关于悬链线问题的计算

10317110 曾炳诚

摘　要：本文利用了四阶 Runge-Kutta 法计算出悬链线方程，同时考虑计算离散情况下的悬链线方程，即铁链，利用二阶牛顿法解决了二元方程的根最终求得其方程。最后利用蒙特卡罗算法尝试解决悬链线问题，证实了蒙特卡罗方法用于解决变分问题的可行性。

关键词：悬链线，四阶 Runge-Kutta 法，边界值问题，二阶牛顿迭代法，蒙特卡罗方法

（一） 引言

悬链线，catenary，是一个古老的难题。这个问题当年伽利略曾经考虑过，在之后牛顿给出了解答。问题是这样的，一条柔软的重绳索，两端固定在同一高度，挂在墙上， 请问这条绳索的形状是怎样的。如图所示：

这条曲线是一条双曲余弦， $\mathrm{y}=\frac{e^{x}+e^{-x}}{2}$ 这个问题其实是一个标准的变分问题，就跟最速降线一样。这个问题可以重新表述为，给定 y(0) =y(a) =0, 求解一条曲线 y=y(x), 使得系统的总能量最小。

最小旋转曲面也是类似的问题：给定一条曲线 y = f(x) ，让这条曲线绕着 x 轴旋转一周，求一条这样的曲线，使得得到的旋转曲面的表面积最小。

（二） 算法思路

1.悬链线方程计算（边界值问题）

已知$\mathrm{E}=\int_{a}^{b}\lambda \mathrm{y}\sqrt{1+y'^{2}}dx$（ λ 为线密度），根据欧拉-拉格朗日方程，使得这个积分取得的机智必须满足$\frac{\mathrm{d}}{\mathrm{dx}}\frac{\partial L}{\partial y'}-\frac{\partial L}{\partial y}=0$，$\mathrm{L}=\mathrm{y}\sqrt{1+\mathrm{y}'^{2}}$，化简后得到微分方程$1+\mathrm{y}'^{2}=yy''$，其中 y(0)=y(a)=0，是一个边界值问题。

对于边界值问题，我们可以先设其初值 y(0)=0 和 y′(0)=u_0，然后利用四阶 Runge-Kutta 法计算出两个试探解。然后将两个解线性相加则可以求得悬链线方程。代码如下：

```
void RungeKutta(double y1[],double y2[])
{
    const double dx=1.0/Length;
    const double dx2=dx*dx/2;

    for(int i=0;i<Length;i++)
    {
        const double y1d1=y2[i];
        const double y2d1=y1[i]-1-y2[i]*y2[i];
        const double y1d2=y2d1;
        const double y2d2=y1d1-2*y2d1*y2[i];

        y1[i+1]=y1[i]+y1d1*dx+y1d2*dx2;
        y2[i+1]=y2[i]+y2d1*dx+y2d2*dx2;
    }
}
```

（利用四阶 Runge-Kutta 计算试探解）

```
int main()
{
    const int l1=Length+1;

    double y11[l1],y12[l1],y21[l1],y22[l1];

    y11[0]=U0;
    y12[0]=0.2;     //The first trial solution
    RungeKutta(y11,y12);

    y21[0]=U0;
    y22[0]=0.5;     //The second trial solution
    RungeKutta(y21,y22);

    const double a=(y21[Length]-U1)/(y21[Length]-y11[Length]);
    const double b=(U1-y11[Length])/(y21[Length]-y11[Length]);

    ofstream out("data.dat");
    for(int i=0;i<l1;i++)
    {
        out<<1.0*i/Length<<"\t"<<-1*(a*y11[i]+b*y21[i])<<endl;
    }
    out.close();

    return 0;
}
```

