H2: Numerical linear algebra

2.1 systems of linear equations

2.1.1 introduction and notation

system of linear equations	a system of m equations with n variables can be written as: $\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_n \end{cases}$ But simpler as a matrix: $\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$
non-singular nxn-matrix	A matrix is non-singular if it satisfies one of the conditions: • $\mathbf{A}$ has an inverse $\mathbf{A}^{-1}$ such that $\mathbf{A}^{-1}\mathbf{A}=\mathbf{I}$ (the identity matrix) • $\det(\mathbf{A}) \neq 0$ • $\text{rank}(\mathbf{A}) = n$ (the **rank** of matrix is the maximum number of linearly independent rows or columns it contains) • for any vector $\mathbf{z} \neq 0$, $\mathbf{Az}$ also must be nonzero. > non-singular systems always have one unique solution: $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}.$

2.1.2 solving linear systems

2.1.2.1 strategy

solution strategy	Multiply both sides if Ax = b by any non-singular matrix M > gives us a new equation: $\mathbf{MAz} = \mathbf{Mb}$ with the same answer: $\mathbf{z} = (\mathbf{MA})^{-1}\mathbf{Mb} = \mathbf{A}^{-1}\mathbf{M}^{-1}\mathbf{Mb} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{x}$ > which matrix M makes the equation simpler??
triangular linear system	= system for which the matrix is triangular: - matrix L = lower triangular: $\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$ - matrix U = upper triangular: $\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}$ > there are strategies to get triangular matrices

2.1.2.2 elementary elimination matrices

Gauss transformation	= matrix M_{ka} eliminates entries in a vector from the *k*th position: $\mathbf{M_{ka}} = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & -m_{k+1} & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -m_n & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ a_{k+1} \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ Where: $m_i = \frac{a_i}{a_k} \quad i = k+1, \cdots, n$ > useful properties: • $\mathbf{M}_k = \mathbf{I} - \mathbf{m}_k\mathbf{e}_k^T$, where $\mathbf{m}_k = [0, \cdots, 0, m_{k+1}, \cdots, m_n]^T$ and $\mathbf{e}_k$ is the *k*th column of the identity matrix • $\mathbf{M}_k^{-1} = \mathbf{I} + \mathbf{m}_k\mathbf{e}_k^T$, which means that $\mathbf{M}_k^{-1}$, denoted as $\mathbf{L}_k$, is the same as $\mathbf{M}_k$ except that the signs of the multipliers are reversed.

