TANA09 Lecture 4

Application

- Image Inpaiting as a linear system of equations.

Basic Matrix Operations.

- Arithmetic complexity.
- BLAS and ATLAS.

Linear Systems of Equations

- Gaussian elimination. The LU decomposition.
- Pivoting.
- Vector and matrix norms.

Interpolation in images

Suppose we have an image I with an irritating detail. Can we solve the problem?

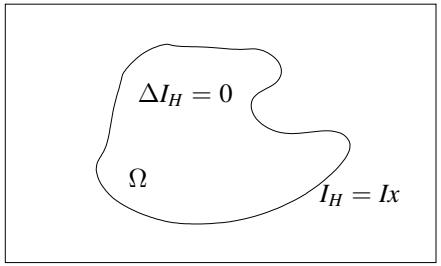
The part of the image we want to replace is detailed. This is an *interpolation problem*. Estimate suitable color values for the pixles we want to replace.

Definition An *image* $I \in \mathbb{R}^{n\times m}$ is a matrix where each *pixel* I_{ij} has a value that represents a *color*.

Definition The interpolation *mask* $M \in \mathbb{R}^{n\times m}$ is $M_{ij} = 1$ is the corresponding *pixel* should be replaced and $M_{ij} = 0$ otherwise.

Color images are treated as three seperate images.

Harmonic interpolation



Definition Let $I(x,y)$ be an image which is considered as unknown in the subdomain Ω. Harmonic interpolation computes $I_H(x,y)$ by solving

$$\begin{cases} \Delta I_H = 0, & \text{in } \Omega, \\ I_H(x,y) = I(x,y), & \text{on } \partial\Omega. \end{cases}$$

Numerical Method Approximate derivatives with finite differences.

$$\frac{\partial^2 I_H}{\partial x^2}(x_i, y_j) \approx \frac{1}{h^2}(I_H(x_{i+1}, y_j) - 2I_H(x_i, y_j) + I_H(x_{i-1}, y_j)).$$

Formulate a linear system of equations $Ax = b$ where each *unknown pixel* (i,j) gives an equation

$$I_H(i+1,j) + I_H(i-1,j) + I_H(i,j+1) + I_H(i,j-1) - 4I_H(i,j) = 0,$$

and each *known pixel* gives

$$I_H(i,j) = I(i,j).$$

The solution vector x contains the unknowns

$$x = (I_H(1,1)\,,\ I_H(1,2)\,,\ I_H(1,3)\ \ldots,\ I_H(m,n-1)\,,\ I_H(m,n))^T.$$

The result after harmoinic inpainting and the original image. More complicated differential equations needed for better results.

The image is of size 384×256 pixlar. The dimension of A is 98304×98304.

Solution of linear systems of equations is one of the most commonly occuring tasks for computers.

Basic matrix operations

Example Let A, B be matrices and x, y, and z be vectors. We want to compute

$$>> z = x + A^{-1}(By + A^{-1}z).$$

How do we implement the formula. How to estimate the amount of work required.

Example Let x and y be vectors. The scalar product $x^T y$ is computed as

$$x^T y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

If n is large and the sign of $x_i y_i$ changes for each i is there a risk for catastrophic cancellation?

Answer Yes! It is difficult to understand why it is possible to solve a large linear system on a computer using floating point arithmetic. Some algorithms can be proved to work.

Computational complexity

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.

Basic Linear Algebra Subroutines (BLAS)

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 precison 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.

Linear Systems of Equations

A *linear system of equations* can be written as

$$Ax = b,$$

where

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}, \qquad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}.$$

Lemma A *linear system of equations* $Ax = b$ has a solution if $b \in \text{Range}(A)$.

Remark Often the number of unknowns is very large. Important to be able to solve efficiently and accurately.

Existance and uniqueness

Definition If the matrix A has linearly independent columns then A is said to be *non-singular*.

Lemma If A is *non-singular* then $Ax = b$ has a unique solution and A^{-1} exists.

Question How to check this?