（主函数）

2.铁链方程模拟

现在我要考虑的问题是一个离散的悬链线问题。我要计算的不再是一条连续的均匀的重绳索自然下垂的形状，而是一条铁链的下垂形状。铁链跟绳索显然不

一样，因为铁链是一段一段的，而绳索是连续而均匀的。假设铁链的每一小段之间没有摩擦，那么现在问题同样可以归结为计算铁链在重力场中的极小势能。假设一根铁链长度为 L，由 N 个小段刚杆组成，每一小段杆长度为△，计算这根铁链在自由悬挂时的形状。根据假设，则有L = NΔ. 铁链悬挂在 A 点和 B 点，A 点坐标为(0, 0)， B 点坐标为(a, 0)，0<a<L，于是铁链可以用 N+1 个点的坐标来描述。

$$
\begin{aligned}
&\overrightarrow{Op_0} = \overrightarrow{OA} = (0,0)\\
&\overrightarrow{p_0p_1} = \Delta(\cos\theta_1, \sin\theta_1)\\
&\cdots\\
&\overrightarrow{p_{i-1}p_i} = \Delta(\cos\theta_i, \sin\theta_i)\\
&\cdots\\
&\overrightarrow{p_{N-1}p_N} = \Delta(\cos\theta_N, \sin\theta_N)\\
&\overrightarrow{Op_N} = \overrightarrow{OB} = (a,0)
\end{aligned}
$$

于是可以得到第 i 个点的坐标为：

$$
\begin{aligned}
\overrightarrow{p_0p_i} &= \sum_{j=1}^{i} \overrightarrow{p_{j-1}p_j}\\
&= \sum_{j=1}^{i} \Delta(\cos\theta_j, \sin\theta_j)\\
&= \Delta(\sum_{j=1}^{i}\cos\theta_j, \sum_{j=1}^{i}\sin\theta_j), i = 0,1,2,\ldots,N
\end{aligned}
$$

因为第 N 个点的坐标限制为(a, 0)，所以得到两个限制条件：

$$
\sum_{j=1}^{N}\cos\theta_j = \frac{a}{\Delta} < N
$$

$$
\sum_{j=1}^{N}\sin\theta_j = 0
$$

然后算出没小段的势能（假设质心在杆的中点）：

$$
\begin{aligned}
E_i &= \lambda\Delta g\frac{1}{2}(y_{i-1} + y_i)\\
&= \frac{1}{2}\lambda\Delta g\Big(\Delta\sum_{j=1}^{i-1}\sin\theta_j + \Delta\sum_{j=1}^{i}\sin\theta_j\Big)\\
&= \lambda\Delta^2 g\Big(\frac{1}{2}\sin\theta_i + \sum_{j=1}^{i-1}\sin\theta_j\Big), i = 1,2,\ldots,N
\end{aligned}
$$

把所有杆势能求和便得到了总势能：

$$
\begin{aligned}
E &= \sum_{i=1}^{N} E_i \\
&= \sum_{i=1}^{N} \lambda \Delta^2 g\Big(\frac{1}{2}\sin\theta_i + \sum_{j=1}^{i-1}\sin\theta_j\Big) \\
&= \lambda \Delta^2 g\Big(\frac{1}{2}\sum_{i=1}^{N}\sin\theta_i + \sum_{i=1}^{N}\sum_{j=1}^{i-1}\sin\theta_j\Big)
\end{aligned}
$$

我们有两个约束条件，引入拉格朗日乘子ϵ_1，ϵ_2，得：

$$
E = \lambda \Delta^2 g\Big(\frac{1}{2}\sum_{i=1}^{N}\sin\theta_i + \sum_{i=1}^{N}\sum_{j=1}^{i-1}\sin\theta_j\Big) + \epsilon_1\Big(\sum_{i=1}^{N}\cos\theta_i - \frac{a}{\Delta}\Big) + \epsilon_2\sum_{i=1}^{N}\sin\theta_i
$$

