High Performance Computing

Assignment 3

Theory Questions

1. **Key Differences between SIMD and MIMD**

SIMD	**MIMD**
Single Instructions Multiple Data Streams	Multiple Instructions Multiple Data Streams
Same instruction is executed simultaneously on all CPUs but on different data streams (as the name says lol)	Capable of executing multiple instructions on multiple data streams (again, as the name says)
Suitable for scientific computing because they involve lot of vector and matrix operations – the vector can be divided into multiple data sets and each data stream executes operations on one particular data set	Suitable for any kind of application because there are separate instructions and data streams
Processors work simultaneously	Processors work asynchronously
EXAMPLE: Thinking very simple, let's say I have a lot of potatoes to peel and cut. Using SIMD, there are different data streams, where each PE will be given a few potatoes and they will all peel and cut it at the same time.	**EXAMPLE**: With MIMD, I can have a few data streams that will peel my potatoes while some others will cut my potatoes. So I will divide the potatoes for peeling first and then as they are done they are sent to cut them
ANOTHER EXMAPLE: Like we did in lab, the vector matrix addition. SIMD would perform the addition operation on all the elements of matrix simultaneously with a single instruction	**ANOTHER EXMAPLE**: Using MIMD, we can divide the matrix into smaller subsets on which the operation is performed and then add up the results

2. **OpenMP is a shared memory model (threads) whereas MPI is a distributed memory model (processes). If you wanted to have global variables in MPI how you create them?**

   MPI does not have shared memory so sharing 'global variables' like in OpenMP does not work the same way. When using MPI, data is unique to each processor. To transfer same data over all the processes, we use MPI existing functions. Like MPI_int, MPI_BCast, but these are for when lets say we are sharing the Vector Matrix or something at the beginning of the program. But global variable that can be shared across all processes? Not sure if that is possible. Each process is a new instance of a program, so the global variable that exists in that process is not exactly global anymore, it is instead just global to THAT process.

I guess, if we had to, we could send the variable back to process0 every time it was updated and BCast to all processes again and then take average of that value with the one they already had and overrise the value? This just seems overly complicated and useless honestly.
It's almost like all processes trying to access the variable from the globally available memory but it would be overly complicated.

3. **Discuss strategies for detecting deadlock, particularly in MPI**
Deadlock would be caused if 2 processors or more are trying to access the same resource at the same time and the code does not take care of this. One way to avoid this would be how we take care of deadlock even in simple computer OS deadlock prevention.

We could prioritize. Now the factors used to prioritize may differ application to application, but even the ones where there is no such 'priority'. We can simply say that processor 3 gets more priority than processor 4.

We can do the Hold and Wait. Where we allocate the resource and wait till execution it finished by THAT process. This will definitely not be efficient because it many processes can be waiting for a long time.

Deadlock can be caused in MPI from MPI_Send and MPI_Recv. One way to solve this is to reverse the order of execution for one of the conditionals.

We need to have a resource/application taking care of deadlock like a banker's algorithm but for the MPI.

4. **Compare the advantages and disadvantages of a different sparse matrix formats: Dictionary of Keys, List of lists, compressed sparse rows/columns. Give an example**.
Sparse Matrix is when large number of elements are zero.

Considering the sparse matrix-
Ex – {[1,0,0,1], [0,2,0,0], [0,0,3,0], [4,0,0,5]}

Dictionary of Keys (DOK)
All non zero elements are stored in a dictionary where key represents the indices and value is the non-zero element. The format is not useful for accessing specific elements and/or iterating over in the sequential order

Ex – the above matrix will be stored as
(0,0) -> 1
(0,3) -> 1
(1,1) -> 2
(2,2) -> 3

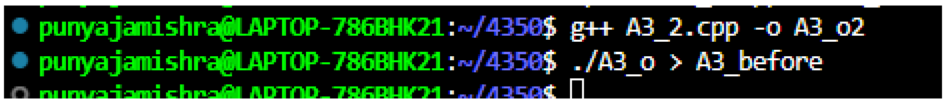
(3,0) -> 4
(3,3) -> 5

List of lists (LIL)
As the name suggests, it is a List of lists, where each row is stored as a list and each entry in the list. The entries are stored as column index to be able to look up the values.
Ex – [[(0,1), (3,1)], [(1,2)], [(2,3)], [(0,4),(3,5)]]

