Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра прикладной математики

Лабораторная работа №3

по дисциплине «Численные методы»

Группа ПМ-63

Студент Майер В. А.

Преподаватель Задорожный А.Г.

Вариант 10

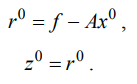
Новосибирск, 2018

1. Цель работы  
   Изучить особенности реализации трехшаговых итерационных методов для СЛАУ с разреженными матрицами. Исследовать влияние предобусловливания на сходимость изучаемых методов на нескольких матрицах большой (не менее 10000) размерности.

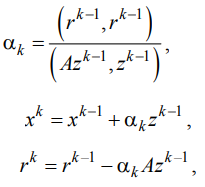
Задание: Сравнить МСГ и ЛОС для несимметричной матрицы. Факторизация LU .

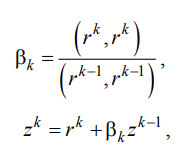
1. Анализ

***Формулы для метода сопряженных градиентов:***Выбирается начальное приближение x0 и полагается



Далее для  производятся следующие вычисления:

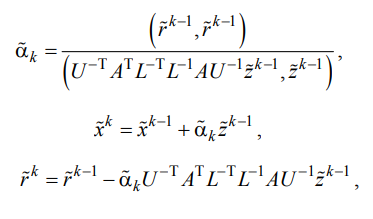


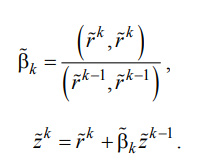


*Для LU Факторизации*  
Выбирается начальное приближение x0 и полагается

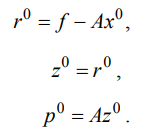
  


Далее для  производятся следующие вычисления:

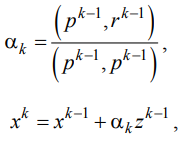


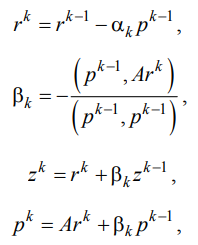


***Формула для локально-оптимальной схемы:***Выбирается начальное приближение x0 и полагается

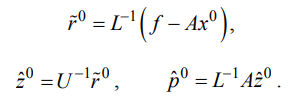


Далее для  производятся следующие вычисления:

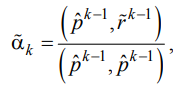


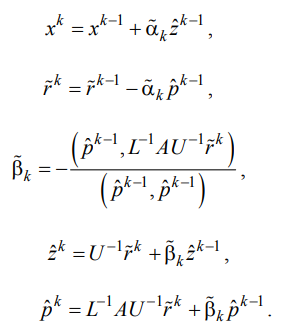


*Для LU Факторизации*Выбирается начальное приближение x0 и полагается



Далее для  производятся следующие вычисления:





1. Матрица с диагональным преобладанием из ЛР2(лабораторной работы 2)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ax = F | 8 | -1 |  |  |  |  | -3 | -1 | -2 |  |  |  |  | 1 |  | -41 |
| -2 | 11 | -3 |  |  |  |  | -2 | -1 | -3 |  |  |  | 2 |  | -44 |
|  | -1 | 9 | -2 |  |  |  |  | -2 | -1 | -3 |  |  | 3 |  | -44 |
|  |  | -3 | 10 | -1 |  |  |  |  | -1 | -3 | -2 |  | 4 |  | -41 |
|  |  |  | -2 | 8 | -1 |  |  |  |  | -2 | -3 |  | 5 |  | -32 |
|  |  |  |  | -2 | 6 | -2 |  |  |  |  | -2 |  | 6 |  | -12 |
| -3 |  |  |  |  | -2 | 8 | -3 |  |  |  |  | \* | 7 | = | 17 |
| -2 | -3 |  |  |  |  | -1 | 8 | -2 |  |  |  |  | 8 |  | 31 |
| -1 | -3 | -2 |  |  |  |  | -3 | 10 | -1 |  |  |  | 9 |  | 43 |
|  | -2 | -1 | -3 |  |  |  |  | -2 | 9 | -1 |  |  | 10 |  | 42 |
|  |  | -2 | -1 | -3 |  |  |  |  | -2 | 11 | -3 |  | 11 |  | 40 |
|  |  |  | -3 | -1 | -2 |  |  |  |  | -1 | 7 |  | 12 |  | 44 |