求极小值得：

$$
\begin{aligned}
\frac{\partial E}{\partial \theta_k} &= \lambda \Delta^2 g\Big(\frac{1}{2}\cos\theta_k + \sum_{i=k+1}^{N}\cos\theta_k\Big) - \epsilon_1\sin\theta_k + \epsilon_2\cos\theta_k \\
&= \Big((1/2 + N - k)\lambda\Delta^2 g + \epsilon_2\Big)\cos\theta_k - \epsilon_2\sin\theta_k \\
&= 0, k = 1, 2, \ldots, N
\end{aligned}
$$

计算可得平衡后每一小段杆的平衡角度：

$$
\begin{aligned}
\tan\theta_k &= \frac{(1/2 + N - k)\lambda\Delta^2 g + \epsilon_2}{\epsilon_1} \\
&= (1/2 + N - k)\alpha + \beta, k = 1, 2, \ldots, N
\end{aligned}
$$

$$
\alpha = \frac{\lambda\Delta^2 g}{\epsilon_1}, \beta = \frac{\epsilon_2}{\epsilon_1}
$$

代入约束条件计算得两个方程：

$$
f(\alpha, \beta) = \sum_{k=1}^{N}\frac{1}{\sqrt{1 + \Big((1/2 + N - k)\alpha + \beta\Big)^2}} - \frac{a}{\Delta} = 0
$$

$$
g(\alpha, \beta) = \sum_{k=1}^{N}\frac{(1/2 + N - k)\alpha + \beta}{\sqrt{1 + \Big((1/2 + N - k)\alpha + \beta\Big)^2}} = 0
$$

利用二阶的牛顿法可求解这个方程：

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \begin{pmatrix} \frac{\partial f(x_0,y_0)}{\partial x} & \frac{\partial f(x_0,y_0)}{\partial y} \\ \frac{\partial g(x_0,y_0)}{\partial x} & \frac{\partial g(x_0,y_0)}{\partial y} \end{pmatrix}^{-1} \begin{pmatrix} f(x_0,y_0) \\ g(x_0,y_0) \end{pmatrix}$$

求出 α 和 β 即可求出所有杆的平衡角度，即可算出所有节点的坐标。

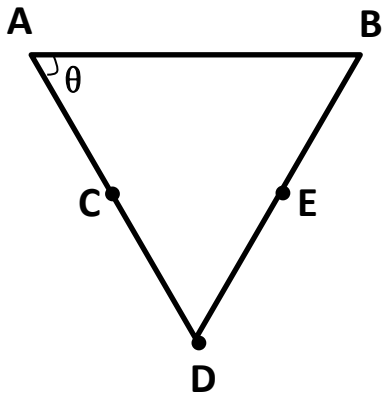
代码如下：

```
int main()
{
    double m[2][2];
    double m1[2][2];
    double m2[2][2];
    double x;
    double y;
    if(N==2)
    {x=-3.0;y=3.0;}       //设定N=2时的初始值
    if(N==4)
    {x=2.0;y=-4.0;}       //设定N=4时的初始值
    if(N==8)
    {x=1.0;y=-4.0;}       //设定N=8时的初始值
    if(N==10)
    {x=-0.8;y=4.0;}       //设定N=10时的初始值
    if(N==20)
    {x=0.1;y=-1.0;}       //设定N=20时的初始值
    double theta=0;
    for(int i=0;i<10;i++)
    {
        CreateMatrix(m,x,y);
        //Show(m);
        BansuiMatrix(m,m1);
        //Show(m1);
        Inverse(m,m1,m2);
        //Show(m2);
        x=x-(m2[0][0]*f(x,y)+m2[0][1]*g(x,y));
        y=y-(m2[1][0]*f(x,y)+m2[1][1]*g(x,y));
        //cout<<x<<"   "<<y<<"   "<<f(x,y)<<"   "<<g(x,y)<<endl;
    }                                                    //利用牛顿法计算出f(α , β )=0，g(α , β )=0的根，迭代十次左右就能得到精确的根
    ofstream out("data.dat");
    for(int i=1;i<N+2;i++)
    {
        double X=0;
        double Y=0;

        for(int j=1;j<i;j++)
        {
            theta=atan((0.5+N-j)*x+y);
            X=X+cos(theta)*delta;
            Y=Y+sin(theta)*delta;
            Y=-abs(Y);
        }
        out<<X<<"\t"<<Y<<endl;
    }                                                    //算出铁链上所有节点的坐标
    out.close();
    return 0;
}
```