Compressed sparse rows/columns
All the no zero elements are stored in 3 arrays – values, column index and row index.
The array of Values and Column Index are of length of the 'number of non zero elements' in the matrix.
The row index array stores the index where the first non zero element occurs.
V = [1,1,2,3,4,5]
Col_index = [0,3,1,2,0,3]
Row_index = [0,1,2,3]

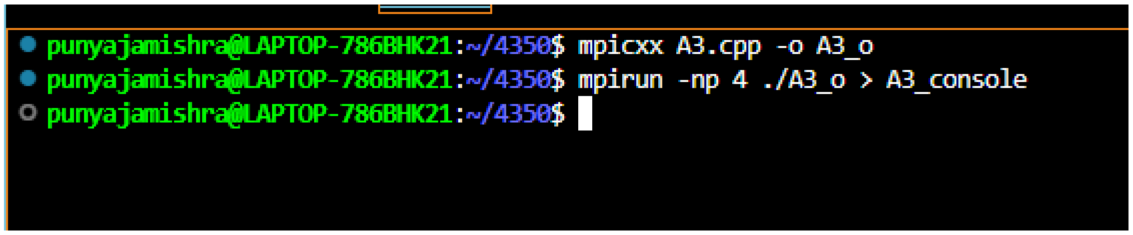
Programming Questions

Before implementing MPI.



```
● punyajamishra@LAPTOP-786BHK21:~/4350$ g++ A3_2.cpp -o A3_o2
● punyajamishra@LAPTOP-786BHK21:~/4350$ ./A3_o > A3_before
○ punyajamishra@LAPTOP-786BHK21:~/4350$ 
```

After implementing the MPI. I have attached the output file to the submission as well, it is called A3_console. I know I did mess up a bit. Its not exactly correct.



```
● punyajamishra@LAPTOP-786BHK21:~/4350$ mpicxx A3.cpp -o A3_o
● punyajamishra@LAPTOP-786BHK21:~/4350$ mpirun -np 4 ./A3_o > A3_console
○ punyajamishra@LAPTOP-786BHK21:~/4350$ 
```

First, let me tell what I wished to implement. I wanted to break the matrix into 4 squares and send each to a process, in which they do the calculations of the while loop. And then its all send back and gathered and implemented one last time (this would be needed just for the common edge of row and column) and that's it. Seems simple if you think about it right?

This is what I tried to implement. I split the matrix into small 'chunks', where the chunk size is SIZE/2 because at the end of the day a matrix size is row/2 and column/2. Since we are using an even square matrix, chunk_size=SIZE/2. I did call MPI_Abort for odd number of matrix because I don't think my code will execute for odd number of SIZE, it might send segmentation fault in the MPI.

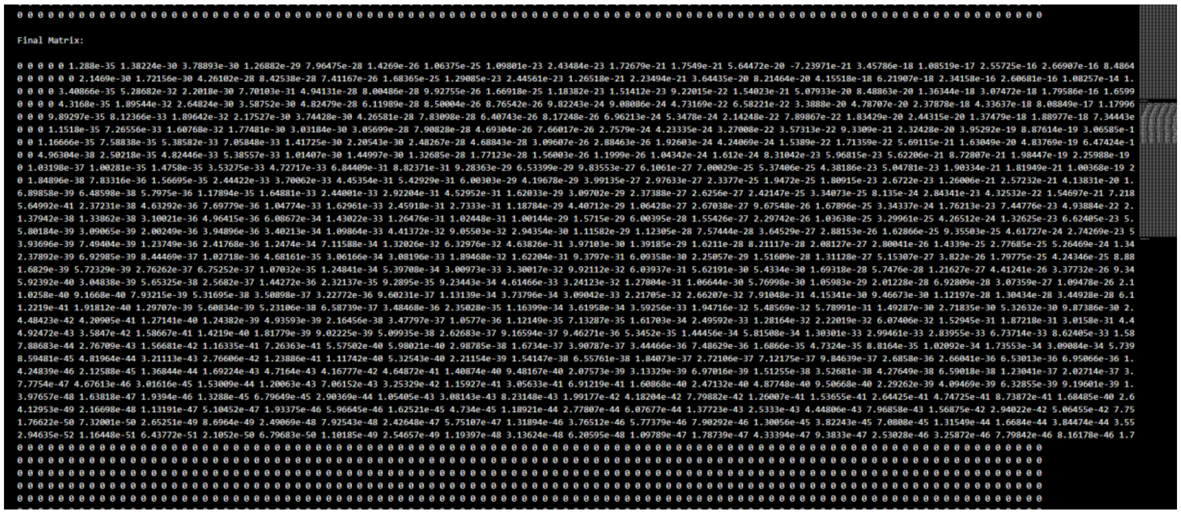
I defined the matrix row start and col start for each process separately to ensure no 2 process tries to access same memory location. Then I distributed the matrix among the 4 processes. And after that all the calculations were done.

