Numerical Linear Algebra
Lecture 1

- Course homepage
  http://www.math.chalmers.se/Math/Grundutb/CTH/tma265/1718

- Course homepage
  http://www.math.chalmers.se/Math/Grundutb/CTH/tma265/1718
- Course coordinator: Larisa Beilina
  larisa@chalmers.se, room 2089

- Course homepage
  http://www.math.chalmers.se/Math/Grundutb/CTH/tma265/1718
- Course coordinator: Larisa Beilina
  larisa@chalmers.se, room 2089
- Registration for the course: contact studieadministrator
  Jeanette Montell, jw@chalmers.se

- Course homepage
  http://www.math.chalmers.se/Math/Grundutb/CTH/tma265/1718
- Course coordinator: Larisa Beilina
  larisa@chalmers.se, room 2089
- Registration for the course: contact studieadministrator
  Jeanette Montell, jw@chalmers.se
- Course literature:
  In this year we will have still this book:
  James W. Demmel: Applied Numerical Linear Algebra, SIAM 1997.
  And smootly go to the book:
  L. Beilina, E. Karchevskii, M. Karchevskii, Numerical Linear Algebra: Theory and Applications, Springer, 2017. Will be selled at Cremona after publication. Now draft of the book is available at the homepage of the course.

- Check course homepage for news. Here some numbers of Master's works in applied mathematics are advertised:
- 1) Efficient implementation of Helmholtz equation with applications in medical imaging
- 2) Optimal control of drugs in the mathematical model of dynamics of a tumor-immune system.
  Main paper for the project "Optimal control of drugs in the mathematical model of dynamics of a tumor-immune system" is available for download from the homepage.
- 3) Optimization approach in the design of approximate cloaking structures

Course in Numerical Linear Algebra
Schedule

Day	Time	Place	
Mon	13:15-15:00	Euler	Lecture
Thu	10:00-11:45	Pascal	Lecture
Wed	13:15-15:00	MVF24,MVF25	Computer Labs
Fr	13:15-15:00	MVF24,MVF25	Computer Labs
October	14.00-18.00	Maskinhuset	Exam
January	14.00-18.00	Maskinhuset	Exam

- To pass this course you should pass the written exam and 2 computer assignments, see description of comp. assignments at the course homepage.
- The two compulsory home assignments should be handed in before the final exam.
- Programs can be written in Matlab or C++/PETSc.
- The final exam is compulsory, written.
- The theory questions will be choosen from the list which is possible download from the course homepage.

Grades

Bonuspoints will be added to the points obtained at written exam. Final grades will be the following:

Grades Chalmers	Points
-	< 15
3	15-20
4	21-27
5	> 27
Grades GU	**Points**
U	< 15
G	15-27
VG	> 27

Deadlines for homeworks and comp. labs

Deadlines for homeworks and comp.ex.:

- Homework 1 and comp. ex. 1: 15 September
- Homework 2: 22 September
- Homework 3 and comp.ex. 2: 6 October
- Homework 4 and comp.ex. 3: 13 October (last comp.lab)
- Comp.ex. 4,5: 20 October

Comp.ex. can be done in groups by 2 students. Reports for homeworks and comp.labs (together with programs) should be sent to my e-mail before the deadline. Hand-written homeworks can be returned directly to me or putted into the red box which is located behind my office.

Comp.labs in PETSc

- PETSc libraries which are a suite of data structures and routines for the scalable (parallel) solution of scientific applications.
- Link to the PETSc documentation: http://www.mcs.anl.gov/petsc/documentation/
- Template for solution of system of equations $Ax = b$ using PETSc is available for download from the course homepage.

PETSc: example of Makefile for running at Chalmers

```
PETSC_ARCH=/chalmers/sw/sup64/petsc-3.7.4
include ${PETSC_ARCH}/lib/petsc/conf/variables
include ${PETSC_ARCH}/lib/petsc/conf/rules
CXX=g++
CXXFLAGS=-Wall -Wextra -g -O0 -c -linclude
-I${PETSC_ARCH}/include
LD=g++
LFLAGS=
OBJECTS=Main.o CG.o Create.o DiscretePoisson2D.o
GaussSeidel.o Jacobi.o PCG.o Solver.o SOR.o
Run=Main
all: $(Run)
%.o: %.cpp $(CXX) $(CXXFLAGS) -o $@ $<
$(Run): $(OBJECTS) $(LD) $(LFLAGS) $(OBJECTS)
$(PETSC_LIB) -o $@
```

- Solve Linear systems of equations using Gaussian elimination with different pivoting strategies and blocking algorithms

- Solve Linear systems of equations using Gaussian elimination with different pivoting strategies and blocking algorithms
- Study and use QR decomposition and SVD decomposition

