

Course Contents

- Standard Matrix Decompositions: LU, QR, SVD, FFT, Eigenvalue,...
- Basic Linear Algebra Operations: Projection, Rotation,...
- Computing and using the Standard Decompositions.
- Non-Linear Equations and Least Squares. The Newton and Gauss–Newton methods.
- Applications: Model fitting, Roots of Polynomials, Text models and Search Engines, Image processing,...

Examination

- Written Exam (4 hp)
- Computer Exercises (2 hp)

Lecturer

- Fredrik Berntsson (fredrik.berntsson@liu.se).

TANA15/Lecture 1 - Contents

Basic Matrix Operations

- Matrix–Matrix multiplication. Operation counts
- Basic Linear Algebra Subroutines (BLAS,ATLAS)

Linear Spaces and Mappings

- Range and Null spaces. Rank. The Inverse.
- Scalar Products, Vector and Matrix Norms. The Transpose.

Theory

- Define a **good** set of standard linear algebra operations and matrix decompositions.
- Show how application problems can be solved by using standard operations.
- Investigate stability properties, error estimates, etc.

Software

- Write efficient and reliable subroutines for computing decompositions.
- Modify existing software to take advantage of modern computer hardware.

Example: Matrix–Matrix multiply

Compute $C = AB$ by

$$c_{ij} = \sum_{k=1}^n a_{ik}b_{kj}.$$

In Matlab

```
C=zeros(n,n);
for i=1:n
    for j=1:n
        for k=1:n
            C(i,j)=C(i,j)+A(i,k)*B(k,j);
        end
    end
end
```

Requires n^3 multiply/additions. Is this the best way?

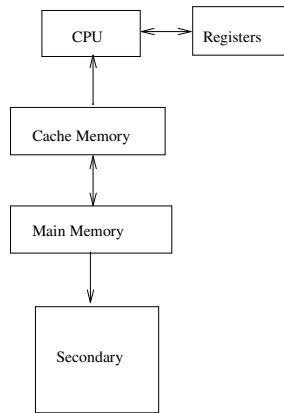
Memory Organization

Data storage and access

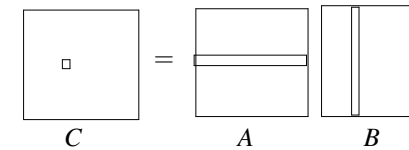
- CPU can **only** access Registers and Cache.
- Data is stored in *blocks*.
- A block can be moved between main and cache memory.

Memory Performance

- CPU and Registers are fast. Low storage capacity.
- Main memory is slow but has high storage capacity.



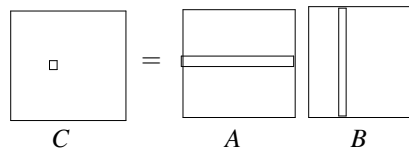
The structure of matrix-matrix multiply



Assumptions Matrices stored by column. One column/memory block. Three columns fit in the Cache memory.

Then the column $B(:, j)$ is stored as one memory block and the elements in the row $A(i, :)$ are stored in different blocks.

Conclusion Computing $A(i, :)^T B(:, j)$ require one main memory access/multiply!



Alternative Store A by rows. Both $A(i, :)$ and $B(:, j)$ fit in Cache. Computing $A(i, :)^T B(:, j)$ requires two Main memory access calls!

Conclusion Computing $C = AB$ requires n^3 multiply/additions and $2n^2$ main memory access calls.

Doesn't store A and B the same way!

$$\begin{pmatrix} C_{11} & \dots & C_{1p} \\ \vdots & & \vdots \\ C_{p1} & & C_{pp} \end{pmatrix} = \begin{pmatrix} A_{11} & \dots & A_{1p} \\ \vdots & & \vdots \\ A_{p1} & & A_{pp} \end{pmatrix} \begin{pmatrix} B_{11} & \dots & B_{1p} \\ \vdots & & \vdots \\ B_{p1} & & B_{pp} \end{pmatrix}$$

Alternative Block storage. Blocks are of size $\sqrt{n} \times \sqrt{n}$. Three blocks fit into cache.

Keep C_{ij} in Cache. Updating $C_{ij} = C_{ij} + A_{ik}B_{kj}$ needs two main memory calls and $(\sqrt{n})^3$ multiply/additions.