Remark Suppose $A \in \mathbb{R}^{m\times n}$, $m > n$, then A^{-1} does not exist. If $b \in \text{Range}(A)$ a solution to $Ax = b$ exists. If $\text{null}(A) = \{0\}$ then the solution is unique.

Lemma If $\det(A) \neq 0$ then A is *non-singular* and A^{-1} exists

Example If

$$A = \begin{pmatrix} 10^{-3} & 0 & 0 \\ 0 & 10^{-3} & 0 \\ 0 & 0 & 10^{-3} \end{pmatrix},$$

then $det(A) = 10^{-9} \approx 0$.

A small value for $\det(A)$ does not mean A is close to singular.

Solving Linear Systems of Equations

Solve $Ax = b$ where

$$\begin{pmatrix} 1 & 2 & 2 \\ 4 & 4 & 2 \\ 4 & 6 & 4 \end{pmatrix} x = \begin{pmatrix} 3 \\ 6 \\ 10 \end{pmatrix}$$

Method Reduce A to upper triangular form using row operations and partial pivoting.

Following the pivoting strategy we exchange rows one and two:

$$\left(\begin{array}{ccc|c} 1 & 2 & 2 & 3 \\ 4 & 4 & 2 & 6 \\ 4 & 6 & 4 & 10 \end{array}\right) \sim \left(\begin{array}{ccc|c} 4 & 4 & 2 & 6 \\ 1 & 2 & 2 & 3 \\ 4 & 6 & 4 & 10 \end{array}\right)$$

Use multipliers $m_{21} = 0.25$ and $m_{31} = 1$ to eliminate a_{21} and a_{31}. Pivot again by exchanging rows 2 and 3.

$$\left(\begin{array}{ccc|c} 4 & 4 & 2 & 6 \\ 0 & 1 & 1.5 & 1.5 \\ 0 & 2 & 2 & 4 \end{array}\right) \sim \left(\begin{array}{ccc|c} 4 & 4 & 2 & 6 \\ 0 & 2 & 2 & 4 \\ 0 & 1 & 1.5 & 1.5 \end{array}\right)$$

Now use a multiplier $m_{32} = 0.5$ to eliminate a_{32}. Then solve the triangular system using backwards substitution.

$$\left(\begin{array}{ccc|c} 4 & 4 & 2 & 6 \\ 0 & 2 & 2 & 4 \\ 0 & 0 & 0.5 & -0.5 \end{array}\right) \implies x = \begin{pmatrix} -1 \\ 3 \\ -1 \end{pmatrix}$$

This is called *Gaussian Elimination*!

Definition Row operations use a *Gauss transformation matrix M* and row exchanges use *Permutation matrix P*.

Example A Gauss-transformation has the structure

$$M\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 - m_{21}x_1 \\ x_3 - m_{31}x_1 \end{pmatrix} \quad \Longrightarrow \quad M = \begin{pmatrix} 1 & 0 & 0 \\ -m_{21} & 1 & 0 \\ -m_{31} & 0 & 1 \end{pmatrix}.$$

and an example of a permutation matrix is

$$P_{23}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_3 \\ x_2 \end{pmatrix} \quad \Longrightarrow \quad P_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Both P^{-1} and M^{-1} exists. What is M^{-1}?

Repeat the steps taken to reduce A to upper triangular form using Gauss transformations and permutation matrices.

First exchange rows 1 and 2

$$P_{12}A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 2 & 2 \\ 4 & 4 & 2 \\ 4 & 6 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 4 & 2 \\ 1 & 2 & 2 \\ 4 & 6 & 4 \end{pmatrix}$$

Second use a Gauss transformation M_1 to eliminate a_{21} and a_{31}.

$$M_1(P_{12}A) = \begin{pmatrix} 1 & 0 & 0 \\ -0.25 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}\begin{pmatrix} 4 & 4 & 2 \\ 1 & 2 & 2 \\ 4 & 6 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 4 & 2 \\ 0 & 1 & 1.5 \\ 0 & 2 & 2 \end{pmatrix}$$