e = 1e-14 maxiter = 100000

|  |  |
| --- | --- |
| Сравнение числа операций | |
| Блочная релаксация B = 2 | Блочная релаксация B = 4 |
| Coud A = 1.04 w = 1.73 k = 177 0.999999999999925 | 7.460698725481052e-14 1.999999999999986 | 1.443289932012704e-14 2.999999999999920 | 7.993605777301127e-14 3.999999999999936 | 6.439293542825908e-14 4.999999999999914 | 8.615330671091215e-14 5.999999999999978 | 2.220446049250313e-14 6.999999999999917 | 8.348877145181177e-14 7.999999999999919 | 8.082423619271140e-14 8.999999999999995 | 5.329070518200751e-15 9.999999999999975 | 2.486899575160351e-14 10.999999999999918 | 8.171241461241152e-14 11.999999999999977 | 2.309263891220326e-14 | Coud A = 0.487 w = 1.72 k = 135 1.000000000000020 | -1.976196983832779e-14 2.000000000000021 | -2.131628207280301e-14 2.999999999999957 | 4.307665335545607e-14 4.000000000000021 | -2.131628207280301e-14 4.999999999999975 | 2.486899575160351e-14 5.999999999999985 | 1.509903313490213e-14 6.999999999999946 | 5.417888360170764e-14 7.999999999999976 | 2.398081733190338e-14 9.000000000000055 | -5.506706202140776e-14 10.000000000000011 | -1.065814103640150e-14 10.999999999999947 | 5.329070518200751e-14 12.000000000000034 | -3.375077994860476e-14 |
| ЛОС | МСГ |
| K = 150  9.999999999999929e-01|7.105427357601002e-15  1.999999999999994e+00|6.439293542825908e-15  2.999999999999989e+00|1.065814103640150e-14  3.999999999999991e+00|8.881784197001252e-15  4.999999999999993e+00|7.105427357601002e-15  5.999999999999994e+00|6.217248937900877e-15  6.999999999999996e+00|3.552713678800501e-15  7.999999999999980e+00|2.042810365310288e-14  8.999999999999995e+00|5.329070518200751e-15  9.999999999999984e+00|1.598721155460225e-14  1.099999999999999e+01|8.881784197001252e-15  1.200000000000001e+01|-5.329070518200751e-15 | K = 22  1.000000000000707e+00|-7.065459328714496e-13  2.000000000000824e+00|-8.237854842718662e-13  3.000000000000873e+00|-8.726352973553730e-13  4.000000000000889e+00|-8.890665981198254e-13  5.000000000000894e+00|-8.943956686380261e-13  6.000000000000862e+00|-8.624212455288216e-13  7.000000000000780e+00|-7.798206524967100e-13  8.000000000000792e+00|-7.922551503725117e-13  9.000000000000821e+00|-8.206768598029157e-13  1.000000000000086e+01|-8.633094239485217e-13  1.100000000000089e+01|-8.935074902183260e-13  1.200000000000089e+01|-8.935074902183260e-13 |
| ЛОС Диагональный | МСГ Диагональный |
| K = 158  9.999999999999258e-01|7.416289804496046e-14  1.999999999999917e+00|8.348877145181177e-14  2.999999999999915e+00|8.482103908136196e-14  3.999999999999915e+00|8.482103908136196e-14  4.999999999999917e+00|8.260059303211165e-14  5.999999999999916e+00|8.437694987151190e-14  6.999999999999918e+00|8.171241461241152e-14  7.999999999999916e+00|8.437694987151190e-14  8.999999999999924e+00|7.638334409421077e-14  9.999999999999913e+00|8.704148513061227e-14  1.099999999999992e+01|7.993605777301127e-14  1.199999999999992e+01|7.993605777301127e-14 | K = 19  9.999999999971332e-01|2.866817894187079e-12  1.999999999996641e+00|3.358868738700949e-12  2.999999999996451e+00|3.548716875911850e-12  3.999999999996377e+00|3.623323863166661e-12  4.999999999996369e+00|3.630873379734112e-12  5.999999999996501e+00|3.499422973618493e-12  6.999999999996827e+00|3.173461493588547e-12  7.999999999996773e+00|3.226752198770555e-12  8.999999999996657e+00|3.343103571751271e-12  9.999999999996485e+00|3.515410185173096e-12  1.099999999999638e+01|3.623767952376511e-12  1.199999999999637e+01|3.625544309215911e-12 |
| ЛОС LU | МСГ LU |
| K = 1000000(max)  9.872085035160885e-01|1.279149648391154e-02  1.985055433567088e+00|1.494456643291153e-02  2.984286187688817e+00|1.571381231118263e-02  3.983851960399779e+00|1.614803960022071e-02  4.983724912886403e+00|1.627508711359749e-02  5.984172380007292e+00|1.582761999270765e-02  6.985495049189658e+00|1.450495081034209e-02  7.985330089313238e+00|1.466991068676204e-02  8.984903385627236e+00|1.509661437276399e-02  9.984194326116368e+00|1.580567388363185e-02  1.098376599092611e+01|1.623400907389083e-02  1.198346724463531e+01|1.653275536468612e-02 | K = 13  1.000000000000354e+00|-3.541611448554249e-13  2.000000000000414e+00|-4.143352327901084e-13  3.000000000000440e+00|-4.400924069614121e-13  4.000000000000452e+00|-4.520828156273637e-13  5.000000000000458e+00|-4.583000645652646e-13  6.000000000000444e+00|-4.440892098500626e-13  7.000000000000396e+00|-3.961275751862559e-13  8.000000000000398e+00|-3.979039320256561e-13  9.000000000000414e+00|-4.138911435802584e-13  1.000000000000044e+01|-4.405364961712621e-13  1.100000000000046e+01|-4.583000645652646e-13  1.200000000000047e+01|-4.689582056016661e-13 |
|  |  |

1. Матрица с обратными элементами на недиагональных элементах из ЛР 2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ax = F | 8 | 1 |  |  |  |  | 3 | 1 | 2 |  |  |  |  | 1 |  | 57 |
| 2 | 11 | 3 |  |  |  |  | 2 | 1 | 3 |  |  |  | 2 |  | 88 |
|  | 1 | 9 | 2 |  |  |  |  | 2 | 1 | 3 |  |  | 3 |  | 98 |
|  |  | 3 | 10 | 1 |  |  |  |  | 1 | 3 | 2 |  | 4 |  | 121 |
|  |  |  | 2 | 8 | 1 |  |  |  |  | 2 | 3 |  | 5 |  | 112 |
|  |  |  |  | 2 | 6 | 2 |  |  |  |  | 2 |  | 6 |  | 84 |
| 3 |  |  |  |  | 2 | 8 | 3 |  |  |  |  | \* | 7 | = | 95 |
| 2 | 3 |  |  |  |  | 1 | 8 | 2 |  |  |  |  | 8 |  | 97 |
| 1 | 3 | 2 |  |  |  |  | 3 | 10 | 1 |  |  |  | 9 |  | 137 |
|  | 2 | 1 | 3 |  |  |  |  | 2 | 9 | 1 |  |  | 10 |  | 138 |
|  |  | 2 | 1 | 3 |  |  |  |  | 2 | 11 | 3 |  | 11 |  | 202 |
|  |  |  | 3 | 1 | 2 |  |  |  |  | 1 | 7 |  | 12 |  | 124 |