3.蒙特卡罗模拟

和离散模型类似，我们可以利用“系统能量最低”这一条件进行蒙特卡罗模拟。为了使铁链的两端点固定，每一小段定长的杆与水平线所形成的角度与上一个小段杆与水平线形成的角度有关。为了化简算法，我们可以利用对称性。考虑一半的铁链的角度就可以考虑到整个模型的能量。不过该模型每一小段杆之间有很强的限制关系，当段数 N 足够大的时候情况会十分复杂导致算法冗长繁琐。这里我们可以从简单入手，考虑 N=4 的情况。

首先我们将这条铁链初始化，根据对称性，我们可以将它初始化成如下图所示的结构：



然后我们随机生成一个数 i 作为杆 AC 与水平线所成的变换过后的角度 θ’。为了满足最终所构成的形状还是一个闭合图形，其余的杆的角度也应该有相应的变化，该变化由函数 Change_theta 给出。然后可以算出所有变换过后的节点的坐标。可以根据

$$E_i = \lambda \Delta g \frac{1}{2}(y_{i-1} + y_i) \qquad E = \sum_{i=1}^{N} E_i$$

分别算出变换前后的系统总能量，将其相减即可算出能量变化 dE。若该变化使得 dE 大于 0，接受更改方案；若 dE 小于 0，则产生一个 0~1 之间的随机概率 p，如果$e^{\frac{dE}{T}} > p$则接受更改方案；否则不更改方案。重复上述步骤，最后用得到的角度可以算出所有节点的坐标。

程序如下：

```
void init_theta(double theta[])  //初始化角度
{
    for(int i = 0; i < m/2; i++)
    {
        theta[i] = M_PI/3;
    }
    for(int i=0;i<m/2;i++)
    {
        theta[m-1-i] = -theta[i];
    }
}

void Change_theta(double theta[],double k,int j)//改变角度
{
    double dY=sin(theta[j])*dx-sin(k)*dx;
    double dX=cos(theta[j])*dx-cos(k)*dx;
    for(int i=j+1;i<m/2;i++)
    {
        theta[i] =acos((a/2-cos(k)*dx)/dx);
    }
    for(int i=0;i<m/2;i++)
    {
        theta[m-1-i] = -theta[i];
    }
}
```

