用 HTML 演示高斯波包通过一维方势阱模型

王剑心 物理学系 17307110210

摘要：薛定谔方程的求解是量子力学中十分重要的内容，本文着力于使用 HTML 这一方便的平台实现高斯波包通过一维方势阱情景的可视化，帮助初学者建立形象的物理模型。

一、引言：

物理学的发展离不开物理模型，物理模型是物理问题的抽象与概括。客观事物多种多样，其中包含的物理性质和规律往往是十分复杂的，因此在物理学习中使用适当的模型来代替实在的客体，可以使事物的性质和规律简单明了的表达，有助于学生更加直观的理解物理规律，提高接受新知识的能力。在互联网化的今天，移动学习和模拟演示实验扮演着越来越重要的角色，而 HTML 就提供了这样一种平台，利用 HTML5Canvas 画布标签以及 Javascript 脚本语言构建跨平台的物理模型，能够为学习者提供更好的学习体验。[1]

薛定谔方程是量子力学中十分重要的内容。形象理解一维含时薛定谔方程解的形式十分必要，本程序利用 HTML 和 Javascript 语言求解初始态为高斯波函数的一维含时薛定谔方程，形象展示波函数通过势阱的变化情况。

二、相关原理

1. 一维薛定谔方程：$\frac{\hbar^2}{2m}\frac{\partial^2\varphi}{\partial x^2}=-i\hbar\frac{\partial\varphi}{\partial t}+V\varphi$

2. 有限差分法：含时薛定谔方程解的形式为$\Psi(x,t)=\Psi(x,0)e^{-\frac{iEt}{\hbar}}$，求解析解会遇到困难，利用 Grank-Nicolson 方法对偏微分方程进行有限差分，即可对方程数值求解。

对于形如$\nabla^2u(x,t)=\frac{\partial u(x,t)}{\partial t}$的方程，利用有限差分代替微分，得到：

$$\frac{\theta}{\Delta x^2}\left(u_{i-1,j}-2u_{i,j}+u_{i+1,j}\right)+\frac{1-\theta}{\Delta x^2}\left(u_{i-1,j-1}-2u_{i,j-1}+u_{i+1,j-1}\right)$$
$$=\frac{1}{\Delta t}\left(u_{i,j}-u_{i,j-1}\right) \qquad (1)$$

$\theta=0$对应向前差分，$\theta=1$对应向后差分，$\theta=1/2$对应 Grank-Nicolson 差分方法。其中i代表每个时刻不同坐标的序号，而j则代表不同时刻的序号。运用该方法对一维薛定谔方程进行有限差分得到：

$$\frac{\hbar^2}{4m\Delta x^2}\left(u_{i-1,j}-2u_{i,j}+u_{i+1,j}+u_{i+1,j+1}-2u_{i+1,j}+u_{i+1,j-1}\right)=-\frac{i\hbar}{\Delta t\left(u_{i,j}-u_{i,j-1}\right)}+Vu_{i,j} \qquad (2)$$

整理后形式变为：

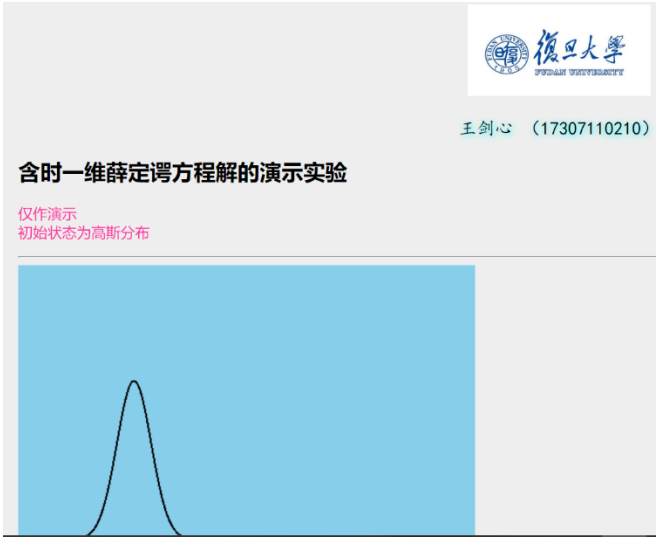
$$\left(I-\frac{V\Delta t}{i\hbar}-\alpha\gamma B\right)u_j=(\alpha\gamma B+I)u_{j-1} \tag{3}$$

