

Санкт-Петербургский Государственный Политехнический Университет
Факультет Технической Кибернетики
Кафедра Информационных и Управляющих Систем

ОТЧЕТ
о лабораторной работе № 3
«Линейная алгебра.»
по численному анализу

Работу выполнил студент:

Голубева А. С.
гр. 3084/1

Преподаватель:

Зимницкий В. А.

Постановка задачи.

Совокупность матриц A , зависящих от коэффициента a_n , задается следующим образом:

$$A = \begin{pmatrix} 1 & 1 & \dots & 1 \\ a_1 & a_2 & \dots & a_n \\ \dots & \dots & \dots & \dots \\ a_1^{n-1} & a_2^{n-1} & \dots & a_n^{n-1} \end{pmatrix}$$

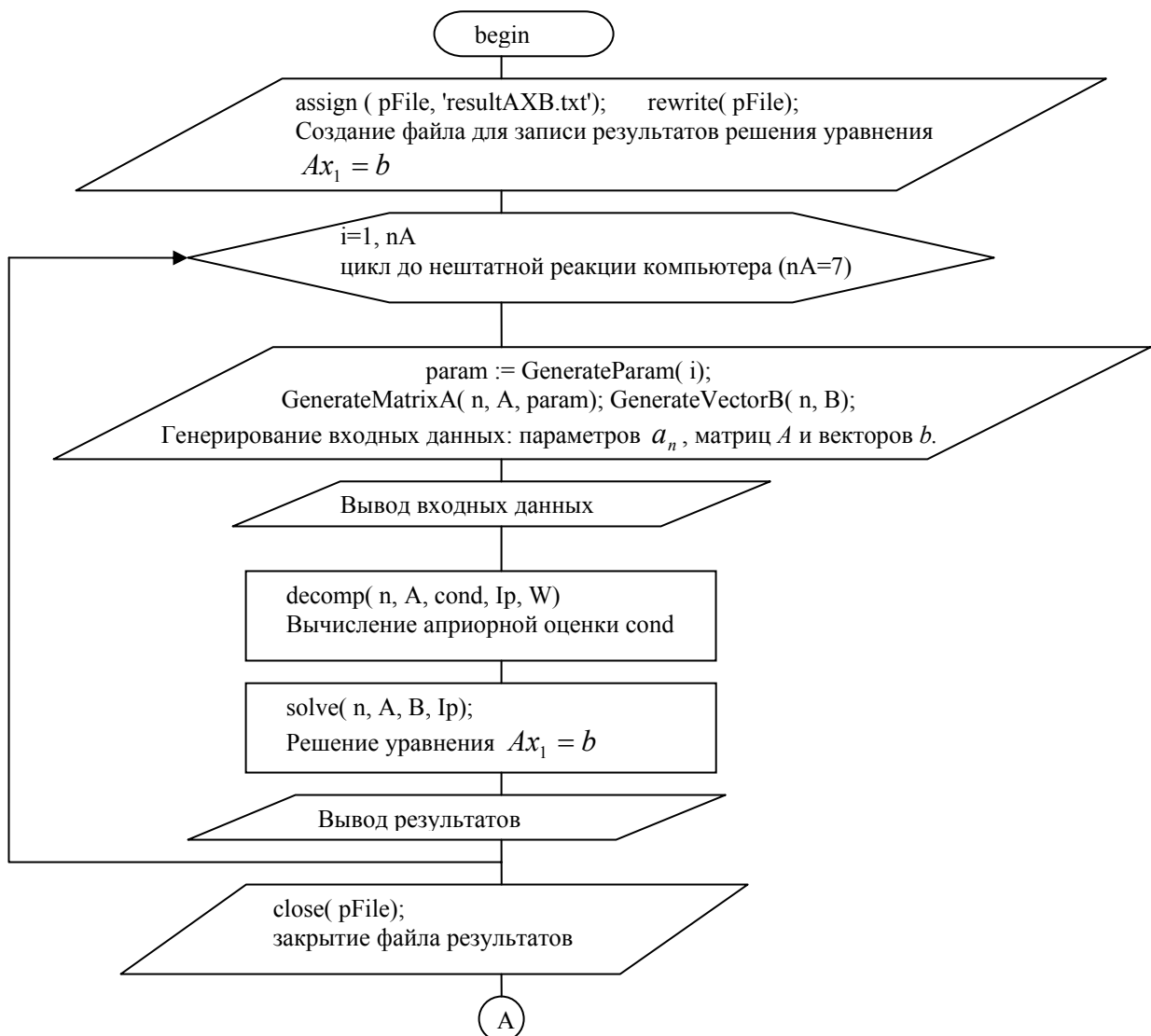
Решить линейную систему, используя программы DECOMP и SOLVE при $b_i = 2^i + \cos(i)$. Коэффициенты a_i выглядят так: $a_1 = 1$, $a_2 = 2$, $a_3 = 3, \dots, a_{n-1} = n-1$, $a_n = 1.1, 1.01, 1.001$ и т.д. вплоть до нештатной реакции компьютера. $n = 6$.