As you will see in the output, the calculations are happening, but I think the way I tried to scatter the matrix was not exactly correct? Not all processes got the desired matrix I wanted.

OR

Since I sent the matrix chunks in the beginning when the only cell with a value is in one part of the matrix, therefore, the values and all calculations happened only in that process. And all other processes had 0 values, therefore, no calculations per say happened.

BEFORE:

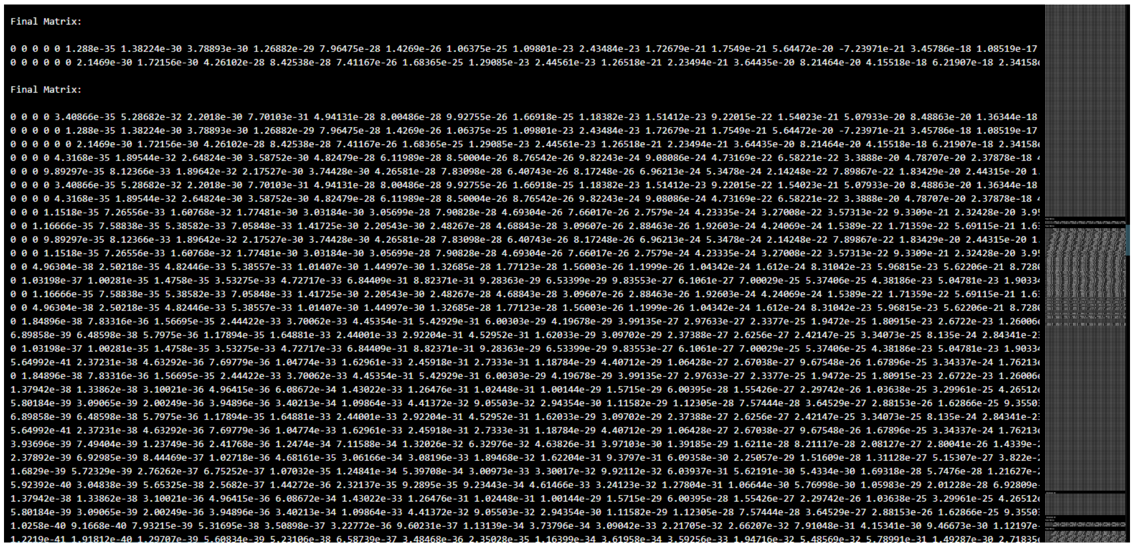


```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Final Matrix:

0 0 0 0 0 1.288e-35 1.38224e-30 3.78893e-30 1.26882e-29 7.96475e-28 1.4269e-26 1.06375e-25 1.09801e-23 2.43484e-23 1.72679e-21 1.7549e-21 5.64472e-20 -7.23971e-21 3.45786e-18 1.08519e-17 2.55725e-16 2.66907e-16 8.4864
0 0 0 0 0 0 2.1469e-30 1.72156e-30 4.26102e-28 8.42538e-28 7.41167e-26 1.68365e-25 1.29085e-23 2.44561e-23 1.26518e-21 2.23494e-21 3.64435e-20 8.21464e-20 4.15518e-18 6.21907e-18 2.34158e-16 2.60681e-16 1.08257e-14 1.
0 0 0 0 3.40866e-35 5.28682e-32 2.2018e-30 7.70103e-31 4.94131e-28 8.00486e-28 9.92755e-26 1.66918e-25 1.18382e-23 1.51412e-23 9.22015e-22 1.54023e-21 5.07933e-20 8.48863e-20 1.36344e-18 3.07472e-18 1.79586e-16 1.6599
0 0 0 0 4.3168e-35 1.89544e-32 2.64824e-30 3.58752e-30 4.82479e-28 6.11989e-28 8.50004e-26 8.76542e-26 9.82243e-24 9.08086e-24 4.73169e-22 6.58221e-22 3.3888e-20 4.78707e-20 2.37878e-18 4.33637e-18 8.08849e-17 1.17996
0 0 0 9.89297e-35 8.12366e-33 1.89642e-32 2.17527e-30 3.74428e-30 4.26581e-28 7.83098e-28 6.40743e-26 8.17248e-26 6.96213e-24 5.3478e-24 2.14248e-22 7.89867e-22 1.83429e-20 2.44315e-20 1.37479e-18 1.88977e-18 7.34443e
0 0 0 1.1518e-35 7.26556e-33 1.60768e-32 1.77481e-30 3.03184e-30 3.05699e-28 7.90828e-28 4.69304e-26 7.66017e-26 2.7579e-24 4.23335e-24 3.27008e-22 3.57313e-22 9.3309e-21 2.32428e-20 3.95292e-19 8.87614e-19 3.06585e-1
0 0 1.16666e-35 7.58838e-35 5.38582e-33 7.05848e-33 1.41725e-30 2.20543e-30 2.48267e-28 4.68843e-28 3.09607e-26 2.88463e-26 1.92603e-24 4.24069e-24 1.5389e-22 1.71359e-22 5.69115e-21 1.63049e-20 4.83769e-19 6.47424e-1
0 0 4.96304e-38 2.50218e-35 4.82446e-33 5.38557e-33 1.01407e-30 1.44997e-30 1.32685e-28 1.77123e-28 1.56003e-26 1.1999e-26 1.04342e-24 1.612e-24 8.31042e-23 5.96815e-23 5.62206e-21 8.72807e-21 1.98447e-19 2.25988e-19
