数值分析大作业 1

杨锟 (2313878)　郭子墨 (2313835)

2026 年 1 月 4 日

1　实验背景

1.1　Runge 现象

1901 年，数学家 Carl Runge 发现了一个令人困惑的现象：对于函数 $f(x) = \frac{1}{1+25x^2}$ 在区间 $x \in [-1, 1]$ 上，如果选择 $n+1$ 个 等间距 的节点对区间进行均匀剖分，并构造其 Lagrange 插值多项式 $P_n(x)$。随着插值多项式阶次 n 的递增，插值误差 $E_n(x) = |f(x) - P_n(x)|$ 的行为表现出明显的两极分化：误差在区间中心 $x \approx 0$ 附近 递减 ，然而，误差在区间边缘 $x \approx \pm 1$ 附近却 递增，$P_n(x)$ 在端点附近出现剧烈的振荡。这种在高次多项式等距插值中，误差在区间边缘迅速发散的现象，被称为 *Runge 现象*。

1.2　Runge 现象的数学解释

我们对 *Runge* 现象给出一个通俗的解释：

Runge 现象的根本原因是我们在使用等距节点进行高次插值时，插值误差项 $E_n(x)$ 表达式中的两个关键因子在高阶时共同发散。

设 $P_n(x)$ 为在节点 $\{x_0, x_1, \ldots, x_n\}$ 上插值函数 $f(x)$ 的 n 次多项式。若 $f \in C^{n+1}[a, b]$，则对任意 $x \in [a, b]$，存在 $\xi = \xi(x) \in (a, b)$ 使得

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_n(x),$$

其中 $\omega_n(x) = \prod_{i=0}^{n} (x - x_i)$ 称为节点多项式。因此，最大误差的上界为

$$\|f - P_n\|_\infty \leq \frac{\max_{\xi \in [a,b]} |f^{(n+1)}(\xi)|}{(n+1)!} \cdot \max_{x \in [a,b]} |\omega_n(x)|.$$

Lebesgue 常数 Λ_n 定义为 Lagrange 基函数之和的最大值：

$$\Lambda_n = \max_{x \in [a,b]} \sum_{i=0}^{n} |l_i(x)|,$$

其中 $l_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$。误差实际上可以表示为与 Λ_n 相关的形式，因为 $\max |\omega_n(x)|$ 与 Λ_n 在本质上决定了插值运算的放大效应。

在区间 $[-1, 1]$ 上取等距节点 $x_k = -1 + \frac{2k}{n}$，$k = 0, \ldots, n$。此时 Lebesgue 常数以指数速度增长：

$$\Lambda_n \sim \frac{2^n}{n \ln n} \quad (n \to \infty),$$

更精确的下界为

$$\Lambda_n \geq \frac{2^n}{en\ln n}.$$

对于 Runge 函数 $f(x) = \frac{1}{1+25x^2}$，其在复平面有极点 $z = \pm i/5$，高阶导数 $|f^{(n+1)}(x)|$ 的增长大致为 $O((n+1)! \cdot c^n)$ 量级，其中 c 与极点距离相关。虽然高阶导数增长迅速，但仍无法抵消 Lebesgue 常数的 2^n 级指数增长，因此整体误差上界随 n 的增大而指数发散，并在端点附近表现出剧烈的震荡。

2 实验过程

2.1 实验内容

本实验对函数 $f(x) = \frac{1}{1+25x^2}$ 在区间 $x \in [-1, 1]$ 上，选择 $n+1$ 个等间距的节点对区间进行均匀剖分，并构造其 Lagrange 插值多项式 $P_n(x)$。我们取了 $n = 2, 4, 6, 8, 10, 12, 14, 16$ 进行实验，绘制插值函数图像，并比较误差函数 $E_n(x)$ 在 $x = -1, 0, 1$ 三点附近函数值随 n 递增的趋势变化。

之后我们展示了其他几个具有 *Runge* 现象的函数：$f_1(x) = 1/(1+100x^2)$, $f_2(x) = 1/(1+e^{-40x})$, $f_3(x) = \mathrm{sech}^2(20x)$, $f_4(x) = \tanh(30x)$。

