

间断有限元第三次作业报告

九所 韩若愚

2022.3.31

1 题目

Considering the following Burgers equation

$$\begin{cases} u_t + (\frac{1}{2}u^2)_x = 0, & -1 \leq x \leq 1, \\ u(x, 0) = \frac{2}{3} + \frac{1}{3}\sin(\pi x), \end{cases} \quad (1)$$

with periodic boundary condition. Code up the DG scheme for P^1 and P^2 , together with 2nd and 3rd order SSP RK method in time, respectively. Use uniform mesh is space.

1. Take the final time at $T = 0.4$. Show error tables of the L^2 error, L^∞ error and $|\int_{-1}^1 (u - u_h)\phi dx|$ with $\phi(x) = \cos(x)$.
2. Take $T = 1.5$. Show pictures of the exact solution and the numerical solution.

2 算法

2.1 真解

由于方程是 Burgers 方程, 所以真解 u 沿特征线 $\frac{dx}{dt} = u$ 不变。而特征线的斜率刚好是 u , 所以特征线是直线。于是真解为 $u(x, t) = u(x - u(\xi, 0)t, 0)$, 其中 ξ 为经过点 (x, t) 的特征线与 $t = 0$ 的交点。这个真解是隐式给出的, 为了得到真解, 需要得到经过点 (x, t) 的特征线的斜率 $u(\xi, 0)$ 。

可以使用牛顿迭代法求 ξ 。因为 (x, t) 和 $(\xi, 0)$ 在同一条直线上，特征线方程 $x = u(\xi, 0)t + \xi$ 可改写为：

$$f(\xi) := x - u_0(\xi)t - \xi = 0$$

其中 $u_0(x) = u(x, 0)$ 。于是可以构造迭代公式：

$$\xi^{(n+1)} = \xi^{(n)} - \frac{f(\xi^{(n)})}{f'(\xi^{(n)})}$$

其中 $\xi^{(n)}$ 表示第 n 个迭代值。

2.2 DG 格式

首先对单元 $[-1, 1]$ 进行均匀剖分。假设将区间均匀剖分为 n 份，令：

$$0 = x_{\frac{1}{2}} < x_{\frac{3}{2}} < \cdots < x_{n-\frac{1}{2}} < x_{n+\frac{1}{2}} = 1 \quad (2)$$

则第 j 个区间为： $I_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ ，每个区间的长度都为 $h = \frac{2}{n}$ 。记 $x_{j+1/2}^- = \lim_{x \in I_j, x \rightarrow x_{j+1/2}} x$ ， $x_{j+1/2}^+ = \lim_{x \in I_{j+1}, x \rightarrow x_{j+1/2}} x$ 。

假设对固定的时间 t ，所求数值解 $u_h(\cdot, t)$ 存在的空间为： $V_h^k := \{v : v|_{I_j} \in P^k(I_j), j = 1, \dots, n\}$ ，其中 k 为给定常数， $P^k(I_j)$ 为定义在 I_j 上的最高次项不超过 k 次的多项式空间。并假设检验函数 $v \in V_h^k$ ，用 v 乘以方程两端并在 I_j 上积分。由于方程中通量函数 $f(u) = \frac{u^2}{2}$ 是非线性的，在构造数值格式时需要要求当 $k = 0$ 时，格式退化为一阶单调 FD 格式，于是空间离散后的半 DG 格式为：

$$\begin{aligned} & \int_{I_j} u_t v \, dx - \int_{I_j} \left(\frac{u^2}{2}\right) v_x \, dx \\ & + \hat{f}_{j+1/2} v(x_{j+1/2}^-) - \hat{f}_{j-1/2} v(x_{j-1/2}^+) = 0, \quad j = 1, \dots, n \end{aligned} \quad (3)$$

其中 $\hat{f}_{j+1/2}$ 是单调数值通量，保证在 $k = 0$ 时格式退化为 1 阶单调格式。 $\hat{f}_{j+1/2} = \hat{f}(u_{j+1/2}^-, u_{j+1/2}^+)$ 。

在 I_j 上取定一组 $P^k(I_j)$ 的基底 $\{\phi_j^l\}_{l=0}^k$ ，则数值解 u_h 在 I_j 上表示为： $u_h(x, t) = \sum_{j=1}^n \sum_{l=0}^k u_j^l(t) \phi_j^l(x)$ ，求解 u_h 即求解系数 u_j^l ， $j = 1, \dots, n$ ， $l =$

