

# BE5B01DEN: Topics for Final State Examination

## Topic: Fundamentals of numerical mathematics.

1. What types of errors we recognize (in general, for a precise number  $x$  and its approximation  $\hat{x}$ ) and what information they tell us? How do those errors appear in calculations? What impact does it have on numerical computing?

### 1. Types of Errors:

**Round-off Error:** This occurs due to the finite precision with which computers represent numbers. When a precise number  $x$  is approximated by a number  $\hat{x}$  that can be represented in the computer's finite precision, the small difference between these values is the round-off error. This type of error is inherent in floating-point calculations.

(舍入误差：这是由于计算机表示数字的精度有限而发生的。当一个精确的数  $x$  用一个数近似时  $\hat{x}$  可以用计算机的有限精度表示，这些值之间的微小差异就是舍入误差。这种类型的错误是浮点计算中固有的。)

**Truncation Error:** This happens when an infinite process (such as an infinite series or an integral) is approximated by a finite one. For example, when we use a finite number of terms of a series to estimate a value, the difference between the true value and the estimated value is the truncation error. This type of error is common in numerical methods like numerical integration or when solving differential equations using discrete approximations.

(当无限过程（例如无限级数或积分）被有限过程逼近时，就会发生这种情况。例如，当我们使用级数的有限项来估计一个值时，真实值和估计值之间的差异就是截断误差。这种类型的错误在数值积分等数值方法中或使用离散近似求解微分方程时很常见。)

### 2. Appearance of Errors in Calculations:

- Round-off errors accumulate in calculations, especially in iterative processes or long computations. This can lead to significant discrepancies in results, particularly in sensitive algorithms.

(舍入误差会在计算中累积，特别是在迭代过程或长时间计算中。这可能会导致结果出现显著差异，特别是在敏感算法中。)

- Truncation errors are determined by how well the numerical method approximates the mathematical problem. The choice of method (e.g., the number of intervals in numerical integration, the step size in numerical solutions of differential equations) can significantly affect the size of the truncation error.

(截断误差取决于数值方法对数学问题的近似程度。方法的选择（例如，数值积分中的区间数、微分方程数值解中的步长）可以显着影响截断误差的大小。)

### 3. Impact on Numerical Computing

- The combined effect of round-off and truncation errors influences the overall accuracy and stability of numerical algorithms. It's crucial to balance these errors, as reducing one can sometimes increase the other.

(舍入误差和截断误差的综合影响影响数值算法的整体精度和稳定性。平衡这些错误至关重要，因为减少一个错误有时会增加另一个错误。)

- In some cases, especially in ill-conditioned problems or with unstable algorithms, errors can grow exponentially, leading to completely inaccurate results.

(在某些情况下，特别是在病态问题或算法不稳定的情况下，错误可能呈指数增长，导致完全不准确的结果。)

- Understanding and managing these errors is critical for choosing appropriate numerical methods and for interpreting the results of numerical computations correctly.

(理解和管理这些误差对于选择适当的数值方法和正确解释数值计算结果至关重要。)

Truncation Error: Truncation errors are introduced when exact mathematical formulas are represented by approximations. An effective way to understand truncation error is through a Taylor Series approximation. Let's say that we want to approximate some function,  $f(x)$  at the point  $x_{i+1}$ , which is some distance,  $h$ , away from the basepoint  $x_i$ , whose true value is shown in black in Figure 1. The Taylor series approximation starts with a single zero order term and as additional terms are added to the series, the approximation begins to approach the true value. However, an infinite number of terms would be needed to reach this true value.

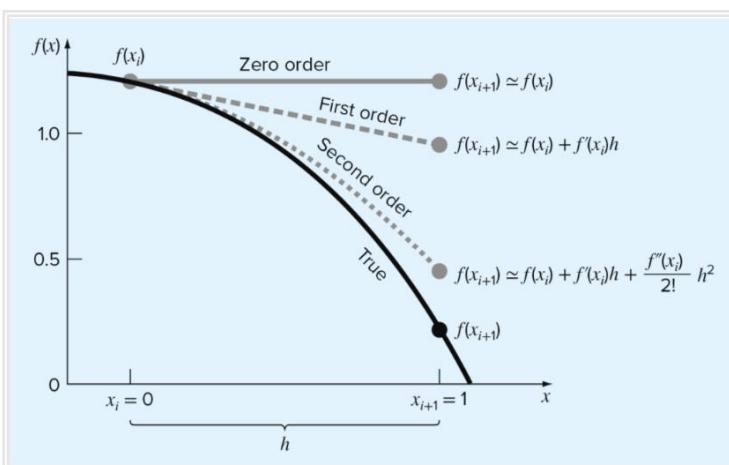


Figure 1: Graphical representation of a Taylor Series approximation (Chapra, 2017)

