Reporte del paper SVD on GPU using CUDA

La descomposición en valores singulares o SVD se utiliza para expresar una matriz real o compleja como el producto de otras 3 matrices. Es útil y usada en el calculo de la pseudoinversa, para la solucion de ecuaciones lineales homogéneas, resolución de mínimos cuadrados y para aproximaciones de una matriz.

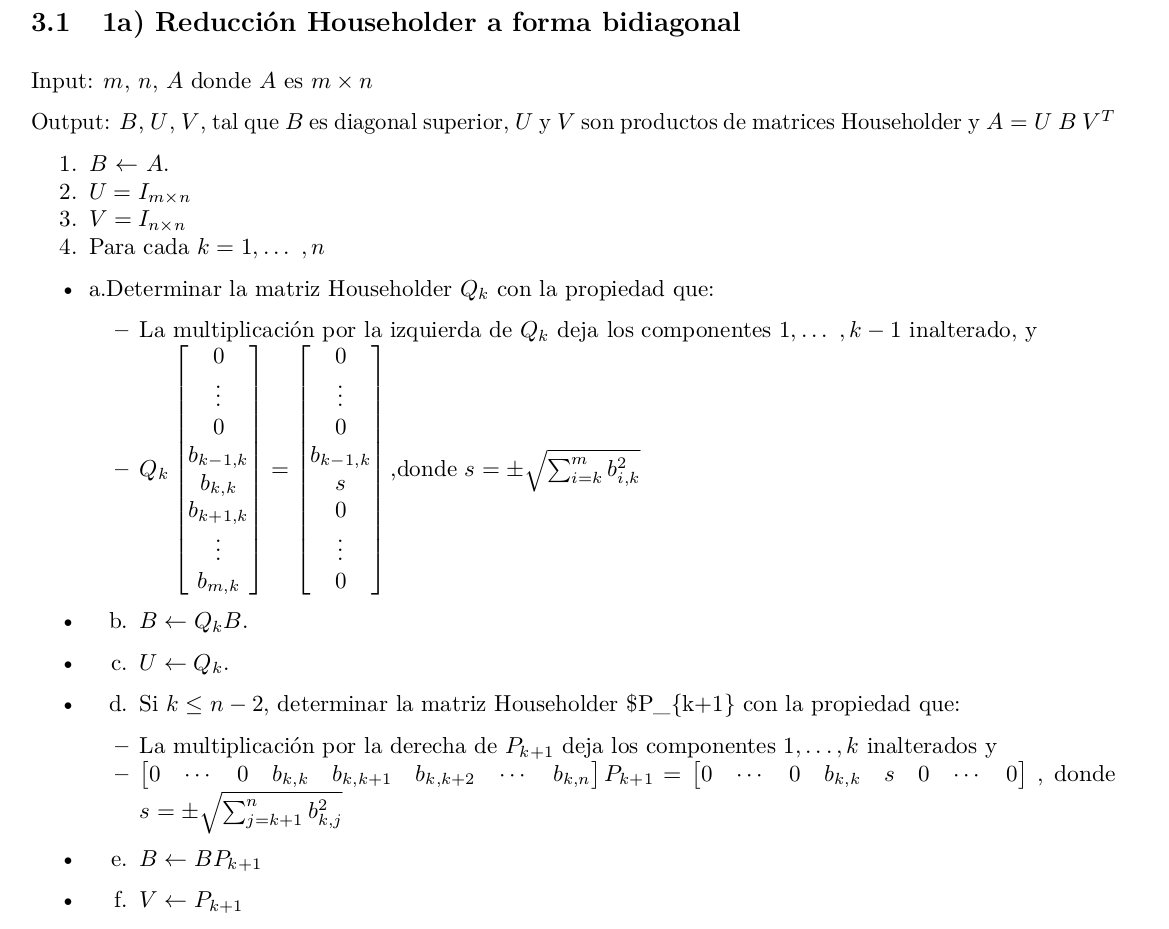
La deficinicion formal de la SVD de una matriz A m x n es de la forma:

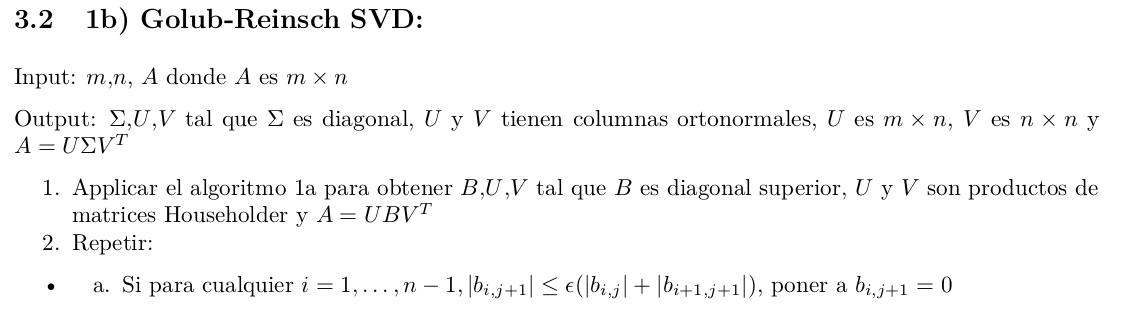
A = UΣVT

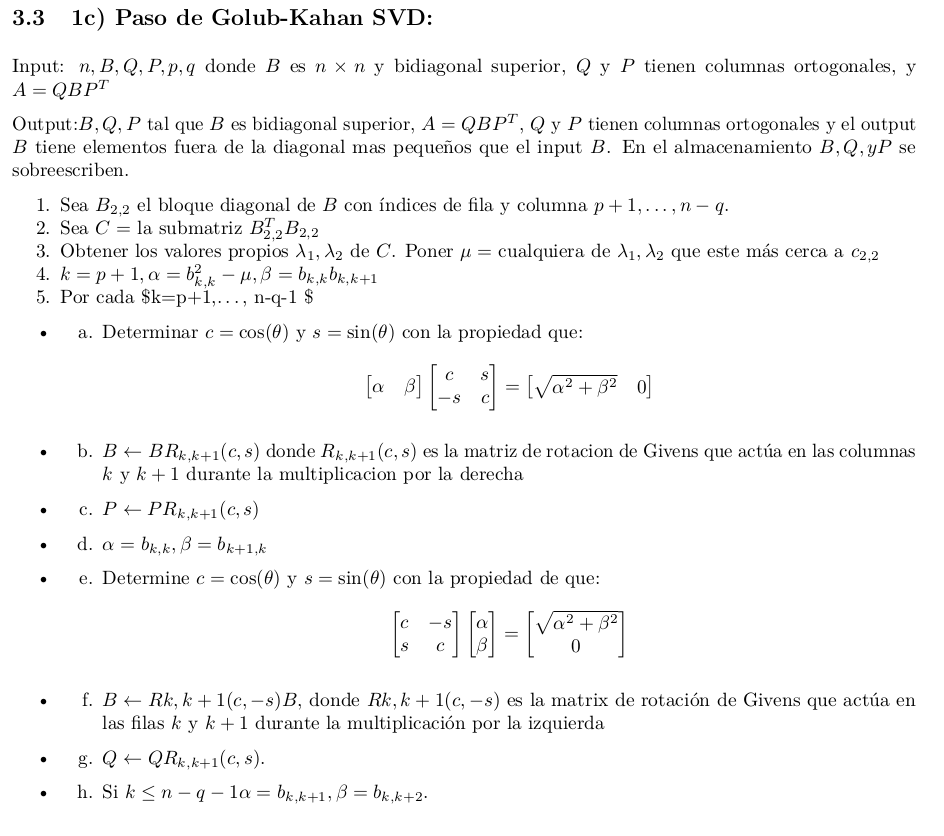
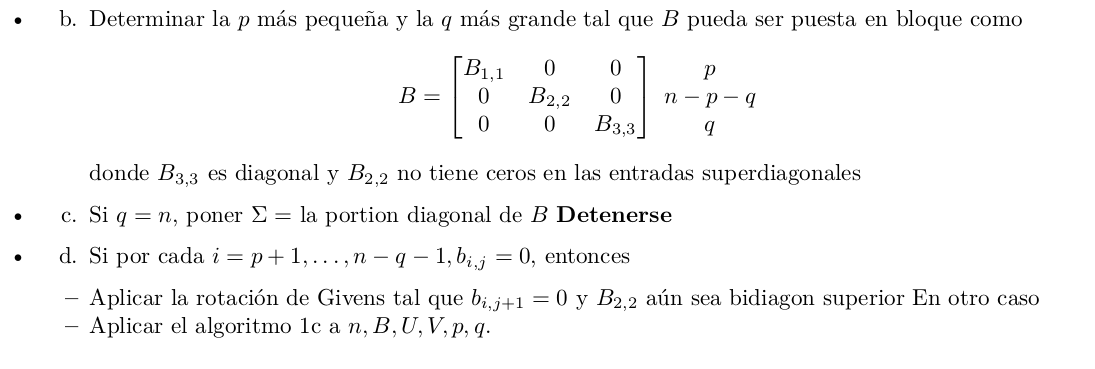
* U matriz ortogonal mxm
* V matriz ortogonal nxn
* Σ es una matriz diagonal mxn con elementos sij = 0 si I es distinto de j y sii ≥ 0 en orden decreciente a lo largo de la diagonal.