0 1.03198e-37 1.00281e-35 1.4758e-35 3.53275e-33 4.72717e-33 6.84409e-31 8.82371e-31 9.28363e-29 6.53399e-29 9.83553e-27 6.1061e-27 7.00029e-25 5.37406e-25 4.38186e-23 5.04781e-23 1.90334e-21 1.81949e-21 1.00368e-19 2
0 1.84896e-38 7.83316e-36 1.56695e-35 2.44422e-33 3.70062e-33 4.45354e-31 5.42929e-31 6.00303e-29 4.19678e-29 3.99135e-27 2.97633e-27 2.3377e-25 1.9472e-25 1.80915e-23 2.6722e-23 1.26006e-21 2.57232e-21 4.13831e-20 1.
6.89858e-39 6.48598e-38 5.7975e-36 1.17894e-35 1.64881e-33 2.44001e-33 2.92204e-31 4.52952e-31 1.62033e-29 3.09702e-29 2.37388e-27 2.6256e-27 2.42147e-25 3.34073e-25 8.135e-24 2.84341e-23 4.32532e-22 1.54697e-21 7.218
5.64992e-41 2.37231e-38 4.63292e-36 7.69779e-36 1.04774e-33 1.62961e-33 2.45918e-31 2.7333e-31 1.18784e-29 4.40712e-29 1.06428e-27 2.67038e-27 9.67548e-26 1.67896e-25 3.34337e-24 1.76213e-23 7.44776e-23 4.93884e-22 2.
1.37942e-38 1.33862e-38 3.10021e-36 4.96415e-36 6.08672e-34 1.43022e-33 1.26476e-31 1.02448e-31 1.00144e-29 1.5715e-29 6.00395e-28 1.55426e-27 2.29742e-26 1.03638e-25 3.29961e-25 4.26512e-24 1.32625e-23 6.62405e-23 5.
5.80184e-39 3.09065e-39 2.00249e-36 3.94896e-36 3.40213e-34 1.09864e-33 4.41372e-32 9.05503e-32 2.94354e-30 1.11582e-29 1.12305e-28 7.57444e-28 3.64529e-27 2.88153e-26 1.62866e-25 9.35503e-25 4.61727e-24 2.74269e-23 5
3.93696e-39 7.49404e-39 1.23749e-36 2.41768e-36 1.2474e-34 7.11588e-34 1.32026e-32 6.32976e-32 4.63826e-31 3.97103e-30 1.39185e-29 1.6211e-28 8.21117e-28 2.08127e-27 2.80041e-26 1.4339e-25 2.77685e-25 5.26469e-24 1.34
2.37892e-39 6.92985e-39 8.44469e-37 1.02718e-36 4.68161e-35 3.06166e-34 3.08196e-33 1.89468e-32 1.62204e-31 9.3797e-31 6.09358e-30 2.25057e-29 1.51609e-28 1.31128e-27 5.15307e-27 3.822e-26 1.79775e-25 4.24346e-25 8.88
1.6829e-39 5.72329e-39 2.76262e-37 6.75252e-37 1.07032e-35 1.24841e-34 5.39708e-34 3.00973e-33 3.30017e-32 9.92112e-32 6.03937e-31 5.62191e-30 5.4334e-30 1.69318e-28 5.7476e-28 1.21627e-27 4.41241e-26 3.37732e-26 9.34
5.92392e-40 3.04838e-39 5.65325e-38 2.5682e-37 1.44272e-36 2.32137e-35 9.2895e-35 9.23443e-34 4.61466e-33 3.24123e-32 1.27804e-31 1.06644e-30 5.76998e-30 1.05983e-29 2.01228e-28 6.92809e-28 3.07359e-27 1.09478e-26 2.1
1.0258e-40 9.1668e-40 7.93215e-39 5.31695e-38 3.50898e-37 3.22772e-36 9.60231e-37 1.13139e-34 3.73796e-34 3.09042e-33 2.21705e-32 2.66207e-32 7.91048e-31 4.15341e-30 9.46673e-30 1.12197e-28 1.30434e-28 3.44928e-28 6.1
