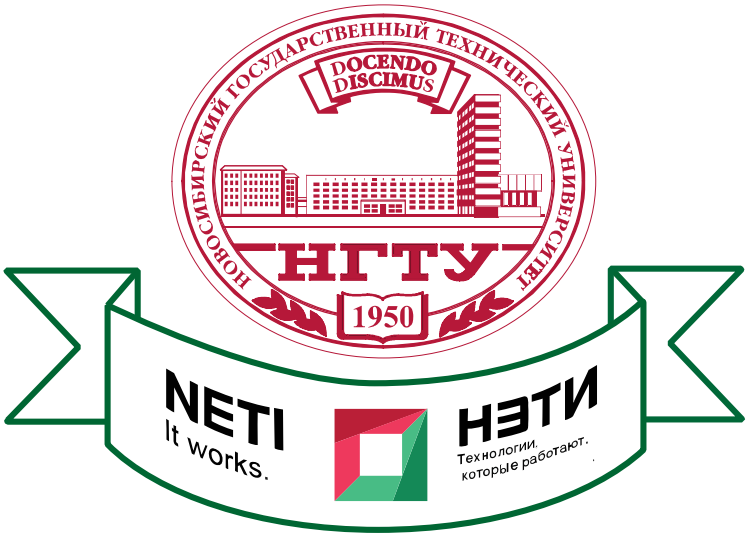
Министерство науки и высшего образования
Российской Федерации

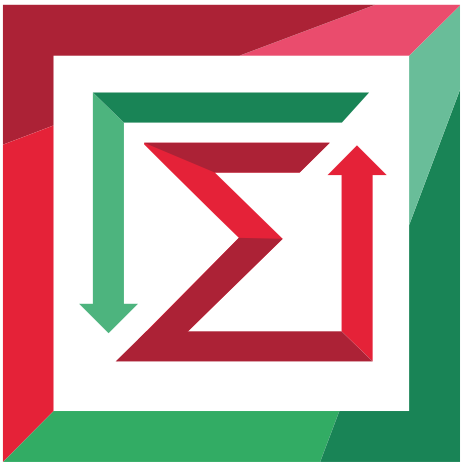
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Новосибирский государственный технический университет»



Кафедра прикладной математики

Практическое задание № 1
по дисциплине «Численные методы»

Прямые методы решения СЛАУ



Факультет:	ПМИ
Группа:	ПМ-71
Студент:	Востриков Вячеслав
Вариант:	6
Преподаватели:	Патрушев И.И.
	Задорожный А.Г.
	Персова М.Г.

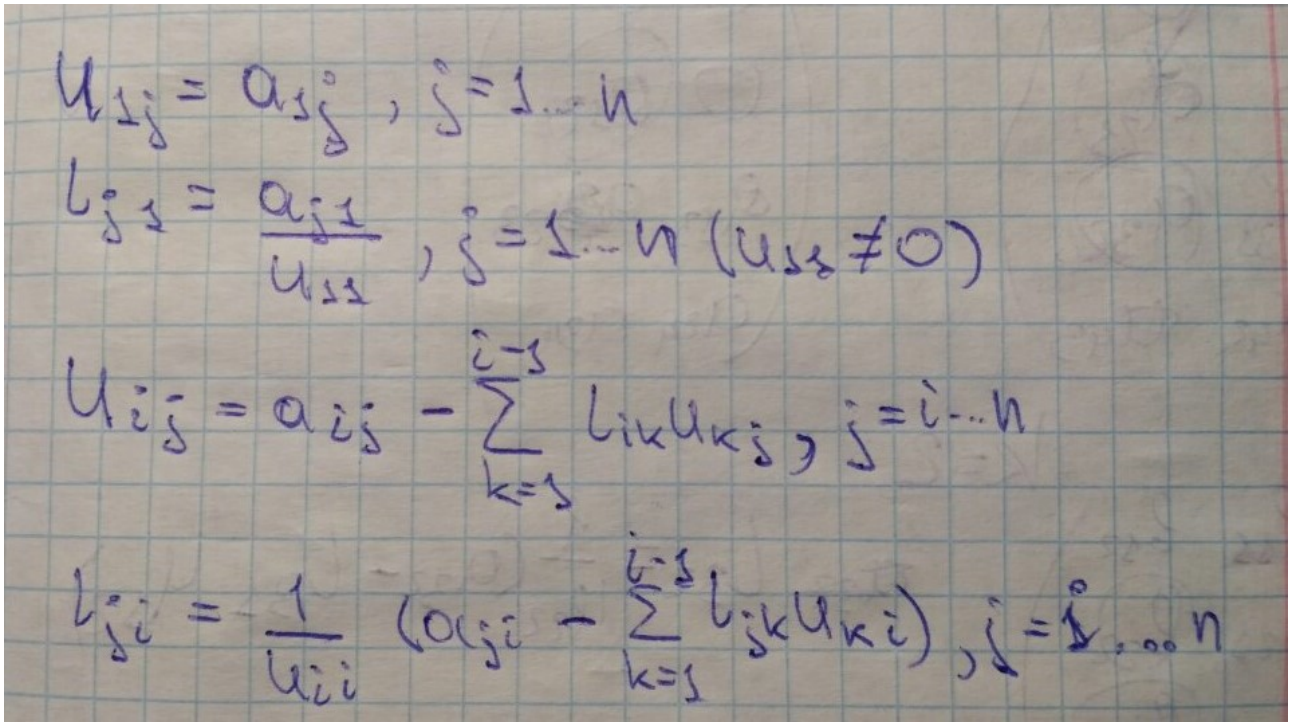
Новосибирск
2019

1. **Цель работы**

   Разработать программу решения СЛАУ прямым методом с хранением матрицы в профильном или ленточном формате. Исследовать накопление погрешности и ее зависимость от числа обусловленности. Сравнить реализованный метод по точности получаемого решения и количеству действий с методом Гаусса.

2. **Теоретическая часть**

   Построенное разложение по варианту: $LU^{(*)}$, где L - нижняя треугольная матрица с 1 на диагонали, а U - верхняя треугольная матрица.