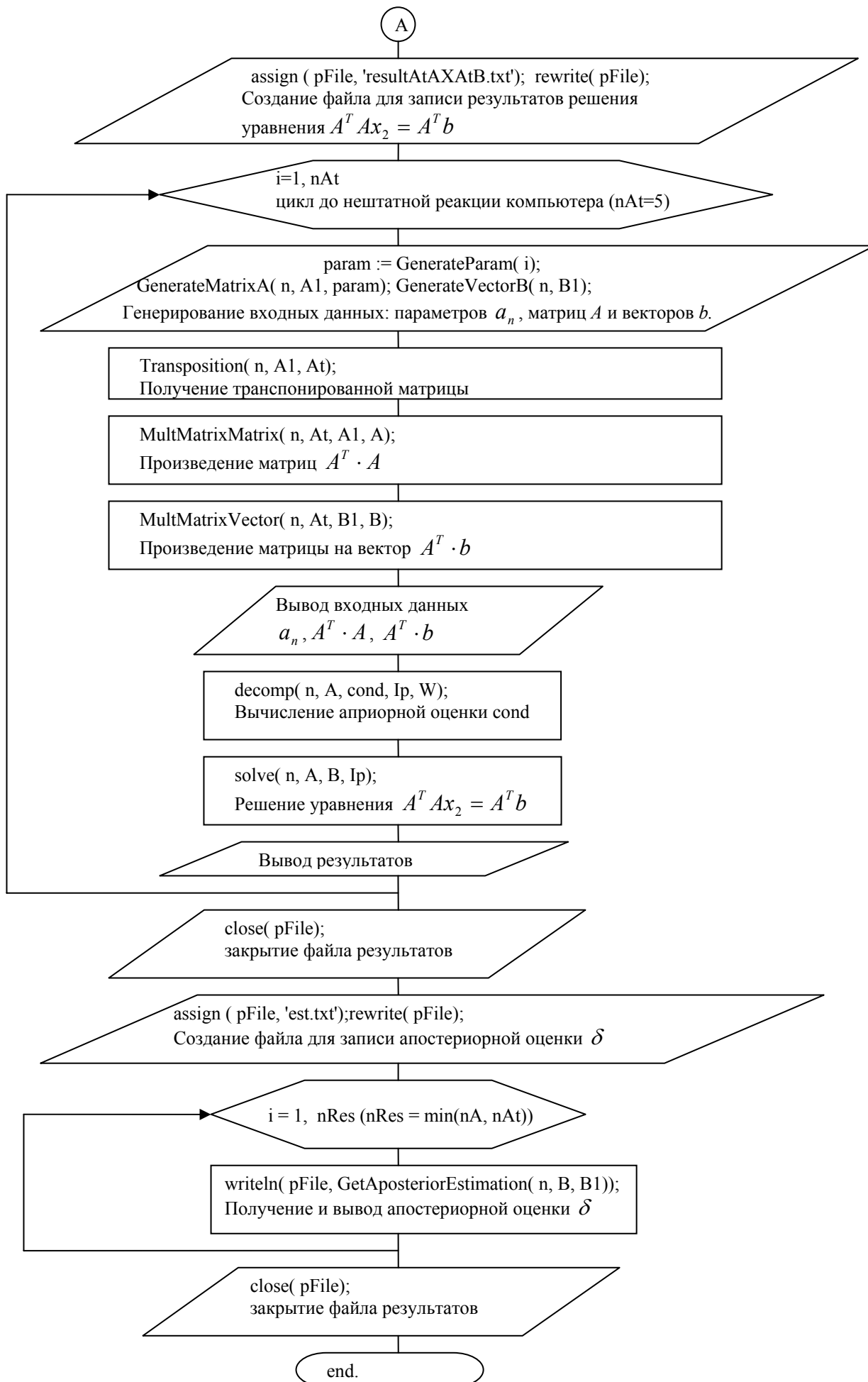
Сравнить решение системы $Ax_1 = b$ с решением системы $A^T Ax_2 = A^T b$, полученной из исходной левой трансформацией Гаусса.