Continue and exchange rows 2 and 3

$$P_{23}(M_1P_{12}A) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 4 & 4 & 2 \\ 0 & 1 & 1.5 \\ 0 & 2 & 2 \end{pmatrix} = \begin{pmatrix} 4 & 4 & 2 \\ 0 & 2 & 2 \\ 0 & 1 & 1.5 \end{pmatrix}$$

Lastly use a Gauss transformation M_2 to eliminate a_{32}

$$M_2(P_{23}M_1P_{12}A) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.5 & 1 \end{pmatrix}\begin{pmatrix} 4 & 4 & 2 \\ 0 & 2 & 2 \\ 0 & 1 & 1.5 \end{pmatrix} = \begin{pmatrix} 4 & 4 & 2 \\ 0 & 2 & 2 \\ 0 & 0 & 0.5 \end{pmatrix} = U$$

We now have $M_2P_{23}M_1P_{12}A = U$ or $P_{12}A = M_1^{-1}P_{23}^TM_2^{-1}U$.

Multiply both sides by P_{23} to obtain

$$P_{23}P_{12}A = P_{23}M_1^{-1}P_{23}^TM_2^{-1}U,$$

or $PA = LU$ where,

$$P = P_{23}P_{12} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 4 & 4 & 2 \\ 0 & 2 & 2 \\ 0 & 0 & 0.5 \end{pmatrix},$$

$$L = P_{23}M_1^{-1}P_{23}^TM_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0.25 & -0.5 & 1 \end{pmatrix}.$$

This is called the *LU decomposition*!

The LU decomposition

Theorem Every non-singular matrix A can be written $PA = LU$, where P is a permutation matrix, L and U are triangular, non–singular, and $|L| \leq 1$.

Lemma Computing the decomposition $PA = LU$ requires $2n^3/3$ multiply/additions to compute.

Example Use the LU decomposition for solving a linear system $Ax = b$. In Matlab

```
>> [L,U,P]=lu(A);
>> y=L\(P*b); x = U\y;
```

Remark Most efficient way to check if a matrix A is non-singular.

The Cholesky decomposition

Definition A matrix $A \in \mathbb{R}^{n\times n}$ is *positive definite* if $x^T Ax > 0$, for every $x \neq 0$.

Proposition If A is symmetric and positive definite then pivoting is not needed and $A = R^T R$ is the *Cholesky decomposition*.

Remark Exactly half the work and memory compared to regular LU-decomposition. In Matlab we use `chol`.

Pivotering

Definition The *partial pivoting* strategy is that the largest element in each column is used for the row operations.

Remark This means that all multipliers $|m_{ij}| \leq 1$. Problems with *cancellation* are minimized.

Example Solve $Ax = b$ where

$$A = \begin{pmatrix} 10^{-5} & 2 \\ 1 & 4 \end{pmatrix}, \text{ and } b = \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

Do the computations in the floating point system $(10, 3, -9, 9)$.

Begin with

$$\left(\begin{array}{cc|c} 10^{-5} & 2 & 2 \\ 1 & 4 & 3 \end{array}\right)$$

Without pivoting we get

$$m_{21} = \text{fl}[1/10^{-5}] = 10^5,$$

and

$$a'_{22} = \text{fl}[4 - 10^5 \cdot 2] = \text{fl}[-1.99996 \cdot 10^5] = -2.000 \cdot 10^5.$$

The triangular system is

$$\left(\begin{array}{cc|c} 10^{-5} & 2 & 2 \\ 0 & -2 \cdot 10^5 & -2 \cdot 10^5 \end{array}\right)$$

Backwards substitution leads to

$$x = (0\ 1)^T.$$

Again solve $Ax = b$ but *pivot* first

$$\left(\begin{array}{cc|c} 10^{-5} & 2 & 2 \\ 1 & 4 & 3 \end{array}\right) \sim \left(\begin{array}{cc|c} 1 & 4 & 3 \\ 10^{-5} & 2 & 2 \end{array}\right)$$