1.2219e-41 1.91812e-40 1.29707e-39 5.60834e-39 5.23106e-38 6.58739e-37 3.48468e-36 2.35028e-35 1.16399e-34 3.61958e-34 3.59256e-33 1.94716e-32 5.48569e-32 5.78991e-31 1.49287e-30 2.71835e-30 5.32632e-30 9.87386e-30 2.
4.48423e-42 4.20905e-41 1.27141e-40 1.24382e-39 4.93593e-39 2.16456e-38 3.47797e-37 1.0577e-36 1.12149e-35 7.13287e-35 1.61703e-34 2.49592e-33 1.28164e-32 2.22019e-32 6.07406e-32 1.52945e-31 1.87218e-31 3.0158e-31 4.4
4.92472e-43 3.5847e-42 1.58667e-41 1.4219e-40 1.81779e-39 9.02225e-39 5.09935e-38 2.62683e-37 9.16594e-37 9.46271e-36 5.3452e-35 1.44456e-34 5.81508e-34 1.30301e-33 2.99461e-33 2.83955e-33 6.73714e-33 8.62405e-33 1.58
7.88683e-44 2.76709e-43 1.56681e-42 1.16335e-41 7.26363e-41 5.57502e-40 5.98021e-40 2.98785e-38 1.6734e-37 3.90787e-37 3.44466e-36 7.48629e-36 1.6866e-35 4.7324e-35 8.8164e-35 1.02092e-34 1.73553e-34 3.09084e-34 5.739
8.59481e-45 4.81964e-44 3.21113e-43 2.76606e-42 1.23886e-41 1.11742e-40 5.32543e-40 2.21154e-39 1.54147e-38 6.55761e-38 1.84073e-37 2.72106e-37 7.12175e-37 9.84639e-37 2.6858e-36 2.66041e-36 6.53013e-36 6.95066e-36 1.
4.24839e-46 2.12588e-45 1.36844e-44 1.69224e-43 4.7164e-43 4.16777e-42 4.64872e-41 1.40874e-40 9.48167e-40 2.07573e-39 3.13329e-39 6.97016e-39 1.51255e-38 3.52681e-38 4.27649e-38 6.59018e-38 1.23041e-37 2.02714e-37 3.
7.7754e-47 4.67613e-46 3.01616e-45 1.53009e-44 1.20063e-43 7.06152e-43 3.25329e-42 1.15927e-41 3.05633e-41 6.91219e-41 1.60868e-40 2.47132e-40 4.87748e-40 9.50668e-40 2.29262e-39 4.09469e-39 6.32855e-39 9.19601e-39 1.
3.97657e-48 1.63818e-47 1.9394e-46 1.3288e-45 6.79649e-45 2.90369e-44 1.05405e-43 3.08143e-43 8.23148e-43 1.99177e-42 4.18204e-42 7.79882e-42 1.26007e-41 1.53655e-41 2.64425e-41 4.74725e-41 8.73872e-41 1.68485e-40 2.6
4.12953e-49 2.16698e-48 1.13191e-47 5.10452e-47 1.93375e-46 5.96645e-46 1.62521e-45 4.734e-45 1.18921e-44 2.77807e-44 6.07677e-44 1.37723e-43 2.5333e-43 4.44806e-43 7.96858e-43 1.56875e-42 2.94022e-42 5.06455e-42 7.75
1.76622e-50 7.32001e-50 2.65251e-49 8.6964e-49 2.49069e-48 7.92543e-48 2.42648e-47 5.75107e-47 1.31894e-46 3.76512e-46 5.77379e-46 7.90292e-46 1.30056e-45 3.82243e-45 7.0808e-45 1.31549e-44 1.6684e-44 3.84474e-44 3.55
2.94635e-52 1.16448e-51 6.43772e-51 2.1052e-50 6.79683e-50 1.10185e-49 2.54657e-49 1.19397e-48 3.13624e-48 6.20595e-48 1.09789e-47 1.78739e-47 4.33394e-47 9.3833e-47 2.53028e-46 3.25872e-46 7.79842e-46 8.16178e-46 1.7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

AFTER:



```
Final Matrix:

0 0 0 0 0 1.288e-35 1.38224e-30 3.78893e-30 1.26882e-29 7.96475e-28 1.4269e-26 1.06375e-25 1.09801e-23 2.43484e-23 1.72679e-21 1.7549e-21 5.64472e-20 -7.23971e-21 3.45786e-18 1.08519e-17
0 0 0 0 0 0 2.1469e-30 1.72156e-30 4.26102e-28 8.42538e-28 7.41167e-26 1.68365e-25 1.29085e-23 2.44561e-23 1.26518e-21 2.23494e-21 3.64435e-20 8.21464e-20 4.15518e-18 6.21907e-18 2.34158

Final Matrix:

0 0 0 0 3.40866e-35 5.28682e-32 2.2018e-30 7.70103e-31 4.94131e-28 8.00486e-28 9.92755e-26 1.66918e-25 1.18382e-23 1.51412e-23 9.22015e-22 1.54023e-21 5.07933e-20 8.48863e-20 1.36344e-18
0 0 0 0 0 1.288e-35 1.38224e-30 3.78893e-30 1.26882e-29 7.96475e-28 1.4269e-26 1.06375e-25 1.09801e-23 2.43484e-23 1.72679e-21 1.7549e-21 5.64472e-20 -7.23971e-21 3.45786e-18 1.08519e-17
0 0 0 0 0 0 2.1469e-30 1.72156e-30 4.26102e-28 8.42538e-28 7.41167e-26 1.68365e-25 1.29085e-23 2.44561e-23 1.26518e-21 2.23494e-21 3.64435e-20 8.21464e-20 4.15518e-18 6.21907e-18 2.34158
0 0 0 0 4.3168e-35 1.89544e-32 2.64824e-30 3.58752e-30 4.82479e-28 6.11989e-28 8.50004e-26 8.76542e-26 9.82243e-24 9.08086e-24 4.73169e-22 6.58221e-22 3.3888e-20 4.78707e-20 2.37878e-18
0 0 0 9.89297e-35 8.12366e-33 1.89642e-32 2.17527e-30 3.74428e-30 4.26581e-28 7.83098e-28 6.40743e-26 8.17248e-26 6.96213e-24 5.3478e-24 2.14248e-22 7.89867e-22 1.83429e-20 2.44315e-20 1.
0 0 0 0 3.40866e-35 5.28682e-32 2.2018e-30 7.70103e-31 4.94131e-28 8.00486e-28 9.92755e-26 1.66918e-25 1.18382e-23 1.51412e-23 9.22015e-22 1.54023e-21 5.07933e-20 8.48863e-20 1.36344e-18
0 0 0 0 4.3168e-35 1.89544e-32 2.64824e-30 3.58752e-30 4.82479e-28 6.11989e-28 8.50004e-26 8.76542e-26 9.82243e-24 9.08086e-24 4.73169e-22 6.58221e-22 3.3888e-20 4.78707e-20 2.37878e-18
0 0 0 1.1518e-35 7.26556e-33 1.60768e-32 1.77481e-30 3.03184e-30 3.05699e-28 7.90828e-28 4.69304e-26 7.66017e-26 2.7579e-24 4.23335e-24 3.27008e-22 3.57313e-22 9.3309e-21 2.32428e-20 3.9
0 0 1.16666e-35 7.58838e-35 5.38582e-33 7.05848e-33 1.41725e-30 2.20543e-30 2.48267e-28 4.68843e-28 3.09607e-26 2.88463e-26 1.92603e-24 4.24069e-24 1.5389e-22 1.71359e-22 5.69115e-21 1.6
