|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | Министерство науки и высшего образования Российской Федерации | | | | | | |  | | |
|  | | |  | | | | | | |  | | |
|  | | ФГБОУ ВО «Пермский государственный национальный исследовательский университет» | | | | | | | | |  | |
|  | | | | |  | | |  | | | | |
|  | | | | ОТЧЕТ  по лабораторной работе №3  «Итерационные методы решения систем линейных алгебраических уравнений»  по дисциплине «Численные методы» | | | | |  | | | |
|  | | | | |  | | |  | | | | |
|  | Работу выполнили  студенты гр. ПМИ-3,4  Трегубова И.Д.,  Третьяков В.Н.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 | | | | |  | Проверил заведующий кафедрой ПМИ,  профессор Русаков С.В.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 | | | | |  |
|  | | | | |  | | |  | | | | |
|  | | | | | Пермь 2020 | | |  | | | | |

**Задание.**

1. Решить систему линейных алгебраических уравнений  итерационными методами с критерием остановки  (по невязке):
   * методом простой итерации;
   * градиентным методом наискорейшего спуска;
   * методом ПВР;
   * методом сопряженных градиентов.

В качества начального приближения выбирать вектор , который во всех вариантах принять равным .

1. Для каждого метода получить число итераций, необходимое для достижения требуемой точности (по невязке), выдавая (на печать) на каждом шаге (или через заданное число шагов)

- значение параметров итерационного метода;

- значение нормы невязки;

- оценку нормы матрицы перехода *q*;

- оценку погрешности приближенного решения.

3) Оценку нормы матрицы перехода осуществлять по формуле



4) В методе простой итерации значения итерационного параметра вычислять по формуле



1. В методе ПВР получить решение при оптимальном значении параметра , которое необходимо определить, варьируя параметр в диапазоне (0, 2) с шагом 0.1, и производя вычисления с критерием остановки(или по критерию минимальности нормы вектора невязки при заданном числе итераций).
2. Провести анализ эффективности рассматриваемых методов.
3. Сравнить решение, полученное итерационным методом, с решением полученным прямым методом.
4. Сравнить фактическое число итераций, необходимое для достижения заданной точности, с теоретической оценкой, вычислив число обусловленности в эвклидовой норме.

**Исходные данные**

**Вариант 27 b)** (Трегубова)

 

**Вариант 28 b)** (Третьяков)

 

**Решение**

Условие окончания каждого из методов: 

**Метод простой итерации**

Оцениваем число итераций по формуле: 

Расчетные формулы:





**Метод наискорейшего спуска**

Оцениваем число итераций по формуле: 

Расчетные формулы:





**Метод ПВР**

Оцениваем число итераций по формуле: 

Расчетные формулы:



**Метод сопряженных градиентов**

Оцениваем число итераций по формуле: 

Расчетные формулы:

На первом шаге берем 



**Тестирование и вывод**

|  |
| --- |
| **Вариант 27b** |
| Variant = 27b  b:  1.0000000 2.0000000 3.0000000 4.0000000  A:  7.8000000 7.0000000 -0.3000000 1.6000000  7.0000000 9.8000000 2.9000000 -0.1000000  -0.3000000 2.9000000 4.9000000 -1.2000000  1.6000000 -0.1000000 -1.2000000 5.0000000  Норма матрицы = 16.2779756  Метод простой итерации  | Норма| Оценка|  Itr| Tau| q| невязки| погрешности| x[1]| x[2]| x[3]| x[4]|  1| 0.1106| 0.442| 9.0433198| 0.5576846| 0.11058| 0.22116| 0.33174| 0.44232|  2| 0.1106| 0.508| 3.0461214| 0.2176528| -0.11266| 0.01557| 0.47517| 0.66698|  3| 0.1106| 0.698| 1.7204701| 0.0679592| -0.01920| 0.16206| 0.62922| 0.82368|  4| 0.1106| 0.844| 1.3300676| 0.0244410| -0.14236| 0.02980| 0.67671| 0.89927|  5| 0.1106| 0.696| 0.8669262| 0.0402623| -0.06871| 0.12180| 0.74683| 0.95970|  6| 0.1106| 0.940| 0.7951733| 0.0054771| -0.13816| 0.03527| 0.75992| 0.98401|  7| 0.1106| 0.660| 0.5120526| 0.0294511| -0.08461| 0.09234| 0.79458| 1.00794|  8| 0.1106| 1.000| 0.5088924| -0.0000111| -0.12450| 0.03526| 0.79711| 1.01440|  9| 0.1106| 0.659| 0.3324093| 0.0194416| -0.08686| 0.07018| 0.81611| 1.02405|  10| 0.1106| 1.011| 0.3357798| -0.0004026| -0.10979| 0.03214| 0.81615| 1.02461|  11| 0.1106| 0.689| 0.2303425| 0.0118469| -0.08360| 0.05306| 0.82768| 1.02850|  12| 0.1106| 0.980| 0.2260758| 0.0005136| -0.09650| 0.02738| 0.82764| 1.02737|  13| 0.1106| 0.713| 0.1608978| 0.0073781| -0.07820| 0.03952| 0.83527| 1.02886|  14| 0.1106| 0.962| 0.1550702| 0.0006875| -0.08508| 0.02190| 0.83569| 1.02743|  15| 0.1106| 0.735| 0.1139886| 0.0046618| -0.07213| 0.02856| 0.84111| 1.02787|  16| 0.1106| 0.951| 0.1085226| 0.0006369| -0.07540| 0.01624| 0.84194| 1.02657|  17| 0.1106| 0.759| 0.0823169| 0.0029750| -0.06606| 0.01952| 0.84600| 1.02654|  18| 0.1106| 0.943| 0.0776690| 0.0005370| -0.06717| 0.01071| 0.84711| 1.02545|  19| 0.1106| 0.783| 0.0608115| 0.0019131| -0.06028| 0.01194| 0.85026| 1.02521|  20| 0.1106| 0.936| 0.0569678| 0.0004414| -0.06013| 0.00549| 0.85150| 1.02431|  21| 0.1106| 0.808| 0.0460470| 0.0012387| -0.05492| 0.00551| 0.85403| 1.02398|  22| 0.1106| 0.930| 0.0428745| 0.0003627| -0.05408| 0.00066| 0.85531| 1.02325|  23| 0.1106| 0.834| 0.0357493| 0.0008076| -0.05003| -0.00000| 0.85738| 1.02288|  24| 0.1106| 0.926| 0.0331101| 0.0003008| -0.04883| -0.00375| 0.85863| 1.02228|  25| 0.1106| 0.858| 0.0284234| 0.0005309| -0.04562| -0.00477| 0.86036| 1.02191|  26| 0.1106| 0.921| 0.0262038| 0.0002523| -0.04426| -0.00773| 0.86154| 1.02140|  27| 0.1106| 0.881| 0.0230904| 0.0003525| -0.04165| -0.00892| 0.86301| 1.02106|  28| 0.1106| 0.918| 0.0212036| 0.0002140| -0.04027| -0.01131| 0.86410| 1.02062|  29| 0.1106| 0.901| 0.0191098| 0.0002370| -0.03811| -0.01254| 0.86536| 1.02030|  30| 0.1106| 0.915| 0.0174909| 0.0001834| -0.03676| -0.01451| 0.86636| 1.01993|  31| 0.1106| 0.918| 0.0160612| 0.0001618| -0.03495| -0.01571| 0.86744| 1.01964|  32| 0.1106| 0.913| 0.0146617| 0.0001583| -0.03369| -0.01737| 0.86835| 1.01932|  33| 0.1106| 0.932| 0.0136673| 0.0001125| -0.03215| -0.01850| 0.86929| 1.01905|  34| 0.1106| 0.911| 0.0124503| 0.0001375| -0.03098| -0.01991| 0.87010| 1.01877|  35| 0.1106| 0.943| 0.0117437| 0.0000799| -0.02966| -0.02096| 0.87093| 1.01853|  36| 0.1106| 0.909| 0.0106806| 0.0001200| -0.02859| -0.02216| 0.87165| 1.01829|  37| 0.1106| 0.952| 0.0101664| 0.0000581| -0.02745| -0.02312| 0.87238| 1.01808|  38| 0.1106| 0.908| 0.0092346| 0.0001051| -0.02648| -0.02416| 0.87302| 1.01786|  39| 0.1106| 0.958| 0.0088506| 0.0000434| -0.02549| -0.02503| 0.87366| 1.01767|  40| 0.1106| 0.907| 0.0080320| 0.0000923| -0.02462| -0.02593| 0.87423| 1.01748|  41| 0.1106| 0.963| 0.0077376| 0.0000333| -0.02375| -0.02671| 0.87479| 1.01731|  42| 0.1106| 0.907| 0.0070170| 0.0000812| -0.02298| -0.02750| 0.87530| 1.01715|  43| 0.1106| 0.967| 0.0067857| 0.0000261| -0.02222| -0.02820| 0.87579| 1.01700|  44| 0.1106| 0.906| 0.0061505| 0.0000715| -0.02153| -0.02889| 0.87625| 1.01685|  45| 0.1106| 0.970| 0.0059645| 0.0000210| -0.02086| -0.02951| 0.87668| 1.01672|  46| 0.1106| 0.912| 0.0054387| 0.0000592| -0.02025| -0.03012| 0.87708| 1.01659|  47| 0.1106| 0.965| 0.0052514| 0.0000211| -0.01966| -0.03067| 0.87747| 1.01647|  48| 0.1106| 0.920| 0.0048295| 0.0000475| -0.01912| -0.03120| 0.87782| 1.01636|  49| 0.1106| 0.958| 0.0046292| 0.0000226| -0.01860| -0.03169| 0.87816| 1.01626|  50| 0.1106| 0.925| 0.0042836| 0.0000389| -0.01811| -0.03217| 0.87848| 1.01616|  51| 0.1106| 0.953| 0.0040844| 0.0000224| -0.01766| -0.03260| 0.87877| 1.01606|  52| 0.1106| 0.929| 0.0037963| 0.0000324| -0.01723| -0.03302| 0.87905| 1.01598|  53| 0.1106| 0.950| 0.0036059| 0.0000214| -0.01682| -0.03340| 0.87932| 1.01589|  54| 0.1106| 0.932| 0.0033624| 0.0000274| -0.01645| -0.03377| 0.87956| 1.01581|  55| 0.1106| 0.947| 0.0031850| 0.0000200| -0.01609| -0.03411| 0.87980| 1.01574|  56| 0.1106| 0.935| 0.0029769| 0.0000234| -0.01575| -0.03443| 0.88002| 1.01567|  57| 0.1106| 0.945| 0.0028141| 0.0000183| -0.01544| -0.03474| 0.88022| 1.01561|  58| 0.1106| 0.936| 0.0026347| 0.0000202| -0.01514| -0.03502| 0.88042| 1.01555|  59| 0.1106| 0.944| 0.0024870| 0.0000166| -0.01486| -0.03529| 0.88060| 1.01549|  60| 0.1106| 0.937| 0.0023313| 0.0000175| -0.01460| -0.03554| 0.88077| 1.01544|  61| 0.1106| 0.943| 0.0021983| 0.0000149| -0.01435| -0.03578| 0.88093| 1.01539|  62| 0.1106| 0.938| 0.0020625| 0.0000153| -0.01412| -0.03600| 0.88108| 1.01534|  63| 0.1106| 0.942| 0.0019434| 0.0000134| -0.01391| -0.03621| 0.88122| 1.01529|  64| 0.1106| 0.939| 0.0018244| 0.0000134| -0.01370| -0.03641| 0.88136| 1.01525|  65| 0.1106| 0.942| 0.0017181| 0.0000119| -0.01351| -0.03660| 0.88148| 1.01521|  66| 0.1106| 0.939| 0.0016137| 0.0000117| -0.01333| -0.03677| 0.88160| 1.01518|  67| 0.1106| 0.941| 0.0015191| 0.0000106| -0.01316| -0.03694| 0.88171| 1.01514|  68| 0.1106| 0.940| 0.0014273| 0.0000103| -0.01300| -0.03709| 0.88182| 1.01511|  69| 0.1106| 0.941| 0.0013432| 0.0000094| -0.01284| -0.03723| 0.88192| 1.01508|  70| 0.1106| 0.940| 0.0012623| 0.0000091| -0.01270| -0.03737| 0.88201| 1.01505|  71| 0.1106| 0.941| 0.0011876| 0.0000084| -0.01257| -0.03750| 0.88209| 1.01502|  72| 0.1106| 0.940| 0.0011163| 0.0000080| -0.01244| -0.03762| 0.88218| 1.01500|  73| 0.1106| 0.941| 0.0010502| 0.0000074| -0.01233| -0.03773| 0.88225| 1.01497|  74| 0.1106| 0.940| 0.0009872| 0.0000071| -0.01222| -0.03784| 0.88233| 1.01495|  75| 0.1106| 0.941| 0.0009286| 0.0000066| -0.01211| -0.03794| 0.88239| 1.01493|  76| 0.1106| 0.940| 0.0008730| 0.0000062| -0.01201| -0.03803| 0.88246| 1.01491|  77| 0.1106| 0.941| 0.0008211| 0.0000058| -0.01192| -0.03812| 0.88252| 1.01489|  78| 0.1106| 0.940| 0.0007720| 0.0000055| -0.01183| -0.03821| 0.88257| 1.01487|  79| 0.1106| 0.940| 0.0007261| 0.0000052| -0.01175| -0.03828| 0.88263| 1.01485|  80| 0.1106| 0.940| 0.0006827| 0.0000049| -0.01168| -0.03836| 0.88268| 1.01484|  81| 0.1106| 0.940| 0.0006420| 0.0000046| -0.01160| -0.03843| 0.88272| 1.01482|  82| 0.1106| 0.940| 0.0006037| 0.0000043| -0.01154| -0.03849| 0.88277| 1.01481|  83| 0.1106| 0.940| 0.0005677| 0.0000040| -0.01147| -0.03855| 0.88281| 1.01480|  84| 0.1106| 0.940| 0.0005338| 0.0000038| -0.01141| -0.03861| 0.88285| 1.01478|  85| 0.1106| 0.940| 0.0005020| 0.0000036| -0.01136| -0.03867| 0.88289| 1.01477|  86| 0.1106| 0.940| 0.0004721| 0.0000034| -0.01130| -0.03872| 0.88292| 1.01476|  87| 0.1106| 0.940| 0.0004439| 0.0000032| -0.01125| -0.03877| 0.88295| 1.01475|  88| 0.1106| 0.940| 0.0004174| 0.0000030| -0.01121| -0.03881| 0.88298| 1.01474|  89| 0.1106| 0.940| 0.0003925| 0.0000028| -0.01116| -0.03885| 0.88301| 1.01473|  90| 0.1106| 0.940| 0.0003691| 0.0000026| -0.01112| -0.03889| 0.88304| 1.01473|  91| 0.1106| 0.940| 0.0003471| 0.0000025| -0.01108| -0.03893| 0.88306| 1.01472|  92| 0.1106| 0.940| 0.0003264| 0.0000023| -0.01105| -0.03897| 0.88309| 1.01471|  93| 0.1106| 0.940| 0.0003069| 0.0000022| -0.01101| -0.03900| 0.88311| 1.01470|  94| 0.1106| 0.940| 0.0002886| 0.0000021| -0.01098| -0.03903| 0.88313| 1.01470|  95| 0.1106| 0.940| 0.0002714| 0.0000019| -0.01095| -0.03906| 0.88315| 1.01469|  96| 0.1106| 0.940| 0.0002552| 0.0000018| -0.01092| -0.03909| 0.88317| 1.01468|  97| 0.1106| 0.940| 0.0002400| 0.0000017| -0.01089| -0.03911| 0.88319| 1.01468|  98| 0.1106| 0.940| 0.0002257| 0.0000016| -0.01087| -0.03914| 0.88320| 1.01467|  99| 0.1106| 0.940| 0.0002122| 0.0000015| -0.01084| -0.03916| 0.88322| 1.01467|  100| 0.1106| 0.940| 0.0001996| 0.0000014| -0.01082| -0.03918| 0.88323| 1.01466|  101| 0.1106| 0.940| 0.0001877| 0.0000013| -0.01080| -0.03920| 0.88325| 1.01466|  102| 0.1106| 0.940| 0.0001765| 0.0000013| -0.01078| -0.03922| 0.88326| 1.01466|  103| 0.1106| 0.940| 0.0001659| 0.0000012| -0.01076| -0.03924| 0.88327| 1.01465|  104| 0.1106| 0.940| 0.0001560| 0.0000011| -0.01075| -0.03926| 0.88329| 1.01465|  105| 0.1106| 0.940| 0.0001467| 0.0000010| -0.01073| -0.03927| 0.88330| 1.01464|  106| 0.1106| 0.940| 0.0001380| 0.0000010| -0.01071| -0.03929| 0.88331| 1.01464|  107| 0.1106| 0.940| 0.0001297| 0.0000009| -0.01070| -0.03930| 0.88332| 1.01464|  108| 0.1106| 0.940| 0.0001220| 0.0000009| -0.01068| -0.03931| 0.88332| 1.01464|  109| 0.1106| 0.940| 0.0001147| 0.0000008| -0.01067| -0.03933| 0.88333| 1.01463|  110| 0.1106| 0.940| 0.0001079| 0.0000008| -0.01066| -0.03934| 0.88334| 1.01463|  111| 0.1106| 0.940| 0.0001014| 0.0000007| -0.01065| -0.03935| 0.88335| 1.01463|  112| 0.1106| 0.940| 0.0000954| 0.0000007| -0.01064| -0.03936| 0.88336| 1.01463|  Метод наискорейшего спуска  | Норма| Оценка|  Itr| Tau| q| невязки| погрешности| x[1]| x[2]| x[3]| x[4]|  1| 0.1399| 0.559| 7.1500000| 0.4405594| 0.13986| 0.27972| 0.41958| 0.55944|  2| 0.0791| 0.407| 4.2440316| 0.3316520| -0.08295| 0.05193| 0.48648| 0.67886|  3| 0.1400| 0.816| 1.5350144| 0.0419215| -0.03489| 0.15399| 0.66227| 0.86473|  4| 0.0792| 0.468| 1.2107626| 0.0988379| -0.11334| 0.06696| 0.68884| 0.90767|  5| 0.1410| 0.787| 0.4982613| 0.0185502| -0.08941| 0.09943| 0.75732| 0.97482|  6| 0.0799| 0.522| 0.4518681| 0.0327408| -0.11587| 0.06369| 0.76880| 0.98982|  7| 0.1460| 0.822| 0.1984051| 0.0063722| -0.10056| 0.07192| 0.79817| 1.01394|  8| 0.0830| 0.605| 0.2102628| 0.0115887| -0.10902| 0.05414| 0.80373| 1.01860|  9| 0.1618| 0.859| 0.0918679| 0.0025142| -0.09564| 0.05472| 0.81899| 1.02690|  10| 0.0881| 0.804| 0.1354920| 0.0029938| -0.09867| 0.04245| 0.82180| 1.02750|  11| 0.1737| 1.062| 0.0728653| -0.0007584| -0.08564| 0.04145| 0.83109| 1.02937|  12| 0.0896| 0.784| 0.1106956| 0.0028203| -0.08742| 0.03124| 0.83261| 