

Excercise Sheet 7

Solution

Lecture Parallel Processing

Winter Term 2025/26

Exercise 1: Point to point communication with MPI

```
if (myrank == 1)
    MPI_Send(&myval, 1, MPI_INT, 3, 13, MPI_COMM_WORLD);
if (myrank == 2)
    MPI_Send(&myval, 1, MPI_INT, 0, 20, MPI_COMM_WORLD);
if (myrank == 3)
    MPI_Recv(&myval, 1, MPI_INT, 1, 13, MPI_COMM_WORLD, &status);
if (myrank == 0)
    MPI_Recv(&myval, 1, MPI_INT, 2, 20, MPI_COMM_WORLD, &status);
```

Exercise 2: Parallelization of a map operation using MPI

```
int n = N/nprocs;
double *lx = new double[n];
double *ly = new double[n];

MPI_Scatter(x, n, MPI_DOUBLE, lx, n, MPI_DOUBLE, 0, MPI_COMM_WORLD);

for (int i=0; i<n; i++)
    ly[i] = complex_fct(lx[i]);

MPI_Gather(ly, n, MPI_DOUBLE, y, n, MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

Exercise 4: Numerical integration using MPI

```
double Parallel_Integration(int n)
{
    int i;
    double h, sum, x;
    double gsum = 0;
    int local_n, start;

    h = 1.0 / (double)n;
    sum = 0.0;

    local_n = (n + myrank) / nprocs;
    start = n / nprocs * myrank;
    if (myrank > nprocs - n % nprocs)
        start += myrank - (nprocs - n % nprocs);

    for (i = 1 + start; i <= start + local_n; i++) {
        x = h * ((double)i - 0.5);
        sum += f(x);
    }

    /* Collect the result in process 0 */
}
```

```

MPI_Reduce(&sum, &gsum, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

return h * gsum;
}

int main(int argc, char* argv[])
{
    ...
    if (myrank == 0) {
        for (;;) {
            std::cout << "Enter the number of intervals: (0=exit) " << std::endl;
            std::cin >> n;
            /* Send n to all processes */
            MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
            if (n <= 0)
                break;

            // serial solution
            ...

            // parallel solution
            std::cout << "Parallel solution:" << std::endl;
            t1 = std::chrono::high_resolution_clock::now();

            MPI_Barrier(MPI_COMM_WORLD);
            res = Parallel_Integration(n);
            std::chrono::duration<double> t_p = std::chrono::high_resolution_clock::now()
                - t1;
            std::cout << "  n: " << n << ", integral is approximatly: "
                << std::setprecision(17) << res << std::endl
                << "  Error is: " << fabs(res - PI) << ", Runtime[sec]: "
                << std::setprecision(6) << t_p / std::chrono::seconds(1) << std::endl;
            std::cout << "  Nprocs: " << nprocs
                << "  Speedup: " << std::setprecision(3)
                << ((t_p > std::chrono::nanoseconds(0)) ? t_s / t_p : 0.0) << std::endl;
        }
    }
    else {
        for (;;) {
            /* Receive n from process 0 */
            MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
            if (n <= 0)
                break;
            /* Calculate PI */
            MPI_Barrier(MPI_COMM_WORLD);
            Parallel_Integration(n);
        }
    }

    MPI_Finalize();
    return 0;
}

```