Linear Algebra
Tim Cootes

Linear Equations
• The simplest linear equation is
  \[ ax = b \]
• One equation, with one unknown, \( x \)
• There are two cases to consider
  1) \( a \neq 0 \)
  2) \( a = 0 \)

Simple Linear Equations
\[ ax = b \]
1) \( a \neq 0 \) There is exactly one solution: \( x = b/a \)
2) \( a = 0 \) If \( b \neq 0 \) no value of \( x \) satisfies \( ax = b \)
   If \( b = 0 \) every value of \( x \) satisfies \( ax = b \)
• In more complicated case (\( m \) equations in \( n \) unknowns), there are either
  – No solutions
  – Exactly one solution
  – An infinite number of solutions

Graphical Approach
• Equations in 2 unknowns can be visualised graphically
  Consider
  \[ a_1 x_1 + a_2 x_2 = b \]
• Since more unknowns than equations, there are an infinite number of solutions
  If \( a_1 = a_2 = 0 \) then if \( b = 0 \) any \((x_1, x_2)\) satisfies
  if \( b \neq 0 \), no solution

Graphical Approach
• Solutions lie on a line

2 Equations, 2 Unknowns
\[
\begin{align*}
  a_{11} x_1 + a_{12} x_2 &= b_1 \\
  a_{21} x_1 + a_{22} x_2 &= b_2
\end{align*}
\]
  \[
  \begin{pmatrix}
    a_{11} & a_{12} \\
    a_{21} & a_{22}
  \end{pmatrix}
  \begin{pmatrix}
    x_1 \\
    x_2
  \end{pmatrix}
  =
  \begin{pmatrix}
    b_1 \\
    b_2
  \end{pmatrix}
\]
Parallel lines

- Lines are parallel

\[
\begin{align*}
  a_{1}x_1 + a_{2}x_2 &= b_1 \\
  a_{3}x_1 + a_{4}x_2 &= b_2 \\
  a_{11} &= ka_{11} \\
  a_{12} &= ka_{12}
\end{align*}
\]

If \( b_1 = b_2 \) then lines overlap: many solutions
If \( b_1 \neq b_2 \) then lines separate: no solutions

3 Unknowns

- A plane in 3D space
- Two planes either
  - Intersect at a line
  - Are parallel and don’t touch
  - Are identical
- 3 equations give 3 planes
  - If not parallel, they intersect at a single point

Many unknowns

- A single linear equations in \( n \) variables
  \[
  a_{1}x_1 + a_{2}x_2 + \ldots + a_{n}x_n = b
  \]
- A hyperplane in \( n \)-D space
  - Hard to visualise
  - Best to pretend it’s 3D and not worry too much...

Linear Dependence

A set of vectors \( \{u_1, u_2, \ldots, u_n\} \)

is called linearly dependent if one depends linearly on the rest:

\[
\begin{align*}
  u_1 &= \lambda_1 u_2 + \lambda_2 u_3 + \cdots + \lambda_n u_n
\end{align*}
\]

Or there are a set of scalars, some of which are non-zero, such that

\[
\begin{align*}
  \alpha_1 u_1 + \alpha_2 u_2 + \alpha_3 u_3 + \cdots + \alpha_n u_n = 0
\end{align*}
\]
Linear Independence

A set of vectors \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}

is called linearly independent if the only solution to

\[ \alpha_1 \mathbf{u}_1 + \alpha_2 \mathbf{u}_2 + \cdots + \alpha_n \mathbf{u}_n = \mathbf{0} \]

is \( \alpha_i = 0 \quad \forall i = 1, m \).

• In \( n \)-D, there are at most \( n \) vectors in a linearly independent set.

Sets of Equations: Regular Case

• Consider system of equations:

\[
\begin{align*}
  a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
  a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
  &\vdots \\
  a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m
\end{align*}
\]

This is can be written as a vector equation

\[ \mathbf{a}_1x_1 + \mathbf{a}_2x_2 + \cdots + \mathbf{a}_nx_n = \mathbf{b} \]

Vector equations

\[ \mathbf{a}_1x_1 + \mathbf{a}_2x_2 + \cdots + \mathbf{a}_nx_n = \mathbf{b} \]

• When \( n=m \) and the vectors \( \mathbf{a}_i \) are linearly independent, then the system is called regular and has exactly one solution.

• Any point in \( n \)-D space can be reached by a unique linear combination of the \( \mathbf{a}_j \).

