Systems of Linear Equations

Linear equation:

\[ a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = b \]

- \(a_1, a_2, \ldots a_n, b\) - constants
- \(x_1, x_2, \ldots x_n\) - variables
- no \(x^2, x^3, \sqrt{x}, \ldots\)
- no cross-terms like \(x_i x_j\)
Systems of Linear Equations

Applications:

1. Reaction stoichiometry (balancing equations)
2. Electronic circuit analysis (current flow in networks)
3. Structural analysis (linear deformations of various constructions)
4. Statistics (least squares analysis)
5. Economics: optimization problems (Nobel prize in economics in 70s for “Linear Programming”).
Systems of Linear Equations

Examples of linear equations:

\[ 7 \, x = 2 \] \quad a \, x = b \quad \text{point in 1D} \\
\[ 3 \, x + 4 \, y = 1 \] \quad a_1 \, x + a_2 \, y = b \quad \text{line in 2D} \\
\[ 2 \, x + 5 \, y - 2 = -3 \] \quad a_1 \, x + a_2 \, y + a_3 \, z = b \quad \text{plane in 3D} \\

What if we have several equations (system)?

How many solutions we will have?
Systems of Linear Equations

Example: What is the stoichiometry of the complete combustion of propane?

\[ C_3H_8 + x O_2 \rightarrow y CO_2 + z H_2O \]

atom balances:

- oxygen \[ 2 x = 2 y + z \]
- carbon \[ 3 = y \] \[ \Rightarrow y = 3 \]
- hydrogen \[ 8 = 2 z \] \[ \Rightarrow z = 4 \]

substitute: \[ 2 x = 10 \] \[ \Rightarrow x = 5 \]

\[ C_3H_8 + 5 O_2 \rightarrow 3 CO_2 + 4 H_2O \]
Systems of Linear Equations

In 2D (2 variables) to solve an SLE is to find an intersection of several lines. 

1 equation: " solutions.

2 equations: a) no solutions (parallel lines)  
b) one solution  
c) " solutions

to have one solution we need the determinant $a_{11}a_{22} - a_{21}a_{12} = 0$, in cases (a) and (c) $a_{11}/a_{21} = a_{12}/a_{22}$.

$\geq 3$ equations: a) no solutions (most likely)  
b) one solution: all equations except 2 are “linear combinations” of the others and may be scrapped if we look at solution only.
Systems of Linear Equations

In general:

If the number of variables $m$ is less than the number of equations $n$ the system is said to be “overdefined”: too many constraints. If the solution still exists, $n-m$ equations may be thrown away.

If $m$ is greater than $n$ the system is “underdefined” and often has many solutions.

We consider only $m = n$ cases.
Matrix Formulation of SLE

Any system of linear equations can be formulated in the matrix form:

\[
\begin{align*}
    a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n &= b_1 \\
    a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n &= b_2 \\
    \vdots \\
    a_{n1} x_1 + a_{n1} x_2 + \ldots + a_{nn} x_n &= b_n
\end{align*}
\]

\[a_{ij}\] - elements of the coefficient matrix \(A\), \(b\) - load vector

\[
\begin{pmatrix}
a_{11} & a_{12} & \ldots & a_{1n} \\
a_{21} & a_{22} & \ldots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \ldots & a_{nn}
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b
\end{pmatrix}
\]

\[A \cdot x = b\]
Matrix Formulation of SLE

SLE in a compact matrix form: \( A \cdot x = b \)
Inverse matrix \( A^{-1} \): \( A \cdot A^{-1} = I = A^{-1} \cdot A \)
\[
A^{-1} \cdot A \cdot x = A^{-1} \cdot b \quad \Rightarrow \quad x = A^{-1} \cdot b
\]
Thus, to solve SLE we need to invert the matrix.

In Matlab:

\[
\gg x = A\backslash b \quad \text{Just one line!!!}
\]

“\( \backslash \)” is a black box. What is inside?
Do we always need \( A^{-1} \) to solve the SLE?
Matrix Formulation of SLE

For n=2:

\[
\begin{pmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{pmatrix}
\cdot
\begin{pmatrix}
  x_1 \\
  x_2
\end{pmatrix}
=
\begin{pmatrix}
  b_1 \\
  b_2
\end{pmatrix}
\]

The solution is:

\[
x_1 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{12}a_{21}} \quad \quad x_2 = \frac{a_{11}b_2 - a_{21}b_2}{a_{11}a_{22} - a_{12}a_{21}}
\]

\(a_{11}a_{22} - a_{12}a_{21} = \text{det}(A)\) is the \textit{determinant} of matrix A.

