PART I: SOLUTION OF LINEAR EQUATIONS

Many computational problems involve the solution of systems of linear equations, methods for solving systems of linear equations are part of the building blocks for computational sciences.

A computational method for the solution of a system of linear equations,

\[ Ax = b, \]

where \( A \) is an \( m \times n \) matrix, can be categorized as one of the following

\( \diamond \) a direct method,

\( \diamond \) an iterative method,

\( \diamond \) a hybrid of direct methods and iterative methods.

In the case that the inverse of a nonsingular matrix \( A \) is needed but not readily available, we seek for numerical solutions of the following matrix equations

\[ AX = I, \quad \text{or} \quad A^T Y = I, \]

each of which can be considered as a linear system with multiple right hand sides.

We are concerned with the following issues

\( \diamond \) the existence of a solution,

\( \diamond \) the uniqueness of solutions,

\( \diamond \) the sensitivity of a solution to perturbations,

\( \diamond \) the stability of a computational method,
diamond the computational complexity of a computational method.

The first two issues are clearly defined in the ideal situation when there is no perturbation involved.

Review:

- There exists a solution iff \( b \in \text{span}(A) \).
- The solution, when exists, is unique iff \( \text{null}(A) = \{0\} \).
- Let \( k = \text{rank}(A) \). Then, \( k = \text{dim}(\text{span}(A)) = \text{dim}(\text{span}(A^T)) \).

**Direct Methods**

Special cases: \( A \) is non-singular \((k = m = n)\) and

1. diagonal, 
   (elementwise inversion)
2. lower or upper triangular, 
   (forward or backward substitution)
3. unitary or orthogonal, 
   in particular, permutation matrices are orthogonal, 
   (conjugate transposition is the inverse)
4. of rank-1 update to the identity matrix 
   \[ A = I - xy^H, \quad y^Hx \neq 1. \]
   (In this case, \( A^{-1} = I - \beta xy^H \) with \( \beta = 1 / (y^Hx - 1) \).)

In these special cases, either the inverse can be obtained easily, or the solution \( x \) can be obtained without computing the inverse.

For the general case the matrix is factored into special matrices, and the solution is obtained by successively solving the
decomposed special systems. The process can be described as follows. Let

\[ A = A_1 A_2 \cdots A_k \]

be a factorization of \( A \), the factors have special structures. Then the solution \( x \), if exists, can be obtained by solving the factored, special systems one by one

\[ A_j y_j = y_{j-1}, \quad j = 1 : k, \quad y_0 = b, x = y_k. \]

We introduce some basic matrix factorizations. In many cases it is known that the matrix in question has linearly independent columns and the fact is taken into account to simplify the factorizations and reduce computation complexity.

- For \( A \) with linearly independent columns (\( k = n \leq m \)).

1. **LU Factorization with Partial Pivoting.** There exists a permutation matrix \( P \), a lower unit triangular matrix \( L \) and an upper triangular matrix \( U \) such that

\[ PA = LU, \quad U = \begin{bmatrix} U_{11} \\ 0 \end{bmatrix}, \]

where \( U_{11} \) is of order \( n \) and nonsingular.

2. **QR Factorization.** There exists a unitary matrix \( Q \), and an upper triangular matrix \( R \) such that

\[ A = QR, \quad R = \begin{bmatrix} R_{11} \\ 0 \end{bmatrix}, \]

where \( R_{11} \) is of order \( n \) and nonsingular.

- For arbitrary matrix,
1. **LU FACTORIZATION WITH TOTAL PIVOTING.** There exist permutation matrices $P_r$ and $P_c$, a lower unit triangular matrix $L$ and an upper triangular matrix $U$ such that

$$P_r A P_c = LU,$$

$$U = \begin{bmatrix} U_{11} & U_{12} \\ 0 & 0 \end{bmatrix},$$

where $U_{11}$ is of order $k = \text{rank}(A)$ and nonsingular. Such a factorization reveals the matrix rank.

