Entangled Basis Finite Element Method PDE solver for Quantum Computer

Abhijatmedhi Chotrattanapituk

Quantum Measurement Group, MIT, Cambridge, MA, USA

Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA

(Dated: October 20, 2023)

1. INTRODUCTION

2. FINITE ELEMENT METHOD

Finite element method (FEM) is a class of popular numerical method for solving differential equation (DE). It is a powerful tool that used in a lot of branches in mathematics, science, and engineering, including material design structural analysis, all kinds of transport (thermal, fluid, mass), electromagnetism, biomechanics, automotive/aerospace, manufacturing, etc.

The core concept behind FEM is similar to other numerical methods in the sense that one do a certain discretization of the DE, whether in space, time, or reciprocal domain, in order to reduce the calculation degree of freedom, from infinite, into a managable finite value that still gives resonable results. Here, we will pitch FEM mostly against finite difference method (FDM) since it is an equally popular method with long history of its development. Also, it is the most straight forward method when one want to discretize a DE. In fact, part of the method we are going to introduce will utilize FDM.

In a generic FEM, one breakdowns (discretizes) the domain of the problem in to small chunks called elements. By assuming that, if the elements are small enough, the DE's solution in a particular element can be described by a simple function. This process is similar to the problem linearization. In fact, many FEM software directly use a variation of linear function as the function for each element. Let's assume for generalization that, for each element k, we have a set of basis functions, $\{\phi_{k,p}|p \in \{0, 1, 2, ..., P_k - 1\}\}$ where P_k is the number of basis functions in that element. There are many possible basis sets one can use like polynomial functions, Fourier basis, harmonic oscillator basis, etc. However, in this work, we will focus on the Fourier basis. Other basis can be directly substituted in with a simple change in some core components. However, for visualization, purpose, we will use a variant of linear function as a demonstration for this section. There are mainly two methods for FEM discretization: non-overlapping where basis functions only describe a particular element, and overlapping where basis functions span over multiple elements. The former is more straight forward, but can have issue at the elements' edges in the case where the chosen basis cannot describe the solution well at the edges. While the later doesn't have this issue, its calculation is more complex.

In order to solve a DE with FEM, we want the result that can best approximate the true solution, i.e., the result must approximately satisfy the DE as well. To evaluate this, it is required to check the DE satisfaction for every single point in the domain. However, in each element point of view, it is enough for the solution to satisfy the DE's weak form which is the DE's domain integration of that element.Since, in an element, the solution is a linear combination of the basis functions, it is convenient to get the derivatives with the linear combination of the derivatives of basis functions. Hence, it is also straight forward to get the domain integration for calculating the DE's weak form.

There are a few advantages of using FEM over FDM especially in spatial axes. While FDM requires the discretization of parameters into parallel grids, FEM has no such requirements, and each element can be any irregular shape. Though, it is common to use polygons especially triangle. This difference causes FEM to be a better choice of DE solver in the case where simulation boundary cannot be easily handle with parallel grid, and when there are singularity (defects, grain boundaries, material changes) points that require denser grid to handle abrupt changes of the functions.

3. TENSOR NETWORK

Tensor

4. TARGET EQUATION

In this work, we are considering a class of partial differential equation (PDE) that is linear in time (LT-PDE), as shown in equation 4.1, of a tensor-valued function, $\tilde{\mathbf{u}}(t, \mathbf{q})$, of time, $t \in \mathbb{R}$, and D-dimensional space, $\mathbf{q} \in \mathbb{R}^D$, where we use boldface variables to indicate tensors:

$$\tilde{\mathbf{h}}(t, \mathbf{q}, \tilde{\mathbf{u}}, \mathbf{D}_\mathbf{q}\tilde{\mathbf{u}}, \mathbf{D}_\mathbf{q}^2\tilde{\mathbf{u}}, \cdots) + \tilde{\mathbf{g}}_1(t, \mathbf{q})\mathrm{D}_t\tilde{\mathbf{u}} + \tilde{\mathbf{g}}_2(t, \mathbf{q})\mathrm{D}_t^2\tilde{\mathbf{u}} + \cdots = 0. \tag{4.1}$$

Here, D_t is time derivative operator, $\mathbf{D}_\mathbf{q}$ is spatial derivative tensor operator, $\tilde{\mathbf{h}}(\cdot)$ and $\tilde{\mathbf{g}}(\cdot)$'s are some functions, and the multiplication (and powers) between two tensors is their tensor product. Also, let the LT-PDE subjects to initial value

$$\tilde{\mathbf{u}}(t_0, \mathbf{q}) = \tilde{\mathbf{u}}^{(0)}(\mathbf{q}). \tag{4.2}$$

It is common to treat a high-order linear ordinary differential equation (ODE) as multiple coupled first-order ODEs. We will apply the similar idea to the time derivative parts of our LT-PDE. Let's define the τ-order time derivative of $\tilde{\mathbf{u}}$ as $\mathbf{u}_\tau$. Without loss of generality, let the LT-PDE has maximum time derivative order of T. Hence, equation 4.1 can be rewritten as

$$\left(\mathrm{D}_t + \begin{bmatrix} 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \frac{\tilde{\mathbf{g}}_1}{\tilde{\mathbf{g}}_T} & \frac{\tilde{\mathbf{g}}_2}{\tilde{\mathbf{g}}_T} & \cdots & \frac{\tilde{\mathbf{g}}_{T-1}}{\tilde{\mathbf{g}}_T} \end{bmatrix}\right) \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{T-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \frac{\tilde{\mathbf{h}}(t, \mathbf{q}, \mathbf{u}_0, \mathbf{D}_\mathbf{q}\mathbf{u}_0, \mathbf{D}_\mathbf{q}^2\mathbf{u}_0, \cdots)}{\tilde{\mathbf{g}}_T} \end{bmatrix} = 0. \tag{4.3}$$

