第一部分 非线性方程组求解

第一部分 实验代码

1. %%
2. % 哈工大数值分析2020年秋研究生，上机实验
3. % 第一部分 | 非线性方程组求解
4. % 时间: 2020/10/15
5. % 学生: 20S\*\*\*\*\*\* \*\*\*
6. % ----------------------------------------------------------
7. % 1、【二分法】
8. % 2、【牛顿法】
9. % 3、【割线法】
10. % 4、【改进的牛顿法】
11. % 5、【拟牛顿法】
12. %%
13. % define the algorithm **for** which be used to solve the nolinear equation
14. % where the variable can be 0、1、2、3、4 , each one corresponds to the algorithm above
16. algorithm\_index = 0;
18. %%
19. % 方法一：二分法
20. % 题目：用二分法计算方程 sin(x)-pow(x,2)/2 = 0 在(1,2)内的根的近似值，要求ε=0.5\*10^(-5)
22. **if** algorithm\_index == 0
23. % 易知函数equation(x)在区间(1,2)内连续可导，且方程的根在区间(1,2)内存在且唯一
24. a = 1;
25. b = 2;
26. count = 0;
27. % 定义允许的误差
28. delta = 0.5\*10^(-5);
29. % 定义函数原型，方便用于不同的方程，提高程序应用的普遍性
30. syms f x;
31. f = sin(x)-x^2/2;
32. % 求解迭代解以及迭代次数
33. [answer,count] = dichotomy(a,b,count,delta,f);
34. fprintf("二分法求非线性方程，解为 %f 迭代了 %d 次\n",answer,count);
35. end
36. %%
37. % 方法二：牛顿法
38. % 题目：用牛顿法求解下列非线性方程的根，题目见实验报告册
40. **if** algorithm\_index == 1
41. % 定义3个方程的初始值
42. x1\_initial = 0.5;
43. x2\_initial = 1;
44. x3\_initial\_1 = 0.45;
45. x3\_initial\_2 = 0.65;
46. % 定义允许的误差以及最大的迭代次数
47. delta = 0.5\*10^(-5);
48. N = 100;
49. % 定义函数原型，方便用于不同的方程，提高程序应用的普遍性
50. syms f x;
51. % 求解方程1
52. count = 0;
53. f = x\*exp(x)-1;
54. [answer,count] = newton(x1\_initial,count,delta,f,N);
55. fprintf("牛顿法求非线性方程1，初始值为 %f 解为 %f 迭代了 %d 次\n",x1\_initial,answer,count);
56. % 求解方程2
57. count = 0;
58. f = x^3-x-1;
59. [answer,count] = newton(x2\_initial,count,delta,f,N);
60. fprintf("牛顿法求非线性方程2，初始值为 %f 解为 %f 迭代了 %d 次\n",x2\_initial,answer,count);
61. % 求解方程3
62. count = 0;
63. f = (x-1)^2\*(2\*x-1);
64. [answer,count] = newton(x3\_initial\_1,count,delta,f,N);
65. fprintf("牛顿法求非线性方程3，初始值为 %f 解为 %f 迭代了 %d 次\n",x3\_initial\_1,answer,count);
66. count = 0;
67. [answer,count] = newton(x3\_initial\_2,count,delta,f,N);
68. fprintf("牛顿法求非线性方程3，初始值为 %f 解为 %f 迭代了 %d 次\n",x3\_initial\_2,answer,count);
69. end
70. %%
71. % 方法三：割线法/多点迭代法
72. % 题目：用割线法求解下列非线性方程的根，题目见实验报告册
74. **if** algorithm\_index == 2
75. % 定义迭代初始点
76. x0\_initial = 0.4;
77. x1\_initial = 0.6;
78. % 定义允许的误差以及最大的迭代次数
79. delta = 0.5\*10^(-5);
80. N = 100;
81. count = 0;
82. % 定义函数原型，方便用于不同的方程，提高程序应用的普遍性
83. syms f x;
84. f = x\*exp(x)-1;
85. % 求解方程
86. [answer,count] = secant(x0\_initial,x1\_initial,count,delta,f,N);
