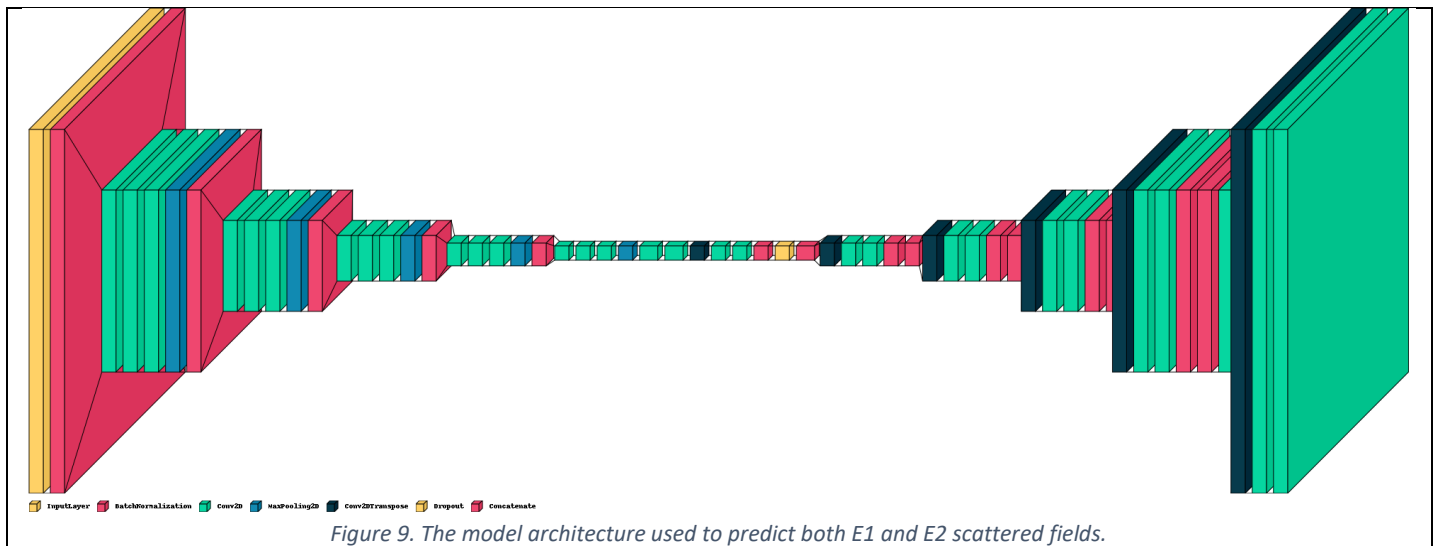
In coding the scalar code problem in Chapter 1 of [1], the student used the Python library “fn_graph” to create an illustration of the complexity of the code. While the scalar problem described is simpler the problem tackled in this project, the code flow is included below for the Bessel-Approach and the full integral solver approach.



Model Architecture Description

The deep learning model was built using the TensorFlow and Keras libraries in Python. Details of the installation versions are found in the conda environment files on the project GitHub repository at <https://github.com/spookworm/CB54>.

A simple visualisation of the layer dimensions is captured using the visualkeras Python library below. This diagram does not feature the skip connections between the encoder and decoder sides of the model.



The full detailed model architecture is visualised using the TensorFlow plot_model library below. This can be used, aside from the code itself, to reconstruct the model architecture in future experimentation. The student recognises that the image resolution suffers within the pdf format and recommends that the reader refer directly to the source “model_plot_EM.png” file located in the subfolder “doc\Project_Design_Implementation” in the GitHub repository.

SovlerEMF2 Infusion

The manner in which Prescient2DL can be used as a stand-alone emulator is obvious: the model is provided with similar information available to the Krylov Iterative Solver and produces a prediction for the target fields with a comparable or lower error than the Krylov solver provides after one iteration. Infusing the predictions into a Method-of-Moments workflow is a more open question. The student decided to go with the simplest approach of using the predicted outputs from Prescient2DL as the initial guess for the Krylov solver. There may be more complex ways to infuse deep learning into conventional methods and the reader is referred to the recent [5] for inspiration for more entry points. As the Krylov Iterative solver at the heart of SolverEMF2 is actually searching for a solution to the contrast source as opposed to the direct scattered fields, the output from the Prescient2DL models undergoes a transformation in ForwardBICGSTABFFTWE.py using the custom_functions_EM.KopE function so it can be used as an initial guess in the BICGSTAB solver. This is not evident in [1] since the code was not intended to be used to infer guesses. No initial guess was postulated in [1] so for the naïve implementation, the incident wave was taken to be the total wave.

SovlerEMF2 Krylov Solver

In the more explicit implementation of the MATLAB code in [1], the main solver is the Conjugate Gradient Krylov solver. However, in section 3.2.3.1 it is made clear that the BICGSTAB Krylov solver performs much better in terms of reduced iteration count. Translating the MATLAB code to Python for these solvers was not straight forward as custom callback functions had to be written so that the outputs from the MATLAB code were comparable to the outputs produced by the ITERBiCGSTABWE function. The Python library `scipy.sparse.linalg` was used to provide the core `bicgstab` and `LinearOperator` functions, however, they calculate the iteration steps in a different manner to the MATLAB calculation and are not easily accessible to debug. The student has formulated the code so that the outputs from the Python code match those produced by MATLAB in case development is required due to errata in [1] or other solvers or adaptations are desired in future work.

Bibliography

- [1] P. M. van den Berg, *Forward and inverse scattering algorithms based on contrast source integral equations*. Hoboken, NJ: Wiley, 2020. [Online]. Available: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781119741602>
- [2] Q. Ren, Y. Wang, Y. Li, and S. Qi, *Sophisticated Electromagnetic Forward Scattering Solver via Deep Learning*. Singapore: Springer, 2022. doi: 10.1007/978-981-16-6261-4.
- [3] C. Brennan and K. McGuinness, "Site-specific Deep Learning Path Loss Models based on the Method of Moments." arXiv, Feb. 02, 2023. doi: 10.48550/arXiv.2302.01052.
- [4] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*, Second edition. Beijing [China] ; Sebastopol, CA: O'Reilly Media, Inc, 2019.
- [5] P. Hennig, M. A. Osborne, and H. Kersting, *Probabilistic numerics: computation as machine learning*. Cambridge New York, NY Melbourne New Delhi Singapore: Cambridge University Press, 2022.