After pivoting we instead get

$$m_{21} = \text{fl}[10^{-5}/1] = 10^{-5},$$

and

$$a'_{22} = \text{fl}[2 - 10^{-5} \cdot 4] = \text{fl}[1.99996] = 2.000.$$

The triangular system is

$$\left(\begin{array}{cc|c} 1 & 4 & 3 \\ 0 & 2 & 2 \end{array}\right)$$

with the solution

$$x = (-1 \;\; 1)^T.$$

The exact solution is

$$x = (-1.0000200\ldots \;\; 1.00000500\ldots)^T.$$

Triangular equation systems

Definition If $\ell_{ij} = 0$, for $i < j$, then L is *lower triangular*

Lemma A lower triangular system $Lx = b$ can be solved using *forward substitution*. The complexity is $\mathcal{O}(n^2)$.

The algorithm can be written

$$x_i = \left(b_i - \sum_{j=1}^{i-1} \ell_{ij} x_j \right) / \ell_{ii}, \qquad i = 1, 2, \ldots, n.$$

Upper triangular systems are solved using *backwards substitution*.

Example Let

$$A = \begin{pmatrix} 5 & 1 & 1 & 1 & 1 \\ 1 & 5 & 1 & 1 & 1 \\ 1 & 1 & 5 & 1 & 1 \\ 1 & 1 & 1 & 5 & 1 \\ 1 & 1 & 1 & 1 & 5 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & 1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & 1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

Suppose we want to solve the system

$$\begin{cases} Ax = \mathrm{e}^{-y}, \\ By = x^2. \end{cases}$$

How to do this?

Remark Similar systems describe the mixing of two fluids. The concentration of fluid 1 in location k in a container is x_k.

Solution Use *fixed point iteration*. In each step we solve both systems. Compute LU decompositions in advance.

In Matlab:

```
x=zeros(n,1);y=x;
[La,Ua,Pa]=lu(A);[Lb,Ub,Pb]=lu(B);
for k=1:10000,
    x=Ua\(La\(Pa*exp(-y)));
    y2=Ub\(Lb\(Pb*x.^2));
    if max(abs(y-y2))<0.5e-8,break,end
    y=y2;
end;
```

After $k = 8$ we have
$y = (\,0.0285\,,\; 0.0451\,,\; 0.0505\,,\; 0.0451\,,\; 0.0285\,)^T$.

Two LU decompositions and 16 triangular systems.

Vector and matrix norms

Observation For error estimates we need a way to measure the *size* of a vector or a matrix.

Definition Let x and y be two vectors in $\mathbb{R}^n$. A *vector norm* satisfies

(1) $\|x\| \geq 0$, for all x.
(2) $\|x\| = 0$, if and only if $x = 0$.
(3) $\|\alpha x\| = |\alpha|\|x\|$ for all scalars α.
(4) $\|x + y\| \leq \|x\| + \|y\|$.

The difference between two vectors is measured by $\|x - y\|$.

Example The most common norms are

- 1-norm $\|x\|_1 = \sum_{i=1}^{n} |x_i|$.
- Euclidean norm $\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}}$.
- Maximum norm $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$.

Lemma The relation $\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1$ holds.

In Matlab there is a function `norm`. Example: `norm(x,inf)`.

Absolute and relative errors

Definition The *absolute error* in the vector $\bar{x}$ is denoted by

$$\|\delta x\| = \|\bar{x} - x\|.$$

The *relative error* is

$$\frac{\|\delta x\|}{\|x\|} = \frac{\|\bar{x} - x\|}{\|x\|}.$$

Example Let $x = (\sqrt{3}\,,\ 1)^T$ and $\bar{x} = (1.73\,,\ 1)^T$. What is the absolute and relative errors?

Exemple Suppose we compute the elements in the vector $x \in \mathbb{R}^n$ with a *relative error* $|\Delta x_i|/|x_i| \leq C\mu$. How large is the relative error in $\bar{x}$ measured in $\|\cdot\|_2$ or $\|\cdot\|_\infty$?