第二章作业

问题叙述：

图 (a) 是载荷为线性分布的均质梁，如图 (b) 所示，该梁的挠度曲线方程是

$$y = \frac{w_0}{120EIL}(-x^5 + 2L^2x^3 - L^4x) \tag{1}$$

其中，$w_0 = 2.5kN/cm$，$E = 50000kN/cm^2$ 为弹性模量，$I = 30000cm^4$ 为截面惯性矩，$L = 600cm$ 为杆长。请计算出该杆的最大挠度值。(提示: 杆最大挠度值在满足 $\frac{dy}{dx} = 0$ 的 x 处达到) 要求分别用二分法、试位法、不动点迭代、Newton-Raphson 法和割线法求解并对各种方法进行比较。

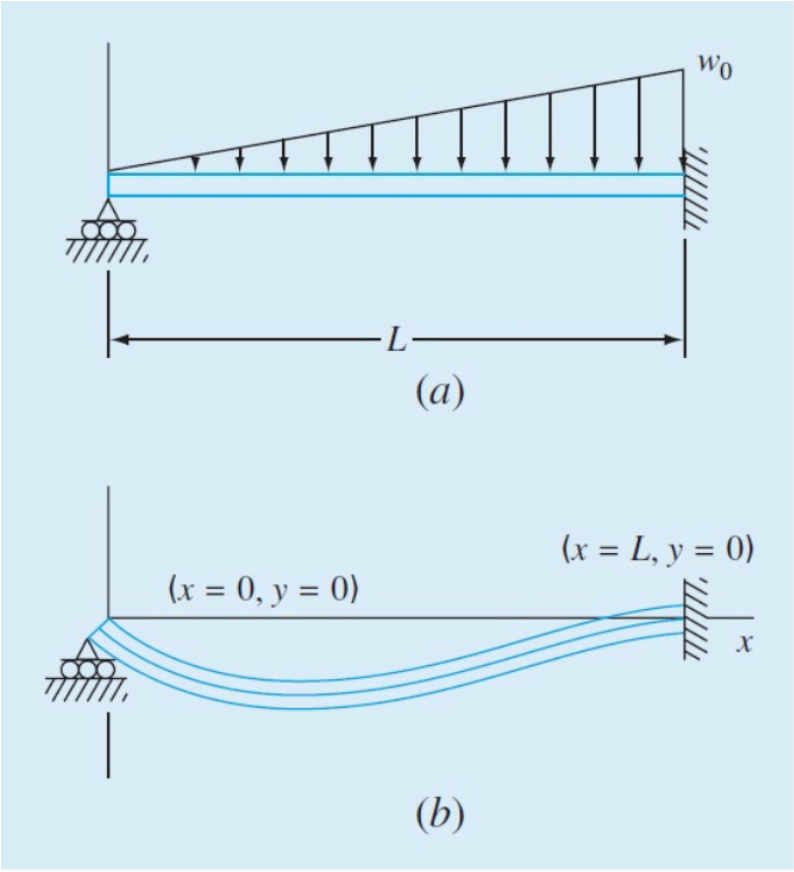

图 1: 问题叙述

问题分析：

原问题需要计算出该杆的最大挠度值，问题实则需要求非线性方程 $\frac{dy}{dx} = 0$ 的解。将解出的 x 值带入原挠度曲线方程，即可计算出最大挠度值。因此，本文主要讨论求解该非线性方程解的方法。

首先对方程求导：

$$f(x) = \frac{dy}{dx} = \frac{w_0}{120EIL}(-5x^4 + 6L^2x^2 - L^4) \tag{2}$$

则 $f(x) = 0$ 等价于：

$$5(\frac{x}{L})^4 - 6(\frac{x}{L})^2 + 1 = 0 \tag{3}$$

令 $(\frac{x}{L})^2 = t$，则等价于：

$$5t^2 - 6t + 1 = 0 \tag{4}$$

显然可以看出，该方程有两个解 $t = \frac{1}{5}$ 和 $t = 1$，带回原方程 $x = \frac{\sqrt{5}}{5}L$ 和 $x = L$，显然后一个解应舍去，因此原方程的解为 $x = \frac{\sqrt{5}}{5}L$。