其中 $\alpha=\frac{\Delta t}{\Delta x^2}$ $\gamma=\frac{\hbar}{4im}$ $\hat{B}=\begin{bmatrix}2&-1&0&0&\dots\\-1&2&-1&0&\dots\\\dots&\dots&\dots&\dots&\dots\\\dots&\dots&\dots&\dots&-1\\0&0&\dots&-1&2\end{bmatrix}$

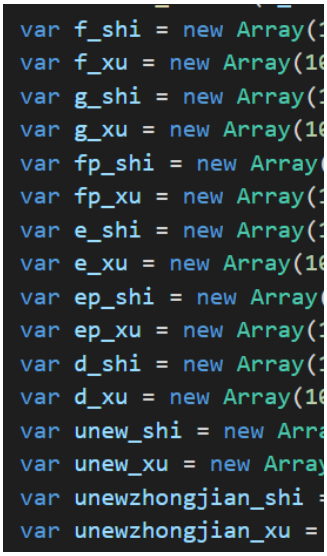
根据（3）式，可以根据由上一时刻波函数的纵坐标组成的数组u_{j-1}数值计算得到该时刻波函数的纵坐标组成的数组u_j。因此，只要给出波函数的初始状态并不断计算，就可以计算得到波函数经过势场后纵坐标的一系列变化。初始的高斯波函数设为$\exp\left(-\frac{(x-x_0)^2}{2\sigma^2}+ikx\right)$。

三、程序主要功能设计说明

整体页面十分简单（如图一），对演示内容做了简单的说明，并设置开始按钮。



图一 页面整体展示



图四 分别存储实部虚部

1. 复数类型运算

由于薛定谔方程涉及复数运算，而 Javascript 语言所支持的数据类型十分有限，并不包括复数类型。因此程序定义了四种复数运算方式（分别是加减乘除（如图二）），然后在后续所有涉及到复数运算的地方都采用大小相同的数组分

```
function fushuchengfa(a,b,c,d){
    var e=new Array();
    e[0]=a*c-b*d;
    e[1]=a*d+b*c;
    return e;
}

function fushuchufa(a,b,c,d){
    var e=new Array();
    e[0]=(a*c+b*d)/(Math.pow(c,2)+Math.pow(d,2));
    e[1]=(b*c-a*d)/(Math.pow(c,2)+Math.pow(d,2));
    return e;
```

图二 定义复数运算

别存储其实部和虚部（如图三图四）。

```
u_shi[0][j]=u_shi[0][j]/Math.pow(con[0],0.5);
u_xu[0][j]=u_xu[0][j]/Math.pow(con[0],0.5);
```