Априорная оценка качества заданных матриц представляется стандартным числом обусловленности из программы DECOMP, а апостериорная $\delta = \|x_1 - x_2\| / \|x_1\|$ вычисляется в ходе

эксперимента $\|\cdot\| = \sqrt{\sum_{i=1}^n x_i^2}$.

1. Блок-схема алгоритма главной программы.





2. Исходный текст программы на Паскале (lab3.pas).

```
program Lab3;
{ Use modules}
uses FMM, printer, crt;

procedure GenerateMatrixA( n: Integer; var A: floatmatrix; param: float);
var i, j : Integer;
begin
  for i := 1 to n do
    begin
      for j := 1 to n - 1 do A[ i, j] := powri( j, i - 1);
      A[ i, n] := powri( param, i - 1);
    end;
  end;

procedure GenerateVectorB( n: Integer; var B: floatvector);
var i: Integer;
begin
  for i := 1 to n do B[ i] := powri( 2, i) + cos( i);
end;

function GenerateParam( i: Integer): float;
begin
  GenerateParam := 1.0 + 1.0 / powri( 10.0, i);
end;

function GetAposteriorEstimation( n: Integer; X1, X2: floatvector): float;
var
  i      : Integer;
  t1, t2: float;
begin
  t1 := 0; t2 := 0;
  for i := 1 to n do
    begin
      t1 := t1 + powri( X1[ i] - X2[ i], 2);
      t2 := t2 + powri( X1[ i], 2);
    end;
  GetAposteriorEstimation := sqrt( t1 / t2);
end;

procedure MultMatrixMatrix( n: Integer; A, B: floatmatrix; var Res:
floatmatrix);
var
  i, j, k: Integer;
begin
  for i := 1 to n do
    begin
      for j := 1 to n do
        begin
          Res[ i, j] := 0.0;
          for k := 1 to n do Res[ i, j] := Res[ i, j] + A[ i, k] * B[ k, j];
        end;
      end;
    end;
end;

procedure MultMatrixVector( n: Integer; A: floatmatrix; B: floatvector; var
Res:
```

```

floatvector);
var
  i, j: Integer;
begin
  for i := 1 to n do
    begin
      Res[ i] := 0.0;
      for j := 1 to n do Res[ i] := Res[ i] + A[ i, j] * B[ j];
    end;
  end;

procedure PrintMatrix( n: Integer; A: floatmatrix; var pFile: TEXT);
var
  i, j: Integer;
begin
  for i := 1 to n do
    begin
      for j := 1 to n do write( pFile, A[ i, j]:20:7);
      writeln( pFile);
    end;
  end;

procedure PrintVector( n: Integer; A: floatvector; var pFile: TEXT);
var
  i: Integer;
begin
  for i := 1 to n do writeln( pFile, A[ i]:20:7);
end;

procedure Transposition( n: Integer; A: floatmatrix; var At: floatmatrix);
var
  i, j: Integer;
begin
  for i:= 1 to n do
    for j := 1 to n do At[ i, j] := A[ j, i];
  end;

{ Constants}
const n      = 6; { Dimension of matrix A and B}
      nA     = 7; { Number of matrix A}
      nAt    = 5;

Var
  A, A1, At : floatmatrix;
  B1, W, B   : floatvector;
  cond       : float;
  Ip         : ivector;
  param      : float;
  i, j, nRes: Integer;
  pFile      : TEXT;
  ResA       : array [1..nA, 1..n] of float;
  ResAt      : array [1..nAt, 1..n] of float;
Begin
  { Clear screen}
  clrscr;
  { Open file to write result data for A*X=B equation}
  assign ( pFile, 'resultAXB.txt');
  rewrite( pFile);