It should be non-zero for the unique solution to exist.

\[a_{11}a_{22} - a_{12}a_{21} = 0 \iff \frac{a_{11}}{a_{21}} = \frac{a_{12}}{a_{22}}\]
Systems of Linear Equations

\[ A = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}, \text{ where } a_1 = (a_{11} \ a_{12}), \ a_2 = (a_{21} \ a_{22}), \text{ so } \det(A) \neq 0 \text{ is equivalent to } \alpha_1 a_1 + \alpha_2 a_2 \neq 0 \text{ for any } \alpha_{1,2} \neq 0. \]

In this case \( a_1 \) and \( a_2 \) are called linearly independent. This is true for any number of equations.

NB: The SLE has a single solution if the coefficient matrix has a non-zero determinant

or

if the vectors \( a_1, a_2, \ldots a_n \) are linearly independent.
Systems of Linear Equations

To solve the SLE without using $A^{-1}$:
1. eliminate $x$ from equation 2: eq.2 - 2 x eq.1.
2. solve equation 2 for $y$.
3. substitute $y$ into equation 1.
4. solve equation 1.

Example: $x + 2 \ y = -1$

$2 \ x + 2 \ y = 0$
Systems of Linear Equations

Or more generally:

Form the equations.

Eliminate variable until eq-s n, n-1, . . . , 1

have 1, 2, 3, . . . , n variables left.

A is now an upper triangular matrix.

Backsubstitute solution of eq. n to eq. n-1,

n-1 to n-2, . . . , 2 to 1 to solve the system
Systems of Linear Equations

To solve SLE we perform invariant operations, which do not change the solutions:

1. add/subtract the same value to/from both sides of the equation
2. multiply/divide both sides of the equation by the same value
3. add/subtract some equation from another one
4. rearrange equations
5. rearrange columns in the coefficients matrix
Systems of Linear Equations

Example:

\[
\begin{align*}
  x + 2y + z &= 0 \\
  2x + 2y + 3z &= 3 \\
  -x + 3y &= -4
\end{align*}
\]

1. Eliminate \(z\) from eq. 2.
2. Eliminate \(y\) from equation 2.
3. Solve eq. 3 for \(x\) and backsubstitute to eqs 1&2
4. Solve eq. 2 for \(y\) and backsubstitute to eq. 1.
5. Solve eq. 1 for \(z\).
## Gaussian Elimination using Matrix Algebra

1. \[ \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = b \]

   - **pivot element, row**

2. m\(_{21}\) = \(-a_{21}/a_{11}\), add row 1 \(x \times m_{21}\) to row 1

   m\(_{31}\) = \(-a_{31}/a_{11}\), add row 1 \(x \times m_{31}\) to row 3

\[ \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \\ b_3^{(2)} \end{bmatrix} = b \]
Gaussian Elimination using Matrix Algebra

3. \( m_{32} = -a^{(2)}_{32}/a^{(2)}_{22} \), add row 2 x \( m_{32} \) to row1

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
0 & a^{(2)}_{22} & a^{(2)}_{23} \\
0 & 0 & a^{(3)}_{33}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
= 
\begin{bmatrix}
b_1 \\
b^{(2)}_2 \\
b^{(3)}_3
\end{bmatrix}
= b
\]

Upper triangular matrix
Gaussian Elimination using Matrix Algebra

4. Solve for \( x_1, x_2, x_3 \) by backsubstitution:

\[
\begin{align*}
x_3 &= \frac{b_3^{(3)}}{a_{33}}; \\
x_2 &= \frac{b_2^{(2)} - a_{23}^{(2)} x_3}{a_{22}^{(2)}}; \\
x_1 &= \frac{b_1 - a_{22}^{(2)} x_2 - a_{13}^{(2)} x_3}{a_{11}};
\end{align*}
\]
Summary

1. Recognizing systems of linear equations.
3. Gaussian elimination to get an upper triangular matrix.
4. Backsubstitution.
Key Concepts from Previous Lecture