2.2 实验原理

本实验的核心方法是构造 Lagrange 插值多项式 $P_n(x)$ 来逼近原函数 $f(x)$。给定 $n+1$ 个互异的插值节点 $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$，其中 $y_j = f(x_j)$，存在唯一一个次数不超过 n 的多项式 $P_n(x)$ 满足 $P_n(x_j) = y_j$。该多项式 $P_n(x)$ 由一组基本多项式 $L_j(x)$ 线性组合而成。Lagrange 基本多项式 $L_j(x)$ 定义为在节点 x_j 处取值为 1，而在所有其他节点 x_k（其中 $k \neq j$）处取值为 0 的 n 次多项式：

$$L_j(x) = \prod_{\substack{k=0 \\ k\neq j}}^{n} \frac{x - x_k}{x_j - x_k} \tag{1}$$

基于此，Lagrange 插值多项式 $P_n(x)$ 的主公式为 $P_n(x) = \sum_{j=0}^{n} y_j L_j(x)$。然而，为了提高高阶插值的数值稳定性和计算效率，本实验采用了重心 Lagrange 插值公式。该公式首先计算重心权重 w_j：

$$w_j = \frac{1}{\prod_{\substack{k=0 \\ k\neq j}}^{n}(x_j - x_k)} \tag{2}$$

然后，插值多项式 $P_n(x)$ 可以通过如下形式高效计算：

$$P_n(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} y_j}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}} \tag{3}$$

插值误差 $R_n(x) = f(x) - P_n(x)$ 的理论表达式为 $R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot \omega_{n+1}(x)$，其中

$$\omega_{n+1}(x) = \prod_{j=0}^{n}(x - x_j) \tag{4}$$

2.3 算法流程与实现

本实验采用 *python 程序* 进行实现，通过程序来研究具有 Runge 性质的函数的误差函数 $E_n(x)$ 在 $x = -1, 0, 1$ 三点附近函数值随 n 递增的趋势变化。主要程序段就是通过一个循环对所有预设的阶数 n 执行以下步骤：

1. **节点生成**：在 $[-1,1]$ 上生成 $n+1$ 个等距节点 x_j 及其对应的函数值 y_j。
2. **插值计算**：调用重心 Lagrange 插值函数，进行 Lagrange 插值。
3. **误差计算**：计算插值多项式与原函数的绝对误差 $E_n(x) = |P_n(x) - f(x)|$。
4. **结果可视化**：绘制并保存插值对比图（原函数与 $P_n(x)$ 的比较），以及分区域（左端点、中心、右端点）的绝对误差对数图，以直观展示误差的发散特性。

2.4 主要程序段

以下代码框展示了实验中用于实现 Lagrange 插值计算的核心函数：

```python
def lagrange_barycentric(x_nodes, y_nodes, x_eval):
    n = len(x_nodes)
    P = np.zeros_like(x_eval, dtype=float)
    weights = np.zeros(n)

    # 1. 计算重心权重 w_j
    for j in range(n):
        prod = 1.0
        for k in range(n):
            if k != j:
                prod *= (x_nodes[j] - x_nodes[k])
        weights[j] = 1.0 / prod if abs(prod) > 1e-20 else 0.0

    # 2. 对每个待评估点 x_eval 进行插值
    for i, x in enumerate(x_eval):
        # 处理 x 恰好是节点的情况
        if np.any(np.abs(x - x_nodes) < 1e-12):
            idx = np.argmin(np.abs(x - x_nodes))
            P[i] = y_nodes[idx]
            continue

        num = den = 0.0
        for j in range(n):
            diff = x - x_nodes[j]
            w = weights[j] / diff
            num += w * y_nodes[j]
            den += w

        # 应用重心公式 P_n(x) = num / den
        P[i] = num / den if abs(den) > 1e-20 else y_nodes[np.argmin(np.abs(x -
            x_nodes))]
    return P
```