This means that, if we define $\mathbf{u}(t, \mathbf{q})$ as the tensor of $\mathbf{u}_\tau$'s, and $\mathbf{h}(t, \mathbf{q}, \mathbf{u}, \mathbf{D}_\mathbf{q}\mathbf{u}, \mathbf{D}_\mathbf{q}^2\mathbf{u}, \cdots)$ as the terms in equation 4.3 that do not contain D_t, then the equation can be simplified to

$$\mathrm{D}_t\mathbf{u} + \mathbf{h}(t, \mathbf{q}, \mathbf{u}, \mathbf{D}_\mathbf{q}\mathbf{u}, \mathbf{D}_\mathbf{q}^2\mathbf{u}, \cdots) = 0. \tag{4.4}$$

Hence, we will assume, for the rest of the work that the LT-PDE contains only the first time derivative. We will also further restrict the tensor-valued function $\mathbf{h}(\cdot)$ to be analytic (ALT-PDE), i.e., it can be represented as a converging series:

$$\mathbf{h}(t, \mathbf{q}, \mathbf{u}, \mathbf{D}_\mathbf{q}\mathbf{u}, \mathbf{D}_\mathbf{q}^2\mathbf{u}, \cdots) = \sum_{i_0, i_1, i_2, \cdots \in \mathbb{Z}_{0+}} \left(\mathbf{h}_{i_0, i_1, i_2, \cdots}(t, \mathbf{q}) \prod_{\chi \in \mathbb{Z}_{0+}} \left(\mathbf{D}_\mathbf{q}^\chi \mathbf{u}\right)^{i_\chi} \right), \tag{4.5}$$

subjected to initial tensor-valued function

$$\mathbf{u}(t_0, \mathbf{q}) = \mathbf{u}^{(0)}(\mathbf{q}). \tag{4.6}$$

To set up for numerical calculation, we will discretize time for this section while the space discretization will be handle in the next section. Since the ALT-PDE is already framed in the context of initial value problem. We will adopt the common method where we will treat the time evolution part with FDM. Let's discretize equation 4.4 into times $\{t_n | n \in \mathbb{Z}_{0+}\}$ with the function $\mathbf{u}$ defined at each time as

$$\mathbf{u}^{(n)}(\mathbf{q}) = \mathbf{u}(t_n, \mathbf{q}). \tag{4.7}$$

Also, define time steps as $\{\Delta_n = t_n - t_{n-1} | n \in \mathbb{Z}_+\}$. Here, we will use the first order method, i.e., Euler method, but it can be easily generalized to the higher one, e.g., Crank-Nicolson

(second order), Runge-Kutta (forth order) methods, etc. With Euler method, equation 4.4 can be written as two possible schemes:

$$\mathbf{u}^{(n+1)} - \mathbf{u}^{(n)} + \Delta_{n+1}\mathbf{h}(t_n, \mathbf{q}, \mathbf{u}^{(n)}, \mathbf{D_q u}^{(n)}, \cdots) = 0, \tag{4.8}$$

and

$$\mathbf{u}^{(n+1)} - \mathbf{u}^{(n)} + \Delta_{n+1}\mathbf{h}(t_{n+1}, \mathbf{q}, \mathbf{u}^{(n+1)}, \mathbf{D_q u}^{(n+1)}, \cdots) = 0, \tag{4.9}$$

where we implicitly omit the spatial dependence of $\mathbf{u}$ for neatness. These two are called explicit, and implicit time evolution schemes, respectively. While both of them are equivalent at the limit $\Delta \to 0$, the first scheme can be unstable, and gives divergence if the number of time steps, or step sizes used are too large. Hence, for the remaining parts of this work we will only consider the implicit scheme method, i.e., equation 4.9. In other words, we are solving ALT-PDE, for the time evolution part, with induction method from the initial value given by equation 4.6 such that, given the solution at time t_n, $\mathbf{u}^{(n)}$, we solve for solution at time t_{n+1}, $\mathbf{u}^{(n+1)}$, by tuning the function until it satisfies equation 4.9. However, to make sure that such tuning is possible, we will modify the target equation into a convex optimization problem by introducing the least square error (LSE),

$$\mathrm{LSE}_{n+1}(\mathbf{u}^{(n+1)}) = \int \left|\mathbf{u}^{(n+1)} - \mathbf{u}^{(n)} + \Delta_{n+1}\mathbf{h}(t_{n+1}, \mathbf{q}, \mathbf{u}^{(n+1)}, \mathbf{D_q u}^{(n+1)}, \cdots)\right|^2 d^D\mathbf{q} \tag{4.10}$$

as the optimization target.

5. ENTANGLED BASIS REPRESENTATION

From the previous section, in order to solve the ALT-PDE at each time step, one need to do convex optimization of the target function given by equation 4.10. We will apply the FEM introduced before as the main method to do the optimization. In this section, we are going to introduce a basis representation that is entangled which is not common in a generic FEM.

Consider two adjacent elements in FEM. Let's further assume that we are only considering overlap element cases with more than one basis function for each element.

6. TENSOR OPTIMIZATION

With the entangled basis representation introduced in the previous section, we are now applying such concept to the optimization of our target function, equation 4.10.

7. MATRIX PRODUCT STATE (MPS) REPRESENTATION

To show some concrete example, in this section, we are going to specialize the problem for the case where we already got efficient way (algorithm) to do TN optimization.

8. IMPLEMENTATION