0 0 0 9.89297e-35 8.12366e-33 1.89642e-32 2.17527e-30 3.74428e-30 4.26581e-28 7.83098e-28 6.40743e-26 8.17248e-26 6.96213e-24 5.3478e-24 2.14248e-22 7.89867e-22 1.83429e-20 2.44315e-20 1.
0 0 0 1.1518e-35 7.26556e-33 1.60768e-32 1.77481e-30 3.03184e-30 3.05699e-28 7.90828e-28 4.69304e-26 7.66017e-26 2.7579e-24 4.23335e-24 3.27008e-22 3.57313e-22 9.3309e-21 2.32428e-20 3.9
0 0 4.96304e-38 2.50218e-35 4.82446e-33 5.38557e-33 1.01407e-30 1.44997e-30 1.32685e-28 1.77123e-28 1.56003e-26 1.1999e-26 1.04342e-24 1.612e-24 8.31042e-23 5.96815e-23 5.62206e-21 8.728
0 1.03198e-37 1.00281e-35 1.4758e-35 3.53275e-33 4.72717e-33 6.84409e-31 8.82371e-31 9.28363e-29 6.53399e-29 9.83553e-27 6.1061e-27 7.00029e-25 5.37406e-25 4.38186e-23 5.04781e-23 1.9033
0 0 1.16666e-35 7.58838e-35 5.38582e-33 7.05848e-33 1.41725e-30 2.20543e-30 2.48267e-28 4.68843e-28 3.09607e-26 2.88463e-26 1.92603e-24 4.24069e-24 1.5389e-22 1.71359e-22 5.69115e-21 1.6
0 0 4.96304e-38 2.50218e-35 4.82446e-33 5.38557e-33 1.01407e-30 1.44997e-30 1.32685e-28 1.77123e-28 1.56003e-26 1.1999e-26 1.04342e-24 1.612e-24 8.31042e-23 5.96815e-23 5.62206e-21 8.728
0 1.84896e-38 7.83316e-36 1.56695e-35 2.44422e-33 3.70062e-33 4.45354e-31 5.42929e-31 6.00303e-29 4.19678e-29 3.99135e-27 2.97633e-27 2.3377e-25 1.9472e-25 1.80915e-23 2.6722e-23 1.26006
6.89858e-39 6.48598e-38 5.7975e-36 1.17894e-35 1.64881e-33 2.44001e-33 2.92204e-31 4.52952e-31 1.62033e-29 3.09702e-29 2.37388e-27 2.6256e-27 2.42147e-25 3.34073e-25 8.135e-24 2.84341e-2
0 1.03198e-37 1.00281e-35 1.4758e-35 3.53275e-33 4.72717e-33 6.84409e-31 8.82371e-31 9.28363e-29 6.53399e-29 9.83553e-27 6.1061e-27 7.00029e-25 5.37406e-25 4.38186e-23 5.04781e-23 1.9033
5.64992e-41 2.37231e-38 4.63292e-36 7.69779e-36 1.04774e-33 1.62961e-33 2.45918e-31 2.7333e-31 1.18784e-29 4.40712e-29 1.06428e-27 2.67038e-27 9.67548e-26 1.67896e-25 3.34337e-24 1.76213
0 1.84896e-38 7.83316e-36 1.56695e-35 2.44422e-33 3.70062e-33 4.45354e-31 5.42929e-31 6.00303e-29 4.19678e-29 3.99135e-27 2.97633e-27 2.3377e-25 1.9472e-25 1.80915e-23 2.6722e-23 1.26006
1.37942e-38 1.33862e-38 3.10021e-36 4.96415e-36 6.08672e-34 1.43022e-33 1.26476e-31 1.02448e-31 1.00144e-29 1.5715e-29 6.00395e-28 1.55426e-27 2.29742e-26 1.03638e-25 3.29961e-25 4.26512