Listing 1: 重心 Lagrange 插值函数 `lagrange_barycentric`

```python
# Runge 函数定义
def f_runge(x):
```

```
    return 1 / (1 + 25 * x**2)

y_true_runge = f_runge(x_plot)
degrees_all = [2, 4, 6, 8, 10, 12, 14, 16]
errors_dict_runge = {}

# 迭代不同阶数 n
for i, n in enumerate(degrees_all):
    x_nodes = np.linspace(-1, 1, n + 1)
    y_nodes = f_runge(x_nodes)

    # 调用核心插值函数
    y_interp = lagrange_barycentric(x_nodes, y_nodes, x_plot)
    error = np.abs(y_interp - y_true_runge)
    errors_dict_runge[n] = error

    # 绘图和保存（单个阶数的插值对比图）...
    plt.plot(x_plot, y_interp, label=f'n={n}')
    # ... plt.savefig(...)

# 绘制误差汇总图（以左端点误差分析为例）
intervals = [("left", [-1, -0.8], "左端点附近误差")] # ...

for prefix, (x_min, x_max), title in intervals:
    mask = (x_plot >= x_min) & (x_plot <= x_max)
    x_local = x_plot[mask]

    plt.figure(figsize=(14, 9))
    for i, n in enumerate(degrees_all):
        error_local = errors_dict_runge[n][mask]
        # 使用 semilogy 绘制对数误差图
        plt.semilogy(x_local, error_local + 1e-16, color=colors[i], label=f'n={n}')

    plt.title(title + '\n（Runge 现象误差变化趋势）')
    plt.ylabel('绝对误差（对数坐标）')
    # ... plt.savefig(...)
```

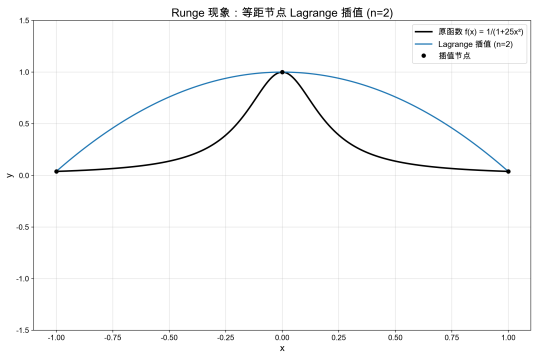
Listing 2: Runge 现象实验主循环（迭代计算与绘图）

3 实验结果与分析

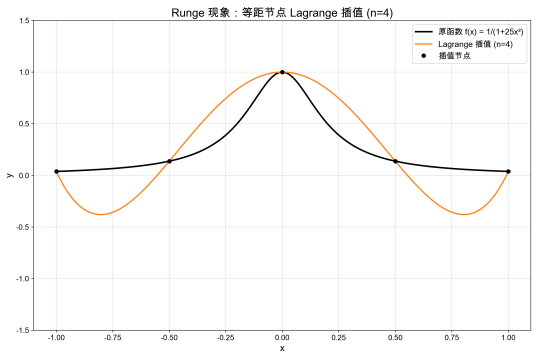
3.1 $f(x) = \frac{1}{1+25x^2}$ 的 Runge 现象

我们绘制了如下的不同 n 取值情况下的 Lagrange 插值函数曲线图、不同插值函数左、中、右侧区域插值误差分析图.

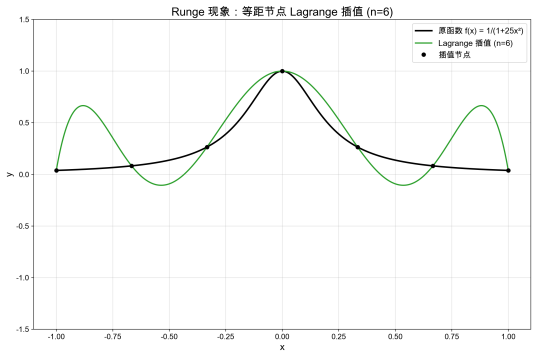
由以下误差图我们能很清楚看到：随着 n 的增大，误差 E_n 在区间中心 $x \approx 0$ 附近 递减，在区间边缘 $x \approx \pm 1$ 附近却 递增，即为 *Runge 现象*。