图三分别存储实部虚部

2.解方程组

程序主体是依照（3）式对波函数不断迭代计算，因此解方程组成为该程序的重要内容。观察到（3）式涉及到的矩阵为三对角矩阵，求解可以使用三对角矩阵线性方程组的解法（如图五）。

```
function solve_matrix(A_shi,A_xu,r_shi,r_xu){
    var f_shi = new Array(1001).fill(0);
    var f_xu = new Array(1001).fill(0);
    var g_shi = new Array(1001).fill(0);
    var g_xu = new Array(1001).fill(0);
    var fp_shi = new Array(1001).fill(0);
    var fp_xu = new Array(1001).fill(0);
    var e_shi = new Array(1001).fill(0);
    var e_xu = new Array(1001).fill(0);
    var ep_shi = new Array(1001).fill(0);
    var ep_xu = new Array(1001).fill(0);
    var d_shi = new Array(1001).fill(0);
    var d_xu = new Array(1001).fill(0);
    var unew_shi = new Array(1001).fill(0);
    var unew_xu = new Array(1001).fill(0);
    var unewzhongjian_shi = new Array(1001).fill(0);
    var unewzhongjian_xu = new Array(1001).fill(0);
    for(var i = 0; i < 1001; i++){
        f_shi[i]=A_shi[i][i];
        f_xu[i]=A_xu[i][i];
        if (i<=999){
            g_shi[i]=A_shi[i][i+1];
            g_xu[i]=A_xu[i][i+1];
        }
    }
    fp_shi[0]=f_shi[0];
    fp_xu[0]=f_xu[0];
    for(var k = 1; k < 1001; k++){//注意这里的k是从1开始的
        e_shi[k]=A_shi[k][k-1];
        ep_shi[k]=fushuchufa(e_shi[k],e_xu[k],fp_shi[k-1],fp_xu[k-1])[0];
        ep_xu[k]=fushuchufa(e_shi[k],e_xu[k],fp_shi[k-1],fp_xu[k-1])[1];
        fp_shi[k]=f_shi[k]-fushuchengfa(ep_shi[k],ep_xu[k],g_shi[k-1],g_xu[k-1])[0];
        fp_xu[k]=f_xu[k]-fushuchengfa(ep_shi[k],ep_xu[k],g_shi[k-1],g_xu[k-1])[1];
    }
    d_shi[0]=r_shi[0];
    d_xu[0]=r_xu[0];
    for(var k = 1; k < 1001; k++){//注意这里的k是从1开始的
        d_shi[k]=r_shi[k]-fushuchengfa(ep_shi[k],ep_xu[k],d_shi[k-1],d_xu[k-1])[0];
        d_xu[k]=r_xu[k]-fushuchengfa(ep_shi[k],ep_xu[k],d_shi[k-1],d_xu[k-1])[1];
    }
    unew_shi[1000]=fushuchufa(d_shi[1000],d_xu[1000],fp_shi[1000],fp_xu[1000])[0];
    unew_xu[1000]=fushuchufa(d_shi[1000],d_xu[1000],fp_shi[1000],fp_xu[1000])[1];
    for(var k = 999; k > -1; k--){
        unewzhongjian_shi[k]=d_shi[k]-fushuchengfa(g_shi[k],g_xu[k],unew_shi[k+1],unew_xu[k+1])[0];
        unewzhongjian_xu[k]=d_xu[k]-fushuchengfa(g_shi[k],g_xu[k],unew_shi[k+1],unew_xu[k+1])[1];
        unew_shi[k]=fushuchufa(unewzhongjian_shi[k],unewzhongjian_xu[k],fp_shi[k],fp_xu[k])[0];
        unew_xu[k]=fushuchufa(unewzhongjian_shi[k],unewzhongjian_xu[k],fp_shi[k],fp_xu[k])[1];
    }
    var unewlast = new Array();
    //alert(unew_shi);//这里也全都是NaN
    unewlast[0]=unew_shi;
    unewlast[1]=unew_xu;
    return unewlast;
}
```

图五 解方程组

3.迭代计算和归一化操作

每次计算得到的该时刻的值都会作为参数重新带入计算下一时刻的波函数纵坐标。

计算过程的每次迭代都会为了保证波函数模平方归一而对得到的纵坐标值进行对应的操作。

最终得到的 y 矩阵，行数为运算的时刻数，每一行都代表该时刻根据上个时刻计算得到的现在的波函数纵坐标值。（如图六）

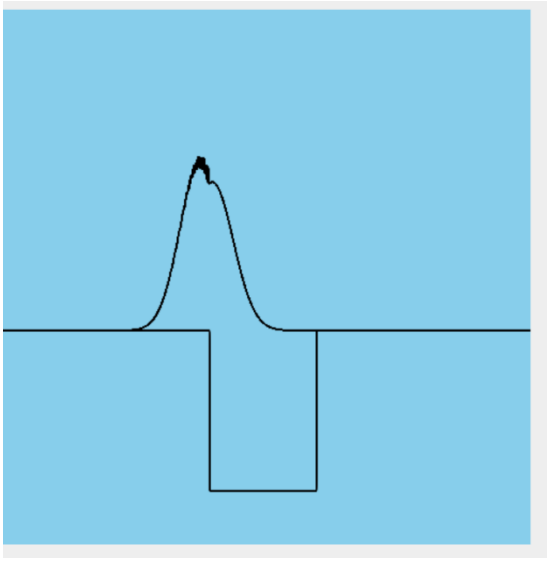
```
for(var i = 1; i < jisuanliang+1; i++){//注意我这里的i是从1开始的，因为我的0已经设计好了
    //jisuan_r(u_shi[i-1],u_xu[i-1]);//这里利用解线性方程组迭代出下一个y[i]的值即可      这里的u_shi 应该也是个矩阵随i迭代
    u_shi[i]=solve_matrix(A_shi,A_xu,jisuan_r(u_shi[i-1],u_xu[i-1])[0],jisuan_r(u_shi[i-1],u_xu[i-1])[1])[0];//这里就很
    u_xu[i]=solve_matrix(A_shi,A_xu,jisuan_r(u_shi[i-1],u_xu[i-1])[0],jisuan_r(u_shi[i-1],u_xu[i-1])[1])[1];
    //alert (u_shi[i]);
    //现在已经不是了
    for(var j = 0; j < 1001; j++){
        y[i][j]=Math.pow(Math.pow(u_shi[i][j],2)+Math.pow(u_xu[i][j],2),0.5);
        con[i] += Math.pow(y[i][j],2);
        //alert (y[i][j]);
    }
    con[i] = con[i]*deltax;
    for(var j = 0; j < 1001; j++){
        y[i][j]=y[i][j]/Math.pow(con[i],0.5);
        u_shi[i][j]=u_shi[i][j]/Math.pow(con[i],0.5);
        u_xu[i][j]=u_xu[i][j]/Math.pow(con[i],0.5);
    }
}
```