5.80184e-39 3.09065e-39 2.00249e-36 3.94896e-36 3.40213e-34 1.09864e-33 4.41372e-32 9.05503e-32 2.94354e-30 1.11582e-29 1.12305e-28 7.57444e-28 3.64529e-27 2.88153e-26 1.62866e-25 9.3550
6.89858e-39 6.48598e-38 5.7975e-36 1.17894e-35 1.64881e-33 2.44001e-33 2.92204e-31 4.52952e-31 1.62033e-29 3.09702e-29 2.37388e-27 2.6256e-27 2.42147e-25 3.34073e-25 8.135e-24 2.84341e-2
5.64992e-41 2.37231e-38 4.63292e-36 7.69779e-36 1.04774e-33 1.62961e-33 2.45918e-31 2.7333e-31 1.18784e-29 4.40712e-29 1.06428e-27 2.67038e-27 9.67548e-26 1.67896e-25 3.34337e-24 1.76213
3.93696e-39 7.49404e-39 1.23749e-36 2.41768e-36 1.2474e-34 7.11588e-34 1.32026e-32 6.32976e-32 4.63826e-31 3.97103e-30 1.39185e-29 1.6211e-28 8.21117e-28 2.08127e-27 2.80041e-26 1.4339e-
2.37892e-39 6.92985e-39 8.44469e-37 1.02718e-36 4.68161e-35 3.06166e-34 3.08196e-33 1.89468e-32 1.62204e-31 9.3797e-31 6.09358e-30 2.25057e-29 1.51609e-28 1.31128e-27 5.15307e-27 3.822e-
1.6829e-39 5.72329e-39 2.76262e-37 6.75252e-37 1.07032e-35 1.24841e-34 5.39708e-34 3.00973e-33 3.30017e-32 9.92112e-32 6.03937e-31 5.62191e-30 5.4334e-30 1.69318e-28 5.7476e-28 1.21627e-
5.92392e-40 3.04838e-39 5.65325e-38 2.5682e-37 1.44272e-36 2.32137e-35 9.2895e-35 9.23443e-34 4.61466e-33 3.24123e-32 1.27804e-31 1.06644e-30 5.76998e-30 1.05983e-29 2.01228e-28 6.92809e-
1.37942e-38 1.33862e-38 3.10021e-36 4.96415e-36 6.08672e-34 1.43022e-33 1.26476e-31 1.02448e-31 1.00144e-29 1.5715e-29 6.00395e-28 1.55426e-27 2.29742e-26 1.03638e-25 3.29961e-25 4.26512
5.80184e-39 3.09065e-39 2.00249e-36 3.94896e-36 3.40213e-34 1.09864e-33 4.41372e-32 9.05503e-32 2.94354e-30 1.11582e-29 1.12305e-28 7.57444e-28 3.64529e-27 2.88153e-26 1.62866e-25 9.3550
1.0258e-40 9.1668e-40 7.93215e-39 5.31695e-38 3.50898e-37 3.22772e-36 9.60231e-37 1.13139e-34 3.73796e-34 3.09042e-33 2.21705e-32 2.66207e-32 7.91048e-31 4.15341e-30 9.46673e-30 1.12197e-
1.2219e-41 1.91812e-40 1.29707e-39 5.60834e-39 5.23106e-38 6.58739e-37 3.48468e-36 2.35028e-35 1.16399e-34 3.61958e-34 3.59256e-33 1.94716e-32 5.48569e-32 5.78991e-31 1.49287e-30 2.71835
```

Please see the outputs properly in the file I provided