HTML5 测地线数值模拟

刘飞阳 16307110153

摘要: 本文介绍了一种基于 RK4 的用于数值模拟弯曲时空测地线的算法, 该算法简单且具有一般性, 适用于一般的时空度规. 在 Schwarzschild 度规下, 将该算法用 JavaScript 实现并内嵌入 HTML5, 便得到了一个 HTML5 的 Schwarzschild 测地线模拟器, 模拟了非旋转黑洞周围的粒子运动. 最后通过对经典广义相对论轨道的模拟再现, 该模拟器的可靠性也得到的验证.

一. 引言

对粒子在对应弯曲时空中的运动的准确描述, 是模拟黑洞和中子星的吸积过程等物理现象的基础. 同时也是广义相对论教学的重要部分. 在没有电磁外力的作用下, 粒子的运动轨迹为对应时空中的测地线. 因此, 测地线的计算在前沿的广相模拟中扮演着重要的作用. 同时在教学上, 几种特殊时空下的粒子轨道也是广义相对论教学的重要内容. 前者需要优秀的测地线算法, 而后者需要能直观展示测地线的工具.

在此背景下, 我们使用了一种简单可拓展的测地线算法用于计算教学上具有特殊意义的 Schwarzschild 测地线(即非自旋黑洞周围的粒子轨道). 用 JavaScript 实现该算法后, 我们方便地做出了一个 HTML5 Schwarzschild 测地线模拟器. 为验证该模拟器的可靠性, 我们用该模拟器再现了广相中经典的三叶草轨道.

二. 测地线

测地线是一组测地线方程的解. 在四维闵可夫斯基时空中 (全文假定时空符号$(-,+,+,+)$), 测地线方程组由四个二阶非线性微分方程组成. (见公式(1), 其中希腊字母可取 0, 1, 2, 3. $\Gamma^{\mu}_{\lambda\sigma}$为 Christoffel symbol, τ为 affine parameter, 4 维位置x^{μ}关于τ的导数定义了协变速度u^{μ}, 见公式(2)).

方程的形式一般而言与对应时空度规(metric)有关. 目前的许多测地线算法仅仅适用于几种特别时空, 如 Schwarzschild 时空(non-rotating blackhole)或者 Kerr 时空(rotating blackhole). 这些算法虽然快, 但是可拓展性不足.

$$\frac{d^2\,x^{\mu}}{d\,\tau^2}+\Gamma^{\mu}_{\lambda\sigma}\,\frac{d\,x^{\lambda}}{d\,\tau}\,\frac{d\,x^{\sigma}}{d\,\tau}=0 \tag{1}$$

$$dx^{\mu}/d\tau=u^{\mu},\qquad u_i=g_{i\mu}u^{\mu} \tag{2}$$

虽然对于 Schwarzschild 时空或者 Kerr 时空而言, 存在测地线的解析解, 但大多数情况下, 解析解是不存在的. 常见的选择是使用数值积分手段计算数值解, 如使用 RK 算法. 为了对测地线进行数值积分, 我们需要对测地线方程组(1)进行变换, 在 ADM (Arnowitt–Deser–Misner)框架下重新表述测地线方程. 在该框架下, 任何时空度规可被写为如下形式:

$$g_{\mu\nu} = \begin{pmatrix} -\alpha^2 + \beta_k\beta^k & \beta_i \\ \beta_j & \gamma_{ij} \end{pmatrix} \tag{3}$$

其逆度规为:

$$g^{\mu\nu} = \begin{pmatrix} -1/\alpha^2 & \beta^i/\alpha^2 \\ \beta^j/\alpha^2 & \gamma^{ij} - \beta^i\beta^j/\alpha^2 \end{pmatrix} \tag{4}$$

其中, γ^{ij}为γ_{ij}的逆, $\beta^i = \gamma^{ij}\beta_j$. 这样对于任何一般的时空度规, 我们都可以很快地写出α, β^i, 和γ_{ij}的表达式. 在以上框架下, 测地线方程(1)可被重新写为一阶微分方程组的形式[1]:

$$\frac{dx^i}{dt} = \gamma^{ij}\frac{u_j}{u^0} - \beta^i \tag{5}$$

$$\frac{du_i}{dt} = -\alpha u^0\,\partial_i\alpha + u_k\,\partial_i\beta^k - \frac{u_j u_k}{2u^0}\partial_i\gamma^{jk} \tag{6}$$

$$u^0 = \left(\gamma^{jk}u_j u_k + \epsilon\right)^{1/2}/\alpha \tag{7}$$

其中$\epsilon = 0$对应无质量粒子的测地线(如光子, 也称 null geodesic), $\epsilon = 1$对应有质量粒子的测地线(称 time-like geodesic).