图六 迭代计算 y 矩阵

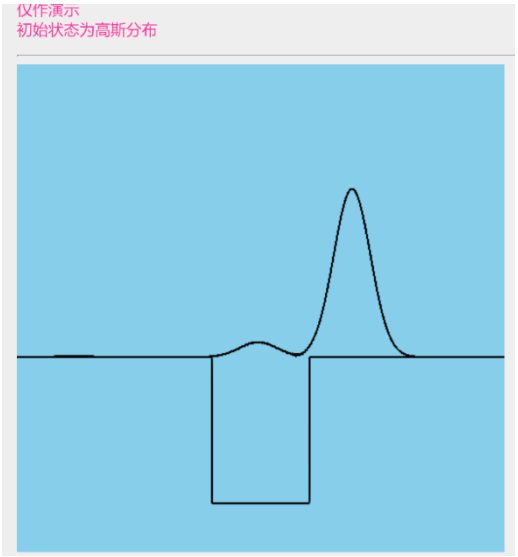
四、程序结果展示和说明

图七图八图九分别展示了波函数刚进入势阱、向右行进的波恰好离开势阱，

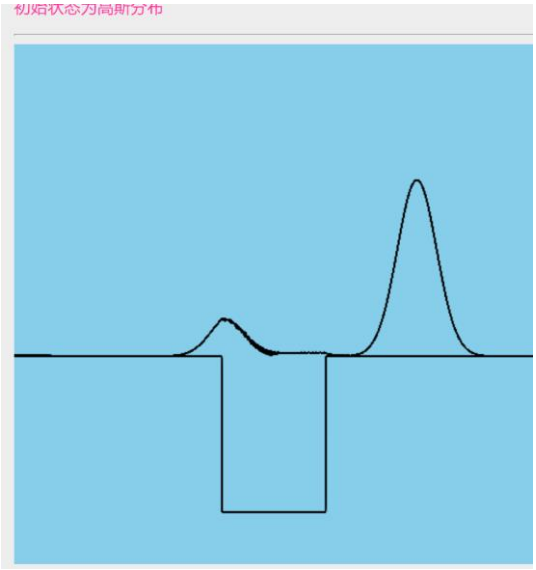
经过势阱后反射（向左行进）的波即将离开势阱时的结果。



图七



图八



图九

1. 结果说明：图中所展示的仅为波函数经过势阱的示意图，高度比例并不是真实的比例，程序迭代运算实际选取的时间间隔与动画演示的间隔也不一致。
2. 模拟结果的分析：从波函数的变化来看，波函数进过势阱的运动情况基本符合理论给出的变化。波函数进入势阱时会有不光滑的变化，未进入但即将进入势阱的小部分会有一些震荡，经过势阱后出现了相反方向运动的波。
3. 程序可改进的方向：对不同形式的势垒势阱可以做更多的模拟，让初学者对于波函数经过势垒和势阱的变化有更加全面深刻地理解。还可以调试出几个比较有代表性的势垒以及势阱的数值，展示出典型的波函数变化。该程序利

用的差分方法实际运算后比较繁琐，可以考虑更加优化可靠的算法实现利用该模型探究波函数透射率与其他物理量进一步的关系。

五、结论

本程序利用 Grank-Nicolson 有限差分方法对初始形式为高斯函数形式的波函数经过方势阱的变化进行了模拟。通过程序的模拟结果可以直观的看到波函数通过方势阱的变化，有助于帮助初学者对波函数以及薛定谔方程建立形象的物理图像。

六．参考文献：

[1]闫慧仙，吴天刚，王祖源，基于 HTML5Canvas 的跨平台物理模型构建，物理与工程，2014 年 S1 期.

[2]Griffiths D J,Schroeter D F.Introduction to quantum mechanics[M].Cambridge University Press,2018

[3]向红军老师，Computational Physics-PDE 计算物理基础 ppt