Homework 1

NERS/ENGR 570 Fall 2020
September 10th, 2020
DUE BY: September 21th, 2020 by 11:59 pm

Exercise 1: LaTeX (20 pts) Create a 1-3 page PDF using LaTeX that describes the mathematics or governing equation of the problem you wish to solve with scientific computing in your research.

The document must include:

- references to at least one journal article and at least one book
- a bibliography generated using BibTeX, BibLaTeX, biber, or something equivalent
- equations and references to those equations in the text
- the use of at least two different math environments (e.g., `\begin{equation}`, `\begin{align}`, `\begin{multline}`, `\begin{gather}`, `\begin{subequations}`)

The math environments must be properly used -- e.g., do not use the `multline` environment for a short equation such as Ax = b.

Also provide a table, code listing, or algorithm. The table, code listing, or algorithm must be properly captioned, well-formatted, and referenced in the text.

<u>Exercise 2:</u> Index Mapping (50 pts)

In this exercise you are asked to write the same program in both C and Fortran. This program will ask the user for some input, allocate an array, then assign default values to that array.

The default values for the array will be determined from the Z-order curve (or Morton Ordering). The Z-order curve is a type of “space-filling curve”. Space filling curves are very useful tools in computer science. They allow one to map a multi-dimensional index to a 1D index. Several kinds of space filling curves exist (e.g. Hilbert, Serpinksi, etc.). The Z-order curve is one of the more simple space filling curves to implement and relies on binary arithmetic. An illustration of the Z-order curve is given below:

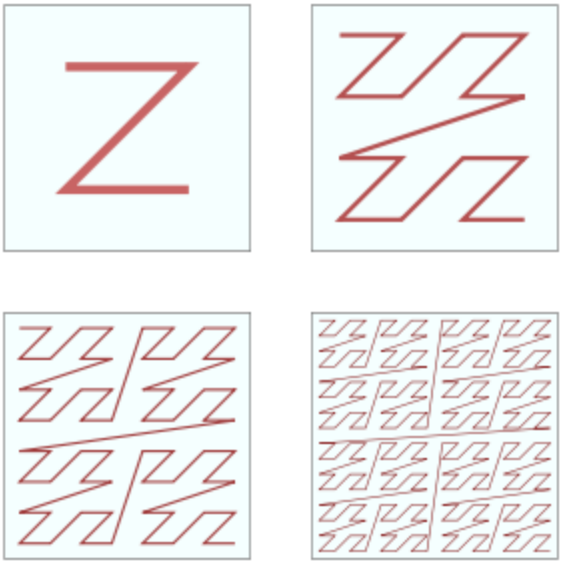


Illustration of Z-order curves [1]

In computer science, the utility of space filling curves comes from the property that 1D indexes close in value are often close “spatially” in the multi-dimensional coordinate system. Taken in the context of the memory hierarchy in computer architectures (which we will discuss in detail later this semester) this can result in improved performance. In fact space filling curves provide the basis for many algorithms that are described as “cache oblivious”. Space filling curves are also often implemented with recursion and can be traversed as a tree; meaning algorithm costs are $O(\log N)$ rather than $O(N)$. Wow!

For example, given a 2D array, instead of using two index variables [i][j] (i,j loops from 1->N), the data can be visited using a single index [s] (s loops from 1->N^2). This new index follows a specific order so that the mapping is a ‘one-to-one’ process.

A simple way to understand the Z-order is to consider it recursively (divide and visit):

- visit top-left part of the (sub)array
- visit top-right part of the (sub)array
- visit bottom-left part of the (sub)array
- visit bottom-right part of the (sub)array

Part A:

In the first part of this exercise, you are asked to write a C program which will allocate memory and call a C function to initialize a 2D array. Your code should be able to:

i) Process the command line to read the first argument `N`. `N` is the dimension for your 2D square array.

ii) `N` could be 2, 4, 8 or 16. Check if the given value is among the choices. If not, print an error message and exit.

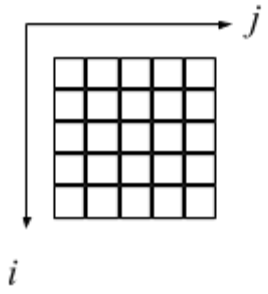
iii) With a valid `N` value, your array should contain integers from 1 to N^2 in a z-order curve format.

iv) In your C program, print out the array returned by the C function and check if it is what you expected.

In this exercise, if the user input is `N=4` the array should be printed in the following format:

```
A=[ 1  2  5  6
    3  4  7  8
    9 10 13 14
   11 12 15 16]
```

Note that this implies the following about the `(i,j)` coordinates.



Part B:

In the second part of this exercise, you are asked to write a FORTRAN program which will allocate memory and call a FORTRAN function to initialize a 2D array. Your code should be able to:

i) Process the command line to read the first argument `N`. `N` is the dimension for your 2D square array.

ii) `N` could be 2, 4, 8 or 16. Check if the given value is among the choices. If not, print an error message and exit.

iii) With a valid `N` value, in the function, allocate the array and assign integers from 1 to N^2 in a z-order curve format.

iv) In your FORTRAN program, print out the array returned by the FORTRAN function call and check if it is what you expected. The array should be printed as it was in Part A.

FORTRAN code for determining the Z-order index from the `(i,j)` coordinates.

```
FUNCTION z_order2D(i,j) RESULT(z)
  INTEGER(4),INTENT(IN) :: i,j
  INTEGER(4) :: z

  INTEGER(4) :: k,ik,jk,zk,ord
  INTEGER(1) :: ib(32),jb(32),zb(64)

  ib=0; jb=0; zb=0;
  ik=i-1; jk=j-1
  DO k=1,32
    ib(k)=INT(MOD(ik,2),1); ik=ik/2
    jb(k)=INT(MOD(jk,2),1); jk=jk/2
  ENDDO

  zk=1
  DO k=1,32
    zb(zk)=ib(k)
    zb(zk+1)=jb(k)
    zk=zk+2
  ENDDO

  z=1; ord=1
  DO k=1,64
    z=z+zb(k)*ord
    ord=ord+ord
  ENDDO

ENDFUNCTION
```

[1] More about Z-order curves (from wikipedia): https://en.wikipedia.org/wiki/Z-order_curve

Extra credit (+10 points on this assignment) if you provide an implementation for computing the z-order index using bit manipulation.

Exercise 3 (30 pts): Mixed languages

- *Part A:* (15 pts) Using your solution for exercise 2, modify the implementation so that the C program calls the Fortran function to initialize the array. Provide the command(s) you used to compile the program
- *Part B:* (15 pts) Using your solution for exercise 2, modify the implementation so that the Fortran program calls the C function to initialize the array. Provide the command(s) you used to compile the program.

Deliverables

Please pack all of your files for this homework into a single tar(zip) file.
Name it as `<user>_HW1.tgz` and submit it through canvas.
Your tar file for the homework should include:

- For Ex1: ***<user>*_Ex1.tex** and ***<user>*_Ex1.bib**
- For Ex2: ***<user>*_Ex2A_main.c *<user>*_Ex2A.c**
  ***<user>*_Ex2B_main.f90 *<user>*_Ex2B.f90**
- For Ex3: ***<user>*_Ex3A_main.c *<user>*_Ex3A.f90**
  ***<user>*_Ex3B_main.f90 *<user>*_Ex3B.c**

Note: To create a zipped tar file for all files in a directory the command is

```
$ tar -cvzf <user>_HW1.tgz ./*
```