1.02862|  13| 0.1757| 1.156| 0.0653107| -0.0015927| -0.07562| 0.03020| 0.83924| 1.02845|  14| 0.0898| 0.742| 0.0949463| 0.0030387| -0.07699| 0.02143| 0.84028| 1.02745|  15| 0.1759| 1.178| 0.0571291| -0.0015556| -0.06667| 0.02052| 0.84561| 1.02682|  16| 0.0898| 0.731| 0.0818594| 0.0027735| -0.06783| 0.01298| 0.84644| 1.02586|  17| 0.1759| 1.183| 0.0494960| -0.0013819| -0.05890| 0.01219| 0.85092| 1.02520|  18| 0.0898| 0.728| 0.0706024| 0.0024287| -0.05988| 0.00569| 0.85162| 1.02435|  19| 0.1759| 1.185| 0.0427456| -0.0012006| -0.05219| 0.00501| 0.85546| 1.02375|  20| 0.0898| 0.727| 0.0608856| 0.0021020| -0.05303| -0.00058| 0.85605| 1.02301|  21| 0.1759| 1.185| 0.0368750| -0.0010367| -0.04640| -0.00116| 0.85935| 1.02248|  22| 0.0898| 0.727| 0.0524974| 0.0018130| -0.04713| -0.00598| 0.85985| 1.02185|  23| 0.1759| 1.185| 0.0317964| -0.0008935| -0.04141| -0.00649| 0.86269| 1.02139|  24| 0.0898| 0.727| 0.0452578| 0.0015622| -0.04204| -0.01064| 0.86313| 1.02084|  25| 0.1759| 1.185| 0.0274109| -0.0007697| -0.03712| -0.01107| 0.86558| 1.02045|  26| 0.0898| 0.727| 0.0390114| 0.0013456| -0.03766| -0.01464| 0.86595| 1.01997|  27| 0.1759| 1.185| 0.0236266| -0.0006630| -0.03342| -0.01502| 0.86806| 1.01963|  28| 0.0898| 0.726| 0.0336231| 0.0011589| -0.03389| -0.01810| 0.86838| 1.01923|  29| 0.1759| 1.185| 0.0203624| -0.0005710| -0.03024| -0.01842| 0.87020| 1.01893|  30| 0.0898| 0.726| 0.0289761| 0.0009981| -0.03064| -0.02107| 0.87047| 1.01858|  31| 0.1759| 1.185| 0.0175475| -0.0004918| -0.02750| -0.02135| 0.87204| 1.01833|  32| 0.0898| 0.726| 0.0249692| 0.0008597| -0.02784| -0.02363| 0.87228| 1.01803|  33| 0.1759| 1.185| 0.0151205| -0.0004235| -0.02513| -0.02387| 0.87362| 1.01781|  34| 0.0898| 0.726| 0.0215148| 0.0007404| -0.02543| -0.02583| 0.87383| 1.01755|  35| 0.1759| 1.185| 0.0130282| -0.0003648| -0.02310| -0.02604| 0.87499| 1.01737|  36| 0.0898| 0.726| 0.0185370| 0.0006376| -0.02335| -0.02773| 0.87517| 1.01714|  37| 0.1759| 1.185| 0.0112247| -0.0003142| -0.02135| -0.02791| 0.87616| 1.01698|  38| 0.0898| 0.726| 0.0159705| 0.0005492| -0.02157| -0.02937| 0.87632| 1.01679|  39| 0.1759| 1.185| 0.0096704| -0.0002706| -0.01984| -0.02952| 0.87718| 1.01665|  40| 0.0898| 0.726| 0.0137587| 0.0004730| -0.02003| -0.03078| 0.87731| 1.01648|  41| 0.1759| 1.185| 0.0083310| -0.0002330| -0.01854| -0.03091| 0.87805| 1.01636|  42| 0.0898| 0.726| 0.0118527| 0.0004073| -0.01870| -0.03199| 0.87816| 1.01622|  43| 0.1759| 1.185| 0.0071768| -0.0002007| -0.01742| -0.03210| 0.87880| 1.01612|  44| 0.0898| 0.726| 0.0102104| 0.0003508| -0.01756| -0.03304| 0.87890| 1.01599|  45| 0.1759| 1.185| 0.0061823| -0.0001729| -0.01645| -0.03313| 0.87945| 1.01591|  46| 0.0898| 0.726| 0.0087954| 0.0003022| -0.01657| -0.03394| 0.87953| 1.01580|  47| 0.1759| 1.185| 0.0053254| -0.0001489| -0.01562| -0.03402| 0.88001| 1.01572|  48| 0.0898| 0.726| 0.0075763| 0.0002602| -0.01573| -0.03471| 0.88008| 1.01563|  49| 0.1759| 1.185| 0.0045872| -0.0001282| -0.01491| -0.03478| 0.88049| 1.01557|  50| 0.0898| 0.726| 0.0065260| 0.0002241| -0.01500| -0.03538| 0.88055| 1.01549|  51| 0.1759| 1.185| 0.0039513| -0.0001104| -0.01429| -0.03544| 0.88090| 1.01543|  52| 0.0898| 0.726| 0.0056212| 0.0001930| -0.01437| -0.03595| 0.88095| 1.01536|  53| 0.1759| 1.185| 0.0034034| -0.0000951| -0.01376| -0.03601| 0.88126| 1.01531|  54| 0.0898| 0.726| 0.0048417| 0.0001662| -0.01383| -0.03645| 0.88130| 1.01526|  55| 0.1759| 1.185| 0.0029315| -0.0000819| -0.01330| -0.03649| 0.88156| 1.01521|  56| 0.0898| 0.726| 0.0041703| 0.0001432| -0.01336| -0.03687| 0.88160| 1.01516|  57| 0.1759| 1.185| 0.0025249| -0.0000705| -0.01291| -0.03691| 0.88183| 1.01513|  58| 0.0898| 0.726| 0.0035920| 0.0001233| -0.01296| -0.03724| 0.88186| 1.01508|  59| 0.1759| 1.185| 0.0021748| -0.0000608| -0.01257| -0.03728| 0.88205| 1.01505|  60| 0.0898| 0.726| 0.0030938| 0.0001062| -0.01262| -0.03756| 0.88208| 1.01502|  61| 0.1759| 1.185| 0.0018731| -0.0000523| -0.01228| -0.03759| 0.88225| 1.01499|  62| 0.0898| 0.726| 0.0026647| 0.0000915| -0.01232| -0.03783| 0.88227| 1.01496|  63| 0.1759| 1.185| 0.0016133| -0.0000451| -0.01203| -0.03786| 0.88242| 1.01493|  64| 0.0898| 0.726| 0.0022950| 0.0000788| -0.01206| -0.03806| 0.88244| 1.01491|  65| 0.1759| 1.185| 0.0013895| -0.0000388| -0.01181| -0.03809| 0.88256| 1.01489|  66| 0.0898| 0.726| 0.0019767| 0.0000678| -0.01184| -0.03827| 0.88258| 1.01486|  67| 0.1759| 1.185| 0.0011968| -0.0000334| -0.01163| -0.03829| 0.88269| 1.01485|  68| 0.0898| 0.726| 0.0017025| 0.0000584| -0.01165| -0.03844| 0.88270| 1.01483|  69| 0.1759| 1.185| 0.0010308| -0.0000288| -0.01147| -0.03846| 0.88280| 1.01481|  70| 0.0898| 0.726| 0.0014663| 0.0000503| -0.01149| -0.03859| 0.88281| 1.01479|  71| 0.1759| 1.185| 0.0008878| -0.0000248| -0.01133| -0.03860| 0.88289| 1.01478|  72| 0.0898| 0.726| 0.0012629| 0.0000433| -0.01134| -0.03872| 0.88290| 1.01476|  73| 0.1759| 1.185| 0.0007646| -0.0000214| -0.01121| -0.03873| 0.88297| 1.01475|  74| 0.0898| 0.726| 0.0010877| 0.0000373| -0.01122| -0.03883| 0.88298| 1.01474|  75| 0.1759| 1.185| 0.0006585| -0.0000184| -0.01111| -0.03884| 0.88304| 1.01473|  76| 0.0898| 0.726| 0.0009368| 0.0000321| -0.01112| -0.03893| 0.88305| 1.01472|  77| 0.1759| 1.185| 0.0005672| -0.0000158| -0.01102| -0.03894| 0.88310| 1.01471|  78| 0.0898| 0.726| 0.0008068| 0.0000277| -0.01103| -0.03901| 0.88310| 1.01470|  79| 0.1759| 1.185| 0.0004885| -0.0000136| -0.01094| -0.03902| 0.88315| 1.01470|  80| 0.0898| 0.726| 0.0006949| 0.0000238| -0.01095| -0.03908| 0.88315| 1.01469|  81| 0.1759| 1.185| 0.0004207| -0.0000117| -0.01088| -0.03909| 0.88319| 1.01468|  82| 0.0898| 0.726| 0.0005985| 0.0000205| -0.01088| -0.03914| 0.88320| 1.01467|  83| 0.1759| 1.185| 0.0003623| -0.0000101| -0.01082| -0.03915| 0.88323| 1.01467|  84| 0.0898| 0.726| 0.0005154| 0.0000177| -0.01083| -0.03919| 0.88324| 1.01466|  85| 0.1759| 1.185| 0.0003121| -0.0000087| -0.01077| -0.03920| 0.88326| 1.01466|  86| 0.0898| 0.726| 0.0004439| 0.0000152| -0.01078| -0.03924| 0.88327| 1.01465|  87| 0.1759| 1.185| 0.0002688| -0.0000075| -0.01073| -0.03924| 0.88329| 1.01465|  88| 0.0898| 0.726| 0.0003823| 0.0000131| -0.01073| -0.03928| 0.88329| 1.01464|  89| 0.1759| 1.185| 0.0002315| -0.0000065| -0.01069| -0.03928| 0.88332| 1.01464|  90| 0.0898| 0.726| 0.0003293| 0.0000113| -0.01070| -0.03931| 0.88332| 1.01464|  91| 0.1759| 1.185| 0.0001994| -0.0000056| -0.01066| -0.03932| 0.88334| 1.01463|  92| 0.0898| 0.726| 0.0002836| 0.0000097| -0.01067| -0.03934| 0.88334| 1.01463|  93| 0.1759| 1.185| 0.0001717| -0.0000048| -0.01064| -0.03934| 0.88335| 1.01463|  94| 0.0898| 0.726| 0.0002443| 0.0000084| -0.01064| -0.03937| 0.88336| 1.01462|  95| 0.1759| 1.185| 0.0001479| -0.0000041| -0.01061| -0.03937| 0.88337| 1.01462|  96| 0.0898| 0.726| 0.0002104| 0.0000072| -0.01062| -0.03939| 0.88337| 1.01462|  97| 0.1759| 1.185| 0.0001274| -0.0000036| -0.01059| -0.03939| 0.88338| 1.01462|  98| 0.0898| 0.726| 0.0001812| 0.0000062| -0.01059| -0.03941| 0.88338| 1.01462|  99| 0.1759| 1.185| 0.0001097| -0.0000031| -0.01058| -0.03941| 0.88339| 1.01461|  100| 0.0898| 0.726| 0.0001560| 0.0000054| -0.01058| -0.03942| 0.88340| 1.01461|  101| 0.1759| 1.185| 0.0000945| -0.0000026| -0.01056| -0.03942| 0.88340| 1.01461|  Метод ПВР - выбор оптимального w  w=0.10000 Itr=271  w=0.20000 Itr=132  w=0.30000 Itr=88  w=0.40000 Itr=67  w=0.50000 Itr=54  w=0.60000 Itr=45  w=0.70000 Itr=38  w=0.80000 Itr=33  w=0.90000 Itr=29  w=1.00000 Itr=26  w=1.10000 Itr=23  w=1.20000 Itr=20  w=1.30000 Itr=17  w=1.40000 Itr=14  w=1.50000 Itr=11  w=1.60000 Itr=14  w=1.70000 Itr=20  w=1.80000 Itr=29  w=1.90000 Itr=57  w\*= 1.50000 ItrMin=11  Метод ПВР  | Норма| Оценка|  Itr| Tau| q| невязки| погрешности| x[1]| x[2]| x[3]| x[4]|  1| 1.5000| 1.416| 2.8254962| -0.4156806| 0.19231| 0.10008| 0.84718| 1.41568|  2| 1.5000| 0.436| 3.4017038| 0.7980876| -0.42529| 0.35737| 0.65851| 0.94408|  3| 1.5000| 0.339| 0.6814885| 0.4083168| -0.32862| 0.20168| 0.72669| 1.15336|  4| 1.5000| 0.562| 0.5060497| 0.0915612| -0.22783| 0.14448| 0.82952| 1.03564|  5| 1.5000| 0.786| 0.3384612| 0.0251974| -0.15907| 0.05196| 0.82331| 1.05648|  6| 1.5000| 0.901| 0.4233589| 0.0091278| -0.07568| 0.01195| 0.87725| 1.02425|  7| 1.5000| 0.377| 0.1279315| 0.0519625| -0.05048| -0.01948| 0.86866| 1.02424|  8| 1.5000| 0.930| 0.1495110| 0.0022110| -0.02126| -0.03125| 0.88608| 1.01614|  9| 1.5000| 0.241| 0.0375800| 0.0221674| -0.01653| -0.03830| 0.88109| 1.01590|  10| 1.5000| 0.980| 0.0369579| 0.0001445| -0.00962| -0.03996| 0.88561| 1.01429|  11| 1.5000| 0.328| 0.0091892| 0.0046375| -0.01008| -0.04067| 0.88334| 1.01448|  12| 1.5000| 0.454| 0.0087823| 0.0012371| -0.00909| -0.04037| 0.88437| 1.01428|  13| 1.5000| 0.834| 0.0040009| 0.0001710| -0.00987| -0.04015| 0.88351| 1.01445|  14| 1.5000| 0.335| 0.0017418| 0.0005708| -0.00988| -0.03986| 0.88374| 1.01447|  15| 1.5000| 1.288| 0.0019016| -0.0000830| -0.01025| -0.03971| 0.88347| 1.01454|  16| 1.5000| 0.293| 0.0005357| 0.0002623| -0.01031| -0.03960| 0.88353| 1.01456|  17| 1.5000| 1.104| 0.0006146| -0.0000113| -0.01043| -0.03955| 0.88345| 1.01458|  18| 1.5000| 0.218| 0.0001528| 0.0000938| -0.01044| -0.03953| 0.88348| 1.01459|  19| 1.5000| 1.092| 0.0001497| -0.0000024| -0.01047| -0.03952| 0.88346| 1.01459|  20| 1.5000| 0.312| 0.0000369| 0.0000196| -0.01047| -0.03952| 0.88347| 1.01459|  Метод сопряженных градиентов  | Норма| Оценка|  Itr| Tau| q| невязки| погрешности| x[1]| x[2]| x[3]| x[4]|  1| 0.1399| 0.559| 7.1500000| 0.4405594| 0.13986| 0.27972| 0.41958| 0.55944| 1.0000000  2| 0.0791| 0.904| 2.7053400| 0.0539063| -0.13013| 0.08147| 0.76317| 1.06497| 1.5687609  3| 0.1569| 0.087| 0.2534751| 0.4615249| -0.13683| 0.08098| 0.80718| 1.03812| 1.0071047  4| 0.7539| 2.871| 0.0803783| -0.0823516| -0.01047| -0.03952| 0.88347| 1.01459| 2.1844368  5| 0.0615| 0.000| 0.0000000| 0.1263609| -0.01047| -0.03952| 0.88347| 1.01459| 1.0000000  Число обусловленности = 30.17389  Теоретическая оценка чиcла итераций  Метод простых итераций 138  Метод наискорейшего спуска 277  Метод ПВР 12  Метод сопряженных градиентов 27  LU разложение  -0.0104691 -0.0395210 0.8834653 1.0145914  Разница между LU разложением и методом простых итераций  0.0001687 -0.0001624 0.0001101 -0.0000345  Разница между LU разложением и методом наискорейшего спуска  0.0000913 -0.0000975 0.0000613 -0.0000201  Разница между LU разложением и методом ПВР  0.0000002 -0.0000035 -0.0000010 -0.0000003  Разница между LU разложением и методом сопряженных градиентов  -0.0000000 0.0000000 -0.0000000 -0.0000000 |

