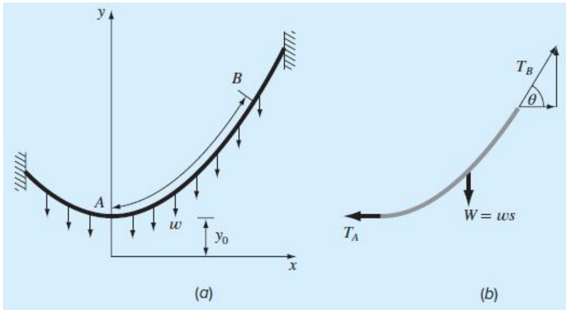
第二章作业

目录

一、问题叙述



如图所示，一段质量均匀分布的电缆线悬挂在两点之间，构成一段悬链，其满足如下微分方程：

$$\frac{d^2y}{dx^2}=\frac{w}{T_A}\sqrt{1+\left(\frac{dy}{dx}\right)^2}$$

其中，T_A 为悬链最低点的张力。若悬链最低点的高度为 y_0，则该微分方程的解析解为：

$$y=\frac{T_A}{w}\cosh\left(\frac{w}{T_A}x\right)+y_0-\frac{T_A}{w}\qquad\qquad \cosh(x)=\frac{e^x+e^{-x}}{2}$$

若 w = 10 和 y_0 = 5 时，悬链在 x = 50 处的高度为 y = 15，求 T_A。

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，并比较各方法的收敛速度。

二、问题分析

本题需先将方程进行转化，过程如下：

$$\boxed{y=\frac{T_A}{w}\cosh\left(\frac{w}{T_A}x\right)+y_0-\frac{T_A}{w}}$$

$$\Downarrow$$

$$\boxed{\text{代入 } w=10,\ y_0=5,\ x=50,\ y=15}$$

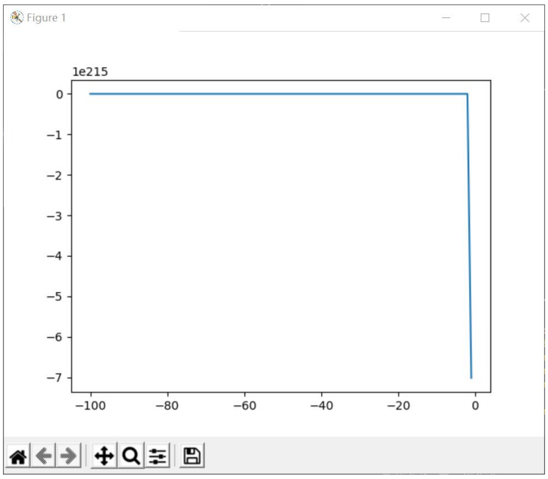
$$\Downarrow$$

$$\boxed{\begin{aligned} f(T_A)&=\frac{T_A}{10}\cosh\left(\frac{500}{T_A}\right)+5-\frac{T_A}{10}-15\\ &=\frac{T_A}{10}\times\left(\cosh\left(\frac{500}{T_A}\right)-1\right)-10\end{aligned}}$$

即需要求解关于 T_A 的非线性方程的根。

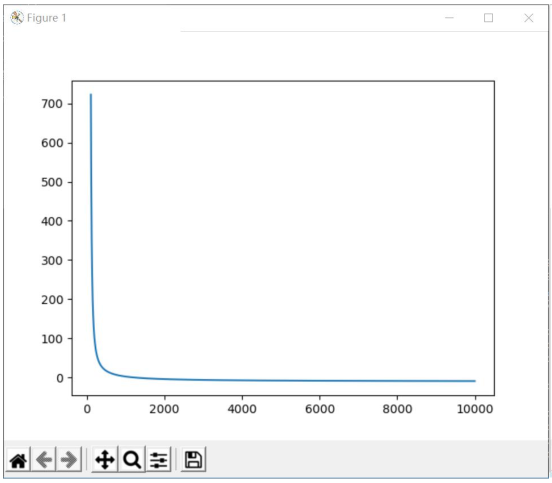
首先我们要确定方程根的区间，由于函数在 x = 0 处无意义，因此分正负半轴画出函数的图像。

