**四阶Runge-Kutta射击法求解边界层方程相似解**

丁旺 张睿谦 张浩

1. **问题描述**

在边界层附近，有以下方程，

|  |  |  |
| --- | --- | --- |
|  |  | （1） |

存在某个，使得，其中，根据连续性方程可得，

|  |  |  |
| --- | --- | --- |
|  |  | （2） |

因为当时，与壁面上流函数的值有关，所以设，所以，

上式化为，

|  |  |  |
| --- | --- | --- |
|  |  | （3） |

将（3）式带入边界层方程可得

|  |  |  |
| --- | --- | --- |
|  |  | （4） |

其中，

|  |  |  |
| --- | --- | --- |
|  |  | （5） |
|  |  | （6） |

要使得上式成为常微分方程与不能依赖于，所以仅对某些特殊形式的外流，才存在相似解。

由（5）（6）可知，

|  |  |  |
| --- | --- | --- |
|  |  | （7） |

积分后可得，

|  |  |  |
| --- | --- | --- |
|  |  | （8） |

将（8）式带入（6）式中可得，

|  |  |  |
| --- | --- | --- |
|  |  | （9） |

令，方程最后化为，

|  |  |  |
| --- | --- | --- |
|  |  | （10） |

解得微分方程（10），即可解得边界层方程。

1. **算法描述**

（1）给定初值，求解微分方程离散解。

令，所以有，

|  |  |  |
| --- | --- | --- |
|  |  | （11） |

假设时，，引入新的变量，，所以有，

|  |  |  |
| --- | --- | --- |
|  |  | （12） |

联立（11）(12)给定任意的初值，利用四阶Runge-Kutta即可进行求解，

（2）对初值进行修正

因为，

|  |  |  |
| --- | --- | --- |
|  |  | （13） |
|  |  | （14） |

所以以进行迭代，将新修正得到的再进行计算，当时，停止迭代，即可获得的离散解。

1. **程序代码**

C++程序源代码

#include<iostream>

#include<fstream>

#include<cmath>

using namespace std;

int main()

{

double Max\_Eta = 8;//Eta范围

int N = 1000;//节点个数

double h = Max\_Eta / N;//步长

double \*f1, \*f2, \*f3, \*f4, \*f5, \*f6, \*Eta;

double K1, K2, K3, K4, L1, L2, L3, L4, M1, M2, M3, M4, N1, N2, N3, N4, P1, P2, P3, P4, Q1, Q2, Q3, Q4;//龙格库塔参数

double S = 1.1, S\_new = 1, S\_old=0, m = 0;//m=-0.05,0,0.2,1

int i;

f1 = new double[N + 1];

f2 = new double[N + 1];

f3 = new double[N + 1];

f4 = new double[N + 1];

f5 = new double[N + 1];

f6 = new double[N + 1];

Eta = new double[N + 1];//分配空间

f1[0] = 0, f2[0] = 0, f3[0] = S, f4[0] = 0, f5[0] = 0, f6[0] = 1;//定义初值

for (i = 0; i <= N; i++)

{

Eta[i] = i\*h;

}//横坐标

do

{

f3[0] = S;

for (i = 0; i<N; i++)

{

K1 = f2[i];

L1 = f3[i];

M1 = -0.5\*(m + 1)\*f1[i] \* f3[i] - m\*(1 - f2[i] \* f2[i]);

N1 = f5[i];

P1 = f6[i];

Q1 = -0.5\*(m + 1)\*(f3[i] \* f4[i] + f1[i] \* f6[i]) + 2 \* m\*f2[i] \* f5[i];

//

//

K2 = f2[i] + 0.5\*h\*L1;

L2 = f3[i] + 0.5\*h\*M1;

M2 = -0.5\*(m + 1)\*(f1[i] + 0.5\*h\*K1)\*(f3[i] + 0.5\*h\*M1) - m\*(1 - (f2[i] + 0.5\*h\*L1)\*(f2[i] + 0.5\*h\*L1));

N2 = f5[i] + 0.5\*h\*P1;

P2 = f6[i] + 0.5\*h\*Q1;