The Taylor Series can be written as follows:

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \cdots + \frac{f^{(n)}(x_i)}{n!}h^n + R_n$$

where  $R_n$  is a remainder term used to account for all of the terms that were not included in the series and is therefore a representation of the truncation error. The remainder term is generally expressed as  $R_n = O(h^{n+1})$  which shows that truncation error is proportional to the step size,  $h$ , raised to the  $n+1$  where  $n$  is the number of terms included in the expansion. It is clear that as the step size decreases, so does the truncation error

(其中  $R_n$  是余数项，用于解释序列中未包含的所有项，因此代表截断误差。余项一般表示为  $R_n = O(h^{n+1})$ ，这表明截断误差与步长  $h$  成正比， $h$  上升到  $n+1$ ，其中  $n$  是展开式中包含的项数。很明显，随着步长减小，截断误差也减小。)

**2. Describe some method (good candidates are bisection or Newton method) for finding roots (that is, solving equations) numerically. Discuss how we recognize when to stop the algorithm.**

**Newton's Method:**

1. **Description:** Newton's method is used to find the roots of a real-valued function  $f(x)$ . The idea is to start with an initial guess  $x_0$  for the root and iteratively improve this guess based on the function's value and its derivative at that point.
2. **Algorithm:**
  - Start with an initial guess  $x_0$ .
  - For each iteration  $n$ , compute the next approximation  $x_{n+1}$  using the formula:  
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$
  - Repeat the process until a stopping criterion is met (discussed below).
3. **Requirements:**
  - The function  $f(x)$  should be differentiable, and its derivative  $f'(x)$  should be continuous.
  - The initial guess  $x_0$  should be chosen close to the actual root for better convergence.
4. **Convergence:** The method usually converges rapidly if the initial guess is close to the true root and the function behaves well (i.e., no sharp bends or cusps near the root).

**Stopping Criteria:**

1. **Absolute or Relative Tolerance on the Function Value:**
  - Stop if  $|f(x_n)|$  is less than a specified tolerance. This checks how close the function value is to zero.
  - Alternatively, stop if the relative change in the function value is small enough, i.e.,  $|f(x_{n+1}) - f(x_n)|/|f(x_n)|$  is below a threshold.
2. **Absolute or Relative Change in x:**
  - Stop if the absolute change in the successive approximations is small, i.e.,  $|x_{n+1} - x_n|$  is less than a specified tolerance.
  - Similarly, the relative change,  $|x_{n+1} - x_n|/|x_n|$ , can be used as a criterion.
3. **Maximum Number of Iterations:**
  - Set a maximum number of iterations to prevent the algorithm from running indefinitely, especially in cases where it may not converge.

Choosing appropriate stopping criteria is crucial for balancing computational efficiency and the accuracy of the result. In practice, a combination of these criteria is often used to ensure that the algorithm stops when a sufficiently accurate solution is found or when it appears not to be converging.

**Bisection Method:**

1. **Description:** The Bisection Method is used to find a root of a continuous function  $f(x)$  within a given interval  $[a, b]$  where  $f(a)$  and  $f(b)$  have opposite signs. The method is based on the Intermediate Value Theorem, which ensures that a root exists within the interval.
2. **Algorithm:**
  - Start with an interval  $[a, b]$  where  $f(a) \cdot f(b) < 0$ .
  - Compute the midpoint  $c = \frac{a+b}{2}$  of the interval.
  - Determine which subinterval  $[a, c]$  or  $[c, b]$  contains a root, based on the sign of  $f(c)$ . If  $f(a) \cdot f(c) < 0$ , the root lies in  $[a, c]$ ; otherwise, it lies in  $[c, b]$ .
  - Update the interval to the subinterval containing the root and repeat the process.
3. **Convergence:** The Bisection Method is guaranteed to converge to a root if the function is continuous on the interval and  $f(a)$  and  $f(b)$  have opposite signs.

**Stopping Criteria:**

1. **Absolute Tolerance on Interval Size:**
  - Stop if the length of the interval  $[a, b]$  is less than a specified tolerance. This checks if the interval is sufficiently small, indicating that the root has been approximated closely enough.
2. **Absolute or Relative Tolerance on the Function Value:**
  - Stop if  $|f(c)|$  is less than a specified tolerance, indicating that the function value at the midpoint is close enough to zero.
  - Alternatively, a relative tolerance based on the change in function value can be used.
3. **Maximum Number of Iterations:**
  - Set a maximum number of iterations to prevent the algorithm from running indefinitely, particularly in cases where the function value does not reach the desired tolerance.