|  |  |
| --- | --- |
| Сравнение числа операций | |
| Блочная релаксация B = 2 | Блочная релаксация B = 6 |
| Coud A = 2.47 w = 1.05 k = 21 1.000000000000034 | -3.352873534367973e-14 1.999999999999970 | 3.042011087472929e-14 3.000000000000122 | -1.216804434989172e-13 3.999999999999966 | 3.419486915845482e-14 4.999999999999983 | 1.687538997430238e-14 5.999999999999916 | 8.437694987151190e-14 6.999999999999969 | 3.108624468950438e-14 8.000000000000094 | -9.414691248821327e-14 8.999999999999964 | 3.552713678800501e-14 10.000000000000036 | -3.552713678800501e-14 10.999999999999970 | 3.019806626980426e-14 12.000000000000041 | -4.085620730620576e-14 | Coud A = 1.88 w = 1.1 k = 15 0.999999999999878 | 1.223465773136923e-13 2.000000000000055 | -5.462297281155770e-14 3.000000000000005 | -5.329070518200751e-15 3.999999999999932 | 6.794564910705958e-14 4.999999999999934 | 6.572520305780927e-14 6.000000000000050 | -4.973799150320701e-14 7.000000000000017 | -1.687538997430238e-14 7.999999999999954 | 4.618527782440651e-14 8.999999999999988 | 1.243449787580175e-14 10.000000000000030 | -3.019806626980426e-14 11.000000000000002 | -1.776356839400250e-15 11.999999999999993 | 7.105427357601002e-15 |
| ЛОС | МСГ |
| K = 39  9.999999999998724e-01|1.275646255294305e-13  1.999999999999961e+00|3.907985046680551e-14  3.000000000000060e+00|-5.995204332975845e-14  3.999999999999885e+00|1.150191053511662e-13  5.000000000000083e+00|-8.260059303211165e-14  5.999999999999607e+00|3.925748615074554e-13  7.000000000000147e+00|-1.465494392505207e-13  8.000000000000071e+00|-7.105427357601002e-14  8.999999999999927e+00|7.283063041541027e-14  1.000000000000017e+01|-1.687538997430238e-13  1.099999999999975e+01|2.504663143554353e-13  1.200000000000036e+01|-3.623767952376511e-13 | K = 15  1.000000000000002e+00|-1.776356839400250e-15  1.999999999999999e+00|6.661338147750939e-16  3.000000000000004e+00|-3.552713678800501e-15  3.999999999999996e+00|3.996802888650564e-15  5.000000000000001e+00|-8.881784197001252e-16  5.999999999999997e+00|2.664535259100376e-15  7.000000000000002e+00|-1.776356839400250e-15  7.999999999999999e+00|8.881784197001252e-16  8.999999999999995e+00|5.329070518200751e-15  1.000000000000000e+01|0.000000000000000e+00  1.099999999999999e+01|5.329070518200751e-15  1.200000000000001e+01|-7.105427357601002e-15 |
| ЛОС Диагональный | МСГ Диагональный |
| K = 35  9.999999999987738e-01|1.226241330698485e-12  2.000000000000770e+00|-7.700506898800086e-13  2.999999999999520e+00|4.796163466380676e-13  3.999999999999693e+00|3.068656440063933e-13  5.000000000000236e+00|-2.362554596402333e-13  5.999999999999244e+00|7.558398351648066e-13  7.000000000001097e+00|-1.096900348329655e-12  7.999999999999917e+00|8.260059303211165e-14  9.000000000000449e+00|-4.494182803682634e-13  9.999999999999959e+00|4.085620730620576e-14  1.100000000000029e+01|-2.877698079828406e-13  1.200000000000053e+01|-5.258016244624741e-13 | K = 15  1.000000000000004e+00|-3.996802888650564e-15  2.000000000000006e+00|-5.773159728050814e-15  2.999999999999989e+00|1.110223024625157e-14  4.000000000000007e+00|-7.105427357601002e-15  4.999999999999994e+00|6.217248937900877e-15  6.000000000000014e+00|-1.421085471520200e-14  6.999999999999996e+00|4.440892098500626e-15  8.000000000000002e+00|-1.776356839400250e-15  9.000000000000002e+00|-1.776356839400250e-15  9.999999999999991e+00|8.881784197001252e-15  1.100000000000002e+01|-1.598721155460225e-14  1.199999999999999e+01|1.421085471520200e-14 |
| ЛОС LU | МСГ LU |
| K = 14  9.999999999999697e-01|3.030908857226677e-14  1.999999999999954e+00|4.574118861455645e-14  3.000000000000031e+00|-3.108624468950438e-14  3.999999999999990e+00|9.769962616701378e-15  5.000000000000016e+00|-1.598721155460225e-14  5.999999999999941e+00|5.861977570020827e-14  7.000000000000084e+00|-8.437694987151190e-14  8.000000000000057e+00|-5.684341886080801e-14  8.999999999999966e+00|3.375077994860476e-14  1.000000000000009e+01|-8.881784197001252e-14  1.099999999999994e+01|6.394884621840902e-14  1.200000000000004e+01|-4.263256414560601e-14 | K = 12  9.999999999999978e-01|2.220446049250313e-15  2.000000000000003e+00|-3.108624468950438e-15  2.999999999999992e+00|7.549516567451064e-15  4.000000000000001e+00|-8.881784197001252e-16  5.000000000000004e+00|-3.552713678800501e-15  5.999999999999998e+00|1.776356839400250e-15  7.000000000000007e+00|-7.105427357601002e-15  7.999999999999992e+00|7.993605777301127e-15  9.000000000000002e+00|-1.776356839400250e-15  1.000000000000000e+01|0.000000000000000e+00  1.100000000000000e+01|3.552713678800501e-15  1.200000000000000e+01|0.000000000000000e+00 |

