|  |  |  |  |
| --- | --- | --- | --- |
| **学生学号** | 0121801100913 | **实验课成绩** |  |



**学 生 实 验 报 告 书**

|  |  |
| --- | --- |
| **实验课程名称** | 计算机数值分析 |
| **开 课 学 院** | 计算机科学与技术学院 |
| **指导教师姓名** | 戚欣 |
| **学 生 姓 名** | 张淳慷 |
| **学生专业班级** | 软件zy1802 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2019 | —— | 2020 | 学年 | 第 | 二 | 学期 |

课程名称：计算机数值分析

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | 插值方法 | | | 实验成绩 |  |
| 实验者 | 张淳慷 | 专业班级 | 软件zy1802 | 实验日期 | 2020年5月7日 |
| 1. 实验概述    1. 实验目的   通过设计、编制、调试2~3个多项式插值、拟合曲线的程序，加深对其数值计算方法及有关的基础理论知识的理解。   * 1. 实验要求   用编程语言实现拉格朗日（Lagrange）插值多项式、牛顿（Newton）插值、用线性函数拟合给定数据的程序。   * 1. 实验内容   用编程语言编程实现以下算法：   1. 已知插值节点序列，用拉格朗日（Lagrange）插值多项式计算的函数在点的近似值。 2. 已知插值节点序列，用牛顿（Newton）插值值多项式计算的函数在点的近似值。 3. 用线性函数拟合给定数据。    1. 实验器材 4. PC机。 5. 编程语言：python | | | | | |

|  |
| --- |
| 1. 实验设计、测试与分析    1. 问题：已知插值节点序列，用拉格朗日（Lagrange）插值多项式计算的函数在点的近似值。       1. 算法描述:   采用了简单的while与for循环，通过if语句判断将拉格朗日累乘时i=j的情况排除在外，最后得解。   * + 1. 程序变量说明   e示插值基函数,f表示最终的多项式结果,i为所选择x的下标   * + 1. 源程序代码及运行结果截图   **源代码：**  print('输入所有插值节点的x值，以空格分开',end ='：')  a = list(map(eval,input().split()))  print('输入所有插值节点的y值，以空格分开',end ='：')  b = list(map(eval,input().split()))  print("输入要求解的x",end = ':')  c = eval(input())  e = 1  f = 0  i = 0  while True:  for m in range(len(a)):  if i ==m:  continue  else:  d = (c-a[m])/(a[i]-a[m])  e = e\*d  f = f+e\*b[i]  e = 1  i = i+1  if i==int(len(a)):  break  print(f)  **运行结果截图：（以y=x\*\*2为例）：**     * 1. 问题：已知插值节点序列，用牛顿（Newton）插值值多项式计算的函数在点的近似值。      1. 算法描述   采用两个函数分别用于收集x与y的值和牛顿插值求和，后一个函数中，将所求出的差商添加到列表当中以供计算高次差商时使用。   * + 1. 程序变量说明   i表示i阶差商 j表示计算所需的n-j数字   * + 1. 源程序代码及运行结果截图   **源代码：**  def Data\_in(xlist,ylist):  list1=input("请输入已知的准确的对应的x,y值：").split(" ");  for i in range(0,len(list1)):  if i%2==0:  xlist.append(float(list1[i]));  else:  ylist.append(float(list1[i]));  return xlist,ylist;  def Newton\_insert(xlist,ylist,x\_in):  n=len(xlist);#n==6  temp\_y=ylist.copy();  for i in range(0,n): #0~n-1 分别代表0阶差商 ~n-1差商  if i==0:  continue;  else:  for j in range(i,n): #i阶差商需要计算 n-i个数  temp\_y[j]=(ylist[j]-ylist[j-1])/(xlist[j]-xlist[j-i]);  ylist=temp\_y.copy()  y\_out=ylist[0];  for k1 in range(1,n):  for k2 in range(0,k1):  ylist[k1]=ylist[k1]\*(x\_in-xlist[k2]);  y\_out+=ylist[k1];  return y\_out  DataX=[];  DataY=[];  DataX,DataY=Data\_in(DataX,DataY);  #print(DataX)  #print(DataY)  X=float(input("请输入要预测的x的值"));  Y=Newton\_insert(DataX, DataY, X);  print(Y)  **运行结果：**     * 1. 问题：用线性函数拟合给定数据。      1. 算法描述   根据正常的求ab的二元公式求出ab的值，然后代入matplotlib给出函数图   * + 1. 程序变量说明   C表示所有x的和，d表示所有y的和，e和h分别原来放xy的乘积以及x的平方   * + 1. 源程序代码及运行结果截图   **源代码：**  import numpy as np  import matplotlib.pyplot as plt  print('输入所有x值，以空格分开',end ='：')  a = list(map(eval,input().split()))  print('输入所有y值，以空格分开',end ='：')  b = list(map(eval,input().split()))  c = sum(a)  d = sum(b)  e = []  f=[]  for i in range(len(a)):  e.append(a[i]\*b[i])  for i in range(len(a)):  f.append(a[i]\*\*2)  g = (sum(a)\*sum(b)-len(a)\*sum(e))/(sum(a)\*\*2-len(a)\*sum(f))  h = (sum(f)\*sum(b)-sum(e)\*sum(a))/(sum(f)\*len(a)-sum(a)\*\*2)  print('所得的函数为p(x)={}+{}x'.format(h,g))  x = np.linspace(0,100)  y = h+g\*x  plt.plot(x,y)  plt.show()  **截图：** |