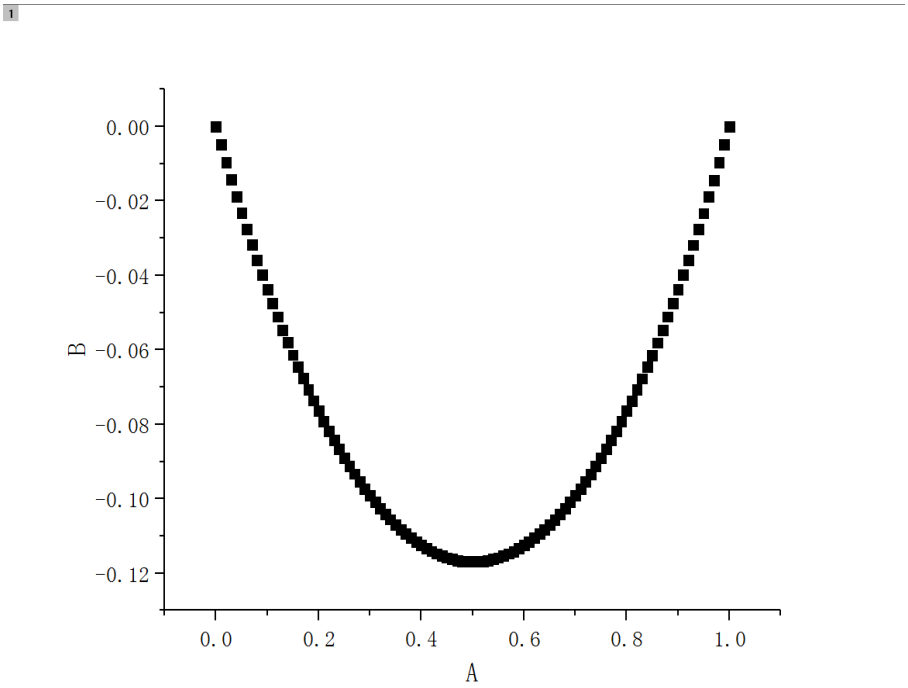
```
void init_y(double y[], double theta[])
{
    y[0]=0;
    for(int i=1;i<=m;i++)
    {
        for(int j=1;j<=i;j++)
        {
            y[i]+=(dx*sin(theta[j-1]))*-1.0;
        }
    }
}

void init_x(double x[], double theta[])
{
    x[0]=0;
    for(int i=1;i<=m;i++)
    {
        for(int j=1;j<=i;j++)
        {
            x[i]+=dx*cos(theta[j-1]);
        }
    }
}
```

```
void MonteCarlo(double theta[],double y[],double x[])
{
    int iaccept=0;              //accept percent
    double theta1[m]={0};
    double y1[m+1]={0};
    double x1[m+1]={0};

    for(int mc=MCS;mc>0;mc--)
    {
                int j=0;
                Copy(theta,theta1,m);

                Zero(y);
                Zero(x);
                Zero(y1);
                Zero(x1);
                init_y(y,theta);
                Copy(y,y1,m+1);

                init_x(x,theta);
                Copy(x,x1,m+1);
                double i=(rand()/32767.0+2.0)/6.0*M_PI;

                double E1=Energy(y);

                theta1[j]=i;
                Change_theta(theta1,i,j);
                init_y(y1,theta1);
                init_x(x1,theta1);

                double E2=0;
                E2=Energy(y1);

                const double de=E1-E2;
                //cout<<"de="<<de<<endl;
                //delta_E=H_old-H_new

                if(de>0||exp(de/T)*RAND_MAX>rand()) // judge whether accept the change of the spin .
                {
                    Copy(theta1,theta,m);
                    iaccept++;
                }
                //cout<<"========================"<<endl;

    }
```

