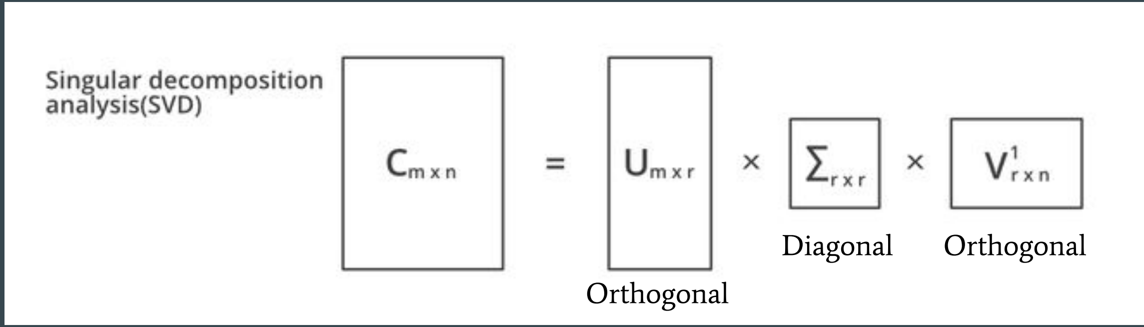
Team 19: Parallelization of SVD

Sibi Raja, Matthew Villaroman, Eliel Dushime, Bowen Zhu

Problem Introduction

- Singular Value Decomposition (SVD) is a matrix factorization method that breaks a matrix into three smaller matrices
- Motivation: SVD is at the forefront of modern data analysis and computation and has countless applications including image compression, machine learning, protein analysis, and Principal Component Analysis (PCA)
- To show our SVD algorithm in action, we will use it to compress images



GeeksforGeeks

Mathematical Model

- One of the standard methods for calculating the SVD of a matrix A_{MxN} is through the eigenvalue decomposition of the symmetric matrix A^TA (assuming that M > N)
- The eigenvalues of A^TA and the singular values in Σ are related by the following:
    - For eigenvalues λ_i of A^TA, the corresponding singular values σ_i of Σ is given by $\sigma_i = \sqrt{\lambda_i}$
- The column vectors of V are exactly the eigenvectors e_i corresponding to the eigenvalues of A^TA, arranged consistent with the singular values in Σ
- U is then calculated via the relation $U = AV\Sigma^{-1}$

$$\sigma_i = \sqrt{\lambda_i}$$

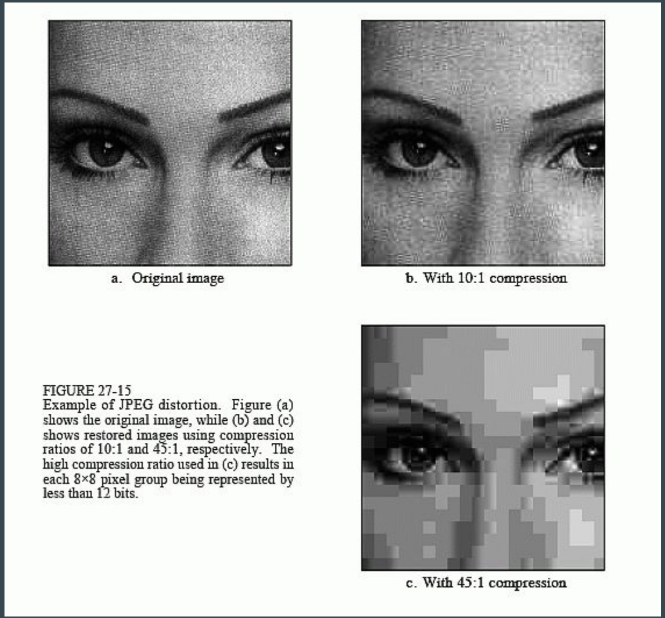
$$\Sigma = \begin{bmatrix} \sqrt{\lambda_1} & 0 & 0 & 0 & 0 \\ 0 & \sqrt{\lambda_2} & 0 & 0 & 0 \\ 0 & 0 & \sqrt{\lambda_3} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \sqrt{\lambda_r} \end{bmatrix}$$

$$V = [\vec{e_1} \quad \vec{e_2} \quad \vec{e_3} \quad \dots \quad \vec{e_r}]$$

$$U = AV\Sigma^{-1}$$

Data for SVD

- Limitation: dimensionality of the matrix
  - Most SVD algorithms assume that the matrix is $m \times n$, where $m > n$ (we will assume this as well)
  - Matrix size can be another limitation as there can be an upper bound on the computing power for calculating SVD
- Will apply SVD for image compression
- The data we will use will come in the form of an image file located on the system itself
  - This can be transformed into a matrix representation using Python (Pillow), which can then be passed into a .cpp file (as a text file)



The Scientist and Engineer's Guide to Digital Signal Processing

Why Parallelization?

- The SVD process is computationally expensive and can be a bottleneck for many applications
- Iterative SVD algorithms may not scale for matrices of larger dimensions → we will attempt to parallelize the SVD computation process
    - With large datasets, running SVD can be time-consuming. Speeding this up can have improvements in the various applications of SVD such as image compression or PCA
- *J. SairaBanu, Rajasekhara Babu and Reeta Pandey:* parallelizing SVD is faster than a sequential process

Plan to Parallelize SVD

- Parallelizing the independent matrix operations → later used in eigenvalue/eigenvectors calculations (OpenMP)
    - SIMD
        - Break up matrix into submatrices (multiple data) with the same operations on each (single instruction)
        - Finding an eigenvector (single instruction) for various eigenvalues (multiple data)
- Shared memory: matrix placed in shared memory while different threads will each read and update certain columns of the matrix in parallel
- Work package
    - Analyze which areas of SVD calculation will contribute to the greatest speedup when parallelized, pinpointing possible data dependencies in calculations and creating solutions, and begin creating a parallel algorithm for SVD

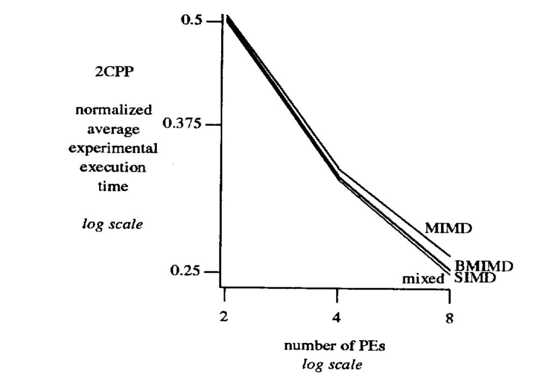


Fig. 6: 2CPP algorithm execution time comparison for different modes of parallelism; 4×4 matrix.

SIMD performs best (Colorado State University)