Mathematical Data Science

Numerical Linear Algebra for Big Data

Martin van Gijzen

Delft University of Technology

March 22, 2018

Outline

Motivation

The Singular Value Decomposition

Image compression

Algorithms for computing eigenvalues and singular values

- The QR algorithm
- Arnoldi's method, Lanczos method and Lanczos bi-diagonalisation

Motivation

- Data has structure

Motivation

- Data has structure
- The goal of data science of to find this structure

Motivation

- Data has structure
- The goal of data science of to find this structure
- By exposing this structure we can for example

Motivation

- Data has structure
- The goal of data science of to find this structure
- By exposing this structure we can for example
  - Extract meaning, information (as in data mining)

Motivation

- Data has structure
- The goal of data science of to find this structure
- By exposing this structure we can for example
  - Extract meaning, information (as in data mining)
  - Reduce the data by representing correlated data as one variable

Motivation

- Data has structure
- The goal of data science of to find this structure
- By exposing this structure we can for example
  - Extract meaning, information (as in data mining)
  - Reduce the data by representing correlated data as one variable
  - Separate signal from noise (meaningless data)

Motivation

- One of the most important methods to detect structure in data is the Singular Value Decomposition (SVD)

Motivation

- One of the most important methods to detect structure in data is the Singular Value Decomposition (SVD)
- The same method is known under different names in different fields: Principal Component Analysis (PCA) in statistics, the discrete Karhunen Loeve Transform (KLT) in signal processing and Proper Orthogonal Decomposition (POD) in mechanical engineering

Motivation

- One of the most important methods to detect structure in data is the Singular Value Decomposition (SVD)
- The same method is known under different names in different fields: Principal Component Analysis (PCA) in statistics, the discrete Karhunen Loeve Transform (KLT) in signal processing and Proper Orthogonal Decomposition (POD) in mechanical engineering
- Applications: Model Order Reduction, Cluster Analysis, Face Recognition, Text Mining, Data Compression, Inverse Problems, ...

Motivation

- One of the most important methods to detect structure in data is the Singular Value Decomposition (SVD)
- The same method is known under different names in different fields: Principal Component Analysis (PCA) in statistics, the discrete Karhunen Loeve Transform (KLT) in signal processing and Proper Orthogonal Decomposition (POD) in mechanical engineering
- Applications: Model Order Reduction, Cluster Analysis, Face Recognition, Text Mining, Data Compression, Inverse Problems, ...
- Today we will discuss the definition and properties of the SVD and illustrate it on image compression

Motivation

- One of the most important methods to detect structure in data is the Singular Value Decomposition (SVD)
- The same method is known under different names in different fields: Principal Component Analysis (PCA) in statistics, the discrete Karhunen Loeve Transform (KLT) in signal processing and Proper Orthogonal Decomposition (POD) in mechanical engineering
- Applications: Model Order Reduction, Cluster Analysis, Face Recognition, Text Mining, Data Compression, Inverse Problems, ...
- Today we will discuss the definition and properties of the SVD and illustrate it on image compression
- Standard algorithms for computing the SVD are not suited for big data sets. We will therefore discuss ideas from Numerical Linear Algebra that can be used for big data.

The Singular Value Decomposition

Let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank r. Then there exist orthogonal matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ such that

$$A = U \Sigma V^T, \quad \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix}$$

where $\Sigma \in \mathbb{R}^{m \times n}$ and $\Sigma_r = diag(\sigma_1, \sigma_2, \cdots, \sigma_r)$, and

$$\sigma_1 \geq \sigma_2 \geq \cdots > 0.$$

The σ_i are called the singular values of A. The columns of U and V are the left and right singular vectors, respectively. Note that

$$Av_i = \sigma_i u_i \quad , \quad A^T u_i = \sigma_i v_i$$

Relation with eigenpairs

Note that

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

So the eigenvalues of $A^T A$ are the squared singular values of A. Moreover, the eigenvectors of $A^T A$ are the right singular vectors of A.

Relation with eigenpairs

Note that

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

So the eigenvalues of $A^T A$ are the squared singular values of A. Moreover, the eigenvectors of $A^T A$ are the right singular vectors of A.

Similarly

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

So the eigenvalues of $A A^T$ are the squared singular values of A. Moreover the eigenvectors of $A A^T$ are the left singular vectors of A.

Singular Values and the Frobenius norm

The Frobenius norm of the matrix A is defined by

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

It is easy to see that this is equal to

$$\|A\|_F = \sqrt{trace(A^T A)}$$

in which $trace(M) = \sum_{i=1}^{n} m_{ii}$. Since the trace of a matrix is equal to the sum of the eigenvalues we have

$$\|A\|_F = \sqrt{\sum_{i=1}^{r}\sigma_i^2}$$

Eckart-Young theorem

Let $A \in \mathbb{R}^{m \times n}$ be of rank r have a singular value decomposition $A = U\Sigma V^T$.

$$\Sigma = \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \qquad \Sigma_r = diag(\sigma_1, \sigma_2, \cdots, \sigma_r)$$

Let $B \in \mathbb{R}^{m \times n}$ be of rank $k < r$. The matrix B that minimizes

$$\|A - B\|_F$$

is given by $B = U\hat{\Sigma}V^T$.

$$\hat{\Sigma} = \begin{pmatrix} \Sigma_k & 0 \\ 0 & 0 \end{pmatrix} \qquad \Sigma_k = diag(\sigma_1, \sigma_2, \cdots, \sigma_k)$$

An application: image compression

- An image can be represented as a matrix, with grey values as entries

An application: image compression

- An image can be represented as a matrix, with grey values as entries
- Of this matrix we can make an SVD

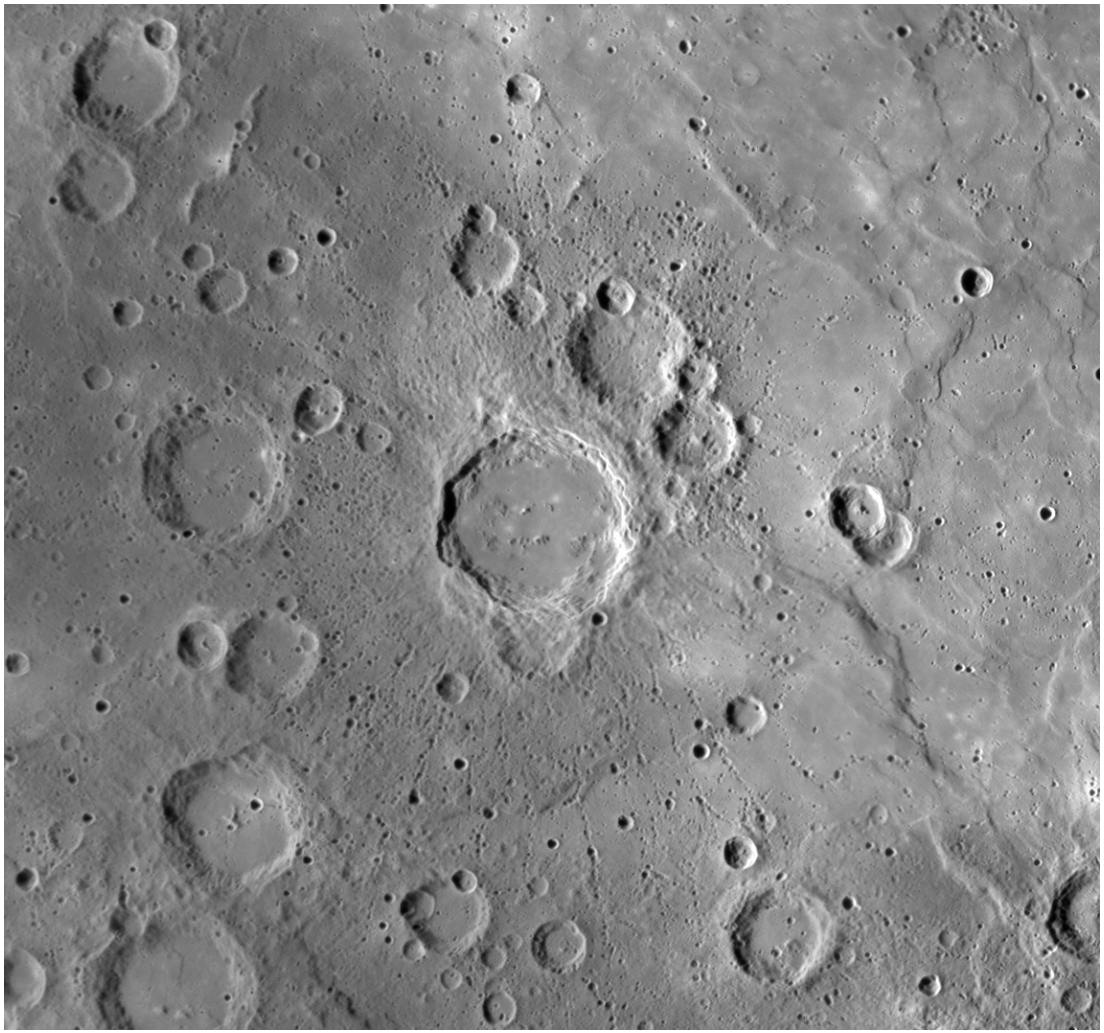
An application: image compression

- An image can be represented as a matrix, with grey values as entries
- Of this matrix we can make an SVD
- Compression can be achieved by making a low-rank approximation, i.e., by using only a limited number of singular values/vectors