这样一来, 原先由四个二阶非线性微分方程组成的系统被替换为六个一阶方程, 更加适合做数值积分. 同时积分变量变成时间 t, 更有物理意义.

三. Runge-Kutta 数值积分

Runge-Kutta 方法是常用的数值积分方法, 我们使用 4 阶 Runge-Kutta 数值积分方法来处理变换后的测地线方程组. 该方法表述如下. 假设我们有一个一阶微分方程系统及其初始条件:

$$d\vec{\boldsymbol{y}}/dt = \vec{\boldsymbol{f}}(t,\vec{\boldsymbol{y}}), \qquad \vec{\boldsymbol{y}}(t_0) = \vec{\boldsymbol{y}}_0 \tag{8}$$

那么我们可以通过迭代的方法求得数值解:

$$\vec{\boldsymbol{y}}_{n+1} = \vec{\boldsymbol{y}}_n + 1/6\left(\vec{\boldsymbol{k}}_1 + \vec{\boldsymbol{k}}_2 + \vec{\boldsymbol{k}}_3 + \vec{\boldsymbol{k}}_4\right), \qquad t_{n+1} = t_n + h \tag{9}$$

其中 h 是每步迭代对应的自变量 t 该变量. $\vec{\boldsymbol{k}}_1, \vec{\boldsymbol{k}}_2, \vec{\boldsymbol{k}}_3, \vec{\boldsymbol{k}}_4$ 定义为:

$$\vec{k}_1 = h\vec{f}(t_n, \vec{y}_n), \qquad \vec{k}_2 = h\vec{f}(t_n + h/2, \vec{y}_n + \vec{k}_1/2)$$
$$\vec{k}_3 = h\vec{f}(t_n + h/2, \vec{y}_n + k_2/2), \qquad \vec{k}_4 = h\vec{f}(t_n + h, \vec{y}_n + \vec{k}_3) \tag{10}$$

用该方法来处理方程组(5)是简单直接的.

四. Schwarzschild 测地线数值计算的 JavaScript 实现

基于二, 三两节的讨论, 我们已经建立起了数值计算测地线的算法. 其简单直接, 适用于一切度规形式. 我们选取 Schwarzschild 度规, 用 JavaScript 来实现该算法. 选择 Schwarzschild 度规的一方面是其相对简单, 具有 SO(3)对称性, 解得的测地线始终保持在二维平面上方便 2d canvas 作图; 另一方面则是由于其在广义相对论教学中的特殊性, 其往往作为第一个被详细介绍的度规, 其对应测地线的解也会得到详细的讨论, 在此背景下 Schwarzschild 测地线的 HTML5 模拟具有一定教学意义.

Schwarzschild 度规的 line element 可以为:

$$\mathrm{d}\,s^2 = -\left(1 - \frac{2\,M}{r}\right)\mathrm{d}\,t^2 + \left(1 - \frac{2\,M}{r}\right)^{-1}\mathrm{d}r^2 + r^2\,\mathrm{d}\,a^2 \tag{11}$$

其中 $\mathrm{d}a^2 = \mathrm{d}\theta^2 + \sin^2\theta\,\mathrm{d}\varphi^2$, 是二维球面上的标准度规. 带入进 ADM 表述后, 得到线性微分方程组为(用 JavaScript 代码表示):

```
function u0(r, ur, uphi, e) {
    return Math.sqrt(((1 - 2/r) * (ur**2) + uphi/(r**2) + e)/(1 - 2/r));
}

function drdt(r, ur, uphi, e) {
    return (1 - 2/r)*ur/u0(r, ur, uphi, e);
}

function dphidt(r, ur, uphi, e) {
    return uphi/((r**2) * u0(r, ur, uphi, e));
}

function durdt(r, ur ,uphi, e) {
    return (-1) * u0(r, ur, uphi, e)/(r**2) + ((-1) * (ur/r)**2 + (uphi**2)/(r**3))/u0(r, ur, uphi, e);
}
```

```
function duphidt(r, ur, uphi, e) {
    return 0;
}
```