The choice of stopping criteria depends on the desired accuracy and computational resources. Typically, a combination of these criteria is used to ensure that the algorithm stops when a sufficiently accurate solution is found or when further iterations are unlikely to produce a significantly better approximation.

The simplest root-finding algorithm is the bisection method. Let  $f$  be a continuous function, for which one knows an interval  $[a, b]$  such that  $f(a)$  and  $f(b)$  have opposite signs (a bracket). Let  $c = (a + b)/2$  be the middle of the interval (the midpoint or the point that bisects the interval). Then either  $f(a)$  and  $f(c)$ , or  $f(c)$  and  $f(b)$  have opposite signs, and one has divided by two the size of the interval.

Newton's method assumes the function  $f$  to have a continuous derivative.

Set the accuracy so that the error is less than the accuracy after many iterations.

**3. Describe some method (most likely the Euler method) for solving initial value problems (an ODE with initial condition) numerically. Discuss the notion of the order of method.**

描述一些以数值方式解决初始值问题（具有初始条件的 ODE）的方法（最有可能是欧拉方法）。讨论方法顺序的概念。

**Euler Method:**

1. **Problem Setup:** Consider an initial value problem described by the ODE  $\frac{dy}{dt} = f(t, y)$  with an initial condition  $y(t_0) = y_0$ .

2. **Method:**

- The basic idea is to approximate the derivative  $\frac{dy}{dt}$  at a point by the slope of the line through that point.
- The method proceeds in steps. Starting from the initial point  $(t_0, y_0)$ , we calculate subsequent points using the formula:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

where  $h$  is the step size,  $y_n$  is the approximation of  $y$  at  $t_n$ , and  $t_{n+1} = t_n + h$ .

3. **Implementation:**

- Choose a step size  $h$  and compute  $y_1, y_2, \dots$  iteratively.
- The smaller the step size, the more accurate the approximation, but the more computations are required.

**Order of the Method:**

- The order of a numerical method for solving differential equations is a measure of the rate at which the error decreases as the step size decreases.
- For the Euler Method, it is a first-order method. This means that the local truncation error (the error made in a single step) is proportional to  $h^2$ , and the global error (the error in the final solution) is proportional to  $h$ .
- Higher-order methods, such as the Runge-Kutta methods, can achieve greater accuracy with fewer steps but are more complex.

In summary, the Euler Method is a straightforward approach to solving initial value problems for ODEs. It is easy to understand and implement but is less accurate than higher-order methods. The concept of order is crucial in numerical analysis, as it helps determine the efficiency and accuracy of different methods for a given problem.

### 方法顺序:

- 求解微分方程的数值方法的阶数是误差随着步长减小而减小的速率的度量。
- 对于欧拉方法来说，它是一阶方法。这意味着局部截断误差（一步中产生的误差）与  $H^2$ ，全局误差（最终解中的误差）与  $H$ 。
- 高阶方法（例如龙格-库塔方法）可以通过更少的步骤实现更高的精度，但也更复杂。

总之，欧拉方法是解决 ODE 初值问题的一种直接方法。它易于理解和实现，但不如高阶方法准确。阶数的概念在数值分析中至关重要，因为它有助于确定针对给定问题的不同方法的效率和准确性。

The assumption is that over a time-span  $\Delta t = t_{n+1} - t_n$ . The original differential equation by

$$\frac{d\vec{y}}{dt} = f(t, \vec{y}) \implies \frac{\vec{y}_{n+1} - \vec{y}_n}{\Delta t} \approx f(t, \vec{y}) \text{ and get } \vec{y}_{n+1} = \vec{y}_n + \Delta t \cdot f(t, \vec{y})$$

Thus, the Euler method gives an iterative scheme by which the future values of the solution can be determined. Graphically, the slope (derivative) of a function is responsible for generating each subsequent approximation of the solution  $\vec{y}(t)$ . It is important to note that the truncation error in the above equation is  $O(\Delta t^2)$

#### 4. Discuss the notion of computational complexity and show how it applies to solving large systems of linear equations using elimination.

##### Computational Complexity:

1. **Definition:** Computational complexity refers to the scaling behavior of the algorithm's resource requirements (like time or space) with respect to the input size. It's usually expressed in terms of "big O" notation, which provides an upper bound on the growth rate of the resource requirement.
2. **Time Complexity:** Time complexity is concerned with how the computational time increases as the size of the input (in this case, the number of equations and variables in the linear system) grows.
3. **Space Complexity:** Space complexity deals with the amount of memory required by an algorithm as a function of the input size.

## **Application to Solving Large Systems of Linear Equations Using Elimination:**

### **1. Gaussian Elimination:**

- Gaussian elimination is a method for solving linear systems that involves three main steps: (a) transforming the system into an upper triangular matrix using forward elimination, (b) back substitution to solve for the variables, and (c) in some cases, additional steps like pivoting for numerical stability.