2.1.2.3 Gaussian elimination and LU factorization	
LU factorization / Gaussian elimination	= process in which the matrix A is triangulated using Gaussian matrices M_k $stel\ dat\ A = \begin{bmatrix} a1 & a4 & a7 \\ a2 & a5 & a8 \\ a3 & a6 & a9 \end{bmatrix}$ $maak\ eerst\ M1 = \begin{bmatrix} 1 & 0 & 0 \\ m1 & 1 & 0 \\ m2 & 0 & 1 \end{bmatrix} met\ m1 = -\frac{a2}{a1}\ en\ m2 = -\frac{a3}{a1}$ $Bereken\ nu\ M1.A = \begin{bmatrix} a1 & b2 & b5 \\ 0 & b3 & b6 \\ 0 & b4 & b7 \end{bmatrix}$ $\mathrm{maak\ dan\ M2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \mathrm{n1} & 1 \end{bmatrix} \mathrm{met\ n1} = -\frac{\mathrm{b4}}{\mathrm{b3}}$ $\mathrm{bereken\ dan\ M1.M2.A} = \ldots$ $\ldots$ $\mathrm{dan\ is\ U} = \mathrm{M1.M2.} \ldots$
2.1.2.4 partial pivoting	
problems with LU factorization	1: the process breaks down if the leading diagonal entry is zero 2: in finite-precision arithmetic, we wish to limit the size of the multipliers > otherwise the previous rounding errors get amplified
Solution: partial pivoting	1: if a diagonal entry is zero, we interchange columns in the matrix 2: always choose the entry of the largest magnitude on or below the diagonal
2.1.2.5 Gauss-Jordan elimination	
Gauss-Jordan elimination	= variation of Gaussian elimination that eliminates both the entries above and below the diagonal Pos: - on parallel computers the workload stays the same > final solutions can be calculated all at once - can be used to calculate the inverse of a matrix neg: is 50% more computationally expensive
2.1.3 special types of linear systems	
special linear systems	= linear systems some special properties > easier way to solve • **Symmetric**: $\mathbf{A} = \mathbf{A}^T$, i.e. $a_{ij} = a_{ji}$ for all i, j • **Positive definite**: $\mathbf{x}^\mathbf{T}\mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \neq 0$ • **Banded**: $a_{ij} = 0$ for all $\|i - j\| > \beta$, with β the **bandwidth** of $\mathbf{A}$ • **Sparse**: most entries of $\mathbf{A}$ are zero
2.1.3.1 symmetric positive definite systems: Cholesky factor	
Cholesky factorization	= if matrix A is symmetric and positive definite > then: $U = L^T$, thus $A = LL^T$ We this property we can find for example in 2D: $\begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix} = \begin{bmatrix} l_{11}^2 & l_{11}l_{21} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 \end{bmatrix}$ thus: • $l_{11} = \sqrt{a_{11}}$ • $l_{21} = a_{12}/l_{11}$ • $l_{22} = \sqrt{a_{22} - l_{21}^2}$ with the properties: • The n square roots are all of positive numbers, so the algorithm is well-defined • Pivoting is not required • Only the lower triangle of $\mathbf{A}$ is accessed, and hence the strict upper triangular portion need not be stored • Only about $n^3/6$ multiplications and a similar number of additions are required. >> we can do this in more dimensions > python program on git

2.1.3.2 Computational complexity	
Computational cost	as seen in examples: • LU factorization of an $n \times n$ matrix takes about $n^3/3$ floating point operations (flops) • A complete matrix inversion takes about n^3 flops and thus is 3 times as expensive • Solving an LU-factorized system using forward and backward substitution takes about n^2 flops. For large systems, this is negligible compared to the factorization phase. • Cramer's rule (in which the system is solved using ratios of determinants) is astronomically expensive
2.1.4 sensitivity and conditioning	
2.1.4.1 vector norms	
vector norm	for an integer p>0: $$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} \|x_i\|^p\right)^{1/p}$$ ex: - 1-norm/Manhattan norm: $$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} \|x_i\|$$ - 2-norm/Euclidean norm: $$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} \|x_i\|^2}$$ > distance - ∞-norm: $$\|\mathbf{x}\|_\infty = \max_{1\le i\le n} \|x_i\|$$
properties of vector norms	In general, for any n-vector $\mathbf{x}$: $$\|\mathbf{x}\|_1 \ge \|\mathbf{x}\|_2 \ge \|\mathbf{x}\|_\infty$$ and $$\|\mathbf{x}\|_1 \le \sqrt{n}\|\mathbf{x}\|_2$$ $$\|\mathbf{x}\|_2 \le \sqrt{n}\|\mathbf{x}\|_\infty$$ $$\|\mathbf{x}\|_1 \le n\|\mathbf{x}\|_\infty$$ And for all p-norms, the following properties hold: • $\|\mathbf{x}\| > 0$ if $\mathbf{x} \neq \mathbf{0}$ • $\|\gamma\mathbf{x}\| = \|\gamma\| \cdot \|\mathbf{x}\|$ for any scalar γ • $\|\mathbf{x}+\mathbf{y}\| \le \|\mathbf{x}\| + \|\mathbf{y}\|$ (triangle inequality)
2.1.4.2 matrix norms	
matrix norm	for a mxn matrix A: $$\|\mathbf{A}\| = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}$$ = maximum stretching the matrix does to a vector: ex: • $\|\mathbf{A}\|_1$, which corresponds the maximum absolute *column* sum of the matrix: $$\|\mathbf{A}\|_1 = \max_j \sum_{i=1}^{m} \|a_{ij}\| \qquad (61)$$ • $\|\mathbf{A}\|_\infty$, which corresponds the maximum absolute *row* sum of the matrix: $$\|\mathbf{A}\|_\infty = \max_i \sum_{j=1}^{n} \|a_{ij}\| \qquad (62)$$