```

```

for i := 1 to nA do
begin
  { Get param for matrix A}
  param := GenerateParam( i);
  { Generate initial data}
  GenerateMatrixA( n, A, param);
  GenerateVectorB( n, B);
  { Print initial data "param, A, B"}
  write( pFile, '-----'); writeln(
pFile);
  write( pFile, 'Parameter an = '); writeln( pFile, param);
  writeln( pFile, 'Matrix A:'); PrintMatrix( n, A, pFile);
  writeln( pFile, 'Matrix B:'); PrintVector( n, B, pFile);
  { Calculate solution of equation A*X=B}
  decomp( n, A, cond, Ip, W); { Calculate gaussian matrix}
  solve( n, A, B, Ip);       { Calculate solution, result in B}
  { Print result of solution A*X=B}
  writeln( pFile, 'Result A*X=B:'); PrintVector( n, B, pFile);
  for j:= 1 to n do ResA[ i, j] := B[ j];
  { Aprior estimation of quality}
  write( pFile, 'Aprior estimation (cond) = '); writeln( pFile, cond);
end;
close( pFile);
{Open file to write result data for At*A*X=At*B equation}
assign ( pFile, 'resultAtAXAtB.txt');
rewrite( pFile);
for i := 1 to nAt do
begin
  { Get param for matrix A}
  param := GenerateParam( i);
  { Generate initial data}
  GenerateMatrixA( n, A1, param);
  GenerateVectorB( n, B1);
  Transposition( n, A1, At);
  MultMatrixMatrix( n, At, A1, A); { Get At*A}
  MultMatrixVector( n, At, B1, B); { Get At*B}
  { Print initial data "param, A, B"}
  write( pFile, '-----'); writeln(
pFile);
  write( pFile, 'Parameter an = '); writeln( pFile, param);
  writeln( pFile, 'Matrix At*A:'); PrintMatrix( n, A, pFile);
  writeln( pFile, 'Matrix At*B:'); PrintVector( n, B, pFile);
  { Calculate solution of equation At*A*X=At*B}
  decomp( n, A, cond, Ip, W); { Calculate gaussian matrix}
  solve( n, A, B, Ip);       { Calculate solution, result in B}
  { Print result of solution At*A*X=At*B}
  writeln( pFile, 'Result At*A*X=At*B:'); PrintVector( n, B, pFile);
  for j:= 1 to n do ResAt[ i, j] := B[ j];
  { Aprior estimation of quality}
  write( pFile, 'Aprior estimation (cond) = '); writeln( pFile, cond);
end;
close( pFile);

assign ( pFile, 'est.txt');
rewrite( pFile);
writeln( pFile, 'Aposterior estimation:');
if nA > nAt then nRes := nAt
else nRes := nA;

```

```

for i := 1 to nRes do
begin
  for j:= 1 to n do B [ j] := ResA[i, j];
  for j:= 1 to n do B1[ j] := ResAt[i, j];
  writeln( pFile, GetAposteriorEstimation( n, B, B1));
end;
close( pFile);
end.

```

3. Полученные результаты.

- Пример формирования матрицы A и вектора b (при $a_n = 1.001$):

Матрица A :

1.0000000	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
1.0000000	2.0000000	3.0000000	4.0000000	5.0000000	1.0010000
1.0000000	4.0000000	9.0000000	16.0000000	25.0000000	1.0020010
1.0000000	8.0000000	27.0000000	64.0000000	125.0000000	1.0030030
1.0000000	16.0000000	81.0000000	256.0000000	625.0000000	1.0040060
1.0000000	32.0000000	243.0000000	1024.0000000	3125.0000001	1.0050100