87. fprintf("割线法求非线性方程，初始值x0为 %f 初始值x1为 %f 解为 %f 迭代了 %d 次\n",x0\_initial,x1\_initial,answer,count);
88. end
89. %%
90. % 方法四：改进的牛顿法
91. % 题目：用改进的牛顿法求解下列非线性方程的根，题目见实验报告册
93. **if** algorithm\_index == 3
94. % 定义迭代初始点
95. x\_initial = 0.55;
96. % 定义允许的误差以及最大的迭代次数
97. delta = 0.5\*10^(-5);
98. N = 100;
99. count = 0;
100. % 定义函数原型，方便用于不同的方程，提高程序应用的普遍性
101. syms f x;
102. f = (x-1)^2\*(2\*x-1);
103. % 求解方程
104. [answer,count] = advance\_newton(x\_initial,count,delta,f,N);
105. fprintf("改进的牛顿法求非线性方程，初始值为 %f 解为 %f 迭代了 %d 次\n",x\_initial,answer,count);
106. end
107. %%
108. % 方法五：拟牛顿法-秩1的拟牛顿法-逆Broyden法
109. % 题目：用拟牛顿法-逆Broyden求解下列非线性方程组的根，题目见实验报告册
110. % 注意，以下Fcn仅适用于3x3阶非线性方程组求解
111. % TODO 改成 NxN阶非线性方程组求解
113. **if** algorithm\_index == 4
114. % 定义迭代初始解向量
115. X\_initial = [1.0 1.0 1.0]';
116. % 定义允许的误差以及最大的迭代次数
117. delta = 0.5\*10^(-5);
118. N = 100;
119. count = 0;
120. % 定义函数原型，方便用于不同的方程，提高程序应用的普遍性
121. syms f\_1 f\_2 f\_3 x y z;
122. f\_1 = x\*y-z^2-1;
123. f\_2 = x\*y\*z+y^2-x^2-2;
124. f\_3 = exp(x)+z-exp(y)-3;
125. F = [f\_1 f\_2 f\_3]';
126. % 求系数矩阵A0
127. A\_initial = [diff(f\_1,x) diff(f\_1,y) diff(f\_1,z); diff(f\_2,x) diff(f\_2,y) diff(f\_2,z); diff(f\_3,x) diff(f\_3,y) diff(f\_3,z)];
128. x = X\_initial(1);
129. y = X\_initial(2);
130. z = X\_initial(3);
131. % 求系数矩阵H0
132. H\_initial = inv(eval(A\_initial));
133. % 求解方程
134. [answer,count] = quasi\_newton(X\_initial,H\_initial,count,delta,F,N);
135. fprintf("拟牛顿法求非线性方程组，初始值为 [%f %f %f]' 解为 [%f %f %f]' 迭代了 %d 次\n",X\_initial(1),X\_initial(2),X\_initial(3),answer(1),answer(2),answer(3),count);
136. end
137. %%
138. % 定义迭代函数实现二分法
139. function [answer,count] = dichotomy(next\_x,next\_y,count,delta,f)
140. answer = (next\_x+next\_y)/2;
142. x = next\_y;
143. **if** eval(f) ==0
144. answer = next\_y;
145. elseif next\_y - next\_x > 2\*delta
146. count = count+1;
147. x = answer;
148. f1 = eval(f);
149. x = next\_x;
150. f2 = eval(f);
151. **if** f1\*f2 > 0
152. next\_x = answer;
153. **else** % 包含了端点值为解的情况
154. next\_y = answer;
155. end
156. [answer,count] = dichotomy(next\_x,next\_y,count,delta,f);
157. end
158. end
159. %%
160. % 定义迭代函数实现牛顿法
161. function [answer,count] = newton(x\_initial,count,delta,f,N)
162. answer = x\_initial;
163. x = answer;
164. answer = answer - eval(f)/eval(diff(f));
165. count = count +1;
166. %**if** (abs(eval(f)) > delta)||(abs(eval(f)/eval(diff(f))) > delta)
167. **if** abs(eval(f)/eval(diff(f))) > delta