这里有 4 个方程而不是六个, 这是由于对称性帮我们消除了两个. 那么相对的 RK4 算法只需要调用这些函数即可, JavaScript 代码如下:

```
function RKupdate(dt, e) {
    var r, phi, ur, uphi;
    r = System.r;
    phi = System.phi;
    ur = System.ur;
    uphi = System.uphi;

    var k1 = {
        r: 0,
        phi: 0,
        ur: 0,
        uphi: 0
        },
        k2 = {
        r: 0,
        phi: 0,
        ur: 0,
        uphi: 0
        },
        k3 = {
            r: 0,
            phi: 0,
            ur: 0,
            uphi: 0
        },
        k4 = {
            r: 0,
            phi: 0,
            ur: 0,
```

```
        uphi: 0
    };
  k1.r = dt * drdt(r, ur, uphi, e);
  k1.phi = dt * dphidt(r, ur, uphi, e);
  k1.ur = dt * durdt(r, ur, uphi, e);
  k1.uphi = dt * duphidt(r, ur, uphi, e);

  k2.r = dt * drdt(r+k1.r/2, ur+k1.ur/2, uphi+k1.uphi/2, e);
  k2.phi = dt * dphidt(r+k1.r/2, ur+k1.ur/2, uphi+k1.uphi/2, e);
  k2.ur = dt *durdt(r+k1.r/2, ur+k1.ur/2, uphi+k1.uphi/2, e);
  k2.uphi = dt * duphidt(r+k1.r/2, ur+k1.ur/2, uphi+k1.uphi/2, e);

  k3.r = dt * drdt(r+k2.r/2, ur+k2.ur/2, uphi+k2.uphi/2, e);
  k3.phi = dt * dphidt(r+k2.r/2, ur+k2.ur/2, uphi+k2.uphi/2, e);
  k3.ur = dt *durdt(r+k2.r/2, ur+k2.ur/2, uphi+k2.uphi/2, e);
  k3.uphi = dt * duphidt(r+k2.r/2, ur+k2.ur/2, uphi+k2.uphi/2, e);

  k4.r = dt * drdt(r+k3.r, ur+k3.ur, uphi+k3.uphi, e);
  k4.phi = dt * dphidt(r+k3.r, ur+k3.ur, uphi+k3.uphi, e);
  k4.ur = dt *durdt(r+k3.r, ur+k3.ur, uphi+k3.uphi, e);
  k4.uphi = dt * duphidt(r+k3.r, ur+k3.ur, uphi+k3.uphi, e);

  dr = (1/6) * (k1.r + 2*k2.r + 2*k3.r + k4.r);
  dphi = (1/6) * (k1.phi + 2*k2.phi + 2*k3.phi + k4.phi);
  dur = (1/6) * (k1.ur + 2*k2.ur + 2*k3.ur + k4.ur);
  duphi = (1/6) * (k1.uphi + 2*k2.uphi + 2*k3.uphi + k4.uphi);

  System.r += dr;
  System.phi += dphi;
  System.ur += dur;
  System.uphi += duphi;
}
```

每调用一次 RKupdate 函数, 就会对储存粒子位置的全局变量进行 RK4 数值更新. 函数输入中,

dt 表示步长, e 代表方程(7)中的 epsilon, 区别粒子有无质量. 只要反复调用该函数, 就会得到数值模拟的测地线.

五. HTML5 模拟结果

将上一节中的 JavaScrit 代码应用到一个 HTML5 项目中是简单直接的, 这里不做赘述, 直接展示结果. 得到的项目是一个带有网页控制组件的 Schwarzschild 测地线模拟器. 其物理意义是模拟不自旋黑洞周围的粒子运动情况. 见图 1.

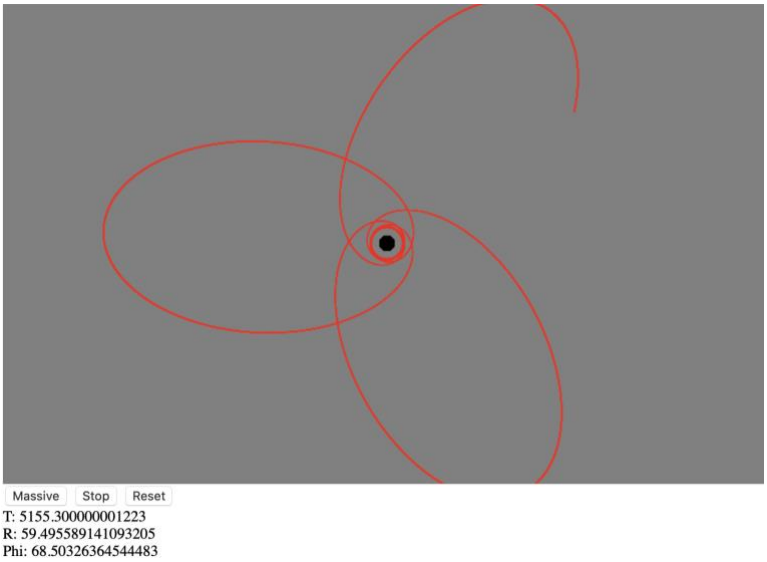

图 1.模拟得到的 Schwarzschild 测地线

图 1 中模拟出的是广相中经典的三叶草轨道[2], 其初始条件为(E = 0.987649, L = 3.9). 佐证了该模拟器及其背后算法的可靠性.

六. 参考文献

[1] F. Bacchini et al. *The Astrophysical Journal Supplement Series, 237:6 (20pp), 2018 July*

[2] Levin et al. *arXiv: 0802.0459v1 [gr-qc] 4 Feb 2008*