1. Матрицы Гильберта

|  |  |
| --- | --- |
| Матрица Гильберта размерности 3 | |
| ЛОС | МСГ |
| K = 4  1.000000000000011e+00|-1.110223024625157e-14  1.999999999999921e+00|7.949196856316121e-14  3.000000000000084e+00|-8.437694987151190e-14 | K = 5  9.999999999989818e-01|1.018185535883731e-12  2.000000000005712e+00|-5.711875417091505e-12  2.999999999994480e+00|5.519584789226428e-12 |
| ЛОС Диагональный | МСГ Диагональный |
| K = 4  1.000000000000012e+00|-1.243449787580175e-14  1.999999999999933e+00|6.705747068735946e-14  3.000000000000064e+00|-6.394884621840902e-14 | K = 4  9.999999999975439e-01|2.456146397378234e-12  2.000000000013249e+00|-1.324851339745692e-11  2.999999999987418e+00|1.258237958268182e-11 |
| ЛОС LU | МСГ LU |
| K = 1  1.000000000000001e+00|-6.661338147750939e-16  1.999999999999999e+00|1.332267629550188e-15  3.000000000000000e+00|0.000000000000000e+00 | K = 1  9.999999999996902e-01|3.097522238704187e-13  2.000000000001891e+00|-1.891375944751417e-12  2.999999999998092e+00|1.907807245515869e-12 |

|  |  |
| --- | --- |
| Матрица Гильберта размерности 6 | |
| ЛОС | МСГ |
| K = 10  9.999999999966608e-01|3.339217791165083e-12  2.000000000182678e+00|-1.826778728286627e-10  2.999999998593028e+00|1.406971872341956e-09  4.000000003727509e+00|-3.727508968154325e-09  4.999999995971637e+00|4.028363420616188e-09  6.000000001528723e+00|-1.528722926025239e-09 | K = 11  1.000037998065438e+00|-3.799806543813844e-05  1.998917185985580e+00|1.082814014420119e-03  3.007314137591349e+00|-7.314137591348580e-03  3.981001768160415e+00|1.899823183958516e-02  5.020945118689133e+00|-2.094511868913340e-02  5.991755383832934e+00|8.244616167066177e-03-12 |
| ЛОС Диагональный | МСГ Диагональный |
| K = 9  1.000000000001346e+00|-1.346256439660465e-12  1.999999999966782e+00|3.321787289678468e-11  3.000000000213733e+00|-2.137334753626874e-10  3.999999999449271e+00|5.507287959005680e-10  5.000000000611416e+00|-6.114158068726283e-10  5.999999999756144e+00|2.438564905560270e-10 | K = 10  1.000003954437686e+00|-3.954437686193302e-06  1.999887619067559e+00|1.123809324405922e-04  3.000757749235300e+00|-7.577492352996984e-04  3.998034271972605e+00|1.965728027395031e-03  5.002165104875860e+00|-2.165104875859747e-03  5.999148383896207e+00|8.516161037928072e-04 |
| ЛОС LU | МСГ LU |
| K = 2  1.000002721918199e+00|-2.721918199277695e-06  1.999918720447272e+00|8.127955272829723e-05  3.000567081090399e+00|-5.670810903994905e-04  3.998491516626089e+00|1.508483373911407e-03  5.001693789136024e+00|-1.693789136024471e-03  5.999323523065830e+00|6.764769341698340e-04 | K = 1  9.999999999981599e-01|1.840083641013734e-12  2.000000000050117e+00|-5.011679959920912e-11  2.999999999672339e+00|3.276605653468323e-10  4.000000000830875e+00|-8.308749244179126e-10  4.999999999100589e+00|8.994112121740727e-10  6.000000000348993e+00|-3.489928346311899e-10 |

