OpenMP and AVX Report

Problem1:

Try to speed up the following code using OpenMP.
You can only use one parallel region, and make sure the average is the same as the one calculated using the sequential version.

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

#define NRA 15000                 /* number of rows in matrix A */
#define NCA 15000                 /* number of columns in matrix A */

double ** create_2d_matrix(int columns, int rows)
{
       double ** mat = new double*[rows];
       for (int i = 0; i < rows; i++)
       {
              mat[i] = new double[columns];
       }
       return mat;
}

int main ()
{
       int i, j, k;
       double **a = create_2d_matrix(NRA,NCA);

       double time1 = omp_get_wtime();
       for (i=0; i<NRA; i++)
              for (j=0; j<NCA; j++)
                     a[i][j]= 1;

       for (i=0; i<NRA; i++)
              for (j=0; j<NCA; j++)
                     a[i][j] *= (i+j)%13;
       double sum = 0;
       for (i=0; i<NRA; i++)
              for (j=0; j<NCA; j++)
                     sum += a[i][j];
       double average = sum/(NRA*NCA);
       printf("average = %6.2f, time = %6.2f",average,omp_get_wtime() - time1);
}
```

Problem2: Prime Numbers

There are 41538 prime numbers in the range [2,500000], Write a sequential program to count how many prime numbers are in the previous range and make sure your count is exactly the previous number, Then write an OpenMP parallel program with different number of threads (2,4,8,12) to speed up the prime numbers counting process, Use OpenMP schedule clause to try **static** and **dynamic** scheduling with different **chunk** sizes (1,100,1000), and include the following in your report:

a) The code including serial and OpenMP parallel programs.
b) Report the time consumed for all different combinations of scheduling chunk sizes and number of threads in two tables, one for static scheduling and the other for dynamic scheduling, and explain using your intuition and understanding the best and worst OpenMP timings, compared with the sequential code timing.
    a. Can you notice the pattern for the time for chunk size = 1 in static and dynamic scheduling?
    b. For threads count that are less than 4, usually dynamic will be faster than static by 2x, Can you explain why?

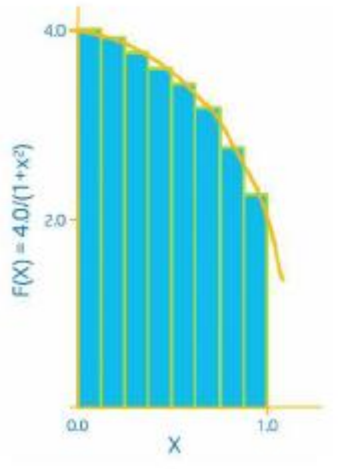
Note: Any parallel code should be faster than the sequential code, and make sure your parallel code calculates the same prime number count as the sequential code.

Problem3: The PI Problem:

Write a program to calculate the value of PI using OpenMP, Pi can be calculated using numerical integration using the following formula:

$$PI = \int_0^1 \frac{4}{1+x^2} dx$$

Your program should handle false sharing and race conditions, and should only use ideas discussed in the experiment, to calculate the numerical integration above, you'll need to calculate the area under the following curve by summing the rectangular areas shown below, use a step size (rectangle width) equals to 0.000000001 to approximate the integration value, and compare your output with the normal PI value.



Problem4: Cosine using AVX:

Knowing that the Taylor expansion of cos(x) is

$$cos(x) = 1 - x^2/2! + x^4/4! - x^6/6! + \dots$$

Write a function that takes x as an input array and return an array of Cos(x), then accelerate this function using AVX

Refer to Experiment17 code, which implements the sine using Taylor expansion with the following formula:

$$sin(x) = x - x^3/3! + x^5/5! - x^7/7! + \dots$$