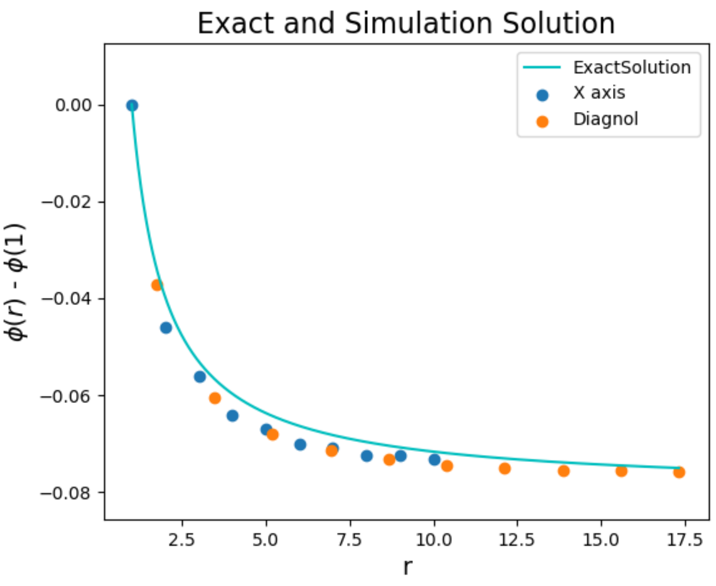
CUDA Parallel Programming Problem Set 10

R08244002 Shin-Rong, Tsai

- **Solve Poisson Equation using FFT**

I. Result

Run and compile the code *poisson_3d.cu*, and plot the result with *plot.py.*



II. Discussion

A. Exact solution for $\nabla^2\phi = \rho$

Use $\phi(r=1)$ as base and let $q=1$.

$$\vec{E} = -\nabla\phi = \frac{q}{4\pi r^2}$$

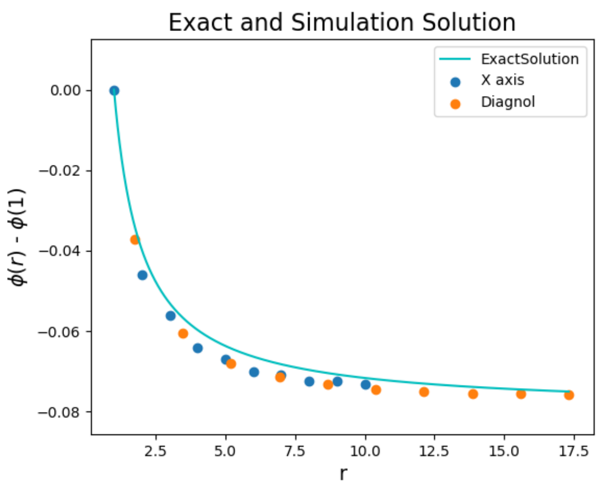
$$\Rightarrow \phi(r)-\phi(1) = \int_r^1 \vec{E}\cdot d\vec{r} = \int_r^1 \frac{q}{4\pi r^2}\cdot d\vec{r} = -\frac{q}{4\pi r}\bigg|_r^1 = \frac{1}{4\pi}\left(\frac{1}{r}-1\right)$$

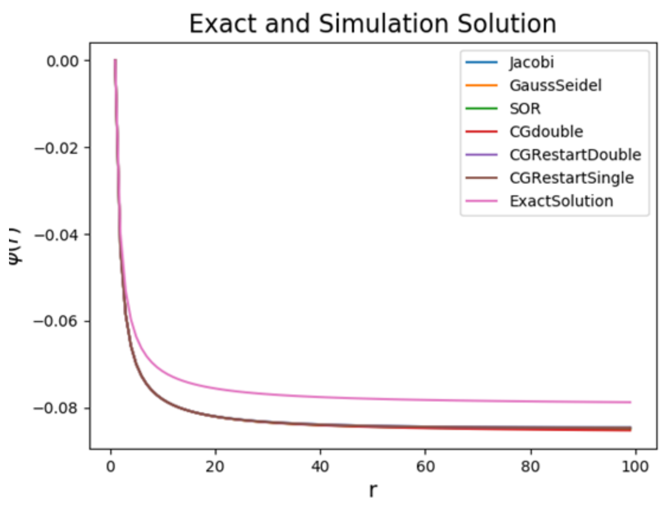
$$\Rightarrow \phi(r) = \frac{1}{4\pi r}-\frac{1}{4\pi}+\phi(1)$$

Set $\phi(1)$ as the basis, we can indeed check that the Poisson Equation we solved using FFT method is correct.

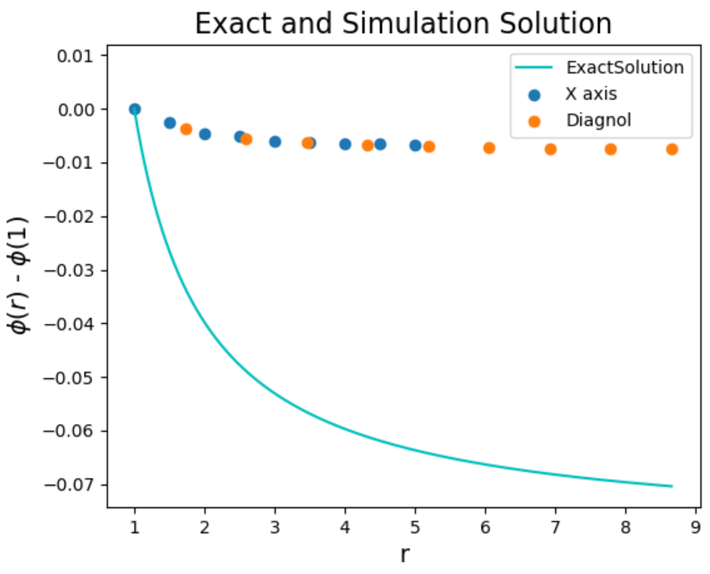
B. Compare with the relaxation method.

I took the result from last semester in course Computational Physics. We can see that FFT method is way more accurate than the other relaxation method, compare only from distance within around 20. I think because when we do FFT, it uses the whole grid as reference, not just the local contributions like the relaxation method.





C. Will it get more accurate if we set $\Delta x = 1.0$ smaller?
I encountered some problems that I still can't figure it out yet. If I set the configuration such that $\Delta x = 0.5$ (something that doesn't equal to 1.0), there is still a relationship of $\frac{1}{r}$ with wrong scaling. The code is in file *poisson_3d_deltax.cu*. I really wished to know where is the problem in my code.



- **Maximum of the grid size to do 3D FFT**
  It is around 460~465 points in one direction. This must be related to the memory a GPU has. I my code, every time I do FFT, there is a data in / out device memory been allocated, which is not really efficient in memory usage. This might make the sample points we can use lesser, since some of the device memory is occupied.

*reference: https://docs.nvidia.com/cuda/cufft/index.html#cuff-free-memory-requirement