3. **Набор тестов**

№	A				b	Результат
1	1	0	0	0	1	1
	0	2	0	0	2	1
	0	0	3	0	3	1
	0	0	0	4	4	1
2	1	1	0	0	1	0.8333333134651184
	5	2	3	0	2	0.1666666716337204
	0	7	3	2	5	-0.8333333134651184
	0	0	8	4	6	3.166666507720947
3	1	2	3	44	1	Ошибка
	0	0	5	7	2	
	6	9	0	0	3	
	0	1	2	1	4	

4. **Влияние увеличения числа обусловленности на точность решения**

   Пусть дана исходная матрица A (незаполненные ячейки имеют значение 0):

   15 -3 -4 -3 -4

   0 4 -2 -1 0 -1

-3	-4	15	-1	-3	-1	-3			
-4	-4	-1	17	-1	-1	-4	-2		
-2	-2	-1	0	15	-2	-2	-3	-3	
	0	-2	-1	-4	13	-2	-2	-1	-1
		-2	-3	-4	-4	17	0	-2	-2
			-3	-1	-4	0	15	-4	-3
				-4	-2	-2	-3	11	0
					-2	-2	0	-4	8

И вектор F^T:

-35	-8	-12	-2	-11	-1	19	13	29	18

Так как при увеличении числа обусловленности изменяется только первый элемент диагонали, для построения вектора F достаточно прибавить к первому элементу то же значение (10^-k). Постепенно увеличивая значение k, проведем исследование. Причем изначально матрица A является вырожденной, но при прибавлении добавки матрица становится невырожденной. А при повышении k матрица приближается к исходной и становится вырожденной.

Вывод: С увеличением числа k увеличивается число обусловленности, следовательно, и падает точность измерения. Также при k = 6, для 4-байтового вещественного типа, и при k = 15 для 8-байтового вещественного типа погрешность выходит в 1 знак.

k	x^k (одинарн)	x^*-x^k (одинарн)	x^k (двойная)	x^*-x^k (двойная)	x^k (скаляр. произв.)	x^*-x^k (скаляр. произв.)
0	9,9999881E-01	1,1920929E-06	9,99999999999993E-01	7,32747196252603E-15	1,0000012E+00	-1,1920929E-06
	1,9999987E+00	1,3113022E-06	1,99999999999999E+00	9,32587340685131E-15	2,0000014E+00	-1,4305115E-06
	2,9999986E+00	1,4305115E-06	2,99999999999999E+00	8,43769498715119E-15	3,0000012E+00	-1,1920929E-06
	3,9999988E+00	1,1920929E-06	3,99999999999999E+00	8,88178419700125E-15	4,0000014E+00	-1,4305115E-06
	4,9999986E+00	1,4305115E-06	4,99999999999999E+00	8,88178419700125E-15	5,0000014E+00	-1,4305115E-06
	5,9999981E+00	1,9073486E-06	5,99999999999999E+00	9,76996261670138E-15	6,0000014E+00	-1,4305115E-06
	6,9999981E+00	1,9073486E-06	6,99999999999999E+00	7,99360577730113E-15	7,0000019E+00	-1,9073486E-06
	7,9999990E+00	9,5367432E-07	7,99999999999999E+00	7,99360577730113E-15	8,0000019E+00	-1,9073486E-06
	8,9999990E+00	9,5367432E-07	8,99999999999999E+00	7,10542735760100E-15	9,0000019E+00	-1,9073486E-06
	9,9999981E+00	1,9073486E-06	9,99999999999999E+00	7,10542735760100E-15	1,0000002E+01	-1,9073486E-06
1	1,0000030E+00	-2,9802322E-06	9,99999999999948E-01	5,18474152499948E-14	9,9998105E-01	1,8954277E-05
	2,0000029E+00	-2,8610229E-06	1,99999999999995E+00	5,32907051820075E-14	1,9999808E+00	1,9192696E-05
	3,0000029E+00	-2,8610229E-06	2,99999999999995E+00	5,32907051820075E-14	2,9999807E+00	1,9311905E-05
	4,0000029E+00	-2,8610229E-06	3,99999999999995E+00	5,37347943918576E-14	3,9999807E+00	1,9311905E-05
	5,0000029E+00	-2,8610229E-06	4,99999999999995E+00	5,32907051820075E-14	4,9999804E+00	1,9550323E-05
	6,0000029E+00	-2,8610229E-06	5,99999999999995E+00	5,32907051820075E-14	5,9999804E+00	1,9550323E-05
	7,0000033E+00	-3,3378601E-06	6,99999999999995E+00	5,41788836017076E-14	6,9999804E+00	1,9550323E-05
	8,0000029E+00	-2,8610229E-06	7,99999999999995E+00	5,24025267623074E-14	7,9999809E+00	1,9073486E-05
	9,0000029E+00	-2,8610229E-06	8,99999999999995E+00	5,32907051820075E-14	8,9999809E+00	1,9073486E-05
	1,0000003E+01	-2,8610229E-06	9,99999999999994E+00	5,50670620214078E-14	9,9999809E+00	1,9073486E-05
2	9,9979764E-01	2,0235777E-04	1,00000000000075E+00	-7,52065076881081E-13	9,9917263E-01	8,2737207E-04
	1,9997983E+00	2,0170212E-04	2,00000000000075E+00	-7,52287121486006E-13	1,9991716E+00	8,2838535E-04