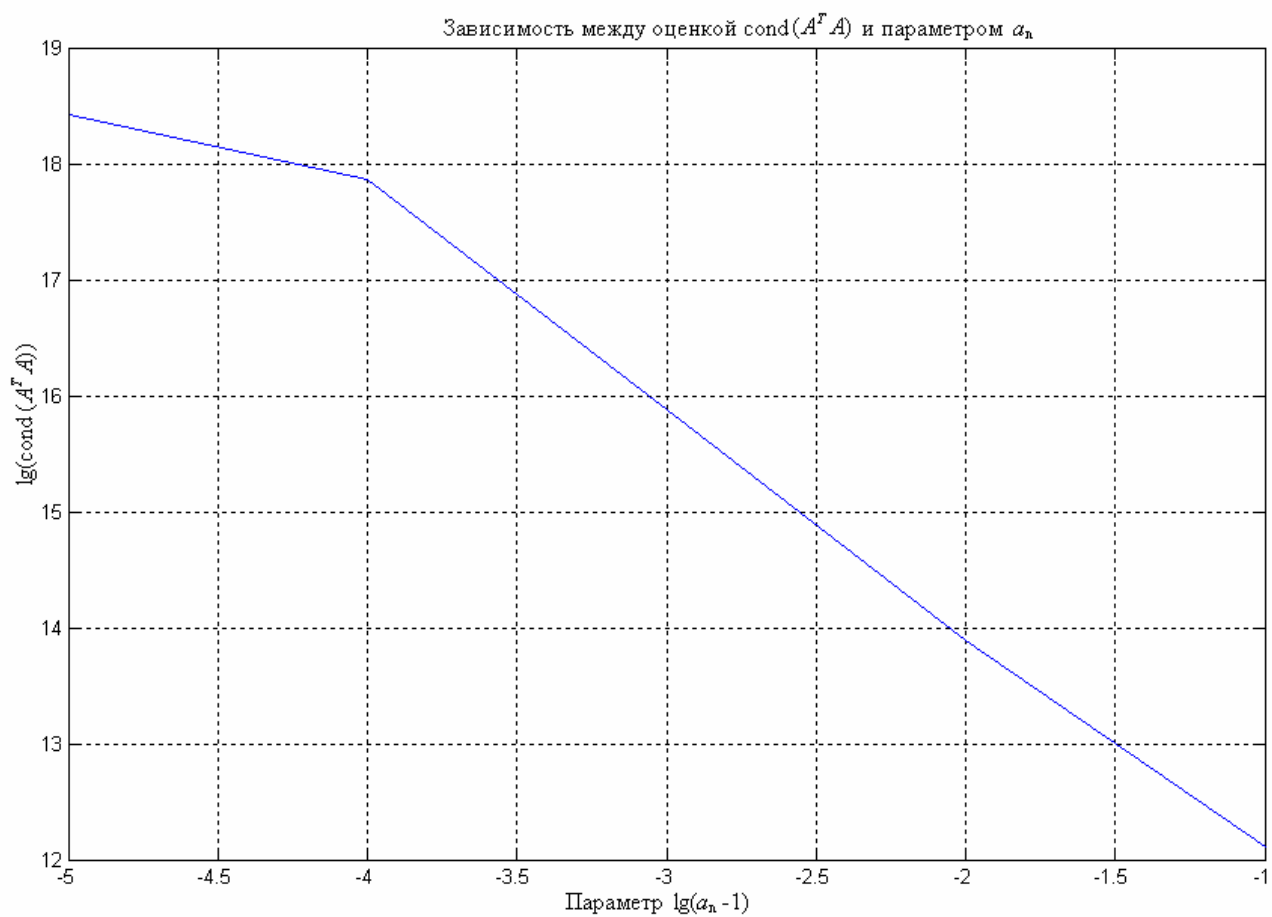