下面用数值计算方法近似求解。定义：

- 如果有 x^* 使 $f(x^*) = 0$，则称 x^* 为方程 $f(x) = 0$ 的根，或称为函数 f(x) 的零点。
- 当 $f(x)$ 为多项式时，即方程为 $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 = 0(a_n \neq 0)$，则称 $f(x) = 0$ 为 n 次代数方程。当 $f(x)$ 包含指数函数或三角函数等特殊函数时，称 $f(x) = 0$ 为超越方程。
- 如果 $f(x)$ 可分为 $f(x) = (x - x^*)^m g(x)$，其中，$g(x^*) \neq 0$，m 为正整数，则称 x^* 为 $f(x) = 0$ 的 m 重根，当 $m = 1$ 时称 x^* 为 $f(x) = 0$ 的单根。

二分法：

基本思想：首先确定有根区间，将区间二等分，通过判断 $f(x)$ 的符号，逐步将有根区间缩小，直至有根区间足够地小，便可求出满足精度要求的近似根。

步骤：

- 猜测初始下界 x_l 和上界 x_u，使 $f(x_l)f(x_u) < 0$；
- 计算中间点 $x_r = \frac{x_l + x_u}{2}$
- 根据情况确定根所在的区间：如果 $f(x_l)f(x_r) < 0$，则根落在左边子区间，取 $x_u = x_r$，返回步骤 2；如果 $f(x_l)f(x_r) > 0$，则根落在右边子区间，取 $x_l = x_r$，返回步骤 2；如果 $f(x_l)f(x_r) = 0$，则 xr 为所要求的根，算法终止。
- 终止条件：x_r 即为所求根或当 ε_a 小于既定的终止条件 ε_s 或迭代次数达到一个上界时，计算终止。

近似误差总是大于真实误差，ε_a 小于 ε_s 时，真实误差一定也小于 ε_s，所以根可以达到要求的精度，算法可以终止。

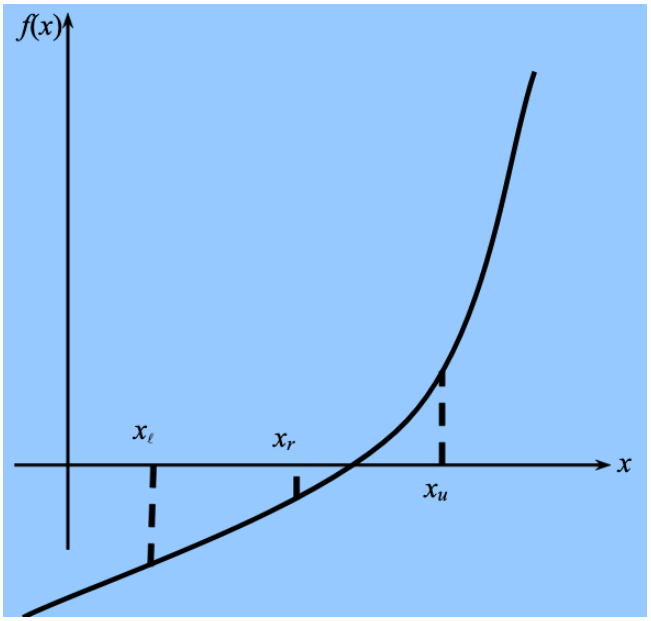

图 2: 二分法

Matlab 代码:

```
clear;
x_l = 0; % the initial value of the left bound of x is 250
x_r = 500; % the initial value of the right bound of x is 300
x_m = mean([x_l,x_r]); % x_m is the mean value of the two bounds
x_g = 0;
e_a = 5e-11; % the principal of 10 Significant Digits
real=600*sqrt(5)/5;
if f(x_l)*f(x_r)>0
    error('invalid input');
else
    while (abs(x_g-x_m)>e_a) % tolerance of accuracy
        if f(x_m)==0 % find the accurate root
            break;
        else
            if f(x_l)*f(x_m)<0;
            x_r = x_m; % update the new estimation to the interval
            else x_l = x_m;
```

```
18                 end
19             end
20         x_g = x_m; % current estimation
21         x_m = mean([x_r,x_l]);
22         end
23 end
24 e_t=abs((x_m-real)/real);
25
26 %%definition of function
27 function fun=f(x)
28 fun=5*(x/600)^4-6*(x/600)^2+1;
29 end
```