- Solution of linear equations
  - Matrix formulation of equation system
  - Decomposition to upper triangular from
  - Back substitution to solve in reverse order
- Gaussian Elimination algorithm
Essence of the Gaussian Elimination Algorithm

- Form the equations
- Successively eliminate variables until the upper triangular form is reached (ELIMINATION STEP)
- Once the elimination has been completed perform a back substitution in the reverse order to obtain solution for each of the variables (BACK SUBSTITUTION STEP)

Extremely valuable algorithm -- Gaussian Elimination
Numerical Problems

1. Scalability - how big a problem can be solved?
   - Physical memory
   - Disk storage
   - Processor time

2. What is the “fastest” algorithm?

3. What is the most “robust” algorithm?
   Numerical stability: what happens if $a_{ij}=0$ etc.?

4. What are the effects of finite precision arithmetic?
Typical Times (Microseconds)

- Multiplication = 1
- Division  = 3
- Addition  = 0.5
- Subtraction = 0.5

How long to solve $A x = b$, when

$$n = 100, 1,000, 1,000,000?$$
Scalability

Code for Gaussian elimination contains 3 loops:
1. it makes $n-1$ runs to eliminate variables
2. $k$-th run goes through $n-k$ rows ($k = 1, \ldots, n-1$)
3. in $i$-th row we calculate $a_{ij}^{(k)} = a_{ij} - m a_{kj}$ $n-k+1$ times

\[
1 \rightarrow \sum_{k=1}^{n-1} \quad 2 \rightarrow (n-k) \quad 3 \rightarrow (n-k+1)
\]

Overall about \(\sum_{k=1}^{n-1} (n-k)(n-k+1)\) operations.
Gaussian Elimination (How many operations)

- **n-1 runs, eliminating n-1 variables**
  
  ```c
  for ( k=0; k < n-1; k++){
      for ( i=k+1; i < n; i++){
          m = A[i][k]/A[k][k];
          for ( j=k; j < n; j++) {  A[i][j] += -m*A[k][j];
              b[i] += -m*b[k];
          }
      }
  }
  ```

- **n-k runs, k-th variable is eliminated from n-k rows**

  ```c
  for ( i=k+1; i < n; i++){
      m = A[i][k]/A[k][k];
      for ( j=k; j < n; j++) {  A[i][j] += -m*A[k][j];
          b[i] += -m*b[k];
      }
  }
  ```

- **n-k+1 runs, eliminating k-th variable form i-th row**

  ```c
  for ( j=k; j < n; j++) {
      A[i][j] += -m*A[k][j];
  }
  ```

November 2002 10.001 Introduction to Computer Methods
Gaussian Elimination (How many operations)

Useful Identities

\[ 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \]

\[ 2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} \]

Gauss reduction

Multiplication

\[ \frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6} \approx O(n^3) \]

Addition

\[ \frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6} \approx O(n^3) \]

Time scales as \( n^3 \) ! A rather poor scalability.
Numerical Stability

What if one of the diagonal elements is a small number $r$, close to zero?

$$r x_1 + x_2 = 1$$

$$x_1 + x_2 = 2$$

Possible problems caused by dividing by $r$:

1. Overflow: $1/r$ is too big.

2. Numerical instability.
Numerical Stability

After elimination

\[ r \ x_1 + x_2 = 1 \]
\[ 0 + (1-1/r) \ x_2 = 2-1/r \]

After substitution

\[ x_2 = (2 - 1/r)/(1-1/r) \]
\[ x_1 = (1 - x_2)/r \]

if \( 1/r >> 2 \), then \( x_2 = 1 \) and \( x_1 = 0 \).

\[ x_2 = (\text{large number})/(\text{large number}) \]
\[ x_1 = (\text{small number})/(\text{large number}) \] is a problem.
Numerical Stability

Solution - remove small numbers from the diagonal by exchanging rows or columns.