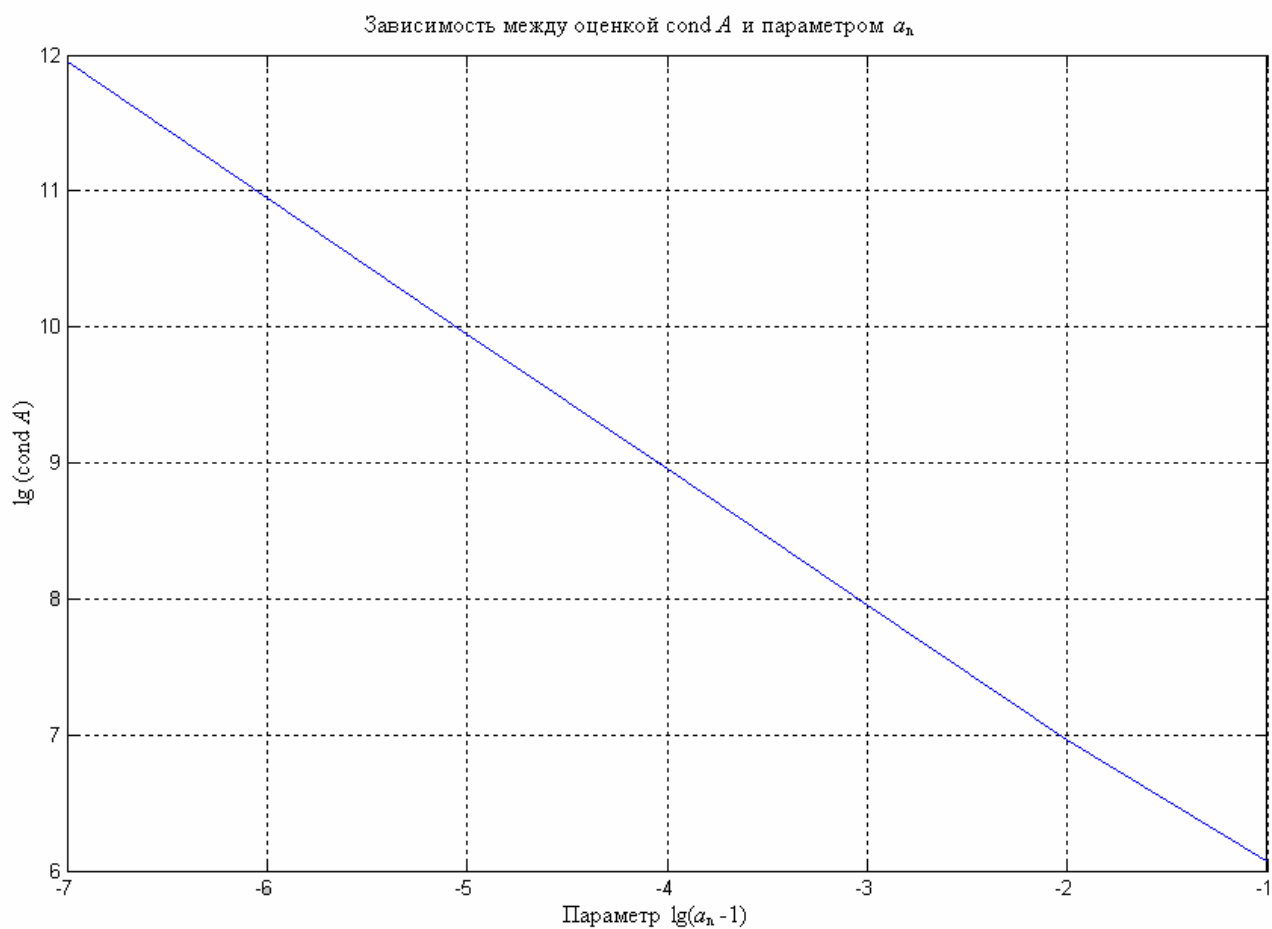
Вектор b :