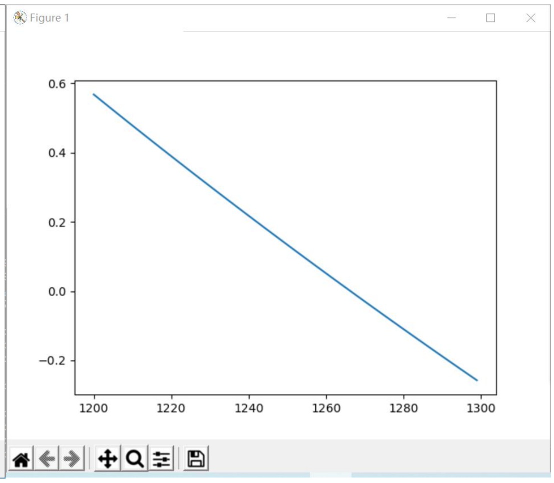
负半轴[-100, 0]图像如下：



注意到纵坐标的指数级别非常大，经计算可得 T_A 趋向于负无穷时，函数收敛于-10，因此函数与负半轴没有交点。

正半轴图像如下：





可以看到根的大致区间在[1260,1280]之间，但本题为了比较各种算法的收敛速度，应将初始区间扩大，所以设置为[100,4000]。同时由于函数在根的附近的导数变化不大，因此不考虑用导数的值来进行循环终止的判断，一律使用近似百分比误差和循环次数进行判断。将容限一致设为 1e-12，将循环次数上限设为 1e4 次。

```
x_l = 100
x_r = 4000
tolerance = 1e-12
n = 0
```

三、二分法

二分法将含有根的区间不断二等分，通过判断 f(x)的符号逐步缩小有根区间。

步骤如下：

1、确定上界和下界，是 f(x_l)*f(x_r) < 0

2、计算中间点

3、根据 f(x)的符号确定根的区间

4、终止条件：近似百分比相对误差<容限

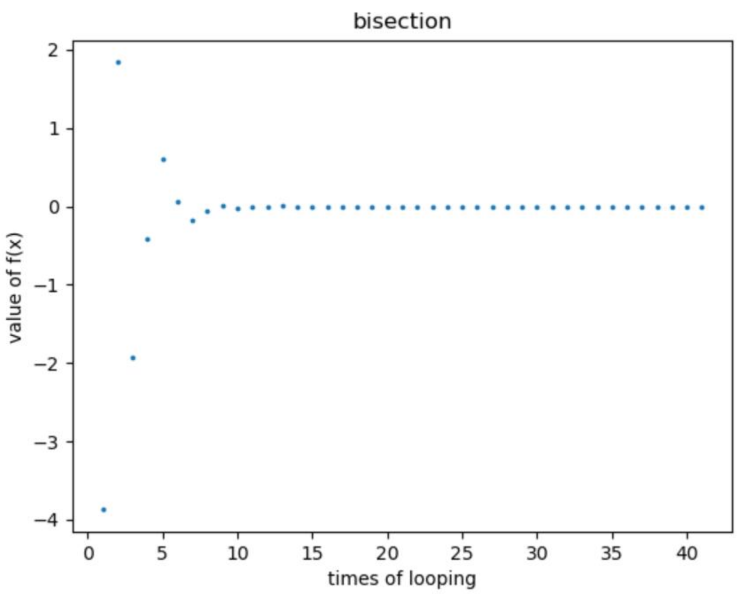
代码主体部分如下：

```
while (f(x_l)*f(x_r) != 0) and (n < 10000):
    x_mid = 0.5 * (x_l + x_r)
    if abs(x_mid-x_l)/x_mid < tolerance:
        break

    if f(x_l)*f(x_mid) < 0:
        x_r = x_mid
    else:
        x_l = x_mid

    n += 1
```

