第四次上机作业

1 问题叙述

在区间 $[-4,4]$ 上给出函数 $f(x)=e^x$ 的等距节点函数值表，用分段二次插值求 e^x 的近似值。要求：

- 等距节点函数值表的步长 h 自行选取，需满足所得到插值函数在 $[-4,4]$ 上的截断误差不超过 10^{-2}；
- 画出原函数 $f(x)$ 及插值函数在 $[-4,4]$ 上的图像。

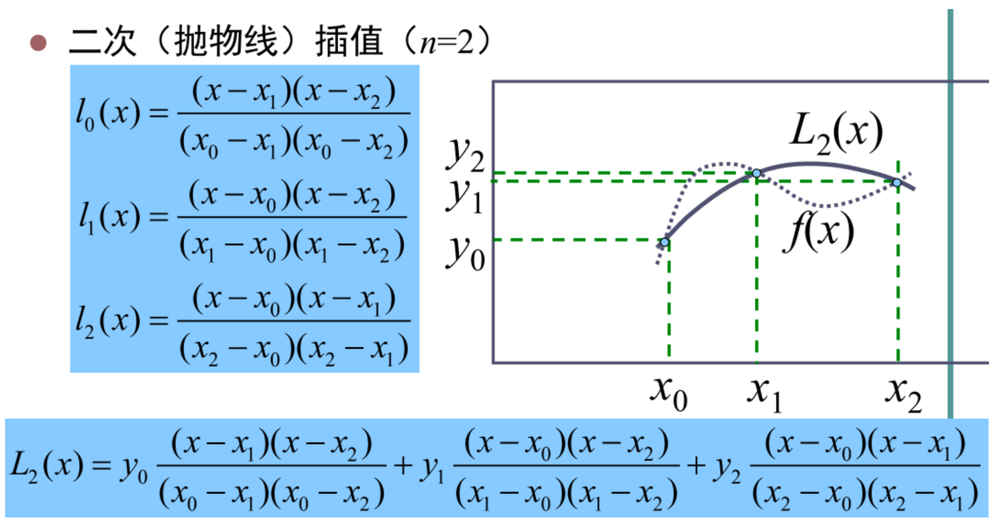
2 问题解答

2.1 节点函数值表

```
%% 分段节点函数表
point_num=58; % Number of sampling points
h=8/(point_num-1); % Sampling point spacing
x_value=-4:h:4;
y_value=exp(x_value);
```

程序运行后，x_value 数组中存放 x_i 的值，y_value 数组中对应存放 y_i 的值。其中 point_num 为取样点个数，h 为取样点间距。表格篇幅较大，且在报告中列意义不大，可运行程序后在工作区查看对应数据。

2.2 二次插值实现



当求插值点 x 的插值（即求 $f(x)$ 的近似值）时，可选取距点 x 最近的三个插值节点 x_{i-1}, x_i, x_{i+1}，在小区间 $[x_{i-1}, x_i, x_{i+1}]$ 上使用二次插值公式，称为分段抛物插值或分段二次插值。

选取距 x 最近的三个差值点，即等价于寻找离 x 最近的 x_i。我们之间使用 x 坐标绝对距离来确定。即在 $\frac{1}{2}(x_{i-1}+x_i)$ 到 $\frac{1}{2}(x_i+x_{i+1})$ 之间的 x 均可用 x_{i-1}, x_i, x_{i+1} 三个节点所确定的二次函数来表示。用此方法在 $[-4, 4]$ 遍历所有 x，即可获得一条分段的二次插值函数曲线。需要注意的是，一头一尾均使用最前/最后三个节点的二次方程来表示。

实现该过程的 Matlab 代码如下：

```
%% 分段二次插值
x_bond=zeros(1,length(x_value)-1); % Calculate boundary value
x_bond(1)=-4;
for i=2:length(x_value)-2
    x_bond(i)=0.5*(x_value(i)+x_value(i+1));
end
x_bond(end)=4;
count=1;
step=1e-6; % Plot sampling point spacing
x=-4:step:4;
for t=-4:step:4
    for j=1:length(x_bond)
        if t<x_bond(j)
            i=j-1;break;
        end
    end
    y0=y_value(i);y1=y_value(i+1);y2=y_value(i+2);
    x0=x_value(i);x1=x_value(i+1);x2=x_value(i+2);
    % Quadratic Lagrange interpolation
    y(count)=y0*(t-x1)*(t-x2)/(x0-x1)/(x0-x2)+y1*(t-x0)*(t-x2)/(x1-x0)/(x1-x2)+y2*(t-x0)*(t-x1)/(x2-x0)/(x2-x1);
    count=count+1;
end
```