$0, \dots, k$ 。令检验函数 $v = \phi_j^m$, $m = 0, \dots, k$, 则方程组 (3) 变为:

$$\begin{aligned} & \sum_{l=0}^k \int_{I_j} \phi_j^l \phi_j^m dx u_j^l - \frac{1}{2} \sum_{p,q=0}^k \int_{I_j} \phi_j^p \phi_j^q (\phi_j^m)_x dx u_j^p u_j^q \\ & + \hat{f}(\sum_{l=0}^k u_j^l \phi_j^l(x_{j+1/2}), u_{j+1/2}^+) \phi_j^m(x_{j+1/2}) \\ & - \hat{f}(u_{j-1/2}^-, \sum_{l=0}^k u_j^l \phi_j^l(x_{j-1/2})) \phi_j^m(x_{j-1/2}) = 0, \quad j = 1, \dots, n \end{aligned} \quad (4)$$

这是关于向量 $\mathbf{u}_j = (u_j^0, \dots, u_j^k)$ 的 m 维方程组。

为便于求解, 假设参考单元 $I = [-1, 1]$, 对每个单元 I_j 都有一个到 I 的微分同胚 $\Phi_j : I_j \rightarrow I$, $\xi := \Phi_j(x) = \frac{2}{h}(x - x_{j-1/2}) - 1$ 。在参考单元上取定一组 $P^k(I)$ 的基底 $\{\phi^l\}_{l=0}^k$, 则由 Φ_j 将 ϕ^l 拉回到 I_j 上得到的函数组 $\{(\Phi_j^{-1})^* \phi^l\}$ 也是 $P^k(I_j)$ 的基底, 不妨就设为 $\{\phi_j^l\}$ 。于是每个单元 I_j 上的计算都可以在 I 上进行, 方程组 (4) 变为:

$$\begin{aligned} & \frac{h}{2} \sum_{l=0}^k \int_I \phi^l \phi^m d\xi u_j^l - \frac{1}{2} \sum_{p,q=0}^k \int_I \phi^p \phi^q (\phi^m)_\xi d\xi u_j^p u_j^q \\ & + \hat{f}(\sum_{l=0}^k u_j^l \phi^l(1), u_{j+1/2}^+) \phi^m(1) \\ & - \hat{f}(u_{j-1/2}^-, \sum_{l=0}^k u_j^l \phi^l(-1)) \phi^m(-1) = 0, \quad j = 1, \dots, n \end{aligned} \quad (5)$$

其中 $u_{j+1/2}^+ = \sum_{l=0}^k u_{j+1}^l \phi^l(-1)$, $u_{j-1/2}^- = \sum_{l=0}^k u_{j-1}^l \phi^l(1)$, j 为循环指标。

(5) 可以写为向量形式:

$$\begin{aligned} & \frac{h}{2} A \frac{d}{dt} \mathbf{u}_j = \frac{1}{2} \mathbf{u}_j B \mathbf{u}_j - C_j \\ & \frac{d}{dt} \mathbf{u}_j = \frac{2}{h} A^{-1} (\frac{1}{2} \mathbf{u}_j B \mathbf{u}_j - C_j) := L_j(\mathbf{u}_j) \end{aligned} \quad (6)$$

其中 A 为 $(k+1) \times (k+1)$ 维矩阵, $A_{ml} = \int_I \phi^m \phi^l d\xi$, B 为三阶张量, $B = \int_I \phi^p \phi^q (\phi^m)_\xi d\xi \omega^p \otimes \omega^q \otimes \omega^m$, 其中 ω^i 是取定的余切标架场, 在本例中即自然标架场的对偶标架场。 C_j 为 m 维向量, $C_{j,m} = \hat{f}(\sum_{l=0}^k u_j^l \phi^l(1), u_{j+1/2}^+) \phi^m(1) - \hat{f}(u_{j-1/2}^-, \sum_{l=0}^k u_j^l \phi^l(-1)) \phi^m(-1)$ 。于是 (6) 可以用 R-K 法求解。求解前还