输出结果如下：

x = 1266.3243604000, n = 41

收敛情况如下：



四、试位法

试位法通过一条直线连接原来的两个点，并将直线与 x 轴的交点作为新的根估计值。

步骤如下：

1、确定上界和下界，是 f(x_l)*f(x_r) < 0

2、计算连线与 x 轴交点的值

3、根据 f(x)的符号确定根的区间

4、终止条件：近似百分比相对误差<容限

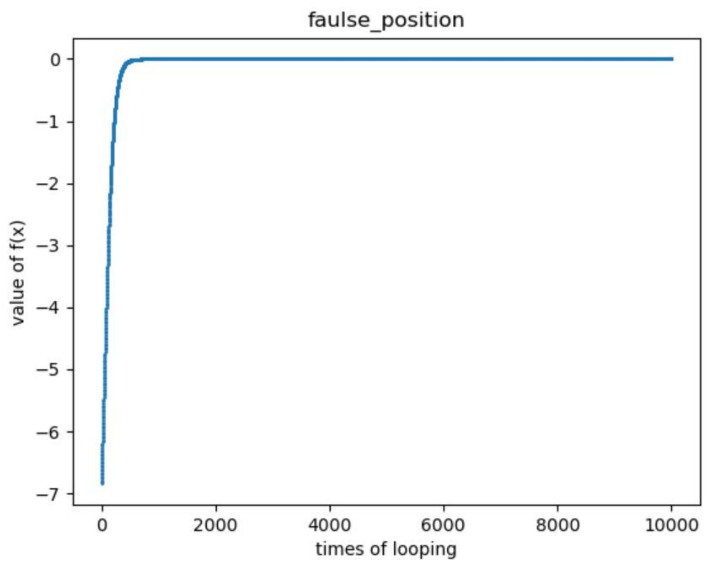
代码主体部分如下：

```
while (f(x_l)*f(x_r) != 0) and (n < 10000):
    x_new = x_r - f(x_r) * (x_l-x_r) / (f(x_l) - f(x_r))
    if f(x_r)*f(x_new) > 0:
        x_r = x_new
        if abs(x_l-x_new)/x_new < tolerance:
            break
    else:
        x_l = x_new
        if abs(x_r-x_new)/x_new < tolerance:
            break

    n += 1
```

输出结果如下：

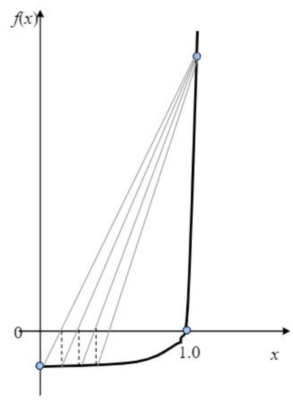
x = 1266.3243603999, n = 10000

收敛情况如下：



五、二分法与试位法的比较

二分法与试位法的总体思想是一致的，都是通过不断缩小含有根的区间来进行逼近，但从本题可以看到，根据图像选出的两个起始点较为匀称地分布在了根的两端，因此二分法的收敛速度较快，而本题的函数图像出现了类似右图的情况，导致试位法的一个划界点始终保持不动，使得收敛速度极慢，因此考虑用修正的试位法。



六、试位法的修正

修正的试位法需额外检测边界是否固定不变。

新增步骤如下：

1、使用计数器确定一个边界两次迭代后是否保持不变

2、若出现上述情况，则将停滞点的边界处的函数值变为原来一半

代码主体部分如下：

```
while (f(x_l)*f(x_r) != 0) and (n < 10000):
    x_new = x_r - f(x_r) * (x_l-x_r) / (f(x_l) - f(x_r))
    if f(x_r)*f(x_new) > 0:
        x_r = x_new
        if flag1 == 0:
            flag1 = 1
            flag2 = 0
        else:
            x_l = half(x_l)
        if abs(x_l-x_new)/x_new < tolerance:
            break
    else:
        x_l = x_new
        if flag2 == 0:
            flag2 = 1
            flag1 = 0
        else:
            x_r = half(x_r)
        if abs(x_r-x_new)/x_new < tolerance:
            break

    n += 1
```