### **2. Complexity Analysis:**

- **Time Complexity:** The time complexity of Gaussian elimination is  $O(n^3)$  for a system with  $n$  variables. This cubic growth rate comes from the fact that for each row, the algorithm performs a series of operations on all remaining rows, leading to a total number of operations proportional to  $n^3$ .
- **Space Complexity:** The space complexity is  $O(n^2)$ , as we need to store an  $n \times n$  matrix and an  $n$ -element vector.

### **3. Implications:**

- For small systems, Gaussian elimination is efficient and practical. However, as the number of variables increases, the cubic time complexity becomes a significant limitation, making the method computationally expensive for very large systems.
- In practice, for large systems, more efficient algorithms (like iterative methods or specialized forms of elimination) or approximation techniques might be used to reduce computational cost.



Computational complexity, a measure of the amount of computing resources (time and space) that a particular algorithm consumes when it runs. Direct methods such as Gauss elimination methods. Matrix is symmetric matrix

### **计算复杂度：**

1. **定义**: 计算复杂度是指算法的资源需求（如时间或空间）相对于输入大小的缩放行为。它通常用“大 O”符号表示，它提供了资源需求增长率的上限。
2. **时间复杂度**: 时间复杂度涉及计算时间如何随着输入大小（在本例中为线性系统中的方程和变量的数量）的增加而增加。
3. **空间复杂度**: 空间复杂度涉及算法所需的内存量，作为输入大小的函数。

### **使用消元法求解大型线性方程组的应用：**

#### **1. 高斯消元法：**

- 高斯消元法是一种求解线性系统的方法，涉及三个主要步骤：(a) 使用前向消元法将系统转换为上三角矩阵，(b) 后代法求解变量，以及(c) 在某些情况下，附加像旋转以获得数值稳定性这样的步骤。

#### **2. 复杂性分析：**

- **时间复杂度**: 高斯消元法的时间复杂度为在 $O(n^3)$ 对于一个系统 $n$ 变量。这种立方增长率来自这样一个事实：对于每一行，算法对所有剩余行执行一系列操作，导致操作总数与 $n^3$ 。
- **空间复杂度**: 空间复杂度为在 $O(n^2)$ ，因为我们需要存储 $n \times n$ 矩阵和 $n$ -元素向量。

#### **3. 影响：**

- 对于小型系统，高斯消去法是高效且实用的。然而，随着变量数量的增加，立方时间复杂度成为一个显着的限制，使得该方法对于非常大的系统来说计算成本昂贵。
- 在实践中，对于大型系统，可以使用更有效的算法（如迭代方法或专门形式的消除）或近似技术来降低计算成本。

5. What is a general solution of an ODE? What is a particular solution of an ODE? How can we determine (choose) a certain particular solution?

**General Solution of an ODE:**

1. **Definition:** The general solution of an ODE encompasses all possible solutions to the differential equation. It includes arbitrary constants (one for each order of the differential equation) because ODEs usually have infinitely many solutions.
2. **Example:** For a first-order ODE, the general solution might be expressed in the form  $y(t) = C \cdot g(t)$ , where  $C$  is an arbitrary constant and  $g(t)$  is a function determined by the ODE.

**Particular Solution of an ODE:**

1. **Definition:** A particular solution is a specific solution to the differential equation that satisfies certain initial conditions or boundary conditions. It is obtained by assigning specific values to the arbitrary constants in the general solution.
2. **Example:** If the initial condition  $y(t_0) = y_0$  is given, the particular solution is the one that satisfies this condition, resulting in a specific value for the constant  $C$ .

**Determining a Particular Solution:**

1. **Use of Initial or Boundary Conditions:** To determine a particular solution, we use the given initial conditions (for an initial value problem) or boundary conditions (for a boundary value problem). These conditions provide the necessary information to solve for the arbitrary constants in the general solution.
2. **Substitution and Solving:** The process typically involves substituting the initial or boundary conditions into the general solution and solving for the constants. This yields a solution that is tailored to the specific conditions of the problem.
3. **Example Process:** For a first-order ODE with the general solution  $y(t) = C \cdot g(t)$  and an initial condition  $y(t_0) = y_0$ , we would set  $y(t_0) = C \cdot g(t_0) = y_0$  and solve for  $C$  to obtain the particular solution.

In summary, the general solution of an ODE represents the family of all possible solutions, characterized by arbitrary constants. A particular solution is a specific member of this family that satisfies given initial or boundary conditions. Determining a particular solution involves substituting these conditions into the general solution and solving for the constants.