|  |  |
| --- | --- |
| Матрица Гильберта размерности 9 | |
| ЛОС | МСГ |
| K = 17  1.000000084742701e+00|-8.474270063452138e-08  1.999995959255261e+00|4.040744738587065e-06  3.000045415994236e+00|-4.541599423557585e-05  3.999804030297414e+00|1.959697025859519e-04  5.000340079926725e+00|-3.400799267252452e-04  5.999942446789525e+00|5.755321047473672e-05  6.999452799349900e+00|5.472006501001658e-04  8.000645802940985e+00|-6.458029409852628e-04  8.999773305839916e+00|2.266941600836958e-04 | K = 15  9.999668074828993e-01|3.319251710065529e-05  2.001081918935363e+00|-1.081918935362580e-03  2.992001159155007e+00|7.998840844993360e-03  4.020237687816004e+00|-2.023768781600399e-02  4.987607422949848e+00|1.239257705015184e-02  5.984221357500068e+00|1.577864249993155e-02  7.008265158907365e+00|-8.265158907365233e-03  8.021220897816528e+00|-2.122089781652825e-02  8.985366281904978e+00|1.463371809502156e-02 |
| ЛОС Диагональный | МСГ Диагональный |
| K = 12  1.000000045627483e+00|-4.562748312508802e-08  1.999997812433161e+00|2.187566839229049e-06  3.000024971179287e+00|-2.497117928657033e-05  3.999888371962689e+00|1.116280373110179e-04  5.000214107253460e+00|-2.141072534600497e-04  5.999889386714200e+00|1.106132858001985e-04  6.999814025591674e+00|1.859744083256842e-04  8.000272615660814e+00|-2.726156608137131e-04  8.999898627155819e+00|1.013728441812134e-04 | K = 81 без выхода по шагу(28 выход по шагу)  9.999967305600207e-01|3.269439979303179e-06  2.000104912741761e+00|-1.049127417607210e-04  2.999245319097894e+00|7.546809021063972e-04  4.001791965061567e+00|-1.791965061567424e-03  4.999273064481610e+00|7.269355183900927e-04  5.998027337749635e+00|1.972662250365076e-03  7.000595057108806e+00|-5.950571088062517e-04  8.002635627906882e+00|-2.635627906881766e-03  8.998326761440126e+00|1.673238559874335e-03 |
| ЛОС LU | МСГ LU |
| K = 1  1.000000001541802e+00|-1.541801797344533e-09  1.999999892414631e+00|1.075853686138828e-07  3.000001834419165e+00|-1.834419165458456e-06  3.999986835895187e+00|1.316410481333463e-05  5.000048473427121e+00|-4.847342712110958e-05  5.999900727779925e+00|9.927222007455327e-05  7.000114282045680e+00|-1.142820456800919e-04  7.999930837757521e+00|6.916224247888181e-05  9.000017115861644e+00|-1.711586164354628e-05 | K = 3  1.698350911616906e+00|-6.983509116169060e-01  -4.803678008776400e+01|5.003678008776400e+01  8.752460333697707e+02|-8.722460333697707e+02  -6.372201221577982e+03|6.376201221577982e+03  2.385269909057800e+04|-2.384769909057800e+04  -4.948664707659600e+04|4.949264707659600e+04  5.763819471662644e+04|-5.763119471662644e+04  -3.521806918074698e+04|3.522606918074698e+04  8.802784730877211e+03|-8.793784730877211e+03 |

|  |  |
| --- | --- |
| Матрица большей размерности 945 | |
| ЛОС | МСГ |
| K = 488 Time = 42 686 microseconds | K = 5689 Time = 781 031 microseconds |
| ЛОС Диагональный | МСГ Диагональный |
| K = 53 Time = 6 017 microseconds | K = 417 Time = 75 430 microseconds |
| ЛОС LU | МСГ LU |
| K = 8 Time = 2 142 microseconds | K = 18 Time = 5 123 microseconds |

|  |  |
| --- | --- |
| Матрица большей размерности 4545 | |
| ЛОС | МСГ |
| K = 2145 Time = 915 234 microseconds | K = 100000(max) Time = 67 379 436 microseconds |
| ЛОС Диагональный | МСГ Диагональный |
| K = 159 Time = 77 869 microseconds | K = 3744 Time = 2 850 002microseconds |
| ЛОС LU | МСГ LU |
| K = 8 Time = 10 618 microseconds | K = 18 Time = 27 820 microseconds |
|  |  |
|  |  |
| Влияние предобуславливания на время итерации | |
| ЛОС | МСГ |
| При диагональном время увеличилось в 1,14 раза | При диагональном время увеличилось в 1,1 раза |
| При LU время увеличилось в 3,01 раза | При LU время увеличилось в 2,02 раза |

1. Код программы

#pragma once

#include "includes.h"

template<class T>

///<summary> Matrix - class for lab 3 4m</summary>

///<remarks>conjugate gradient metod and local - optimal scheme 4 asymmetric matrix</remarks>

class Matrix

{

public:

Matrix() {}

~Matrix() {}

void OpenFile()

{

ifstream in("kuslau.txt");

in >> N >> Maxiter >> e;

in.close();

ig.resize(N + 1);

f.resize(N);

di.resize(N);

result.resize(N);

in.open("ig.txt");

for (size\_t i = 0; i <= N; i++)

in >> ig[i];

in.close();

M = ig[N];

jg.resize(M);

ggl.resize(M);

ggu.resize(M);

in.open("jg.txt");

for (size\_t i = 0; i < M; i++)

in >> jg[i];

in.close();

in.open("ggl.txt");

for (size\_t i = 0; i < M; i++)

in >> ggl[i];

in.close();

in.open("ggu.txt");

for (size\_t i = 0; i < M; i++)

in >> ggu[i];

in.close();

in.open("di.txt");

for (size\_t i = 0; i < N; i++)

in >> di[i];

in.close();

in.open("pr.txt");

for (size\_t i = 0; i < N; i++)

in >> f[i];

in.close();

}

//Set matrix dimension

///<param name = "n"> New matrix dimension</param>

void SetN(size\_t n) { N = n; }

//Get matrix dimention

size\_t n() { return N; }

//Возращает резултат операций

vector<T> Result() { return result; }

//Возращает количество операций

size\_t K() { return k; }

//Set max iteration

///<param name = "Max"> New max ieration </param>

void Setmaxiter(size\_t Max) { Maxiter = Max; }

//Get max iteration

size\_t maxiter() { return Maxiter; }

///<param name = "i"> i = 0 без предобуславливония

///i = 1 диагональное

///i = 2 Холесского</param>

void preconditioning(int i)

{

switch (i)

{

//Без предобуславливония

case 0:

break;

//Диагональное

case 1:

// factorizationLU(false);

factorizationLLT();

break;

//Холесского

// case 2:

// factorizationLLT(true);

//break;

//LU

case 3:

factorizationLU(true);

break;

default:

break;

}

}

//i = 0 без предобуславливония

//i = 1 диагональное

//i = 2 Холесского</param>

void preconditioning()

{

preconditioning(3);

}

void ConjugateGradient(int metod)

{

NormF = Norm(f);

switch (metod)

{

//Без

case 0:

{

vector<T> Buf(N), Buf1(N), d(N), Best = result;

r.resize(N);

MultTMatrixVector(f, Buf);

f = Buf;