（三） 结果展示与分析

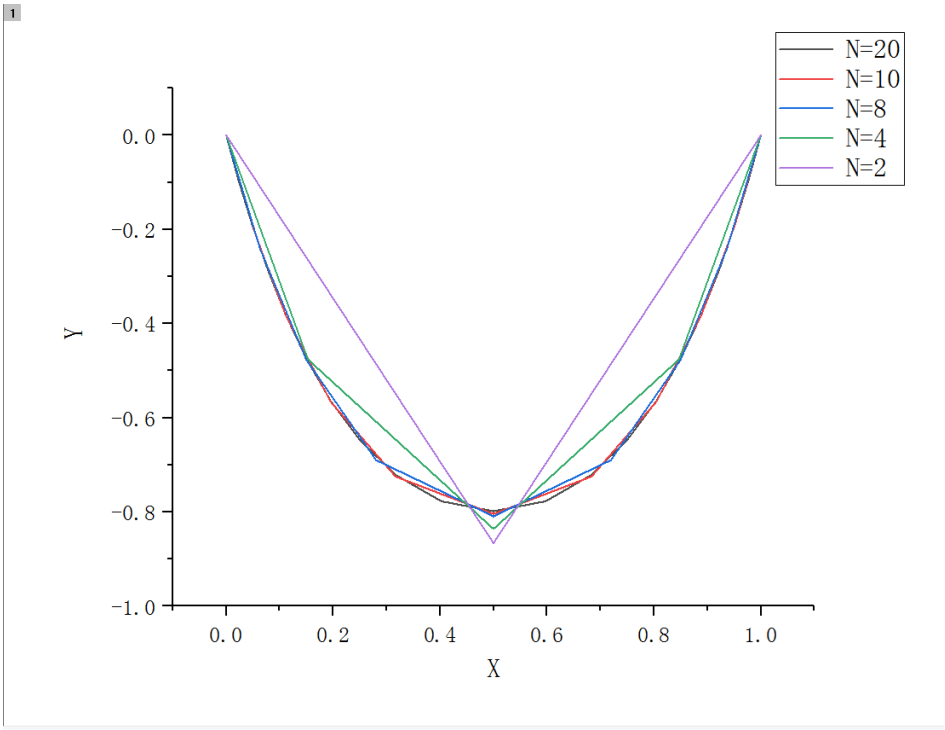
1.连续悬链线：（端点坐标为（0，0）和（1，0））

程序运行结果如图所示



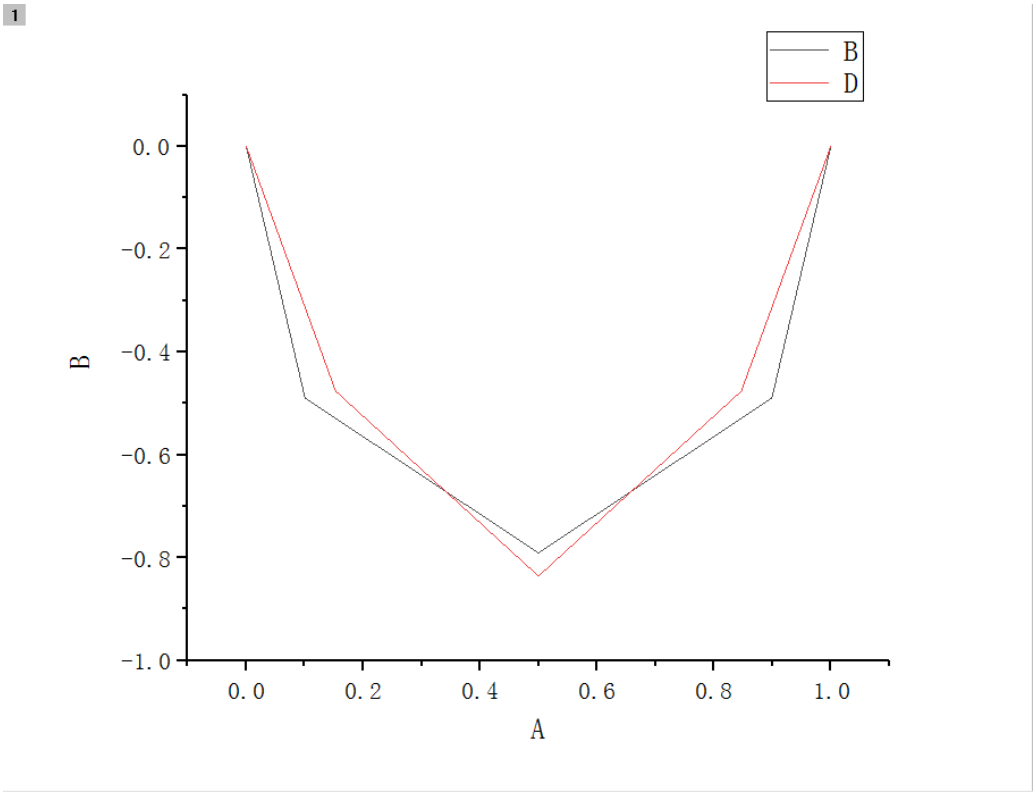
这是边界值问题解出的悬链线方程。

2.离散情况（铁链）：（端点坐标为（0，0）和（1，0），铁链总长度为 2）

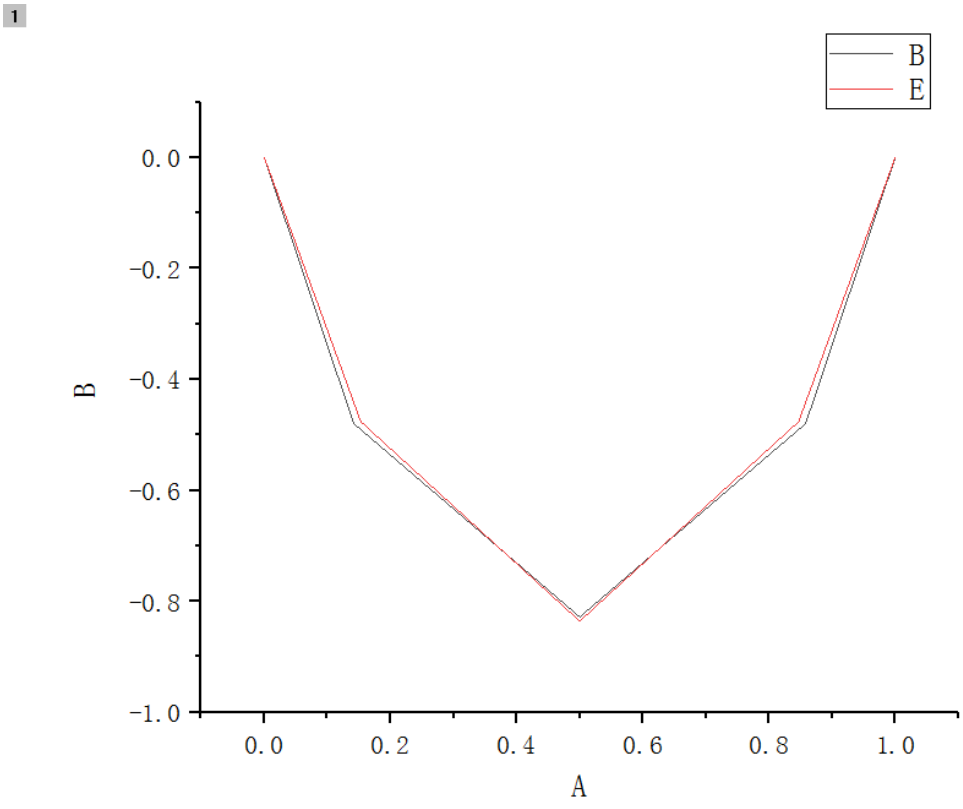


可以观察到，当 N=20 时，铁链和均匀绳子的平衡态构型没有太大差别了，由此可推知，继续增大 N 的数目可无限趋近于均匀悬链线构型。

3.蒙特卡罗模拟离散情况（当 N=4 时）:（端点坐标为（0，0）和（1，0），铁链总长度为 2）



可以看出此时有较大的误差，不过当我们把 exp(de/T)*RAND_MAX>rand()命令注释掉之后运行得到的结果误差就小了许多：



黑线为蒙特卡罗模拟结果，可以看出和理论值算出的结果相差不大，证实了用蒙特卡罗计算悬链线的可行性。

（四） 结论

离散情况下使用二阶牛顿法求解二元方程时，其优点是只需要迭代五次左右便可以得到方程解，运算效率高，但需要注意的是缺点是，它的初始值的选择对最终结果有很大的影响，较小的误差可能导致运算结果发散。

运用蒙特卡罗方法计算时，可以发现该问题计算量过小，限制条件多，并不能完全体现该方法的优势。对于随机性更强的模型，蒙特卡罗方法将有更好的发挥空间。

参考文献：

老大中，变分法基础（第三版），北京：国防工业出版社