计算结果：

表 1: 二分法

x_m	与真值误差
2.683281572999761e+02	4.660544117029547e-15

由于二分法的特殊性，我们可以准确地知道循环次数，即

$$n = \lceil log_2(\frac{x_r - x_l}{E_{a,d}}) \rceil \tag{5}$$

本题中用了 44 次迭代达到了小数点后 10 位有效数字的精度。

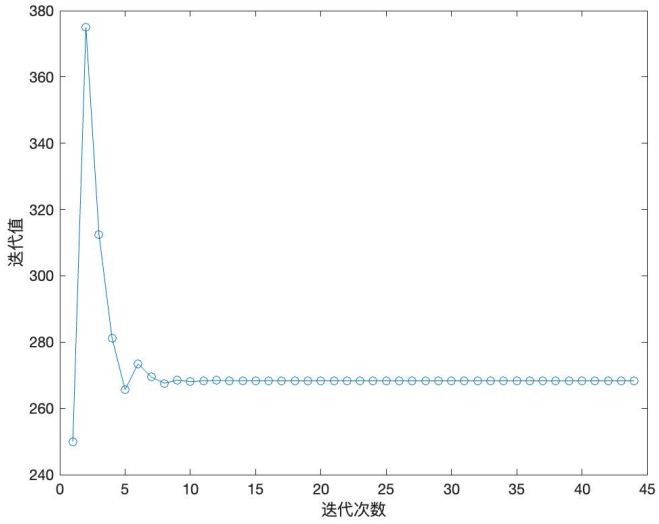

图 3: 二分法结果

试位法：

通过一条直线连接 $[x_l, f(x_l)]$ 和 $[x_u, f(x_u)]$，直线与 x 轴的交点作为新的根的估计值。

$$\frac{f(x_l)}{x_r - x_l} = \frac{f(x_u)}{x_r - x_u} \tag{6}$$

$$x_r = \frac{x_l f_i - x_u f_l}{f_u - f_l} \tag{7}$$

$$x_r = x_u - \frac{f_u(x_l - x_u)}{f_l - f_u} \tag{8}$$

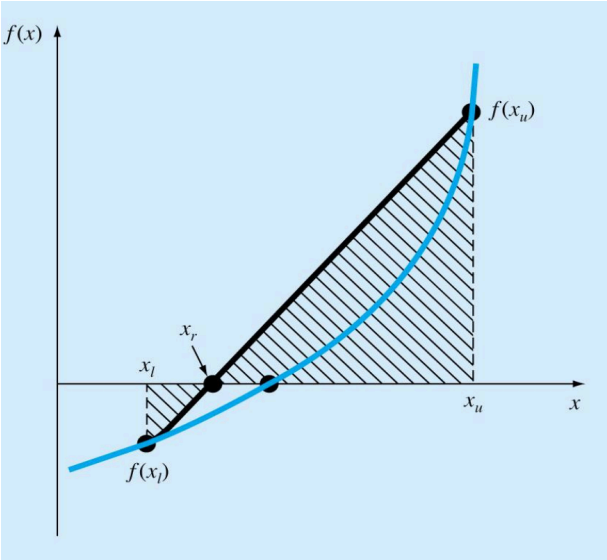

图 4: 试位法

Matlab 代码：

```
clc,clear;
x_l = 0; % the initial value of the left bound of x is 250
x_r = 500; % the initial value of the right bound of x is 300
x_c = x_r; % x_c is the cross point of the two bound-line and the x-axis
x_t = 0; % x_t is the target ROOT of the eqation
e_a = 5e-11; % the principal of 10 Significant Digits
i = 0; % iterations
real = 600*sqrt(5)/5; %real number
while (abs(x_t-x_c)>e_a)
x_t=x_c; % update the target value with the cross point
```

```
if f(x_t)==0
break;
else
x_c = x_r - f(x_r)*(x_l-x_r)/(f(x_l)-f(x_r)); % get the
    cross point
    if f(x_c)*f(x_r)<0
        x_l = x_c;
    else x_r = x_c;
    end
end
i = i+1;
end
e_t=abs((x_c-real)/real);
%%definition of function
function fun=f(x)
fun=5*(x/600)^4-6*(x/600)^2+1;
end
```