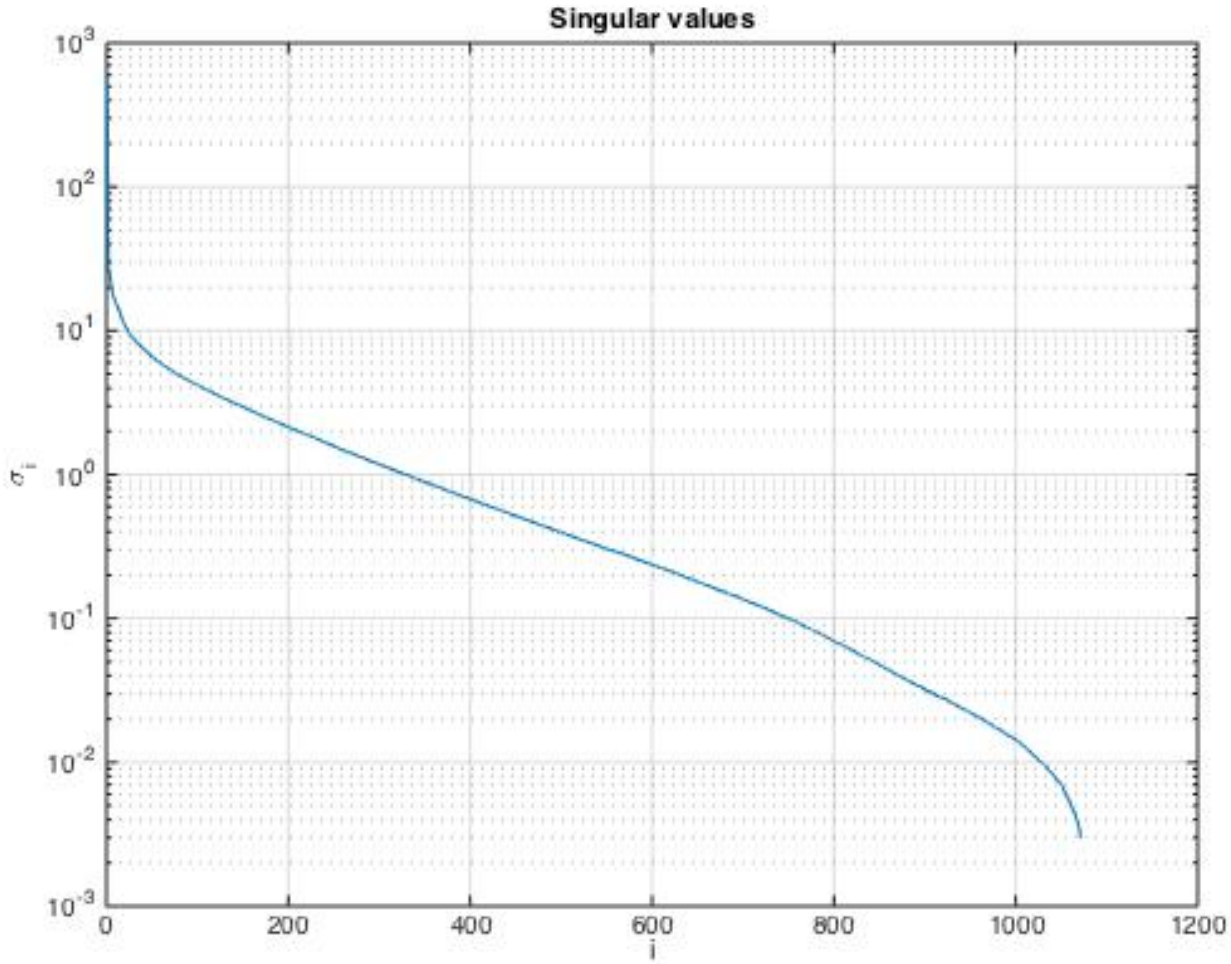
An application: image compression

- An image can be represented as a matrix, with grey values as entries
- Of this matrix we can make an SVD
- Compression can be achieved by making a low-rank approximation, i.e., by using only a limited number of singular values/vectors
- We can take $\|A - \hat{A}\|_F / \|A\|_F$ as measure for the error in the compression. Here $\hat{A}$ is the optimal low rank approximation to A.