|  |
| --- |
| **Вариант 28b** |
| Variant = 28b  b:  1.0000000 2.0000000 3.0000000 4.0000000  A:  8.7000000 -2.3000000 -5.4000000 -1.6000000  -2.3000000 7.0000000 6.4000000 1.5000000  -5.4000000 6.4000000 9.2000000 0.3000000  -1.6000000 1.5000000 0.3000000 8.2000000  Норма матрицы = 18.3562781  Метод простой итерации  | Норма| Оценка|  Itr| Tau| q| невязки| погрешности| x[1]| x[2]| x[3]| x[4]|  1| 0.0981| 0.392| 10.1979323| 0.6077636| 0.09806| 0.19612| 0.29418| 0.39224|  2| 0.0981| 0.704| 6.2959939| 0.1162889| 0.37401| 0.03742| 0.24027| 0.44697|  3| 0.0981| 0.865| 4.6311302| 0.0371184| 0.35879| 0.07567| 0.47910| 0.52591|  4| 0.0981| 0.640| 2.9631096| 0.0858964| 0.50403| -0.07726| 0.46808| 0.52634|  5| 0.0981| 1.123| 2.7379495| -0.0188628| 0.48511| -0.08561| 0.63988| 0.57203|  6| 0.0981| 0.707| 1.9033447| 0.0503735| 0.57858| -0.20703| 0.65057| 0.57418|  7| 0.0981| 1.043| 1.6621238| -0.0052588| 0.57093| -0.23105| 0.77725| 0.60682|  8| 0.0981| 0.739| 1.1931048| 0.0331197| 0.63658| -0.32461| 0.79970| 0.61182|  9| 0.0981| 1.021| 1.0882631| -0.0019734| 0.63780| -0.35397| 0.89524| 0.63620|  10| 0.0981| 0.759| 0.8007906| 0.0230609| 0.68577| -0.42644| 0.92294| 0.64267|  11| 0.0981| 1.013| 0.7513180| -0.0009306| 0.69216| -0.45669| 0.99634| 0.66131|  12| 0.0981| 0.775| 0.5654538| 0.0165512| 0.72807| -0.51354| 1.02534| 0.66826|  13| 0.0981| 1.008| 0.5399870| -0.0004750| 0.73696| -0.54249| 1.08267| 0.68276|  14| 0.0981| 0.788| 0.4148342| 0.0121454| 0.76437| -0.58767| 1.11073| 0.68957|  15| 0.0981| 1.005| 0.4005907| -0.0002326| 0.77414| -0.61427| 1.15614| 0.70102|  16| 0.0981| 0.800| 0.3133754| 0.0090896| 0.79542| -0.65060| 1.18211| 0.70738|  17| 0.0981| 1.003| 0.3047991| -0.0000944| 0.80510| -0.67442| 1.21853| 0.71654|  18| 0.0981| 0.810| 0.2421699| 0.0069284| 0.82187| -0.70391| 1.24191| 0.72229|  19| 0.0981| 1.001| 0.2366745| -0.0000156| 0.83096| -0.72489| 1.27141| 0.72969|  20| 0.0981| 0.818| 0.1905007| 0.0053709| 0.84435| -0.74903| 1.29206| 0.73479|  21| 0.0981| 0.999| 0.1867973| 0.0000276| 0.85260| -0.76728| 1.31617| 0.74083|  22| 0.0981| 0.825| 0.1519667| 0.0042269| 0.86341| -0.78717| 1.33418| 0.74529|  23| 0.0981| 0.998| 0.1493695| 0.0000493| 0.87075| -0.80292| 1.35401| 0.75025|  24| 0.0981| 0.830| 0.1225734| 0.0033707| 0.87956| -0.81938| 1.36958| 0.75411|  25| 0.0981| 0.996| 0.1206944| 0.0000582| 0.88599| -0.83290| 1.38599| 0.75821|  26| 0.0981| 0.834| 0.0997331| 0.0027181| 0.89321| -0.84658| 1.39936| 0.76152|  27| 0.0981| 0.996| 0.0983403| 0.0000597| 0.89879| -0.85812| 1.41298| 0.76493|  28| 0.0981| 0.839| 0.0818208| 0.0021972| 0.90475| -0.86954| 1.42441| 0.76777|  29| 0.0981| 0.994| 0.0806607| 0.0000722| 0.90955| -0.87936| 1.43577| 0.77060|  30| 0.0981| 0.857| 0.0686729| 0.0016222| 0.91450| -0.88890| 1.44550| 0.77302|  31| 0.0981| 0.975| 0.0665111| 0.0002445| 0.91861| -0.89724| 1.45499| 0.77539|  32| 0.0981| 0.871| 0.0576307| 0.0012203| 0.92273| -0.90524| 1.46326| 0.77744|  33| 0.0981| 0.961| 0.0550761| 0.0003236| 0.92623| -0.91231| 1.47120| 0.77942|  34| 0.0981| 0.882| 0.0483685| 0.0009344| 0.92967| -0.91901| 1.47822| 0.78117|  35| 0.0981| 0.950| 0.0457614| 0.0003480| 0.93265| -0.92499| 1.48488| 0.78283|  36| 0.0981| 0.891| 0.0406038| 0.0007274| 0.93553| -0.93062| 1.49081| 0.78431|  37| 0.0981| 0.942| 0.0381251| 0.0003414| 0.93805| -0.93568| 1.49641| 0.78570|  38| 0.0981| 0.897| 0.0340955| 0.0005745| 0.94047| -0.94042| 1.50143| 0.78695|  39| 0.0981| 0.937| 0.0318320| 0.0003186| 0.94261| -0.94469| 1.50613| 0.78812|  40| 0.0981| 0.902| 0.0286398| 0.0004595| 0.94463| -0.94867| 1.51037| 0.78918|  41| 0.0981| 0.932| 0.0266240| 0.0002882| 0.94644| -0.95228| 1.51433| 0.79016|  42| 0.0981| 0.906| 0.0240651| 0.0003715| 0.94814| -0.95563| 1.51791| 0.79105|  43| 0.0981| 0.929| 0.0222994| 0.0002555| 0.94967| -0.95867| 1.52123| 0.79188|  44| 0.0981| 0.909| 0.0202278| 0.0003030| 0.95110| -0.96149| 1.52426| 0.79264|  45| 0.0981| 0.926| 0.0186985| 0.0002235| 0.95239| -0.96406| 1.52706| 0.79333|  46| 0.0981| 0.911| 0.0170076| 0.0002489| 0.95360| -0.96644| 1.52961| 0.79397|  47| 0.0981| 0.924| 0.0156934| 0.0001935| 0.95469| -0.96861| 1.53197| 0.79456|  48| 0.0981| 0.913| 0.0143042| 0.0002057| 0.95570| -0.97060| 1.53412| 0.79509|  49| 0.0981| 0.923| 0.0131811| 0.0001664| 0.95662| -0.97243| 1.53610| 0.79559|  50| 0.0981| 0.914| 0.0120337| 0.0001707| 0.95747| -0.97412| 1.53792| 0.79604|  51| 0.0981| 0.922| 0.0110778| 0.0001424| 0.95825| -0.97566| 1.53959| 0.79645|  52| 0.0981| 0.915| 0.0101260| 0.0001422| 0.95897| -0.97708| 1.54112| 0.79684|  53| 0.0981| 0.921| 0.0093146| 0.0001213| 0.95962| -0.97838| 1.54253| 0.79719|  54| 0.0981| 0.916| 0.0085225| 0.0001187| 0.96023| -0.97957| 1.54382| 0.79751|  55| 0.0981| 0.920| 0.0078353| 0.0001031| 0.96078| -0.98067| 1.54501| 0.79780|  56| 0.0981| 0.916| 0.0071743| 0.0000994| 0.96129| -0.98168| 1.54609| 0.79807|  57| 0.0981| 0.920| 0.0065931| 0.0000875| 0.96176| -0.98260| 1.54709| 0.79832|  58| 0.0981| 0.917| 0.0060404| 0.0000833| 0.96218| -0.98345| 1.54801| 0.79855|  59| 0.0981| 0.919| 0.0055494| 0.0000741| 0.96258| -0.98423| 1.54885| 0.79876|  60| 0.0981| 0.917| 0.0050864| 0.0000699| 0.96294| -0.98494| 1.54963| 0.79895|  61| 0.0981| 0.919| 0.0046719| 0.0000627| 0.96327| -0.98560| 1.55034| 0.79913|  62| 0.0981| 0.917| 0.0042836| 0.0000588| 0.96357| -0.98620| 1.55099| 0.79929|  63| 0.0981| 0.919| 0.0039339| 0.0000530| 0.96385| -0.98675| 1.55159| 0.79944|  64| 0.0981| 0.917| 0.0036079| 0.0000494| 0.96411| -0.98726| 1.55214| 0.79958|  65| 0.0981| 0.919| 0.0033130| 0.0000447| 0.96435| -0.98773| 1.55264| 0.79970|  66| 0.0981| 0.918| 0.0030391| 0.0000416| 0.96456| -0.98816| 1.55310| 0.79982|  67| 0.0981| 0.918| 0.0027905| 0.0000378| 0.96476| -0.98855| 1.55353| 0.79993|  68| 0.0981| 0.918| 0.0025602| 0.0000350| 0.96494| -0.98891| 1.55392| 0.80002|  69| 0.0981| 0.918| 0.0023506| 0.0000319| 0.96511| -0.98924| 1.55428| 0.80011|  70| 0.0981| 0.918| 0.0021569| 0.0000295| 0.96526| -0.98955| 1.55461| 0.80019|  71| 0.0981| 0.918| 0.0019802| 0.0000269| 0.96540| -0.98982| 1.55491| 0.80027|  72| 0.0981| 0.918| 0.0018172| 0.0000248| 0.96553| -0.99008| 1.55518| 0.80034|  73| 0.0981| 0.918| 0.0016683| 0.0000227| 0.96565| -0.99032| 1.55544| 0.80040|  74| 0.0981| 0.918| 0.0015311| 0.0000209| 0.96576| -0.99053| 1.55567| 0.80046|  75| 0.0981| 0.918| 0.0014056| 0.0000191| 0.96586| -0.99073| 1.55589| 0.80051|  76| 0.0981| 0.918| 0.0012901| 0.0000176| 0.96595| -0.99091| 1.55608| 0.80056|  77| 0.0981| 0.918| 0.0011844| 0.0000161| 0.96604| -0.99108| 1.55627| 0.80061|  78| 0.0981| 0.918| 0.0010871| 0.0000148| 0.96612| -0.99123| 1.55643| 0.80065|  79| 0.0981| 0.918| 0.0009980| 0.0000136| 0.96619| -0.99137| 1.55658| 0.80069|  80| 0.0981| 0.918| 0.0009161| 0.0000125| 0.96625| -0.99150| 1.55672| 0.80072|  81| 0.0981| 0.918| 0.0008410| 0.0000115| 0.96631| -0.99162| 1.55685| 0.80075|  82| 0.0981| 0.918| 0.0007719| 0.0000105| 0.96637| -0.99173| 1.55697| 0.80078|  83| 0.0981| 0.918| 0.0007087| 0.0000097| 0.96642| -0.99183| 1.55708| 0.80081|  84| 0.0981| 0.918| 0.0006505| 0.0000089| 0.96646| -0.99192| 1.55718| 0.80083|  85| 0.0981| 0.918| 0.0005972| 0.0000081| 0.96651| -0.99201| 1.55727| 0.80086|  86| 0.0981| 0.918| 0.0005482| 0.0000075| 0.96655| -0.99208| 1.55735| 0.80088|  87| 0.0981| 0.918| 0.0005033| 0.0000069| 0.96658| -0.99216| 1.55743| 0.80090|  88| 0.0981| 0.918| 0.0004620| 0.0000063| 0.96661| -0.99222| 1.55750| 0.80091|  89| 0.0981| 0.918| 0.0004241| 0.0000058| 0.96664| -0.99228| 1.55756| 0.80093|  90| 0.0981| 0.918| 0.0003893| 0.0000053| 0.96667| -0.99234| 1.55762| 0.80095|  91| 0.0981| 0.918| 0.0003574| 0.0000049| 0.96670| -0.99239| 1.55768| 0.80096|  92| 0.0981| 0.918| 0.0003281| 0.0000045| 0.96672| -0.99243| 1.55773| 0.80097|  93| 0.0981| 0.918| 0.0003012| 0.0000041| 0.96674| -0.99247| 1.55777| 0.80098|  94| 0.0981| 0.918| 0.0002765| 0.0000038| 0.96676| -0.99251| 1.55782| 0.80099|  95| 0.0981| 0.918| 0.0002539| 0.0000035| 0.96678| -0.99255| 1.55786| 0.80100|  96| 0.0981| 0.918| 0.0002330| 0.0000032| 0.96680| -0.99258| 1.55789| 0.80101|  97| 0.0981| 0.918| 0.0002139| 0.0000029| 0.96681| -0.99261| 1.55792| 0.80102|  98| 0.0981| 0.918| 0.0001964| 0.0000027| 0.96683| -0.99264| 1.55795| 0.80103|  99| 0.0981| 0.918| 0.0001803| 0.0000025| 0.96684| -0.99267| 1.55798| 0.80103|  100| 0.0981| 0.918| 0.0001655| 0.0000023| 0.96685| -0.99269| 1.55801| 0.80104|  101| 0.0981| 0.918| 0.0001520| 0.0000021| 0.96686| -0.99271| 1.55803| 0.80105|  102| 0.0981| 0.918| 0.0001395| 0.0000019| 0.96687| -0.99273| 1.55805| 0.80105|  103| 0.0981| 0.918| 0.0001281| 0.0000017| 0.96688| -0.99275| 1.55807| 0.80106|  104| 0.0981| 0.918| 0.0001176| 0.0000016| 0.96689| -0.99277| 1.55809| 0.80106|  105| 0.0981| 0.918| 0.0001079| 0.0000015| 0.96690| -0.99278| 1.55810| 0.80107|  106| 0.0981| 0.918| 0.0000991| 0.0000014| 0.96690| -0.99279| 1.55812| 0.80107|  Метод наискорейшего спуска  | Норма| Оценка|  Itr| Tau| q| невязки| погрешности| x[1]| x[2]| x[3]| x[4]|  1| 0.0986| 0.394| 10.1433333| 0.6056523| 0.09859| 0.19717| 0.29576| 0.39435|  2| 0.0827| 0.592| 6.4331782| 0.1609208| 0.33201| 0.06178| 0.24874| 0.43895|  3| 0.1224| 1.037| 3.7130305| -0.0086572| 0.36863| 0.07167| 0.49082| 0.53255|  4| 0.0890| 0.586| 3.0091006| 0.1001809| 0.49863| -0.07023| 0.47805| 0.52971|  5| 0.1251| 1.397| 2.3438519| -0.0563180| 0.48980| -0.09720| 0.67627| 0.58175|  6| 0.0893| 0.628| 2.0968503| 0.0737884| 0.58781| -0.22163| 0.66474| 0.57784|  7| 0.1251| 1.311| 1.5747397| -0.0387024| 0.57410| -0.24887| 0.82787| 0.61979|  8| 0.0893| 0.638| 1.4246757| 0.0590776| 0.65401| -0.35293| 0.81810| 0.61633|  9| 0.1251| 1.298| 1.1320632| -0.0309639| 0.64178| -0.37615| 0.95312| 0.65093|  10| 0.0893| 0.640| 1.0234990| 0.0486701| 0.70782| -0.46251| 0.94500| 0.64803|  11| 0.1251| 1.295| 0.8458034| -0.0255070| 0.69756| -0.48184| 1.05685| 0.67668|  12| 0.0893| 0.640| 0.7633635| 0.0402900| 0.75226| -0.55341| 1.05012| 0.67427|  13| 0.1251| 1.295| 0.6481074| -0.0211149| 0.74375| -0.56944| 1.14280| 0.69801|  14| 0.0893| 0.640| 0.5841021| 0.0333792| 0.78907| -0.62874| 1.13722| 0.69601|  15| 0.1251| 1.295| 0.5055187| -0.0174931| 