Numerical Linear Algebra

Kevin Corcoran

March 17, 2022

Contents

5: Final Project

Wed 09 Mar 2022 09:35

1 Part 1

1.1 SVD for image compression

The first 10 largest singular values

$$\begin{gathered} 663180.2023 \\ 85706.5957 \\ 62129.0257 \\ 34664.6330 \\ 31861.7923 \\ 21872.7216 \\ 19628.4428 \\ 18434.9377 \\ 13693.8154 \\ 12815.2083 \end{gathered}$$

The singular value for $k = 20$

$$\sigma_k \approx 7528.0246523376873.$$

The singular value for $k = 40$

$$\sigma_k \approx 5489.1246638996563.$$

The singular value for $k = 80$

$$\sigma_k \approx 3948.7799793164731.$$

The singular value for $k = 160$

$$\sigma_k \approx 2668.2235780120509.$$

The singular value for $k = 320$

$$\sigma_k \approx 1515.8659320175318.$$

The singular value for $k = 640$

$$\sigma_k \approx 821.89312579199247.$$

The singular value for $k = 1280$

$$\sigma_k \approx 513.56803215302216.$$

The singular value for $k = 2560$

$$\sigma_k \approx 179.11503509371889.$$

The original image

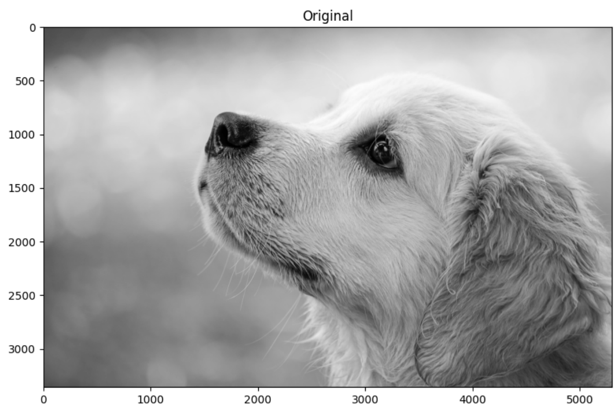

Figure 1: Original

compressed images

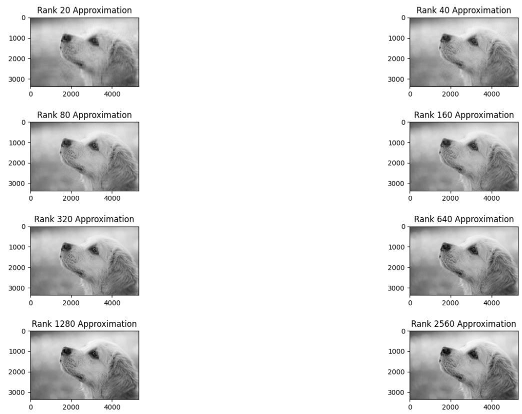

Figure 2: Sigma k

As we increase the number of singular values the image becomes clearer on reconstruction. Plotting the averaged Frobenius norm

$$E_k = \frac{\|A - A_{\sigma_k}\|_F}{mn}.$$

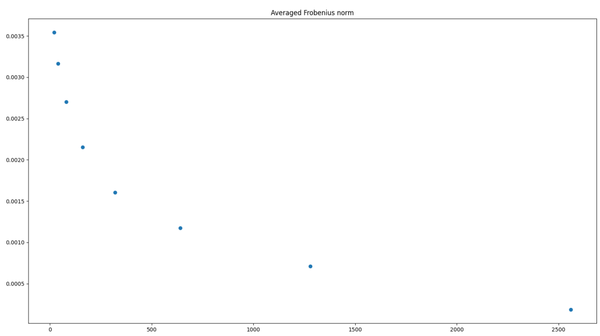

Figure 3: Error

and the value of k, where the error $E_k < 10^{-3}$, is

$$\boxed{k = 1280}.$$

1.2 Iterative Methods

1.2.1 Gauss-Jacobi and Gauss-Seidel

Both algorithms "split" the matrix into a sum of parts. This split,

$$A = M - N.$$

assuming M is invertible, induces an iterative method

$$\begin{aligned} &Mx_{k+1} = Nx_k + b \\ \Leftrightarrow\ &x_{k+1} = M^{-1}Nx_k + M^{-1}b \end{aligned}$$