### ODE 的一般解:

1. **定义:** ODE 的通解包含微分方程的所有可能解。它包含任意常数（微分方程的每一阶都有一个常数），因为 ODE 通常有无限多个解。
2. **示例:** 对于一阶 ODE，通解可能表示为以下形式  $y(t) = C \cdot \text{克(吨)}$ ，在哪里  $C$  是任意常数并且  $\text{克(吨)}$  是由 ODE 确定的函数。

### ODE 的特解:

1. **定义:** 特解是微分方程满足一定初始条件或边界条件的特解。它是通过给通解中的任意常数赋予特定值而获得的。
2. **示例:** 如果初始条件  $y(t_0) = \text{和}_0$  给出后，特定的解就是满足该条件的解，从而产生常数的特定值  $C$ 。

### 确定特定的解决方案:

1. **初始或边界条件的使用:** 为了确定特定的解决方案，我们使用给定的初始条件（对于初始值问题）或边界条件（对于边界值问题）。这些条件提供了求解通解中的任意常数所需的信息。
2. **替换和求解:** 该过程通常涉及将初始条件或边界条件替换为通解并求解常数。这会产生针对问题的具体情况量身定制的解决方案。
3. **示例过程:** 对于具有通解的一阶 ODE  $y(t) = C \cdot \text{克(吨)}$  和一个初始条件  $y(t_0) = \text{和}_0$ ，我们会设置  $y(t_0) = C \cdot \text{克(吨)}_0 = \text{和}_0$  并解决  $C$  以获得特定的解决方案。

总之，ODE 的通解表示所有可能解的族，其特征为任意常数。特定解是该族中满足给定初始或边界条件的特定成员。  
↓

在笔记有写 - 待定系数法

### 待定系数法:

待定系数法是一种用于求解特定类型的非齐次线性常微分方程特解的方法。这种方法特别适用于方程右侧是多项式、指数函数、正弦或余弦函数等特定形式的函数时。

#### 1. 方法步骤:

- 首先确定对应的齐次方程的一般解。
- 根据非齐次方程右侧的特定函数形式，假设一个特解的形式。这个假设的形式通常与右侧函数的类型相关，但可能包含一些未知系数。
- 将假设的特解代入原方程，通过比较系数的方法求出这些未知系数的具体值。
- 找到的特解即为原非齐次方程的特解。

#### 2. 例子:

- 假设有一个线性非齐次常微分方程  $y'' + py' + qy = Ax^2 + Bx + C$ , 其中  $A, B, C$  为常数。
- 对于这个方程，我们可以假设一个特解的形式为  $y_p = ax^2 + bx + c$ , 其中  $a, b, c$  是待定的系数。
- 将  $y_p$  及其导数代入原方程，通过比较等号两边的系数来求解  $a, b, c$ 。

综上所述，在已知常微分方程的一般解的情况下，我们可以通过待定系数法来确定一个满足特定条件的特解。这个方法通过假设特解的形式并求解其中的未知系数来实现。

6. What can we say about the set of solutions of a homogeneous linear ODE? How does it help us find a general solution? Where do we get the fundamental system? (It is enough to know the basic answer with characteristic numbers, no need to go into details about higher multiplicity.)

关于齐次线性常微分方程的解集我们能说些什么？它如何帮助我们找到通用解决方案？我们从哪里获得基本系统？（知道特征数的基本答案就足够了，无需深入探讨更高重数的细节。）

#### Set of Solutions of a Homogeneous Linear ODE:

1. **Homogeneous Linear ODE:** A homogeneous linear ODE is an equation of the form  $L(y) = 0$ , where  $L$  is a linear differential operator. For example, a second-order homogeneous linear ODE can be written as  $a_2(t)y'' + a_1(t)y' + a_0(t)y = 0$ , where  $a_0(t)$ ,  $a_1(t)$ , and  $a_2(t)$  are coefficient functions.
2. **Properties:** The set of solutions of a homogeneous linear ODE has two important properties:
  - **Linearity:** If  $y_1(t)$  and  $y_2(t)$  are solutions, then any linear combination  $c_1y_1(t) + c_2y_2(t)$ , where  $c_1$  and  $c_2$  are constants, is also a solution.
  - **Superposition Principle:** The general solution can be constructed as a linear combination of independent solutions.

#### Fundamental System (Basis of Solutions):

1. **Definition:** A fundamental system (or basis) of solutions for a homogeneous linear ODE of order  $n$  is a set of  $n$  linearly independent solutions  $\{y_1(t), y_2(t), \dots, y_n(t)\}$ .
2. **Importance:** The fundamental system is crucial because it allows us to express the general solution as a linear combination of these independent solutions.