	2,9997981E+00	2,0194054E-04	3,00000000000075E+00	-7,53175299905706E-13	2,9991715E+00	8,2850456E-04
	3,9997983E+00	2,0170212E-04	4,00000000000075E+00	-7,53175299905706E-13	3,9991715E+00	8,2850456E-04
	4,9997973E+00	2,0265579E-04	5,00000000000075E+00	-7,52287121486006E-13	4,9991717E+00	8,2826614E-04
	5,9997973E+00	2,0265579E-04	6,00000000000075E+00	-7,53175299905706E-13	5,9991708E+00	8,2921982E-04
	6,9997978E+00	2,0217896E-04	7,00000000000075E+00	-7,53175299905706E-13	6,9991713E+00	8,2874298E-04
	7,9997959E+00	2,0408630E-04	8,00000000000075E+00	-7,53175299905706E-13	7,9991713E+00	8,2874298E-04
	8,9997969E+00	2,0313263E-04	9,00000000000075E+00	-7,54951656745106E-13	8,9991713E+00	8,2874298E-04
	9,9997969E+00	2,0313263E-04	1,00000000000008E+01	-7,54951656745106E-13	9,9991703E+00	8,2969666E-04
3	9,9528235E-01	4,7176480E-03	1,00000000001131E+00	-1,13120623979057E-11	9,9544960E-01	4,5503974E-03
	1,9952816E+00	4,7184229E-03	2,00000000001131E+00	-1,13127285317205E-11	1,9954487E+00	4,5512915E-03
	2,9952817E+00	4,7183037E-03	3,00000000001131E+00	-1,13145048885599E-11	2,9954493E+00	4,5506954E-03
	3,9952815E+00	4,7185421E-03	4,00000000001131E+00	-1,13136167101402E-11	3,9954493E+00	4,5506954E-03
	4,9952822E+00	4,7178268E-03	5,00000000001131E+00	-1,13127285317205E-11	4,9954486E+00	4,5514107E-03
	5,9952817E+00	4,7183037E-03	6,00000000001131E+00	-1,13136167101402E-11	5,9954486E+00	4,5514107E-03
	6,9952822E+00	4,7178268E-03	7,00000000001131E+00	-1,13127285317205E-11	6,9954495E+00	4,5504570E-03
	7,9952822E+00	4,7178268E-03	8,00000000001131E+00	-1,13118403533008E-11	7,9954491E+00	4,5509338E-03
	8,9952822E+00	4,7178268E-03	9,00000000001131E+00	-1,13118403533008E-11	8,9954500E+00	4,5499802E-03
	9,9952812E+00	4,7187805E-03	1,00000000000113E+01	-1,13136167101402E-11	9,9954481E+00	4,5518875E-03
4	1,0270261E+00	-2,7026057E-02	1,00000000001256E+00	-1,25641719250780E-11	1,0422293E+00	-4,2229295E-02
	2,0270264E+00	-2,7026415E-02	2,00000000001256E+00	-1,25641719250780E-11	2,0422301E+00	-4,2230129E-02
	3,0270269E+00	-2,7026892E-02	3,00000000001256E+00	-1,25646160142878E-11	3,0422301E+00	-4,2230129E-02
	4,0270271E+00	-2,7027130E-02	4,00000000001256E+00	-1,25641719250780E-11	4,0422301E+00	-4,2230129E-02
	5,0270267E+00	-2,7026653E-02	5,00000000001256E+00	-1,25641719250780E-11	5,0422301E+00	-4,2230129E-02
	6,0270262E+00	-2,7026176E-02	6,00000000001256E+00	-1,25641719250780E-11	6,0422297E+00	-4,2229652E-02
	7,0270262E+00	-2,7026176E-02	7,00000000001256E+00	-1,25632837466583E-11	7,0422301E+00	-4,2230129E-02
	8,0270262E+00	-2,7026176E-02	8,00000000001256E+00	-1,25641719250780E-11	8,0422297E+00	-4,2229652E-02
	9,0270262E+00	-2,7026176E-02	9,00000000001256E+00	-1,25641719250780E-11	9,0422297E+00	-4,2229652E-02
	1,0027026E+01	-2,7026176E-02	1,00000000000126E+01	-1,25641719250780E-11	1,0042230E+01	-4,2229652E-02
5	1,2307694E+00	-2,3076940E-01	1,00000000226167E+00	-2,26166552153018E-09	1,8367324E+00	-8,3673239E-01
	2,2307699E+00	-2,3076987E-01	2,00000000226167E+00	-2,26166863015465E-09	2,8367336E+00	-8,3673358E-01
	3,2307699E+00	-2,3076987E-01	3,00000000226167E+00	-2,26166907424385E-09	3,8367338E+00	-8,3673382E-01
	4,2307701E+00	-2,3077011E-01	4,00000000226167E+00	-2,26166907424385E-09	4,8367338E+00	-8,3673382E-01
	5,2307701E+00	-2,3077011E-01	5,00000000226167E+00	-2,26166818606544E-09	5,8367338E+00	-8,3673382E-01
	6,2307701E+00	-2,3077011E-01	6,00000000226167E+00	-2,26166907424385E-09	6,8367333E+00	-8,3673334E-01
	7,2307696E+00	-2,3076963E-01	7,00000000226167E+00	-2,26166818606544E-09	7,8367333E+00	-8,3673334E-01
	8,2307692E+00	-2,3076916E-01	8,00000000226167E+00	-2,26166818606544E-09	8,8367329E+00	-8,3673286E-01
	9,2307692E+00	-2,3076916E-01	9,00000000226167E+00	-2,26166818606544E-09	9,8367338E+00	-8,3673382E-01
	1,0230770E+01	-2,3077011E-01	1,00000000022617E+01	-2,26166996242227E-09	1,0836733E+01	-8,3673286E-01
6			9,99999985550496E-01	1,44495040377279E-08	1,0999996E+01	-9,9999962E+00
			1,99999998555049E+00	1,44495067022632E-08	1,1999998E+01	-9,9999981E+00
			2,99999998555049E+00	1,44495055920402E-08	1,2999998E+01	-9,9999981E+00
			3,99999998555049E+00	1,44495064802186E-08	1,3999999E+01	-9,9999990E+00
			4,99999998555049E+00	1,44495064802186E-08	1,4999998E+01	-9,9999981E+00
			5,99999998555049E+00	1,44495064802186E-08	1,5999998E+01	-9,9999981E+00
			6,99999998555049E+00	1,44495064802186E-08	1,7000000E+01	-1,0000000E+01
			7,99999998555049E+00	1,44495064802186E-08	1,7999998E+01	-9,9999981E+00