properties of matrix norms	• $\|\mathbf{A}\| > 0$ if $\mathbf{A} \neq \mathbf{0}$ • $\|\gamma\mathbf{A}\| = \|\gamma\| \cdot \|\mathbf{A}\|$, for any scalar γ • $\|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$ • $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|$ • $\|\mathbf{Ax}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\|$, for any vector $\mathbf{x}$
2.1.4.3 matrix condition number	
condition number	= a measure of how close a matrix is to being singular for a nonsingular square matrix A with respect to a given matrix norm > the condition number is defined by: $\operatorname{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$
2.1.4.4 error estimation	
condition number and error	For a non-singular system Ax=b with solution x > let x' be the solution to the perturbed system: $\mathbf{Ax}' = \mathbf{b} + \Delta\mathbf{b}$, with Δx = x'-x the difference in solutions This results in: $\mathbf{Ax}' = \mathbf{A}(\mathbf{x}+\Delta\mathbf{x}) = \mathbf{Ax} + \mathbf{A}\Delta\mathbf{x} = \mathbf{b} + \Delta\mathbf{b}$ Consequently, $\mathbf{A}\Delta\mathbf{x} = \Delta\mathbf{b}$, and hence $\Delta\mathbf{x} = \mathbf{A}^{-1}\Delta\mathbf{b}$. Now taking norms we find: • $\|\mathbf{b}\| = \|\mathbf{Ax}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\|$ or $\|\mathbf{x}\| \geq \frac{\|\mathbf{b}\|}{\|\mathbf{A}\|}$ • $\|\Delta\mathbf{x}\| = \|\mathbf{A}^{-1}\Delta\mathbf{b}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\Delta\mathbf{b}\|$ combining these gives us: $\boxed{\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \operatorname{cond}(\mathbf{A})\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}}$ > condition number acts as an amplification factor for the relative change in solution with respect to a relative change in the right hand sided vector
condition number and matrix error	For deviations E to the matrix A, such that: $(\mathbf{A}+\mathbf{E})\mathbf{x}' = \mathbf{b}$, we find: $\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}'\|} \leq \operatorname{cond}(\mathbf{A})\frac{\|\mathbf{E}\|}{\|\mathbf{A}\|}$
2.1.4.5 residual	
residual r	for an approximate solution x' of the system Ax = b, residual r is defined as: $\mathbf{r} = \mathbf{b} - \mathbf{Ax}'$ > r=0 if $\|\mathbf{x} - \mathbf{x}'\| = 0$. if we multiply Ax=b with a number, the solution remains the same > however, the residual will be multiplied by the same number
relative residual	= $\frac{\|\mathbf{r}\|}{(\|\mathbf{A}\| \cdot \|\mathbf{x}'\|)}$
relative residual and condition number	We can calculate: $\|\Delta\mathbf{x}\| = \|\mathbf{x}' - \mathbf{x}\| = \|\mathbf{A}^{-1}(\mathbf{Ax}' - \mathbf{b})\| = \|-\mathbf{A}^{-1}\mathbf{r}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{r}\|$ > dividing both by ||x'|| gives us: $\boxed{\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}'\|} \leq \operatorname{cond}(\mathbf{A})\frac{\|\mathbf{r}\|}{\|\mathbf{A}\| \cdot \|\mathbf{x}'\|}}$
software	To solve linear systems in python, see git.