**General Solution:**

1. **Formulation:** The general solution of an  $n$ th-order homogeneous linear ODE is a linear combination of the  $n$  solutions in the fundamental system:  
$$y(t) = c_1 y_1(t) + c_2 y_2(t) + \dots + c_n y_n(t)$$
where  $c_1, c_2, \dots, c_n$  are arbitrary constants.
2. **Characteristic Numbers:** For constant coefficient linear ODEs, the fundamental system is often found using characteristic numbers (or eigenvalues). These are the roots of the characteristic equation associated with the ODE, and they determine the form of the independent solutions.

**Example:**

- For a second-order homogeneous linear ODE with constant coefficients, the characteristic equation might be  $r^2 + ar + b = 0$ . The roots of this equation provide the basis for constructing the fundamental system of solutions, from which the general solution is formed.

In summary, understanding the set of solutions of a homogeneous linear ODE and the concept of a fundamental system is essential for constructing the general solution. The general solution is a linear combination of linearly independent solutions, which can often be found using characteristic numbers for ODEs with constant coefficients.

### 齐次线性 ODE 的解集：

1. **齐次线性 ODE**: 齐次线性 ODE 是以下形式的方程  $Ly = 0$ , 在哪里  $L$  是线性微分算子。例如, 二阶齐次线性 ODE 可以写为  $A_2(t)y'' + A_1(t)y' + A_0(t)y = 0$ , 在哪里  $A_0(t), A_1(t)$ , 和  $A_2(t)$  是系数函数。
2. **属性**: 齐次线性 ODE 的解集有两个重要属性:
  - **线性度**: 如果  $y_1(t)$  和  $y_2(t)$  是解, 则任意线性组合  $C_1y_1(t) + C_2y_2(t)$ , 在哪里  $C_1$  和  $C_2$  是常数, 也是一个解。
  - **叠加原理**: 通解可以构造为独立解的线性组合。

### 基本系统 (解决方案的基础) :

1. **定义**: 齐次线性 ODE 解的基本系统 (或基础)  $n$  是一组  $n$  线性独立解  $\{y_1(t), y_2(t), \dots, y_n(t)\}$ 。
2. **重要性**: 基本系统至关重要, 因为它允许我们将通解表示为这些独立解的线性组合。

### 一般解决方案:

1. **公式**: 的通解  $n$  三阶齐次线性 ODE 是以下各项的线性组合  $n$  基本系统解决方案:
$$y(t) = C_1y_1(t) + C_2y_2(t) + \dots + C_ny_n(t)$$
在哪里  $C_1, C_2, \dots, C_n$  是任意常数。
2. **特征数**: 对于常系数线性 ODE, 通常使用特征数 (或特征值) 来找到基本系统。这些是与 ODE 相关的特征方程的根, 它们确定独立解的形式。

### 例子:

- 对于具有常数系数的二阶齐次线性 ODE, 特征方程可能是  $r^2 + \lambda r + \mu = 0$ 。该方程的根为构造基本解系提供了基础, 并由此形成通解。

总之, 理解齐次线性 ODE 的解集和基本系统的概念对于构造通解至关重要。通解是线性独立解的线性组合, 通常可以使用具有常数系数的 ODE 的特征数来找到。  
↓

见迭代法

7. What can we say about solutions of a general (non-homogeneous) linear ODE? How does it help us in solving such an equation?

(对于一般 (非齐次) 线性 ODE 的解我们能说些什么? 它如何帮助我们解这样的方程?)

见待定系数法的例题

### Solutions of a General Linear ODE:

1. **General Linear ODE:** A non-homogeneous linear ODE can be written in the form  $L(y) = g(t)$ , where  $L$  is a linear differential operator, and  $g(t)$  is a non-zero function known as the forcing term or source term.
2. **Properties:** The solutions of a non-homogeneous linear ODE have two key properties:
  - **Superposition Principle:** The solution of the non-homogeneous equation can be expressed as the sum of the general solution of the corresponding homogeneous equation and a particular solution of the non-homogeneous equation.
  - **Structure of Solutions:** The general solution of the non-homogeneous ODE is the sum of the general solution of the homogeneous equation and any particular solution of the non-homogeneous equation.

### Solving a General Linear ODE:

1. **Find the General Solution of the Homogeneous Equation:** First, solve the corresponding homogeneous equation  $L(y) = 0$ . The general solution of this equation involves finding a fundamental set of  $n$  linearly independent solutions, where  $n$  is the order of the ODE.
2. **Find a Particular Solution of the Non-Homogeneous Equation:** Next, find a particular solution to the non-homogeneous equation. This can be done using various methods such as the method of undetermined coefficients, variation of parameters, or integral transforms.
3. **Form the General Solution:** Combine the general solution of the homogeneous equation with the particular solution of the non-homogeneous equation to form the general solution of the non-homogeneous ODE.