MultMatrixVector(Best, Buf);

MultTMatrixVector(Buf, Buf1);

Subtraction(f, Buf1, r);

z = r;

T rLastScolar = Scolar(r);

T CoudA;

T min = 0.;

while (checkEnd(rLastScolar, CoudA))

{

MultMatrixVector(z, Buf);

MultTMatrixVector(Buf, d);

a = rLastScolar / Scolar(d, z);

MultVectorOnT(z, a, Buf);

Sum(Best, Buf, Buf1);

Best = Buf1;

rLast = r;

MultVectorOnT(d, a, Buf);

Subtraction(r, Buf, Buf1);

r = Buf1;

T lol = Scolar(r);

b = lol / rLastScolar;

rLastScolar = lol;

MultVectorOnT(z, b, Buf);

Sum(r, Buf, z);

if (min > CoudA || !min)

{

min = CoudA;

result = Best;

}

k++;

}

break;

}

// Диаганальное

case 1:

//LLT

case 2:

{

vector<T> Buf(N), Buf1(N), d(N);

r.resize(N);

MultMatrixVector(result, Buf);

Subtraction(f, Buf, d); //r = U^-t \* A ^t \* L^ -T \* L-1(f - Ax)

Diagonal(D, d, Buf);// L d1 = d

Diagonal(D, Buf, Buf1); //L^t d1 = d

MultTMatrixVector(Buf1, d); // A^t \* d = d1

Diagonal(D, d, r); // U^t r = d

z = r;

x.resize(N);

#pragma region Умножение\_x-\_=\_Ux

for (size\_t i = 0; i < N; i++)

{

x[i] += D[i] \* result[i];

}

#pragma endregion

k = 0;

//xbest = x;

vector<T> Best = x;

T rLastScolar = Scolar(r);

T CoudA;

T min = 0.;

while (checkEnd(rLastScolar, CoudA))

{

Diagonal(D, z, Buf);

MultMatrixVector(Buf, Buf1);

Diagonal(D, Buf1, Buf);

Diagonal(D, Buf, Buf1);

MultTMatrixVector(Buf1, Buf);

Diagonal(D, Buf, d);

T s = Scolar(d, z);

if (!s)

break;

a = rLastScolar / s;

MultVectorOnT(z, a, Buf);

Sum(x, Buf, Buf1);

x = Buf1;

rLast = r;

MultVectorOnT(d, a, Buf);

Subtraction(r, Buf, Buf1);

r = Buf1;

T rScolar = Scolar(r);

b = rScolar / rLastScolar;

MultVectorOnT(z, b, Buf);

Sum(r, Buf, z);

rLastScolar = rScolar;

k++;

if (min > CoudA || !min)

{

min = CoudA;

Best = x;

}

}

Diagonal(D, Best, result);

break;

}

//LU

case 3:

{

vector<T> Buf(N), Buf1(N), d(N);

r.resize(N);

MultMatrixVector(result, Buf);

Subtraction(f, Buf, Buf1); //r = U^-t \* A ^t \* L^ -T \* L-1(f - Ax)

Down(L, Buf1, Buf);// L d1 = d

Up(L, Buf, Buf1); //L^t d1 = d

MultTMatrixVector(Buf1, d); // A^t \* d = d1

Down(U, D, d, r); // U^t r = d

z = r;

x.resize(N);

#pragma region Умножение\_x-\_=\_Ux

for (size\_t i = 0; i < N; i++)

{

for (size\_t j = ig[i]; j < ig[i + 1]; j++)

x[i] += U[j] \* result[jg[j]];

x[i] += D[i] \* result[i];

}

#pragma endregion

k = 0;

xbest = x;

vector<T> Best = x;

T rLastScolar = Scolar(r);

T Coud;

T min = 0.;

while (checkEnd(rLastScolar, Coud))

{

Up(U, D, z, Buf);

MultMatrixVector(Buf, Buf1);

Down(L, Buf1, Buf);

Up(L, Buf, Buf1);

MultTMatrixVector(Buf1, Buf);

Down(U, D, Buf, d);

T s = Scolar(d, z);

if (!s)

break;

a = rLastScolar / s;

MultVectorOnT(z, a, Buf);

Sum(x, Buf, x);

MultVectorOnT(d, a, Buf);

Subtraction(r, Buf, Buf1);

r = Buf1;

T rScolar = Scolar(r);

b = rScolar / rLastScolar;

rLastScolar = rScolar;

MultVectorOnT(z, b, Buf);

Sum(r, Buf, z);

if (min > Coud || !min)

{

min = Coud;

Best = x;

}

k++;

}

Up(U, D, Best, result);

break;

}

default:

break;

}

}

void LocalOptimalScheme(int metod)

{

NormF = Norm(f);

switch (metod)

{

case 0:

{

vector<T> d(N), Buf(N), Buf1(N);

r.resize(N);

p.resize(N);

MultMatrixVector(result, Buf);

Subtraction(f, Buf, r);

z = r;

MultMatrixVector(z, p);

k = 0;

T ScolarP = Scolar(p);

T ScolarR = Scolar(r);

//while (checkEnd(ScolarR, ScolarP, ScolarR))

while (checkEnd(r))

{

if (!ScolarP)

break;

a = Scolar(p, r) / ScolarP;

MultVectorOnT(z, a, Buf);

Sum(result, Buf, Buf1);

result = Buf1;

MultVectorOnT(p, a, Buf);

Subtraction(r, Buf, Buf1);

r = Buf1;

T gg = Scolar(r);

MultMatrixVector(r, d);

b = -1.0 \* Scolar(p, d) / ScolarP;

MultVectorOnT(z, b, Buf);

Sum(r, Buf, z);

MultVectorOnT(p, b, Buf);