- Solve Linear systems of equations using Gaussian elimination with different pivoting strategies and blocking algorithms
- Study and use QR decomposition and SVD decomposition
- Solve eigenvalue problems based on transformation techniques for symmetric and non-symmetric matrices

- Solve Linear systems of equations using Gaussian elimination with different pivoting strategies and blocking algorithms
- Study and use QR decomposition and SVD decomposition
- Solve eigenvalue problems based on transformation techniques for symmetric and non-symmetric matrices
- Use computer algorithms, programs and software packages (BLAS/LAPACK, MATLAB, PETSC)

- Solve Linear systems of equations using Gaussian elimination with different pivoting strategies and blocking algorithms
- Study and use QR decomposition and SVD decomposition
- Solve eigenvalue problems based on transformation techniques for symmetric and non-symmetric matrices
- Use computer algorithms, programs and software packages (BLAS/LAPACK, MATLAB, PETSC)
- Solve real physical problems by modelling these problems via NLA

- A linear system is a mathematical model of a system which uses definition of a linear operator. Linear systems have important applications in automatic control theory, signal processing, and telecommunications. For example, the propagation medium for wireless communication systems can often be modeled by linear systems.
- A general deterministic system can be described by operator, H, that maps an input, x(t), as a function of t to an output, y(t), a type of black box description. Linear systems satisfy the properties of superposition and scaling or homogeneity. Given two valid inputs $x_1(t), x_2(t)$ as well as their respective outputs

$$y_1(t) = H\{x_1(t)\}; y_2(t) = H\{x_2(t)\}$$

a linear system must satisfy to the equation

$$\alpha y_1(t) + \beta y_2(t) = H\{\alpha x_1(t) + \beta x_2(t)\}$$

for any scalar values of α and β.

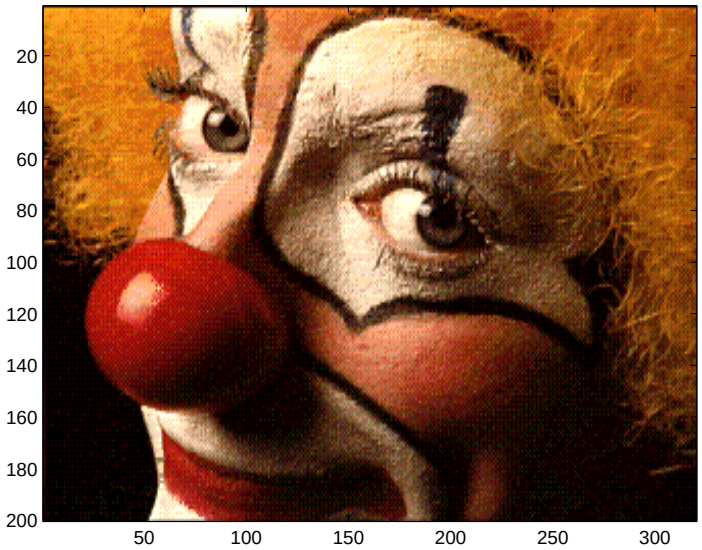
Example of application of linear systems: image compression using SVD

Definition SVD *Let A be an arbitrary m-by-n matrix with $m \geq n$. Then we can write $A = U\Sigma V^T$, where U is m-by-n and satisfies $U^T U = I$, V is n-by-n and satisfies $V^T V = I$, and $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$, where $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. The columns $u_1, \ldots, u_n$ of U are called $left\ singular\ vectors$. The columns $v_1, \ldots, v_n$ of V are called $right\ singular\ vectors$. The σ_i are called $singular\ values$. (If $m < n$, the SVD is defined by considering A^T.)*