### Example:

- For a non-homogeneous second-order linear ODE  $y'' + ay' + by = g(t)$ , solve the corresponding homogeneous equation  $y'' + ay' + by = 0$  to get the general solution of the homogeneous part. Then find a particular solution to  $y'' + ay' + by = g(t)$ . The sum of these two gives the general solution of the non-homogeneous ODE.

In summary, the general solution of a non-homogeneous linear ODE is composed of the general solution of the corresponding homogeneous equation plus a particular solution of the non-homogeneous equation. This structure is key to solving non-homogeneous linear ODEs, as it provides a systematic approach to finding a complete solution.

### 一般线性常微分方程的解：

1. **一般线性 ODE**: 非齐次线性 ODE 可以写成以下形式  $Ly = g(t)$ ，在哪里  $L$  是一个线性微分算子，并且  $g(t)$  是一个非零函数，称为强制项或源项。
2. **属性**: 非齐次线性 ODE 的解有两个关键属性：
  - **叠加原理**: 非齐次方程的解可以表示为相应齐次方程的通解与非齐次方程的特解之和。
  - **解的结构**: 非齐次常微分方程的通解是齐次方程的通解与非齐次方程的任意特解之和。

### 求解一般线性常微分方程：

1. **求齐次方程的通解**: 首先解对应的齐次方程  $Ly = 0$ 。该方程的通解涉及找到一组基本方程  $n$  线性独立解，其中  $n$  是 ODE 的阶数。
2. **求非齐次方程的特解**: 接下来，求非齐次方程的特解。这可以使用各种方法来完成，例如未定系数法、参数变化法或积分变换法。
3. **形成通解**: 将齐次方程的通解与非齐次方程的特解相结合，形成非齐次 ODE 的通解。

### 例子：

- 对于非齐次二阶线性 ODE  $y'' + p(t)y' + q(t)y = g(t)$ ，求解相应的齐次方程  $y'' + p(t)y' + q(t)y = 0$  得到齐次部分的通解。然后找到一个特定的解决方案  $y'' + p(t)y' + q(t)y = g(t)$ 。这两者之和给出了非齐次 ODE 的通解。

综上所述，非齐次线性常微分方程的通解由相应齐次方程的通解加上非齐次方程的特解组成。这种结构是求解非齐次线性 ODE 的关键，因为它提供了寻找完整解决方案的系统方法。

8. What can we say about the set solutions of a system of linear ODEs? How can we find a general solution? (Again, the simplest case of simple real eigenvalues is enough).

见待定系数法的三种情况与高阶线性常微分方程



ChatGPT

待定系数法是一种用于求解线性非齐次常微分方程特解的技术，特别适用于高阶线性常微分方程。在应用待定系数法时，根据非齐次项（即方程右侧）的形式，通常有三种不同的情况需要考虑：

1. 多项式型非齐次项：

- 当非齐次项  $g(t)$  是一个多项式时，待定系数法的特解通常也假设为多项式形式。例如，如果  $g(t)$  是二次多项式  $At^2 + Bt + C$ ，那么特解  $y_p(t)$  可以假设为  $at^2 + bt + c$ ，其中  $a, b, c$  是待定系数。

2. 指数型非齐次项：

- 当非齐次项是指数函数形式，如  $g(t) = Ae^{kt}$ ，特解  $y_p(t)$  通常假设为相同形式的指数函数  $Ce^{kt}$ ，其中  $C$  是待定系数。如果  $k$  是对应齐次方程的特征根，特解的形式需要进行相应的调整，例如乘以  $t$  或更高次的  $t$  的幂。

3. 正弦或余弦型非齐次项：

- 当非齐次项是正弦或余弦函数，如  $g(t) = A \cos(\omega t) + B \sin(\omega t)$ ，特解  $y_p(t)$  通常假设为  $C \cos(\omega t) + D \sin(\omega t)$ ，其中  $C, D$  是待定系数。与指数型情况类似，如果  $\omega$  对应于齐次方程的特征根，特解的形式需要调整。

在确定了特解的假设形式后，将其代入原方程，然后通过比较方程两边的系数来求解待定系数。这样就可以得到原非齐次方程的特解。最终，原方程的一般解是它的齐次解和这个特解的和。

待定系数法是解决特定类型的非齐次线性常微分方程的有效工具，特别是在非齐次项具有特定形式时。它依赖于对特解形式的正确假设和代数技巧来确定系数。

### **Set of Solutions of a System of Linear ODEs:**

1. **System of Linear ODEs:** Consider a system of linear ODEs represented in matrix form as  $\mathbf{Y}' = A\mathbf{Y}$ , where  $\mathbf{Y}$  is a vector of dependent variables,  $A$  is a matrix of coefficients, and  $\mathbf{Y}'$  denotes the derivative of  $\mathbf{Y}$  with respect to the independent variable (typically time).
2. **Linearity of Solutions:** The set of solutions of a linear system has the property of linearity, meaning if  $\mathbf{Y}_1$  and  $\mathbf{Y}_2$  are solutions, then any linear combination  $c_1\mathbf{Y}_1 + c_2\mathbf{Y}_2$  is also a solution.

