Celestial Mechanics and Astrometry
天体力学和天体测量学
§5 多体系统的计算机模拟

Lecturer: 黄志琦

http://zhiqihuang.top/cm

准备知识：算符的指数函数

如果 $\hat{A}$ 是算符，定义

$$e^{\hat{A}} = \sum_{n=0}^{\infty} \frac{1}{n!} \hat{A}^n.$$

那么，如果有两个算符 $\hat{A}, \hat{B}$，

$$e^{\hat{A}+\hat{B}} = e^{\hat{A}} e^{\hat{B}}$$

一般还成立吗？如果已知这两个算符对易呢？

哈密顿方程

设 q 为广义坐标， p为广义动量。
如果已知哈密顿量 $H(p,q)$， 则(x,p) 的时间导数由哈密顿方程给出:

$$\dot{p} = -\frac{\partial H}{\partial q}$$

$$\dot{q} = \frac{\partial H}{\partial p}$$

练练手，熟悉一下

设 $H = \frac{1}{2}p^2$，写出哈密顿方程。

练练手，熟悉一下

设 $H = \frac{1}{2}q^2$，写出哈密顿方程。

烧脑的观点来了

我们可以把任意二元函数 $H(\cdot,\cdot)$等价地看成一个 (p,q) 到 $(\dot{p},\dot{q})$ 的映射规则:

$$\dot{p} = -\frac{\partial H(p,q)}{\partial q}$$

$$\dot{q} = \frac{\partial H(p,q)}{\partial p}$$

抽象地写就是

$$\begin{pmatrix}\dot{p}\\ \dot{q}\end{pmatrix} = \hat{H}\begin{pmatrix}p\\ q\end{pmatrix}$$

这里的 $\hat{H}$ 就是一个“操作规则”， 或者说，一个算符。这个算符把函数 (p,q)映射为了函数 $(\dot{p},\dot{q})$。注意一般来说这个算符是非线性的。

抽象解

虽然 $\hat{H}$ 这个算符一般是非线性的，已经复杂到无法用矩阵来描述了，但这不妨碍我们抽象地把

$$\begin{pmatrix} \dot{p} \\ \dot{q} \end{pmatrix} = \hat{H} \begin{pmatrix} p \\ q \end{pmatrix}$$

进行“积分”，得到

$$\begin{pmatrix} p(t) \\ q(t) \end{pmatrix} = e^{\hat{H}t} \begin{pmatrix} p(0) \\ q(0) \end{pmatrix}$$

这里的

$$e^{\hat{H}t} := \sum_{n=0}^{\infty} \frac{1}{n!} \hat{H}^n t^n.$$

例子：谐振子

一维谐振子的哈密顿量可以写成：

$$H(p,q) = \frac{1}{2}p^2 + \frac{1}{2}q^2$$

其中 q 是位移，p 是动量。
(请自行根据量纲补充合适的单位，这些不是我们讨论的重点)

例子：谐振子

哈密顿方程为：

$$\dot{p} = -q$$

$$\dot{q} = p$$

简单起见，假设初始条件为 $p(0) = 0,\ q(0) = 1$。我们想求出以后的 $p(t), q(t)$是如何演化的。

假装我们的高数已经下车，但是C语言刚刚上车。准备用数值方法求解这个问题。

(我其实主要是针对对没有解析解的情况，只是为了说明问题方便而用你们熟悉的谐振子作为例子)

C语言来了

```
int N = 10000;
double dt = 0.01, p = 0, q = 1;
for(int i = 0; i < N; i++){
 p = p - q * dt;
 q = q + p * dt;
}
printf("%f %f %f\n", N*dt, p, q);
```

实测输出结果:

100.000000 0.506013 0.865060

稍稍换一下次序

```
int N = 10000;
double dt = 0.01, p = 0, q = 1;
for(int i = 0; i < N; i++){
 q = q + p * dt;
 p = p - q * dt;
}
printf("%f %f %f\n", N*dt, p, q);
```

实测输出结果:

```
100.000000 0.506013 0.860000
```

前面两段代码为何会给出不同的结果?

蛙跳法

跟严格解 $p(100) = -\sin 100 = 0.5063656$,
$q(100) = \cos(100) = 0.86231887$ 做比较，我们发现结果还算是相当不错的。

这种用哈密顿方程交替更新 p, q 的方法叫做“蛙跳法” (leapfrog method)。它通常能够用很低的内存消耗和算法复杂度给出较好近似结果。

思考题

用级数展开大法证明：

$$e^{(\hat{H}_1+\hat{H}_2)\Delta t} \approx e^{\frac{\hat{H}_1\Delta t}{2}} e^{\hat{H}_2\Delta t} e^{\frac{\hat{H}_1\Delta t}{2}}$$

误差至多是 $O(\Delta t^3)$ 的量级。
（推导时要注意 $\hat{H}_1, \hat{H}_2$ 一般是不对易的，即 $\hat{H}_1\hat{H}_2 \neq \hat{H}_2\hat{H}_1$）

蛙跳法的原理

把谐振子的哈密顿量拆成两部分的和 $H = H_1 + H_2$，其中 $H_1 = \frac{1}{2}p^2$, $H_2 = \frac{1}{2}q^2$。然后利用前面的近似

$$e^{(\hat{H}_1+\hat{H}_2)\Delta t} \approx e^{\frac{\hat{H}_1\Delta t}{2}} e^{\hat{H}_2\Delta t} e^{\frac{\hat{H}_1\Delta t}{2}}$$

求解谐振子的 (p, q) 的演化。

如果取 $H_2 = \frac{1}{2}p^2$, $H_1 = \frac{1}{2}q^2$，结果会怎样?

（注意：用H_1和 H_2单独作为哈密顿量时，程序其实在进行严格求解。也就是误差完全来自于上述算符等式的 $O(\Delta t^3)$误差。）

蛙跳法的原理

当你把很多步连起来的时候，

$$e^{(\hat{H}_1+\hat{H}_2)\Delta t}e^{(\hat{H}_1+\hat{H}_2)\Delta t}e^{(\hat{H}_1+\hat{H}_2)\Delta t}\ldots \approx e^{\frac{\hat{H}_1\Delta t}{2}}e^{\hat{H}_2\Delta t}e^{\hat{H}_1\Delta t}e^{\hat{H}_2\Delta t}e^{\hat{H}_1\Delta t}\ldots$$

这就是数值计算中的蛙跳法。

思考题

你能把前面的程序加以简单改进，使得在几乎完全相同的计算量下，结果的精度得以大大改善吗?

最后：请欣赏下6阶3重蛙跳

$$
\begin{aligned}
e^{(\hat{A}+\hat{B}+\hat{C})dt} = \; & e^{c_3\hat{A}dt/2} e^{c_3\hat{B}dt/2} e^{c_3\hat{C}dt} e^{c_3\hat{B}dt/2} e^{(c_3+c_2)\hat{A}dt/2} \\
& \times e^{c_2\hat{B}dt/2} e^{c_2\hat{C}dt} e^{c_2\hat{B}dt/2} e^{(c_2+c_1)\hat{A}dt/2} \\
& \times e^{c_1\hat{B}dt/2} e^{c_1\hat{C}dt} e^{c_1\hat{B}dt/2} e^{(c_1+c_0)\hat{A}dt/2} \\
& \times e^{c_0\hat{B}dt/2} e^{c_0\hat{C}dt} e^{c_0\hat{B}dt/2} e^{(c_0+c_1)\hat{A}dt/2} \\
& \times e^{c_1\hat{B}dt/2} e^{c_1\hat{C}dt} e^{c_1\hat{B}dt/2} e^{(c_1+c_2)\hat{A}dt/2} \\
& \times e^{c_2\hat{B}dt/2} e^{c_2\hat{C}dt} e^{c_2\hat{B}dt/2} e^{(c_2+c_3)\hat{A}dt/2} \\
& \times e^{c_3\hat{B}dt/2} e^{c_3\hat{C}dt} e^{c_3\hat{B}dt/2} e^{c_3\hat{A}dt/2} \\
& + O(dt^7) \; ,
\end{aligned}
$$

$$
\begin{aligned}
c_1 &= -1.17767998417887\ldots \; , \\
c_2 &= 0.235573213359357\ldots \; , \\
c_3 &= 0.784513610477560\ldots \; , \\
c_0 &= 1 - 2(c_1 + c_2 + c_3) \; .
\end{aligned}
$$

讨论：多体模拟代码

关键问题：当两个质点非常接近时，它们位置和速度都变化得非常快，以至于需要取很短的步长才能准确模拟。

Homework

- 写一个多质点在引力作用下运动的代码。如果觉得三维空间有点困难，可以写二维空间的。