			8,99999998555049E+00	1,44495082565754E-08	1,9000000E+01	-1,0000000E+01
			9,99999998555049E+00	1,44495064802186E-08	2,0000000E+01	-1,0000000E+01
7			1,00000001884718E+00	-1,88471802609058E-08		
			2,00000001884718E+00	-1,88471798168166E-08		
			3,00000001884718E+00	-1,88471811490842E-08		
			4,00000001884718E+00	-1,88471807049950E-08		
			5,00000001884718E+00	-1,88471798168166E-08		
			6,00000001884718E+00	-1,88471789286382E-08		
			7,00000001884718E+00	-1,88471815931734E-08		
			8,00000001884718E+00	-1,88471798168166E-08		
			9,00000001884718E+00	-1,88471798168166E-08		
			1,00000000188472E+01	-1,88471780404598E-08		
8			1,00000037694366E+00	-3,76943664059937E-07		
			2,00000037694366E+00	-3,76943663837892E-07		
			3,00000037694366E+00	-3,76943664726070E-07		
			4,00000037694366E+00	-3,76943663837892E-07		
			5,00000037694366E+00	-3,76943664726070E-07		
			6,00000037694366E+00	-3,76943663837892E-07		
			7,00000037694366E+00	-3,76943663837892E-07		
			8,00000037694366E+00	-3,76943663837892E-07		
			9,00000037694366E+00	-3,76943663837892E-07		
			1,00000003769437E+01	-3,76943663837892E-07		
9			9,99993717606412E-01	6,28239358801963E-06		
			1,99999371760641E+00	6,28239358957394E-06		
			2,99999371760641E+00	6,28239358801963E-06		
			3,99999371760641E+00	6,28239358890781E-06		
			4,99999371760641E+00	6,28239358935190E-06		
			5,99999371760641E+00	6,28239359024008E-06		
			6,99999371760641E+00	6,28239359112825E-06		
			7,99999371760641E+00	6,28239359024008E-06		
			8,99999371760641E+00	6,28239359024008E-06		
			9,99999371760641E+00	6,28239359201643E-06		
10			1,00013821526399E+00	-1,38215263988828E-04		
			2,00013821526399E+00	-1,38215263989938E-04		
			3,00013821526399E+00	-1,38215263991270E-04		
			4,00013821526399E+00	-1,38215263990382E-04		
			5,00013821526399E+00	-1,38215263990382E-04		
			6,00013821526399E+00	-1,38215263990382E-04		
			7,00013821526399E+00	-1,38215263991270E-04		
			8,00013821526399E+00	-1,38215263991270E-04		
			9,00013821526399E+00	-1,38215263991270E-04		
			1,00001382152640E+01	-1,38215263989494E-04		
11			1,00150810607012E+00	-1,50810607012364E-03		
			2,00150810607012E+00	-1,50810607012453E-03		
			3,00150810607013E+00	-1,50810607012586E-03		
			4,00150810607013E+00	-1,50810607012541E-03		

			5,00150810607012E+00	-1,50810607012453E-03		
			6,00150810607012E+00	-1,50810607012453E-03		
			7,00150810607012E+00	-1,50810607012364E-03		
			8,00150810607013E+00	-1,50810607012630E-03		
			9,00150810607013E+00	-1,50810607012630E-03		
			1,00015081060701E+01	-1,50810607012275E-03		
12			9,94974874371858E-01	5,02512562814172E-03		
			1,99497487437186E+00	5,02512562814217E-03		
			2,99497487437186E+00	5,02512562814195E-03		
			3,99497487437186E+00	5,02512562814239E-03		
			4,99497487437186E+00	5,02512562814239E-03		
			5,99497487437186E+00	5,02512562814150E-03		
			6,99497487437186E+00	5,02512562814239E-03		
			7,99497487437186E+00	5,02512562814150E-03		
			8,99497487437186E+00	5,02512562814061E-03		
			9,99497487437186E+00	5,02512562814417E-03		
13			1,23178807947020E+00	-2,31788079470197E-01		
			2,23178807947020E+00	-2,31788079470199E-01		
			3,23178807947020E+00	-2,31788079470200E-01		
			4,23178807947020E+00	-2,31788079470199E-01		
			5,23178807947020E+00	-2,31788079470199E-01		
			6,23178807947020E+00	-2,31788079470200E-01		
			7,23178807947020E+00	-2,31788079470197E-01		
			8,23178807947020E+00	-2,31788079470199E-01		
			9,23178807947020E+00	-2,31788079470199E-01		
			1,02317880794702E+01	-2,31788079470201E-01		
14			-3,52941176470587E-01	1,35294117647059E+00		
			6,47058823529412E-01	1,35294117647059E+00		
			1,64705882352941E+00	1,35294117647059E+00		
			2,64705882352941E+00	1,35294117647059E+00		
			3,64705882352941E+00	1,35294117647059E+00		
			4,64705882352941E+00	1,35294117647059E+00		
			5,64705882352941E+00	1,35294117647059E+00		
			6,64705882352941E+00	1,35294117647059E+00		
			7,64705882352941E+00	1,35294117647059E+00		
			8,64705882352941E+00	1,35294117647059E+00		
15			-1,04906215106289E-16	1,00000000000000E+00		
			1,00000000000000E+00	9,99999999999999E-01		
			2,00000000000000E+00	9,99999999999999E-01		
			3,00000000000000E+00	9,99999999999999E-01		
			4,00000000000000E+00	1,00000000000000E+00		
			5,00000000000000E+00	1,00000000000000E+00		
			6,00000000000000E+00	1,00000000000000E+00		
			7,00000000000000E+00	1,00000000000000E+00		
			8,00000000000000E+00	1,00000000000000E+00		
			9,00000000000000E+00	1,00000000000000E+00		