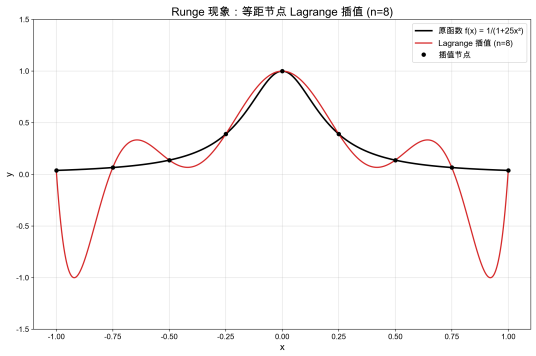
(a) $n = 2$



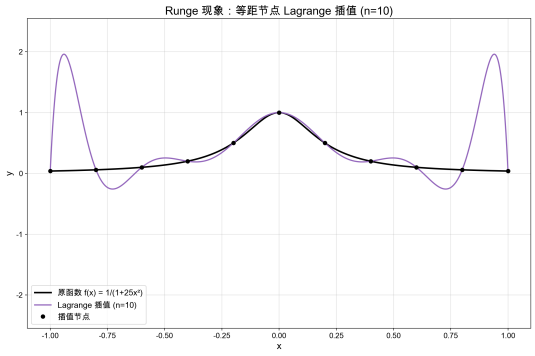
(b) $n = 4$



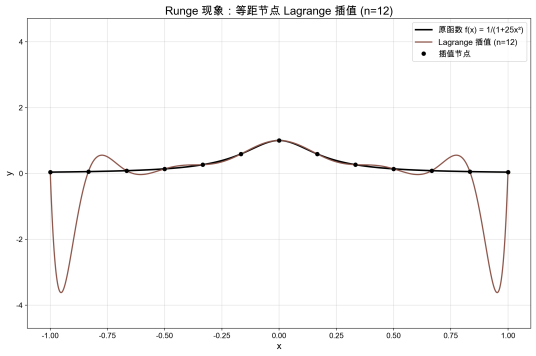
(c) $n = 6$



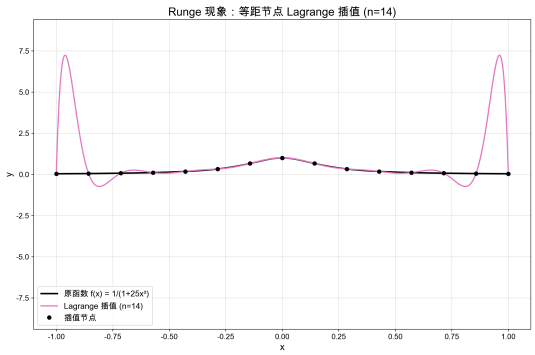
(d) $n = 8$



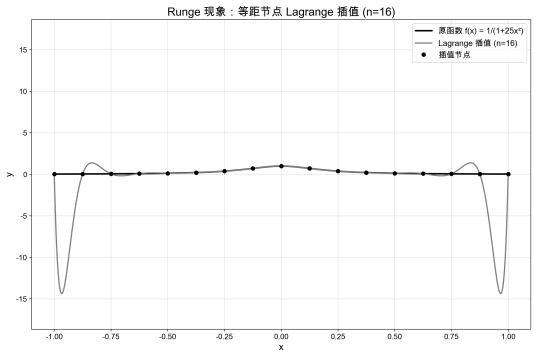
(e) $n = 10$



(f) $n = 12$



