F.M. Faulstich **Math 6590: Homework assignment 1** **Due:** Friday Jan. 19, 2024.

Exercise 1 *Let $\mathbf{A} \in \mathbb{H}_n$ be PSD. Prove the inequalities*

$$1 \le \operatorname{intdim}(\mathbf{A}) \le \operatorname{rank}(\mathbf{A}). \tag{1}$$

Show that the upper bound is saturated if $\mathbf{A}$ is an orthogonal projector.

Exercise 2 *Let M be a square matrix partitioned as*

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}. \tag{2}$$

Let $\mathbf{A}$ be nonsingular, prove that

$$\det(\mathbf{M}/\mathbf{A}) = \det(\mathbf{M})/\det(\mathbf{A}). \tag{3}$$

Exercise 3 *Prove that the spectral norm is unitarily invariant, i.e., let $\mathbf{A} \in \mathbb{C}^{m\times n}$, then*

$$\|\mathbf{WAS}\| = \|\mathbf{A}\|$$

for all unitary matrices $\mathbf{W} \in \mathbb{C}^{m\times m}$ and $\mathbf{S} \in \mathbb{C}^{n\times n}$.

Exercise 4 *In this programming assignment, you will implement and test your own randomized SVD:*

1. *Write a MATLAB program that performs the randomized range sketch. Your routine should take a matrix $\mathbf{A}$ and a parameter ℓ as input and return the orthonormal matrix $\mathbf{Q}_\ell$ s.t.*
$$A \approx \mathbf{Q}_\ell \mathbf{Q}_\ell^* \mathbf{A}$$
2. *Write a MATLAB program that performs the randomized range finder discussed in class.*
3. *The randomized range sketch as well as the randomized range finder can be used as a sketch. Wrap both routines as "Stage A" (you may use an input flag to steer if the range finder or sketch is used), and write an extension using "Stage B" discussed in class. This extension should take $\mathbf{A}$ and ε as input and return an approximate SVD based on the two-stage low-rank approximation procedure outlined in class. You may use the MATLAB internal routine `svd` for this stage.*
4. *It is now time to test your routines and convince me that this algorithm is worth using (at least for some matrices). As a benchmark use the MATLAB internal routine `svd`*
   (a) *Investigate matrices with exponentially decaying singular values. To that end, create a matrix $\mathbf{\Sigma}$ with decreasing values along its diagonal and reverse the SVD to obtain $\mathbf{A}$. Then, run your implementation of the randomized SVD over a range of sampling parameter values $\ell \in \{1, 2, \dots 150\}$. For each ℓ computed two values:*
   - $\sigma_{\ell+1}$ *(This is the minimum approximation error by EYM)*
   - $e_\ell = \|(\mathbf{1} - \mathbf{Q}_\ell \mathbf{Q}_\ell^*)\mathbf{A}\|$ *(This is the actual error)*

   To know what we are aiming at, compare with Fig. 7.2 in [?]. Then **also** *plot the singular values you picked for Σ and compare them with the computed singular values with your code and MATLAB's SVD.*

(b) *Change* $\mathbf{\Sigma}$ *above to have rapidly decaying singular values as well as slowly decaying singular values. What do you observe? Again, try to scientifically convince me that your code is useful for some systems, but be also honest in presenting the code's weaknesses.*

(c) *Perform timings of your codes. Remember that you should make* **at least** *five (rather more!) runs per input and take the average time to obtain a smooth and representable timing plot. Perform timing experiments and scale the considered matrices up. Try to push your code to outperform the MATLAB built-in SVD in terms of speed.*

(d) *Experiment with MATLAB profiler to find computational bottlenecks in your code.*