Ασκήσεις Αριθμητικής Γραμμικής Αλγεβρας.

Νίκος Μαυρογιαννόπουλος (6205)

18 Δεκεμβρίου 2001

Περιεχόμενα

1 Γενικά

Για την υλοποίηση των τριών εργασιών χρησιμοποιήθηκε η γλώσσα C++. Ενω για τις ανάγκες των εργασιών δημιουργήθηκαν και χρησιμοποιήθηκαν δυο κοινές κλάσεις. Είναι οι

- matrix: Ορίζει τους πίνακες $(n*n)$, και καθορίζει τις πράξεις μεταξύ πινάκων
- vector: Ορίζει τα διανύσματα, και καθορίζει τις πράξεις μεταξύ αριθμών και διανυσμάτων, όπως και το γινόμενο πίνακα επί διανυσμα.

Τα προγράμματα εκτελέστηκαν σε υπολογιστή με χαρακτηριστικά:

- i686 me 32 bit floating point.
- GNU C++ compiler

Επίσης δοκιμάστηκαν και στον socrates.cc.uoi.gr με τον native CC compiler.

2 Μέθοδος του Gauss

2η Εργαστηριακή Ασκηση στην Αριθμητική γραμμική Αλγεβρα

2.1 Ερώτημα II

1. Με την μέθοδο Crammer (σε υπολογιστή) βρίσκεται ότι η λύση είναι ένα διανυσμα στο R^{50} με 1 παντού.

2. Ακριβέστερα αποτελέσματα για τον πίνακα II δίνει η μέθοδος Gauss, με μερική οδήγηση.

Η λύση που έδωσε η μέθοδος Gauss χωρίς οδήγηση είναι:

```
[1 1 1 1 1 0.999999 1 0.999997 1.00001 0.999989
 1.00002 0.999954 1.00009 0.999817 1.00037 0.999268
 1.00146 0.99707 1.00586 0.988281 1.02344 0.953125 1.09375
 0.8125 1.375 0.25 2.5 -2 7 -11 25 -46.9999 96.9996 -190.999
 384.994 -766.977 1536.91 -3070.62 6143.5 -12281 24553 -49055
 97921 -195071 387073 -761855 1.47456e+06 -2.75251e+06 4.71859e+06
 -6.29146e+06
]
```

Ενω η λύση με τη μεθοδο Gauss με μερική οδήγηση είναι:

```
[ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 0.999998 1 0.999992 1.00002 0.999969 1.00006 0.999878
  1.00024 0.999512 1.00098 0.998047 1.00391 0.992188 1.01562 0.968765
  1.06244 0.875244 1.24902 0.503906 1.98438 -0.9375 4.75 -6 13 -15
]
```

Η μεγαλύτερη απόκλιση της δευτερης μεθόδου είναι -16, ενω της πρώτης είναι της τάξης $-6 * 10^6$.

Η μεθοδος με μερική οδήγηση είναι πιο ακριβής αφού χρησιμοποιεί οσο το δυνατόν μεγαλύτερο οδηγό και άρα το 'mjk' είναι μικρότερο. Αυτό έχει ώς αποτέλεσμα στον παρακάτω κώδικα το λάθος να μην επεκτείνεται όσο στην πρώτη μέθοδο. Αυτό συμβαίνει αφου το 'mjk' είναι πολλαπλασιαστής και όσο μικρότερος είναι τόσο μικρότερη είναι εξάπλωση του λάθους.

```
  for (j = k + 1; j < n; j++) {
   // m[j,k] = a[j,k]/a[k,k];
   mjk = val = A.get(j, k) / A.get(k, k);
   A.set( j, k, 0); // This is required!!

   for (l = k + 1; l < n; l++) {
    val = A.get( j, l) - mjk * A.get( k, l);

    A.set( j, l, val);

   }
   val = B.get( j) - mjk * B.get( k);
   B.set( j, val);
  }
```

2.2 gauss.cpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// To arqe'io peri'eqei sunart'hseic gia ton upologism'o tou sust'hmatoc
// pin'akwn Ax=B, me th m'ejodo tou Gauss (me merik'h od'hghsh kai qwr'ic).