计算结果：

表 2: 试位法

x_m	与真值误差
2.683281572999748e+02	0

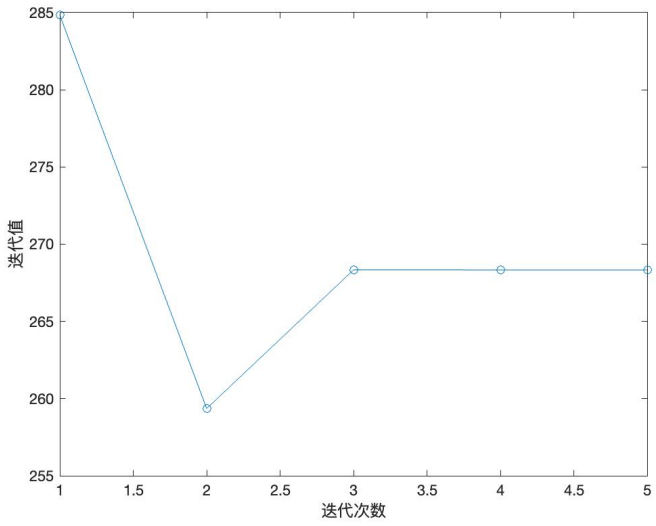

图 5: 试位法结果

相比二分法，可以看出经过 5 次迭代后，试位法直接收敛到了真值。同时，在此问题中函数在 [0,500] 区间内变化情况比较平缓，因此也不会出现“一个划界点保持不动”的现象。由此可见，对于此问题，试位法更加有效。

然而，试位法存在缺陷。如图可见，该函数在如图初值的条件下收敛的很慢。这是因为它违反了试位法的前提！在这个例子中，$f(x_l)$ 更接近 0，但根接近 x_u。因此，试位法的 while 循环条件中除了检查迭代近似误差外，还需要把根的估计值代入原方程中，检验是否接近 0。

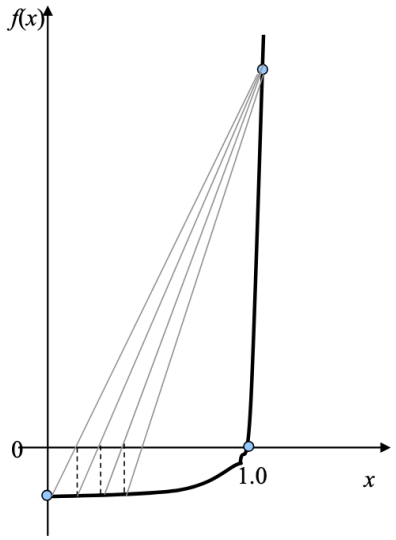

图 6: 试位法缺陷

不动点迭代：

重组方程 $f(x) = 0$ 为 $g(x) = x$，定义：

- 函数 $g(x)$ 的一个不动点 (Fixed point) 是指一个实数 P，满足 $P = g(P)$。
- 迭代 $p_{n+1} = g(p_n) \quad n = 0, 1, 2, \cdots$，称为不动点迭代。

定理：

- 设 g 是一个连续函数，且 $p_n{}_{n=0}^{\infty}$ 是由不动点迭代生成的序列。如果 $lim_{n\to p_n} = P$，则 P 是 $g(x)$ 的一个不动点。
- 设函数 $g \in C[a, b]$，如果对于所有 $x \in [a, b]$，映射 $y = g(x)$ 的范围满足 $y \in [a, b]$，则函数 g 在 $[a, b]$ 内有一个不动点。此外，设 $g'(x)$ 定义在 (a, b) 内，且对于所有 $x \in (a, b)$，存在正常数 $K < 1$，使得 $|g'(x)| \leq K < 1$，则函数 g 在 $[a, b]$ 内有唯一的不动点 P。

$$f(x) = 0 \tag{9}$$

可以改写为:

$$x = g(x) = 600\sqrt{\frac{5}{6}(\frac{x}{600})^4 + \frac{1}{6}} \tag{10}$$

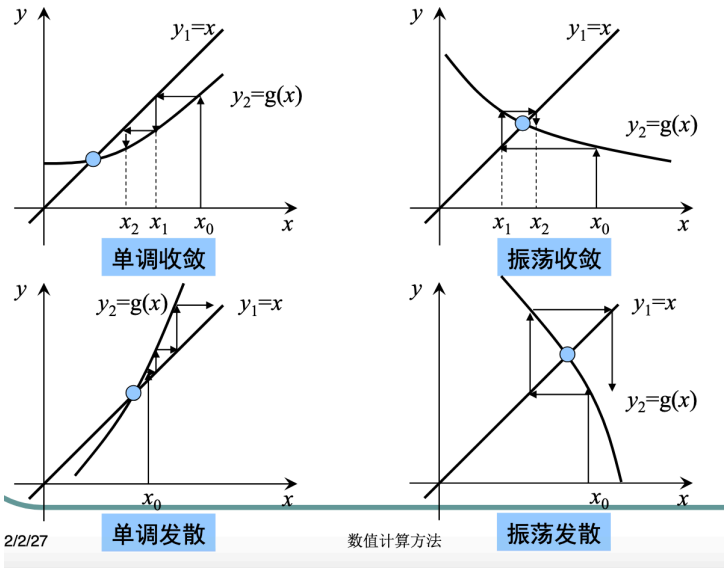

图 7: 不动点迭代

```
clc,clear
p0 = 250; % the initial value is 250
P(1) = p0;
max1 = 10000; % maxium iterations
real = 600*sqrt(5)/5; %real number
et(1) = abs((P(1)-real)/real);
tol = 5e-11; % the principal of 10 Significant Digits
for k=2:max1
    P(k)=f(P(k-1));%不动点迭代
    err=abs(P(k)-P(k-1));
    ea(k)=err/(abs(P(k))+eps);%计算相对误差
    p=P(k);
    et(k)=abs((P(k)-real)/real);
    x(k)=k;
    if ea(k)<tol,break;end
end
if k == max1
    disp('maximum number of iterations exceeded')
end
P=P';%记录每次迭代的值
```

```
21 ea=ea';
22 et=et';
23
24 function fun=f(x)
25 fun=600*sqrt(5/6*(x/600)^4+1/6);
26 end
```

计算结果：

表 3: 迭代法

x	与真值误差
2.683281572953767e+02	1.713618518957441e-11

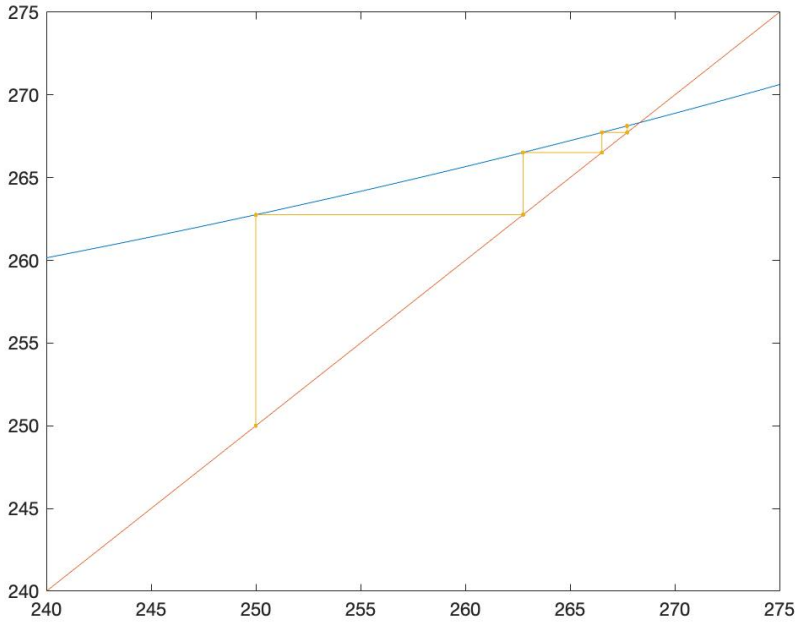

图 8: 不动点迭代计算结果

由图可见，选取的函数 $g(x)$ 满足单调收敛的条件，最终经过 21 次迭代收敛到计算结果，迭代速度较快，但相比于试位法误差较大。

在不动点迭代法中选取的函数对结果影响很大。由于此问题在第二次上机作业中已经详细地讨论过了，这里给出简略的结论。在零点附近 $g(x)$ 的导数的绝对值不能大于 1，否则会出现单调发散、振荡发散的情况。同时，选取的函数和初值对结果影响也很大，最好通过图像对函数、初值进行判断后再进行迭代。