### **Finding the General Solution (Simple Real Eigenvalues):**

1. **Eigenvalues and Eigenvectors:** Begin by computing the eigenvalues and eigenvectors of the coefficient matrix  $A$ . In the simplest case, assume  $A$  has  $n$  distinct real eigenvalues  $\lambda_1, \lambda_2, \dots, \lambda_n$ .
2. **Solutions for Individual Eigenvalues:** For each eigenvalue  $\lambda_i$ , find the corresponding eigenvector  $\mathbf{v}_i$ . The solution associated with  $\lambda_i$  is  $\mathbf{Y}_i(t) = e^{\lambda_i t} \mathbf{v}_i$ .
3. **General Solution:** The general solution of the system is a linear combination of these individual solutions:  
$$\mathbf{Y}(t) = c_1 e^{\lambda_1 t} \mathbf{v}_1 + c_2 e^{\lambda_2 t} \mathbf{v}_2 + \dots + c_n e^{\lambda_n t} \mathbf{v}_n$$
where  $c_1, c_2, \dots, c_n$  are arbitrary constants, typically determined by the initial conditions of the system.

### **Example:**

- For a system of two linear ODEs with a matrix  $A$  having two distinct real eigenvalues  $\lambda_1$  and  $\lambda_2$  and corresponding eigenvectors  $\mathbf{v}_1$  and  $\mathbf{v}_2$ , the general solution is  $\mathbf{Y}(t) = c_1 e^{\lambda_1 t} \mathbf{v}_1 + c_2 e^{\lambda_2 t} \mathbf{v}_2$ .

In summary, to find the general solution of a system of linear ODEs with distinct real eigenvalues, compute the eigenvalues and eigenvectors of the coefficient matrix, and construct the solution as a linear combination of the exponential solutions formed by each eigenvalue-eigenvector pair. The linearity of solutions is a fundamental property that guides the construction of the general solution.

### 线性常微分方程组的解集：

1. **线性 ODE 系统**: 考虑以矩阵形式表示的线性 ODE 系统  $\mathbf{y}' = A\mathbf{y}$ , 在哪里  $A$  是系数矩阵, 并且  $\mathbf{y}'$  表示导数  $\mathbf{y}$  关于自变量 (通常是时间)。
2. **解的线性度**: 线性系统的解集具有线性属性, 这意味着如果  $\mathbf{y}_1$  和  $\mathbf{y}_2$  是解, 则任意线性组合  $C_1\mathbf{y}_1 + C_2\mathbf{y}_2$  也是一个解决方案。

### 求通解 (简单实特征值) :

1. **特征值和特征向量**: 首先计算系数矩阵的特征值和特征向量  $A$ 。在最简单的情况下, 假设  $A$  有  $n$  个不同的实特征值  $\lambda_1, \lambda_2, \dots, \lambda_n$ 。
2. **单个特征值的解决方案**: 对于每个特征值  $\lambda_i$ , 求对应的特征向量  $\mathbf{v}_i$ 。相关解决方案  $\mathbf{y}(t) = C_i e^{\lambda_i t} \mathbf{v}_i$ 。
3. **通解**: 系统的通解是这些单独解的线性组合:  
$$\mathbf{y}(t) = C_1 e^{\lambda_1 t} \mathbf{v}_1 + C_2 e^{\lambda_2 t} \mathbf{v}_2 + \dots + C_n e^{\lambda_n t} \mathbf{v}_n$$
在哪里  $C_1, C_2, \dots, C_n$  是任意常数, 通常由系统的初始条件确定。

### 例子:

- 对于具有矩阵的两个线性 ODE 系统  $A$  具有两个不同的实特征值  $\lambda_1$  和  $\lambda_2$  和相应的特征向量  $\mathbf{v}_1$  和  $\mathbf{v}_2$ , 一般解为  $\mathbf{y}(t) = C_1 e^{\lambda_1 t} \mathbf{v}_1 + C_2 e^{\lambda_2 t} \mathbf{v}_2$ .

总之, 要找到具有不同实特征值的线性 ODE 系统的通解, 需要计算系数矩阵的特征值和特征向量, 并将解构造为由每个特征值-特征向量对形成的指数解的线性组合。解的线性是指导通解构造的基本属性。