(g) $n = 14$



(h) $n = 16$

图 1: 不同插值点数 (n) 下的 Lagrange 插值函数图

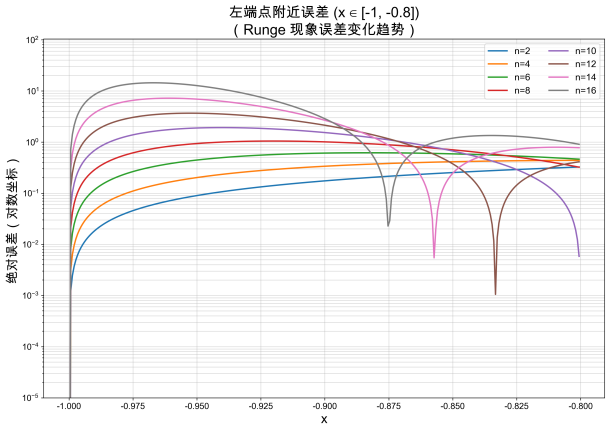


图 2: 左侧区域 ($x \in [-1, -0.8]$) 误差分布

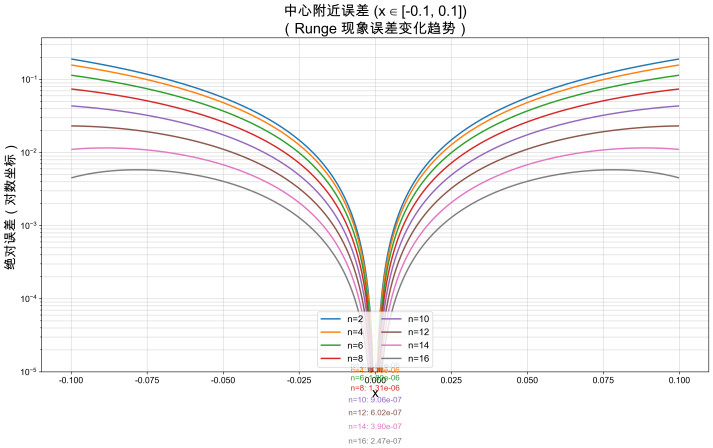


图 3: 中心区域 ($x \in [-0.1, 0.1]$) 误差分布

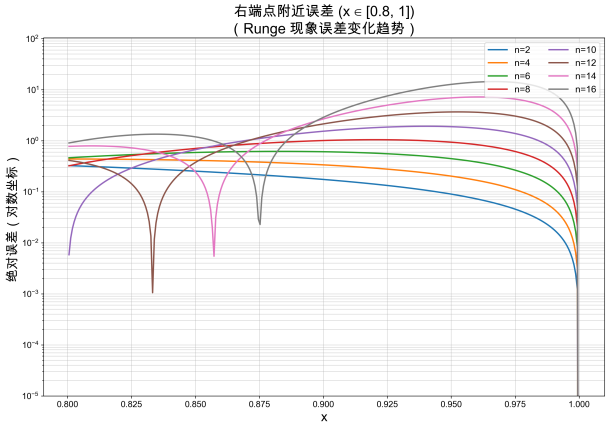
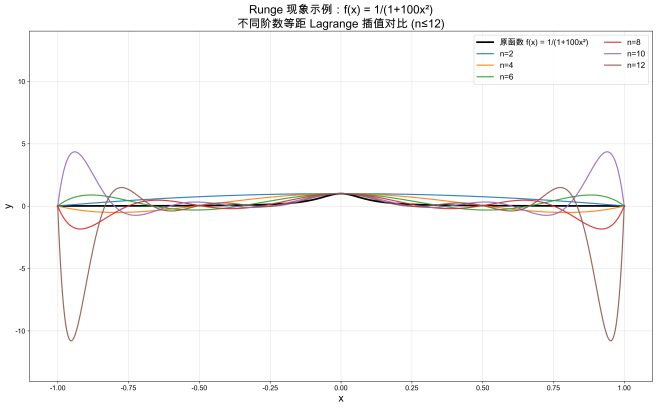