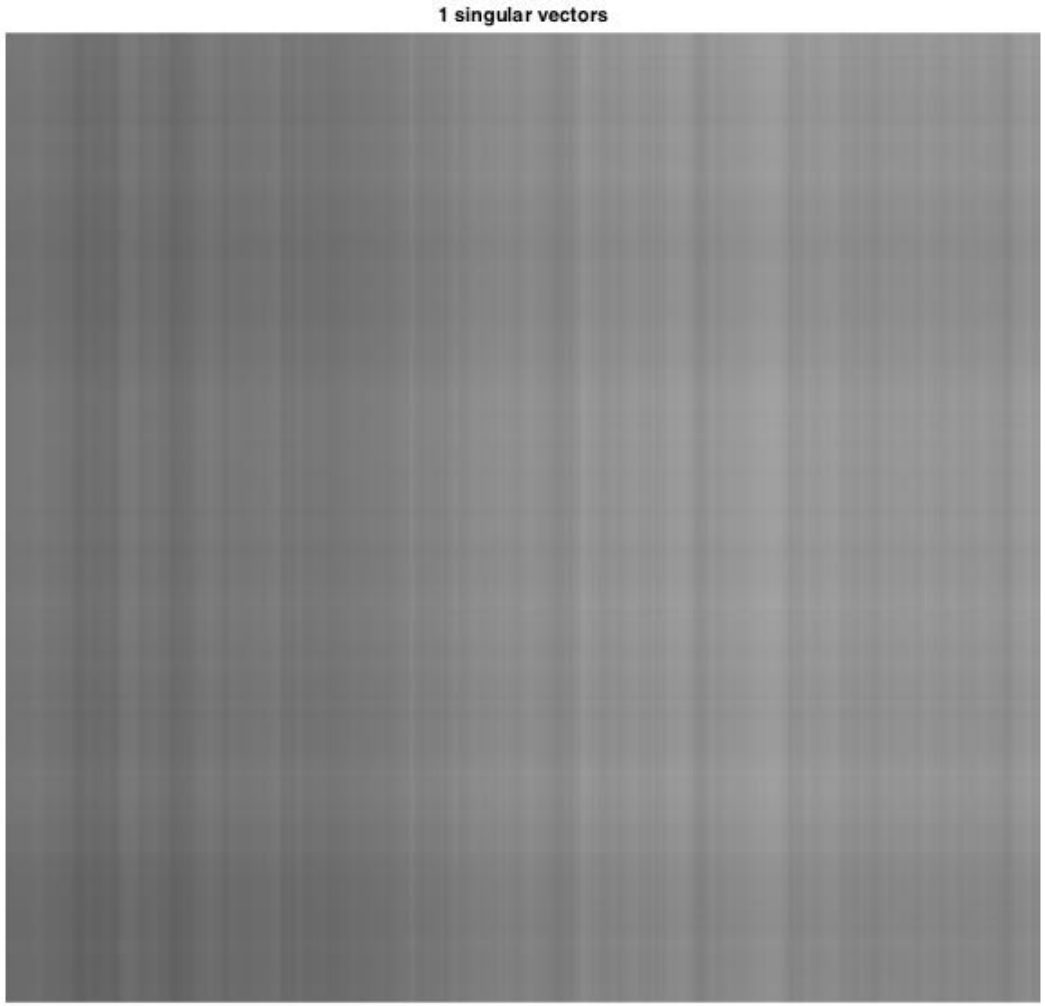
Example: image of Mercury



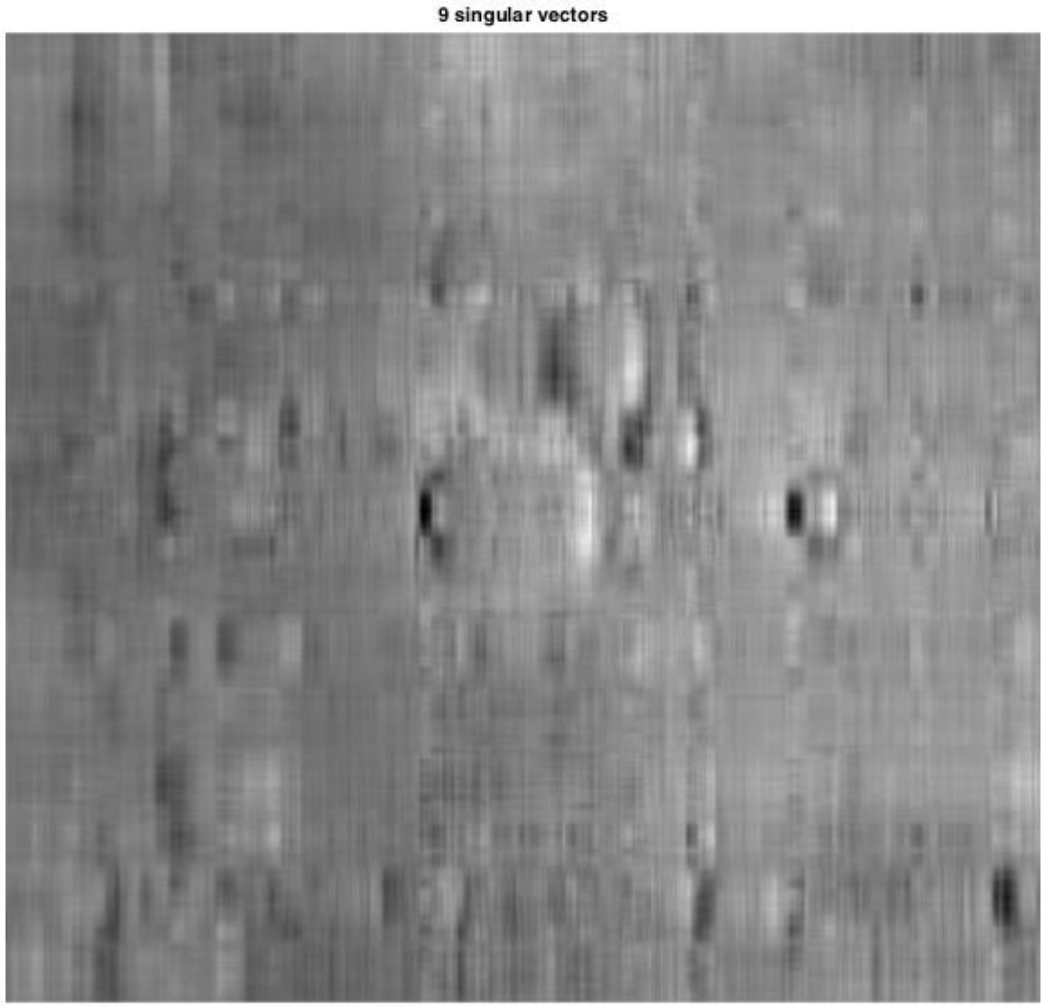
Singular Values



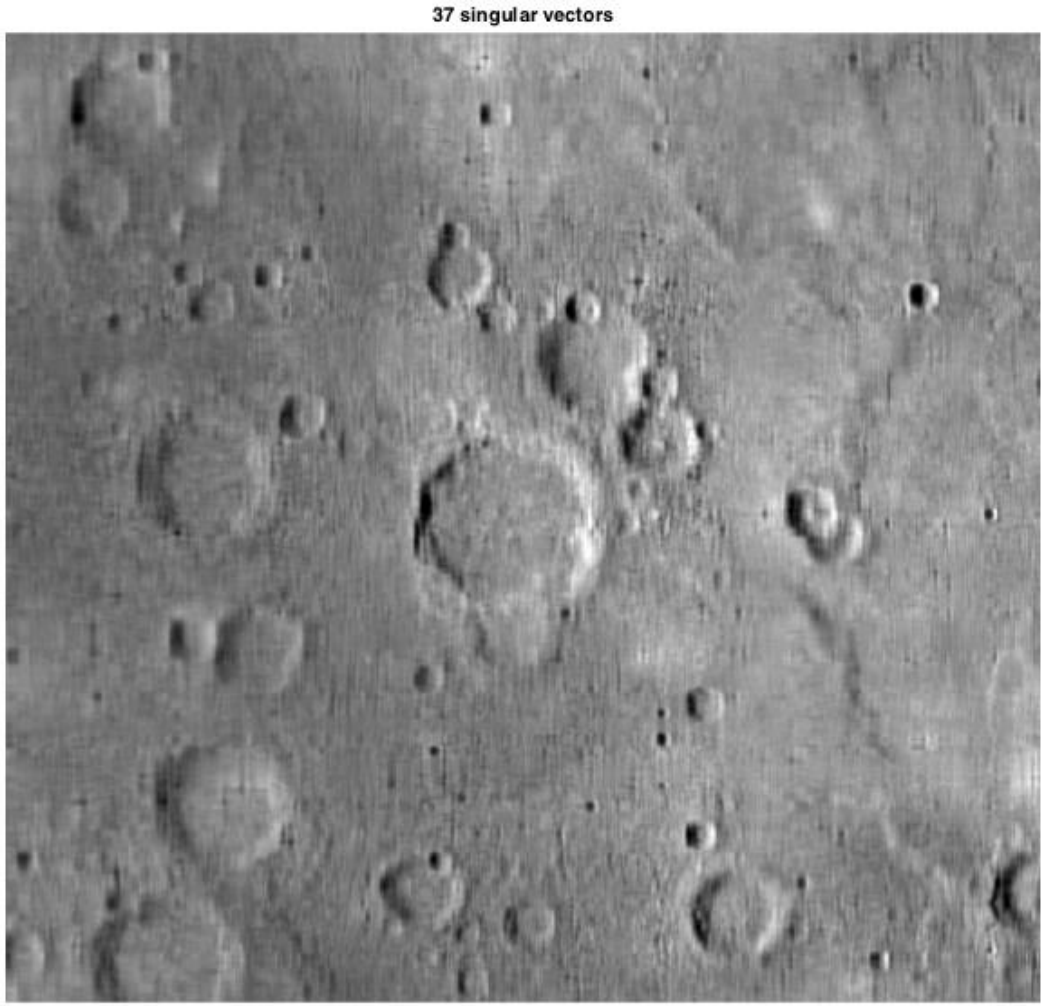
Compression error 0.05



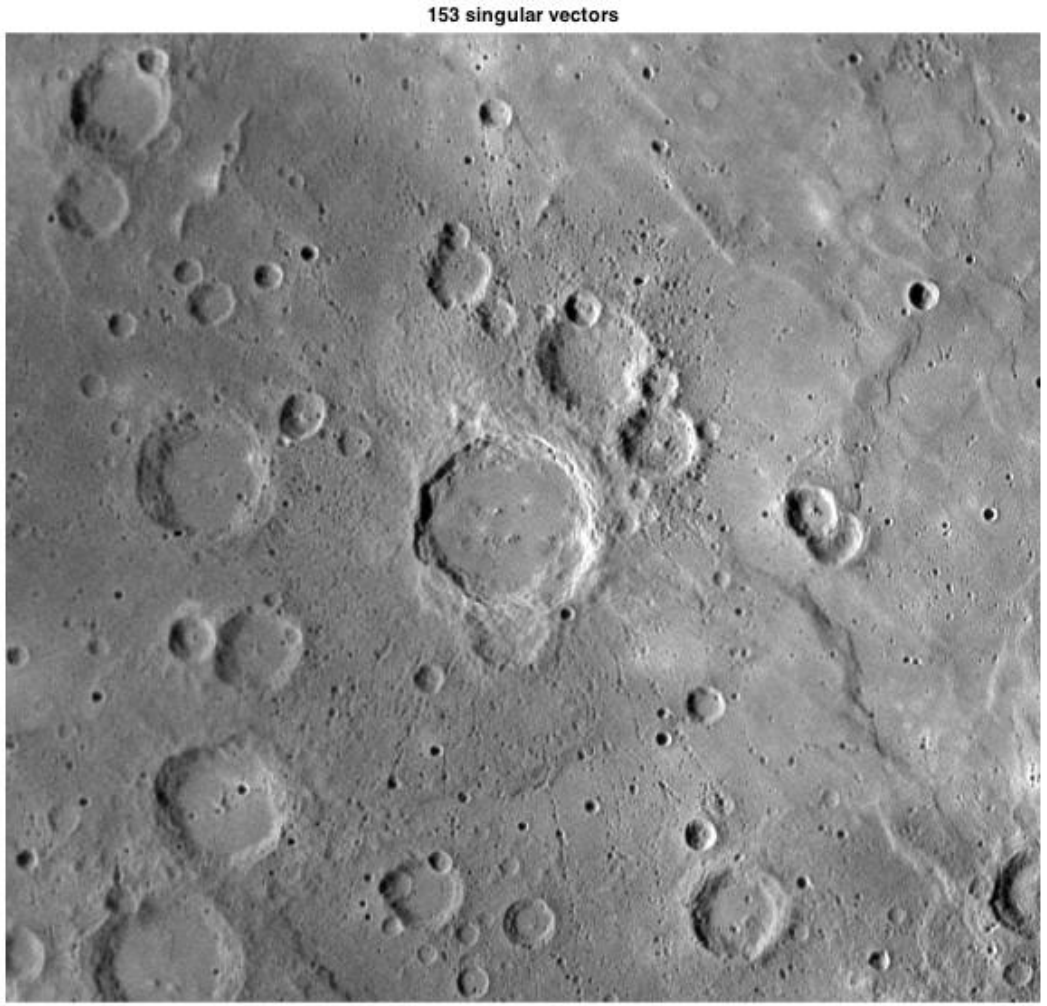
Compression error 0.01



Compression error 0.005



Compression error 0.001



Results

$\|A - \hat{A}\|_F / \|A\|_F$	Rank	Numbers to store	compression factor
0.05	1	2216	553
0.01	9	19944	61
0.005	37	81992	15
0.001	153	339048	3.6
0	1071	1225224	1

Numerical results for SVD image compression

How to compute the SVD?

- So far we have only discussed properties and applications of the SVD.

How to compute the SVD?

- So far we have only discussed properties and applications of the SVD.
- The next question that we discuss is: How can we compute the SVD?

How to compute the SVD?

- So far we have only discussed properties and applications of the SVD.
- The next question that we discuss is: How can we compute the SVD?
- Since we explicitly consider 'big data' we will discuss next a technique that is well suited for very large problems.

