MA453/553 – Homework/Lab # 3
Parallelizating serial codes using MPI
Due: 11/08/2024

Download the file `hw3_files.zip`, send it (`scp`) to your **wxsession** account. Move it in some appropriate work directory and `unzip` it. The zip file contains the three Python codes `mc_vol.py`, `num_int.py` and `adv_diffusion.py`, and skeleton codes for MPI parallelization: `mc_vol_skeleton_mpi.py` and `adv_diffusion_skeleton_mpi.py` using `mpi4py` package. Give a look to both serial an parallel versions of the codes and complete parallelization filling the correct arguments in the MPI functions.

---

1. The code `mc_vol.py` approximates the volume of the solid ("Ice Cream Cone") that lies above the cone $z = \sqrt{x^2 + y^2}$ and below the sphere $x^2+y^2+(z-1)^2 = 1$ using the Monte Carlo integration method. The exact volume is computed using the triple integral in spherical coordinates.

$$\int_0^{2\pi} \int_0^{\pi/4} \int_0^{2\cos\phi} \rho^2 \sin\phi \, d\rho d\phi d\theta = \pi$$

This is an embarassingly parallel algorithm, you just need to collect the results from each worker. Let n workers are assigned to do the same task and each sends the result to the master process and the master calculates the average.

```
def mc_volume(n):
    """ Approximates volume of the solid below the sphere x^2+y^2+(z-1)^2=1 and
    above  the cone z^2=x^2+y^2 using Monte Carlo integration with n random points."""
    count=0
    for i in range(n):
        x=random.uniform(-1,1)
        y=random.uniform(-1,1)
        z=random.uniform(0,2)
        if x**2+y**2 < z**2 and x**2+y**2 < z*(2-z):
            count += 1
    return 8*count/n
```

2. The code `num_int.py` contains several versions of Trapezoid rule implementation. Choose the NumPy version and parallelize it using MPI. Parallelize this code by breaking the interval $[a, b]$ into n parts and let each of the n workers call the same function `trapezoid_rule()` with different values of a and b depending on their process Id `rank`) (task parallelism). Work on the `__main__` part of the code. You can use point to point communication `MPI.COMM_WORLD.Send()` and `MPI.COMM_WORLD.Recv()`, or/and collective communication `MPI.COMM_WORLD.Reduce()` with the operation `MPI.SUM`. See the strategy in `midpoint3.py` (`mpi4py_basics.zip`).

```
def trapezoid_rule(a,b,n):
    x,dx=np.linspace(a,b,n+1,retstep=True)
    y=f(x)
    tsum=np.sum(y[1:-1])
    return dx*(tsum+0.5*(y[0]+y[-1]))
```

3. The code `add_diffusion.py` numerically solves the one dimensional advection diffusion equation $u_t + au_x = Du_{xx}$, $a > 0$ with an initial condition $u(x, 0) = f(x)$ using an explicit finite difference scheme.

$$u_i^{n+1} = u_i^n + \Delta t \left[a \left(\frac{u_i^n - u_{i+1}^n}{\Delta x} \right) + D \left(\frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} \right) \right]$$

Here, you need to do domain decomposition and every worker should know their left and right neighbors' `rank` to exchange their updated values at the every time step.Only two workers working on the left and right boundary points updates the real boundary conditions. Focus on the function `update_pde_expl()`.

```
def update_pde_expl(u1,dx,dt):
    N=len(u1)-1
    u2=np.zeros(N+1)
    # boundary conditions u(0,t)=u(L,t)=0
    u2[0]=0.0; u2[N]=0.0
    for i in range(1,N-1):
        adv=a*(u1[i]-u1[i+1])/dx
        diff=D*(u1[i-1]-2*u1[i]+u1[i+1])/(dx*dx)
        u2[i]=u1[i]+dt*(adv+diff)
    return u2
```

4. Beyond "mega" (10^6) and "giga" (10^9) are "tega" (10^{12}), "peta" (10^{15}) and "exa" (10^{18}). Determine the correct time unit that it takes a single computer to do a floating point operation if it does (1) one megaflop per second, (2) one gigaflop per second, (3) one teraflop per second and (4) one petaflop per second.

