## Help Required

* CB MATLAB code for VEFIE, cannot see the derivation matching, am I missing something? Please explain.
* Is EFVIE (VEFIE) the same as Contrast Source Integral Equations in [1]? If yes then is [1] realised in the MATLAB code using MoM? This question is a time limitation one, I would obviously work this out but maybe you can short cut the answer?

## My Mistakes

* Initially I thought I was trying to build a ML model to challenge conventional methods. This would have led to the approach of trying to establish governing equations using ML.
* Then I thought that the aim was to accelerate a conventional method, which is still relevant, but I didn’t understand the bottlenecks of the existing conventional method. There appears to be three bottlenecks in generating the solution of the linear system to solve the integral:
  + the sampling rate required for number of cells per wavelength (This might be accelerated using idea discussed later. CB, how is the number of points per wavelength analytically derived? Does such a bound exist? (beyond scope but machine learning the lower bound might be useful if not analytical?));
  + the time and memory required to construct the Ax=b matrices (This one is accelerated by FFT);
  + the convergence of the solver (My assumption was that the ML was trying to target this as the code [1] had a significant resource expense here according to the profiler. If this really is the bottleneck when problems are at scale then creating a wrapper is still fine project. It may be superior to MC idea for sampling rate discussed later.);
* Show CB the image “disc\_per\_lambda\_25.PNG” in the tbs folder.

## Actual Project

* The actual idea is to build a machine learning emulator (surrogate model) that can do what conventional methods do with tiny computational cost over a defined range of input parameters. BUT this could be considered a recasting of the forward problem as an inverse one since Unet is reconstructing/denoising the partial information given at input stage. It’s a forward regression but can be reformulated as inputs are provided with some solution data to an inverse problem of hole filling, an inverse-problem. The inverse problem for scattering gives rise to ill-posed Fredholm equations of the First Kind, (just in case that matters later for DL).
* For example, such an emulator would assist in the early stages of a design project.
  + Literature on this topic exists [2], [3]. (just found this morning! See google search terms: “build emulator that converges to unique solution”, “build machine learning emulator”)
  + An issue with this approach is that there is no convergence check or error estimator when running blind ML approach. These checks exist in MoM based on the iterative solver used and accuracy inputs.
* My idea: Build an emulator for early design stages but construct it in a way where, as more allowed computational time is allowed, the scene converges to the solution that would have been found using conventional methods. This is inspired by rendering approaches (since rendering equation is Fredholm 2nd Kind too and this area has had intense research NVIDIA TwoMinutePapers recently) and <https://nljones.github.io/Accelerad/index.html> .
  + My approach is that MC importance sampling (or updates from the iterative solver) can be used to populate a repeatedly updated matrix that is fed into a DL model used to reconstruct/denoise the field values. The DL then produces slowly improving solution output. (Show CB your Accelerad Dublin showcase here.)
  + Since samples are from the forward integral (or the update from the iterative solver), a unique solution must exist.
  + The forward problem will eventually converge using MCMC but can be terminated prematurely with a guess depending on designer requirements (while convergence will be faster through guidance from the biased DL inference for the iterative solver update).
  + The biased DL inference is based-off an ill-posed inverse problem. If the emulator just provides one reconstruction, as in the CBKMG paper, then the whole project is just a forward problem recast as an inverse problem where the partial information is the input source, geometry, materials. You could cast it as a sort of Bayesian problem where a generic 05 is put in as a scaled guess for final field. If some final field data is provided as the iterations increase this “prior” is updated and the model takes that into account as it has been trained using a loss function to handle that input type?
  + If the construction of the linear system matrices is still a bottleneck then drop the iterative wrapper technique and look at [4] for improving the MC sampling approach using ray-tracing in combination with MoM.
  + If lambda/10 is a rule of thumb (criterion), can this be reduced using the ideas of importance sampling for boundary element methods etc. Try to calculate the solution on a coarse grid with less samples and then use model to perform super-resolution to reconstruct to desired resolution? The ML reconstruction model would need to be trained on higher resolution training data than required at the output. We want high convergence rates for low discretization rates.
  + Can the samples be considered a multi-dimensional array father than a refined grid? See yellow postit note drawing. Every iteration of samples is another summed sample…so input size doesn’t change but sample level increases?
  + In itself, MC is worth pursuing as it can be used to generate large numbers of training samples from a single simulation to train physics-informed loss function [4], [5].
* For motivation of developing emulator for design support, see “Subjects with access to real-time simulation feedback tested more design options, reported higher confidence in design performance and increased satisfaction with the design task, and produced better-performing final designs with respect to spatial daylight autonomy and enhanced simplified daylight glare probability” [6].
* For examples of SOTA see “biased image reconstruction using deep learning” and “unet for image reconstruction”.

## Bibliography

[1] P. M. van den Berg, *Forward and inverse scattering algorithms based on contrast source integral equations*. Hoboken, NJ: Wiley, 2020.

[2] M. F. Kasim *et al.*, “Building high accuracy emulators for scientific simulations with deep neural architecture search,” *Mach. Learn.: Sci. Technol.*, vol. 3, no. 1, p. 015013, Dec. 2021, doi: 10.1088/2632-2153/ac3ffa.

[3] J. J. Thiagarajan *et al.*, “Designing accurate emulators for scientific processes using calibration-driven deep models,” *Nat Commun*, vol. 11, no. 1, Art. no. 1, Nov. 2020, doi: 10.1038/s41467-020-19448-8.

[4] C. Delgado and M. F. Cátedra, “Combination of ray-tracing and the method of moments for electromagnetic radiation analysis using reduced meshes,” *J. Comput. Phys.*, vol. 361, no. C, pp. 412–423, May 2018, doi: 10.1016/j.jcp.2018.01.040.

[5] Y. Guan, T. Fang, D. Zhang, and C. Jin, “Solving Fredholm Integral Equations Using Deep Learning,” *Int. J. Appl. Comput. Math*, vol. 8, no. 2, p. 87, Mar. 2022, doi: 10.1007/s40819-022-01288-3.

[6] N. L. Jones and C. F. Reinhart, “Effects of real-time simulation feedback on design for visual comfort,” *Journal of Building Performance Simulation*, vol. 12, no. 3, pp. 343–361, May 2019, doi: 10.1080/19401493.2018.1449889.