Conclusion Still need n^3 multiply/additions. But only $2(\sqrt{n})^3 = 2n^{1.5}$ main memory access calls.

Matrix-Matrix multiply

Compute $C = AB$ by

$$c_{ij} = \sum_{k=1}^n a_{ik}b_{kj}.$$

Remark This is not a *definition*. It is one possible algorithm for computing the matrix C representing the composite mapping $A \circ B$.

Question The algorithm requires n^3 multiplications and $n^2(n-1)$ additions. Is it possible to do better?

Lemma Computing the product $C = AB$ requires *at least* $\mathcal{O}(n^2)$ arithmetic operations.

This is the only result that exists!

Strassen's method requires $\mathcal{O}(n^{2.807})$ operations. The currently best algorithm requires $\mathcal{O}(n^{2.3727})$. By Virginia Vassilevska Williams.

Remark Very large matrices are often *sparse*, i.e. most elements a_{ij} are zero, and other algorithms are much more efficient.

Strassen's Matrix-Matrix multiply

Regular matrix-matrix multiply is

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} \end{pmatrix}.$$

This requires 8 multiplications (and 4 additions). An equivalent formula is

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} p_1+p_4-p_5+p_7 & p_3+p_5 \\ p_2+p_4 & p_1+p_3-p_2+p_6 \end{pmatrix},$$

where

$$\begin{aligned} p_1 &= (a_{11} + a_{22})(b_{11} + b_{22}), & p_2 &= (a_{21} + a_{22})b_{11}, & p_3 &= a_{11}(b_{12} - b_{22}), \\ p_4 &= a_{22}(b_{21} - b_{11}), & p_5 &= (a_{11} + a_{12})b_{22}, & p_6 &= (a_{21} - a_{11})(b_{11} + b_{12}), \\ & & p_7 &= (a_{12} - a_{22})(b_{21} + b_{22}). \end{aligned}$$

Only 7 multiplications (and 18 additions). Volker Strassen, 1969.

Operation counts

Lemma A matrix-matrix multiply $C = AB$ requires $\mathcal{O}(n^3)$ operations.

Lemma A matrix-vector multiply $y = Ax$ requires $\mathcal{O}(n^2)$ operations.

Example Suppose $A, B \in \mathbb{R}^{n \times n}$. How much computational work is needed to evaluate the product

$$y = ABx.$$

Lemma Computing an *outer product* $A = uv^T$ requires $\mathcal{O}(n^2)$ operations.

Example How should we compute the matrix-vector product

$$y = Ax, \quad \text{where } A = uv^T, \quad u, v \in \mathbb{R}^n,$$

and how many arithmetic operations and memory slots are needed?

Remark Estimating the amount of work is important. The difference between $\mathcal{O}(n^2)$ and $\mathcal{O}(n^4)$ is huge for large n .

Standard set of basic linear algebra operations

- Level 1: Scalar–Vector.
- Level 2: Matrix–Vector.
- Level 3: Matrix–Matrix.

Software (C/C++, Fortran, Matlab)

- Efficient implementations available for most computers.
- Takes advantage of complex memory systems.
- Reference implementation available on www.netlib.org.
- Level 3 operations gains the most from code optimization!

Example A SAXPY call computes $z = \alpha x + y$ where α is a scalar and x, y are vectors. The S means single precision or 32 bit floating point numbers. A DGEMM call computes $C := \alpha AB + \beta C$, in 64 bit arithmetic.

Automatically Tuned Linear Algebra Subroutines (ATLAS)

- Implements most of the routines from BLAS and much more.
- Available from

<http://math-atlas.sourceforge.net/>

or package managers in Linux. Try

```
>> yum info atlas
>> man dgemm
```

in the computer laboratory.

- Download the source and compile. Automatically detects cache size, memory read/write speed, etc, and produce close to the best available code.

Basic concepts

A matrix $A \in \mathbb{R}^{m \times n}$ represents a *linear mapping* from \mathbb{R}^n to \mathbb{R}^m .

The *range* of the matrix A is the linear subspace

$$\text{Range}(A) = \{y \in \mathbb{R}^m \text{ such that } y = Ax \text{ for some } x \in \mathbb{R}^n\}.$$

Remark Similarly the *domain* is the set $x \in \mathbb{R}^n$ such that $y = Ax$ is defined. This is not as often used since typically $\text{Domain}(A) = \mathbb{R}^n$.