0.78201| -0.64203| 1.21401| 0.71568|  16| 0.0893| 0.640| 0.4551088| 0.0276574| 0.81956| -0.69117| 1.20939| 0.71403|  17| 0.1251| 1.295| 0.3994575| -0.0144945| 0.81371| -0.70217| 1.27302| 0.73032|  18| 0.0893| 0.640| 0.3593362| 0.0229169| 0.84483| -0.74289| 1.26919| 0.72895|  19| 0.1251| 1.295| 0.3187487| -0.0120101| 0.83998| -0.75201| 1.32192| 0.74245|  20| 0.0893| 0.640| 0.2865590| 0.0189890| 0.86577| -0.78575| 1.31874| 0.74132|  21| 0.1251| 1.295| 0.2562619| -0.0099516| 0.86175| -0.79331| 1.36243| 0.75251|  22| 0.0893| 0.640| 0.2302739| 0.0157343| 0.88312| -0.82127| 1.35980| 0.75157|  23| 0.1251| 1.295| 0.2072329| -0.0082459| 0.87979| -0.82753| 1.39600| 0.76084|  24| 0.0893| 0.640| 0.1861481| 0.0130375| 0.89749| -0.85069| 1.39382| 0.76006|  25| 0.1251| 1.295| 0.1683590| -0.0068326| 0.89473| -0.85588| 1.42382| 0.76774|  26| 0.0893| 0.640| 0.1511850| 0.0108029| 0.90940| -0.87507| 1.42201| 0.76710|  27| 0.1251| 1.295| 0.1372803| -0.0056615| 0.90712| -0.87937| 1.44687| 0.77346|  28| 0.0893| 0.640| 0.1232477| 0.0089513| 0.91927| -0.89528| 1.44537| 0.77293|  29| 0.1251| 1.295| 0.1122688| -0.0046911| 0.91738| -0.89884| 1.46596| 0.77820|  30| 0.0893| 0.640| 0.1007738| 0.0074170| 0.92745| -0.91202| 1.46472| 0.77776|  31| 0.1251| 1.295| 0.0920327| -0.0038871| 0.92588| -0.91497| 1.48179| 0.78213|  32| 0.0893| 0.640| 0.0825970| 0.0061458| 0.93422| -0.92589| 1.48076| 0.78176|  33| 0.1251| 1.295| 0.0755896| -0.0032208| 0.93292| -0.92834| 1.49490| 0.78538|  34| 0.0893| 0.640| 0.0678313| 0.0050924| 0.93984| -0.93739| 1.49405| 0.78508|  35| 0.1251| 1.295| 0.0621817| -0.0026688| 0.93876| -0.93941| 1.50576| 0.78808|  36| 0.0893| 0.640| 0.0557940| 0.0042196| 0.94449| -0.94691| 1.50506| 0.78782|  37| 0.1251| 1.295| 0.0512177| -0.0022114| 0.94360| -0.94859| 1.51477| 0.79031|  38| 0.0893| 0.640| 0.0459524| 0.0034963| 0.94835| -0.95480| 1.51418| 0.79010|  39| 0.1251| 1.295| 0.0422311| -0.0018323| 0.94761| -0.95619| 1.52223| 0.79216|  40| 0.0893| 0.640| 0.0378871| 0.0028971| 0.95154| -0.96134| 1.52174| 0.79199|  41| 0.1251| 1.295| 0.0348512| -0.0015183| 0.95093| -0.96249| 1.52841| 0.79370|  42| 0.0893| 0.640| 0.0312646| 0.0024005| 0.95419| -0.96676| 1.52801| 0.79355|  43| 0.1251| 1.295| 0.0287814| -0.0012580| 0.95368| -0.96771| 1.53353| 0.79497|  44| 0.0893| 0.640| 0.0258182| 0.0019891| 0.95638| -0.97125| 1.53320| 0.79485|  45| 0.1251| 1.295| 0.0237825| -0.0010424| 0.95596| -0.97204| 1.53777| 0.79602|  46| 0.0893| 0.640| 0.0213332| 0.0016482| 0.95820| -0.97497| 1.53750| 0.79592|  47| 0.1251| 1.295| 0.0196612| -0.0008638| 0.95785| -0.97562| 1.54129| 0.79689|  48| 0.0893| 0.640| 0.0176358| 0.0013657| 0.95970| -0.97805| 1.54106| 0.79681|  49| 0.1251| 1.295| 0.0162606| -0.0007157| 0.95942| -0.97859| 1.54420| 0.79762|  50| 0.0893| 0.640| 0.0145851| 0.0011316| 0.96095| -0.98060| 1.54401| 0.79755|  51| 0.1251| 1.295| 0.0134525| -0.0005930| 0.96071| -0.98105| 1.54662| 0.79822|  52| 0.0893| 0.640| 0.0120662| 0.0009376| 0.96199| -0.98272| 1.54646| 0.79816|  53| 0.1251| 1.295| 0.0111324| -0.0004914| 0.96179| -0.98309| 1.54862| 0.79871|  54| 0.0893| 0.640| 0.0099850| 0.0007769| 0.96284| -0.98447| 1.54849| 0.79867|  55| 0.1251| 1.295| 0.0092145| -0.0004072| 0.96268| -0.98478| 1.55028| 0.79912|  56| 0.0893| 0.640| 0.0082646| 0.0006438| 0.96355| -0.98593| 1.55017| 0.79908|  57| 0.1251| 1.295| 0.0076284| -0.0003374| 0.96342| -0.98618| 1.55165| 0.79946|  58| 0.0893| 0.640| 0.0068419| 0.0005334| 0.96414| -0.98713| 1.55156| 0.79943|  59| 0.1251| 1.295| 0.0063162| -0.0002796| 0.96403| -0.98734| 1.55279| 0.79975|  60| 0.0893| 0.640| 0.0056650| 0.0004420| 0.96463| -0.98813| 1.55271| 0.79972|  61| 0.1251| 1.295| 0.0052305| -0.0002316| 0.96453| -0.98830| 1.55373| 0.79998|  62| 0.0893| 0.640| 0.0046911| 0.0003662| 0.96503| -0.98896| 1.55367| 0.79996|  63| 0.1251| 1.295| 0.0043318| -0.0001919| 0.96495| -0.98910| 1.55451| 0.80017|  64| 0.0893| 0.640| 0.0038851| 0.0003035| 0.96537| -0.98964| 1.55446| 0.80016|  65| 0.1251| 1.295| 0.0035878| -0.0001590| 0.96530| -0.98976| 1.55516| 0.80033|  66| 0.0893| 0.640| 0.0032179| 0.0002515| 0.96564| -0.99021| 1.55512| 0.80032|  67| 0.1251| 1.295| 0.0029719| -0.0001318| 0.96559| -0.99031| 1.55570| 0.80047|  68| 0.0893| 0.640| 0.0026654| 0.0002084| 0.96587| -0.99068| 1.55566| 0.80046|  69| 0.1251| 1.295| 0.0024618| -0.0001092| 0.96583| -0.99076| 1.55614| 0.80058|  70| 0.0893| 0.640| 0.0022079| 0.0001726| 0.96606| -0.99107| 1.55611| 0.80057|  71| 0.1251| 1.295| 0.0020394| -0.0000905| 0.96603| -0.99114| 1.55651| 0.80067|  72| 0.0893| 0.640| 0.0018290| 0.0001431| 0.96622| -0.99139| 1.55648| 0.80066|  73| 0.1251| 1.295| 0.0016895| -0.0000750| 0.96619| -0.99145| 1.55681| 0.80075|  74| 0.0893| 0.640| 0.0015152| 0.0001185| 0.96635| -0.99166| 1.55679| 0.80074|  75| 0.1251| 1.295| 0.0013997| -0.0000621| 0.96633| -0.99171| 1.55707| 0.80081|  76| 0.0893| 0.640| 0.0012553| 0.0000982| 0.96646| -0.99188| 1.55705| 0.80080|  77| 0.1251| 1.295| 0.0011596| -0.0000515| 0.96644| -0.99192| 1.55728| 0.80086|  78| 0.0893| 0.640| 0.0010400| 0.0000814| 0.96655| -0.99206| 1.55726| 0.80086|  79| 0.1251| 1.295| 0.0009608| -0.0000427| 0.96653| -0.99210| 1.55745| 0.80090|  80| 0.0893| 0.640| 0.0008617| 0.0000674| 0.96662| -0.99222| 1.55744| 0.80090|  81| 0.1251| 1.295| 0.0007960| -0.0000353| 0.96661| -0.99224| 1.55759| 0.80094|  82| 0.0893| 0.640| 0.0007139| 0.0000559| 0.96669| -0.99234| 1.55758| 0.80094|  83| 0.1251| 1.295| 0.0006595| -0.0000293| 0.96667| -0.99236| 1.55771| 0.80097|  84| 0.0893| 0.640| 0.0005915| 0.0000463| 0.96674| -0.99245| 1.55771| 0.80097|  85| 0.1251| 1.295| 0.0005465| -0.0000243| 0.96673| -0.99246| 1.55781| 0.80099|  86| 0.0893| 0.640| 0.0004901| 0.0000384| 0.96678| -0.99253| 1.55781| 0.80099|  87| 0.1251| 1.295| 0.0004528| -0.0000201| 0.96677| -0.99255| 1.55789| 0.80101|  88| 0.0893| 0.640| 0.0004061| 0.0000318| 0.96681| -0.99260| 1.55789| 0.80101|  89| 0.1251| 1.295| 0.0003751| -0.0000167| 0.96681| -0.99262| 1.55796| 0.80103|  90| 0.0893| 0.640| 0.0003365| 0.0000263| 0.96684| -0.99266| 1.55796| 0.80103|  91| 0.1251| 1.295| 0.0003108| -0.0000138| 0.96684| -0.99267| 1.55802| 0.80104|  92| 0.0893| 0.640| 0.0002788| 0.0000218| 0.96687| -0.99271| 1.55801| 0.80104|  93| 0.1251| 1.295| 0.0002576| -0.0000114| 0.96686| -0.99272| 1.55806| 0.80106|  94| 0.0893| 0.640| 0.0002310| 0.0000181| 0.96689| -0.99275| 1.55806| 0.80105|  95| 0.1251| 1.295| 0.0002134| -0.0000095| 0.96688| -0.99276| 1.55810| 0.80107|  96| 0.0893| 0.640| 0.0001914| 0.0000150| 0.96690| -0.99279| 1.55810| 0.80106|  97| 0.1251| 1.295| 0.0001768| -0.0000079| 0.96690| -0.99279| 1.55814| 0.80107|  98| 0.0893| 0.640| 0.0001586| 0.0000124| 0.96692| -0.99282| 1.55813| 0.80107|  99| 0.1251| 1.295| 0.0001465| -0.0000065| 0.96691| -0.99282| 1.55816| 0.80108|  100| 0.0893| 0.640| 0.0001314| 0.0000103| 0.96693| -0.99284| 1.55816| 0.80108|  101| 0.1251| 1.295| 0.0001214| -0.0000054| 0.96693| -0.99284| 1.55818| 0.80109|  102| 0.0893| 0.640| 0.0001089| 0.0000085| 0.96694| -0.99286| 1.55818| 0.80108|  103| 0.1251| 1.295| 0.0001006| -0.0000045| 0.96694| -0.99286| 1.55820| 0.80109|  104| 0.0893| 0.640| 0.0000902| 0.0000071| 0.96695| -0.99287| 1.55820| 0.80109|  Метод ПВР - выбор оптимального w  w=0.10000 Itr=421  w=0.20000 Itr=198  w=0.30000 Itr=124  w=0.40000 Itr=89  w=0.50000 Itr=69  w=0.60000 Itr=55  w=0.70000 Itr=44  w=0.80000 Itr=37  w=0.90000 Itr=31  w=1.00000 Itr=26  w=1.10000 Itr=21  w=1.20000 Itr=17  w=1.30000 Itr=14  w=1.40000 Itr=9  w=1.50000 Itr=12  w=1.60000 Itr=15  w=1.70000 Itr=22  w=1.80000 Itr=29  w=1.90000 Itr=64  w\*= 1.40000 ItrMin=9  Метод ПВР  | Норма| Оценка|  Itr| Tau| q| невязки| погрешности| x[1]| x[2]| x[3]| x[4]|  1| 1.4000| 0.599| 6.6780273| 0.4010207| 0.16092| 0.47402| 0.12710| 0.59898|  2| 1.4000| 0.969| 3.7598847| 0.0186239| 0.53666| 0.11487| 0.70745| 0.52428|  3| 1.4000| 0.834| 2.5843929| 0.0964226| 0.73851| -0.36906| 1.11590| 0.71231|  4| 1.4000| 0.661| 1.4042482| 0.1642872| 0.88200| -0.68870| 1.37316| 0.74498|  5| 1.4000| 0.580| 0.7062052| 0.1342963| 0.93826| -0.87405| 1.49550| 0.78848|  6| 1.4000| 0.450| 0.3086188| 0.1019839| 0.96467| -0.95742| 1.54747| 0.79699|  7| 1.4000| 0.396| 0.1142693| 0.0503656| 0.97060| -0.99042| 1.56331| 0.80284|  8| 1.4000| 0.258| 0.0281906| 0.0244873| 0.97128| -0.99893| 1.56556| 0.80275|  9| 1.4000| 0.234| 0.0059306| 0.0065211| 0.96979| -0.99907| 1.56357| 0.80252|  10| 1.4000| 1.135| 0.0085605| -0.0002693| 0.96855| -0.99696| 1.56131| 0.80185|  11| 1.4000| 0.822| 0.0072255| 0.0004032| 0.96768| -0.99511| 1.55972| 0.80149|  12| 1.4000| 0.641| 0.0044621| 0.0006672| 0.96725| -0.99392| 1.55886| 0.80125|  13| 1.4000| 0.516| 0.0022641| 0.0005767| 0.96705| -0.99330| 1.55845| 0.80115|  14| 1.4000| 0.438| 0.0009648| 0.0003448| 0.96698| -0.99303| 1.55830| 0.80111|  15| 1.4000| 0.345| 0.0003210| 0.0001763| 0.96696| -0.99294| 1.55826| 0.80110|  16| 1.4000| 0.206| 0.0000702| 0.0000736| 0.96697| -0.99292| 1.55826| 0.80110|  Метод сопряженных градиентов  | Норма| Оценка|  Itr| Tau| q| невязки| погрешности| x[1]| x[2]| x[3]| x[4]|  1| 0.0986| 0.394| 10.1433333| 0.6056523| 0.09859| 0.19717| 0.29576| 0.39435| 1.0000000  2| 0.0827| 0.979| 4.4065457| 0.0082229| 0.48471| 0.09020| 0.36314| 0.64083| 1.4599141  3| 0.2211| 2.177| 1.2517967| -0.4546500| 0.96227| -0.44728| 1.20391| 0.59046| 2.2670001  4| 0.1578| 0.649| 0.7999305| 0.2951056| 0.96698| -0.99295| 1.55829| 0.80111| 1.4752789  5| 0.0615| 0.000| 0.0000000| 0.5456693| 0.96698| -0.99295| 1.55829| 0.80111| 1.0000000  Число обусловленности = 21.96338  Теоретическая оценка числа итераций  Метод простых итераций 101  Метод наискорейшего спуска 202  Метод ПВР 10  Метод сопряженных градиентов 23  LU разложение  0.9669832 -0.9929513 1.5582896 0.8011114  Разница между LU разложением и методом простых итераций  0.0000792 -0.0001568 0.0001696 0.0000422  Разница между LU разложением и методом наискорейшего спуска  0.0000377 -0.0000771 0.0000891 0.0000222  Разница между LU разложением и методом ПВР  0.0000165 -0.0000291 0.0000295 0.0000075  Разница между LU разложением и методом сопряженных градиентов  -0.0000000 0.0000000 -0.0000000 -0.0000000 |