5. Исследование на матрице Гильберта различной размерности

k	x^k (одинарн)	x^*-x^k (одинарн)	x^k (двойная)	x^*-x^k (двойная)	xk (скаляр, произв,)	x*-xk (скаляр, произв,)
1	1,0000000E+00	0,0000000E+00	1,00000000000000E+00	0,00E+00	1,0000000E+00	0,0000000E+00
2	9,9999964E-01	3,5762787E-07	9,99999999999975E-01	2,50E-14	9,9999976E-01	2,4000000E-07
	2,0000007E+00	-7,1525573E-07	2,00000000000005E+00	-5,02E-14	2,0000005E+00	-5,0000000E-07
3	1,0000012E+00	-1,1920929E-06	9,99999999999769E-01	2,31E-13	9,9999525E-01	4,7500000E-06
	1,9999903E+00	9,6559525E-06	2,00000000000131E+00	-1,31E-12	2,0000272E+00	-2,7200000E-05
	3,0000107E+00	-1,0728836E-05	2,99999999999874E+00	1,26E-12	2,9999734E+00	2,6600000E-05
4	9,9997139E-01	2,8610229E-05	9,99999999999750E-01	2,50E-13	9,9999884E-01	1,1600000E-06
	2,0003219E+00	-3,2186508E-04	2,00000000000199E+00	-1,99E-12	2,0000093E+00	-9,3000000E-06
	2,9992275E+00	7,7247620E-04	2,99999999999638E+00	3,62E-12	2,9999807E+00	1,9300000E-05
	4,0005007E+00	-5,0067902E-04	4,00000000000187E+00	-1,87E-12	4,0000119E+00	-1,1900000E-05
5	1,0001148E+00	-1,1480000E-04	9,99999999999328E-01	6,72E-13	9,9980693E-01	1,9307000E-04
	1,9976189E+00	2,3811000E-03	2,00000000000552E+00	-5,52E-12	2,0034931E+00	-3,4931000E-03
	3,0108576E+00	-1,0857600E-02	2,99999999999649E+00	3,51E-12	2,9853027E+00	1,4697300E-02
	3,9830155E+00	1,6984500E-02	3,99999999998360E+00	1,64E-11	4,0217813E+00	-2,1781300E-02
	5,0085135E+00	-8,5135000E-03	5,00000000001591E+00	-1,59E-11	4,9895014E+00	1,0498600E-02
6	9,9816918E-01	1,8308200E-03	9,99999999995613E-01	4,39E-12	1,0004581E+00	-4,5810000E-04
	2,0535684E+00	-5,3568400E-02	2,00000000004844E+00	-4,84E-11	1,9870615E+00	1,2938500E-02
	2,6310308E+00	3,6896920E-01	3,00000000000215E+00	-2,15E-12	3,0858150E+00	-8,5815000E-02
	4,9724751E+00	-9,7247510E-01	3,99999999939744E+00	6,03E-10	3,7813147E+00	2,1868530E-01
	3,9157352E+00	1,0842648E+00	5,00000000115131E+00	-1,15E-09	5,2368404E+00	-2,3684040E-01
	6,4306226E+00	-4,3062260E-01	5,99999999939896E+00	6,01E-10	5,9082565E+00	9,1743500E-02
7	9,9434304E-01	5,6569600E-03	9,99999999591096E-01	4,09E-10	1,0179768E+00	-1,7976800E-02
	2,2135539E+00	-2,1355390E-01	2,00000001633666E+00	-1,63E-08	1,2886630E+00	7,1133700E-01
	1,0251199E+00	1,9748801E+00	2,99999984237384E+00	1,58E-07	9,8075950E+00	-6,8075950E+00
	1,1438265E+01	-7,4382650E+00	4,00000061367291E+00	-6,14E-07	-2,2323260E+01	2,6323260E+01
	-8,2955532E+00	1,3295553E+01	4,99999887355962E+00	1,13E-06	5,3042825E+01	-4,8042825E+01
	1,7253086E+01	-1,1253086E+01	6,00000097439771E+00	-9,74E-07	-3,5357869E+01	4,1357869E+01
	3,3684211E+00	3,6315789E+00	6,99999967976998E+00	3,20E-07	2,0536052E+01	-1,3536052E+01
8			9,99999999465276E-01	5,35E-10		
			2,00000002718260E+00	-2,72E-08		
			2,99999965722309E+00	3,43E-07		
			4,00000180991784E+00	-1,81E-06		
			4,99999521514792E+00	4,78E-06		

			6,00000667659648E+00	-6,68E-06		
			6,99999530054052E+00	4,70E-06		
			8,00000131422900E+00	-1,31E-06		
9			9,99999996897904E-01	3,10E-09		
			2,00000021032587E+00	-2,10E-07		
			2,99999647167277E+00	3,53E-06		
			4,00002507305477E+00	-2,51E-05		
			4,99990823954306E+00	9,18E-05		
			6,00018718404796E+00	-1,87E-04		
			6,99978506563310E+00	2,15E-04		
			8,00012985876299E+00	-1,30E-04		
			8,99996789788910E+00	3,21E-05		
10			1,00000002744718E+00	-2,74E-08		
			1,99999767735980E+00	2,32E-06		
			3,00004870138092E+00	-4,87E-05		
			3,99956269904645E+00	4,37E-04		
			5,00206490484456E+00	-2,06E-03		
			5,99437124566976E+00	5,63E-03		
			7,00916899999340E+00	-9,17E-03		
			7,99119403026945E+00	8,81E-03		
			9,00459804276620E+00	-4,60E-03		
			9,99899365504177E+00	1,01E-03		
11			1,00000032612681E+00	-3,26E-07		
			1,99996499532988E+00	3,50E-05		
			3,00092587023537E+00	-9,26E-04		
			3,98948660486775E+00	1,05E-02		
			5,06343325970098E+00	-6,34E-02		
			5,77460383966878E+00	2,25E-01		
			7,49517935480327E+00	-4,95E-01		
			7,31972615937441E+00	6,80E-01		
			9,56884051482525E+00	-5,69E-01		
			9,73527743738678E+00	2,65E-01		
			1,10525618854686E+01	-5,26E-02		
12			1,00000006506506E+00	-6,51E-08		
			1,99998932871423E+00	1,07E-05		
			3,00039670026756E+00	-3,97E-04		
			3,99391398279265E+00	6,09E-03		

			5,04877089043284E+00	-4,88E-02		
			5,77029183707640E+00	2,30E-01		
			7,67676913941088E+00	-6,77E-01		
			6,71746495657288E+00	1,28E+00		
			1,05624311409741E+01	-1,56E+00		
			8,81780206385322E+00	1,18E+00		
			1,15054923431685E+01	-5,05E-01		
			1,19066773934031E+01	9,33E-02		

Вывод: для числа 4-х байтового уже при k = 6 погрешность выходит в первый знак. Для скалярного произведения при k = 7, а для 8 байтового числа при k = 12.

