1. General Remarks

In this assignment you are asked to use CUDA libraries `cuBLAS` and `cuSOLVER` to solve a least square problem via Truncated SVD.

For most recent documentation of `cuBLAS` visit

`https://docs.nvidia.com/cuda/cublas/index.html`

For most recent documentation of `cuSOLVER` visit

`https://docs.nvidia.com/cuda/cusolver/index.html`

2. Backgound

You are given a real $n \times n$ matrix A. You want to find an approximate solution x to the following least squares problem

$$argmin_{x \in R^n} ||Ax - b||_2 \tag{1}$$

Your rhs vector b is obtained by multiplying A by a vector e with all components being ones.

A admits the following SVD

$$A = U\Sigma V^T$$

where U and V are $n \times n$ orthogonal matrices, and Σ is a real $n \times n$ diagonal matrix with diagonal elements

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$$

It turns out that, as long as $\sigma_n > 0$, the solution to (1) is

$$x = V\Sigma^{-1}U^T b$$

Let

$$U = [u_1, u_2, \ldots, u_n],\ V = [v_1, v_2, \ldots, v_n]$$

Then x can also be written as

$$x = \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i} v_i \tag{2}$$

You find out that for a certain $1 < k < n$

$$\sigma_k >> \sigma_{k+1}$$

It turns out that by colecting the first k columns of U and V, and taking the leading k singular values from Σ, one can obtain a good approximation of A.

Let, in Matlab notation,

$$U_k = U(:, 1:k),\ V_k = V(:, 1:k),\ \Sigma_k = \Sigma(1:k, 1:k)$$

Then for

$$A_k = U_k \Sigma_k V_k^T$$

we have that

$$||A - A_k||_2 \approx \sigma_{k+1}$$

$A_k = U_k \Sigma_k V_k^T$ is known as a Truncated SVD (TSVD).

Truncated SVD is often used to get an approximate solution to (1). That approximation is derived from (2) and is given as

$$x_k = \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} v_i \tag{3}$$

or in matrix form

$$x_k = V_k \Sigma_k^{-1} U_k^T b \tag{4}$$

Now, because we truncated the SVD decomposition we will find that often $e = ||x - x_k||_2$ is large however the residual error $r = ||A_k x_k - b||_2$ will be of the order of σ_{k+1}.

Your problem will be to find x_k, e, r.

3. Data

You will be given a matrix A for which there is substential gap between singular values σ_k and σ_{k+1}. You will need to find k.

The data is stored in a text file `MyMatrix.txt` and should be loaded to the host using the same code `c_read_mat.c` that you used in Assignmment 3.

4. Steps to be implemented

1. As a solution x to (1) create an n-length vector of all ones.
2. Load data to the host.
3. Move A and x to the device.
4. Using the library matrix-vector multiplication routine `cublasSgemv` set $b = Ax$.
5. Using `cuSolver` library SVD routines

   - `cusolverStatus_t cusolverDnSgesvd_bufferSize`
   - `cusolverStatus_t cusolverDnSgesvde`

   find the SVD of A, $A = U\Sigma V^T$ (note that Σ is returned as an ordered vector of singular values, $\Sigma = [\sigma_1, \sigma_2, ..., \sigma_n]$).

6. Find k. This part is more an art than science if not much is known where A came from. For the purpose of this assignment let us take the first k for which

$$\frac{\sigma_{k+1}}{\sigma_k} \leq 10^{-3}$$

   Care must be taken so one avoids division by zero. For that `cublasIsamin` might prove useful.

7. Form U_k, V_k, Σ_k (Σ_k is a vector).

8. Find x_k from (3) or (4),

   - using the matrix-vector multiplication routine `cublasSgemv` form $b_k = U_k^T b$.
   - scale elements of b_k by the corresponding singular values, for that write CUDA kernel(s) that accomplish such scaling,
   - once you have the modified b_k, that is $d_k = \Sigma_k^{-1} U_k^T b_k$, use `cublasSgemv` to form $x_k = V_k d_k$.

9. Compute the error $e = ||x_k - x||_2$ using `cublasSnrm2`.

10. Compute Ax_k using `cublasSgemv` and $Ax_k - b$ using `cublasSaxpy`.

11. Compute the residual error $r_k = ||Ax_k - b||_2$ using `cublasSnrm2`.

12. Move x_k, e, r to the host.

13. Print e, r and the first 8 entries of x_k.

5. Requirements

A template for the code will be placed in `/calsses/assignments/hw4/`

    /classes/assignments/hw4

Your code must be well commented. There should not be there any unessential lines. In particular, **remove all unessential lines that were present in the template**.

Measure the execution time from Step (3) to Step (12) (inclusive).

Your findings, a discussion of results, graphs and tables, if any, should be saved in a file

    your_net_id_hw4_writeup.pdf

In the write-up, explain how your kernel function(s) work, and how and where you call them.

If you include tables or graphs, please present them in a way that can be easily understandable.

Your code should be saved in a file

    your_net_id_hw4_code.c.

(Print instructions how to compile and execute the code in the first line the file `your_net_id_hw4_code.c`.)