How to compute the SVD?

- So far we have only discussed properties and applications of the SVD.
- The next question that we discuss is: How can we compute the SVD?
- Since we explicitly consider 'big data' we will discuss next a technique that is well suited for very large problems.
- In principle we can compute the SVD by solving the eigenvalues for $A^T A$ (for the left and right singular vectors) and AA^T for the left singular vectors. But how do we solve these eigenproblems if A is very large?

How to compute the SVD?

- So far we have only discussed properties and applications of the SVD.
- The next question that we discuss is: How can we compute the SVD?
- Since we explicitly consider 'big data' we will discuss next a technique that is well suited for very large problems.
- In principle we can compute the SVD by solving the eigenvalues for $A^T A$ (for the left and right singular vectors) and AA^T for the left singular vectors. But how do we solve these eigenproblems if A is very large?
- Moreover, can't we compute the right and left singular vectors simultaneously?

The QR-algorithm for eigenvalues

The standard technique to solve small or dense eigenvalue problems, is the QR algorithm.
The method starts with the matrix A_0, factors it by Gram-Schmidt into $Q_0 R_0$ and then reverses the factors: $A_1 = R_0 Q_0$.
Repeating this process yields

$$A_k = Q_k R_k \qquad A_{k+1} = R_k Q_k$$

The matrix A_k becomes more and more upper triangular and finally the eigenvalues will be on the main diagonal.

Large and sparse problems

- Large and sparse problems do not admit algorithms that change the structure of the matrix (like the QR decomposition).

Large and sparse problems

- Large and sparse problems do not admit algorithms that change the structure of the matrix (like the QR decomposition).
- The way around this is to compute the eigenvalues/singular values of the matrix projected onto a (small) subspace (Model Order Reduction). The resulting small problem can be solved with the QR algorithm.

Large and sparse problems

- Large and sparse problems do not admit algorithms that change the structure of the matrix (like the QR decomposition).
- The way around this is to compute the eigenvalues/singular values of the matrix projected onto a (small) subspace (Model Order Reduction). The resulting small problem can be solved with the QR algorithm.
- A common choice is to project onto the so-called Krylov subspace.

The Krylov subspace

The space $span\{v, Av, A^2v, \cdots, A^{k-1}v\}$ is called the *Krylov subspace* of dimension k, corresponding to matrix A and initial vector v and is denoted by

$$K^k(A; v) = span\{v, Av, A^2v, \cdots, A^{k-1}v\}$$

The Krylov subspace

The space $span\{v, Av, A^2v, \cdots, A^{k-1}v\}$ is called the *Krylov subspace* of dimension k, corresponding to matrix A and initial vector v and is denoted by

$$K^k(A; v) = span\{v, Av, A^2v, \cdots, A^{k-1}v\}$$

Generating a basis for this subspace only requires some very basic operations:

- Matrix-vector multiplication
- Inner products
- Vector updates

None of these operations change the structure of the matrix

Algorithms to compute a basis

In order to project onto the Krylov subspace, we need a basis for it.
We will discuss three algorithms:

- Arnoldi's method for nonsymmetric matrices
- Lanczos method for symmetric matrices
- Lanczos bi-diagonalisation method for general (non-square matrices)

All these method compute an *orthonormal basis*.

Arnoldi's method

Choose a starting vector q_1 with $\|q_1\|_2 = 1$.

FOR	$k = 1, \cdots$ DO	iteration
	$v = Aq_k$	expansion
	FOR $i = 1, ..., k$	orthogonalisation
	$\quad h_{i,k} = q_i^T v$	
	$\quad v = v - h_{i,k} q_i$	
	END FOR	
	$h_{k+1,k} = \|v\|_2$	
	IF $h_{k+1,k} = 0$ STOP	invariant subspace spanned
	$q_{k+1} = v / h_{k+1,k}$	new basis vector
END	FOR	

The Arnoldi relation

The Arnoldi method can be summarised in a compact way. Let

$$H_k = \begin{bmatrix} h_{1,1} & \cdots & \cdots & h_{1,k} \\ h_{2,1} & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ O & & h_{k,k-1} & h_{k,k} \end{bmatrix}$$

and $Q_k = [q_1 \; q_2 \cdots q_k]$ then

$$AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k^T$$

Here e_k is the $k-th$ canonical basis vector in $\mathbb{R}^k$.

Solving an eigenvalue probem

If we want to approximately solve an eigenvalue problem

$$Ax = \lambda x$$

we can construct vector $\tilde{x} \in K^k(A; q_1)$ by setting $\tilde{x} = Q_k y$. Substituting and using a Galerkin condition gives

$$Q_k^T A Q_k y = \tilde{\lambda} Q_k^T Q_k y \quad \Leftrightarrow \quad H_k y = \tilde{\lambda} y$$

This small eigenvalue problem can be solved with the QR method. The eigenvalues of H_k are called Ritzvalues.