168. **if** count < N
169. [answer,count] = newton(answer,count,delta,f,N);
170. **else**
171. fprinf("Error, can not solve this equation in a limited count of %d",N);
172. end
173. end
174. end
175. %%
176. % 定义迭代函数实现割线法
177. function [answer,count] = secant(x0\_initial,x1\_initial,count,delta,f,N)
178. answer = x1\_initial;
179. answer\_k = x1\_initial;
180. answer\_k\_1 = x0\_initial;
181. x = answer\_k\_1;
182. f0 = eval(f);
183. x = answer\_k;
184. f1 = eval(f);
185. answer = answer\_k - f1/(f1-f0)\*(answer\_k-answer\_k\_1);
186. count = count +1;
187. **if** abs(f1/(f1-f0)\*(answer\_k-answer\_k\_1)) > delta
188. **if** count < N
189. [answer,count] = secant(answer\_k,answer,count,delta,f,N);
190. **else**
191. fprinf("Error, can not solve this equation in a limited count of %d",N);
192. end
193. end
194. end
195. %%
196. % 定义迭代函数实现改进的牛顿法
197. function [answer,count] = advance\_newton(x\_initial,count,delta,f,N)
198. answer = x\_initial;
199. x = answer;
200. answer = answer - 2\*eval(f)/eval(diff(f));
201. count = count +1;
202. fprintf("第%d次迭代，迭代解为%f 两次解的差值为%f delta为%f\n",count,answer,abs(2\*eval(f)/eval(diff(f))),delta);
203. %**if** (abs(eval(f)) > delta)||(abs(eval(f)/eval(diff(f))) > delta)
204. **if** abs(2\*eval(f)/eval(diff(f))) > delta
205. **if** count < N
206. [answer,count] = advance\_newton(answer,count,delta,f,N);
207. **else**
208. fprintf("Error, can not solve this equation in a limited count of %d",N);
209. end
210. end
211. end
212. %%
213. % 定义迭代函数实现拟牛顿法-逆Broyden法
214. function [answer,count] = quasi\_newton(X\_initial,H\_initial,count,delta,F,N)
215. answer = X\_initial;
216. x = X\_initial(1);
217. y = X\_initial(2);
218. z = X\_initial(3);
219. F\_i = eval(F);
220. answer = answer - H\_initial\*F\_i;
221. count = count + 1;
222. % 比较差值向量的 X(i+1) - X(i)的无穷范数与delta
223. **if** norm(H\_initial\*F\_i,inf) > delta
224. R =  - H\_initial\*eval(F);
225. x = answer(1);
226. y = answer(2);
227. z = answer(3);
228. F\_i\_1 = eval(F);
229. Y = F\_i\_1 - F\_i;
230. H\_initial = H\_initial + (R-H\_initial\*Y)\*(R'\*H\_initial)/(R'\*H\_initial\*Y);
231. **if** count < N
232. [answer,count] = quasi\_newton(answer,H\_initial,count,delta,F,N);
233. **else**
234. fprinf("Error, can not solve this equation set in a limited count of %d",N);
235. end
236. end
237. end
239. %% ------------------END OF THE **FILE**------------------