2. **QR FACTORIZATION WITH COLUMN PIVOTING.** There exists a permutation matrix $P$, a unitary matrix $Q$, and an upper triangular matrix $R$ such that

$$A P = QR,$$

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix},$$

where $R_{11}$ is of order $k = \text{rank}(A)$ and nonsingular.

The following matrix decompositions are useful for matrix analysis as well as matrix computations.

- **Eigenvalue Decomposition (EVD).** For any square matrix $A$, there exists a nonsingular matrix $X$ and an upper bi-diagonal matrix $J$ such that

$$A = X J X^{-1}.$$

It is called the eigenvalue decomposition of $A$. The diagonal elements of $J$ are *eigenvalues* of $A$. When $J$ is diagonal, $A$ is said to be diagonalizable by similarity transformation and have a non-defect eigen-system. In such case, a function of $A$, $f(A)$ may be studied in the spectral domain. In particular, if $p$ is a polynomial, then $p(A) = X p(J) X^{-1}$. A Hermitian matrix is diagonalizable by unitary, similarity transformation.
Singular value decomposition (SVD). For any matrix, there exist unitary matrices $U$ and $V$, and a non-negative diagonal matrix $\Sigma = \text{diag}(\sigma_i)$ such that

$$A = U\Sigma V^H.$$ 

The diagonal elements of $\Sigma$ are the *singular values* of $A$. It may be assumed without loss of generality that the singular values $\sigma_i$ are ordered so that $\sigma_i \geq \sigma_{i+1}$. The columns of $U$ and $V$ are, respectively, the left and right *singular vectors*.

The relationships between Hermitian EVD and SVD are as follows.

○ A SVD is related to a pair of symmetric positive definite EVDs

$$A^H A = V\Sigma^2 V^H, \quad \text{and} \quad AA^H = U\Sigma^2 U^H,$$

○ A SVD is related to a symmetric indefinite EVD

$$\begin{bmatrix} 0 & A \\ A^H & 0 \end{bmatrix} = 1/2 \begin{bmatrix} U & U \\ V & -V \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix} \begin{bmatrix} U & U \\ V & -V \end{bmatrix}^H.$$ 

We use *factorization* to indicate that the factoring computation can be carried out with finite many arithmetic operations. In fact the proofs for such factorizations are often computationally constructive. We use *decomposition* to indicate that the exact decomposition may not necessarily be obtained with finite many arithmetic operations. In numerical computations, the decompositions are computed up to a precision related to the machine precision, and involve only finite many arithmetic operations. There are other factorizations and decompositions, many are variations of the basic ones introduced above.
With each of the factorizations or decompositions, the existence and uniqueness of the solution can be specifically studied. The EVD is often used in convergence analysis of iterative methods for the solution of linear equations. The SVD is often employed to study the sensitivity of the solution to perturbation, or the stability of an algorithm. It can be also used in the analysis of LS solution.

**Sensitivity analysis**

In sensitivity analysis we ask whether or not the solution changes dramatically with a small perturbation in the data \((A, b)\). Perturbations in the data may come from many sources, including the rounding errors in numerical computations.

We illustrate sensitivity analysis with the case that \(A\) is non-singular and we use 2-norm for the metric on the change. We consider the effects of perturbations in the matrix \(A\) and in the right hand side separately, and leave their joint effect as an exercise problem.

1. sensitivity to perturbation in the right hand side \(b\)

\[
A(x + \Delta x) = b + \Delta b.
\]

First we have \(\Delta x = A^{-1}\Delta\). Then, we can verify the following bound on \(\Delta x\), via the SVD of \(A\),

\[
\frac{\|\Delta x\|_2}{\|x\|_2} \leq \kappa_2(A) \frac{\|\Delta b\|_2}{\|b\|_2}.
\]

where \(\kappa_2 = \sigma_{\text{max}}/\sigma_{\text{min}} = \|A\|_2\|A^{-1}\|_2\) is called the *condition number* in 2-norm, which indicates the sensitivity of the solution to the perturbation. Via the SVD, we can also find the
perturbations in $b$ that maximize the deduced perturbation in $x$ and achieve the perturbation bound. Therefore, a large condition number is undesirable, unless we have control over the perturbation.