2.3 估算截断误差

$R_n(x) = f(x) - P_n(x)$ 表示用 $P_n(x)$ 表示 $f(x)$ 时，在点 x 处产生的误差。

若 $f(x)$ 在区间 $[a,b]$ 上有直到 $n+1$ 阶导数 $f^{(n+1)}(x)$ 存在，$P_n(x)$ 为 $f(x)$ 在 $n+1$ 个节点 x_i 上的 n 次插值多项式，则 $\forall x \in [a,b]$ 有：

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x) \qquad \xi \in (a,b),\text{且依赖于}x$$

设用 x_{i-1}、x_i、x_{i+1} 获得二次插值曲线，并设 $x = x_i + hs$，则

$$|R_n(x)| = |\frac{e^{\xi}}{3!}(hs+h)(hs)(hs-h)| \le |\frac{\sqrt{3}e^4h^3}{27}| \le 0.01 \tag{1}$$

解得：

$$h \le 0.1419 \tag{2}$$

即：

$$point\_num \ge 58 \tag{3}$$

使用程序验证：

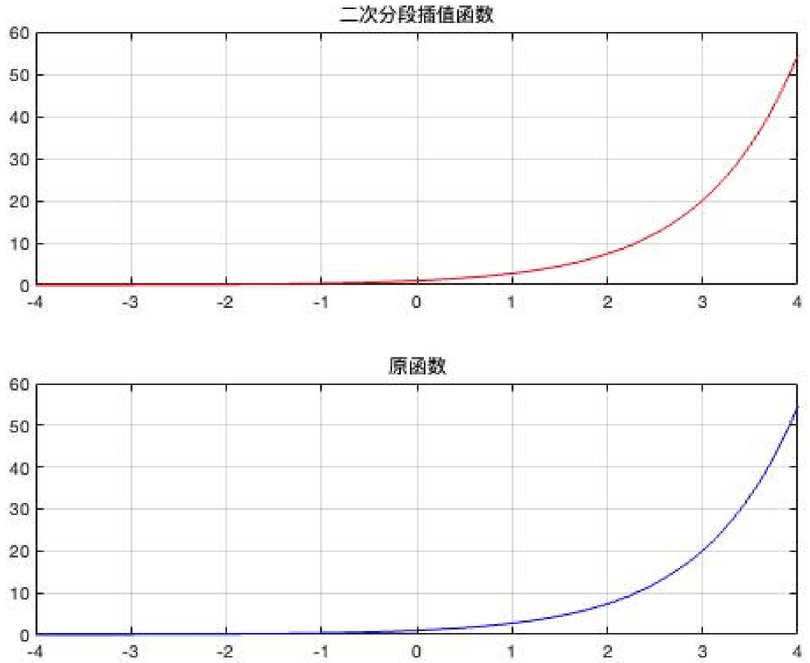
```
%% 估算截断误差
y_exp=exp(x);
Rn=abs(y-y_exp);
Rn_max=max(Rn); % Find the maximum truncation error
```

可以求得当 $point\_num = 58$ 时，R_n 的最大值为 0.008598，满足小于 0.01 的限制。

实际上，由于估算中的不等式放缩的等号并不能够取到，实际调试时发现只需要 $point\_num \ge 56$ 的时候就可以保证 R_n 小于 0.01。

2.4 图像绘制

图 1: 插值后函数和原函数对比



Matlab 代码:

```
%% 画图
subplot(2,1,1)
plot(x,y,'r');
grid on
syms x0;
title('二次分段插值函数')
subplot(2,1,2)
plot(x,y_exp,'b');
xlim([-4,4])
title('原函数')
grid on
```

从图像中完全分辨不出原函数和插值函数的区别，说明插值后与原函数符合的很好。

3 小结

通过本次上机实验，我初步掌握了二次插值求函数近似值的方法。实验中我使用理论分析与编程实践相结合的方法，用实际结果较为完美地检验了理论结论。实验中我发现两个函数非常接近，若画在一张图上效果很不好，因此我学习使用了 subplot() 函数，成功将两个函数分开画在一张图内。