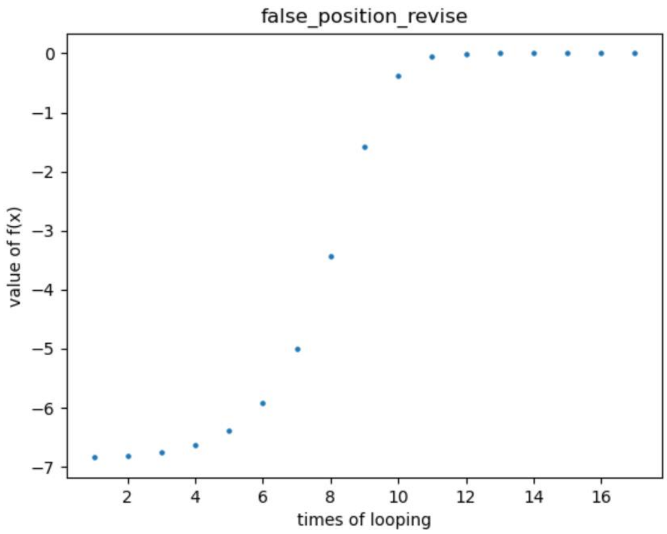
其中函数 half(x)用于寻找函数值为原来一半的点：

```
def half(x):
    if f(x) > 0:
        for i in range(int(x), 10000):
            if f(i) < 0.5*f(x):
                return i
    else:
        for i in range(int(x), 0):
            if f(i) > 0.5*f(x):
                return i
```

输出结果如下：

x = 1266.3243603999, n = 17

收敛情况如下：



循环仅仅迭代了 17 次就达到了终止条件，证明了修正后的试位法效果很好。

七、不动点迭代法

步骤如下：

1、选取 g(x)使得 f(x) = g(x) - x

2、选取初始点进行迭代

3、终止条件：近似百分比相对误差<容限

对于不动点法来说，首先要保证 g(x) = f(x) + x 的一阶导的绝对值小于 1，由正半轴图像可知，函数的一阶导从-∞变化到 0，因此通过循环找到一阶导大于-0.5 的点作为循环的起始点：

```
for i in range(100, 10000, 100):
    if (df(i) + 1) > -0.5:
        x_1 = i
        break
```

求得的 x_1 的值为 200，从而进行不动点迭代。

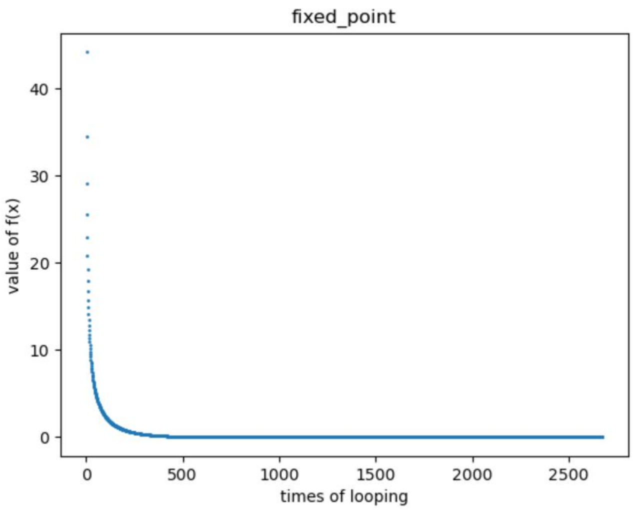
代码主体部分如下：

```
while n < 10000:
    x_2 = g(x_1)
    if abs(x_2-x_1)/x_2 < tolerance:
        break
    x_1 = x_2

    n += 1
```

输出结果如下：

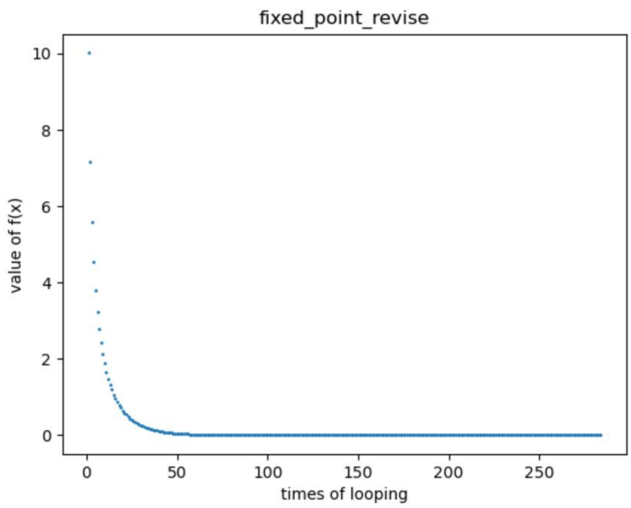
x = 1266.3243602447, n = 2674

收敛情况如下：



尽管迭代的循环次数没有达到上限，但是可以看到输出的结果并没有之前的方法精确，考虑到不同的迭代函数 g(x)的收敛情况不同，上述过程只是简单的选取 g(x) = f(x) + x，现在通过计算重新选取 g(x) = x*cosh(500/x) - 100，得到的结果如下：

