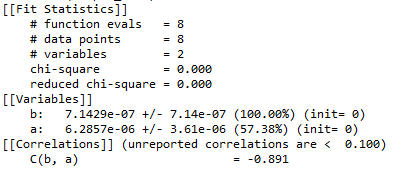
Code is attached as separate .py and .cpp files.

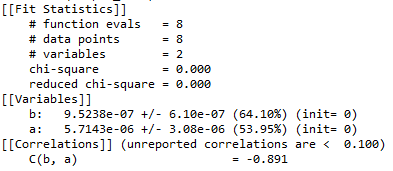
A measure of the communication time was taken by sending an array of integers with length varied between n=1 and n=8 from one processor to another and then back again. These transmissions were performed 100 times and averaged, then fit to a line to determine the intercept and slope.

For communication between processors on one node:



The time to ping pong a packet of length 0 on the same node would then be 0.71 micro seconds, with an increase of 36 micro seconds per integer sent.

For communication between nodes:



The time to ping pong a packet of length 0 on different nodes would then be 0.95 micro seconds, with an increase of 57 micro seconds per integer sent.

An attempt was made to perform a matrix transposition via the MPI\_Alltoall() method. The time for this communication with a matrix of n = 4 was t = 130 micro seconds or 0.13 milli seconds. This function should be equivalent to n^2 send/receives of a single integer, which would take about 70 micro seconds. However, the results of the transpose did not appear to be correct in my implementation, though all documentation online indicates that the function does behave as a transpose operation. The number of floating point operations that could have been performed in the measured transpose time would be approximately 390 thousand assuming a clock rate of 3GHz and 1 operation per clock cycle. The FFT algorithm is estimated as taking 4\*N\*log\_2(N)-6N+8 at the lower bound. Again assuming a 3GHz clock rate and 1 operation per clock cycle, a vector of N=4 would take ~5 nano seconds to perform an FFT on. This shows us that even if the communications went at an ideal speed of 70 micro seconds, the processors can perform their calculations 14000 times faster than MPI can send the data.