***Team Yugi-Moto!***

***Padraic Rowley:12304905***

***Augustas Baublys:12302965***

**Optimization of matrix multiplication in C**

Strassen’s Algorithm

One optimization we found online was the Strassen algorithm. It divides a large matrice into a series of smaller easily computable matrices. It was a mathematical formulae used to optimize matrix multiplication as follows.

1. Partition A, B and and C into 4 equal parts: A = A11 A12 A21 A22 , B = B11 B12 B21 B22 , C = C11 C12 C21 C22

2. Evaluate the intermediate matrices: M1 = (A11 + A22) (B11 + B22) , M2 = (A21 + A22) B11 ,M3 = A11 (B12 – B22) ,M4 = A22 (B21 – B11) ,M5 = (A11 + A12) B22 ,M6 = (A21 – A11) (B11 + B12) ,M7 = (A12 – A22) (B21 + B22)

3. Construct C using the intermediate matrices: C11 = M1 + M4 – M5 + M7 , C12 = M3 + M5, C21 = M2 + M4 ,C22 = M1 – M2 + M3 + M6

However as well as this, the algorithm could only be used in certain matrix parameters as well as would prove more inefficient for smaller matrices.

1. Partition A and B into quarter matrices as described above.

2. Compute the intermediate matrices: 1. If the sizes of the matrices are greater than a threshold value, multiply them recursively using Strassen's algorithm. 2. Else use the traditional matrix multiplication algorithm.

3. Construct C using the intermediate matrices.

Use of recursion

The division into quadrants was done through recursion rather than than a massive set of nested for-loops to ensure any unnecessary variables were not recreated. This was a simple,minute change that would only really take effect in the case of massive arrays, however since the strassen algorithm’s only used for larger matrices in the first place it can make a notable difference at run time.

Filter for smaller numbers

Since the Strassen Algorithm is only faster with a larger matrice and can sometimes have problems with odd numbers , a filter was put in place to multiply normally if the strassen method was going to be more of a hindrance than an assistance. This took the form of a simple “if” statement at the start of the multiplication to check its size.

**Optimization of matrix multiplication in C**

Use of O3 compiler

Although not technically a code improvement, it undoubtedly makes an exponential difference to run speed. For one it changes multiplication and division by 2 to a simple single bit shift action. This is extremely useful to the Strassen method which quarters each matrix instance by dividing its column and row length by 2.

Assignment of matrix from 4 quadrants

When the recursion completes, the quadrants of each matrix created from the initial matrix must be recombined and returned as 1 single matrix. This was done in 1 single for-loop that only only ran for a quarter of the amount of numbers to be computed, this was done by adding each quadrant using a “I”and “j” in the loops as differences from the starting points of each quadrant being inserted into the new matrix.

**Failed Optimizations**

Threading of Matrix creation/”real” and “imag”

This was a route we planned on going down as it ensured data would not be corrupted between the 2 thread but was decided that it would do more harm than good to run time.

Removal of matrix C

We considered removing matrix C and instead modifying matrix A with reference to B to save time created an entire new matrix of equivelant size but once the Strassen method was agreed as our main improvement that proved impossible.

**Potential further Optimizations**

Threading quadrants

One major improvement that could have been made but was unfeasible due to personal constraints was to thread each quadrants computation. This would only be helpful at a massive matrix size since the quadrants would exponentially increase in number the smaller they got. To put all of these quadrants in parallel would speed up the whole calculation as each level of the recursive function could collapse in unison and compete at a speed referable to the amount of recursions rather than the number of quadrant matrices created. As seen in our code we have a thread attempt made for calculating the added quads of each matrix but could not find a solution to return each quad safely and uncorrupted.