2. Sensitivity to perturbation in the matrix $A$

$$(A + \Delta A)(x + \Delta x) = b.$$ 

If $\|A^{-1}\Delta A\|_2 < 1/k$, for $k > 1$, then

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq \frac{\|A^{-1}\Delta A\|_2}{1 - \|A^{-1}\Delta A\|_2} < \frac{k}{k - 1}\kappa_2(A)\frac{\|\Delta A\|_2}{\|A\|_2}.$$ 

Thus, $\kappa_2$ also indicates the sensitivity of the solution to the perturbation in the matrix.

3. The above analysis tells that a solution with small residual, $\|b - Ax\|_2/\|b\|_2$, is not necessarily a good solution. A solution with small residual is good when the system is well-conditioned, i.e., the condition number is not unreasonably large.

When $(A, b)$ are the first-hand data, we may say the sensitivity of the solution is determined by the problem itself. So, when the solution is sensitive, we can blame on the posed problem. However, in the case that the system of linear equations, $(A, b)$, is derived from another problem, such as from an interpolation problem or a LS problem, we shall take the sensitivity issue into account in the problem formulation. In other words, we ask the question whether the problem is well posed.

Algorithm stability

The words “stability analysis” are seen in many places with different meanings in different circumstances. Here we introduce the
concept of the backward stability of an algorithm for the solution of a system of linear equations. An algorithm is backward stable if (1) the computed solution \( \bar{x} \) can be expressed as the exact solution to a perturbed system

\[
(A + \Delta A)\bar{x} = b + \Delta b,
\]

and (2) the perturbations are small in some sense. This concept separates, to a great extend, the algorithm analysis from the sensitivity analysis on the posed problem itself.

The stability of an algorithm may be analyzed a priori or/and estimated a posteriori. A priori backward analysis is valuable in evaluating the effectiveness of an algorithm. For example, the LU factorization without pivoting is proved unstable, except for some special systems (such as a symmetric positive definite system). However such analysis is not always feasible. A posteriori estimation is relatively easy. Let \( r = b - A\bar{x} \) be the residual of the computed solution \( \bar{x} \). Then \( \bar{x} \) satisfies the following perturbed equations.

\[
\begin{align*}
\diamond \text{Perturbation in the right hand side :} \\
A\bar{x} &= b + \Delta b, \quad \Delta b = -r.
\end{align*}
\]

It tells that if \( \|r\|_2 \) is relatively larger than \( \|b\|_2 \), then the residual amounts to a large perturbation in the right hand side.

\[
\begin{align*}
\diamond \text{Perturbation in the matrix :} \\
(A + \Delta A)\bar{x} &= b, \quad \Delta A = ry^H/(y^H\bar{x}), \quad y^H\bar{x} \neq 0,
\end{align*}
\]

i.e., the perturbation in \( A \) is of rank-1 and \( y \) can be any vector not orthogonal to \( \bar{x} \). A special choice of \( y \) is \( \bar{x} \). With this
choice we see that if $\|r\|_2$ is relatively larger than $\|\bar{x}\|_2$, then the residual amounts to a large perturbation in the matrix.

In practice, the residual $r$ is to be computed in higher precision.

**Algorithm complexity**

The direct methods, in general, are of $O(n^3)$ in arithmetic complexity and $O(n^2)$ in memory space requirement for a system with an $n \times n$ matrix, where the decompositions are computed to a precision related to the machine precision. Systems of small size can be solved quickly by these methods on modern computers.

There are many computational problems with matrices that are so large that the methods of such complexity are too expensive to use. Some of the matrices are sparse, the iterative methods are used instead to exploit the sparsity. For large and dense matrices, compressed representations are sought first, and then the iterative methods are employed to exploit the compressed representations.