x = 1266.3243603866, n = 284



求得的根更加精确，迭代次数也显著减小，所以 g(x)的选取对不动点迭代法来说至关重要。

八、Newton-Raphson 法

牛顿法通过作起始点的切线获得新的 x 值，然后不断更新并逼近真值。

步骤如下：

1、选取初值

2、计算初值处切线与 x 轴的交点

3、更新初值

4、终止条件：近似百分比相对误差<容限

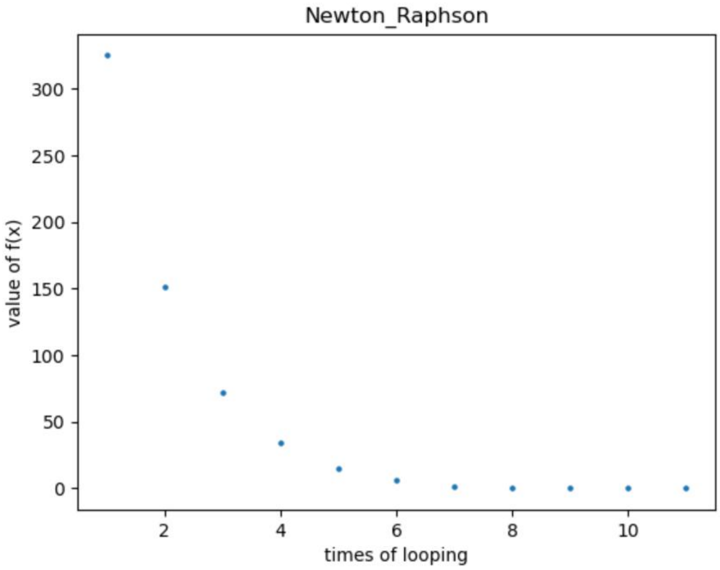
代码主体部分如下：

```
while n < 10000:
    x_2 = x_1 - f(x_1) / df(x_1)
    if abs(x_2-x_1)/x_2 < tolerance:
        break
    x_1 = x_2

    n += 1
```

输出结果如下：

x = 1266.3243603999, n = 11

收敛情况如下：



牛顿法的收敛速度非常快，因为它的收敛阶是二阶。同时，本题的函数未出现多个根、一阶导或二阶导为零、局部极值点震荡的情况，因此不会导致牛顿法产生缺陷。

九、割线法

割线法与试位法和牛顿法很类似，无需界定根的范围，在正割的基础上用两点斜率代替牛顿法的切线斜率。

步骤如下：

1、任意选取两个初值

2、计算两点连线与 x 轴的交点

3、根据 f(x)的符号更新两点的值

4、终止条件：近似百分比相对误差<容限

由于无需根据根的值确定初始值，因此选择 x_l = 100, x_r = 300。

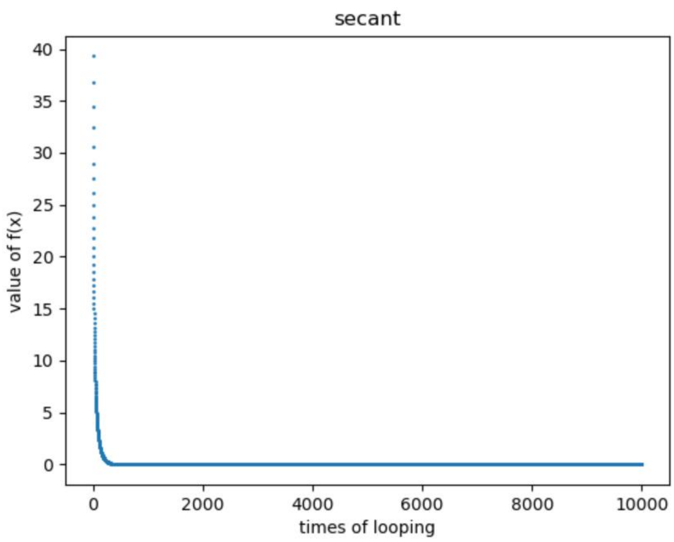
代码主体部分如下：

```
while (abs((x_l-x_r)/x_l)>tolerance) and (n < 10000):
    x_new = x_l - f(x_l) * (x_l-x_r) / (f(x_l)-f(x_r))
    if f(x_l)*f(x_r) > 0:
        if (abs(x_l)-abs(x_new)) < (abs(x_r)-abs(x_new)):
            x_r = x_new
        else:
            x_l = x_new
    else:
        if f(x_new)*f(x_l) > 0:
            x_r = x_new
        else:
            x_l = x_new

    n += 1
```