Newton-Raphson 方法：

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i) \tag{11}$$

$f(x)$ 与 x 轴的交点处，$f(x) = 0$

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i) \tag{12}$$

$$x_{i+1} = x_i - \frac{f(x)}{f'(x)} \tag{13}$$

因此等价于寻找 $g(x) = x - \frac{f(x)}{f'(x)}$ 的不动点。

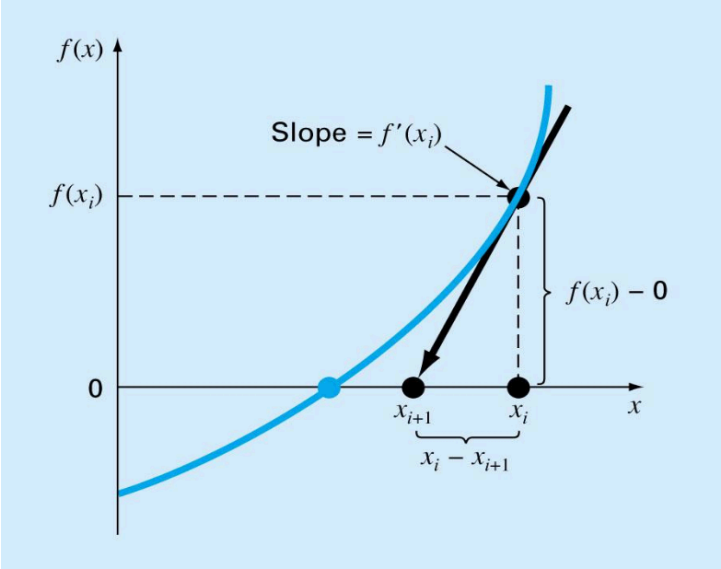

图 9: Newton-Raphson 方法

```
clc,clear;
x1 = 400;
x2 = x1-f(x1)/df(x1);
e = 5e-11; % Absolute tolerance of error
iterations = 0;
real = 600*sqrt(5)/5;
while (abs(x2-x1)>e)
    x1 = x2;
    x2 = x1-f(x1)/df(x1); % Newtown formula
    iterations = iterations+1;
end
e_t=abs((x2-real)/real);

%%definition of function
```

```
function fun=f(x)
fun=5*(x/600)^4-6*(x/600)^2+1;
end

function fun=df(x)
fun=20/600*(x/600)^3-12/600*(x/600);
end
```