6. **Сравнение реализованного метода с методом Гаусса**

Сравнение методов на матрицах Гильберта по точности:

LU* - разложение		Метод гаусса	
xk	x*- xk	xk	x*- xk
1,000000000000000E+00	0,000000000000000E+00	1,000000000000000E+00	0,000000000000000E+00
1,000000000000000E+00	0,000000000000000E+00	1,000000000000000E+00	0,000000000000000E+00
2,000000000000000E+00	0,000000000000000E+00	2,000000000000000E+00	0,000000000000000E+00
1,000000000000010E+00	-9,992007221626410E-15	1,000000000000000E+00	0,000000000000000E+00
1,999999999999960E+00	3,996802888650560E-14	1,999999999999980E+00	1,998401444325280E-14
3,000000000000040E+00	-3,996802888650560E-14	3,000000000000020E+00	-1,998401444325280E-14
1,000000000000030E+00	-2,997602166487920E-14	1,000000000000040E+00	-3,996802888650560E-14
1,999999999999610E+00	3,899103262483550E-13	1,999999999999530E+00	4,700684286262910E-13
3,000000000000950E+00	-9,499068198692840E-13	3,000000000001140E+00	-1,139977001685110E-12
3,999999999999380E+00	6,199485369506870E-13	3,999999999999250E+00	7,500666754367560E-13
1,000000000000020E+00	-1,998401444325280E-14	9,999999999995740E-01	4,259925745486730E-13
1,999999999999690E+00	3,099742684753440E-13	2,000000000008160E+00	-8,160139230994900E-12
3,000000000001420E+00	-1,420197293100500E-12	2,999999999964430E+00	3,557021344136050E-11
3,999999999997670E+00	2,330136084083280E-12	4,000000000054010E+00	-5,401012970196460E-11
5,000000000001220E+00	-1,220357148667970E-12	4,999999999973490E+00	2,651034947120930E-11
1,000000000001900E+00	-1,900035684343490E-12	1,000000000000260E+00	-2,600142323672120E-13
1,999999999946410E+00	5,359002130944650E-11	1,999999999991280E+00	8,719913680010900E-12
3,000000000361050E+00	-3,610498566786190E-10	3,000000000065180E+00	-6,517986150811340E-11
3,999999999062560E+00	9,374399034811630E-10	3,999999999818350E+00	1,816498063078600E-10
5,000000001033810E+00	-1,033810370643100E-09	5,000000000210810E+00	-2,108100360942440E-10
5,999999999592840E+00	4,071596393373510E-10	5,999999999913730E+00	8,626965808389290E-11
9,999999999729520E-01	2,704803048203530E-11	1,000000000002760E+00	-2,760014439218140E-12
2,000000001055890E+00	-1,055890042067630E-09	1,999999999872400E+00	1,275999306216140E-10
2,999999989987480E+00	1,001252014631860E-08	3,000000001355350E+00	-1,355350054410560E-09
4,000000038466420E+00	-3,846642027127700E-08	3,999999994355020E+00	5,644980038965740E-09
4,999999930125910E+00	6,987408962544350E-08	5,000000010890950E+00	-1,089095036377330E-08
6,000000059936290E+00	-5,993628970912820E-08	5,999999990211770E+00	9,788229782259350E-09
6,999999980438090E+00	1,956191031382560E-08	7,000000003315740E+00	-3,315739682818730E-09
9,999999999462170E-01	5,378297807112630E-11	1,000000000156510E+00	-1,565099161382480E-10
2,000000002729620E+00	-2,729620085517580E-09	1,999999991538210E+00	8,461789935410020E-09

2,999999965422870E+00	3,457713004806350E-08	3,000000111090900E+00	-1,110909000345830E-07
4,000000183822520E+00	-1,838225198724790E-07	3,999999396585640E+00	6,034143598121490E-07
4,999999510395410E+00	4,896045897595510E-07	5,000001628886310E+00	-1,628886310278690E-06
6,000000688226820E+00	-6,882268204222440E-07	5,999997690528520E+00	2,309471479833290E-06
6,999999512184470E+00	4,878155301923930E-07	7,000001646130450E+00	-1,646130449728390E-06
8,000000137307450E+00	-1,373074507426960E-07	7,999999534966640E+00	4,650333602640440E-07
1,000000000996160E+00	-9,961600433427980E-10	9,999999999849220E-01	1,507804991973670E-11
1,999999932962150E+00	6,703784993788990E-08	2,000000000571450E+00	-5,714499984321720E-10
3,000001112339330E+00	-1,112339329978340E-06	2,999999995251120E+00	4,748879955940310E-09
3,999992186719750E+00	7,813280249902020E-06	4,000000010327680E+00	-1,032768004449740E-08
5,000028275412290E+00	-2,827541228977990E-05	5,000000023401640E+00	-2,340163973713060E-08
5,999942920504960E+00	5,707949503985790E-05	5,999999859785390E+00	1,402146097007060E-07
7,000064915519100E+00	-6,491551909970640E-05	7,000000242029000E+00	-2,420290003968260E-07
7,999961121223910E+00	3,887877609010100E-05	7,999999815490230E+00	1,845097701291820E-07
9,000009534880700E+00	-9,534880700101670E-06	9,000000053165870E+00	-5,316586992876180E-08
1,000000000567870E+00	-5,678699732669660E-10	1,000000003468520E+00	-3,468519915728050E-09
1,999999932892280E+00	6,710772004758780E-08	1,999999696334530E+00	3,036654701027430E-07
3,000001710993370E+00	-1,710993370096500E-06	3,000006525502470E+00	-6,525502469934000E-06
3,999982483981540E+00	1,751601846011750E-05	3,999940304844900E+00	5,969515509995920E-05
5,000090948842570E+00	-9,094884256999340E-05	5,000286036413000E+00	-2,860364129997350E-04
5,999733485055970E+00	2,665149440304050E-04	5,999210990735090E+00	7,890092649098790E-04
7,000459655885630E+00	-4,596558856295730E-04	7,001297928133080E+00	-1,297928133079830E-03
7,999537609482360E+00	4,623905176401880E-04	7,998743125278220E+00	1,256874721780040E-03
9,000250887728590E+00	-2,508877285904990E-04	9,000660924795090E+00	-6,609247950901680E-04
9,999943283222090E+00	5,671677791063700E-05	9,999854461942040E+00	1,455380579606920E-04
1,000000043276410E+00	-4,327641001111720E-08	1,000000051716790E+00	-5,171678996163110E-08
1,999995475233990E+00	4,524766010094440E-06	1,999994565041110E+00	5,434958890049300E-06
3,000117157453060E+00	-1,171574530598460E-04	3,000141349969400E+00	-1,413499693998510E-04
3,998692894619980E+00	1,307105380019990E-03	3,998416922233360E+00	1,583077766639910E-03
5,007770542482770E+00	-7,770542482769650E-03	5,009442579039900E+00	-9,442579039900420E-03
5,972735526095560E+00	2,726447390443990E-02	5,966772112329120E+00	3,322788767088000E-02
7,059249303633460E+00	-5,924930363346000E-02	7,072394285747410E+00	-7,239428574740980E-02
7,919372561472760E+00	8,062743852724010E-02	7,901260023346420E+00	9,873997665358040E-02
9,066859921294650E+00	-6,685992129465030E-02	9,082046233859810E+00	-8,204623385980980E-02
9,969114458223300E+00	3,088554177669960E-02	9,962030035774750E+00	3,796996422524930E-02
1,100609214178260E+01	-6,092141782600540E-03	1,100750187344360E+01	-7,501873443599650E-03
1,000000095808750E+00	-9,580875004466800E-08	9,999999914048770E-01	8,595122946708500E-09
1,999987657051290E+00	1,234294870999800E-05	2,000001357477790E+00	-1,357477790175920E-06
3,000393695386350E+00	-3,936953863501460E-04	2,999950629584820E+00	4,937041518005000E-05
3,994569767235990E+00	5,430232764009890E-03	4,000745933622110E+00	-7,459336221096180E-04
5,040235976479040E+00	-4,023597647903990E-02	4,994092571781610E+00	5,907428218390190E-03
5,821546926034900E+00	1,784530739651000E-01	6,027555500902510E+00	-2,755550090251010E-02
7,501346967401220E+00	-5,013469674012200E-01	6,919481380548060E+00	8,051861945194010E-02
7,085809167139020E+00	9,141908328609800E-01	8,151499818375900E+00	-1,514998183759000E-01
1,007881933160970E+01	-1,078819331609700E+00	8,816603921706360E+00	1,833960782936400E-01
9,205236300247120E+00	7,947636997528790E-01	1,013797957491370E+01	-1,379795749137000E-01
1,133219518392920E+01	-3,321951839291990E-01	1,094130328305700E+01	5,869671694300040E-02

