



计算机与软件工程学院

上机报告

**（ 2021/2022 学年 第 1 学期 ）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | **数值计算（C++）** | | | | | |
| 课程代码 | **190901319** | | | | | |
| 上机时间 | 2021 | 年 | 12 | 月 | 22 | 日 |
| 指导单位 | 物联网工程系 | | | | | |
| 任课教师 | 李显勇 | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 学生姓名 | 李子涵 | | |
| 学 号 | 3120190971401 | | |
| 成 绩 |  | 年级专业 | 计算机科学与技术 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称** | **【实验8】 幂法，反幂法** | | | **实验地点** |  |
| **实验类型** | **验证** | **实验学时** | **2** | **实验日期** | 2021.12.22 |
| **实验目的和要求**   1. 了解**求解矩阵的按模最大(最小)特征值及其特征向量的**基本理论和算法（即**幂法/反幂法**）； 2. 提交以下文档（用**学号+姓名**作为文件夹名，将所有内容放于该文件夹中）：   1）实验报告  2）程序代码 | | | | | |
| **实验环境(实验设备)**  windows XP及以上版本；PC；Matlab7.0及以上版本；其它高级语言 | | | | | |
| **实验原理及内容**  **一、实验原理**  根据**幂法/反幂法的**相关知识和算法编程完成本实验  **二、实验内容**  已知矩阵         1. 用幂法的规范运算计算矩阵A的按模最大的特征值及其特征向量； 2. 用反幂法的规范运算计算矩阵A的按模最小的特征值及其特征向量。   **三、实验过程（可以文字说明+运行结果截图）**  '''  文件名：西华大学数值计算C++实验报告——实验八  内容： 幂法，反幂法  当前版本：1.0  完成作者：李子涵  学号：3120190971401  完成日期：2021.12.22  '''  import numpy as np  import numpy as np  def eig\_power(A,v0,eps):      """      幂法：矩阵，初始,误差      """      xk=x0      n=1      yk=xk      xk\_last=xk      while(1):          xk=A@yk          yk=xk/max(abs(xk))          print("x"+str(n)+"转置:")          print(xk.T)          print("y"+str(n)+"转置:")          print(yk.T)          if(n>100 or (xk==xk\_last).all()):#全部相等              print("最大特征值：")              print(max(xk))              print("对应特征向量转置：")              print(yk.T)              break          xk\_last=xk          n+=1  def eig\_invpower(A,v0,eps,p=0):      """      反幂法      """      uk = v0      flag = 1      val\_old = 0      n = 0      if p!=0:          A = A-p\*np.eye(len(A))      LU,La,Ua,order0,order1 = Doolittle\_pivot(np.asarray(A))   # PA=LU      while flag:          n = n+1          vk = solveLineq(La,Ua,np.asarray(uk)[order1,:])          val = vk[np.argmax(np.abs(vk))]          uk = np.asmatrix(vk.reshape(len(A),1))/val          if (np.abs(1/val-val\_old)<eps):              flag = 0          val\_old = 1/val      print('特征向量:',np.asarray(uk).flatten())      print('迭代次数:',n)      return 1/val+p, uk  def Doolittle\_pivot(A):  # A为np.array，而不是np.matrix      n = len(A)      LU = A.copy()      order1 = np.arange(n)      for r in range(n):          ut = LU[:r,r].reshape(r,1)          si = A[r:,r] - np.sum(ut\*LU[r:,:r].T,axis=0)          ir = np.argmax(np.abs(si))          if ir!=0:              LU[[r,r+ir],:] = LU[[r+ir,r],:]              order1[[r,r+ir]] = order1[[r+ir,r]]          lt = LU[r,:r].reshape(r,1)          LU[r,r:] = LU[r,r:] - np.sum(lt\*LU[:r,r:],axis=0)          if r==n-1:              continue          LU[r+1:,r] = (LU[r+1:,r] - np.sum(ut\*LU[r+1:,:r].T,axis=0))/LU[r,r]      U = np.triu(LU)      L = np.tril(LU) - np.diag(np.diag(LU)) + np.eye(n)      order0 = []      [order0.insert(i,np.where(order1==i)[0][0]) for i in range(n)]      return LU,L,U,order0,order1  def solveLineq(L,U,b):  # b为np.array，而不是np.matrix      rows = len(b)      y = np.zeros(rows)      y[0] = b[0]/L[0,0]      for k in range(1,rows):          y[k] = (b[k] - np.sum(L[k,:k]\*y[:k]))/L[k,k]      x = np.zeros(rows)      k = rows-1      x[k] = y[k]/U[k,k]      for k in range(rows-2,-1,-1):          x[k] = (y[k] - np.sum(x[k+1:]\*U[k,k+1:]))/U[k,k]      return x    if \_\_name\_\_ == '\_\_main\_\_':      A = np.matrix([[4,   -1,      1],                     [16,   -2,      -2],                     [16,   -3,   -1]], dtype='float')      eps = 1e-10      eps0=1e-5      v0 = np.matrix([[1],[1],[1]], dtype='float')      print("幂法")      val,uk = eig\_power(A,v0,eps0)      print("反幂法")      val,uk = eig\_invpower(A,v0,eps,1.2679)    实验结果：  幂法：    反幂法： | | | | | |

|  |
| --- |
| **四、实验小结**（包括问题和解决方法、心得体会、意见与建议等）    求解幂法时，程序陷入死循环，后经过多方面的排查，最终修改代码得出正确结果。 |