需要得到 \mathbf{u}_j 的初值, 对初值 $u(x, 0)$ 做到 $P^k(I_j)$ 上的 L^2 投影: 对任意 $j = 1, 2, \dots, n$, $m = 0, 1, \dots, k$, 有

$$\int_{I_j} u(x, 0) \phi_j^m(x) dx = \int_{I_j} \sum_{l=0}^k u_j^l(0) \phi_j^l(x) \phi_j^m(x) dx.$$

SSPRK(2,2) 和 SSPRK(3,3) 格式分别如下:

SSPRK(2, 2) :

$$\begin{aligned} u^{(0)} &= u^n \\ u^{(1)} &= u^{(0)} + \Delta t F(u^{(0)}) \\ u^{n+1} &= \frac{1}{2} u^{(0)} + \frac{1}{2} u^{(1)} + \frac{1}{2} \Delta t F(u^{(1)}) \end{aligned}$$

SSPRK(3, 3) :

$$\begin{aligned} u^{(0)} &= u^n \\ u^{(1)} &= u^{(0)} + \Delta t F(u^{(0)}) \\ u^{(2)} &= \frac{3}{4} u^{(0)} + \frac{1}{4} u^{(1)} + \frac{1}{4} \Delta t F(u^{(1)}) \\ u^{n+1} &= \frac{1}{3} u^{(0)} + \frac{2}{3} u^{(2)} + \frac{2}{3} \Delta t F(u^{(2)}) \end{aligned}$$

同时为保证稳定性, 对时间步长 Δt 的选取还要满足 CFL 条件。

3 计算结果

为了方便计算, 取 $P^k(I)$ 上的基函数为 Legendre 函数。数值通量选为 Lax-Fridrichs 通量:

$$\hat{f}(u^-, u^+) = \frac{1}{2}(f(u^-) + f(u^+) - \alpha(u^- - u^+))$$

其中 $\alpha = \max_{u|_{I_j}} |f'(u)|$ 。为了满足 CFL 条件并观察到超收敛性, 将时间步长 Δt 取为 h^2 。算法使用 C++ 实现。

在终止时刻 $t = 0.4$ 时计算真解与数值解的 L^2 范数、 L^∞ 范数误差和误差 $|\text{int}_{-1}^1 (u - u_h) \cos(x) dx|$ 。 P^1, P^2 元的误差表如下 (保留三位小数):

当终止时刻 $T = 1.5$ 时, 在 $x = 0$ 处特征线会相交, 出现激波。当 $n = 80$ 时得到的真解 u 和数值解 u_h 的图像如下:

表 1: P^1

n	L^2 error	order	L^∞ error	order	H^{-k} error	order
10	1.005e-2		2.670e-2		1.297e-4	
20	2.678e-3	1.908	7.895e-3	1.758	2.410e-5	2.428
40	6.941e-4	1.948	2.126e-3	1.893	3.410e-6	2.821
80	1.765e-4	1.975	5.537e-4	1.941	4.464e-7	2.933
160	4.448e-5	1.988	1.413e-4	1.970	5.694e-8	2.971
320	1.116e-5	1.995	3.566e-5	1.986	7.238e-9	2.976

表 2: P^2

n	L^2 error	order	L^∞ error	order	H^{-k} error	order
10	1.091e-3		4.615e-3		6.730e-6	
20	1.472e-4	2.900	7.674e-4	2.588	1.481e-7	5.506
40	1.914e-5	2.943	1.027e-4	2.902	3.382e-9	5.453
80	2.442e-6	2.970	1.383e-5	2.893	1.655e-10	4.353
160	3.082e-7	2.986	1.759e-6	2.975	4.036e-11	2.036
320	3.896e-8	2.984	2.342e-7	2.909	5.900e-11	-0.548