Q2 = -0.5\*(m + 1)\*((f3[i] + 0.5\*h\*M1)\*(f4[i] + 0.5\*h\*N1) + (f1[i] + 0.5\*h\*K1)\*(f6[i] + 0.5\*h\*Q1)) + 2 \* m\*(f2[i] + 0.5\*h\*L1)\*(f5[i] + 0.5\*h\*P1);

//

//

K3 = f2[i] + 0.5\*h\*L2;

L3 = f3[i] + 0.5\*h\*M2;

M3 = -0.5\*(m + 1)\*(f1[i] + 0.5\*h\*K2)\*(f3[i] + 0.5\*h\*M2) - m\*(1 - (f2[i] + 0.5\*h\*L2)\*(f2[i] + 0.5\*h\*L2));

N3 = f5[i] + 0.5\*h\*P2;

P3 = f6[i] + 0.5\*h\*Q2;

Q3 = -0.5\*(m + 1)\*((f3[i] + 0.5\*h\*M2)\*(f4[i] + 0.5\*h\*N2) + (f1[i] + 0.5\*h\*K2)\*(f6[i] + 0.5\*h\*Q2)) + 2 \* m\*(f2[i] + 0.5\*h\*L2)\*(f5[i] + 0.5\*h\*P2);

//

//

K4 = f2[i] + h\*L3;

L4 = f3[i] + h\*M3;

M4 = -0.5\*(m + 1)\*(f1[i] + h\*K3)\*(f3[i] + h\*M3) - m\*(1 - (f2[i] + h\*L3)\*(f2[i] + h\*L3));

N4 = f5[i] + h\*P3;

P4 = f6[i] + h\*Q3;

Q4 = -0.5\*(m + 1)\*((f3[i] + h\*M3 )\*(f4[i] + h\*N3) + (f1[i] + h\*K3)\*(f6[i] + h\*Q3)) + 2 \* m\*(f2[i] + h\*L3)\*(f5[i] + h\*P3);

//

f1[i + 1] = f1[i] + h\*(K1 + 2\*K2 + 2\*K3 + K4) / 6.0;

f2[i + 1] = f2[i] + h\*(L1 + 2\*L2 + 2\*L3 + L4) / 6.0;

f3[i + 1] = f3[i] + h\*(M1 + 2\*M2 + 2\*M3 + M4) / 6.0;

f4[i + 1] = f4[i] + h\*(N1 + 2\*N2 + 2\*N3 + N4) / 6.0;

f5[i + 1] = f5[i] + h\*(P1 + 2\*P2 + 2\*P3 + P4) / 6.0;

f6[i + 1] = f6[i] + h\*(Q1 + 2\*Q2 + 2\*Q3 + Q4) / 6.0;

//四阶龙格-库塔法

}

/\*

f1'=f2, f2'=f3, f3'=-0.5\*(m+1)\*f1\*f3-m\*(1-f2\*f2)

\*/

/\*

f4=df1/dS, f5=df2/dS, f6=df3/dS

f4'=f5, f5'=f6, f6'=-0.5\*(m+1)(f1\*f6+f3\*f4)+2\*m\*f2\*f5

\*/

S\_old = S;

S\_new = S + (1 - f2[N]) / f5[N];

S = S + (1 - f2[N]) / f5[N];

/\*

f5=df2/dS; (f2[N](应有值)-f2[N])/(S\_应有值-S）=f5[N];

S\_应有值=S+(1-f2[N])/f5[N]

\*/

cout << S << endl;

} while (fabs(S\_new - S\_old) > 0.001);

//

//输出

ofstream ofile("result\_1.txt");

for (int i = 0; i<= N; i++)

{

ofile << Eta[i] << '\t' << f2[i] << endl;;

}

ofile.close();

delete[] f1;

f1 = NULL;

delete[] f2;

f2 = NULL;

delete[] f3;

f3 = NULL;

delete[] f4;

f4 = NULL;

delete[] f5;

f5 = NULL;

delete[] f6;

f6 = NULL;

delete[] Eta;

Eta = NULL;

return 0;

}

1. **计算结果**

当m=-0.05,0,0.2,1时，以为横坐标，为纵坐标，绘制如下图像，