Sum(d, Buf, p);

ScolarP = Scolar(p);

k++;

}

break;

}

//DLLT

case 1:

//LLT

case 2:

{

vector<T> d(N), Buf(N), Buf1(N);

r.resize(N);

z.resize(N);

p.resize(N);

MultMatrixVector(result, Buf);

Subtraction(f, Buf, Buf1);

Diagonal(D, Buf1, r);

Diagonal(D, r, z);

MultMatrixVector(z, Buf);

Diagonal(D, Buf, p);

k = 0;

while (checkEnd(r))

{

T s = Scolar(p);

if (!s)

break;

a = Scolar(p, r) / s;

MultVectorOnT(z, a, Buf);

Sum(result, Buf, Buf1);

result = Buf1;

MultVectorOnT(p, a, Buf);

Subtraction(r, Buf, Buf1);

r = Buf1;

Diagonal(D, r, Buf);

MultMatrixVector(Buf, Buf1);

Diagonal(D, Buf1, d);

b = -1.0 \* Scolar(p, d) / s;

MultVectorOnT(z, b, Buf);

Diagonal(D, r, Buf1);

Sum(Buf1, Buf, z);

MultVectorOnT(p, b, Buf);

Sum(d, Buf, p);

k++;

}

break;

}

//LU

case 3:

{

vector<T> d(N), Buf(N), Buf1(N);

r.resize(N);

p.resize(N);

MultMatrixVector(result, Buf);

Subtraction(f, Buf, Buf1);

Down(L, Buf1, r);

Up(U, D, r, z);

MultMatrixVector(z, Buf);

Down(L, Buf, p);

k = 0;

while (checkEnd(r))

{

T s = Scolar(p);

if (!s)

break;

a = Scolar(p, r) / s;

MultVectorOnT(z, a, Buf);

Sum(result, Buf, Buf1);

result = Buf1;

MultVectorOnT(p, a, Buf);

Subtraction(r, Buf, Buf1);

r = Buf1;

Up(U, D, r, Buf);

MultMatrixVector(Buf, Buf1);

Down(L, Buf1, d);

b = -1.0 \* Scolar(p, d) / s;

Up(U, D, r, Buf);

MultVectorOnT(z, b, Buf1);

Sum(Buf, Buf1, z);

MultVectorOnT(p, b, Buf);

Sum(d, Buf, p);

k++;

}

break;

}

default:

break;

}

}

private:

vector<T> f; // Массив правой части

vector<T> di; //Диагональные элементы матрицы А

vector<T> ggu; //Верхний треугольник матрицы А в разреженном формате

vector<T> ggl; //Нижний треугольний матрицы А в разреженном формате

vector<T> L; // Матрица L

vector<T> U; // НЕ диагональыне элементы матрицы U

vector<T> D; // Диагональные элементы матрицы U

vector<size\_t> ig;// Массив индексов

vector<size\_t> jg; // Другой массив индексов

vector<T> result; //Результат x

vector<T> r, rLast, z, p, x, xbest;

size\_t N, M, Maxiter, k;

T e, a, b, NormF;

//Проверка окончания по невязки и по maxiter

///<param name = "ScolarR"> Входной параметр. Сколяр вектора R</param>

///<param name = "Return"> Выходной параметр. Невязка</param>

//Проверка конца у МСГ

bool checkEnd(T ScolarR, T &Return)

{

if (k == Maxiter)

return false;

Return = sqrt(ScolarR) / NormF;

if (e > Return)

return false;

return true;

}

///<param name = "ScolarR"> Входной параметр. Сколяр вектора R</param>

///<param name = "ScolarP"> Входной параметр. Сколяр вектора P</param>

///<param name = "ReturnR"> Выходной параметр. Сколяр вектора R</param>

//Проверка конца у ЛОС

bool checkEnd(T ScolarR, T ScolarP, T &ReturnR)

{

if (k == Maxiter)

return false;

T R = sqrt(ScolarR) / NormF;

T g = Scolar(r);

if (e > R)

return false;

a = Scolar(p, r) / Scolar(p);

T h = pow(a, 2) \* ScolarP;

ReturnR = ScolarR - h;

return true;

}

bool checkEnd(vector<T> r)

{

if (k == Maxiter)

return false;

if (e > Norm(r) / NormF)

return false;

return true;

}

//Прямой ход

void Down(vector<T> Matrix, vector<T> Diagonal, vector<T> R, vector<T> &x)

{

for (size\_t i = 0; i < N; i++)

{

T sum = 0;

for (size\_t j = ig[i]; j < ig[i + 1]; j++)

sum += Matrix[j] \* x[jg[j]];

x[i] = (R[i] - sum) / Diagonal[i];

}

}

void Down(vector<T> Matrix, vector<T> R, vector<T> &x)

{

for (size\_t i = 0; i < N; i++)

{

T sum = 0;

for (size\_t j = ig[i]; j < ig[i + 1]; j++)

sum += Matrix[j] \* x[jg[j]];

x[i] = R[i] - sum;

}

}

//Обратный ход

void Up(vector<T> Matrix, vector<T> Diagonal, vector<T> R, vector<T> &v)

{

v = R;

for (int i = N - 1; i >= 0; i--)

{

v[i] /= Diagonal[i];

for (size\_t j = ig[i]; j < ig[i + 1]; j++)

{

size\_t p = jg[j];

v[p] -= Matrix[j] \* v[i];

}

}

}

void Up(vector<T> Matrix, vector<T> R, vector<T> &v)

{

v = R;

for (int i = N - 1; i >= 0; i--)

{

for (size\_t j = ig[i]; j < ig[i + 1]; j++)

{

size\_t p = jg[j];

v[p] -= Matrix[j] \* v[i];

