# MATH 700: INTRODUCTION TO PARALLEL COMPUTING USING MPI

MADHU NAYAKKANKUPPAM
HOMEWORK 4

Assigned: **October 09, 2001** Due: **October 23, 2001**

Name: ________________________
ID #: ________________________
E–mail: ________________________

---

## INSTRUCTIONS

- This homework has **three (3) pages**. Make sure you have the entire homework.
- **The first page of your submitted homework must be this page. Staple** all the sheets of your solution on the *top, left corner.* Please save trees by using both sides of the paper, but start every question on a fresh page.
- Leave your completed assignment on the lectern in the classroom *at the start of class on the due date.* **Late homework will generally not be accepted.**

---

For instructor's use only:
Score: /20

---

**Reading exercise (0 points):** Read Ch. 6 and 8.4.1 of the textbook.

**Background work (0 points):** Look up the BLAS (Basic Linear Algebra Subroutines) man pages and learn how to use the functions `dnrm2`, `daxpy`, `dger`, `dgemm`. A good starting point is `www.netlib.org/blas`.

**Matrix Multiplication (20 points):** The purpose of this exercise is to manipulate derived types, NOT to come up with the best method for matrix multiplication in parallel.

The product of $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$ can be written as the following sum of outer products

$$AB = \sum_{i=1}^{k} a_i b_i, \tag{1}$$

where $a_i$ and $b_i$ are the $i^{\text{th}}$ column of $A$ and the $i^{\text{th}}$ row of $B$ respectively. You have to implement this for *dense matrices* in parallel along the lines indicated below:

- On the root process, read in $m, k, n$ (from command line, file or the console).
- On the root process, generate random matrices $A$ and $B$ with these dimensions.
- Build a derived type for communicating blocks of columns of $A$
- Build a derived type for communicating blocks of rows of $B$
- Block–scatter the columns of $A$
- Block–scatter the rows of $B$
- Compute, on each process, a local sum of outer products as in (1). In this step, you will need the BLAS routine `dger`.
- Reduce the results onto the root process, say to a matrix $C$, which supposedly contains $AB$ (if your code worked correctly)
- Finally, on the root process, verify your answer thus: Compute $D = AB$ process, and print out the Frobenius norm of the error $\|C - D\|_F$. In this step, you will need the BLAS routines `dgemm`, `daxpy` and `dnrm2`.

Note the following:

**Matrix format:** All matrices contain double precision numbers in column–major format. If you program in C, you could use the following structure for dense matrices:

```
typedef struct tagDMAT{ /* dense matrix */
  unsigned int nrows;   /* number of rows */
  unsigned int ncols;   /* number of cols */
  double *v;            /* pointer to nrows x ncols entries */
  /* matrix entries are in column major format */
} DMAT;
```

although you are free to modify this if you want.

**Matrix sizes:** You may assume that the number of processes divides $k$, so $A$ and $B$ can be evenly distributed by blocks of columns and blocks of rows respectively.

**Coding style:** Write well–documented, modular code consisting of simple functions; **this will help you in subsequent homeworks.** (Some credit will be allocated for coding style.) One large "main program" is very hard to understand.

**BLAS:** The BLAS library is available at: `/usr/local/libblas.a` on `pc51.math.umbc.edu`. (The man pages are already in your path.)

**What to turn in:** Turn in the following:

1. **In no more than 15 sentences**, explain clearly what derived types you constructed to block–scatter $A$ and $B$.
2. Make a table with 5 different test cases of triples $(m, n, k)$ along with the error measure $\|C - D\|_F$.
3. Attach a listing of your source code.
4. Store all your files in the directory `xxxx`, where `xxxx` are the last 4 digits of your ID number. Include a README text file in this directory, if you want to add special comments or notes for me (*e.g.* compilation notes). Create a gzipped tar file by executing:

```
% tar cvf xxxx.tar xxxx
% gzip xxxx.tar
```

Send me the resulting file (`xxxx.tar.gz`) as an e-mail attachment (madhu@math.umbc.edu) before the due date. Please do not make a submission until you are certain that this is the version you want to submit, and that it tallies with the hard–copy you will hand in. (However, if you must make multiple submissions, only the latest submission will be considered.)