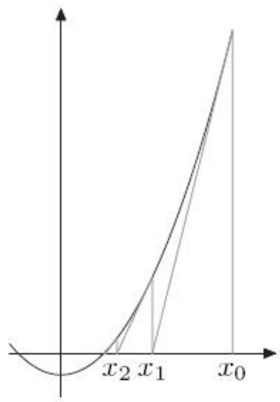
Exp 4 数值计算方法

牛顿迭代法



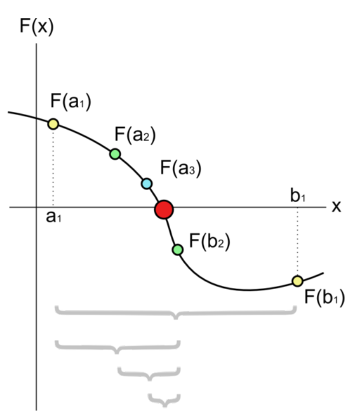
首先，选择一个接近函数$f(x)$零点的x_0，计算相应的$f(x_0)$和切线斜率$f'(x_0)$（这里f'表示函数f的导数）。然后我们计算穿过点$(x_0, f(x_0))$并且斜率为$f'(x_0)$的直线和x轴的交点的x坐标，也就是求如下方程的解：

$$0 = (x - x_0) \cdot f'(x_0) + f(x_0)$$

我们将新求得的点的x坐标命名为x_1，通常x_1会比x_0更接近方程$f(x) = 0$的解。因此我们现在可以利用x_1开始下一轮迭代。迭代公式可化简为如下所示：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

二分法

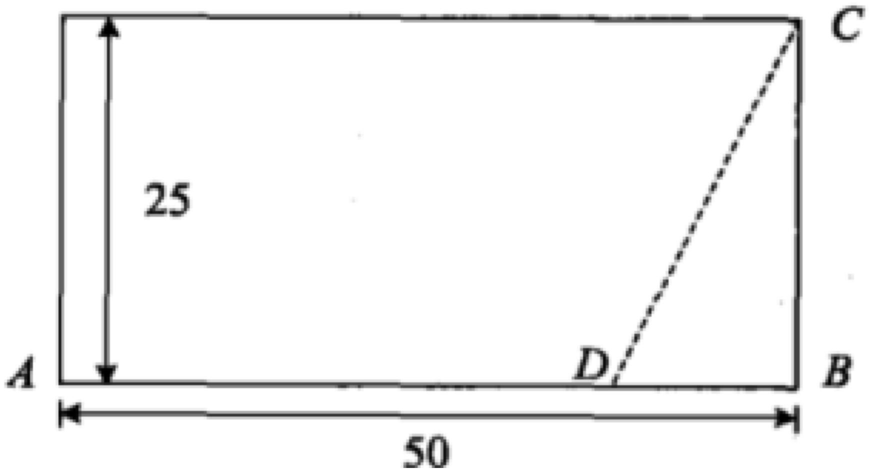


若要求已知连续函数$f(x) = 0$的根(x的解)，则:

1. 先找出一个区间$[a, b]$，使得$f(a)$与$f(b)$异号。根据介值定理，这个区间内一定包含着方程式的根
2. 求该区间的中点$m = \frac{a+b}{2}$，并找出$f(m)$的值

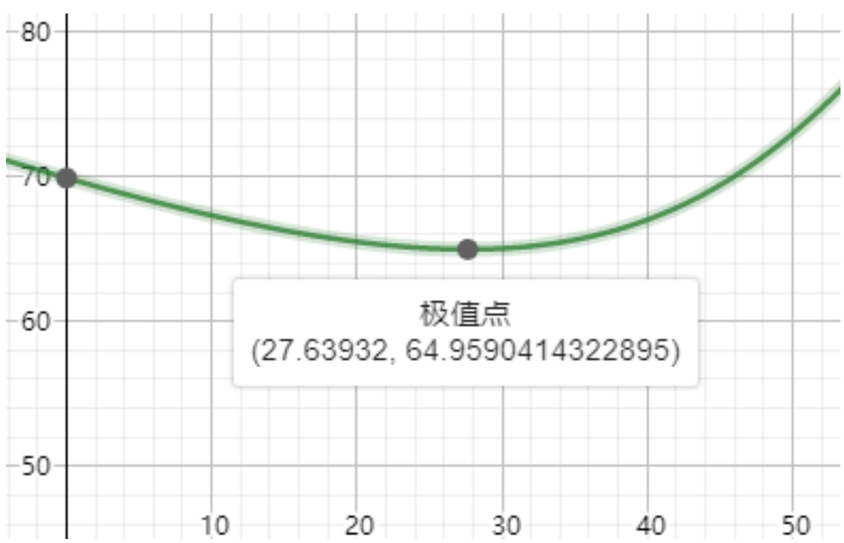
3. 若$f(m)$与$f(a)$正负号相同则取$[m, b]$为新的区间, 否则取$[a, m]$
4. 重复第2和第3步至理想精确度为止