第一部分 实验结果

1. 二分法求非线性方程，解为 1.404415 迭代了 17 次
3. 牛顿法求非线性方程1，初始值为 0.500000 解为 0.567143 迭代了 4 次
4. 牛顿法求非线性方程2，初始值为 1.000000 解为 1.324718 迭代了 5 次
5. 牛顿法求非线性方程3，初始值为 0.450000 解为 0.500000 迭代了 4 次
6. 牛顿法求非线性方程3，初始值为 0.650000 解为 0.500000 迭代了 9 次
8. 割线法求非线性方程，初始值x0为 0.400000 初始值x1为 0.600000 解为 0.567143 迭代了 4 次
10. 牛顿下山法
11. 下山因子为2，初始值为0.55时，在100次迭代次数内不能达到精度要求
12. Error, can not solve **this** equation in a limited count of 100
14. 拟牛顿法求非线性方程组，初始值为 [1.000000 1.000000 1.000000]' 解为 [1.777672 1.423961 1.237471]' 迭代了 10 次

第二部分 高斯（列）主元消去法

第二部分 实验代码

1. %%
2. % 哈工大数值分析2020年秋研究生，上机实验
3. % 第二部分 | 线性方程组求解/高斯列主元消去法
4. % 时间: 2020/10/22
5. % 学生: 20S\*\*\*\*\*\* \*\*\*
6. % ----------------------------------------------------------
7. % 1、【高斯消去法】
8. % 2、【高斯列主元消去法】
10. % 若高斯消去法解线性方程组时，如果主元元素等于0，则消去法无法继续，或者主元元素接近于0，继续使用消去法将导致不稳定现象，
11. % 此时需要使用高斯列主元消去法
12. %%
13. % 定义要求解的方程组一
14. A = [10^(-8) 2 3; -1 3.712 4.623; -2 1.072 5.643];
15. b = [1 2 3];
16. % 定义要求解的方程组二
17. C = [4 -2 4; -2 17 10; -4 10 9];
18. d = [10 3 7];
19. %%
20. % 分别用高斯列主元消去法和高斯消去法求解方程组
21. [answer] = gauss\_elimination(A,b);
22. fprintf("高斯法 方程一 answer = [ %f %f %f ]\n",answer(1),answer(2),answer(3));
23. [answer] = gauss\_principal\_element\_elimination(A,b);
24. fprintf("高斯列主元法 方程一 answer = [ %f %f %f ]\n",answer(1),answer(2),answer(3));
25. [answer] = gauss\_elimination(C,d);
26. fprintf("高斯法 方程二 answer = [ %f %f %f ]\n",answer(1),answer(2),answer(3));
27. [answer] = gauss\_principal\_element\_elimination(C,d);
28. fprintf("高斯列主元法 方程二 answer = [ %f %f %f ]\n",answer(1),answer(2),answer(3));
29. %%
30. %定义函数实现高斯消去法
31. function [answer] = gauss\_elimination(A,b)
32. % 获取系数矩阵的阶数
33. [count] = size(A,1);
34. % 消元过程
35. **for** i = 1:count-1
36. **for** j = i:count-1
37. **for** n = 1:count-i
38. A(i+n,j+1) =  A(i+n,j+1)-A(i,j+1)\*A(i+n,i)/A(i,i);
39. end
40. end
41. **for** n = 1:count-i
42. b(i+n) = b(i+n)-b(i)\*A(i+n,i)/A(i,i);
43. end
44. end
45. % 回代求解过程
46. answer = zeros(count,1);
47. **for** i = 1:count
48. answer(count-i+1) = b(count-i+1);
49. **if** i>1
50. **for** j = 1:i-1
51. answer(count-i+1) = answer(count-i+1) - answer(count-i+1+j)\*A(count-i+1,count-i+1+j);
52. end
53. end
54. answer(count-i+1) = answer(count-i+1)/A(count-i+1,count-i+1);
55. end
56. end
57. %%
58. %定义函数实现高斯列主元消去法
59. function [answer] = gauss\_principal\_element\_elimination(A,b)
60. % 获取系数矩阵的阶数
61. [count] = size(A,1);
62. % 消元过程
63. **for** i = 1:count-1
64. % 每一次消元前进行列选主元
65. cursor = i; max = abs(A(i,i));
66. **for** m = i:count
67. **if** abs(A(m,i))>max
68. max = abs(A(m,i));
69. cursor = m;
70. end
71. end
72. % 列主元行交换
73. **if** cursor ~= i
74. row\_temp = A(i,:);
75. A(i,:) = A(cursor,:);
76. A(cursor,:) = row\_temp;
77. b\_temp = b(i);
78. b(i) = b(cursor);
79. b(cursor) = b\_temp;
80. end
81. % 消元
82. **for** j = i:count-1
83. **for** n = 1:count-i
84. A(i+n,j+1) =  A(i+n,j+1)-A(i,j+1)\*A(i+n,i)/A(i,i);
85. end
86. end
87. **for** n = 1:count-i
88. b(i+n) = b(i+n)-b(i)\*A(i+n,i)/A(i,i);
89. end
90. end
91. % 回代求解过程
92. answer = zeros(count,1);
93. **for** i = 1:count
94. answer(count-i+1) = b(count-i+1);
95. **if** i>1
96. **for** j = 1:i-1
97. answer(count-i+1) = answer(count-i+1) - answer(count-i+1+j)\*A(count-i+1,count-i+1+j);
98. end
99. end
100. answer(count-i+1) = answer(count-i+1)/A(count-i+1,count-i+1);
101. end
102. end
104. %% ------------------END OF THE **FILE**------------------

第二部分 实验结果

1. 高斯法 方程一 answer = [ -0.491058 -0.050886 0.367257 ]
2. 高斯列主元法 方程一 answer = [ -0.491058 -0.050886 0.367257 ]
3. 高斯法 方程二 answer = [ 0.196429 -0.892857 1.857143 ]
4. 高斯列主元法 方程二 answer = [ 0.196429 -0.892857 1.857143 ]

