Eng 67 / 115
Lab 6

Goal: To explore the concepts of **granularity** and **load balance** control in *functional decomposition*. This assignment builds upon the project designed in Labs 1 and 2.

Required MPI Routines:

- `MPI_Init()`
- `MPI_Finalize()`
- `MPI_Comm_size()`
- `MPI_Comm_rank()`
- `MPI_Recv()`
- `MPI_Send()`

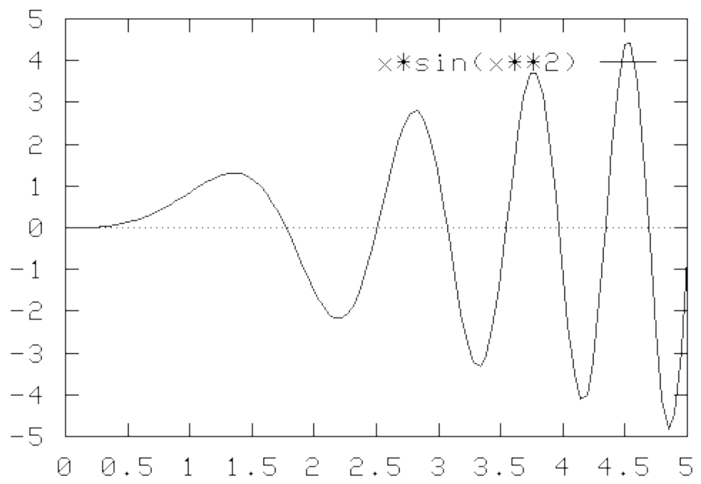
- **`MPI_Wtime()`**

Tasks

1. Using the same concepts used in Lab 2 (***and reusing the code*** !!!) – develop a message passing program that computes the integral over some range using some fixed number of processes with the same decomposition technique i.e. divide the range into components, solve each component by a single process that executes some number of threads.

2. Modify your program to compute and print the *minimum*, *maximum*, and *average* number of trapezoid calculations performed at each processor, and the overall all wall-clock time to complete the computation using MPI_Wtime().

3. The number of trapezoids required to get an accurate estimate of the integral over a particular region should depend on the smoothness of the function over the range. Use your concurrent program to integrate the function $x*sin(x^2)$ shown below.



3. Graph the variation in the *average number of trapezoid calculations* performed by each processor (sum of number computed by each thread) as a function of the *precision* – this variation represents that amount of work each processor performs and directly reflects the *load balance the processors*. Show error bars on your graph corresponding to the minimum and maximum number of calculations for each average calculation. Make sure you use inputs that cause the function to be executed over several minutes. The number of trapezoidal iterations required for *equal-sized sub-ranges* of the integration may vary by several orders of magnitude for a given level of precision. Consequentially, the load balance may vary considerably and the total time for the computation is governed by the slowest process.

4. **ENG 115:** Improve your decomposition method to alleviate the load imbalance by implementing a **REUSABLE** *manager-worker load balancer* based on the communication structure implemented in Lab5. The manager generates a large number of partitions p (at least 10 times larger than the number of processors). You might place the partitions to be solved in a *queue*? Each worker will repeatedly request a partition to solve, solve it, and a return a partial result to the manager to be added into the total integral calculation.

   To demonstrate an improvement in load balance, calculate the total number of trapizod calculations performed by each process for all partitions that are integrated by that process. Calculate the min / max / average number of trapizoid calculations over the set of processes. The number of evaluations used by each process should be roughly the same and close to the average. Vary the number of partitions and observe how the distribution changes. If you prefer, you may use a standard deviation to explore these ideas instead of min/max.

RESEARCH:

Take a look at the lecture video for Lab 7 and make sure you understand how you might optimize your reusable load-balancing module. Ensure that you understand:

- How to improve the initial delay associated with partitioning
- How you might diminish the ramp up time
- How you might increase the overall processor utilization