单峰函数极值



如图所示，已知某游泳池的长度为50米,宽度为25米。某人游泳速度为0.8米/秒，步行速度为1.2米/秒。
编程求解从A点到B点间何处下水游到C点时间最短，输出该点距A点的长度x和所求最短时间y。
说明:距离的长度精度控制在0.1米或以下。

$$t(x) = \frac{x}{1.2} + \frac{\sqrt{(50-x)^2+25^2}}{0.8}$$



穷举法

长度精度要求在0.1m以下即可，因此以0.1m为步长，穷举所有的距离长度，找到时间最小的解。

部分穷举

$t = f(x)$凸函数，区间内有唯一极值点（导数等于0的点），导数是单调增的（先小于0，后大于0）。

算法：

从原点开始穷举，时间是先减小后增大的，所以当穷举到$f(x_{i+1}) > f(x_i)$时，说明极值点x^*在x_i附近，可以提前退出穷举，返回x_i。

数值求导+二分法

当导函数比较复杂时，可以使用数值方法求解函数在某一点的导数。本题可使用数值方法求解区间端点的导数，然后用二分法找到导数等于0的点。

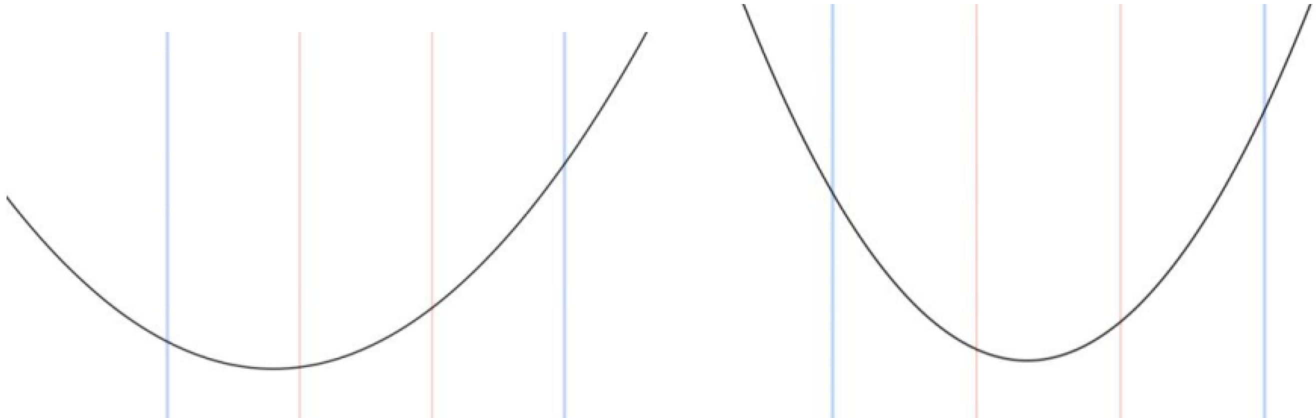
数值求导方法：

- 差商型求导方法
- 插值型求导方法

```c
// 前向差商求导法
#include <stdio.h>
#include <math.h>
#define EPS 1e-6
#define DELTA 1e-6

// 时间t关于x的函数
double t(double x)
{
    return x / 1.2 + sqrt((50 - x) * (50 - x) + 625) / 0.8;
}

double dt(double x)
{
    return (t(x + DELTA) - t(x)) / DELTA;
}

int main(int argc, char **argv)
{
    double low = 0, high = 50, mid;
    int i;
    for (i = 0; high - low >= EPS; i++)
    {
        mid = (high + low) / 2;
        if (dt(mid) * dt(high) > 0)
            high = mid;
```

```c
            else
                low = mid;
        }
        printf("x=%lf, t=%lf, iters=%d\n", low, t(low), i);
    }

// x=27.639320, t=64.959041, iters=26
```

三分法



将区间三分，根据拉格朗日中值定理，每个区间的两个端点都可以算出该区间内某点的导数值，这样三个区间能算出三个导数点，与0比较：

- $f'(x_1) < 0, f'(x_2) \le 0, f'(x_3) > 0$，去掉左边区间
- $f'(x_1) < 0, f'(x_2) \ge 0, f'(x_3) > 0$，去掉右边区间

这样就可以去掉两端的某个区间，将剩下的区间继续三分。

为了加快收敛速度，可以将两端区间设的尽可能的大，中间尽可能的小，这样每次都会去掉很大一部分区间。极限情况是中间区间长度趋向0，这相当于求中点导数的二分法。

```c
// 三分法
#include <stdio.h>
#include <math.h>
#define EPS 1e-6
#define RATIO 10

// 时间t关于x的函数
double t(double x)
{
    return x / 1.2 + sqrt((50 - x) * (50 - x) + 625) / 0.8;
}

int main(int argc, char **argv)
{
    double low = 0, high = 50, ml, mh, mid_len;
```

```c
    int i;
    for (i = 0; high - low >= EPS; i++)
    {
        mid_len = (high - low) / (2 * RATIO + 1);
        ml = low + mid_len * RATIO;
        mh = ml + mid_len;
        if (t(ml) < t(mh))
            high = mh;
        else
            low = ml;
    }
    printf("x=%lf, t=%lf, iters=%d\n", low, t(low), i);
}

//RATIO=1:  x=27.639320, t=64.959041, iters=44
//RATIO=10: x=27.639321, t=64.959041, iters=28
```

费马原理

利用光学里的费马原理直接求解方程，可以求得精确解，但是适用范围小，不能解决其他复杂方程的求极值问题，故不做讨论。