May be done by pivoting:
Exchanging rows just “renumbers” equations.
Exchanging columns “reindexes” variables.
Numerical Stability

Let’s exchange rows in the previous example.

a) \( x_1 + x_2 = 2 \)  
   b) \( x_1 + x_2 = 2 \)  
   c) \( x_1 = 2 - x_2 \)  
   \( r x_1 + x_2 = 1 \)  
   \( (1-r) x_2 = 1-2r \)  
   \( x_2 = (1-2r)/(1-r) \)

The correct answer:

if \( r << 1 \), \( x_2 = 1 \) and \( x_1 = 1 \)
Numerical Stability

Pivoting algorithm:

Searches for the largest $a_{ik}$ in each row below the current one to use for the next elimination step, and rearranges the rows so that $m_{ik}$ is always less than one.
Numerical Stability

Example:

Augmented matrix: \( n \times (n+1) \)

\[
\begin{bmatrix}
0.0001 & 0.5 & 0.5 \\
0.4 & -0.3 & 0.1
\end{bmatrix} = [A, b]
\]

Use 4 digit arithmetic:

\( 9.9999 \ldots \rightarrow 9.9998 \)

\[
\begin{bmatrix}
0.0001 & 0.5 & 0.5 \\
0 & -2000 & -2000
\end{bmatrix}
\]

\( 0.1 - (4000)(0.5) = -1999.9 = -2000 \)

\( -0.3 - (4000)(0.5) = -2000.3 = -2000 \)

\[
x_2 = \frac{-2000}{-2000} = 1 \quad x_1 = \frac{1}{0.0001} (0.5 - (0.5)(1)) = 0
\]
Numerical Stability

\[
\tilde{A} = \begin{bmatrix}
0.4 & -0.3 & | 0.1 \\
0.0001 & 0.5 & | 0.5 \\
0.5 & -0.3 & | 0.1 \\
0 & 0.5 & | 0.5 \\
\end{bmatrix}
\]

\[
0.5 + (0.3)0.000025 = 0.5 \\
0.5 - (0.1)0.000025 = 0.5
\]

\[
x_2 = \frac{0.5}{0.5} = 1 \\
x_1 = \frac{1}{0.4} (0.1 + (0.3)(1)) = 1
\]

“1” instead of “0”
Quite a difference!
LU factorization.

Linear system $Ax = b$ is solved by Gaussian elimination.

Matrix $A$ is fixed, but we have a set of $b$-s: $b_1, b_2, b_3$…

How to avoid repeating the solution for $A$ and do it only for $b$-s??

*Answer*: Express Gaussian elimination as a matrix.

Each step of elimination is represented by some elementary matrix acting on $A$ and on $b$. Overall GE will be represented by the product of these matrices.
LU factorization.

If $A$ is nonsingular, $A = LU$,

$U$ - upper diagonal, $L$ - lower diagonal

1. $Ax = b$ $\Rightarrow$ $LUx = b$ $\Rightarrow$ $L(Ux) = b$ $\Rightarrow$ $Ly = b$

$y$ is found by forward substitution.

2. $Ux = y$

$x$ is found by backsubstitution

Gaussian elimination $== L^{-1}$.

LU factorization is a standard way to solve SLE in case $A$ is a non-singular matrix.
LU factorization.

*Elementary matrix:* A matrix obtained from identity matrix by the following “elementary row operations” is called elementary matrix.

*Elementary row operations:*

1. multiply a non-zero constant throughout a row
2. interchange two rows
3. add a constant multiple of another row

(remember invariant operations ?)
LU factorization.

Examples of elementary matrices:

\[
\begin{bmatrix}
1 & 0 \\
0 & -5
\end{bmatrix}
\text{ multiply the second row of } I_2 \text{ by } -5
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 3 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\text{ add 3 times the 3rd row to the 1st row of } I_3
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\text{ interchange the 2nd & the 4th rows of } I_4
\]

November 2002 10.001 Introduction to Computer Methods
How do elementary matrices work?

\[ E = \begin{bmatrix}
1 & 0 & 0 \\
-2 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \quad \text{“adding } -2 \text{ x first row to the second row”}
\]

\[ E \cdot b = \begin{bmatrix}
1 & 0 & 0 \\
-2 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 - 2b_1 \\ b_3 \end{bmatrix} \]
How do elementary matrices work?

\[
P = \begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix}
\]

“Interchanging the first and the third rows”:
permutation matrix

\[
P \cdot b = \begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} b_3 \\ b_2 \\ b_1 \end{bmatrix}
\]
Elementary Operations.