**Краткие выводы**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Метод | Реальное кол-во итераций | Теоретическая оценка числа итераций | X[1] | X[2] | X[3] | X[4] |
| Вариант 27b | | | | | | |
| Простых итераций | 112 | 138 | -0.01064 | -0.03936 | 0.88336 | 1.01463 |
| Наискорейшего спуска | 101 | 277 | -0.01056 | -0.03942 | 0.88340 | 1.01461 |
| ПВР | 20 | 12 | -0.01047 | -0.03952 | 0.88347 | 1.01459 |
| Сопряженных градиентов | 5 | 27 | -0.01047 | -0.03952 | 0.88347 | 1.01459 |
| LU разложения | - | - | -0.0104691 | -0.0395210 | 0.8834653 | 1.0145914 |
| Вариант 28b | | | | | | |
| Простых итераций | 106 | 101 | 0.96690 | 0.99279 | 1.55812 | 0.80107 |
| Наискорейшего спуска | 104 | 202 | 0.96695 | 0.99287 | 1.55820 | 0.80109 |
| ПВР | 16 | 10 | 0.96697 | -0.99292 | 1.55826 | 0.80110 |
| Сопряженных градиентов | 5 | 23 | 0.96698 | -0.99295 | 1.55829 | 0.80111 |
| LU разложения | - | - | 0.9669832 | -0.9929513 | 1.5582896 | 0.8011114 |