}

}

}

//Умноженик числа на вектор

inline void MultVectorOnT(vector<T> a, T b, vector<T> &c)

{

for (size\_t i = 0; i < a.size(); i++)

c[i] = a[i] \* b;

}

//Mult matrix and vector f

///<param name = "f"> Matrix be multtiplication on thix vector.</param>

vector<T> MultMatrixVector(vector<T> f)

{

vector<T> v(N);

for (size\_t i = 0; i < N; i++)

{

for (size\_t j = ig[i]; j < ig[i + 1]; j++)

{

v[i] += ggl[j] \* f[jg[j]];

v[jg[j]] += ggu[j] \* f[i];

}

v[i] += di[i] \* f[i];

}

return v;

}

vector<T> MultTMatrixVector(vector<T> f)

{

vector<T> v(N);

for (size\_t i = 0; i < N; i++)

{

for (size\_t j = ig[i]; j < ig[i + 1]; j++)

{

v[i] += ggu[j] \* f[jg[j]];

v[jg[j]] += ggl[j] \* f[i];

}

v[i] += di[i] \* f[i];

}

return v;

}

void MultMatrixVector(vector<T> f, vector<T> &v)

{

v.clear();

v.resize(N);

for (size\_t i = 0; i < N; i++)

{

for (size\_t j = ig[i]; j < ig[i + 1]; j++)

{

v[i] += ggl[j] \* f[jg[j]];

v[jg[j]] += ggu[j] \* f[i];

}

v[i] += di[i] \* f[i];

}

}

void MultTMatrixVector(vector<T> f, vector<T> &v)

{

v.clear();

v.resize(N);

for (size\_t i = 0; i < N; i++)

{

for (size\_t j = ig[i]; j < ig[i + 1]; j++)

{

v[i] += ggu[j] \* f[jg[j]];

v[jg[j]] += ggl[j] \* f[i];

}

v[i] += di[i] \* f[i];

}

}

//Residual a - b

inline void Subtraction(vector<T> a, vector<T> b, vector<T> &v)

{

for (size\_t i = 0; i < a.size(); i++)

v[i] = a[i] - b[i];

}

//Sum a + b

inline void Sum(vector<T> a, vector<T> b, vector<T> &v)

{

for (size\_t i = 0; i < a.size(); i++)

v[i] = a[i] + b[i];

}

//Норма вектора а

inline T Norm(vector<T> a)

{

return sqrt(Scolar(a));

}

//Скалярное произведение 1 элемента

inline T Scolar(vector<T> a)

{

T sum = 0;

for (size\_t i = 0; i < N; i++)

sum += pow(a[i], 2);

return sum;

}

//Скалярное произведение 2 элемента

inline T Scolar(vector<T> a, vector<T> b)

{

T sum = 0;

for (size\_t i = 0; i < N; i++)

sum += a[i] \* b[i];

return sum;

}

//factorizationLU is a method in the Matrix class. МБ даже работает

///<param name = "h"> if this param true, then LU, else diagonal LU </param>

void factorizationLU(bool h)

{

if (h)

{

L.resize(ggl.size());

D.resize(di.size());

U.resize(ggu.size());

for (size\_t i = 0; i < N; i++)

{

T sum = 0;

for (size\_t j = ig[i]; j < ig[i + 1]; j++)

{

T sum1 = 0, sum2 = 0;

int jj = jg[j];

for (size\_t k = ig[i], k2 = ig[jj]; k < j && k2 < ig[jj + 1];)

{

int p1 = jg[k];

int p2 = jg[k2];

if (p1 == p2)

{

sum1 += L[k] \* U[k2];

sum2 += L[k2] \* U[k];

k++; k2++;

}

else

{

if (p1 < p2)

k++;

else

k2++;

}

}

L[j] = (ggl[j] - sum1) / D[jg[j]];

U[j] = ggu[j] - sum2;

sum += L[j] \* U[j];

}

D[i] = di[i] - sum;

}

}

else

{

L.resize(ggl.size());

U.resize(ggu.size());

D = di;

}

}

//factorizationLU is a method in the Matrix class. МБ даже работает

///<param name = "h"> if this param true, then Holisskigo, else diagonal Holisskigo </param>

void factorizationLLT()

{

L.resize(ggl.size());

D.resize(N);

for (size\_t i = 0; i < N; i++)

D[i] = sqrt(di[i]);

}

};

#include "includes.h"

#include "Matrix.h"

#include "chrono"

#include "ctime"

enum {

Cc = 0,

DLUSQ = 1,

// LLT = 2,

LU = 3

};

//#define TIME

#define NOper 100

const int metod = LU;

int main()

{

#ifdef TIME

std::chrono::time\_point<std::chrono::system\_clock> start, end;

Matrix<double> p;

p.OpenFile();

start = chrono::system\_clock::now();

for (size\_t i = 0; i < NOper; i++)

{

Matrix<double> g = p;

g.preconditioning(metod);

//g.ConjugateGradient(metod);

g.LocalOptimalScheme(metod);

}

end = chrono::system\_clock::now();

auto d = end - start;

auto delt = chrono::duration\_cast<chrono::microseconds>(end - start).count();

cout << "\t Time = " << delt/NOper << " microseconds\n";

system("Pause");

#else

Matrix<double> g;

g.OpenFile();

g.preconditioning(metod);

g.ConjugateGradient(metod);

//g.LocalOptimalScheme(metod);

vector<double> out = g.Result();

int d;

cout << g.K() << endl;

cout.setf(ios::scientific);

cout.precision(15);

for (size\_t i = 0; i < out.size(); i++)

cout << out[i] << "|" << i - out[i] + 1 << "\n";

cout.unsetf(ios::scientific);

cin >> d;

#endif // TIME

return 0;

}