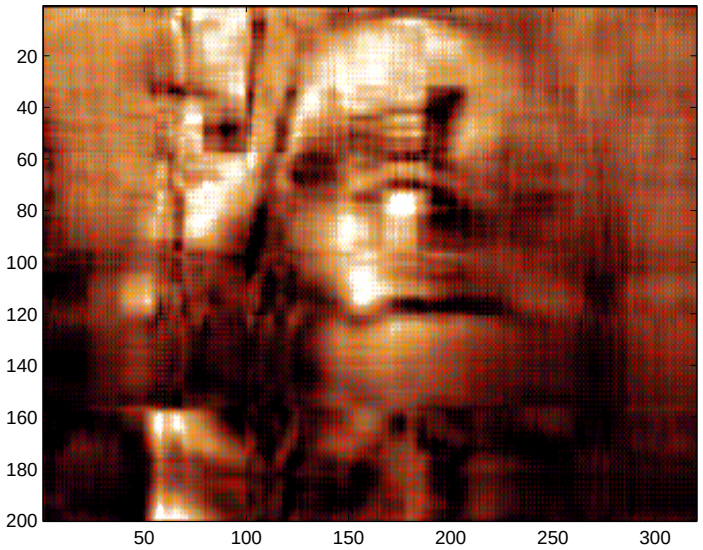
Theorem

Write $V = [v_1, v_2, \ldots, v_n]$ and $U = [u_1, u_2, \ldots, u_n]$, so $A = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T$ (a sum of rank-1 matrices). Then a matrix of rank $k < n$ closest to A (measured with $||\cdot||_2$ is $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ and $||A - A_k||_2 = \sigma_{k+1}$. We may also write $A_k = U\Sigma_k V^T$ where $\Sigma_k = diag(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$.

Example of application of linear systems: image compression using SVD



a) Original image



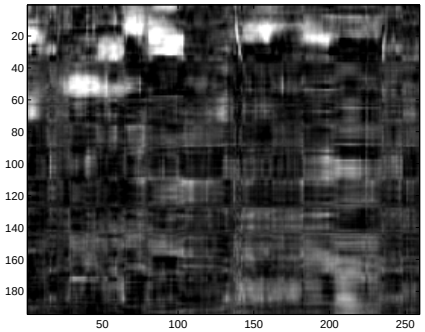
b) Rank k=20 approximation

Example of application of linear systems: image compression using SVD in Matlab

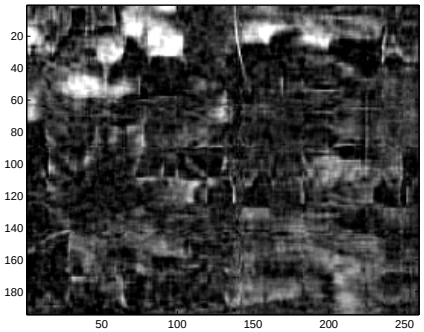
See path for other pictures:
/matlab-2012b/toolbox/matlab/demos
load clown.mat;
Size(X) $= m \times n = 320 \times 200$ pixels.
[U,S,V] = svd(X);
colormap(map);
k=20;
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
Now: size(U)= $m \times k$, size(V)= $n \times k$.

Example of application of linear systems: image compression using SVD in Matlab

a) Original image



b) Rank k=10 approximation



c) Rank k=20 approximation

d) Rank k=50 approximation

Example of application of linear systems: image compression using SVD for arbitrary image

To get image on the previous slide, I took picture in jpg-format and loaded it in matlab like that:
A = imread('autumn.jpg');
You can not simply apply SVD to A: svd(A) Undefined function 'svd' for input arguments of type 'uint8'.
Apply type "double" to A: DA = double(A), and then perform
[U,S,V] = svd(DA);
colormap('gray');
k=20;
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
Now: size(U)= $m \times k$, size(V)= $n \times k$.

Example of application of linear systems: image deblurring

Original Image

Blurred Image

Figure: left: exact matrix $\mathbf{X}$, right: approximated matrix $\mathbf{B}$

The blurring model

Consider a grayscale image

- $\mathbf{X}$: $m \times n$ matrix representing the exact image
- $\mathbf{B}$: $m \times n$ matrix representing the blurred image

The blurring model

Consider a grayscale image

- $\mathbf{X}$: $m \times n$ matrix representing the exact image
- $\mathbf{B}$: $m \times n$ matrix representing the blurred image

Assume linear blurring.

$$\mathbf{x} = \mathrm{vec}(\mathbf{X}) = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \in \mathbb{R}^N, \quad \mathbf{b} = \mathrm{vec}(\mathbf{B}) = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} \in \mathbb{R}^N$$

