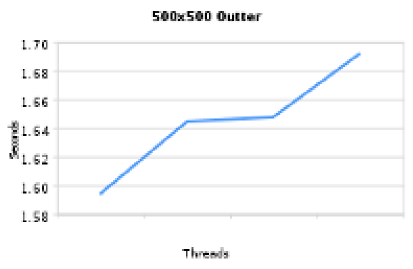
Benjamin Rhodes
Project 4

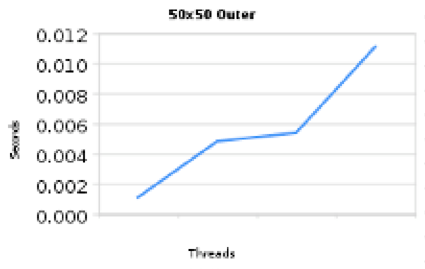
Code simply adds the required pragmas to the original program. Also switched the k and i index for the sum calculation.

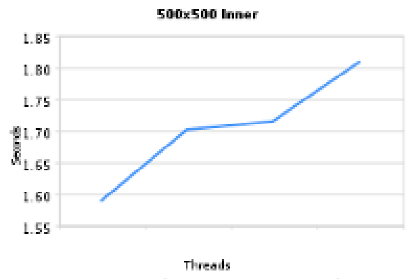
The software run much slower that expected using even a small number of threads. I think that the problem size needs to be much larger in order to take advantage of multiple processors.

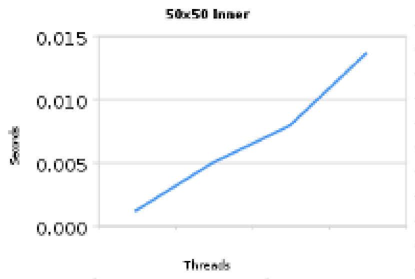
All bellow data is average of 3 runs.

	Time in seconds.			
Not Parallelized				
50x50	0.00065			
500x500	1.01275			
Parallelized Outer	1	4	8	16
50x50	0.001125	0.004902	0.005387	0.011124
500x500	1.59468	1.645194	1.647669	1.691891
Parallelized Inner				
50x50	0.001207	0.005036	0.007988	0.013687
500x500	1.590473	1.701817	1.716132	1.809129









outer_paralle.c

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

#define M 500
#define N 500
#define THREADS 8

int main(int argc, char *argv)
{
  omp_set_num_threads(THREADS);

  int i, j, k;
  double sum;
  double **A, **B, **C;
  A = malloc(M*sizeof(double *));
  B = malloc(M*sizeof(double *));
  C = malloc(M*sizeof(double *));

  for (i = 0; i < M; i++) {
    A[i] = malloc(N*sizeof(double));
    B[i] = malloc(N*sizeof(double));
    C[i] = malloc(N*sizeof(double));
  }

  double start, end;
  for (i = 0; i < M; i++) {
    for (j = 0; j < N; j++) {
      A[i][j] = j*1;
      B[i][j] = i*j+2;
      C[i][j] = j-i*2;
    }
  }

  start = omp_get_wtime();

  #pragma omp parallel shared(A, B, C) private(i, j, k)
  {
    #pragma omp parallel for
    for (i = 0; i < M; i++)
    {
      for (j = 0; j < N; j++)
      {
        sum = 0;
        for (k=0; k < M; k++) {
          sum += A[i][k]*B[k][j];
        }

        C[i][j] = sum;
      }
    }
  }

  end = omp_get_wtime();

  printf("Time of computation: %f\n", end-start);
}
```

inner_parallel.c

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

#define M 50
#define N 50
#define THREADS 8

int main(int argc, char *argv)
{
  omp_set_num_threads(THREADS);

  int i, j, k;
  double sum;
  double **A, **B, **C;
  A = malloc(M*sizeof(double *));
  B = malloc(M*sizeof(double *));
  C = malloc(M*sizeof(double *));

  for (i = 0; i < M; i++) {
```

```c
    A[i] = malloc(N*sizeof(double));
    B[i] = malloc(N*sizeof(double));
    C[i] = malloc(N*sizeof(double));
  }

  double start, end;
  for (i = 0; i < M; i++) {
    for (j = 0; j < N; j++) {
      A[i][j] = j*1;
      B[i][j] = i*j+2;
      C[i][j] = j-i*2;
    }
  }

  start = omp_get_wtime();


  for (i = 0; i < M; i++)
  {
    #pragma omp parallel shared(A, B, C, i) private(j, k)
    {
      #pragma omp parallel for
      for (j = 0; j < N; j++)
      {
        sum = 0;
        for (k=0; k < M; k++) {
          sum += A[i][k]*B[k][j];
        }

        C[i][j] = sum;
      }
    }
  }

  end = omp_get_wtime();

  printf("Time of computation: %f\n", end-start);
}
```