Как видим, в обоих случаях наиболее точный ответ по сравнению с прямым методом LU разложения даёт метод сопряженных градиентов, далее – ПВР-метод и метод наискорейшего спуска. Наименее точным является метод простых итераций. Можно заметить, что при малых значениях n метод сопряженный градиентов дает результат приблизительно за n шагов, то есть работает как прямой метод.

**Текст программы**

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <iomanip>

#include <math.h>

#include <time.h>

**using** **namespace** std;

**const** **int** N = 4;

**const** **double** gamma = 0.9;

**const** **double** epsilon = 0.0001;

**const** **char**\* fileInput = "input2.txt"; // "testin.txt"; //

**const** **char**\* fileOutput = "output2.txt";//"testout.txt"; //

ofstream output;

**double** condA = 0;

**double** normA = 0;

#**pragma region ввод - вывод**

**double**\*\* readMatrix() //считывание исходной матрицы

{

ifstream file;

file.open(fileInput, ios::in);

**double**\*\* m = **new** **double**\* [N];

**for** (**int** i = 0; i < N; i++)

m[i] = **new** **double**[N];

**for** (**int** i = 0; i < N; i++)

**for** (**int** j = 0; j < N; j++)

file >> m[i][j];

file.close();

**return** m;

}

**void** printMatrix(**double**\*\* m, **bool** exp) // печать матрицы в файл

{

**for** (**int** i = 0; i < N; i++)

{

**for** (**int** j = 0; j < N; j++)

**if** (exp) output << setw(20) << scientific << m[i][j];

**else** output << setw(12) << setprecision(7) << fixed << m[i][j];

output << endl;

}

}

**void** printVector(**double**\* v, **bool** exp) // печать вектора в файл

{

**for** (**int** j = 0; j < N; j++)

**if** (exp) output << setw(20) << scientific << v[j];

**else** output << setw(12) << setprecision(7) << fixed << v[j];

output << endl;

}

**void** printHead()

{

output << setw(8) << " " << setw(8) << " " << setw(7) << " " << "|" << setw(12) << "Норма" << "|" << setw(12) << "Оценка" << "|\n";

output << setw(7) <<"Itr"<< "|" << setw(7) << "Tau" << "|" << setw(7) << "q" << "|" << setw(12) << "невязки" << "|" << setw(12) << "погрешности" << "|";

output << setw(12) << "x[1]" << "|" << setw(12) << "x[2]" << "|" << setw(12) << "x[3]" << "|" << setw(12) << "x[4]" << "|\n";

}

**void** printRow(**int** itr, **double** tau, **double** q, **double** norm, **double** err, **double**\* v, **bool** isAlpha, **double** alpha)

{

output << setw(7) << itr << "|";

output << setw(7) << setprecision(4) << fixed << tau << "|";

output << setw(7) << setprecision(3) << fixed << q << "|";

output << setw(12) << setprecision(7) << fixed << norm << "|";

output << setw(12) << setprecision(7) << fixed << err << "|";

**for** (**int** i = 0; i < N; i++)

output << setw(12) << setprecision(5) << fixed << v[i] << "|";

**if** (isAlpha)

output << setw(12) << setprecision(7) << fixed << alpha;

output << endl;

}

**double**\*\* createAndCopyMatrix(**double**\*\* A) //создание аналогичной матрицы

{

**double**\*\* B = **new** **double**\* [N];

**for** (**int** i = 0; i < N; i++)

B[i] = **new** **double**[N];

**for** (**int** i = 0; i < N; i++)

**for** (**int** j = 0; j < N; j++)

B[i][j] = A[i][j];

**return** B;

}

**void** deleteMatrix(**double**\*\* m) //удаление матрицы, очищение памяти

{

**for** (**int** i = 0; i < N; i++)

**delete**[] m[i];

**delete**[] m;

}

#**pragma endregion**

#**pragma region умножение-вычтание**

**double**\*\* multMatrix(**double**\*\* A, **double**\*\* B) // перемножение двух матриц

{

**double**\*\* res = **new** **double**\* [N];

**for** (**int** i = 0; i < N; i++)

res[i] = **new** **double**[N];

**for** (**int** i = 0; i < N; i++)

**for** (**int** j = 0; j < N; j++)

{

res[i][j] = 0;

**for** (**int** k = 0; k < N; k++)

res[i][j] += A[i][k] \* B[k][j];

}

**return** res;

}

**double**\* multMatrVec(**double**\*\* A, **double**\* b) // перемножения матрицы и вектора

{

**double**\* v = **new** **double**[N];

**for** (**int** i = 0; i < N; i++)

{

v[i] = 0;

**for** (**int** k = 0; k < N; k++)

v[i] += A[i][k] \* b[k];

}

**return** v;

}

**double**\*\* subMatrices(**double**\*\* a, **double**\*\* b)

{

**double**\*\* res = **new** **double**\* [N];

**for** (**int** i = 0; i < N; i++)

res[i] = **new** **double**[N];

**for** (**int** i = 0; i < N; i++)

{

**for** (**int** j = 0; j < N; j++)

{

res[i][j] = a[i][j] - b[i][j];

}

}

**return** res;

}

**double**\* subVec(**double**\* a, **double**\* b)

{

**double**\* v = **new** **double**[N];

**for** (**int** i = 0; i < N; i++)

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

**return** v;

}

// скалярное произведение векторов

**double** scalarMult(**double**\* a, **double**\* b)

{

**double** sum = 0;

**for** (**int** i = 0; i < N; i++)

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

**return** sum;

}

#**pragma endregion**

**void** transposeMatrix(**double**\*\* m) // транспонированная матрица

{

**for** (**int** i = 0; i < N - 1; i++)

**for** (**int** j = i + 1; j < N; j++)

swap(m[i][j], m[j][i]);

}

**double** EuclideanNorm(**double**\*\* A) //Евклидова норма

{

**double** max = 0, elem\_max = 0, fi;

**int** k = 0, i\_max, j\_max;

**bool** f = **true**;

**double**\*\* At = createAndCopyMatrix(A);

transposeMatrix(At);

**double**\*\* AtA = multMatrix(At, A);

**double**\*\* H = **new** **double**\* [N];

**double**\*\* Ht = **new** **double**\* [N];;

**for** (**int** i = 0; i < N; i++)

{

H[i] = **new** **double**[N];

Ht[i] = **new** **double**[N];

}

**while** (f)

{

elem\_max = 0;

**for** (**int** i = 0; i < N - 1; i++)

**for** (**int** j = i + 1; j < N; j++)

**if** (abs(AtA[i][j]) > elem\_max)

{

elem\_max = abs(AtA[i][j]);

i\_max = i;

j\_max = j;

}

f = abs(elem\_max) > 0.0001;

**if** (f)

{

**double** temp = 2 \* AtA[i\_max][j\_max] / (AtA[i\_max][i\_max] - AtA[j\_max][j\_max]);

fi = 0.5 \* atan(temp);

**for** (**int** i = 0; i < N; i++)

**for** (**int** j = 0; j < N; j++)

H[i][j] = i == j ? 1 : 0;

H[i\_max][i\_max] = cos(fi);

H[j\_max][j\_max] = cos(fi);

H[i\_max][j\_max] = -sin(fi);

H[j\_max][i\_max] = sin(fi);

deleteMatrix(Ht);

Ht = createAndCopyMatrix(H);

transposeMatrix(Ht);

**double**\*\* AtA2 = multMatrix(Ht, AtA);

**double**\*\* AtA3 = multMatrix(AtA2, H);

**for** (**int** i = 0; i < N; i++)

**for** (**int** j = 0; j < N; j++)

AtA[i][j] = AtA3[i][j];

deleteMatrix(AtA2);

deleteMatrix(AtA3);

}

}

**for** (**int** i = 0; i < N; i++)

**if** (abs(AtA[i][i]) > max)

max = abs(AtA[i][i]);

deleteMatrix(At);

deleteMatrix(AtA);

deleteMatrix(H);

deleteMatrix(Ht);

**return** sqrt(max);

}

**double** EuclidianVector(**double**\* b)

{

**double** res = 0;

**for** (**int** i = 0; i < N; i++) {

res += b[i] \* b[i];

}

**return** sqrt(res);

}

**double** normVector(**double**\* v)

{

**double** max = abs(v[0]);

**for** (**int** i = 1; i < N; i++)

**if** (abs(v[i]) > max)

max = abs(v[i]);

**return** max;

}

**double**\* solveSystem(**double**\*\* U, **double**\*\* L, **double**\* b)

{

**double** y[N];

**for** (**int** i = 0; i < N; i++) {

y[i] = b[i];

**for** (**int** j = 0; j < i; j++) {

y[i] -= y[j] \* L[i][j];

}

y[i] /= L[i][i];

}

**double**\* x\_new = **new** **double**[N];

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

x\_new[i] = y[i];

**for** (**int** j = i + 1; j < N; j++) {

x\_new[i] -= U[i][j] \* x\_new[j];

}

}

**return** x\_new;

}

**double**\*\* inverseMatrix(**double**\*\* U, **double**\*\* L, **double**\*\* P) //обратная матрица

{

**double**\*\* A\_inverse = **new** **double**\* [N];

**for** (**int** i = 0; i < N; i++)

A\_inverse[i] = **new** **double**[N];

**double**\* e = **new** **double**[N];

**double**\* Pe;

**for** (**int** k = 0; k < N; k++)

{

**for** (**int** i = 0; i < N; i++)

e[i] = 0;

e[k] = 1;

Pe = multMatrVec(P, e);

**double**\* x = solveSystem(U, L, Pe);

**for** (**int** i = 0; i < N; i++)

A\_inverse[i][k] = x[i];

**delete**[]x;

**delete**[]Pe;

}

**delete**[]e;

**return** A\_inverse;

}

#**pragma region LU разложение**

**void** swapRows(**double**\*\* m, **int** a, **int** b) // перестановка строк

{

**double** temp;

**for** (**int** i = 0; i < N; i++)

{

temp = m[a][i];

m[a][i] = m[b][i];

m[b][i] = temp;

}

}

**int** indexMax(**double**\*\* m, **int** j) //номер строки с макс. элементом по столбцу

{

**int** inx = j;

**for** (**int** i = j; i < N; i++)

**if** (abs(m[i][j]) > abs(m[inx][j]))

inx = i;

**return** inx;

}

**void** getLU(**double**\*\* U, **double**\*\* L, **double**\*\* P, **double**\*\* A)

{

**double**\*\* subA = createAndCopyMatrix(A);

**double** maxE;

**int** i, j, k, m;

**int** r = N; // ранг матрицы

**for** (k = 0; k < N; k++)

{

m = indexMax(U, k); //номер строки с макс. элементом

**if** (k != m) //поменять строки местами

{

swapRows(U, m, k);

swapRows(L, m, k);

swapRows(P, m, k);

swapRows(subA, m, k);

//countswaps++;

output << " k = " << k + 1 << " m = " << m + 1 << endl;

}

maxE = U[k][k];

**for** (j = k; j < N; j++)

U[k][j] /= maxE;

**if** (abs(maxE) < 0.00001) r = k;

**for** (i = k + 1; i < N; i++)

{

**double** tmp = U[i][k];

**for** (**int** j = k; j < N; j++)

U[i][j] = U[i][j] - U[k][j] \* tmp;

}

**for** (**int** j = 0; j <= k; j++) {

L[k][j] = subA[k][j];

**for** (**int** i = 0; i < j; i++)

L[k][j] -= L[k][i] \* U[i][j];