So we'd like M to be a good approximation for A where $Mx = y$ is cheap and easy to solve. The splitting $A = M - N$ converges to

$$Ax = b.$$

for A nonsingular and iff the spectral radius $\rho(M^{-1}N) < 1$

The Jacobi method corresponds to the splitting $M = D$ and $N = -L - U$ where D is the diagonal part of A, and L and U are the lower and upper triangular part respectively. For a strictly diagonally dominant matrix A

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \qquad \text{for} \quad i = 1, \ldots, n.$$

Then the iterative scheme,

$$\begin{aligned} &T = M^{-1}N = -D^{-1}(L + U) \\ &\Rightarrow ||T||_\infty < 1 \\ &\Rightarrow \rho(T) \leq ||T||_\infty < 1 \end{aligned}$$

converges.

The Gauss-Seidel scheme, on the otherhand, corresponds to the splitting $M = D + L$ and $N = -U$. This algorithm uses updated values as soon as they become available, but has less clear convergence criteria. Though it manages to converge in the cases when Gauss-Jacobi fails. In both I set a maximum iteration of 1000 and exit when the error grows too large. I use the two norm and fortrans huge function which returns the largest value for the data inputted data type

$$\|Ax - b\|_2 < \text{huge}(\|Ax - b\|_2).$$

Run the code for a 10 x 10 matrix A with $D = 2, 5, 10, 100, 1000$ and plot the error $||b - Ax||_2$ for each value of D and for both Jacobi and Gauss-Seidel