1,193985885240870E+01	6,014114759130070E-02	1,201078605309960E+01	-1,078605309960070E-02

Сравнение методов по количеству операций:

Для $LU^{(*)}$ разложения: $\frac{2n^3}{3} + 3n^2 + \frac{7n}{3}$.

Для модифицированного метода Гаусса: $\frac{2n^3}{3} + \frac{7n^2}{2} + \frac{5n}{6}$.

Вывод: Не смотря на то, что $LU^{(*)}$ – разложение имеет меньшие затраты, у мод. Метода Гаусса точность выше (с выбором ведущего элемента).

7. **Текст программы**

Lab_1.cpp:

```
#include <iostream>
#include <fstream>
#include "locale.h"
#include <vector>

using namespace std;

// Здесь меняется точность
typedef float type_data;
typedef float scal;

int n_size; // размерность матрицы
int m_size; // полуширина

bool check(int i, int j)
{
	int sup1 = 1, sup2 = 0;
	while (sup2 < m_size || sup1 < n_size)
	{
		if (i == sup1 && j == sup2)
			return false;
		sup2++;
		sup1++;
	}
	return true;
}

int index_di(int i, int j)
{
	int key = 0;
	int sup1 = 1, sup2 = 0;
	int str1 = 1, str2 = 0;
	for (;;)
	{
		if (sup2 >= m_size || sup1 >= n_size)
			break;
		while (sup2 < m_size || sup1 < n_size)
		{
			if (i == sup1 && j == sup2)
				return key;
			sup1++;
			sup2++;
		}
		key++;
		sup1 = str1 + key;
		sup2 = str2;
	}
}
```

```
void read_func(vector<type_data> &vec_b, vector< vector<type_data> > &Up_m, vector<
vector<type_data> > &L_m, vector<type_data> &di)
{
   // Чтение
   ifstream fcin;
   char* vector_addr = new char[15]{ "b_vector.txt" };
   char* in_au = new char[10]{ "au.txt" };
   char* in_di = new char[10]{ "di.txt" };
   char* in_al = new char[10]{ "al.txt" };

   fcin.open(vector_addr);
   for (int i = 0; i < n_size; i++)
      fcin >> vec_b[i];
   fcin.close();
   delete(vector_addr);

   fcin.open(in_di);
   for (int i = 0; i < n_size; i++)
      fcin >> di[i];
   fcin.close();
   delete(in_di);


   fcin.open(in_au);
   for (int i = 0; i < n_size; i++)
      for (int j = 0; j < m_size; j++)
         fcin >> Up_m[i][j];
   fcin.close();
   delete(in_au);

   fcin.open(in_al);
   for (int i = 0; i < n_size; i++)
      for (int j = 0; j < m_size; j++)
         fcin >> L_m[i][j];
   fcin.close();
   delete(in_al);
   //

}

void LU_decomposition(vector< vector<type_data> >& Up_m, vector< vector<type_data> >& L_m,
vector<type_data>& di)
{
   int add_j = 1, support = 0;
   int j = 0, l = 0;
   scal sup_scal = 0, sup_sc1 = 0, sup_sc2 = 0;

   for (int i = 0; i < n_size; i++)
   {

      for (j = m_size - 1; j >= 0; j--)
      {
         if (check(i, j))
         {
            for (int k = j + 1; k < m_size && i != j && i > j; k++)
            {
               sup_sc1 += L_m[i][k] * Up_m[i - j - 1][k - 1 - j];
               sup_sc2 += Up_m[i][k] * L_m[i - j - 1][k - 1 - j];
            }
            L_m[i][j] = (L_m[i][j] - sup_sc1);
            Up_m[i][j] = Up_m[i][j] - sup_sc2;
            sup_sc1 = 0;
            sup_sc2 = 0;
```

```
            }
            if (i != j && i > j)
               L_m[i][j] = L_m[i][j] / di[index_di(i, j)];

         }

         for (int k = 0; k < m_size; k++)
            sup_scal += L_m[i][k] * Up_m[i][k];
         di[i] = di[i] - sup_scal;
         sup_scal = 0;
      }
   }

   void forward_motion(vector< vector<type_data> > L_m, vector<type_data>& vec_y,
   vector<type_data> vec_b)
   {
      // Решение y (L*y=b) "Прямой ход"
      scal sup_scal = 0;
      for (int i = 0; i < n_size; i++)
         vec_y[i] = vec_b[i];

      int k = 0;

      for (int i = 1; i < n_size; i++) // Так как первый вектор уже найден
      {
         for (int j = i - 1; j >= 0 && k < m_size; j--)
         {
            sup_scal += L_m[i][k] * vec_y[j];
            k++;
         }
         vec_y[i] += - sup_scal;
         sup_scal = 0;
         k = 0;
      }
   }

   void back_motion(vector< vector<type_data> > Up_m, vector<type_data>& vec_x,
   vector<type_data> vec_y, vector<type_data> di)
   {
      // Решение x (U*x=y) "Обратный ход"
      scal sup_scal = 0;
      for (int i = 0; i < n_size; i++)
         vec_x[i] = vec_y[i];
      vec_x[n_size - 1] /= di[n_size - 1];

      int support = 0;

      for (int i = n_size - 2; i >= 0; i--)
      {
         support = 0;
         for (int j = i + 1; j < n_size && support < m_size; j++)
         {
            sup_scal += Up_m[j][support] * vec_x[j];
            support++;
         }
         vec_x[i] = vec_x[i] - sup_scal;
         sup_scal = 0;
         vec_x[i] /= di[i];
      }

   }

   void output_func(vector<type_data> vec_x)
   {
```

```
   char* output = new char[10]{ "out.txt" };
   ofstream fout;
   fout.precision(16);

   fout.open(output);
   for (int i = 0; i < n_size; i++)
      fout << vec_x[i] << endl;
   fout.close();
   delete(output);
}

void init_f()
{
       // Объявление и выделение памяти векторов и матрицы
       vector<type_data> vec_b(n_size);
       vector<type_data> vec_y(n_size);
       vector<type_data> vec_x(n_size);

       //
       // Ориентироваться в матрице никак (x, y), а как запись индексации в обычных матрицах
       vector< vector<type_data> > Up_m(n_size, vector<type_data>(m_size));
       vector< vector<type_data> > L_m(n_size, vector<type_data>(m_size));
       vector<type_data> di(n_size);
       //

   read_func(vec_b, Up_m, L_m, di);
   LU_decomposition(Up_m, L_m, di);
   forward_motion(L_m, vec_y, vec_b);
   back_motion(Up_m, vec_x, vec_y, di);
   output_func(vec_x);
}

int main()
{
       setlocale(LC_ALL, "rus");
       char* info_addr = new char[10]{ "info.txt" }; // Запись в виде: размерность матрицы,
ширина ленты
       ifstream fcin_1(info_addr);
       fcin_1 >> n_size >> m_size;
       fcin_1.close();
       init_f();
}
```