$\mathbf{A}$ $N \times N$ matrix, with $N = m \cdot n$

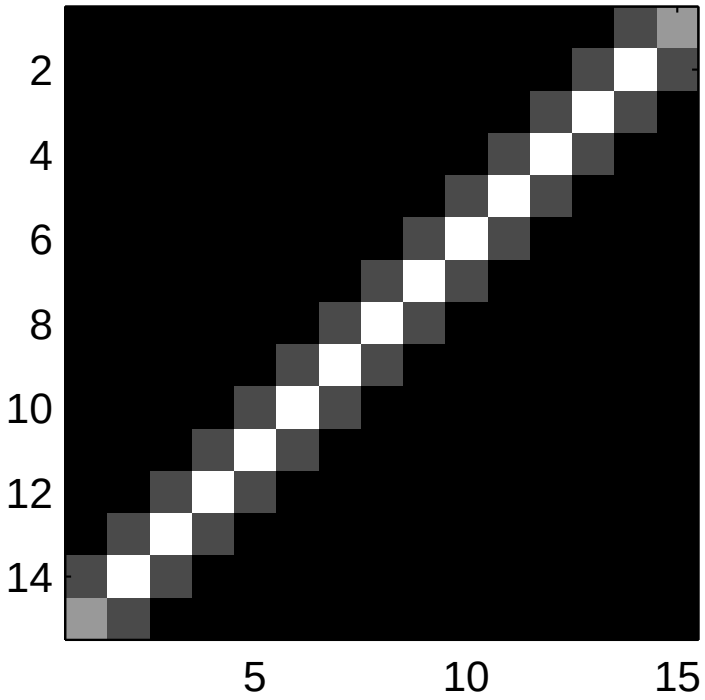
$$\mathbf{Ax} = \mathbf{b}$$

Knowing $\mathbf{X}$ and $\mathbf{A}$ it is straightforward to compute the blurred image.

Motion blur

Motion Blurred Image

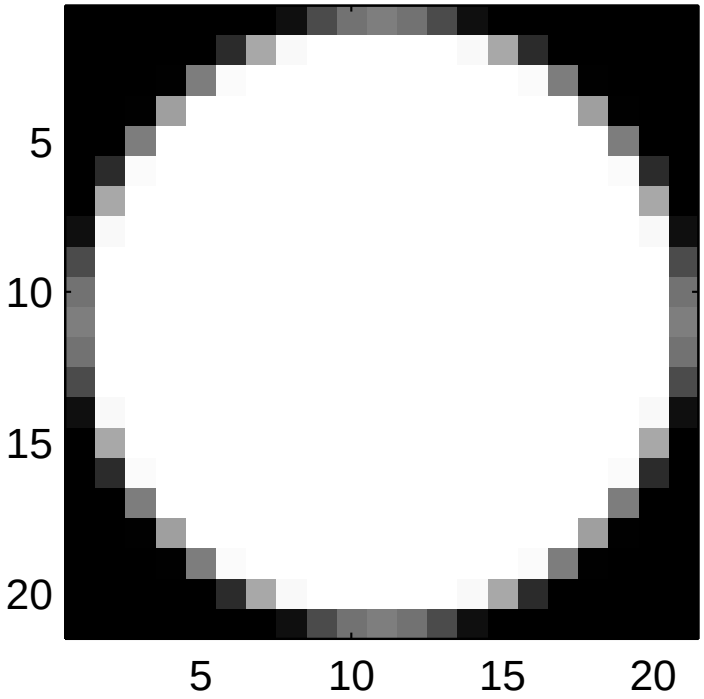
PSF



Out-of-focus blur

Blurred Image

PSF



Gaussian blur

Gaussian Blurred Image

PSF

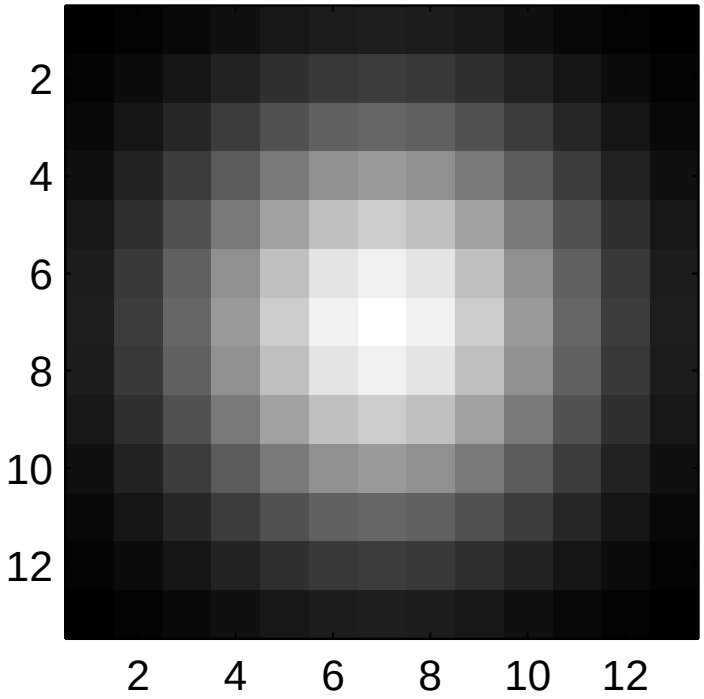


Image deblurring: solution of an inverse problem

Let H be the Hilbert space H^1 and let $\Omega \subset \mathbb{R}^m, m = 2, 3$, be a convex bounded domain. Our goal is to solve a Fredholm integral equation of the first kind for $x \in \Omega$

$$\int_{\Omega} K(x-y)z(x)dx = u(y), \tag{1}$$

where $u(y) \in L_2(\bar{\Omega})$, $z(x) \in H$, $K(x-y) \in C^k\left(\overline{\Omega}\right), k \geq 0$ be the kernel of the integral equation.
Let us rewrite (1) in an operator form as

$$A(z) = u \tag{2}$$

with an operator $A : H \to L_2(\bar{\Omega})$ defined as

$$A(z) := \int_{\Omega} K(x-y)z(x)dx. \tag{3}$$

Ill-posed problem.