\[
Ax = \begin{bmatrix}
2 & 1 & 1 \\
4 & 1 & 0 \\
-2 & 2 & 1
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} = \begin{bmatrix}
b_1 \\
b_2 \\
b_3
\end{bmatrix} = b
\]

Elementary matrices for Gaussian elimination:

<table>
<thead>
<tr>
<th>2 row - 2x1st row</th>
<th>3 row + 1st row</th>
<th>3 row + 3x2 row</th>
</tr>
</thead>
</table>
| \[
E_1 = \begin{bmatrix}
1 & 0 & 0 \\
-2 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\] | \[
E_2 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 1
\end{bmatrix}
\] | \[
E_3 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 3 & 1
\end{bmatrix}
\] |
Elementary Operations.

Inverting elementary matrices:

\( E_k^{-1} \), \( (E_k^{-1})_{ii} = (E_k)_{ii}, \quad (E_k^{-1})_{ij} = -(E_k)_{ij} \) for \( i \neq j \)

\[
E_1 = \begin{bmatrix}
1 & 0 & 0 \\
-2 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}, \quad E_1^{-1} = \begin{bmatrix}
1 & 0 & 0 \\
2 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
P = \begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix} = P^{-1}
\]
LU factorization.

\[ E_k E_{k-1} \ldots E_1 (A \, x) = E_k E_{k-1} \ldots E_1 b \]

\[ \begin{align*}
L^{-1} A \, x &= L^{-1} b, \\
E_k E_{k-1} \ldots E_1 &= L^{-1}
\end{align*} \]

On the other hand: \( L = E_1^{-1} E_2^{-1} \ldots E_k^{-1} \)

Also:

\[
E_1 = \begin{bmatrix}
1 & 0 & 0 \\
-m_{21} & 1 & 0 \\
0 & 0 & 1
\end{bmatrix},
E_2 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
-m_{31} & 0 & 1
\end{bmatrix},
E_3 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & -m_{32} & 1
\end{bmatrix}
\]

\( m\)-\( s \) – coefficients form Gaussian elimination.

\[
L^{-1} = E_3 E_2 E_1 = \begin{bmatrix}
1 & 0 & 0 \\
-m_{21} & 1 & 0 \\
-m_{31} & -m_{32} & 1
\end{bmatrix}
\]

\( U \) is obtained after The elimination: \( U = L^{-1} A \).
Gains due to LU factorization?

We care only for $L^{-1}$: it allows for calculation of load vectors without repeating Gaussian elimination. For every $b_j$ we need to calculate $L^{-1}b_j$ instead of performing a complete Gaussian elimination.

$n$ times (row \cdot column) \sim n^2$ operations instead of $n^3$! Quite a difference.
Lecture Summary

- Solution of linear equations
  - Matrix formulation of equation system
  - Decomposition to upper triangular form
  - Back substitution to solve in reverse order
- Gaussian Elimination algorithm
- LU decomposition if many $b$-s for the same $A$. 
Gaussian Elimination Algorithm

Forward Reduction:
for k=1,...,n-1
  for i=k+1,...,n
    \( l_{ik} = \frac{a_{ik}}{a_{kk}} \)
    for j=k+1,...,n
      \( a_{ij} = a_{ij} - l_{ik}a_{kj} \)
    end loop j
  end loop i
b_i = b_i - l_{ik}b_k
end loop i
end loop k

Back substitution:
for k = n,...,1
  \( x_k = b_k \)
  for i=k+1,...,n
    \( x_k = x_k - a_{ki}x_i \)
  end loop i
  \( x_k = x_k / a_{kk} \)
end of loop k
Gaussian Elimination II

**Forward Reduction:**
for \( k = 1, \ldots, n-1 \)
for \( i = k + 1, \ldots, n \)
\[
\begin{align*}
  l_{ik} &= \frac{a_{ik}}{a_{kk}} \\
  \text{Row}_i &= \text{Row}_i - l_{ik} \times \text{Row}_k \\
  b_i &= b_i - l_{ik} \times b_k
\end{align*}
\]
end loop \( i \)
end loop \( k \)

**Back substitution:**
for \( k = n, \ldots, 1 \)
\[
\begin{align*}
  x_k &= b_k - A(k,:) \times x \\
  x_k &= x_k/a_{kk}
\end{align*}
\]
end of loop \( k \)