Method_gauss.cpp:

```
#include <iostream>
#include <fstream>
#include "locale.h"
#include <vector>

using namespace std;

// Здесь меняется точность
typedef float type_data;

int n_size; // размерность матрицы

void output_func(vector<type_data> vec_x)
{
   char* address = new char[10]{ "out.txt" };
   ofstream fout;

   fout.open(address);
```

```
   for (int i = 0; i < n_size; i++)
      fout << vec_x[i] << endl;
}

void method_gauss(vector< vector<type_data> >& Matr)
{
   vector<type_data> vec_x(n_size);
   double e = 0.0000000000000001;
   int flag = 1;

   for (int k = 0; k < n_size; k++)
   {

      type_data max = abs(Matr[k][k]);
      int imax = k;
      for (int j = k; j < n_size; j++)
      {
         if (abs(Matr[j][k]) > max)
         {
            max = (abs(Matr[j][k]));
            imax = j;
         }
      }
      if (max < e)
      {
         cout << "Система не имеет решения";
         flag = 0;
      }
      else
         if (imax != k)
         {
            type_data g;
            g = Matr[k][n_size];
            Matr[k][n_size] = Matr[imax][n_size];
            Matr[imax][n_size] = g;

            for (int r = k; r < n_size; r++)
            {
               g = Matr[k][r];
               Matr[k][r] = Matr[imax][r];
               Matr[imax][r] = g;
            }
         }

      for (int i = k + 1; i < n_size; i++)
      {
         type_data t = Matr[i][k] / Matr[k][k];
         Matr[i][n_size] -= t * Matr[k][n_size];

         for (int j = k + 1; j < n_size; j++)
         {
            Matr[i][j] -= t * Matr[k][j];
         }
      }
   }

   int n_sup = n_size - 1;
   vec_x[n_sup] = Matr[n_sup][n_size] / Matr[n_sup][n_sup];
   for (int k = n_size - 2; k >= 0; k--)
   {
      type_data sum = 0;
      for (int i = k + 1; i < n_size; i++)
         sum += Matr[k][i] * vec_x[i];
      vec_x[k] = (Matr[k][n_size] - sum) / Matr[k][k];
```

```
    }

    if (flag)
        output_func(vec_x);
}

void read_func(vector< vector<type_data> >& Matr)
{
    // Чтение
    ifstream fcin;
    char* vector_addr = new char[15]{ "b_vector.txt" };
    char* in_m = new char[15]{ "Matrix.txt" };

    fcin.open(vector_addr);
    for (int i = 0; i < n_size; i++)
        fcin >> Matr[i][n_size];
    fcin.close();
    delete(vector_addr);

    fcin.open(in_m);
    for (int i = 0; i < n_size; i++)
        for (int j = 0; j < n_size; j++)
            fcin >> Matr[i][j];
    fcin.close();
    delete(in_m);
    //

}

int main()
{
    setlocale(LC_ALL, "rus");
    char* info_addr = new char[10]{ "info.txt" }; // Запись в виде: размерность матрицы,
ширина ленты
    ifstream fcin_1(info_addr);

    fcin_1 >> n_size;
    fcin_1.close();
    delete(info_addr);

    vector< vector<type_data> > Matr(n_size, vector<type_data>(n_size + 1));

    read_func(Matr);
    method_gauss(Matr);
}
```