Let the function $z(x) \in H^1$ of the equation (1) be unknown in the domain Ω. Determine the function $z(x)$ for $x \in \Omega$ assuming the functions $K(x-y) \in C^k\left(\overline{\Omega}\right), k \geq 0$ and $u(x) \in L_2(\Omega)$ in (1) are known.
Let $\delta > 0$ be the error in the right-hand side of the equation (1):

$$A\left(z^*\right) = u^*, \quad \left\|u - u^*\right\|_{L_2(\sigma)} \leq \delta. \tag{4}$$

where u^* is the exact right-hand side corresponding to the exact solution z^*.
To find the approximate solution of the equation (1) we minimize the functional

$$M_\alpha\left(z\right) = \|Az - u\|^2_{L_2(\Omega)} + \alpha\|z\|^2_{H_1(\Omega)}, \tag{5}$$

$$M_\alpha : H^1 \to \mathbb{R},$$

where $\alpha = \alpha\left(\delta\right) > 0$ is the small regularization parameter.

We consider now more general form of the Tikhonov functional (5). Let W_1, W_2, Q be three Hilbert spaces, $Q \subseteq W_1$ as a set, the norm in Q is stronger than the norm in W_1 and $\overline{Q} = W_1$, where the closure is understood in the norm of W_1. We denote scalar products and norms in these spaces as

$$(\cdot,\cdot), \|\cdot\| \text{ for } W_1,$$

$$(\cdot,\cdot)_2, \|\cdot\|_2 \text{ for } W_2$$

$$\text{and } [\cdot,\cdot], [\cdot] \text{ for } Q.$$

Let $A : W_1 \to W_2$ be a bounded linear operator. Our goal is to find the function $z(x) \in Q$ which minimizes the Tikhonov functional

$$E_\alpha(z) : Q \to \mathbb{R}, \tag{6}$$

$$E_\alpha(z) = \frac{1}{2}\|Az - u\|_2^2 + \frac{\alpha}{2}[z - z_0]^2, u \in W_2; z, z_0 \in Q, \tag{7}$$

where $\alpha \in (0,1)$ is the regularization parameter. To do that we search for a stationary point of the above functional with respect to z satisfying $\forall b \in Q$

$$E'_\alpha(z)(b) = 0. \tag{8}$$

The following lemma is well known for the case $W_1 = W_2 = L_2$.

Lemma 1. *Let $A : L_2 \to L_2$ be a bounded linear operator. Then the Fréchet derivative of the functional (5) is*

$$E'_\alpha(z)(b) = (A^*Az - A^*u, b) + \alpha[z - z_0, b], \forall b \in Q. \tag{9}$$

In particular, for the integral operator (1) we have

$$\begin{aligned} E'_\alpha(z)(b) = \int_\Omega b(s) \Bigg[\int_\Omega z(y) \left(\int_\Omega K(x-y)K(x-s)dx \right) dy \\ - \int_\Omega K(x-s)u(x)\,dx \Bigg] ds \end{aligned} \tag{10}$$

$$+\alpha[z - z_0, b], \forall b \in Q.$$

Lemma 2 is also well known, since $A : W_1 \to W_2$ is a bounded linear operator. We formulate this lemma only for our specific case.

Lemma 2. *Let the operator $A : W_1 \to W_2$ satisfies conditions of Lemma 1. Then the functional $E_\alpha(z)$ is strongly convex on the space Q with the convexity parameter κ* such that

$$(E'_\alpha(x) - E'_\alpha(z), x - z) \geq \kappa[x - z]^2, \forall x, z \in Q. \tag{11}$$

Similarly, the functional $M_\alpha(z)$ is also strongly convex on the Sobolev space H_1:

$$\left(M'_\alpha(x) - M'_\alpha(z), x - z\right)_{H_1} \geq \kappa ||x - z||^2_{H_1}, \forall x, z \in H_1, \quad (12)$$

Find z via any gradient-like method. For example, perform usual gradient update

$$z^{k+1} = z^k + \beta E'_\alpha\left(z^k\right)(b). \quad (13)$$

until $||z^{k+1} - z^k||$ converges.