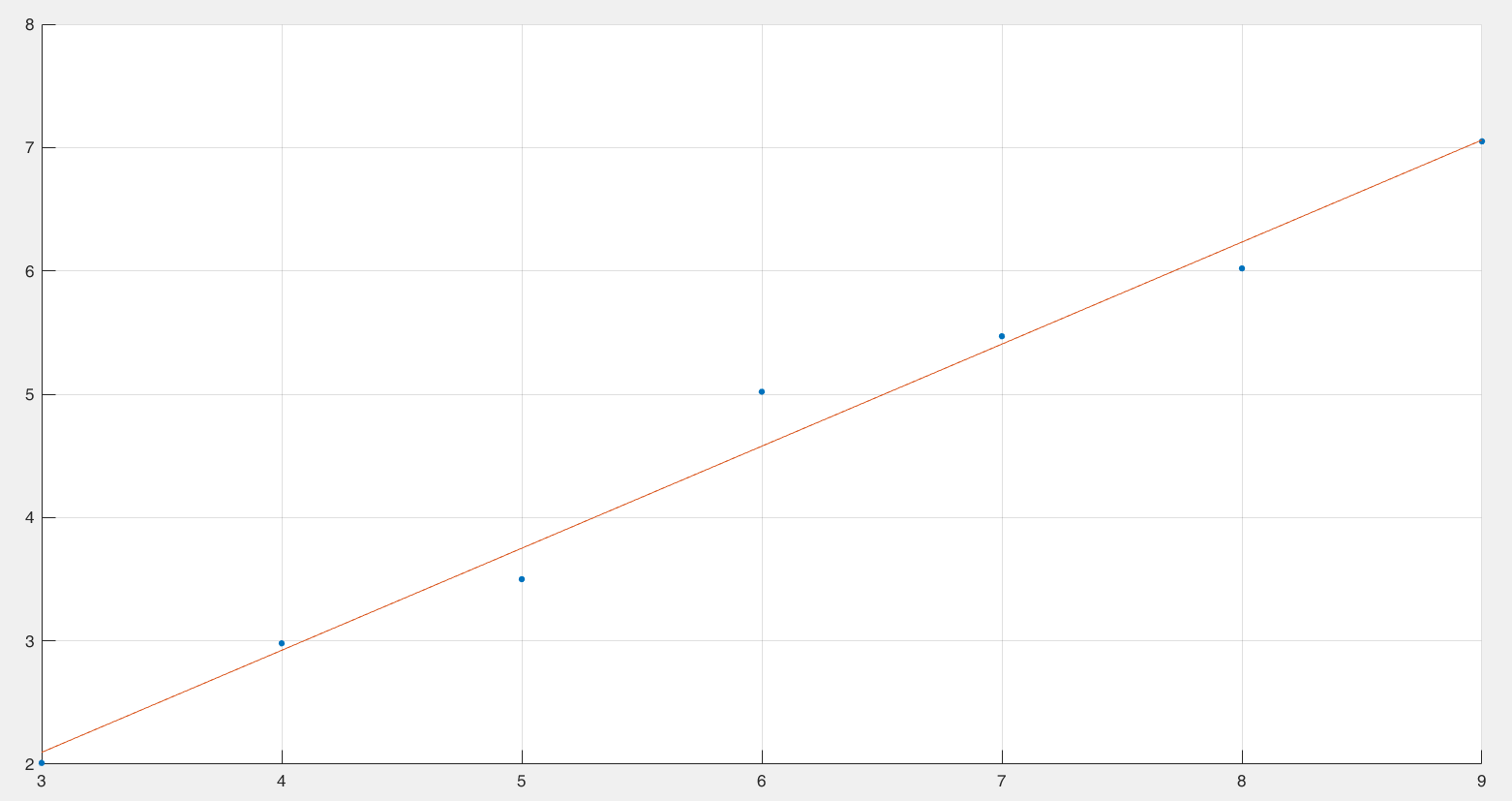
第三部分 多项式最小二乘拟合

第三部分 实验代码

1. %%
2. % 哈工大数值分析2020年秋研究生，上机实验
3. % 第三部分 | 最小二乘拟合/Least squares fitting
4. % 时间: 2020/10/29
5. % 学生: 20S\*\*\*\*\*\* \*\*\*
6. % ----------------------------------------------------------
7. % 1、【利用最小二乘法处理实验数据】
8. %%
9. % 定义拟合的函数模型及多项式的最高的幂数和数据点
10. syms f x;
11. r = 1;
12. X = [3 4 5 6 7 8 9];
13. Y = [2.01 2.98 3.50 5.02 5.47 6.02 7.05];
14. % 调用函数进行拟合
15. [answer] = Lsf(r,X,Y);
16. fprintf("多项式最小二乘拟合 系数分别为\n");
17. **for** i = 0:r
18. fprintf("  %f  ",answer(i+1));
19. end
20. fprintf("\n");
21. % 绘制图像比较拟合的结果
22. scatter(X,Y,15,'filled');
23. hold on;
24. grid on;
25. num = 0;
26. fit\_x = zeros(1,ceil((max(X)-min(X))/0.01)+1);
27. fit\_y = zeros(1,ceil((max(X)-min(X))/0.01)+1);
28. **for** m = min(X):0.01:max(X)
29. num = num + 1;
30. fit\_x(num) = m;
31. **for** j = 0:r
32. fit\_y(num) = fit\_y(num) + answer(j+1)\*fit\_x(num)^j;
33. end
34. end
35. plot(fit\_x,fit\_y);
36. %%
37. % 定义函数求解多项式最小二乘拟合系数
38. function [answer] = Lsf(r,X,Y)
39. % 定义法方程系数矩阵及方程右侧向量
40. A = zeros(r+1);
41. b = zeros(r+1,1);
42. % 由最小二乘原则/最佳平方逼近求出法方程各个系数/多元函数求极值
43. **for** i = 0:r
44. **for** j = 0:r
45. **for** n = 0:length(X)-1
46. A(i+1,j+1) = A(i+1,j+1) + 1\*X(n+1)^(i+j);
47. end
48. end
49. end
50. % 算出法方程右侧向量
51. **for** i = 0:r
52. **for** n = 0:length(X)-1