}

/\*output << " U = " << endl;

        printMatrix(U, 0);

        output << " L = " << endl;

        printMatrix(L, 0);\*/

}

//output << " Rang(A) = " << r << endl;

deleteMatrix(subA);

}

#**pragma endregion**

//значение нормы невязки

**double** getNormResidual(**double**\*\* A, **double**\* x\_next, **double**\* x, **double**\* b)

{

**return** normVector(subVec(multMatrVec(A, x), b)) / normVector(x\_next);

}

//считаем оценку нормы матрицы перехода q;

**double** getNormQ(**double**\* x\_next, **double**\* x, **double**\* x\_prev)

{

**return** normVector(subVec(x\_next, x)) / normVector(subVec(x, x\_prev));

}

// считаем оценку погрешности

**double** getErr(**double**\* x\_next, **double**\* x, **double** q)

{

**return** normVector(subVec(x\_next, x)) \* (1 - q) / q;

}

**void** copyVectors(**double**\* x\_next, **double**\* x, **double**\* x\_prev)

{

**for** (**int** i = 0; i < N; i++)

{

x\_prev[i] = x[i];

x[i] = x\_next[i];

}

}

**double** getTau(**double**\*\* A, **double**\* x, **double**\* b)

{

**double**\* r = subVec(multMatrVec(A, x), b);

**double**\* Ar = multMatrVec(A, r);

**return** scalarMult(r, r) / scalarMult(Ar, r);

}

**double**\* methodSimpleIterations(**double**\*\* A, **double**\* b)

{

output << "\n Метод простой итерации \n";

printHead();

**double**\* x = **new** **double**[N]{ 0.0, 0.0, 0.0, 0.0 };

**double** x\_prev[N] = { 1.0, 1.0, 1.0 ,1.0 };

**double** x\_next[N] = { 0.0, 0.0, 0.0 ,0.0 };

**int** itr = 1;

**double** q, norm, err;

**double** tau = gamma \* 2 / normA;

**do**

{

//получение следующего значения вектора

**for** (**int** i = 0; i < N; i++)

{

**double** temp = 0;

x\_next[i] = 0;

**for** (**int** j = 0; j < N; j++)

temp += A[i][j] \* x[j];

x\_next[i] = x[i] + tau \* (b[i] - temp);

}

//норма невязки

norm = getNormResidual(A, x\_next, x, b);

//норма вектора q

q = getNormQ(x\_next, x, x\_prev);

//погрешность

err = getErr(x\_next, x, q);

printRow(itr, tau, q, norm, err, x\_next, 0, 0);

itr++;

copyVectors(x\_next, x, x\_prev);

} **while** (norm > epsilon);

**return** x;

}

**double**\* methodSteepestDescent(**double**\*\* A, **double**\* b)

{

output << "\n Метод наискорейшего спуска\n";

printHead();

**double**\* x = **new** **double**[N] { 0.0, 0.0, 0.0, 0.0 };

**double** x\_prev[N] = { 1.0, 1.0, 1.0 ,1.0 };

**double** x\_next[N] = { 0.0, 0.0, 0.0 ,0.0 };

**double**\* r;

**int** itr = 1;

**double** q, norm, err;

**double** tau;

**do**

{

tau = getTau(A, x, b);

r = subVec(multMatrVec(A, x), b);

//получение следующего значения вектора

**for** (**int** i = 0; i < N; i++)

{

**double** temp = 0;

x\_next[i] = 0;

**for** (**int** j = 0; j < N; j++)

temp += A[i][j] \* x[j];

x\_next[i] = x[i] + tau \* (b[i] - temp);

/\* x\_next[i] = x[i] - tau \* r[i];\*/

}

//норма невязки

norm = getNormResidual(A, x\_next, x, b);

//норма вектора q

q = getNormQ(x\_next, x, x\_prev);

//погрешность

err = getErr(x\_next, x, q);

printRow(itr, tau, q, norm, err, x\_next, 0, 0);

itr++;

copyVectors(x\_next, x, x\_prev);

**delete**[] r;

} **while** (norm > epsilon);

**return** x;

}

**void** getNextVectSOR(**double**\*\* A, **double**\* x\_next, **double**\* x, **double**\* b, **double** omega)

{

//формула с лекции

**for** (**int** i = 0; i < N; i++)

{

**double** sumOne = 0;

**double** sumTwo = 0;

**for** (**int** j = 0; j < i; j++)

sumOne += A[i][j] \* x\_next[j];

**for** (**int** k = i + 1; k < N; k++)

sumTwo += A[i][k] \* x[k];

**double** x\_next\_wave = (1 / A[i][i]) \* (b[i] - sumOne - sumTwo);

x\_next[i] = x[i] + omega \* (x\_next\_wave - x[i]);

}

}

**double** getOptimalOmega(**double**\*\* A, **double**\* b)

{

output << "\n Метод П�'Р - выбор оптимального w\n";

**double** omega = 0.1;

**double** optimalPEPEGA = omega;

**double** eps = 0.01;

**double** norm = 0;

**int** minItr = INT\_MAX;

**while** (omega < 2)

{

**double** x[N] = { 0.0, 0.0, 0.0, 0.0 }; //k

**double** x\_next[N] = { 0.0, 0.0, 0.0 ,0.0 }; //k+1

**int** itr = 0;

**do** {

itr++;

getNextVectSOR(A, x\_next, x, b, omega);

norm = getNormResidual(A, x\_next, x, b);

**for** (**int** i = 0; i < N; i++)

x[i] = x\_next[i];

} **while** (norm > eps);

**if** (itr < minItr)

{

minItr = itr;

optimalPEPEGA = omega;

}

output << " w=" << omega << " Itr=" << itr << endl;

omega += 0.1;

}

output << endl;

output << " w\*= " << optimalPEPEGA << " ItrMin=" << minItr << endl;

**return** optimalPEPEGA;

}

**double**\* methodSOR(**double**\*\* A, **double**\* b)

{

**double** omega = getOptimalOmega(A, b);

**double**\* x = **new** **double**[N] { 0.0, 0.0, 0.0, 0.0 }; //k

**double** x\_prev[N] = { 1.0, 1.0, 1.0 ,1.0 }; //k-1

**double** x\_next[N] = { 0.0, 0.0, 0.0 ,0.0 }; //k+1

**int** itr = 0;

**double** norm, q, err;

output << "\n Метод П�'Р\n";

printHead();

**do**

{

itr++;

//получение следующего значения вектора

getNextVectSOR(A, x\_next, x, b, omega);

//норма вектора q

q = getNormQ(x\_next, x, x\_prev);

//погрешность

err = getErr(x\_next, x, q);

//норма невязки

norm = getNormResidual(A, x\_next, x, b);

copyVectors(x\_next, x, x\_prev);

printRow(itr, omega, q, norm, err, x\_next, 0, 0);

} **while** (norm > epsilon);

**return** x;

}

**double** getScalarR(**double**\*\* A, **double**\* x, **double**\* b)

{

**double**\* r = subVec(multMatrVec(A, x), b);

**return** scalarMult(r, r);

}

**double**\* methodConjugateGradient(**double**\*\* A, **double**\* b)

{

**double**\* x = **new** **double**[N] { 0.0, 0.0, 0.0, 0.0 }; //k

**double** x\_prev[N] = { 1.0, 1.0, 1.0 ,1.0 }; //k-1

**double** x\_next[N] = { 0.0, 0.0, 0.0 ,0.0 }; //k+1

**int** itr = 0;

**double** norm, q, err, tau, tauPrev, alpha = 1;

output << "\n Метод сопряженных градиентов\n";

printHead();

**double** scalarR;

**double** scalarRPrev;

**double**\* r;

**do** {

itr++;

tau = getTau(A, x, b);

scalarR = getScalarR(A, x, b);

scalarRPrev = getScalarR(A, x\_prev, b);

r = subVec(multMatrVec(A, x), b);

//на первом шаге alpha = 1. Формулы по лекции(в соотв. топике)

**if** (itr != 1)

alpha = 1 / (1 - (tau \* scalarR) / (tauPrev \* alpha \* scalarRPrev));

**for** (**int** i = 0; i < N; i++)

x\_next[i] = alpha \* x[i] + (1 - alpha) \* x\_prev[i] - tau \* alpha \* r[i];

//норма вектора q

q = getNormQ(x\_next, x, x\_prev);

//оценка погрешности

err = getErr(x\_next, x, q);

//норма невязки

norm = getNormResidual(A, x\_next, x, b);

copyVectors(x\_next, x, x\_prev);

tauPrev = tau;

printRow(itr, tau, q, norm, err, x\_next, 1, alpha);

} **while** (norm > epsilon);

**return** x;

}

**int** main()

{

srand(time(0));

output.open(fileOutput);

**double**\*\* A = readMatrix(); //исходная матрица А

**double**\*\* U = createAndCopyMatrix(A); // матрица U изначально равна исходной

**double**\*\* L = **new** **double**\* [N]; // матрица L

**double**\*\* P = **new** **double**\* [N]; //матрица перестановок

**double**\*\* A\_reverse; // А^(-1)

**double** b[] = { 1, 2, 3, 4 };

**for** (**int** i = 0; i < N; i++)

{

P[i] = **new** **double**[N];

L[i] = **new** **double**[N];

}

**for** (**int** i = 0; i < N; i++)

**for** (**int** j = 0; j < N; j++)

{

L[i][j] = 0; // матрица L изначально нулевая

P[i][j] = i == j ? 1 : 0; // матрица P изначально единичная

}

getLU(U, L, P, A);

**double**\* Pb = multMatrVec(P, b);

**double**\* x = solveSystem(U, L, Pb);

A\_reverse = inverseMatrix(U, L, P);

output << " Variant = 27b" << endl;

output << " b:" << endl;

printVector(b, 0);

output << " A:" << endl;

printMatrix(A, 0);

normA = EuclideanNorm(A);

condA = normA \* EuclideanNorm(A\_reverse);

output << "\n Норма матрицы = " << normA << endl;

**double**\* x\_simpleIter = methodSimpleIterations(A, b);

**double**\* x\_descent = methodSteepestDescent(A, b);

**double**\* x\_SOR = methodSOR(A, b);

**double**\* x\_Gradient = methodConjugateGradient(A, b);

output << "\n Число обусловленности = " << setw(10) << setprecision(5) << fixed << condA << endl;

output << "\n Теоретическая оценка чиcла итераций ";

output << "\n Метод простых итераций " << **int**(log(1 / epsilon) / 2 \* condA);

output << "\n Метод наискорейшего спуска " << **int**(log(1 / epsilon) \* condA);

output << "\n Метод П�'Р " << **int**(log(1 / epsilon) / 4 \* sqrt(condA));

output << "\n Метод сопряженных градиентов " << **int**(log(2 / epsilon) / 2 \* sqrt(condA));

output << "\n\n LU разложение \n";

printVector(x, 0);

output << "\n Разница между LU разложением и методом простых итераций \n";

printVector(subVec(x, x\_simpleIter), 0);

output << "\n Разница между LU разложением и методом наискорейшего спуска \n";

printVector(subVec(x, x\_descent), 0);

output << "\n Разница между LU разложением и методом П�'Р \n";

printVector(subVec(x, x\_SOR), 0);

output << "\n Разница между LU разложением и методом сопряженных градиентов \n";

printVector(subVec(x, x\_Gradient), 0);

deleteMatrix(A);

deleteMatrix(U);

deleteMatrix(L);

deleteMatrix(P);

**delete**[] Pb;

**delete**[] x;

**delete**[] x\_simpleIter;

**delete**[] x\_descent;

**delete**[] x\_SOR;

**delete**[] x\_Gradient;

output.close();

}