CPS 303 High Performance Computing

Wensheng Shen
Department of Computational Science
SUNY Brockport
Chapter 4 An application: numerical integration

- Use MPI to solve a problem of numerical integration with the trapezoidal rule.
4.1 The trapezoidal rule

Definite integral of a nonnegative function
Trapezoids approximating definite integral

\[ \int_{a}^{b} f(x) \, dx \]
The ith trapezoid

\[ \frac{1}{2} h [f(x_{i-1}) + f(x_i)] \]

\[ \frac{1}{2} h [f(x_0) + f(x_1)] + \frac{1}{2} h [f(x_1) + f(x_2)] + \ldots + \frac{1}{2} h [f(x_{n-1}) + f(x_n)] \]

\[ = \frac{h}{2} [f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + f(x_n)] \]

\[ = [f(x_0)/2 + f(x_n)/2 + f(x_1) + f(x_2) + \ldots + f(x_{n-1})]h \]
#include <stdio.h>

main()
{
    float integral; /* store result in integral */
    float a, b; /* left and right endpoints */
    int n; /* Number of trapezoids */
    float h; /* trapezoid base width */
    float x;
    int i;

    float f(float x); /* function we're integrating */
    printf("Enter a, b, and n\n");
    scanf("%f %f %d", &a, &b, &n);

    h = (b-a)/n;
    integral = (f(a) + f(b))/2.0;
    x = a;
    for(i=1; i<=n-1; i++)
    {
        x = x + h;
        integral = integral + f(x);
    }
    integral = integral*h;
    printf("with n=%d trapezoids, our estimate\n", n);
    printf("of the integral from %f to %f = %f\n", a, b, integral);
}
float f(float x)
{
    float return_val;
    /* calculate f(x). Store calculation in return_val. */
    ...
    return return_val;
}
In this case, data is just the interval $[a, b]$ and the number of trapezoid. We can parallelize the trapezoidal rule program by assigning a subinterval of $[a, b]$ to each process, and having that process estimate the integral of $f$ over the subinterval.
Suppose there are $p$ processes and $n$ trapezoids, and, in order to simplify the discussion, we also suppose that $n$ is divisible by $p$. Then it is natural for the first process to calculate the area of the first $n/p$ trapezoids, the second process to calculate the area of the next $n/p$, etc.

MPI identifies each process by a nonnegative integer. So if there are $p$ processes, the first is process 0, the second process 1, ..., and the last process $p-1$. 
## Assignments of subintervals to processes

<table>
<thead>
<tr>
<th>Process</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[a, a+nh/p]</td>
</tr>
<tr>
<td>1</td>
<td>[a+nh/p, a+2nh/p]</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>[a+inh/p, a+(i+1)nh/p]</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>P-1</td>
<td>[a+(p-1)nh/p, b]</td>
</tr>
</tbody>
</table>

Assignments of subintervals to processes
Each process needs the following information:
- The number of processes, $p$
- Its rank
- The entire interval of integration, $[a, b]$
- The number of subintervals, $n$

Each process sends its result to process 0, and process 0 will do the final addition.
Algorithm

- Each process calculates “its” interval of integration.
- Each process estimates the integral of $f(x)$ over its interval using the trapezoidal rule.
- Each process $\neq 0$ sends its integral to 0.
- Process 0 sums the calculations received from the individual processes and prints the result.
Global and local variables

- Global variables: variables whose contents are significant on all processes, such as a, b, and n in the previous code;

- Local variables: variables whose contents are significant only on individual processes, such as local_a, local_b, and local_n in the previous code

- Use different names for global and local variables.
In the previous code, the input data, a, b, and n, are hardwired. If we want to change any of these, we must edit and recompile the program.

We assume that process 0 can do both input and output. Let us further assume all processes can do input and output.

We add the following piece of code and run it with two processes:

```c
scanf("%f %f %d", &a, &b, &n);
```

0 1 1024 (user types in)

Who gets the data? Process 0, process 1, or both, or process 0 gets the 0 and 1, but process 1 gets 1024?

If several processes attempt to simultaneously write data to the terminal screen, are the data printed in order?

It is out of control
Solution: let process 0 be the master process to do I/O

- Only process 0 can do I/O, and we need process 0 to send the user input to the other processes.
- We need to write another function, Get_data(), dealing with input and output.
- Other ways: modify existing code, collective communication, next chapter.
Void get_data(float* a_ptr, float* b_ptr, int* n_ptr, int my_rank, int p)
{
    int source = 0;
    int dest;
    int tag;
    MPI_Status status;
    if(my_rank == 0)
    {
        printf("enter a, b, and n\n");
        scanf("%f %f %d", a_ptr, b_ptr, n_ptr);
        for(dest = 1; dest < p; dest++)
        {
            tag = 0;
            MPI_Send(a_ptr, 1, MPI_FLOAT, dest, tag, MPI_COMM_WORLD);
            tag = 1;
            MPI_Send(b_ptr, 1, MPI_FLOAT, dest, tag, MPI_COMM_WORLD);
            tag = 2;
            MPI_Send(n_ptr, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);
        }
    }
    else
    {
        tag = 0;
        MPI_Recv(a_ptr, 1, MPI_FLOAT, source, tag, MPI_COMM_WORLD, &status);
        tag = 1;
        MPI_Recv(b_ptr, 1, MPI_FLOAT, source, tag, MPI_COMM_WORLD, &status);
        tag = 2;
        MPI_Recv(a_ptr, 1, MPI_INT, source, tag, MPI_COMM_WORLD, &status);
    }
}
We have 8 trapezoids and 3 processes, how can we distribute the eight trapezoids to 3 processes? We take the floor of \((n/p)\). \(n/p=8/3=2.6\), floor\((n/p)\)=2. Processes 0 and 1 have 2 trapezoids each, process 3 has 4 trapezoids. We let the first \(p-1\) processes to have floor\((n/p)\)+1, and the \(p\)th process to have \(n-(\text{floor}(n/p)+1)*(p-1)\) trapezoids.

**The floor of \((n/p)\)**

\[
\begin{vmatrix}
\frac{n}{p}
\end{vmatrix}
\]
The Simpson’s rule

Three points integration, the number of intervals must be an even number.

If the number of total panels is n, then the number of Simpson panels is n/2
\text{ith interval} \\
[a+(i-1)h, a+ih] \\
\text{(i+1)th interval} \\
[a+ih, a+(i+1)h] \\
\text{Area} = h/3(f(x_{i-1}) + 4f(x_i) + f(x_{i+1})) \\
h = \frac{b-a}{n}
In general,

\[
\int_a^b f(x) \, dx = \frac{h}{3} \left[ f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n) \right]
\]
The derivation of Simpson’s rule: undetermined coefficients

Using 3-point integration

\[ \int_{a}^{b} f(x) \, dx \approx \alpha f(a) + \beta f \left( \frac{a + b}{2} \right) + \gamma f(b) \]

Quadratic polynomial

\[ f(x) = c_1 x^2 + c_2 x + c_3 \]
The relationship between Simpson’s rule and the trapezoid rule

Midpoint rule
\[
\int_{a}^{b} f(x) \approx (b - a) f\left(\frac{a + b}{2}\right) = A
\]

Trapezoid rule
\[
\int_{a}^{b} f(x) \approx (b - a) \frac{f(a) + f(b)}{2} = B
\]

Simpson’s rule
\[
\int_{a}^{b} f(x) \approx \frac{2A + B}{3}
\]