2.2 Linear Least Squares

2.2.1 introduction

overdetermined problem

= problem Ax=b for which A is no longer square, but a mxn matrix with m>n
ie: there are more measurement data points than unknown variables

> there is noise on the measurements
> we want to model the data as closely as possible
> minimize the norm of the residual r = b-Ax

2.2.2 normal equations

objective function ϕ(x)

define:

$$\phi(\mathbf{x}) = \|\mathbf{r}\|_2^2 = \mathbf{r}^T\mathbf{r} = (\mathbf{b}-\mathbf{A}\mathbf{x})^T(\mathbf{b}-\mathbf{A}\mathbf{x}) = \mathbf{b}^T\mathbf{b} - \mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}$$

to minimize this function, we need to find the point that satisfies $\nabla\phi(\mathbf{x}) = \mathbf{0}$:

$$\mathbf{0} = \nabla\phi(\mathbf{x}) = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{b}$$

> Where we used the identity
>
> - $(\mathbf{B}\mathbf{A})^T = \mathbf{A}^T\mathbf{B}^T$
>
> and
>
> - $\nabla(\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}) = 2\mathbf{A}^T\mathbf{A}\mathbf{x}$
> - $\nabla(\mathbf{b}^T\mathbf{A}\mathbf{x}) = \mathbf{A}^T\mathbf{b}$
> - $\nabla(\mathbf{x}^T\mathbf{A}^T\mathbf{b}) = \mathbf{A}^T\mathbf{b}$
> - $\nabla(\mathbf{b}^T\mathbf{b}) = 0$

To minimize **x** for ϕ we need to satisfy the nxn symmetric linear system:

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$$

2.2.3 problem transformations

2.2.3.1 orthogonal transformations

orthogonal transformation

= preserves the Euclidean norm of any vector v

$$\|\mathbf{Q}\mathbf{v}\|_2^2 = (\mathbf{Q}\mathbf{v})^T\mathbf{Q}\mathbf{v} = \mathbf{v}^T\mathbf{Q}^T\mathbf{Q}\mathbf{v} = \mathbf{v}^{\mathbf{Tv}} = \|\mathbf{v}\|_2^2$$

A square real matrix Q is orthogonal if the columns are orthogonal
ie: $Q^TQ = I$

>> useful in numerical computations, since these matrices don't amplify errors
> BUT they are computationally more expensive

2.2.3.2 triangular least squares problems

triangular systems in least squares problems

are triangular systems a suitable target for our transformation?

consider:

$$\begin{bmatrix}\mathbf{R}\\\mathbf{O}\end{bmatrix}\mathbf{x} \cong \begin{bmatrix}\mathbf{c}_1\\\mathbf{c}_2\end{bmatrix} \tag{12}$$

with $\mathbf{R}$ an $n \times n$ upper triangular matrix and $\mathbf{O}$ a $(m-n) \times n$ null matrix.

the least squares residual is given by

$$\|\mathbf{r}\|_2^2 = \|\mathbf{c_1} - \mathbf{R}\mathbf{x}\|_2^2 + \|\mathbf{c_2}\|_2^2 \tag{13}$$

If we solve the triangular system $\mathbf{R}\mathbf{x} = \mathbf{c_1}$ (which can easily be achieved with back-substitution) we have found the least squares solution $\mathbf{x}$ and we can conclude that the minimum sum of squares is