2.5403023
 3.5838532
 7.0100075
 15.3463564
 32.2836622
 64.9601703

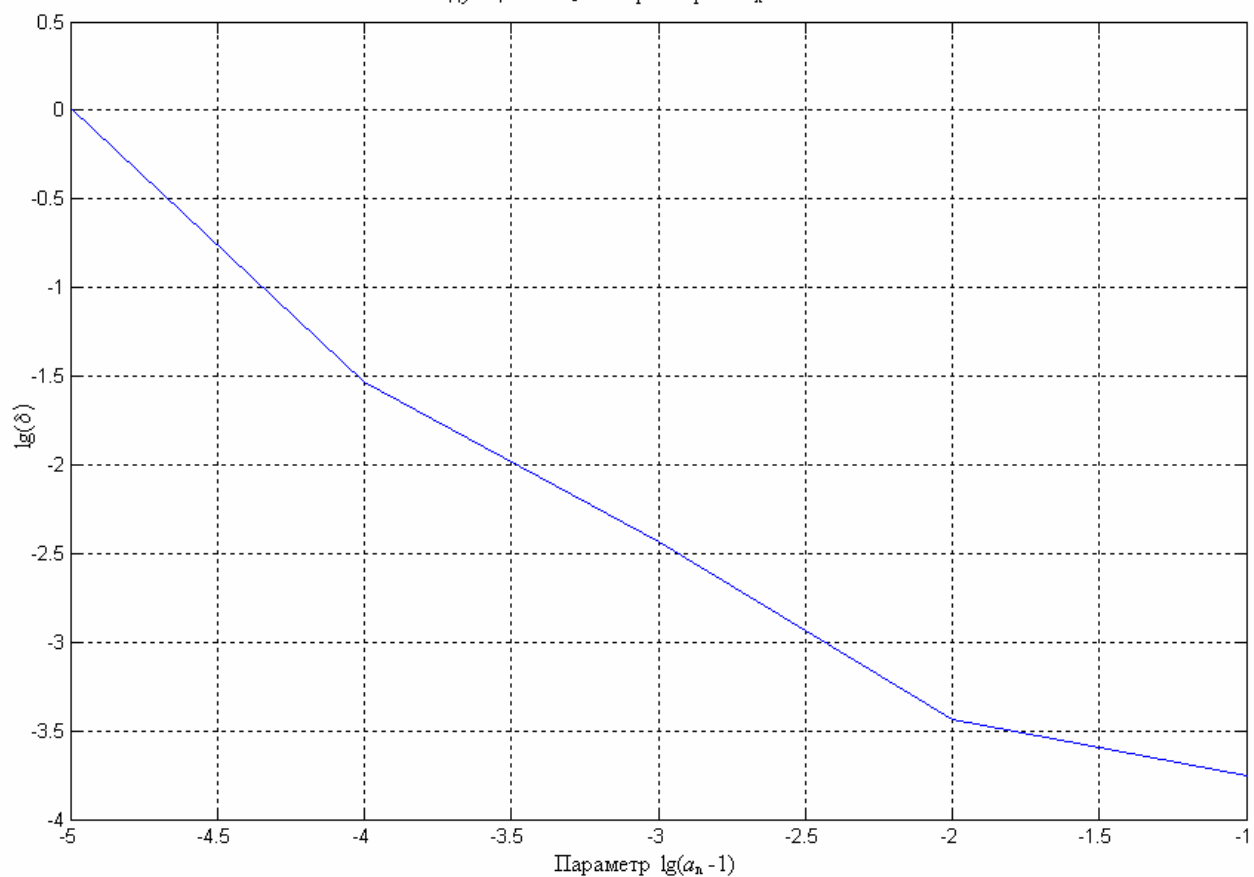
- Сравнительная таблица полученных результатов:

a_n	1.1	1.01	1.001	1.0001	1.00001	1.000001	1.0000001
x_1	9.0722825	65.1972893	626.4473483	6238.9481583	62363.9456990	623608.5900700	6235045.1983000
	0.2187183	-0.0332458	-0.0559453	-0.0581927	-0.0584173	-0.0584513	-0.0586691
	1.4530262	1.5420900	1.5505553	1.5513976	1.5514818	1.5514967	1.5516187
	-0.5528593	-0.5787502	-0.5812538	-0.5815033	-0.5815283	-0.5815331	-0.5815764
	0.0878063	0.0914131	0.0917648	0.0917999	0.0918034	0.0918041	0.0918109
	-7.7386717	-63.6784940	-624.9121670	-6237.4113574	-62362.4087360	-623607.0530900	-6235043.6612000
x_2	9.0736193	65.2206561	628.7580002	6055.7265071	-2039.4085598	-	-
	0.2191200	-0.0326186	-0.0498798	-0.1062196	-1.7455957		
	1.4528131	1.5417434	1.5472090	1.5778989	2.4822787		
	-0.5527853	-0.5786284	-0.5800804	-0.5907970	-0.9079034		
	0.0877949	0.0913942	0.0915833	0.0932377	0.1422912		
	-7.7402581	-63.7022405	-627.2264854	-6054.1606810	2041.9652145		
cond A	1.17E+06	9.17E+06	8.94E+07	8.92E+08	8.91E+09	8.91E+10	8.91E+11
cond($A^T A$)	1.29E+12	7.84E+13	7.47E+15	7.27E+17	2.61E+18	-	-
δ	1.76E-04	3.65E-04	3.69E-03	2.93E-02	1.03	-	-

- *Графики зависимостей:*



Зависимость между оценкой δ и параметром a_n



- Сравнительный график:

