AA 372

Homework I

1. **Read/Write is much slower than computation:** Write a simple program (in whichever language you are comfortable, Fortran or C) to verify that read/write to disk is much slower than computation. You can write a code corresponding to the following pseudocode (also see the accompanying Fortran code *read_2d_data.f90*):

```
do i=1,imax
do j=1,jmax
    a(i,j) = float(i-j)/float(i+j)
    write(11,*) a(i,j)
enddo
enddo
```

   Find the wall-time it takes (the accompanying Fortran code has commands to measure wall time; please look up analogous commands for C if that is your preferred language) to run this code. Compare this time with the case when the write statement is commented. Why does commenting out the write statement make the code run much faster?

   Now comment the write statement and compare the **execution speed of different mathematical operations.** Instead of *a(i,j) = float(i-j)/float(i+j)* try *a(i,j) = float(i-j) + float(i+j)* and *a(i,j) = float(i-j) * float(i+j)*. How does the wall-time change? Now try more complicated functions *a(i,j) = (float(i-j)/float(i+j))**0.5*, *a(i,j) = float(i-j)**(float(i)/float(j))*, and so on. Compare the wall-times for each case. Can you explain which operations took longer and why?

   See whether changing the order of i and j loops in the code makes any difference? How does the time taken depend on imax, jmax? Plot wall-time as a function of imax (=jmax) for both cases, the inner i loop and the inner j loop. This should teach you something about different cache sizes, whether 2D arrays are saved in row major or column major fashion in memory, and why it is **faster to use the data stored adjacent in the memory.**

   Compiler optimizations: Try compiling the code with -O0 (compiler optimizations turned off) flag and compare with the time it takes to run with the default compiler options, and with -O2 and -O3. These optimization flags are available on most compilers; read the man pages for gcc or ifort or gfortran (depending on the compiler that you are using) to know more.

2. **Bonus:** Did you know that the run-time for if-then statements (in general control statements) depends on whether the condition is predictable (true/false)? This is because the microprocessor speculatively executes the predicted branch. If there is misprediction the partially executed steps are discarded and the correct branch is executed with a delay. This strategy is followed because its faster to compute than wait to check if the condition is met. Read more at the link provided ("branch prediction") in references. Write a simple program and show that indeed the run-time depends on the sequence of true/false in the conditional statement.