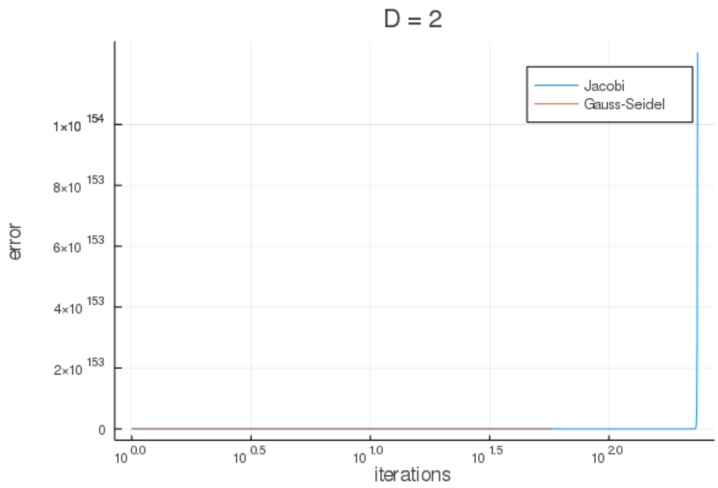

Figure 4: Error D = 2

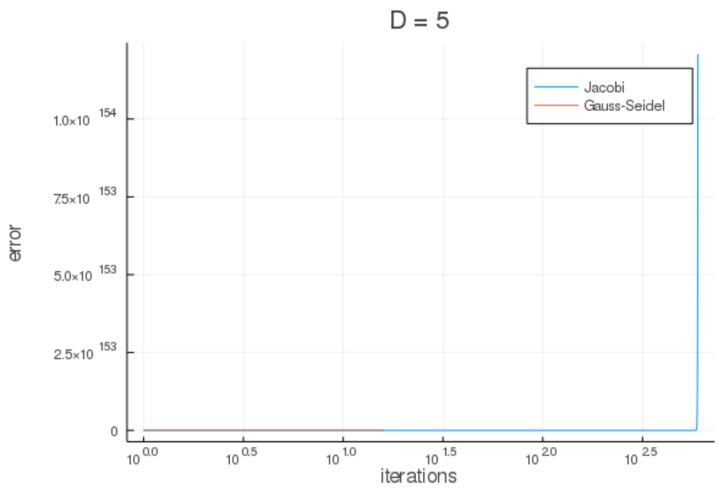

Figure 5: Error D = 5

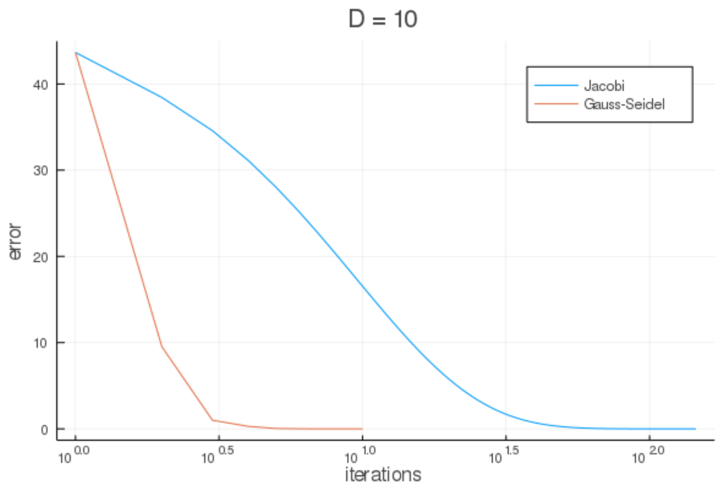

Figure 6: Error D = 10

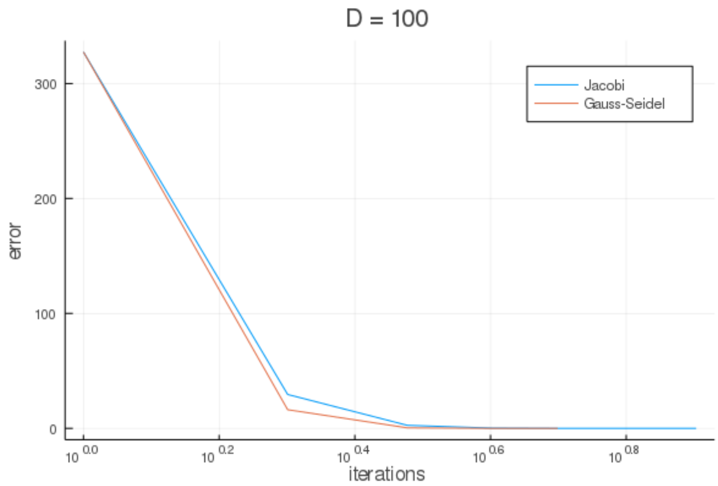

Figure 7: Error D = 100

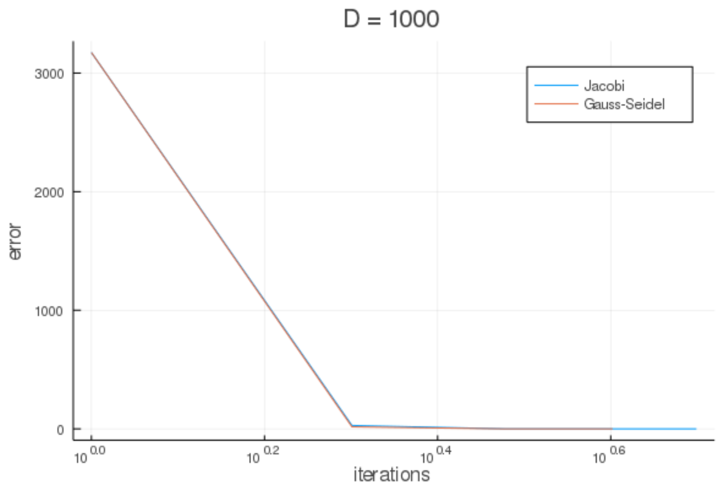

Figure 8: error D = 1000

Jacobi didn't converge for $D = 2, 5$ since, in those cases, the matrix A was not diagonally dominant.