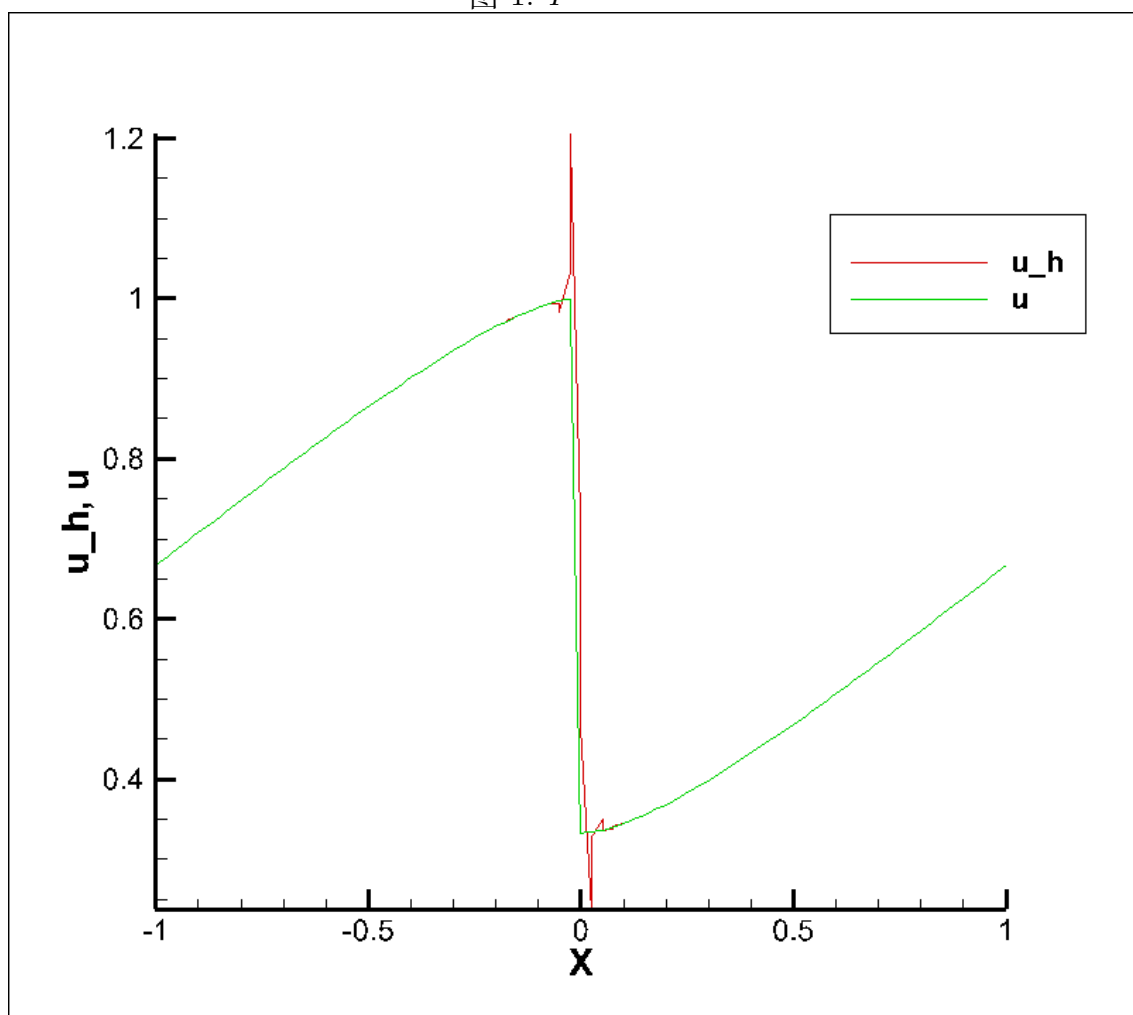