A is symmetric

According to the Arnoldi relation we have

$$Q_k^T A Q_k = H_k \ .$$

Moreover, if A is symmetric we have

$$H_k^T = Q_k^T A^T Q_k = Q_k^T A Q_k = H_k \ .$$

So if A is symmetric, H_k is symmetric and upper Hessenberg, this means that H_k must be tridiagonal.

A is symmetric (2)

So

$$H_k = \begin{bmatrix} h_{1,1} & h_{2,1} & & O \\ h_{2,1} & \ddots & \ddots & \\ & \ddots & \ddots & h_{k,k-1} \\ O & & h_{k,k-1} & h_{k,k} \end{bmatrix}.$$

With $\alpha_k = h_{k,k}$ and $\beta_k = h_{k-1,k}$ the Arnoldi method simplifies to the (symmetric) Lanczos method.

With the Lanczos method it is possible to compute a new orthonormal basis vector using only the two previous basis vectors.

The Lanczos method

Choose a starting vector q_1 with $\|q_1\|_2 = 1$

$\beta_1 = 0$	$q_0 = 0$	initialization
FOR	$k = 1, \cdots$ DO	iteration
	$\alpha_k = q_k^T A q_k$	
	$v = Aq_k - \alpha_k q_k - \beta_k q_{k-1}$	new direction
		orthogonal to
	$\beta_{k+1} = \|v\|_2$	previous q
	$q_{k+1} = v/\beta_{k+1}$	normalization
END FOR		

The Lanczos method

Let

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & & 0 \\ \beta_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & \beta_k \\ & & & \beta_k & \alpha_k \end{bmatrix}.$$

and

$$Q_k = [q_1 \; q_2 \ldots q_k]$$

Then $AQ_k = Q_k T_k + \beta_{k+1} q_{k+1} e_k^T$

Lanczos bi-diagonalisation

The Lanczos bi-diagonalisation algorithm is derived by applying Lanczos to

$$\begin{pmatrix} O & A \\ A^T & 0 \end{pmatrix}$$

with starting vector $q_1 = \frac{1}{\|w\|} \begin{pmatrix} w \\ 0 \end{pmatrix}$

Lanczos bi-diagonalisation (2)

The second vector in the Krylov subspace becomes

$$\frac{1}{\|w\|}\begin{pmatrix} 0 \\ A^T w \end{pmatrix}$$

After normalisation we obtain

$$\frac{1}{\|A^T w\|}\begin{pmatrix} 0 \\ A^T w \end{pmatrix}$$

Repeating this procedure shows that we get alternatingly orthogonal vectors $\begin{pmatrix} u \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ v \end{pmatrix}$

Lanczos bi-diagonalisation (3)

This observation leads to the following

Bi-diagonalisation algorithm (Golub and Kahan)

Choose w

$\beta_1 u_1 = w \quad \alpha_1 v_1 = A^T u_1$

FOR $i = 1, \cdots$ DO

$$\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i$$
$$\alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i$$

END FOR

with $\alpha_i > 0$ and $\beta_i > 0$ such that $\|u_i\| = \|v_i\| = 1$.

Lanczos bi-diagonalisation

With $U_k = [u_1, u_2, \cdots, u_k], \quad V_k = [v_1, v_2, \cdots, v_k]$ and

$$B_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix},$$

it follows that

$$\begin{aligned} AV_k &= U_{k+1}B_k \\ A^T U_{k+1} &= V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T \end{aligned}$$

Computing the SVD

Remember that a singular value σ and its left and right singular vectors u and $v$are related by

$$Av = \sigma u \quad , \quad A^T u = \sigma v$$

Now consider this relation for B_k

$$B_k \tilde{v} = \tilde{\sigma} \tilde{u} \quad , \quad B_k^T \tilde{u} = \tilde{\sigma} \tilde{v}$$

Substitution in the Lanczos bi-diagonalisation recurrences gives

$$\begin{aligned} AV_k \tilde{v} &= \tilde{\sigma} U_{k+1} \tilde{u} \\ A^T U_{k+1} \tilde{u} &= \tilde{\sigma} V_k \tilde{v} + \alpha_{k+1} v_{k+1} e_{k+1}^T \tilde{u} \end{aligned}$$

So the singular values of B_k converge to the singular values of A, and the vectors $U_{k+1}\tilde{u}$ and $V_k \tilde{v}$ to the left and right singular vectors.

Concluding remarks

- Today we discussed the SVD, and some applications
- We also discussed a technique to compute the SVD that is suitable for large and sparse problems
- Next week we look at a related subject: the solution of large least-squares problems with an application to computerized tomography (CT)