Notions from linear algebra

- Matrices (Identity matrix, triangular, singular, symmetric, positive definite, conjugate transpose, rank, norm)

Notions from linear algebra

- Matrices (Identity matrix, triangular, singular, symmetric, positive definite, conjugate transpose, rank, norm)
- Matrix operations, inverse, transposition, scalar (inner) product, outer product

Notions from linear algebra

- Matrices (Identity matrix, triangular, singular, symmetric, positive definite, conjugate transpose, rank, norm)
- Matrix operations, inverse, transposition, scalar (inner) product, outer product
- Gaussian elimination

Notions from linear algebra

- Matrices (Identity matrix, triangular, singular, symmetric, positive definite, conjugate transpose, rank, norm)
- Matrix operations, inverse, transposition, scalar (inner) product, outer product
- Gaussian elimination
- Eigenvalues

Notions from linear algebra

- Matrices (Identity matrix, triangular, singular, symmetric, positive definite, conjugate transpose, rank, norm)
- Matrix operations, inverse, transposition, scalar (inner) product, outer product
- Gaussian elimination
- Eigenvalues
- Norms

Notions from linear algebra

- Matrices (Identity matrix, triangular, singular, symmetric, positive definite, conjugate transpose, rank, norm)
- Matrix operations, inverse, transposition, scalar (inner) product, outer product
- Gaussian elimination
- Eigenvalues
- Norms
- LU-factorization, pivoting, row echelon form

Identity matrix

The identity matrix or unit matrix of size n is the $n \times n$ square matrix with ones on the main diagonal and zeros elsewhere. It is denoted by I_n, or simply by I.

$$I_1 = \begin{bmatrix} 1 \end{bmatrix}, \; I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \; \cdots, \quad I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

When A has size $m \times n$, it is a property of matrix multiplication that $I_m A = A I_n = A$.

Using the notation that is sometimes used to concisely describe diagonal matrices, we can write:

$I_n = \operatorname{diag}(1, 1, ..., 1)$.

It can also be written using the Kronecker delta notation:

$(I_n)_{ij} = \delta_{ij}$.

Triangular matrix

- A square matrix is called lower triangular if all the entries above the main diagonal are zero.

$$L = \begin{bmatrix} l_{1,1} & & & & 0 \\ l_{2,1} & l_{2,2} & & & \\ l_{3,1} & l_{3,2} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{n,1} & l_{n,2} & \dots & l_{n,n-1} & l_{n,n} \end{bmatrix}$$

- A square matrix is called upper triangular if all the entries below the main diagonal are zero.

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \dots & u_{1,n} \\ & u_{2,2} & u_{2,3} & \dots & u_{2,n} \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & u_{n-1,n} \\ 0 & & & & u_{n,n} \end{bmatrix}$$

Triangular matrix

- A triangular matrix is one that is either lower triangular or upper triangular.
- A matrix that is both upper and lower triangular is a diagonal matrix.

$$D_n = \begin{bmatrix} d_{1,1} & 0 & \cdots & 0 \\ 0 & d_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{n,n} \end{bmatrix}$$

Singular matrix

A square matrix that does not have a matrix inverse. A matrix is singular if its determinant is 0. For example, there are 10 2×2 singular $(0, 1)$-matrices:

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Symmetric and positive definite matrix

- A symmetric matrix is a square matrix that is equal to its transpose. Let A be a symmetric matrix. Then:

$$A = A^T.$$

  If the entries of matrix A are written as $A = (a_{ij})$, then the symmetric matrix A is such that $a_{ij} = a_{ji}$.

- An $n \times n$ real matrix M is positive definite if $z^T Mz > 0$ for all non-zero vectors z with real entries ($z \in \mathbb{R}^n$), where z^T denotes the transpose of z.

- An $n \times n$ Hermitian matrix M is positive definite if $z^* Mz$ is real and positive for all non-zero complex vectors z, where z^* denotes the conjugate transpose of z.

Examples

- The following matrix is symmetric:

$$\begin{bmatrix} 1 & 7 & 3 \\ 7 & 4 & -5 \\ 3 & -5 & 6 \end{bmatrix}.$$

- Every diagonal matrix is symmetric, since all off-diagonal entries are zero.

Examples

- The nonnegative matrix

$$M_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

is positive definite.

For a vector with entries

$$\mathbf{z} = \begin{bmatrix} z_0 \\ z_1 \end{bmatrix}$$

the quadratic form is

$$\begin{bmatrix} z_0 & z_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \end{bmatrix} = \begin{bmatrix} z_0 \cdot 1 + z_1 \cdot 0 & z_0 \cdot 0 + z_1 \cdot 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \end{bmatrix} = z_0^2 + z_1^2;$$

when the entries $z0, z1$ are real and at least one of them nonzero, this is positive.