|  |
| --- |
| 1. 实验小结与心得体会   **实验小结：**  总的来说算法的难度只能说是基本的练习题的感觉，基本功得到了一定的锻练，通过算法具体实现了各个插值多项式的运用，感觉对于该类知识掌握的更加牢固了。 |

课程名称：计算机数值分析

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | 数值积分与微分 | | | 实验成绩 |  |
| 实验者 | 张淳慷 | 专业班级 |  | 实验日期 | xxxx年xx月xx日 |
| 1. 实验概述    1. 实验目的   通过设计、编制、调试2~3个数值积分与微分算法的程序，加深对其数值计算方法及有关的基础理论知识的理解。   * 1. 实验要求   用编程语言实现复化梯形积分、Romberg积分的程序。   * 1. 实验内容   用编程语言编程实现以下算法：   1. 用复化梯形公式的自动控制误差算法求积分。 2. Romberg积分算法求积分。    1. 实验器材 3. PC机。 4. 编程语言python | | | | | |

|  |
| --- |
| 1. 实验设计、测试与分析    1. 问题：用复化梯形公式的自动控制误差算法求积分。       1. 算法描述   基本的照方程写相应代码，通过while函数反复计算精度   * + 1. 源程序代码及运行结果截图   **源代码：**  import math  from scipy.misc import derivative  def sum\_fun\_x(x, func): # 累加项  return sum([func(each) for each in x])  def integral(a, b, n, func): # 梯形复化公式  h = (b - a)/float(n)  x = [a + i\*h for i in range(1, n)]  return h/2 \* (func(a) + 2 \* sum\_fun\_x(x, func) + func(b))  def func(x): # 函数  return math.pow(math.e,x)  def calculN(h,p): # 根据精度得到n  n = 2  tp = math.fabs(math.pow(h, 2) \* (derivative(func, b) - derivative(func, a)))  tp /= n \* n  while(tp > p):  tp \*= n \* n  n += 1  tp /= n \* n  return n  a, b = 0, 1 # 区间值  p = 10e-5 # 精度  n = calculN(b - a,p)  print(‘计算结果为{}’.format(integral(a, b, n, func)))  运行结果：     * 1. print(integral(a, b, n, func))问题：Romberg积分算法求积分。      1. 算法描述   通过if语句确定相应的公式选用，m用于判断需使用哪一个公式进行运算   * + 1. 程序变量说明   A为积分限，b为积分上限，eps表示精度，t表示复化梯形公式，s表示simpson公式 ，c表示cotes公式，r表示romberg公式，选用e\*\*-x为例子。   * + 1. 源程序代码及运行结果截图   **源代码：**  import math  a=0 # 积分下限  b=1 # 积分上限  eps=10\*\*-5 # 精度  T=[] # 复化梯形公式  S=[] # Simpson公式  C=[] # Cotes公式  R=[] # Romberg公式  def func(x): # 被积函数  y=math.exp(-x)  return y  def Romberg(a,b,eps,func):  h = b - a  T.append(h \* (func(a) + func(b)) / 2)  ep=eps+1  m=0  while(ep>=eps):  m=m+1  t=0  for i in range(2\*\*(m-1)-1):  t=t+func(a+(2\*(i+1)-1)\*h/2\*\*m)\*h/2\*\*m  t=t+T[-1]/2  T.append(t)  if m>=1:  S.append((4\*\*m\*T[-1]-T[-2])/(4\*\*m-1))  if m>=2:  C.append((4\*\*m\*S[-1]-S[-2])/(4\*\*m-1))  if m>=3:  R.append((4\*\*m\*C[-1]-C[-2])/(4\*\*m-1))  if m>4:  ep=abs(10\*(R[-1]-R[-2]))  Romberg(a,b,eps,func)  print("积分结果为：{:.5f}".format(R[-1]))  **截图：** |