5. What is the work/memory ratio ρ_{wm} of the number of floating point operations to the number of data values that have to be obtained from memory, or written to memory, in the computation of the product of two square matrices $\mathbf{A} = (a_{ij})$ and $\mathbf{B}$ which is defined by

$$(\mathbf{AB})_{ij} := \sum_{k=1}^{n} a_{ik} b_{kj} \qquad \text{for } i, j = 1, \cdots, n$$

as a function of n? (Assume that both $\mathbf{A}$ and $\mathbf{B}$ must be obtained from memory and the product is written back to memory. However, the denominator should just be the volume of the data, not the number of memory references that might occur in particular algorithm. for example, the term b_{11} in the above sum occurs n times but should be counted only once.)

6. The peak performance of the world's most powerful parallel computer has undergone an exponential growth. The table below lists the history between 1993 and 2016, according to the Top500 List. Suppose the growth can be described mathematically by the following formula:

$$G(t) = G_0 \cdot 2^{\frac{(t-t_0)}{t*}}$$

where $G(t)$ denotes the development of the theoretical peak performance as a function of time, in terms of GFLOPS, and G_0 denotes the GFLOPS at t_0. Estimate the value of $t*$, i.e., the average length of time over which the peak performance doubles. Use the method of least squares. If the same growth continues, what will the peak performance of the most powerful parallel computer be in year 2025?

Most Powerful Supercomputer – Historical TOP500 List

Year	Supercomputer	Peak speed	Location
1993	Fujitsu Numerical Wind Tunnel	124.50 GFLOPS	National Aerospace Laboratory, Tokyo, Japan
1993	Intel Paragon XP/S 140	143.40 GFLOPS	DoE-Sandia National Laboratories, USA
1994	Fujitsu Numerical Wind Tunnel	170.40 GFLOPS	National Aerospace Laboratory, Tokyo, Japan
1996	Hitachi CP-PACS/2048	368.2 GFLOPS	University of Tsukuba, Tsukuba, Japan
1997	Intel ASCI Red/9152	1.338 TFLOPS	DoE-Sandia National Laboratories, USA
1999	Intel ASCI Red/9632	2.3796 TFLOPS	
2000	IBM ASCI White	7.226 TFLOPS	DoE-Lawrence Livermore National Laboratory, USA
2002	NEC Earth Simulator	35.86 TFLOPS	Earth Simulator Center, Yokohama, Japan
2004	IBM Blue Gene/L	70.72 TFLOPS	DoE/IBM Rochester, USA
2005		136.8 TFLOPS	DoE Lawrence Livermore National Laboratory, USA
2006		280.6 TFLOPS	
2007		478.2 TFLOPS	
2008	IBM Roadrunner	1.026 PFLOPS	DoE-Los Alamos National Laboratory, USA
2009	Cray Jaguar	1.759 PFLOPS	DoE-Oak Ridge National Laboratory, USA
2010	Tianhe-IA	2.566 PFLOPS	National Supercomputing Center, Tianjin, China
2011	Fujitsu K computer	10.51 PFLOPS	RIKEN, Kobe, Japan
2012	IBM Sequoia	16.32 PFLOPS	Lawrence Livermore National Laboratory, USA
2012	Cray Titan	17.59 PFLOPS	Oak Ridge National Laboratory, USA
2013	NUDT Tianhe-2	33.86 PFLOPS	Guangzhou, China
2016	Sunway TaihuLight	93.01 PFLOPS	National Supercomputing Center, Wuxi, China
2017	SW26010 260C, Sunway	125.436 PFLOPS	NRCPC National Supercomputing Center Wuxi, China
2018	IBM Power System AC922, Summit	200.79 PFLOPS	IBM DOE/SC/Oak Ridge National Laboratory, USA
2020	Fugaku, A64FX 48C 2.2GHz, Fujitsu	537.21 PFLOPS	RIKEN Center for Computational Science, Japan
2022	HPE Cray EX235a Frontier	1685.65 PFLOPS	Oak Ridge National Laboratory, USA
2024	HPE Cray EX235a Frontier	1714.81 PFLOPS	DOE/SC/Oak Ridge National Laboratory

7. Run all the parallelized code from **1 -3** in **wxsession** and **vega** and report their parallel performances.