Research Memo 3

To: Rachel N. Slaybaugh

From: Weixiong Zheng

Subject: Code `XTrans` progress

Date: July 3, 2017

Abstract

In this report, we will describe current progress of the research-purpose code originally designed for DG-even parity while later changed to generalized steady-state neutronics code.

1 My buddy: you gotta have a name for it.

When meeting with Vincent M. Laboure at ANS Annual and talking about the fact I am writing my own code, Vincent suggested I name it instead of using "my code".

It was designed for DFEM-even parity, it is rather changed/being changed to a framework allowing developer to plug in new methods without being bothered by tremendous amount of specific gadgets from the non-STL library `deal.II`. So I think, indeed, we need a name for this.

I actually like the name of X-division in LANL. So I later started to use the name `XTrans`. Now I think I eventually accomplish a big part of the whole thing, naming.

2 Where we are: what we have and what we have ever tested

2.1 Parallelism

After messing with `deal.II` MPI-related objects and utilizing `PETSc` data types, the code be run in both serial mode and parallel mode through MPI. It has been tested using 1, 2 and 4 cores on my Mac and shows the speedup effects when using more cores.

2.2 Problem scale on personal computers.

For research purpose, I imagine we will have up to a few to hundreds of millions of DoFs per problem. On my Mac, a problem with around 21 millions DoFs was tested using 4 cores. It consumed about 10 GB RAMs, which actually gave me a little surprise as I expected worse memory consumption. So I would assume roughly around the scale we can run on one single personal computer.

Maybe we should think about install it on clusters.

2.3 Even parity

Current code only implements even-parity equation as the transport solver. In terms of spatial discretization, CFEM and DFEM are both implemented. In fact, `TransportBase<dim>` allows for easy implementation of CFEM and DFEM if you are clear about the mathematical formulations.

For future, I just assigned I and Marrisa to implement CFEM-SAAF, which I prefer over even parity.

2.4 Reflective boundary

Functionalities for reflective boundary has been enabled. Basically, the index are precomputed for each direction per boundary. The results thereafter are utilized to form a Hash table from `std::map`.

For even parity, reflective boundary is difficult for direct implementation in conventional way. However, through using the explicit reflective boundary condition, reflective boundary is realized in `XTrans`. The trade-off is to use BiCGStab instead of CG for solving the whole system.

3 Illustrations of the way we do it.

As I talked to Rachel, the code would take in both first-order and second-order forms.

With DFEM, we will be able to solve

$$\mathcal{L}\psi = q_{\mathrm{t}}. \tag{1}$$

Implementation will be some students deriving a class with providing bilinear forms.

On the other hand, with current implementation of even parity (with both CFEM and DFEM), or even later CFEM-SAAF and CFEM-LS (if interested), it is actually equivalent to define a symmetrization operator $\mathrm{Sym}(\cdot)$ s.t. we are solving a symmetrized transport equation:

$$\mathrm{Sym}(\mathcal{L}\psi = q_{\mathrm{t}}). \tag{2}$$

For instance, LS equation is

$$\mathcal{L}^{\dagger}\mathcal{L}\psi = \mathcal{L}^{\dagger}q_{\mathrm{t}}, \tag{3}$$

where SAAF is equivalent to

$$\mathcal{L}^{\dagger}\frac{1}{\sigma_{\mathrm{t}}}\mathcal{L}\psi = \mathcal{L}^{\dagger}\frac{1}{\sigma_{\mathrm{t}}}q_{\mathrm{t}}, \tag{4}$$

4 What are the limitations?

`XTrans` is roughly in a good shape, at least on the right track to get a good shape. However, we should be aware of the limitations existing.

4.1 You cannot do cell-wise spatial sweep.

We will have a first-order solver, yet, not based on cell-wise sweep. Domain-decomposition based parallel sweep is actually not natural to implement based on `deal.II`. However, one could utilize `deal.II` functionalities in MPI to enable this. It's possible, but needs to bypass a lot of `deal.II` things. The advantage I can see is that one does not need to worry about DFEM-related low-level work (basis functions, etc). And MPI wrapper functions are actually looking more friendly than typical MPI functions. The worst case would not be worse than writing a first order sweep code from scratch.

For now, the first order solver will go into the FEM track, that is assemble the global matrix per direction per group and inverse it.

4.2 Difficulty in advanced algorithms.

The first thing I can think of is Matrix-Free implementation. Well, it would not be a problem if we restrict ourselves to "research purpose" code with solve small-to-intermediate scale problems with linear finite element. It would not be hard to imagine that one would have memory issues when using high-order elements with 3D whole core calculations. Also, though matrix-vector product based methods (current implementation) is faster than matrix-free methods for linear elements, it is much slower using high order elements.

So this would be a future suggestion: try out matrix-free implementations.

The second is actually a consequence of the first limitation: we would not be able to do JFNK through NOX (`Trilinos`) or SNES (`PETSc`) effectively as JFNK relies on interpreting behavior of matrix-vector product instead of do matrix-vector product directly.

5 What we need to think about?

5.1 What is `XTrans` to `BART`?

This is the first thing coming into my mind. For Marissa's project, `XTrans` is indeed the way to go as we've already had much progress on the code and hopefully will be able to do calculations

for eigenvalue problems by the end of the summer. But, if we do `BART`, which will take interested individuals to participate, that would be great to train everybody in the directions of method developing, collaboration and coding something complex (I bet it would be more complex than most of students' `Python` codes). So I still need to think about this: how we could incorporate existing work in `XTrans` into the `BART`.

5.2 What kind of the work I should assign to students instead of getting my own hand dirty?

Indeed, Postdocs are the labors taking the heavy missions fast. However, a balance still needs to be carefully found between assigning missions ourselves and students because of the nature of writing the whole codes: ramping up students and letting them learning things (mathematics and implementations). The question is actually how to find and manage the balance. Maybe this is also a good point for me to learn.