$$\|\mathbf{r}\|_2^2 = \|\mathbf{c_2}\|_2^2 \tag{14}$$

2.2.3.3 QR-Factorization	
QR-factorization	transformation to a triangular form A: $$\mathbf{A} = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\\mathbf{O}\end{bmatrix}$$ (where $\mathbf{Q}$ is an $m \times m$ orthogonal matrix and $\mathbf{R}$ is an $n \times n$ upper triangular matrix. Then the residual equals $$\|\mathbf{r}\|_2^2 = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 = \|\mathbf{b} - \mathbf{Q}\begin{bmatrix}\mathbf{R}\\\mathbf{O}\end{bmatrix}\mathbf{x}\|_2^2 = \|\mathbf{Q^Tb} - \begin{bmatrix}\mathbf{R}\\\mathbf{O}\end{bmatrix}\mathbf{x}\|_2^2 = \|\mathbf{c_1} - \mathbf{Rx}\|_2^2 + \|\mathbf{c_2}\|_2^2$$ the solution to **Rx = c$_1$** gives the least squares solution **x** for the original problem
2.2.3.4 Householder transformations	
Householder matrix	= orthogonal transformation which annihilates targeted components of a vector $$\mathbf{H} = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}} \quad (18)$$ with $\mathbf{v}$ a nonzero vector. It can be shown that $\mathbf{H} = \mathbf{H}^{-1} = \mathbf{H}^{\mathbf{T}}$, which means that $\mathbf{H}$ is orthogonal and symmetric.
annihilating all but the first component of a vector	We want a v such that it annihilates all the components of a vector a except the first: $$\mathbf{Ha} = \begin{bmatrix}\alpha\\0\\\vdots\\0\end{bmatrix} = \alpha\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} = \alpha\mathbf{e}_1$$ Using the definition of $\mathbf{H}$ we find $$\alpha\mathbf{e}_1 = \mathbf{Ha} = \left(\mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}\right)\mathbf{a} = \mathbf{a} - 2\mathbf{v}\frac{\mathbf{v}^T\mathbf{a}}{\mathbf{v}^T\mathbf{v}}$$ and thus $$\mathbf{v} = (\mathbf{a} - \alpha\mathbf{e}_1)\frac{\mathbf{v}^T\mathbf{v}}{2\mathbf{v}^T\mathbf{a}}$$ The scalar factor is irrelevant as it cancels out in the expression for $\mathbf{H}$, so we find $$\mathbf{v} = (\mathbf{a} - \alpha\mathbf{e}_1)$$ To preserve the norm and avoid cancellation $$\alpha = -\mathbf{sign}(a_1)\|\mathbf{a}\|_2$$
annihilating all but the first k component of a vector	If we split up a given m-vector $\mathbf{a}$ as $$\mathbf{a} = \begin{bmatrix}\mathbf{a}_1\\\mathbf{a}_2\end{bmatrix} \quad (24)$$ where $\mathbf{a}_1$ is a $(k-1)$-vector with $1 \leq k < m$. If we then take the householder vector to be $$\mathbf{v} = \begin{bmatrix}\mathbf{0}\\\mathbf{a}_2\end{bmatrix} - \alpha\mathbf{e}_k \quad (25)$$ where $\alpha = -\mathbf{sign}(a_k)\|\mathbf{a}_2\|_2$, then the resulting Householder transformation annihilates the last $m-k$ components of $\mathbf{a}$.