计算结果：

表 4: Newton-Raphson 方法

x	与真值误差
2.683281572999748e+02	2.118429144104339e-16

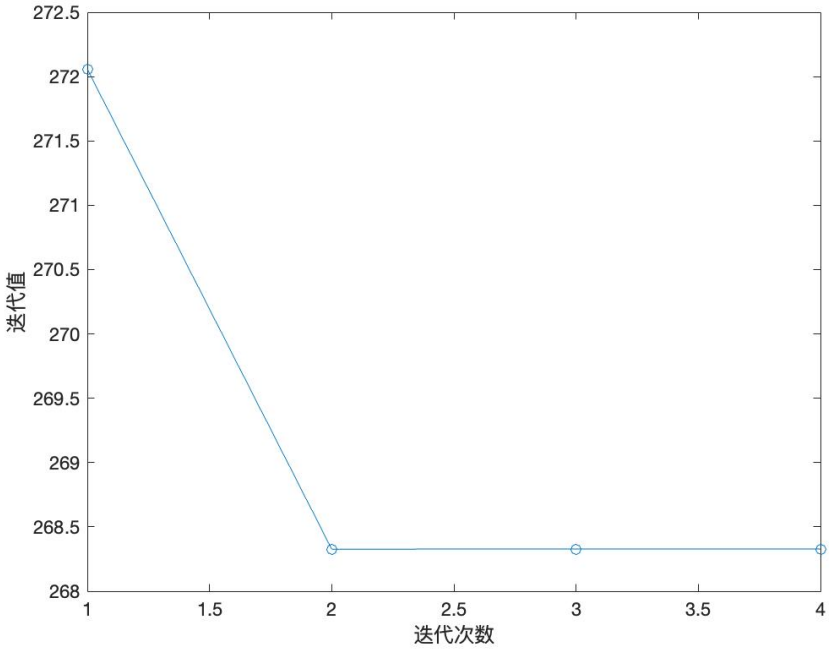

图 10: Newton-Raphson 方法计算结果

使用 Newton-Raphson 方法经过 4 次迭代后即收敛。该方法收敛速度很快，且与真值误差极小，对于本题是一种效果很好的方法。

然而，牛顿法存在一个缺陷，如果初值取在函数 $f'(x) = 0$ 附近，迭代次数会大大增加，且可能给出不感兴趣的解。例如，本题的函数曲线如下，从图像上可以明显看出若取的初值太大或太小（例如超过 500）会收敛到不感兴趣的解，若选取到极值点附近则会使迭代次数大大增加甚至死循环。我经过 matlab 实践发现确实

如此，例如将初值设为 500 会收敛到 600. 因此，在选取初值时，应当考虑是否在极值点附近，以保证能够收敛到正确结果。

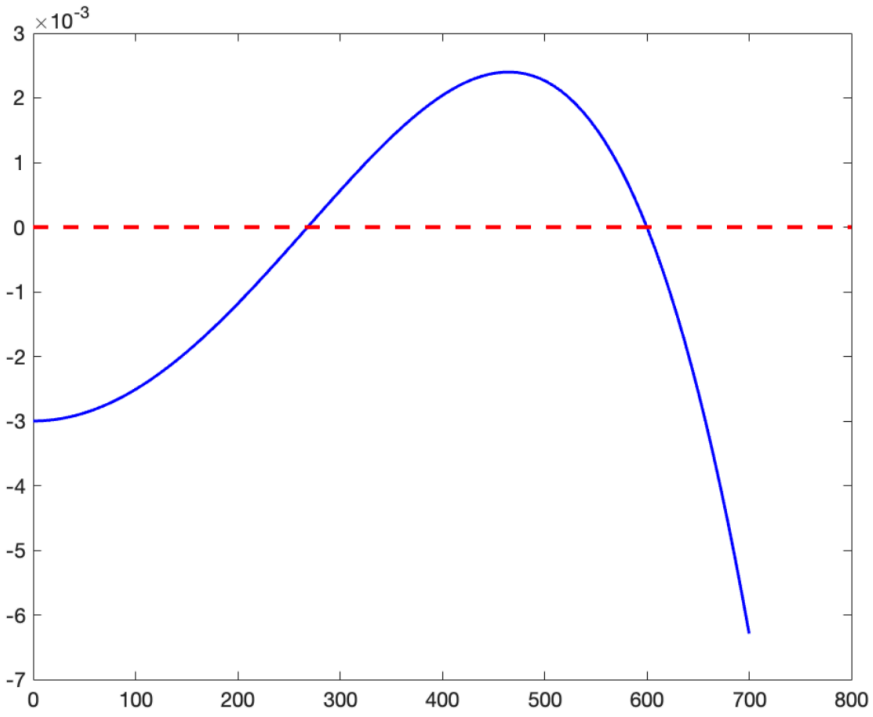

图 11: 挠度曲线一阶导

割线法：

$$\frac{1}{f'(x_i)} = \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} \tag{14}$$