A matrix in which some elements are negative may still be positive-definite. An example is given by

$$M_1 = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}.$$

It is positive definite since for any non-zero vector

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

we have

$$x^{\mathrm{T}} M_1 x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= \begin{bmatrix} (2x_1 - x_2) & (-x_1 + 2x_2 - x_3) & (-x_2 + 2x_3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= 2{x_1}^2 - 2x_1x_2 + 2{x_2}^2 - 2x_2x_3 + 2{x_3}^2$$

$$= {x_1}^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + {x_3}^2$$

which is a sum of squares and therefore nonnegative; in fact, each squared summa can be zero only when $x_1 = x_2 = x_3 = 0$, so M_1 is indeed positive-definite.

Conjugate transpose matrix

The conjugate transpose, Hermitian transpose, Hermitian conjugate, or adjoint matrix of an m-by-n matrix A with complex entries is the n-by-m matrix A^* obtained from A by taking the transpose and then taking the complex conjugate of each entry (i.e., negating their imaginary parts but not their real parts). The conjugate transpose is formally defined by

$$(\mathbf{A}^*)_{ij} = \overline{\mathbf{A}_{ji}}$$

where the subscripts denote the i, j-th entry, and the overbar denotes a scalar complex conjugate. (The complex conjugate of $a + bi$, where a and b are reals, is $a - bi$.)
This definition can also be written as

$$\mathbf{A}^* = (\overline{\mathbf{A}})^{\mathrm{T}} = \overline{\mathbf{A}^{\mathrm{T}}}$$

where $\mathbf{A}^{\mathrm{T}}$ denotes the transpose and $\overline{\mathbf{A}}$, denotes the matrix with complex conjugated entries.

The conjugate transpose of a matrix A can be denoted by any of these symbols:

$$\mathbf{A}^* \text{ or } \mathbf{A}^{\mathrm{H}},$$

commonly used in linear algebra.
Example
If

$$\mathbf{A} = \begin{bmatrix} 3+i & 5 & -2i \\ 2-2i & i & -7-13i \end{bmatrix}$$

then

$$\mathbf{A}^* = \begin{bmatrix} 3-i & 2+2i \\ 5 & -i \\ 2i & -7+13i \end{bmatrix}$$

Basic remarks

- A square matrix A with entries a_{ij} is called Hermitian or self-adjoint if $A = A^*$, i.e., $a_{ij} = \overline{a_{ji}}$.
- normal if $A^*A = AA^*$.
- unitary if $A^* = A^{-1}$. a unitary matrix is a (square) $n \times n$ complex matrix A satisfying the condition $A^*A = AA^* = I_n$, where I_n is the identity matrix in n dimensions.
- Even if A is not square, the two matrices A^*A and AA^* are both Hermitian and in fact positive semi-definite matrices.
- Finding the conjugate transpose of a matrix A with real entries reduces to finding the transpose of A, as the conjugate of a real number is the number itself.

Row echelon form

In linear algebra a matrix is in row echelon form if

- All nonzero rows (rows with at least one nonzero element) are above any rows of all zeroes [All zero rows, if any, belong at the bottom of the matrix]
- The leading coefficient (the first nonzero number from the left, also called the pivot) of a nonzero row is always strictly to the right of the leading coefficient of the row above it.
- All entries in a column below a leading entry are zeroes (implied by the first two criteria).

This is an example of 3×4 matrix in row echelon form:

$$\left[\begin{array}{ccc|c} 1 & a_1 & a_2 & a_3 \\ 0 & 2 & a_4 & a_5 \\ 0 & 0 & -1 & a_6 \end{array}\right]$$

Row echelon form

A matrix is in reduced row echelon form (also called row canonical form) if it satisfies the additional condition: Every leading coefficient is 1 and is the only nonzero entry in its column, like in this example:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & b_1 \\ 0 & 1 & 0 & b_2 \\ 0 & 0 & 1 & b_3 \end{array}\right]$$

Note that this does not always mean that the left of the matrix will be an identity matrix. For example, the following matrix is also in reduced row-echelon form:

$$\left[\begin{array}{cccc|c} 1 & 0 & 1/2 & 0 & b_1 \\ 0 & 1 & -1/3 & 0 & b_2 \\ 0 & 0 & 0 & 1 & b_3 \end{array}\right]$$

Rank

- Column rank of a matrix A is the maximum number of linearly independent column vectors of A. The row rank of a matrix A is the maximum number of linearly independent row vectors of A. Equivalently, the column rank of A is the dimension of the column space of A, while the row rank of A is the dimension of the row space of A.
- A result of fundamental importance in linear algebra is that the column rank and the row rank are always equal. It is commonly denoted by either $rk(A)$ or *rank* A. Since the column vectors of A are the row vectors of the transpose of A (denoted here by A^T), column rank of A equals row rank of A is equivalent to saying that the rank of a matrix is equal to the rank of its transpose, i.e. $rk(A) = rk(A^T)$.
- The rank of an $m \times n$ matrix cannot be greater than m nor n. A matrix that has a rank as large as possible is said to have full rank; otherwise, the matrix is rank deficient.

