# Project 3

Jack Jiang

March 19, 2021

## Preparation

1. My work plan for Project 2 was semi-effective. It helped me pace my work for the project, however, I did not account for debugging. Debugging messed my whole timeline for the plan up and so I will need to allocate my time into the coding portion for this project to be safe. This means that I will start and finish the coding portion earlier and have more time to do so in order to follow my plan accordingly. My work plan for completing this project is to start the coding portion after I finish this assignment and complete it at least 3 days before the project deadline. In order to complete the required tasks of the project, I will have to modify my project 2 code to have it run with OpenMP and parallelize my functions. To do this, I will have to implement OpenMP into my code and use it to parallelize the specified functions/parts of the simulation. This should take around 1-2 days and I will test it by making sure my functions are actually being parallelized properly and so run-times are getting faster and by going to office hours. I will also have to implement a function and a program for computing different properties of the wave equation and to measure timing of both the simulated and true solutions. I will do this by using my pseudo-code generated in preparation task 4 to help with my error functions and programs. I believe this should take about 1 day. I will test these tasks by making sure the computations are correct, accurate, and reasonable. Then, I will need to update output programs and write a batch script. I will do this by making sure the correct output is being produced and refer to previous projects/labs with SLURM scripts. This should take about 1 day and I will test it by running both and check the outputs. For the analysis portion of this project, I will have to run strong and weak scalability studies for varying arguments and create plots. I will do this by referring to my scalability study plan produced in task 5 of the preparation and previous labs. This should take about 1-2 days. I will then have to run different iterations of my Monte Carlo simulator for a wide range of n values, which should take some time to run, and plot my results. I will do this by following the task as directed. This will take time to run as n gets quite large so I will be safe and say this should take 1-2 days, accounting for ARC resource requirements. Next, I will have to

include images for different time-steps and generate the time history of the error. I will do this by referencing previous projects/labs and going to office hours for help if needed. This will take about 1 days. Adding up the total time, in the worst case, it will take about 7-8 days to complete this project and I will need to get started as soon as possible.

2. - My implementation did not have a very detailed data structure, while the provided solution had a data structure that was very detailed and organized.
   - The solution had a separate C file for all the wave equation computations while I had 1 singular C file for my solution.
   - The solution has a lot of error checking whereas my solution does not so it is something I should be thinking about implementing more often.

3. In our simulation of the wave equation, parallelism in the context of openMP can be found in the evaluate_standing_wave(...) function and the wave_timestep(...), in both of these functions' nested for loops.

4. Pseudo-code for norm of error:

```
import math.h

float norm_error(u_sim, u_true)
    int nx = u_sim->N
    int ny = u_sim->N
    //              1    /    nx *   nx
    float const = 1/(float(nx)*float(ny))
    float sum = 0

    Array2D_f error
    allocate_Array2D_f(&error, ny, nx)
    initialize(&error)

    // u_sim - u_true
    for (int j = 0; j < ny-1; j++)
        for (int i = 0; i < nx-1; i++)
            int idx = ji_to_idx(j, i, nx)
            error->data[idx] = u_sim->data[idx] -
            u_true->data[idx]

   // sum of j=0:ny-1, i=0:nx-1 a(xi,yi)^2
   for (int j = 0; j < ny-1; j++)
        for (int i = 0 i< nx-1; i++)
            int idx = ji_to_idx(j, i, nx)
            sum += pow(error->data[idx], 2)
    sum = sqrt(sum)
```

```
deallocate_Array2D_f(&error)

return const * sum
```

5. (a) The difference between a strong scalability study and a weak scalability study lies in the differences between the 2 scalabilities.Strong scalability is the measured speedup as the number of processors increases on a fixed size problem whereas weak scalability measures the speedup as the number of processors increases with fixed work per processor [1]. In a strong scalability study, $T_1$ and $T_p$ can be observed and so a constant problem size is examined, while in a weak scalability study, $T_1$ cannot be directly observed and a constant work per processor is examined [1].

   (b) (i) $W_P = (1-\alpha)\frac{N_p}{P} + \alpha N_P$

   (ii) $W_P = (1-\alpha)W_1 + \alpha P W_1$

   (iii) $N_P = n_1 * P$

   (iv) If $n_1 = n_x = n_y = 2501$, $n_2 = 5002$, $n_4 = 10004$, $n_8 = 20008$, $n_16 = 40016$, $n_32 = 80032$, $n_64 = 160064$.

6. I discussed concepts and problems with Shaan Chudasama for this assignment.

## References

[1] part18.01-gustafson$_s$*calability.pdf*