Running each algorithm with a matrix A full of ones, except on the diagonal where $a_{ii} = i$, Jacobi does not converge, but Gauss-Seidel converges in 1 step to

$$\begin{pmatrix} -8 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

The error on a log-linear plot

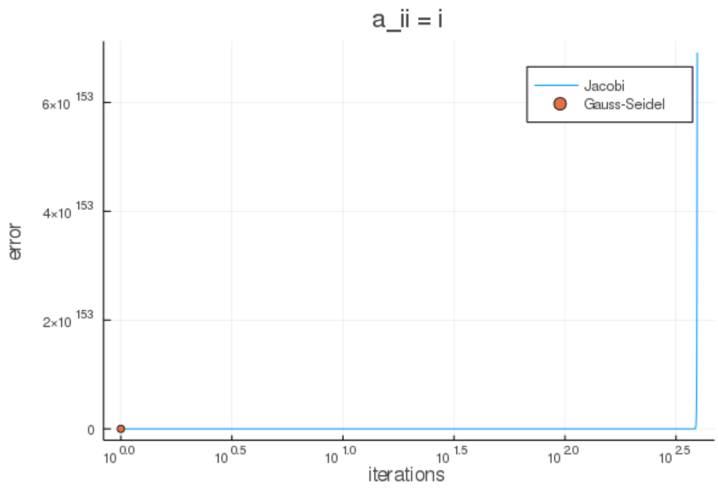

Figure 9: Aii

- Conjugate Gradient, and other similar algorithms, can be thought of as minimizing the objective function

$$\phi(x) = \frac{1}{2}x^T Ax - x^T b.$$

The solution to which converges to $Ax = b$, and corresponds to the scheme

$$x_{k+1} = x_k + \alpha_k p_k.$$

for some direction p_k. For Steepest Descent (or Gradient Descent) this direction is the residual or the steepest slope. However, with Conjugate Gradient, the direction p_{k+1} is chosen to be A-conjugate to p_k.

$$p_{k+1} A p_k = 0.$$

Conjugate Gradient converges in at most m steps since the vectors p_k are guaranteed to be linearly independent. Which means our solution x^* can be written as a linear combination of these vectors, and addition directions offer no more information. For well conditioned matrices, Conjugate Gradient can converge in as few as 2 steps

proof:

- Prove the smart conjugate gradient is equivalent to the basic conjugate gradient.

- Table comparing number of iterations until convergence between 3 algorithms.

Table 1: Number of Iterations

D	2	5	10	100	1000
Jacobi	DNC	DNC	145	8	5
Gauss-Seidel	58	16	10	5	4
Conjugate Gradient	2	2	2	2	2

Jacobi doesn't converge until it becomes diagonally dominant at $D = 10$. Both Jacobi and Gauss-Seidel converge faster for larger diagonal elements. This is because the matrix becomes better conditioned.

- A diagonal pre-conditioner is not very useful for these matrices since they are already well conditioned (ie the maximum and minimum eigenvalues only differ by a small amount). Since the matrix is symmetric positive definite.
$$\kappa(A) = \frac{\lambda_1}{\lambda_m}.$$

- Running the algorithm again with a matrix A full of ones, except on the diagonal where $a_{ii} = i$, Conjugate Gradient takes more than 2 steps to converge.

  For a 10 x 10 matrix it takes 10 iterations to converge to the solution
$$\begin{pmatrix} -8 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

  For a 100×100 matrix it takes 62 iterations to converge to the solution
$$\begin{pmatrix} -98 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

  These matrices take longer to converge since they are ill-conditioned. For the 10×10 case, the largest eigenvalue $\lambda_{10} \approx 15.3$ and the smallest is $\lambda_1 \approx 0.23$. For the 100×100 case, the largest $\lambda_{100} \approx 157.7$ and the smallest $\lambda_1 \approx 0.15$