Cofactor

In linear algebra, the cofactor (sometimes called adjunct, see below) describes a particular construction that is useful for calculating both the determinant and inverse of square matrices. Specifically the cofactor of the (i, j) entry of a matrix, also known as the (i, j) cofactor of that matrix, is the signed minor of that entry.

Informal approach to minors and cofactors

Finding the minors of a matrix A is a multi-step process:

- Choose an entry a_{ij} from the matrix.
- Cross out the entries that lie in the corresponding row i and column j.
- Rewrite the matrix without the marked entries.
- Obtain the determinant M_{ij} of this new matrix.

If $i + j$ is an even number, the cofactor C_{ij} of a_{ij} coincides with its minor: $C_{ij} = M_{ij}$.

Otherwise, it is equal to the additive inverse of its minor: $C_{ij} = -M_{ij}$.

Formal definition of cofactor

If A is a square matrix, then the minor of its entry a_{ij}, also known as the (i, j) minor of A, is denoted by M_{ij} and is defined to be the determinant of the submatrix obtained by removing from A its i-th row and j-th column.
It follows: $C_{ij} = (-1)^{i+j} M_{ij}$ and C_{ij} is called the cofactor of a_{ij}.

Example

Given the matrix

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

suppose we wish to find the cofactor C_{23}. The minor M_{23} is the determinant of the above matrix with row 2 and column 3 removed.

$$M_{23} = \begin{vmatrix} b_{11} & b_{12} & \square \\ \square & \square & \square \\ b_{31} & b_{32} & \square \end{vmatrix} \quad \text{yields} \quad M_{23} = \begin{vmatrix} b_{11} & b_{12} \\ b_{31} & b_{32} \end{vmatrix} = b_{11}b_{32} - b_{31}b_{12}$$

Using the given definition it follows that

$$\begin{aligned} C_{23} &= (-1)^{2+3}(M_{23}) \\ C_{23} &= (-1)^{5}(b_{11}b_{32} - b_{31}b_{12}) \\ C_{23} &= b_{31}b_{12} - b_{11}b_{32}. \end{aligned}$$

Application of cofactors: computation of matrix inversion

Writing the transpose of the matrix of cofactors, known as an adjugate matrix, can also be an efficient way to calculate the inverse of small matrices, but this recursive method is inefficient for large matrices. To determine the inverse, we calculate a matrix of cofactors:

$$\mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|}\left(\mathbf{C}^{\mathrm{T}}\right)_{ij} = \frac{1}{|\mathbf{A}|}\left(\mathbf{C}_{ji}\right) = \frac{1}{|\mathbf{A}|}\begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{21} & \cdots & \mathbf{C}_{n1} \\ \mathbf{C}_{12} & \mathbf{C}_{22} & \cdots & \mathbf{C}_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{1n} & \mathbf{C}_{2n} & \cdots & \mathbf{C}_{nn} \end{pmatrix}$$

where $|A|$ is the determinant of A, C_{ij} is the matrix of cofactors, and C^T represents the matrix transpose.

Example: inversion of 2×2 matrices

The cofactor equation listed above yields the following result for 2×2 matrices. Inversion of these matrices can be done easily as follows:

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

This is possible because $1/(ad - bc)$ is the reciprocal of the determinant of the matrix in question, and the same strategy could be used for other matrix sizes.

Example: inversion of 3×3 matrices

A computationally efficient 3×3 matrix inversion is given by

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & k \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & K \end{bmatrix}^{T} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} A & D & G \\ B & E & H \\ C & F & K \end{bmatrix}$$

where the determinant of A can be computed by applying the rule of Sarrus as follows:
$\det(\mathbf{A}) = a(ek - fh) - b(kd - fg) + c(dh - eg)$.
If the determinant is non-zero, the matrix is invertible, with the elements of the above matrix on the right side given by

$$\begin{array}{lll} A = (ek - fh) & D = (ch - bk) & G = (bf - ce) \\ B = (fg - dk) & E = (ak - cg) & H = (cd - af) \\ C = (dh - eg) & F = (gb - ah) & K = (ae - bd). \end{array}$$