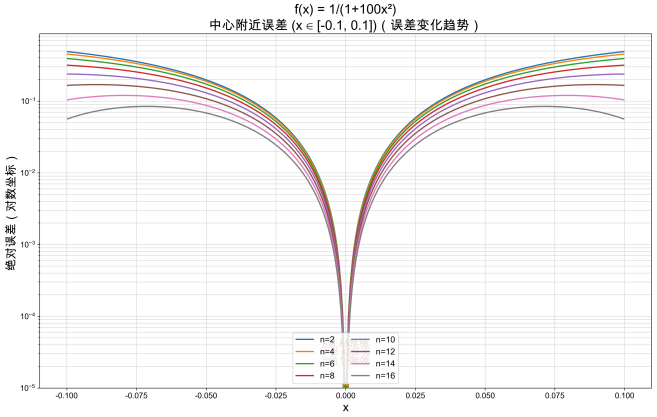
图 4: 右侧区域 ($x \in [0.5, 8]$) 误差分布

3.2 其他具有 Runge 现象的函数

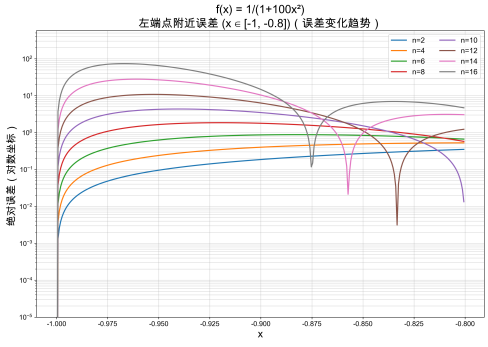
3.2.1 $f_1(x) = 1/(1+100x^2)$



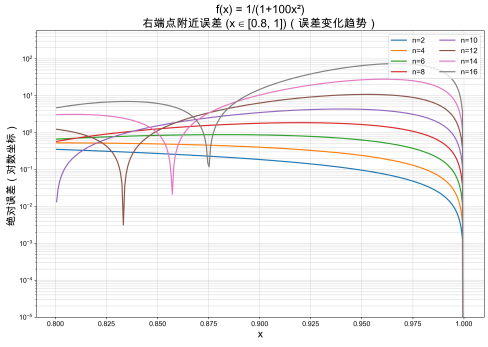
(a) $f_1(x) = 1/(1+100x^2)$ 插值函数汇总图



(b) 中心区域 ($x \in [-0.1, 0.1]$) 误差分布



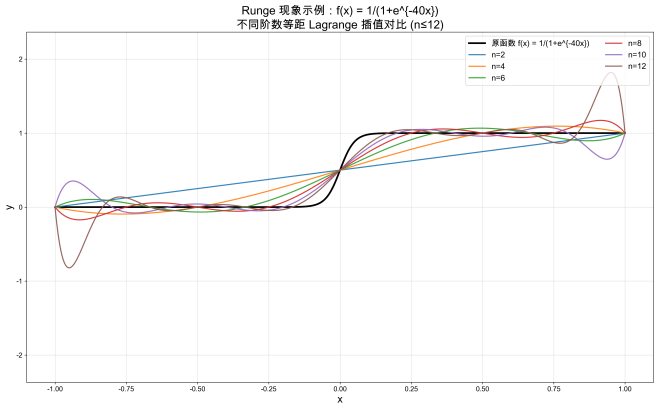
(c) 左侧区域 ($x \in [-1, -0.8]$) 误差分布



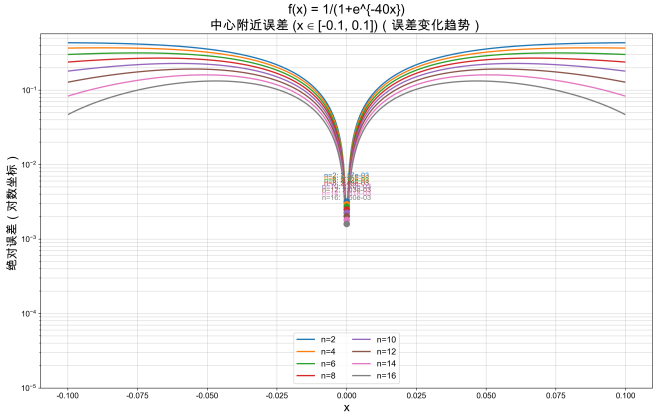
(d) 右侧区域 ($x \in [0.8, 1]$) 误差分布

图 5: $f_1(x) = 1/(1+100x^2)$ 的插值函数汇总图 + 不同插值函数中、左、右侧区域插值误差分析图

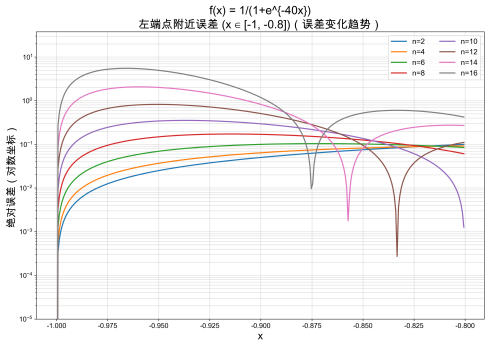
3.2.2 $f_2(x) = 1/(1+e^{-40x})$



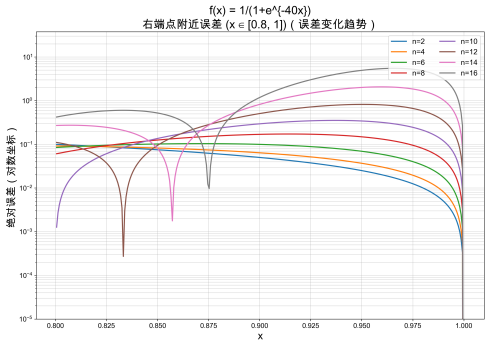