Linear Equations

\[
\begin{align*}
  a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
  a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
  &\vdots \\
  a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m
\end{align*}
\]

\[ \mathbf{A}x = \mathbf{b} \]

Regular Case

• Consider case in which \( n=m \)
  – \( \mathbf{A} \) is square

  • Various ways of decomposing \( \mathbf{A} \)
  • Lead to methods of solving \( \mathbf{A}x = \mathbf{b} \)
LU Decomposition

• Any square matrix can be written as a product:
  \[ A = LU \]
  – Where L is lower triangular, U is upper triangular

\[
\begin{bmatrix}
  l_{11} & 0 & \cdots & 0 \\
  l_{21} & l_{22} & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  l_{n1} & l_{n2} & \cdots & l_{nn}
\end{bmatrix}
\begin{bmatrix}
  u_{11} & u_{12} & \cdots & u_{1n} \\
  0 & u_{22} & \cdots & u_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & u_{nn}
\end{bmatrix}
\]

Solving Ax = b using LU decomp.

\[ Ax = (LU)x = L(Ux) = b \]

• First solve for y \[ Ly = b \]
• Then solve for x \[ Ux = y \]

• How does this help?
  • It’s easy to solve equations formed by triangular matrices

Solving triangular linear eqns

\[
Ly = \begin{bmatrix}
  y_1 \\
  l_{21}y_1 \\
  \vdots \\
  l_{n1}y_1
\end{bmatrix} = b
\]

• Solve in sequence
  \[ l_{11}y_1 = b_1 \quad y_1 = b_1 / l_{11} \]
  \[ l_{21}y_1 + l_{22}y_2 = b_2 \quad y_2 = (b_2 - l_{21}y_1) / l_{22} \]

In general \[ y_i = \frac{1}{l_{ii}} \left( b_i - \sum_{j=1}^{i-1} l_{ij} y_j \right) \]
  (Problems if this is near zero)

Solving Linear Eqns using LUD

• Calculate L and U
  – Standard implementations (eg ‘Crout’s algorithm) exist
• Forward substitute to find y
• Back substitute to find x

• For multiple sets of equations \[ Ax = b \]
  – Only do decomposition once
  – Then multiple forward/backward substitutions

Inverse Matrix

• The inverse of a square matrix \( A \) is a matrix \( A^{-1} \) such that
  \[ A^{-1}A = AA^{-1} = I \]

• The columns of \( A^{-1} \) can be found by solving
  \[ \begin{bmatrix}
    1 & 0 & \cdots & 0 \\
    a_{12} & 1 & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{1n} & a_{2n} & \cdots & 1
  \end{bmatrix}
  \]

\[ A^{-1} = \begin{bmatrix}
  c_1 \\
  c_2 \\
  \vdots \\
  c_n
\end{bmatrix} \]
Determinant of a Matrix
• The determinant, |A|, is a measure of the relative volume change applied by the transformation represented by A
  – It is the volume of the unit square/cube/hyper-cube after transforming by A

\[
\begin{pmatrix}
0 & 1 \\
1 & 0 \\
c & d
\end{pmatrix} \rightarrow \begin{pmatrix}\cdot \end{pmatrix}
\]

Determinants and LEs
• In order to have a unique solution to \( Ax = b \)
  for all values of \( b \), the determinant of \( A \), |A|, must be non-zero.
• If |A|=0, then the transformation \( x' = Ax \)
  flattens the unit cube in at least one dimension

\[
\begin{pmatrix}
0 & 1 \\
1 & 0 \\
c & d
\end{pmatrix} \rightarrow \begin{pmatrix}\cdot \end{pmatrix}
\]

Determinants and LEs
• If |A|=0 then either there are
  – no solutions to \( Ax = b \), or
  – infinitely many solutions to \( Ax = b \)
• Before attempting to solve system of equations, need to check the determinant!