QR factorization using householder transformations	By sequentially performing this transformation for all the columns from left to right of a matrix $\mathbf{A}$, we can get the desired upper triangular matrix: $$\mathbf{H}_n \dots \mathbf{H}_1 \mathbf{A} = \begin{bmatrix} \mathbf{R} \\ \mathbf{O} \end{bmatrix} \qquad (26)$$ The product of orthogonal householder transformations is itself an orthogonal matrix, which we define as $$\mathbf{Q}^T = \mathbf{H}_n \dots \mathbf{H}_1 \qquad \text{or, equivalently} \qquad \mathbf{Q} = \mathbf{H}_n^T \dots \mathbf{H}_1^T \qquad (27)$$ Such that $$\mathbf{A} = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ \mathbf{O} \end{bmatrix} \qquad (28)$$ which shows that we have indeed calculated the QR factorization of $\mathbf{A}$. To solve the least squares system $\mathbf{Ax} \cong \mathbf{b}$, we solve the equivalent system $$\begin{bmatrix} \mathbf{R} \\ \mathbf{O} \end{bmatrix} \mathbf{x} \cong \mathbf{Q}^T \mathbf{b} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} \qquad (29)$$
2.2.4 rank deficiency	
rank deficiency	So far we assumed rank(A) = n > if rank(A) ≠ n, we can still perform QR factorization of A > However: the upper triangular matrix will be singulae
2.2.5 Singular value decomposition	
single value decomposition	= strategy where we reduce to a diagonal linear least square system > for a mxn matrix A this has the form: $$\mathbf{A} = \mathbf{U\Sigma V^T} \qquad (34)$$ where $\mathbf{U}$ is an $m \times m$ orthogonal matrix, $\mathbf{V}$ is an $n \times n$ orthogonal matrix, and $\mathbf{\Sigma}$ is an $m \times n$ diagonal matrix, with $$\sigma_{ij} = \begin{cases} 0, & \text{for } i \neq j \\ \sigma_i \geq 0, & \text{for } i = j \end{cases} \qquad (35)$$ The diagonal entries σ_i are called the **singular values** of $\mathbf{A}$ and are usually ordered so that $\sigma_{i-1} \geq \sigma_i, i = 2, \dots, \min\{m, n\}$, i.e. from largest value (upper left) to smallest value (bottom right). The columns $\mathbf{u_i}$ of $\mathbf{U}$ and $\mathbf{v_i}$ of $\mathbf{V}$ are the corresponding left and right **singular vectors**.
2.2.5.1 other applications of SVD	
Euclidean matrix norm	As stated before in the linear systems notebook, the matrix norm corresponding to the Euclidean vector norm is equal to the largest singular value of the matrix, $$\|\mathbf{A}\|_2 = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|_2}{\|\mathbf{x}\|_2} = \sigma_{\max} \qquad (40)$$
Euclidean condition number	for a matrix A this is given by: $$\text{cond}_2(\mathbf{A}) = \frac{\sigma_{\max}}{\sigma_{\min}} \qquad (41)$$ Note that, just as before, we find $\text{cond}_2(\mathbf{A}) = \infty$ for singular matrices, because there, $\sigma_{\min} = 0$.
rank determination	the rank of a matrix is equal to the number of nonzero singular values it has

pseudoinverse	we can define an inverse fir non-square matrices as the pseudoinverse: • Define the pseudoinverse of a scalar σ as $1/\sigma$ (or 0 if $\sigma = 0$) • Define the pseudoinverse of a (possibly rectangular) diagonal matrix by transposing the matrix and taking the scalar pseudo-inverse of each entry. now: The **pseudoinverse** of a general matrix $\mathbf{A}$ is given by $$\mathbf{A}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^\mathrm{T} \quad (43)$$ • If the matrix $\mathbf{A}$ is square and nonsingular this definition agrees with $\mathbf{A}^{-1}$. • In all cases, the solution to a least squares problem $\mathbf{A}\mathbf{x} \cong \mathbf{b}$ is given by $\mathbf{A}^+\mathbf{b}$. An other (computationally less good) way to find the pseudo-inverse can be obtained via the normal equations $$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} \quad (44)$$ we see that $$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \quad (45)$$ is a solution of the least squares problem $\mathbf{A}\mathbf{x} \cong \mathbf{b}$. Consequently, the pseudoinverse $\mathbf{A}^+$ is also given by $$\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \quad (46)$$
2.2.6 sensitivity and condition number	
calculating condition number	Generalizing the definition of a condition number to an $m \times n$ matrix with $\mathrm{rank}(\mathbf{a}) = n$, we define $$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^+\|_2$$ By convention, $\mathrm{cond}(\mathbf{A}) = \infty$ if $\mathrm{rank}(\mathbf{A}) < n$ Let's now also generalize the expression, $$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}'\|} \leq \mathrm{cond}(\mathbf{A})\frac{\|\mathbf{r}\|}{\|\mathbf{A}\| \cdot \|\mathbf{x}'\|}$$
2.2.8 which method to use	
which method to use	- normal equations: easiest method to implement > computationally expensive > error proportional to $[\mathrm{cond}(A)]^2$ - Householder method: most efficient and accurate > for square systems it requires the same amount of work > for strongly overdetermined it's only half as efficient - SVD: most expensive, but most robust and reliable