山东大学 计算机 学院

数值计算 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201300140122 | 姓名： 于仲源 | | 班级： 2013级2班 |
| 实验题目： 线性方程组的直接解法 | | | |
| 实验学时：4 | | 实验日期： 2017/3/22 | |
| 实验目的：  用LU 分解、前代、回代函数、部分列主元求解线性方程组，用Cholesky 分解求解对称正定方程组，分析残差与误差的关系。 | | | |
| 硬件环境：  Thinkpad 笔记本 | | | |
| 软件环境：  VS2013 | | | |
| 实验步骤与内容：   1. 实现部分列主元的矩阵LU 分解函数，返回三个矩阵P，L，U   int LUfenjie(){    int n; double z;double \*\*a; int i, j;  printf("请输入矩阵A的大小:\n");  scanf("%d", &n);  a = (double\*\*)malloc(sizeof(double\*)\*n);//为二维数组分配n行  for (i = 0; i<n; i++)a[i] = (double\*)malloc(sizeof(double)\*n);  double \*\*a\_temp; a\_temp = (double\*\*)malloc(sizeof(double\*)\*n);// 为二维数组分配n行  for (i = 0; i<n; i++)a\_temp[i] = (double\*)malloc(sizeof(double)\*n);    //LU分解需要的一些变量  int k; double \*\*m; double max;  //为m矩阵分配内存并初始化成零  m = (double\*\*)malloc(sizeof(double\*)\*n);  for (i = 0; i<n; i++){  m[i] = (double\*)malloc(sizeof(double)\*n);  }  for (i = 0; i<n; i++){  for (j = 0; j<n; j++)  m[i][j] = 0;  }  printf("请逐行输入矩阵A\n");  for (i = 0; i<n; i++)  for (j = 0; j<n; j++){  scanf("%lf ", &z);  a\_temp[i][j] = a[i][j] = z;  }  printf("输出矩阵A\n");  for (i = 0; i<n; i++){  for (j = 0; j<n; j++)  printf(" %lf ", a[i][j]);  printf("\n");  }  double \*b;  double s;  b = (double\*)malloc(sizeof(double)\*n);  printf("请输入向量b\n");  scanf("%lf", &s);  for (i = 0; i<n; i++){  scanf("%lf", &s);  b[i] = s;  }  printf("输出向量b\n");  for (i = 0; i<n; i++)  printf("%lf ", b[i]);  printf("\n");  //开始LU分解  for (k = 0; k<n - 1; k++){  if (a[k][k] == 0){  printf("主元为零！\n");  return 0;  }  //部分主元法  for (i = k + 1; i<n; i++)  m[i][k] = a[i][k] / a[k][k];  for (j = k + 1; j<n; j++)  for (i = k + 1; i<n; i++)  a[i][j] = a[i][j] - m[i][k] \* a[k][j];  }  printf("输出矩阵m\n");  //输出测试  for (i = 0; i<n; i++){  for (j = 0; j<n; j++)  printf("%lf ", m[i][j]);  printf("\n");  }  printf("输出矩阵L\n");  //输出测试  for (i = 0; i<n; i++){  for (j = 0; j<n; j++)  printf("%lf ", m[i][j]);  printf("\n");  }  printf("输出矩阵U\n");  //输出测试  for (i = 0; i<n; i++){  for (j = 0; j<n; j++){  if (i>j)  a[i][j] = 0;  printf("%lf ", a[i][j]);  }  printf("\n");  }  for (i = 0; i<n; i++){  for (j = 0; j<n; j++)  if (i == j)  m[i][j] = 1;  }  }  实验截图：     1. 实现求解上三角方程组的函数   //求解Ux=y;  //上三角回代法  double \*x;  x = (double\*)malloc(sizeof(double)\*n);  for (i = 0; i<n; i++)  x[i] = 0;  for (j = n - 1; j >= 0; j--){  if (a[j][j] == 0){  printf("矩阵奇异\n");  return 0;  }  x[j] = y[j] / a[j][j];  for (i = 0; i<j; i++)  y[i] = y[i] - a[i][j] \* x[j];  }  printf(" \n");  printf("输出x值\n");  for (i = 0; i<n; i++)  printf("%lf ", x[i]);  printf("\n ");   1. 