输出结果如下：

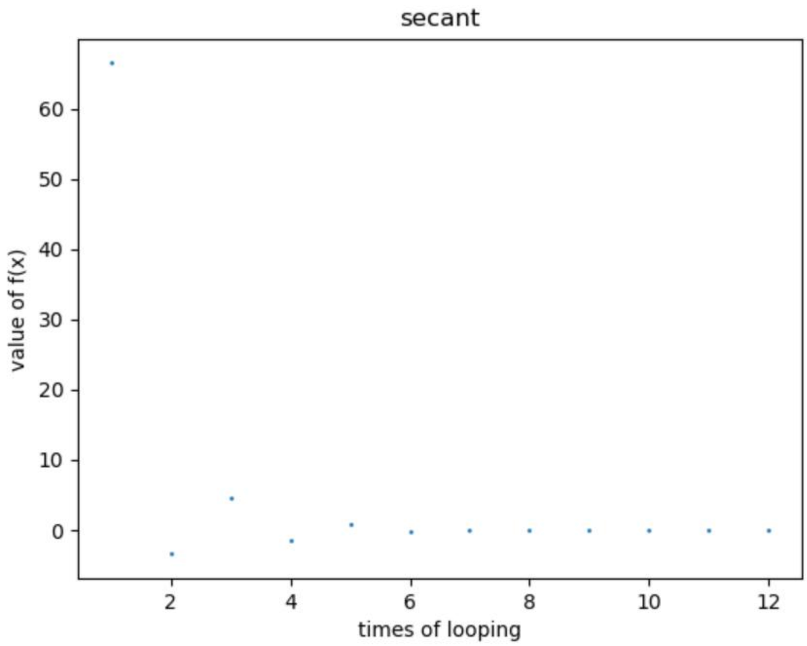
x = 1266.3243603999, n = 10000

收敛情况如下：



割线法出现了循环次数达到上限的情况，考虑到可能是初值选取的问题，反复试验后发现 x_l = 2000, x_r = 3000 时，收敛速度非常快，且精度很高：

x = 1266.3243603999, n = 12



十、割线法的修正

由于初值难以确定，考虑用独立变量的微小扰动进行优化：

```
delta = 0.01
x_l = 100
x_r = x_l * (1+delta)
```

这样的话循环终止条件无法用容限来估计，因此通过循环次数 n 来决定，发现设置 n 的上限为 100 时即可达到之前方法的精度：

x = 1266.3243603999, n = 100

十一、总结

本题总共通过五种基本方法和三个修正进行计算。

对于二分法和试位法来说，首先需要确定根的范围，这对于二分法的效率有直接的影响，而试位法会受函数性质的影响，因此都过判断边界点是否保持不变来进行修正。

不动点迭代则受 g(x)选取的影响，要使其一阶导的绝对值小于 1，收敛速度与二分法和试位法差不多。

牛顿法的收敛速度是最快的，但需要求原函数的一阶导，增加了编程的难度，也可能遇到发散、漏根等情况，本题的函数变化趋势相对简单，因此用牛顿法效果很好。

割线法的稳定性也受初值的影响，因此利用微小变量使得初值的数量减少为1，收敛速度大大增加，高于二分法、试位法、不动点法，低于牛顿法，但割线法可能会遇到不收敛的情况，因此需要反复尝试初值的取法。