|  |
| --- |
| 1. 实验小结与心得体会   在进行romberg积分计算的代码中有一些重复性相对较高的代码尚未改写成函数的方式，同时if语句的判断略显冗杂无无序，体现不出代码的美感，迫于时间无奈暂未对其进行修改，之后应该会把改代码改正的更为完善，通过代码的方式实现了数值积分的球阀，深刻的感觉到对于该类知识的基础进一步牢固了。 |

课程名称：计算机数值分析

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | 常微分方程初值问题的数值解法 | | | 实验成绩 |  |
| 实验者 | 张淳慷 | 专业班级 |  | 实验日期 | xxxx年xx月xx日 |
| 1. 实验概述    1. 实验目的   通过设计、编制、调试1~2个求常微分方程初值问题的数值解解的程序，加深对其数值计算方法及有关的基础理论知识的理解。   * 1. 实验要求   用编程语言实现用改进的欧拉（Euler）公式求解常微分方程初值问题、用四阶龙格-库塔(Runge-Kutta)方法求解常微分方程初值问题的程序。   * 1. 实验内容   用编程语言编程实现以下算法：   1. 用改进的欧拉（Euler）公式求解常微分方程初值问题。 2. 用四阶龙格-库塔(Runge-Kutta)方法求解常微分方程初值问题。    1. 实验器材 3. PC机。 4. 编程语言：xxx | | | | | |

|  |
| --- |
| 1. 实验设计、测试与分析    1. 问题：用改进的欧拉（Euler）公式求解常微分方程初值问题。       1. 算法描述   通过for循环来重现迭代过程，套用对应的改进欧拉公式得解   * + 1. 程序变量说明   A表示积分下限，b表示积分上限，h表示步长，y\_next表示yn，采用dy/dx = y-2x/y y0=1为例子   * + 1. 源程序代码及运行结果截图   源代码：  #采用dy/dx = y-2x/y y0=1为例子计算y在x为1时的值  #精度取10\*\*5  a=x=0 # 积分下限  b=1 # 积分上限  h = 0.2 #步长  y = y\_next=1  def f(x,y):  m = (y-2\*x)/y  return m  for i in range(int((b-a)/h)):  c = x+h  y\_p = y\_next + h\*(y\_next - 2\*x/y\_next)  y\_c = y\_next + h\*(y\_p - 2\*c/y\_p)  y\_next = (y\_p+y\_c)/2  x = x+h  print('所得答案为{:.5f}'.format(y\_next))  截图：     * 1. 问题：用四阶龙格-库塔(Runge-Kutta)方法求解常微分方程初值问题。      1. 算法描述   通过for循环来重现迭代过程，套用对应的四阶r-k方法得解   * + 1. 程序变量说明   A为积分下限，b为积分上限，h为步长，xy分别表示xn yn，采用dy/dx = y-2x/y y0=1为例子   * + 1. 源程序代码及运行结果截图   **源代码**：  #采用dy/dx = y-2x/y y0=1为例子计算y在x为1时的值  a=x=0 # 积分下限  b=1 # 积分上限  h = 0.2 #步长  y =1  def f(x,y):  m = y-2\*x/y  return m  for i in range(int((b-a)/h)):  c = x+h  k1 = f(x,y)  k2 = f((x+h/2),(y+(h/2)\*k1))  k3 = f((x+h/2),(y+(h/2)\*k2))  k4 = f(c,(y+h\*k3))  y = y + (h/6)\*(k1 + 2\*k2 + 2\*k3 +k4)  x = x+h  print('答案为{:.5f}'.format(y))  **截图：** |