53. b(i+1) = b(i+1) + 1\*X(n+1)^i\*Y(n+1);
54. end
55. end
56. % 求出系数矩阵后，利用上个实验中定义的高斯列主元消去法求解非齐次线性方程组
57. [answer] = gauss\_principal\_element\_elimination(A,b);
58. end
60. %%
61. %定义函数实现高斯列主元消去法
62. function [answer] = gauss\_principal\_element\_elimination(A,b)
63. % 获取系数矩阵的阶数
64. [count] = size(A,1);
65. % 消元过程
66. **for** i = 1:count-1
67. % 每一次消元前进行列选主元
68. cursor = i; max = abs(A(i,i));
69. **for** m = i:count
70. **if** abs(A(m,i))>max
71. max = abs(A(m,i));
72. cursor = m;
73. end
74. end
75. % 列主元行交换
76. **if** cursor ~= i
77. row\_temp = A(i,:);
78. A(i,:) = A(cursor,:);
79. A(cursor,:) = row\_temp;
80. b\_temp = b(i);
81. b(i) = b(cursor);
82. b(cursor) = b\_temp;
83. end
84. % 消元
85. **for** j = i:count-1
86. **for** n = 1:count-i
87. A(i+n,j+1) =  A(i+n,j+1)-A(i,j+1)\*A(i+n,i)/A(i,i);
88. end
89. end
90. **for** n = 1:count-i
91. b(i+n) = b(i+n)-b(i)\*A(i+n,i)/A(i,i);
92. end
93. end
94. % 回代求解过程
95. answer = zeros(count,1);
96. **for** i = 1:count
97. answer(count-i+1) = b(count-i+1);
98. **if** i>1
99. **for** j = 1:i-1
100. answer(count-i+1) = answer(count-i+1) - answer(count-i+1+j)\*A(count-i+1,count-i+1+j);
101. end
102. end
103. answer(count-i+1) = answer(count-i+1)/A(count-i+1,count-i+1);
104. end
105. end
106. %% ------------------END OF THE **FILE**------------------