Dicho paper, expone la posibilidad de computar la SVD a través de una GPU de NVDIA, pues en esta plataforma se han desarrollado diversos algoritmos que explotan la arquitectura en paralelo de una GPU, además de tener la posibilidad de integrar diferentes librerías numéricas como ATLAS, LAPACK MKL LAPAK,etc.

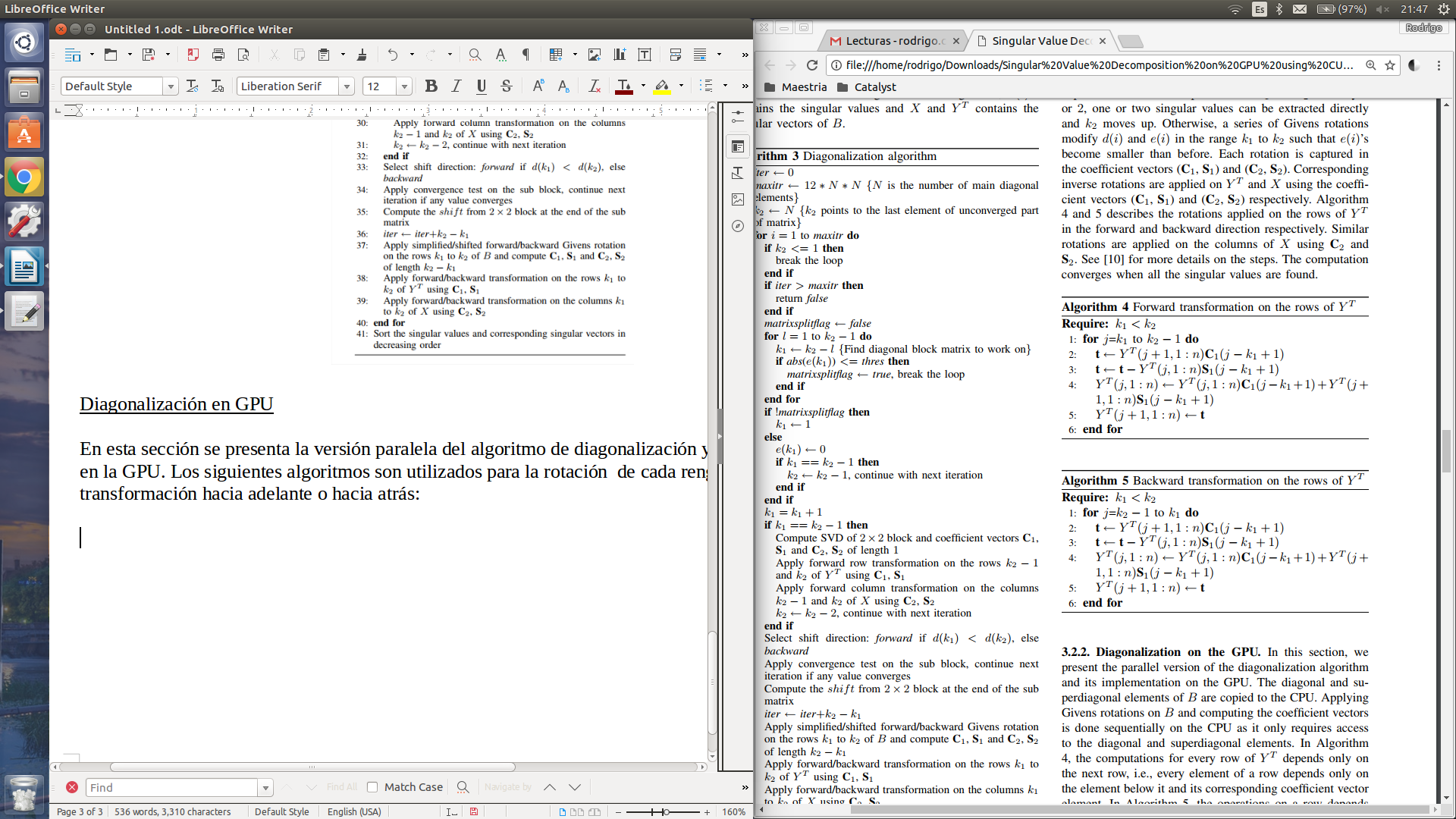
Para calcular la SVD, el autor del paper, usó el algoritmo GOLUB- REINSCH, el cuál es como sigue:







Posteriormente, el paper menciona la manera de llevar a cabo la diagonalización en la GPU.

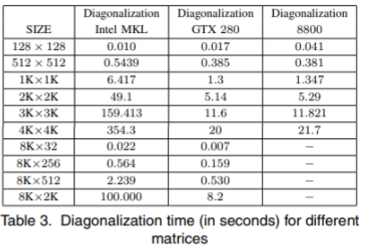


El algoritmo divide cada línea de la matriz en bloques. Cada thread opera en un elemento de la fila. Esta división de líneas en bloques y formar bucles puede ser hecho eficientemente con una arquitectura CUDA debido a que cada thread realiza cómputo independiente. Los datos requeridos de cada bloque, son guardados en una memoria compartida y las operaciones pueden ser realizadas eficientemente en un multiprocesador.

Debido a que las transformaciones de columna son similares a las transformaciones de renglón, utilizamos transformación de renglón del kernel en las filas de XT en lugar de las columnas de X. Todos los threads de un bloque son usados para copiar los sectores de la memoria global a la memoria compartida que requiere que los vectores estén acomodados al bloque múltiple más cercano.

Después de realizar la descripción del algortimo, se reportan los resultados de la implementacion en INTEL MKL 10.0.4 LAPACK y MATLAB

Se generaron 10 matrices densas aleatorias de números de precisión singular para cada tamaño. El algoritmo de SVD fue ejecutado 10 veces. Para evitar una muestra mala o buena se realizó un promedio de matrices aleatorias para cada tamaño. A continuación, se pueden observar algunas gráficas y tablas para visualizar los resultados.



Finalmente me parece que es acertado el uso de este algoritmo que explota el paralelismo en la arquitectura GPU y alcanza un desempeño computacional alto. Y me pareció interesante que la bidagonalización de la matriz se realizara completamente en GPU usando la librería de CUBLAS.

Así mismo reflexiono en que este paper ya tiene tiempo por lo que de su publición a la fecha, es probable que la potencia de las GPU permitan un desempeño sin precendentes, tanto para esta tarea, como para otras en el futuro.