|  |
| --- |
| 1. 实验小结与心得体会   本来准备是写个完整的含有用户输入输出的方法，但是在实际操作中发现，如果要进行这方面的编码的话，那么会涉及到十分复杂的函数的输入的识别，尤其是当输入的函数并不是正常的多次式函数而是诸如指数函数一类的输入，虽然通过提前给予选项的方法（即让用户选择是输入一次二次或是指数方程等）可以在一定程度上避免该类问题，但是我认为这项实验的初衷是考察关键处的代码翻译，，及欧拉公式与romberg公式的迭代部分，所以采用了写demo的方法来实现，并未采用用户输入函数的方式，这在某种意义上有一定的局限性，之后会就用户自有输入函数得对应解的功能进行进一步完善。但是通过对于处置问题解法的算法实现让我进一步加深了对于这写公式的理解与运用。 |

课程名称：计算机数值分析

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | 方程求根的数值方法 | | | 实验成绩 |  |
| 实验者 | 张淳慷 | 专业班级 |  | 实验日期 | xxxx年xx月xx日 |
| 1. 实验概述    1. 实验目的   通过设计、编制、调试2~3个用数值方法求方程根的程序，加深对方程求根的数值计算方法及有关的基础理论知识的理解。   * 1. 实验要求   用编程语言实现二分法、Newton迭代法、弦截法求方程根的程序。   * 1. 实验内容   用编程语言编程实现以下算法：   1. 用二分法求的根。 2. 用牛顿(Newton)迭代法求在附近的根。 3. 用弦截法求的根。    1. 实验器材 4. PC机。 5. 编程语言：python | | | | | |

|  |
| --- |
| 1. 实验设计、测试与分析    1. 问题：用二分法求的根。       1. 算法描述   采用while循环通过判断对应点的函数值与0的大小比较来进行相应的赋值   * + 1. 程序变量说明   a，b分别表示积分的上限和下限，c表示反复迭代的x，#选择y = x-1/(e\*\*x)为例子   * + 1. 源程序代码及运行结果截图   源代码：  #选择y = x-1/(e\*\*x)为例子  import math  a = 1  b = 0  c = 1  def f(x):  y = x-1/(math.e\*\*x)  return y  while True :  c = (a+b)/2  if f(c)>0:  a = c  else:  b = c  if abs(f(c))<10\*\*-5:  print('答案为',end = ':')  print(c)  break  截图：     * 1. 问题：用牛顿(Newton)迭代法求在附近的根。      1. 算法描述   采用while循环以模仿牛顿法中的迭代过程，同时为防止导数为0的情况出现，设置if语句用于判断。   * + 1. 程序变量说明   a表示其在零点附近的值， #选择y = x-1/(e\*\*x)为例子   * + 1. 源程序代码及运行结果截图   源代码：  import math  a = 0.5  def f(x):  y = x-1/(math.e\*\*x)  return y  def g(x):  y = 1+1/(math.e\*\*x)  return y  while True:  if g(a)==0:  print("出现奇异点{}".foramt(a))  break  else:  a = a-(f(a)/g(a))  if abs(f(a))<10\*\*-6:  print('答案为：{:.6f}'.format(a))  break  截图：     * 1. 问题：用弦截法求的根。      1. 算法描述   采用while函数模拟迭代的过程，后用简单代码附属弦截法的公式即可   * + 1. 程序变量说明   a为零点附近的点，c为题目所给的的已知函数值的点，选择y = x-1/(e\*\*x)为例子   * + 1. 源程序代码及运行结果截图   import math  a=0.6  c = 0.5  def f(x):  y = x-1/(math.e\*\*x)  return y  while True:  a = a-(f(a)/(f(a)-f(c)))\*(a-c)  if abs(f(a))<10\*\*-6:  print('答案为：{:.6f}'.format(a))  break |

|  |
| --- |
| 截图：     1. 实验小结与心得体会   通过算法具体实现了方程求根的数值方法，感觉对于该类知识掌握的更加牢固了。 |