第三部分 实验结果

1. 多项式最小二乘拟合 系数分别为
2. 0.386429    0.827500



第四部分 龙贝格积分法

第四部分 实验代码

1. %%
2. % 哈工大数值分析2020年秋研究生，上机实验
3. % 第四部分 | 龙贝格积分法/Romberg Integral
4. % 时间: 2020/10/29
5. % 学生: 20S\*\*\*\*\*\* \*\*\*
6. % ----------------------------------------------------------
7. % 1、【龙贝格积分法计算以下积分的近似值】
9. %%
10. % 定义被积函数原型及精度要求
11. syms f x;
12. delta = 7\*10^(-6);
13. T = zeros(8);
14. % 求解定积分1
15. f = x^3;
16. a = 6; b = 100;m = 1;
17. % f = 4/(1+x^2);
18. % a = 0; b = 1;m = 1;
19. [answer,m] = Romberg(f,a,b,m,delta,T);
20. fprintf("\n求解定积分1，解为 %6f    m =  %d  验证解为%f\n-------------------------------------\n",answer, m, **int**(eval(f),x,a,b));
21. % 求解定积分2
22. f = sin(x)/x;
23. a = 0; b = 1;m = 1;
24. [answer,m] = Romberg(f,a,b,m,delta,T);
25. fprintf("\n求解定积分2，解为 %f    m =  %d  验证解为%f\n-------------------------------------\n",answer, m, **int**(eval(f),x,a,b));
26. % 求解定积分3
27. f = sin(x^2);
28. a = 0; b = 1;m = 1;
29. [answer,m] = Romberg(f,a,b,m,delta,T);
30. fprintf("\n求解定积分3，解为 %f    m =  %d  验证解为%f\n-------------------------------------\n",answer, m, **int**(eval(f),x,a,b));
31. %%
32. % 定义Romberg积分迭代函数
33. function [answer,m] = Romberg(f,a,b,m,delta,T)
34. **if** m == 1
35. x = a;
36. **if** x == 0
37. syms x;
38. fa = limit(eval(f),x,0);
39. **else**
40. fa = eval(f);
41. end
42. x = b;
43. fb = eval(f);
44. T(1,1) = 0.5\*(b-a)\*(fa+fb);
45. end
46. **for** i = 0:m
47. % 计算龙贝格T-数表
48. **if**  i == 0 % T型求积公式
49. sum = 0;
50. **for** j = 0:2^(m-1)-1
51. x = a + (b-a)/2^(m-1)\*(j+0.5);
52. **if** x == 0
53. syms x;
54. sum = sum + limit(eval(f),x,0);
55. **else**
56. sum = sum + eval(f);
57. end
58. end
59. T(m+1,i+1) = 0.5\*T(m,i+1)+0.5\*(b-a)/2^(m-1)\*sum;
60. **else** % 高阶求积公式
61. T(m+1,i+1) = (4^i\*T(m+1,i)-T(m,i))/(4^i-1);
62. end
63. end
64. % 计算龙贝格数表上的对角线最后两元素之差，与给定的精度进行比较
65. **if** abs(T(m+1,m+1)-T(m,m)) <= delta
66. answer = T(m+1,m+1);
67. fprintf("求解完成!  T-数表为\n\n");
68. **for** i = 0:m
69. **for** j = 0:i
70. fprintf("%f  ",T(i+1,j+1));
71. end
72. fprintf("\n");
73. end
74. **else**
75. % 未达到精度要求，继续迭代求解
76. [answer,m] = Romberg(f,a,b,m+1,delta,T);
77. end
78. end
79. %% ------------------END OF THE **FILE**------------------

第四部分 实验结果

1. 求解完成!  T-数表为
3. 47010152.000000
4. 30502295.000000  24999676.000000
5. 26375330.750000  24999676.000000  24999676.000000
7. 求解定积分1，解为 24999676.000000    m =  2  验证解为24999676.000000
8. -------------------------------------
9. 求解完成!  T-数表为
11. 0.920735
12. 0.939793  0.946146
13. 0.944514  0.946087  0.946083
14. 0.945691  0.946083  0.946083  0.946083
16. 求解定积分2，解为 0.946083    m =  3  验证解为0.946083
17. -------------------------------------
18. 求解完成!  T-数表为
20. 0.420735
21. 0.334070  0.305181
22. 0.315975  0.309944  0.310261
23. 0.311680  0.310249  0.310269  0.310269
24. 0.310620  0.310267  0.310268  0.310268  0.310268
26. 求解定积分3，解为 0.310268    m =  4  验证解为0.310268