(a) $f_2(x) = 1/(1+e^{-40x})$ 插值函数汇总图



(b) 中心区域 ($x \in [-0.1, 0.1]$) 误差分布



(c) 左侧区域 ($x \in [-1, -0.8]$) 误差分布

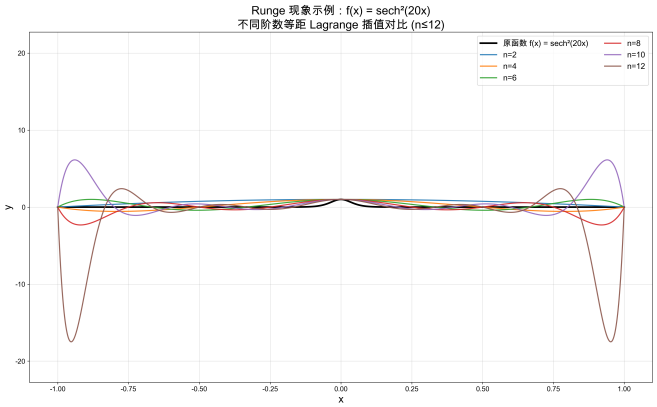


(d) 右侧区域 ($x \in [0.8, 1]$) 误差分布

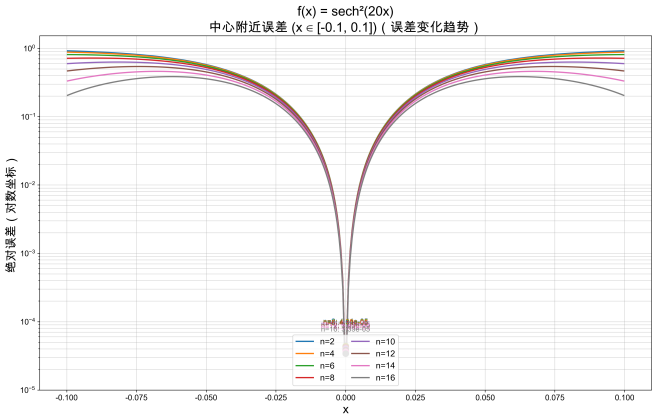
图 6: $f_2(x) = 1/(1+e^{-40x})$ 的插值函数汇总图 + 不同插值函数中、左、右侧区域插值误差分析图

3.2.3 $f_3(x) = \mathbf{sech}^2(20x)$

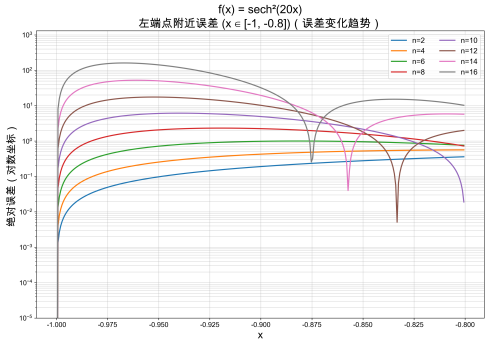
$$\operatorname{sech}^2(20x) = \left[\frac{1}{\cosh(20x)}\right]^2 = \frac{1}{\cosh^2(20x)} = \frac{4}{(e^{20x} + e^{-20x})^2}$$



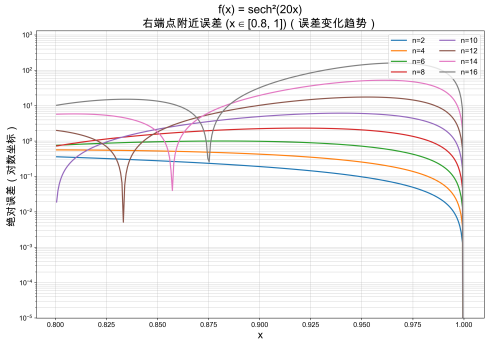
(a) $f_3(x) = \operatorname{sech}^2(20x)$ 插值函数汇总图



