Compared to traditional solvers: Classical methods such as FEM/FDM are designed to solve a PDE once for a given parameter. In contrast, our method approximates the solution operator of a PDE i.e. the mapping from parameter to the solution. The goal of this work is to propose a data-driven surrogate model which offers flexibility in downstream applications as it can produce accurate approximations cheaply and quickly, rather than propose a method for solving PDEs. Our method is consistent in function space i.e. the error is independent of the resolution of the input or output functions which is a crucial property for application that very few deep learning methods posses.

- Traditional solvers: solve one instance of the equation at a time; require the explicit form of the PDE; have a speed-accuracy trade-off based on resolution: accurate on fine grids, less accurate on coarse grids.
- Neural operator: learn a family of equations from data; don't need the explicit knowledge of the equation; much faster to evaluate than any classical method; no training needed for new equations; error rate is consistent with resolution.

Example: inverse problem of 1-d Burgers' equation: Suppose we wish to use a function-space MCMC method to find the Bayesian solution of the inverse problem associated with the 1-d Burgers' equation: given a noisy solution at time 1, we wish to sample the distribution of possible initial conditions. This requires tens of thousands of evaluations of the equation with different initial conditions. Using the classical approach with which we generate our data (details in Appendix), each forward run takes around 2 seconds, which amounts to **28 hours** of computation time for 50,000 MCMC steps. Our approach approximates each solution in only 0.01 seconds, amounting to **8 minutes** of computation time. The computational savings are much further embellished when considering more complex PDEs.

Review 1: Thank you for the detailed review and helpful suggestions. Methods like those in Chebfun are not designed for parametric equations and are much slower compared to our method; we are happy to add a demonstrative example. Furthermore, conservation properties are kept so long as they are in the data; we are happy to add an example.

- **Technicality:** While we agree that certain sections are more mathematically formal than others, we have done this in order to balance reader comprehension with page-limit requirements. We believe it is important to concretely way out our problem of interest while not so important to give the correct function space every time.
- **Figure:** We appreciate your figure suggestion and would be happy to use it with your permission.
- **Graphs:** While we agree that our methodology can be stated without graphs or GNNs, they are currently the only efficient deep learning tool for implementing it. Furthermore graphs allow for a straight forward extension to domains that are manifolds, a direction we wish to pursue in the future.
- **Complexity:** We measure complexity in terms of the sparsity of K as this is what affects all computations. Originally this complexity is $O(n^2)$. Truncating the integration reduces it to $O(n^2 r^d)$ $(r < 1)$, while the Nyström approximation brings this to $O(m^2 r^d)$ $(m \ll n)$. These approximations are indeed used throughout the V-cycle, a point we will clarify in our revision. (line 140. - 152.)
- **Presentation:** Note that $u(x)$ cannot be identically zero, please see the cited theorem. Furthermore we treat $f(x)$ as fixed and therefore omit it from the architecture. While we agree that the approximation from (2) to (5) is not precise, neither is neural network design. We simply use it as guiding principle, combining it with deep learning intuition such as the addition of the W matrix which makes learning K easier.

Review 2: Thank you for your comments. Please note that our work considers both 1-d and 2-d problems (Section 4).

- **MgNet:** While MgNet is an interesting work, it is substantially different from ours. In particular, they employ the multigrid method which is based on a residual correction while we use the multipole method which is based on a decomposition of the integral kernel. Furthermore our graph-based convolutions allow our method to achieve consistent error for any discretization while continuous convolutions are tied to a particular grid due to their local parameterization. Lastly, MgNet is only applied to image classification tasks (Cifar) while we do function to function regression, arguably, a much more general task. We would be happy to add a citation and will try to include a numerical comparison in the final version, but unfortunately their code is currently not online.
- **Other methods:** While methods such as PINNs and Deep ritz are interesting, they are not suitable for our goals. They both require solving a non-convex optimization problem for each parameter of interest which is slow and costly (weeks of computation for the above inverse problem). Such methods are useful when the underlying physical domain is high dimensional and FEM becomes infeasible, but this is not a setting our work considers. In the parametric setting, we are not aware of any methods with better accuracy than ours.

Review 3: Thank you for your nice suggestions. We designed our method to approximate parametric PDEs and therefore never tested it on standard graph datasets such as Cora. While this is an interesting future direction as quickly processing global information on large graphs is crucial in applications, it is well beyond the scope of the current work.