实现求解下三角方程组的函数   //求解Ly=b;  //下三角的前代法  double \*y;  y = (double\*)malloc(sizeof(double)\*n);  for (i = 0; i<n; i++)  y[i] = 0;  for (j = 0; j<n; j++){  if (m[j][j] == 0){  printf("矩阵是奇异的\n");  return 0;  }  y[j] = b[j] / m[j][j];  for (i = j + 1; i<n; i++){  b[i] = b[i] - m[i][j] \* y[j];  }  }  printf("输出y值\n");  for (i = 0; i<n; i++)  printf("%lf ", y[i]);  printf("\n");   1. 用上面的函数求解方程组Ax=b, 并计算残差r=b-Ax     输入矩阵：    输出LU分解矩阵：    输出计算结果：    可见残差为0，结果精确度很高。   1. 实现对称正定矩阵的Cholesky 分解函数   //cholesky分解  int k;  for(k=0;k<n;k++){  a[k][k]=sqrt(a[k][k]);  for(i=k+1;i<n;i++)  a[i][k]=a[i][k]/a[k][k];//调整当前列  for(j=k+1;j<n;j++)  for(i=k+1;i<n;i++)  a[i][j]=a[i][j]-a[i][k]\*a[j][k];  }  (6) 用Cholesky 分解求解下列方程组    希尔伯特矩阵的生成：  for (i = 0; i < n; i++){  for (j = 0; j<n; j++){  a\_temp[i][j] = a[i][j] = 1.0 / (i + j - 1 + 2);  }  }  结果截图：           1. 用Cholesky 分解方法求解线性方程组Hx=b,计算近似解ˆx 。  |  |  | | --- | --- | | n | bT | | 5 | [2.2833,1.45,1.0929,0.8845,0.7456] | | 6 | [2.45,1.5929,1.2179,0.9956,0.8456,0.7365] | | 7 | [2.5929 1.7179 1.3290 1.0956 0.9365 0.8199 0.7301] | | 8 | [2.7179 1.8290 1.4290 1.1865 1.0199 0.8968 0.8016 0.7254] | | 9 | [2.8290 1.9290 1.5199 1.2699 1.0968 0.9682 0.8682 0.7879 0.7217] | | 10 | [2.9290 2.0199 1.6032 1.3468 1.1682 1.0349 0.9307 0.8467 0.7773 0.7188] |  |  |  | | --- | --- | | n | xT | | 5 | [1.0,1.0,1.0,1.0,1.0] | | 6 | [1.0,1.0,1.0,1.0,1.0,1.0] | | 7 | [1.0,1.0,1.0,1.0,1.0,1.0,1.0] | | 8 | [1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0] | | 9 | [1.000000,1.000000,1.000000,1.000002,0.999992,1.000016,0.999982,1.000011,0.999997] | | 10 | [1.000000,1.000000,0.999998,1.000021,0.999901,1.000272,0.999555,1.000429,0.999775,1.000049] |   部分结果截图：       1. 比较残差 r=b-Hx、误差的无穷范数。   当n很小时，误差为零，随着n的增大，x产生误差，并随之增大。并且残差在n较小时几乎为零，当n=10时，产生残差r=1e-15,并且随着n的增大，残差也增大。  (c) 随着n 的增加，误差和残差如何变化？为什么？  随着n的增加，误差无残差都随之增大，因为，Hibert矩阵随着n增大，1/n，楚列斯基分解中开方及除法产生的舍入误差会增大  附：楚列斯基分解希尔伯特矩阵的C代码：  int Hilbert(){  int n; double z;double \*\*a; int i, j;  double \*b;  printf("请输入矩阵A的大小:\n");scanf("%d", &n);  b = (double\*)malloc(sizeof(double)\*n);  a = (double\*\*)malloc(sizeof(double\*)\*n);//为二维数组分配n行  for (i = 0; i<n; i++)a[i] = (double\*)malloc(sizeof(double)\*n);  double \*\*a\_temp; a\_temp = (double\*\*)malloc(sizeof(double\*)\*n);//为二维数组分配n行  for (i = 0; i<n; i++)a\_temp[i] = (double\*)malloc(sizeof(double)\*n);  double \*\*c;c = (double\*\*)malloc(sizeof(double\*)\*n);  for (i = 0; i<n; i++)c[i] = (double\*)malloc(sizeof(double)\*n);  /\*  printf("请逐行输入矩阵A\n");  for (i = 0; i<n; i++)  for (j = 0; j<n; j++){  scanf("%lf ", &z);  a\_temp[i][j] = a[i][j] = z;  }  printf("输出矩阵A\n");  for (i = 0; i<n; i++){  for (j = 0; j<n; j++)  printf(" %lf ", a[i][j]);  printf("\n");  }  \*/  //希尔伯特矩阵的生成  for (i = 0; i < n; i++){  for (j = 0; j<n; j++){  a\_temp[i][j] = a[i][j] = 1.0 / (i + j - 1 + 2);  }  }    //显示H矩阵  for (i = 0; i < n; i++){  for (j = 0; j<n; j++){  printf(" %lf ",a[i][j]);  }printf("\n");  }    //计算b数组  for (i = 0; i < n; i++)  b[i] = 0;  for (i = 0; i < n; i++){  for (j = 0; j<n; j++){  b[i]+=a[i][j];  }  }  printf("向量b：\n");  for (i = 0; i < n; i++)  printf(" %lf ",b[i]);  //cholesky分解  int k;  for (k = 0; k<n; k++){  a[k][k] = sqrt(a[k][k]);  for (i = k + 1; i<n; i++)  a[i][k] = a[i][k] / a[k][k];//调整当前列  for (j = k + 1; j<n; j++)  for (i = k + 1; i<n; i++)  a[i][j] = a[i][j] - a[i][k] \* a[j][k];  }  /\*  //改进的平方根算法  for (k = 0; k<n; k++){  for (i = 0; i<k; i++)  a[k][k] -= a[i][i] \* a[k][i] \* a[k][i];  for (j = k + 1; j<n; j++)  {  for (i = 0; i<k; i++)  a[j][k] -= a[j][i] \* a[i][i] \* a[k][i];  a[j][k] /= a[k][k];  }  }  \*/  for (i = 0; i<n; i++)  for (j = 0; j<n; j++)  c[i][j] = a[i][j];  for (i = 0; i<n - 1; i++)  for (j = i + 1; j<n; j++)  c[i][j] = 0;  printf("\n分解以后矩阵：\n");  for (i = 0; i < n; i++){  for (j = 0; j<n; j++){  printf(" %lf ", c[i][j]);  }printf("\n");  }  printf("方程的解为\n");  //求解Ly=b;  //下三角的前代法  double \*y;  y = (double\*)malloc(sizeof(double)\*n);  for (i = 0; i<n; i++)  y[i] = b[i];  for (int i = 0; i < n; i++)  {  for (int j = 0; j<i; j++)  {  y[i] = y[i] - y[j] \* c[i][j];  }y[i] = y[i] / c[i][i];  }  printf("\n y的值为：\n");  for (int i = 0; i < n; i++)  printf(" %lf ",y[i]);  //求解最终解x  double \*x;  x = (double\*)malloc(sizeof(double)\*n);  for (int i = 0; i < n; i++)  x[i] = y[i];  for (int i = n-1; i >= 0; i--)  {  for (int j = n-1; j>i; j--)  {  x[i] = x[i] - x[j] \* c[j][i];  }x[i] = x[i] / c[i][i];  }  printf("\n x的值为：\n");  for (int i = 0; i < n; i++)  printf(" %lf ", x[i]);    //计算残差  printf("\n 输出残差\n");  double \*r;  r = (double\*)malloc(sizeof(double)\*n);  for (i = 0; i<n; i++)  r[i] = 0;  for (i = 0; i<n; i++){//i是行，遍历所有行  for (j = 0; j < n; j++){//一行中从左到右  r[i] += a\_temp[i][j] \* x[j];  }  }  double \*can;  can = (double\*)malloc(sizeof(double)\*n);  for (i = 0; i<n; i++)  can[i] = b[i] - r[i];  for (i = 0; i<n; i++)  printf("%lf ", can[i]);  return 0;  } | | | |
| 结论分析与体会：  本次实验加深了对LU分解以及楚列斯基分解的认识。确实值得花费了许多时间研究。同时，通过网络上各种资源，也研究了相应的算法的改进。 | | | |