(b) 中心区域 ($x \in [-0.1, 0.1]$) 误差分布



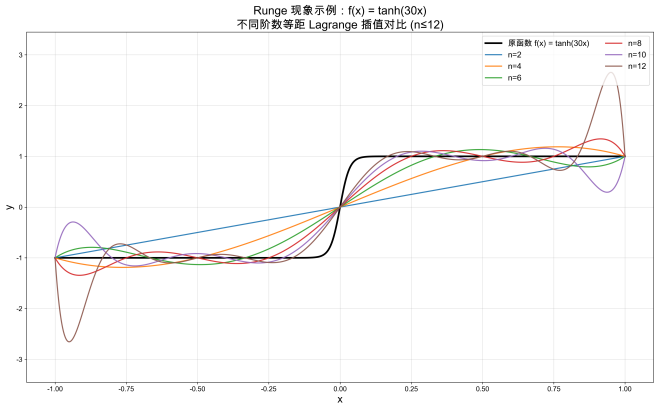
(c) 左侧区域 ($x \in [-1, -0.8]$) 误差分布



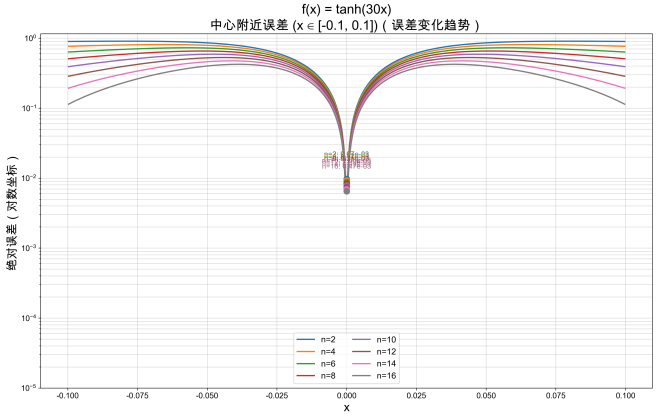
(d) 右侧区域 ($x \in [0.8, 1]$) 误差分布

图 7: $f_3(x) = \operatorname{sech}^2(20x)$ 的插值函数汇总图 + 不同插值函数中、左、右侧区域插值误差分析图

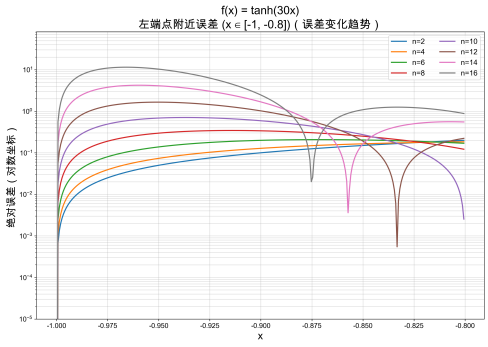
3.2.4 $f_4(x) = \tanh(30x)$



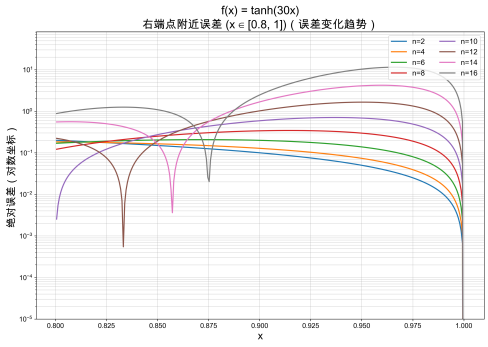
(a) $f_4(x) = \tanh(30x)$ 插值函数汇总图



(b) 中心区域 ($x \in [-0.1, 0.1]$) 误差分布



(c) 左侧区域 ($x \in [-1, -0.8]$) 误差分布



(d) 右侧区域 ($x \in [0.5, 0.8]$) 误差分布

图 8: $f_4(x) = \tanh(30x)$ 的插值函数汇总图 + 不同插值函数中、左、右侧区域插值误差分析图