$$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} \tag{15}$$

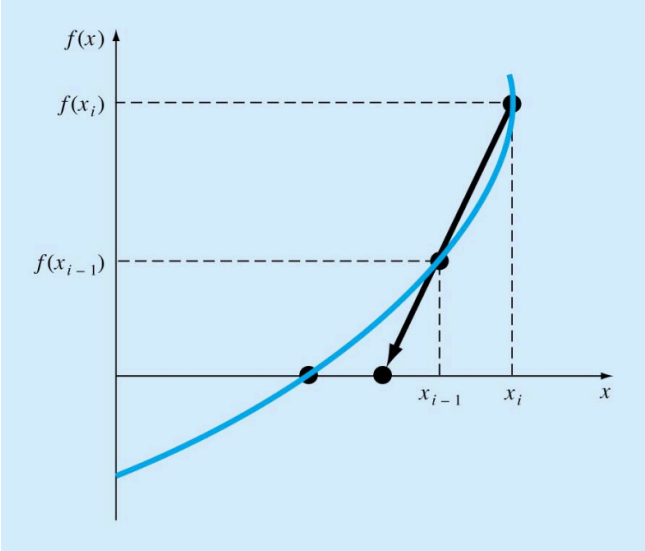

图 12: 割线法

```
clc,clear;
x1 = 250;
x2 = 300;
e = 5e-11; % Absolute tolerance of error
iterations = 0;
real = 600*sqrt(5)/5;
while (abs(x2-x1)>e)
    t = x2;
    x2 = x1-f(x1)*(x2-x1)/(f(x2)-f(x1)); % Newtown
        formula
    x1 = t;
    iterations = iterations+1;
end
e_t=abs((x2-real)/real);
%%definition of function
function fun=f(x)
fun=5*(x/600)^4-6*(x/600)^2+1;
end
```

表 5: 迭代法

x	与真值误差
2.683281572999748e+02	2.118429144104339e-16

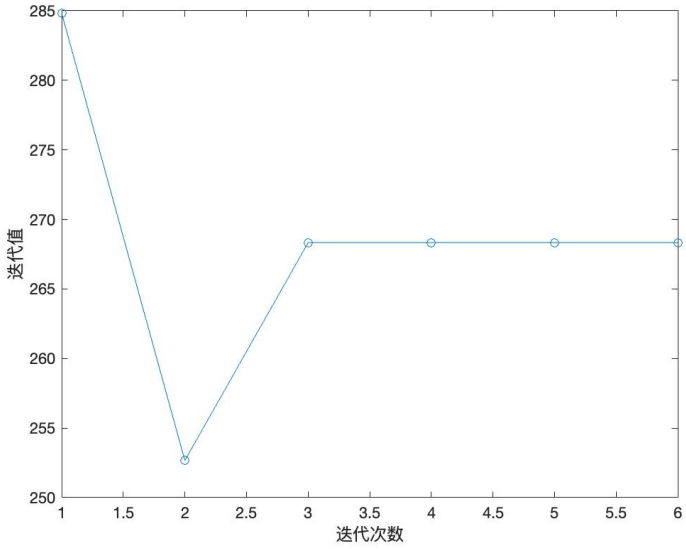

图 13: 割线法计算结果

类似于 Newton-Raphson 方法，割线法仅用了 6 次迭代便收敛到了一个误差极小的值。

小结：

本文从实际问题出发，已知桥梁挠度曲线方程，利用 5 种常见的非线性方程求解方法，对最大挠度值对应的 x 位置进行了计算。回到原问题，我们利用精度最高的 Newton-Raphson 方法给出的 x 的值带入原挠度曲线方程计算出最大挠度参考的值：

$$|y_{max}| = 0.5151900620159 52(cm) \tag{16}$$

表 6: 不同方法比较

方法	收敛速度	稳定性	精度	备注
二分法	慢	通常收敛	高	
试位法	快	通常收敛	高	
不动点迭代	较慢	有可能发散	高	需要选取合适的函数
Newton-Raphson 方法	快	有可能发散	高	需要求导
割线法	快	有可能发散	高	

本文在使用 5 种常见的非线性方程求解方法的基础上对各方法进行了发散性的讨论。对于本题而言，试位法、Newton-Raphson 方法、割线法在迭代速度和精度上占据了明显的优势，而剩下两种方法效果较差。然而，无论哪种方法，都有其相应的缺陷。例如试位法在本题非常有效但是对于某些特定的函数迭代次数将大

大增加。牛顿法在适当初值条件下收敛的快且精度高，但初值不适当时无法收敛到预期解。

因此，总的来看，不同的方法有不同的应用场景，其优劣程度没有绝对的定论，在不同函数和不同初始条件的情况下都会体现出不同的求解效果。因此，我们在求解非线性方程的时候，要秉承具体问题具体分析的原则，借助函数图像等手段初步推断出最优的解法，再对非线性方程组进行有效的解决。