Definition The *rank* of a matrix is

$$\text{Rank}(A) = \dim(\text{Range}(A))$$

Remark If $A \in \mathbb{R}^{n \times m}$ then $\text{Rank}(A) \leq \min(n, m)$.

Lemma Let $A \in \mathbb{R}^{n \times n}$. If $\text{Rank}(A) = n$ then there exists an *inverse* A^{-1} such that $x = A^{-1}y$ for every x, y such that $y = Ax$.

Example Prove that $(AB)^{-1} = B^{-1}A^{-1}$.

Norms and Scalar products

Definition Let $x \in \mathbb{R}^n$. The *norm* $\|x\|$ is a measure of the *size* of x .

Example The most commonly used norms are

$$\|x\|_2 = \left(\sum_{i=1}^n x_i^2 \right)^{\frac{1}{2}} \quad \text{and} \quad \|x\|_\infty = \max_{1 \leq i \leq n} |x_i|.$$

They satisfy the relation

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n} \|x\|_\infty.$$

Remark There are many different norms that are used.

Example Consider a linear system $Ax = b$. Existence of a solution?

Definition Let $A \in \mathbb{R}^{n \times m}$. The *null space* is

$$\text{Null}(A) = \{x \in \mathbb{R}^n \text{ such that } Ax = 0\}.$$

Definition The identity mapping I is defined by $Ix = x$ for every $x \in \mathbb{R}^n$.

Remark If the inverse of A exists then $A^{-1}A = I$.

Definition The *Scalar product* (x, y) measures the angle between x and y . If $(x, y) = 0$ then x and y are *orthogonal*.

Example The space \mathbb{R}^n is a Hilbert space with the scalar product $(x, y) = x^T y$. We have $\|x\|_2^2 = (x, x)$.

Lemma The *Cauchy-Schwarz inequality* $|(x, y)| \leq \|x\| \|y\|$ holds.

Definition Let $\| \cdot \|$ be a vector norm. A matrix norm is

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}.$$

Remark The matrix norm is *induced* from a vector norm.

Lemma Suppose A is a matrix. Then

$$\|A\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^n |A_{ij}|.$$

Lemma Suppose A and B are matrices and $\| \cdot \|$ is a matrix norm *induced* from a vector norm. Then the *submultiplicative property* $\|AB\| \leq \|A\|\|B\|$ holds.

Example Prove that $\|A\|\|A^{-1}\| \geq 1$ for *any* matrix norm induced by a vector norm.

Matlab

In order to compute the *rank* or the *nullspace* of a matrix we use

```
>> k = rank( A );  
>> V = null( A );
```

The columns of V are an orthogonal basis for $\text{Null}(A)$.

In order to compute norms there is a function

```
>> norm( x , 2 )  
>> norm( A , 'fro' )
```

that computes most different norms. The inverse is computed using

```
>> inv(A)
```

Definition The *Frobenius* norm of a matrix A is

$$\|A\|_F = \left(\sum_{i=1}^n \sum_{j=1}^m |a_{ij}|^2 \right)^{1/2}.$$

Remark The norm $\| \cdot \|_F$ is *not* induced by a vector norm.

The Transpose

Definition The *transpose* of a matrix $A \in \mathbb{R}^{n \times m}$ is a matrix $A^T \in \mathbb{R}^{m \times n}$ defined by $(A^T)_{ij} = (A)_{ji}$.

Lemma $(AB)^T = B^T A^T$

Proof Look at a component of the matrix $(AB)^T$

$$\begin{aligned} ((AB)^T)_{ij} &= (AB)_{ji} = \sum_{k=1}^p a_{jk} b_{ki} = \sum_{k=1}^p (A^T)_{kj} (B^T)_{ik} = \\ &= \sum_{k=1}^p (B^T)_{ik} (A^T)_{kj} = (B^T A^T)_{ij}. \end{aligned}$$

Definition The *transpose* of A is the matrix A^T that satisfies $(Ax, y) = (x, A^T y)$ for every pair of vectors x, y .

Lemma If A maps \mathbb{R}^n into \mathbb{R}^m then $(A)_{ij} = (A^T)_{ji}$.

Proof Use the standard basis $\{e_i\}$ and the scalar product $(x, y) = x^T y$.

Corollary $(AB)^T = B^T A^T$.

Remark Compare with the *adjoint* from functional analysis. The proof gives more insight!