1. General Remarks

In this assignment you are asked to write an OpenMP code which predicts positions and velocity of n bodies (or particles) interacting according to Newton laws. It is known as the n-body problem.

(Pleasze refer to lecture notes for Lecture 7.)

For each $i = 1, ..., n$ the input to the problem is:

- mass m_i
- initial position $r_i(0)$
- initial velocity $v_i(0)$
- integratiopn stepsize Δt
- number of steps N.

The output is:

- positions $r_i(t)$ for $t \in \{\Delta t, 2\Delta t, ..., N\Delta t\}$
- velocities $v_i(t)$ for $t \in \{\Delta t, 2\Delta t, ..., N\Delta t\}$

The following equations are used:

1. force exerted on particle i by particle j
$$f_{i,j}(t) = -\frac{g m_i m_j}{||r_i(t) - r_j(t)||_2^2} \frac{(r_i(t) - r_j(t))}{||r_i(t) - r_j(t)||_2}$$
2. $f_{i,j}(t) = -f_{j,i}(t)$
3. force exerted on i by all particles is
$$F_i(t) = \sum_{j \neq i} f_{i,j}(t) = -g m_i \sum_{j=i} \frac{m_j}{||r_i(t) - r_j(t)||_2^3}(r_i(t) - r_j(t))$$
(In order to avoid division by zero, when calculating $||a-b||_2^2 = \sum_{j=1}^2 (a_1 - b_1)^2 + (a_2 - b_2)^2$, add a small constant, like $\sum_{j=1}^2 (a_1 - b_1)^2 + (a_2 - b_2)^2 + c^2$.)

4. Newton's second law

   $$F_i(t) = m_i \cdot a_i(t) = m_i r_i''(t),\ i = 0, 1, ..., n-1.$$

   from which it follows

   $$a_i(t) = \frac{F_i(t)}{m_i} \tag{1}$$

We considered the "force" matrix

$$\begin{pmatrix} F_0 \\ F_1 \\ \vdots \\ F_{N-1} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & f_{0,1} & f_{0,2} & \cdots & f_{0,n-1} \\ -f_{0,1} & 0 & f_{1,2} & \cdots & f_{1,n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -f_{0,n-1} & -f_{1,n-1} & -f_{2,n-1} & \cdots & 0 \end{pmatrix}}_{F} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

We noted, that as $f_{i,j}(t) = -f_{j,i}(t)$ it suffices to compute only the (strictly) upper triangular part.

We then discussed how the computations in the n-body problem can be divided among threads. There were two proposals

- assign blocks of rows of the entire force matrix to separate threads
- assign rows of the triangular part only to separate threads

You are asked to implement both strategies and find out for what n and number of threads p one strategy results in faster execution of the n-body simulations.

2. Positions and velocities

Positions and velocities are updated using Taylor series expension in the following way.

1. move coordinates to the "center of momentum frame", see

   `https://en.wikipedia.org/wiki/Center-of-momentum_frame`,

   there the total momentum in the center-of-momentum system venishes, and the center of the mass moves with constant velocity, see

   `https://en.wikipedia.org/wiki/N-body_problem`

2. based on initial positions, compute initial acceleration from (??)

3. then the velocity is updated as

   $$v_i(t + \frac{\Delta t}{2}) = v_i(t) + a_i(t)\frac{\Delta t}{2}$$

   this is called "half kick"

4. then the position is updated
$$r_i(t+\Delta t) = r_i(t) + v_i(t+\frac{\Delta t}{2})\Delta t$$
this is called "drift"
5. accelerations a_i are computed based of new positions from (**??**)
6. velocity is updated
$$v_i(t+\Delta t) = v_i(t+\frac{\Delta t}{2}) + a_i\frac{\Delta t}{2}$$
this is called "half kick"
7. compute and record total energy, it should be preserved, however, due to the numerical approximations it may drift a bit
8. if not finished go back to (3)

3. Requirements

A template for the code can be found in

`/classes/ece5720/assignments/hw3`

To check the numerical correctness of your algorithm compute the following

Your algorithm should be benchmarked for a range of parameters n and p where

- n is the number of particles, followed by
- p is the number of threads.

n should start from MIN_N, then doubled until n = MAX_N. Try to make MAX_N as large as the ecelinux cluster will allow you.

p should start from 1, then doubled until twice the number of cores is reached.

Measure the execution time for all pairs (n, p).

Write a document that describes how your programs work. Sketch the key elements of your parallelization strategy. Explain how your program partitions the data and work among threads and how they are synchronized. Explain whether the workload is distributed evenly among threads. Justify your implementation choices.

Your findings, a discussion of results, graphs and tables should be saved in a file

`your_net_id_hw2_writeup.pdf`.

Please present your tables and graphs in a way so they are easily readable. For example, if certain combinations of (number o threads, matrix dimension) do not bring any new information, you may omit them from your graphs. But then explain why you are omitting them.

Your code (with instructions how to compile and execute it) should be saved in a file `your_net_id_hw3_code.c`.

Submit seperately these two files.