Programming Assignment 2

Due on October 4th no later than midnight (11:59pm)

Instructions:

- Implement the Jacobi Rotations algorithm for computing the SVD using OpenMP.
- Compare your implementation with the provided SVD code which is sequential.
- Use the Validation.m (Octave/Matlab) file in order to verify that your output is correct.
- Use the file randomMatrix.py (Python) for generating a dense matrix. You can use the same matrix file through the whole assignment.
- The epsilon for convergence point must be no larger than 10^{-8}. You can go smaller if desired [how about 10^{-15}?]).
- Provide a report file presenting the behavior of your code with different sizes of matrix. How does this change impact the performance? You can use "binary" sizes (32, 64, 128, …) and go up to 1024. Did you accept the 4096 challenge?
- Mention in the report what happens if I leave the matrix size fixed and change the number of threads. For this exercise you can use the highest size only, just use different numbers of threads.
- Provide a README file.

How to use the files?

- Generate the matrix file: python randomMatrix.py N N (NxN is the matrix size)
- Once the "matrix" file is generated compile SVD.cpp: CC –O3 SVD.cpp –o SVD (remember to use CC as capital letter for C++, cc is for C and –xopenmp is the flag for OpenMP)
- Run the program: ./SVD N N (NxN is the matrix size)
    - Optional arguments:
        - -t : print the number of iterations and execution time
        - -p : print the U,V,S result
        - -d : generate the matrix files for Validation.m
- You can use the arguments you need at the same time
- To validate: octave Validation.m | matlab Validation.m
- This will present a result for U,V,S and the numbers presented should be close to epsilon.
- You can download Octave free here. Octave is a Matlab-like open source product.