Determinant using LUD
• If \( T \) is triangular, \( T = \{t_{ij}\} \)
  \(|T| = \prod_{i=1}^{n} t_{ii}\)

Positive Definite Matrices
• A square matrix \( M \) is called positive definite if for all non-zero vectors \( v \),
  \( v^T M v > 0 \)
• \( x = Mv \) applies a transformation to \( v \)
  – A combination of rotations and shears
  \( v^T x = v^T v = \|v\|^2 \cos \theta > 0 \) if \( \theta < 90^\circ \)
• Thus \( M \) is positive definite if no vector \( v \) is transformed by \( Mv \) so that it has a component in the opposite direction to \( v \)
**Cholesky Decomposition**

- If a square matrix $M$ is symmetric and +ive definite, then $M$ can be decomposed into a product of matrices as follows

$$ M = LL^T $$

$ L = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix}, \quad |M| = |L|^2 = \prod_{i=1}^{n} l_{ii} > 0 $

**Eigenvector Decomposition**

- Any square, symmetric matrix $M$ can be decomposed as follows

$$ M = UDU^T $$

Diagonal matrix $D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$

$U$ is a pure rotation matrix in $\mathbb{R}^n$

$D$ is a diagonal matrix of the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$

$$ U^TU = UU^T = I $$

**Eigenvectors and values**

$$ M = UDU^T $$

$$ MU_j = \lambda_j u_j $$

$ Mu_j = \lambda_j u_j$

The transformation $x = Mu_j$ scales each eigenvector $u_j$ by $\lambda_j$

**Eigenvectors**

- The columns of the rotation matrix are the “Eigen-vectors” of $M$

- The elements of the diagonal matrix are the “eigenvalues” of $M$

Each eigenvector, $u_j$, has a corresponding eigenvalue, $\lambda_j$

Note: There are numerical algorithms for computing the eigenvectors/values of matrices

**Determinant of a symmetric matrix**

- If $M$ is symmetric,

$$ |M| = |UDU^T| = |U||D||U^T| = |D| |1| = \lambda_1 \lambda_2 \ldots \lambda_n $$

$ |M| = |L|^2 = \prod_{i=1}^{n} l_{ii} > 0 $
Positive Definite Symmetric Matrices

- If \( M \) is symmetric and +ive definite, then all its eigenvalues are >0

The transformation \( x = Mu \) scales each eigenvector \( u_i \) by \( \lambda_i \).

If \( \lambda_i \leq 0 \) then \( Mu_i \), flips \( u_i \), so \( M \) cannot be +ive definite.

Solving Regular LEs using eigenvectors

- If \( A \) is symmetric and \( |A| \) is not zero

\[
Ax = b
\]

\[
UDU^T x = b
\]

\[
(UD^{-1}U^T)UDU^T x = (UD^{-1}U^T)b
\]

\[
UD^{-1}(U^T U)DU^T x = UD^{-1}U^T b
\]

\[
U(D^{-1}D)U^T x = UD^{-1}U^T b
\]

\[
UU^T x = UD^{-1}U^T b
\]

\[
x = UD^{-1}U^T b
\]

\[
A^{-1} = UD^{-1}U^T
\]

Singular Value Decomposition

- Any \( m \times n \) matrix with \( m \geq n \) can be decomposed as follows

\[
A = UWV^T
\]

- Orthogonal columns

- Diagonal matrix

- Diagonal elements are called ‘Singular Values’

SVD and Eigenvectors

- Eigenvector decomposition is a special case of SVD for square, symmetric matrices

\[
A = UWV^T
\]

If \( A = A^T \) then \( U = V \) and \( A = UWU^T \)

- Columns of \( U \) are eigenvectors

- Elements of \( W \) are eigenvalues

SVD of square matrices
Matlab commands

Cholesky:
\[ U = \text{chol}(M) \]
\[ (U^*U = M) \]

LU Decomposition:
\[ [L, U] = \text{lu}(M) \]
\[ (L*U = M) \]

SVD:
\[ [U, W, V] = \text{svd}(M, 0) \]
\[ (U*W*V^T = M) \], \( U \) is m x n, \( V \) is n x n

Eigenvectors of a square matrix:
\[ [U, D] = \text{eig}(M) \]
\[ (U*D*U^T = M) \]

VXL (vnl/algo/vnl_cholesky.h)

Cholesky:
\[ \text{vnl_matrix<double> } M; \]
\[ \text{vnl_cholesky } \text{cholesky}(M); \]
\[ \text{vnl_matrix<double> } L = M.\text{lower_triangle}(); \]
\[ \text{vnl_matrix<double> } U = M.\text{upper_triangle}(); \]
\[ \text{double } \text{det} = \text{cholesky.} \text{determinant}(); \]

\[ \text{vnl_vector<double> } \text{rhs}; \]
\[ // \text{Solve } Mx = \text{rhs}; \]
\[ \text{vnl_vector<double> } x = \text{cholesky.solve(rhs)}; \]