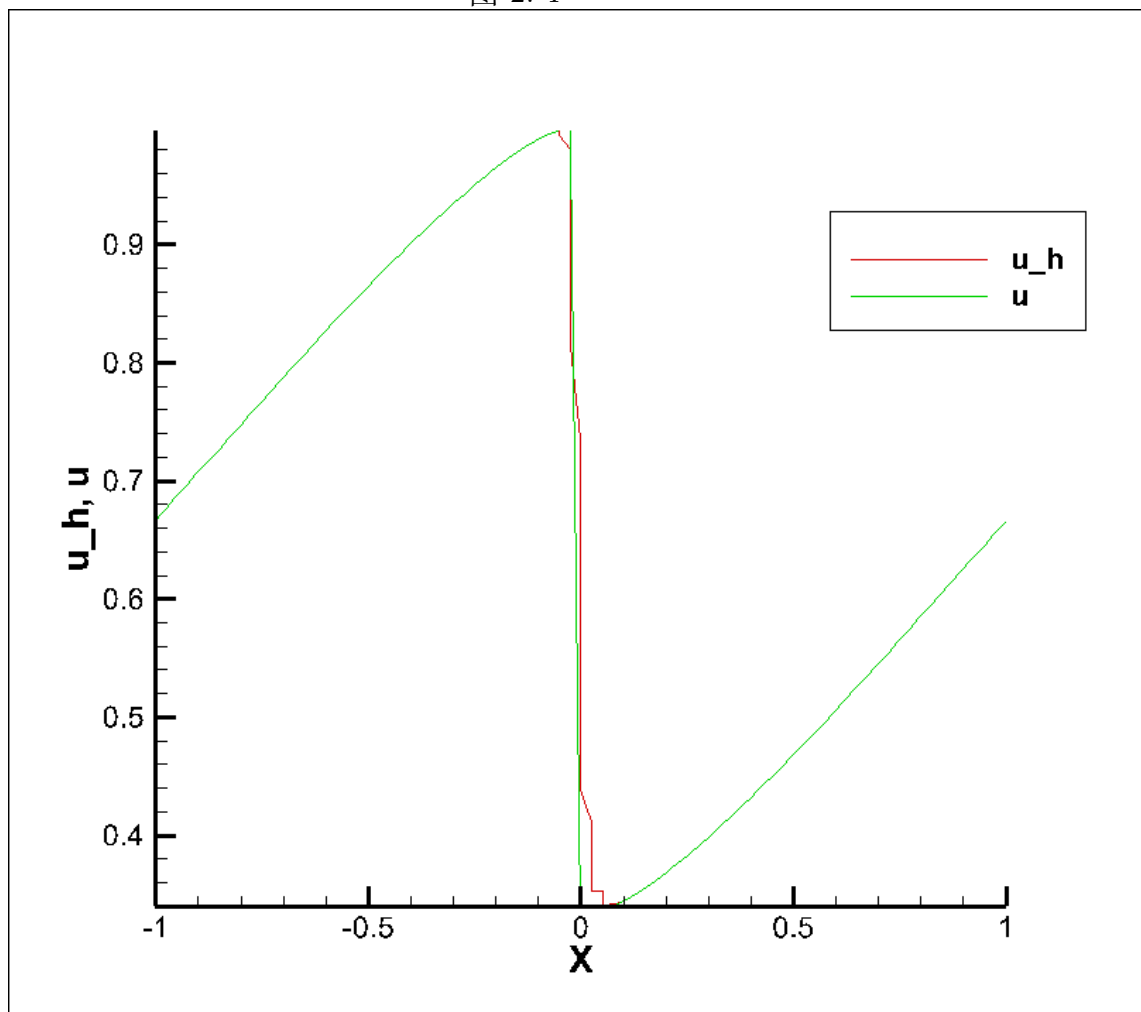
图 1: P^1 

图 2: P^2 

4 分析

可以看到当 k 分别等于 1,2 时, L^2 和 L^∞ 误差分别在 2, 3 附近。当 $k = 1$ 时, 负模误差接近 3, 出现了超收敛现象。而当 $k = 2$ 时负模的误差一直在减小, 在 $n \leq 80$ 时存在超收敛现象, 但是 $n > 80$ 后误差基本没有改变反而增长了。如果将 Δt 调小, 情况也没有太大改变。我的程序使用的是双精度浮点数, 在 C++ 中应该能达到 10^{308} 数量级, 按道理应该不会出现这种情况的, 但是到目前为止我还没有想明白问题在哪里。

而当 $T = 1.5$ 时, 计算结果在 $k = 0$ 时在 $= 0$ 处出现了明显振荡, 而当 $k = 1$ 时震荡并不是十分明显。

5 源代码

本次报告程序使用 C++ 编译。

```
1      #include <iostream>
2      #include <cmath>
3      #include <fstream>
4      using namespace std;
5
6      //先定义一些全局的变量
7      const int n = 80;    //划分单元个数
8      const int k = 1;    //多项式最高次项次数
9      const double h = (double) 2/n;    //空间步长
10     const double dt = (double) h * h;    //时间步
        长
11     const double pi = 3.1415926;
12
13     double p[n+1];    //节点位置,  $p_j = j * h = x_{\{j$ 
         $+1/2\}$ ,  $j = 0, 1, \dots, n$ 
14     //存储不变的系数矩阵
15
```



```
16      const double Legendre[5] =
           {-0.9061798459, -0.5384693101, 0.,
17      0.5384693101, 0.9061798459};
18      const double LegendreCo[5] =
           {0.2369268851, 0.4786286705, 0.5688888889,
19      0.4786286705, 0.2369268851};
20
21      //*****函数声明*****//
22      double u_0(double y);    //初值
23      double u_exact(double y, double t);    //真解
24      double f(double y); //通量函数
25      double phi(int l, double y);    //参考单元基函
           数
26      double** initial();    //计算初始时刻  $u_j$ 
27      double flux(double ul, double ur);    //数值通量
           计算
28      double** L(double** ut);    //用于计算RK的函
           数,  $u_t = F(u)$ 
29      double** RK22(double** un);    //2步二阶RK
30      double** RK33(double** un);
31      double** RK(int k, double** un);
32      //*****声明完毕*****//
33
34      int main()
35      {
36      int i, j, l;
37      double t, temp1, temp2, norm1, norm2, norm3,
           xi;
38      double T = 1.5;
39      double** u1 = new double* [n];
40      double** u2 = new double* [n];
```

```
41     for (i=0; i<n; i++)
42     {
43         u1[i] = new double [k+1];
44         u2[i] = new double [k+1];
45     }
46     for (j=0; j<=n; j++)
47     {
48         p[j] = j * h - 1;
49     }
50
51
52     u1 = initial();
53     t = 0;
54     while(t<T - 1e-10)
55     {
56
57         t = t + dt;
58         u2 = RK(k, u1);
59
60         u1 = u2;
61         cout<<t<<endl;
62     }
63
64     /*
65     norm1 = 0;
66     norm2 = 0;
67     norm3 = 0;
68     for (j=1; j<=n; j++)
69     {
70
71         for (i=0; i<5; i++)
```

```
72     {
73         xi = Legendre[i];
74         temp1 = 0;
75         for (l=0; l<=k; l++)
76         {
77             temp1 = temp1 + u1[j-1][l] * phi(l,xi);
78         }
79
80         temp2 = h * (xi + 1) / 2. + p[j-1];
81         temp1 = u_exact(temp2,T) - temp1;
82
83         if (abs(temp1) > norm2)
84         {
85             norm2 = abs(temp1);
86         }
87
88         norm3 = norm3 + LegendreCo[i] * temp1 * cos(
            temp2);
89
90         temp1 = temp1 * temp1;
91         norm1 = norm1 + LegendreCo[i] * temp1;
92     }
93
94 }
95 norm1 = norm1 * h / 2.;
96 norm1 = sqrt(norm1);
97 norm3 = norm3 * h / 2.;
98 norm3 = abs(norm3);
99 cout<<"L2="<<norm1<<endl<<"Linf="<<norm2<<endl
    <<"Lphi="<<norm3<<endl; //*/
100
```

```
101     {
102         const char* fn = "DGLecture\\homework3\\output
        .plt";
103         remove(fn);
104         fstream f, f1;
105         f.open(fn, ios::out | ios::app);
106         f<<"VARIABLES="<<"X"<<" , "<<"u_h"<<" , "<<"u"<<
            endl;
107         for (j=1; j<=n; j++)
108         {
109             temp1 = 0;
110             temp2 = 0;
111             for (l=0; l<=k; l++)
112             {
113                 temp1 = temp1 + u1[j-1][l] * phi(l,-1);
114                 temp2 = temp2 + u1[j-1][l] * phi(l,1);
115             }
116             f<<"\t"<<p[j-1]<<"\t"<<temp1<<"\t"<<u_exact(p[
                j-1],T)<<endl;
117             f<<"\t"<<p[j]<<"\t"<<temp2<<"\t"<<u_exact(p[j
                ],T)<<endl;
118         }
119         f.close();
120     }
121
122     for (i=0; i<n; i++)
123     {
124         delete [] u1[i];
125         delete [] u2[i];
126     }
127     delete [] u1;
```

```
128         delete [] u2;
129
130         system("pause");
131     }
132
133     double u_0(double y)
134     {
135         return sin(pi*y) / 3.0 + 2.0 / 3.0 ;
136     }
137
138     double u_exact(double y, double t)
139     {
140         int time=1;
141         double ans, xi1=0, xi2=0.1;
142         double e = 1e-6;
143
144         //if (t==1.5) //1.5时刻发生激波，单独算
145         {
146             if (y < -1+2*t/3.0)
147             {
148                 y = y+2;
149             }
150         }
151
152         while(abs(xi1 - xi2)>e)
153         {
154             xi1 = xi2;
155             xi2 = xi1 + ( y - u_0(xi1) * t - xi1 ) / (t *
                pi * cos(pi*xi1)/3.0 + 1);
156             time++;
157             if (time > 10000)
```

```
158     {
159         cout<<"error"<<endl;
160         cout<<y<<endl;
161         break;
162     }
163 }
164 ans = u_0(xi2);
165
166 return ans;
167 }
168
169 double f(double y)
170 {
171     return y * y /2;
172 }
173
174 double phi(int l, double y)
175 {
176     if (l==0)
177     {
178         return 1;
179     }
180     else if (l == 1)
181     {
182         return y;
183     }
184     else if (l == 2)
185     {
186         return (3*y*y - 1)/2;
187     }
188     else{
```

```
189         return 0;
190     }
191 }
192
193 double** initial()
194 {
195     double ans, temp;
196     int j, l, m;
197     double** ut = new double* [n];
198     double* Bt = new double [n];
199     for (j=0; j<n; j++)
200     {
201         ut[j] = new double [k+1];
202     }
203
204     for (j=1; j<=n; j++)
205     {
206         for (m=0; m<=k; m++)
207         {
208             ans = 0;
209             for (l=0; l<5; l++)
210             {
211                 temp = h * (Legendre[l] + 1)/2 + p[j-1];
212                 ans = ans + LegendreCo[l] * u_0(temp) * phi(m,
213                     Legendre[l]);
214             }
215             ans = ans / 2;
216             Bt[m] = ans;
217         }
218
219         double A[3][3] = {{1,0,0},{0,3,0},{0,0,5}};
```

```
219         for (m=0; m<=k ;m++)
220         {
221             ut[j-1][m] = 0;
222             for (l=0; l<=k; l++)
223             {
224                 ut[j-1][m] = ut[j-1][m] + A[m][l] * Bt[l];
225             }
226         }
227
228     }
229
230     delete [] Bt;
231
232     return ut;
233 }
234
235 double flux(double ul, double ur)
236 {
237     double ans, alpha;
238     int i, l;
239     if ( ul <= ur)
240     {
241         //ans = f(ul); //Godnov
242         alpha = ur; //Lax-Friedrichs
243     }
244     else{
245         //ans = f(ul);
246         alpha = ul;
247     }
248
249     ans = f(ul) + f(ur) - alpha * (ur - ul);
```



```
250         ans = ans * 0.5;
251
252     return ans;
253 }
254
255 double** L(double** ut)
256 {
257     int i, j, l, m, p, q;
258     double ul, ur;
259     double** ans = new double* [n];
260     for (i=0; i<n; i++)
261     {
262         ans[i] = new double [k+1];
263     }
264
265     double A[3] = {1,3,5};
266     double B[3][3][3] =
267         {{ {0,0,0}, {0,0,0}, {0,0,0} },
268          { {2,0,0}, {0,2.0/3.0,0}, {0,0,2.0/5.0} },
269          { {0,2,0}, {2,0,4.0/5.0}, {0,4.0/5.0,0} } };
270
271     for (j=1; j<=n; j++)
272     {
273         for (m=0; m<=k; m++)
274         {
275             ans[j-1][m] = 0;
276
277             for (p=0; p<=k; p++)
278             {
279                 for (q=0; q<=k; q++)
```

```
278     {
279         ans[j-1][m] = ans[j-1][m] + ut[j-1][p] * B[m][
                p][q] * ut[j-1][q];
280     }
281 }
282 ans[j-1][m] = ans[j-1][m] / 2;
283
284 //计算第一个数值通量
285 {
286     ul = 0;
287     ur = 0;
288     q = j;
289     if (q == n)
290     {
291         q = 0;
292     }
293     for (l=0; l<=k; l++)
294     {
295         ul = ul + ut[j-1][l] * phi(l,1);
296         ur = ur + ut[q][l] * phi(l,-1);
297     }
298     ans[j-1][m] = ans[j-1][m] - flux(ul,ur) * phi(
                m,1);
299
300 }
301
302 //计算第二个数值通量
303 {
304     ul = 0;
305     ur = 0;
306
```

```
307         p = j - 2;
308         if ( p == -1)
309         {
310             p = n - 1;
311         }
312         for ( l=0; l<=k; l++)
313         {
314             ul = ul + ut[p][l] * phi(l,1);
315             ur = ur + ut[j-1][l] * phi(l,-1);
316         }
317         ans[j-1][m] = ans[j-1][m] + flux(ul,ur) * phi(
            m,-1);
318     }
319
320     ans[j-1][m] = ans[j-1][m] * A[m] / h;
321 }
322 }
323
324 return ans;
325 }
326
327 double** RK22(double** un)
328 {
329     int j,l,m;
330     double ul;
331     double** ans = new double* [n];
332     for (j=0; j<n; j++)
333     {
334         ans[j] = new double [k+1];
335     }
336     double** u0 = new double* [n];
```

```
337     double** u1 = new double* [n];
338     double** u2 = new double* [n];
339
340     for (j=0; j<n; j++)
341     {
342         u0[j] = new double [k+1];
343         u1[j] = new double [k+1];
344         u2[j] = new double [k+1];
345     }
346
347     for (j=1; j<=n; j++)
348     {
349         for (l=0; l<=k; l++)
350         {
351             u0[j-1][l] = un[j-1][l];
352         }
353     }
354
355     u1 = L(u0);
356     for (j=1; j<=n; j++)
357     {
358         for (l=0; l<=k; l++)
359         {
360             u1[j-1][l] = u1[j-1][l] * dt + u0[j-1][l];
361         }
362     }
363
364
365     u2 = L(u1);
366     for (j=1; j<=n; j++)
367     {
```

```
368     for (l=0; l<=k; l++)
369     {
370         u2[j-1][l] = u0[j-1][l] * 0.5 + u1[j-1][l] *
            0.5 + u2[j-1][l] * 0.5 * dt;
371         ans[j-1][l] = u2[j-1][l];
372     }
373 }
374
375
376 delete [] u0;
377 delete [] u1;
378 delete [] u2;
379
380 return ans;
381 }
382
383 double** RK33(double** un)
384 {
385     int j,l,m;
386     double ul;
387     double** ans = new double* [n];
388     for (j=0; j<n; j++)
389     {
390         ans[j] = new double [k+1];
391     }
392     double** u0 = new double* [n];
393     double** u1 = new double* [n];
394     double** u2 = new double* [n];
395     double** u3 = new double* [n];
396
397     for (j=0; j<n; j++)
```

```
398     {
399         u0[j] = new double [k+1];
400         u1[j] = new double [k+1];
401         u2[j] = new double [k+1];
402         u3[j] = new double [k+1];
403     }
404
405     for (j=1; j<=n; j++)
406     {
407         for (l=0; l<=k; l++)
408         {
409             u0[j-1][l] = un[j-1][l];
410         }
411     }
412
413     u1 = L(u0);
414     for (j=1; j<=n; j++)
415     {
416         for (l=0; l<=k; l++)
417         {
418             u1[j-1][l] = u1[j-1][l] * dt + u0[j-1][l];
419         }
420     }
421
422     u2 = L(u1);
423     for (j=1; j<=n; j++)
424     {
425         for (l=0; l<=k; l++)
426         {
427             u2[j-1][l] = u0[j-1][l] * 3 / 4 + u1[j-1][l]
```

```
        /4 + u2[j-1][1] * dt / 4;
429     }
430 }
431
432     u3 = L(u2);
433     for (j=1; j<=n; j++)
434     {
435         for (l=0; l<=k; l++)
436         {
437             u3[j-1][1] = u0[j-1][1] / 3 + 2 * u2[j-1][1] /
                3 + 2 * dt * u3[j-1][1] / 3;
438             ans[j-1][1] = u3[j-1][1];
439         }
440     }
441
442
443     delete [] u0;
444     delete [] u1;
445     delete [] u2;
446     delete [] u3;
447
448     return ans;
449 }
450
451 double** RK(int k, double** un)
452 {
453     double** u2;
454     if ( k == 1 )
455     {
456         u2 = RK22(un);
457     }
```

```
458         else if (k == 2)
459             {
460                 u2 = RK33(un);
461             }
462         else{
463             ;
464         }
465
466         return u2;
467     }
```