#include <common.hpp>
#include <vector.hpp>
#include <matrix.hpp>
```

```
#include <crammer.hpp>

// Edw epil'egetai o alg'orijmoc ( merik'h od'hghsh 'h apl'oc).

inline void
alg_choice (const uint k, const uint n, matrix & A, vector & B,
            const uint type)
{
   uint j;

// ean type==0 t'ote e'inai o apl'oc alg'orijmoc
// ean type!=0 qrhsimopoie'itai merik'h od'hghsh.

   switch (type)
     {
     case 0:
        for (j = k; j < n; j++)
          {
             if (A.get (j, k) != 0)
               {
                  A.swaplines (j, k);
                  B.swaplines (j, k);
                  break;
               }
          }
        break;
     default:
        {                       // merik'h od'hghsh
           real max, tmp;
           uint max_line;       // krat'aei thn gramm'h tou m'egistou

           max = A.get (k, k);
           max_line = k;
           for (j = k; j < n; j++)
             {
                // br'iskei to megal'utero stoiqe'io sthn st'hlh
                tmp = A.get (j, k);
                if (tmp > max)
                  {
                     max = tmp;
                     max_line = j;
                  }
             }
           // enallag'h gramm'wn
           A.swaplines (max_line, k);
           B.swaplines (max_line, k);
```

```
        }
     }
}

// ean type==0 t'ote e'inai o apl'oc alg'orijmoc
// ean type!=0 qrhsimopoie'itai merik'h od'hghsh.
// Warning: matrix A and vector B are modified.
vector *
gauss_algorithm (matrix & A, vector & B, uint type)
{
   int n = A.getn ();           // n*n
   int k, j, l;
   real val, s;
   vector *x = new vector (n);
   real mjk;

   /* Eliminate Step */
   for (k = 0; k < n - 1; k++)
     {
        alg_choice (k, n, A, B, type);

        for (j = k + 1; j < n; j++)
          {
             // m[j,k] = a[j,k]/a[k,k];
             mjk = val = A.get (j, k) / A.get (k, k);
             A.set (j, k, 0);   // This is required!!

             for (l = k + 1; l < n; l++)
               {
                  val = A.get (j, l) - mjk * A.get (k, l);

                  A.set (j, l, val);

               }
             val = B.get (j) - mjk * B.get (k);
             B.set (j, val);
          }
     }


   /* Back Substitute step */
   x->set (n - 1, B.get (n - 1) / A.get (n - 1, n - 1));
   for (k = n - 1; k >= 0; k--)
     {
        s = B.get (k);
        for (j = k + 1; j < n; j++)
```

```
          {
             // no -= since it must be defined
             s = s - A.get (k, j) * x->get (j);
          }
        x->set (k, s / A.get (k, k));
     }
   return x;
}


#define DIM 50
#define SET_A_VALUES \
for (i=0;i<DIM;i++) \
for (j=0;j<DIM;j++) { \
A.set(i, j, 0); \
if (i==j) A.set( i, j, 6); \
if (i==(j-1)) A.set( i, j, 1); \
if (i==(j+1)) A.set( i, j, 8); \
}

#define SET_B_VALUES \
for (i=0;i<DIM;i++) { \
sum = 0; \
for (j=0;j<DIM;j++) { \
sum = sum + A.get( i, j); \
} \
B.set( i, sum); \
}

void
test_gauss ()
{

   matrix A (DIM);
   vector B (DIM);
   vector *x;
   uint i, j;
   real sum;

   // Put the matrix values
   SET_A_VALUES;

   // put the vector values
   SET_B_VALUES;

   cout << "A: " << endl;
```

```
   cout << A << endl << endl;

   cout << "B: " << endl;
   cout << B << endl;
   cout << endl;


   cout << endl;
   x = gauss_algorithm (A, B, 0);
   cout << "L'ush me Gauss [apl'oc]: \n[" << *x << "]\n";
   delete x;

   SET_A_VALUES;
   SET_B_VALUES;

   cout << endl;
   x = gauss_algorithm (A, B, 1);
   cout << "L'ush me Gauss [merik'h od'hghsh]: \n[" << *x << "]\n";
   delete x;

   SET_A_VALUES;
   SET_B_VALUES;

   cout << endl;
   cout << "L'ush me Crammer: \n";
   x = crammer_algorithm (A, B);
   cout << "[" << *x << "]\n";
   delete x;

}

#define SET_A_VALUES2 \
A.set( 0, 0, 0.0001); \
A.set( 0, 1, 1); \
A.set( 1, 0, 1); \
A.set( 1, 1, 1)

#define SET_B_VALUES2 \
B.set( 0, 1); \
B.set( 1, 2)

#define DIM2 2
void
test_gauss2 ()
{
```

```
    matrix A (DIM2);
    vector B (DIM2);
    vector *x;

    // Put the matrix and vector values
    SET_A_VALUES2;
    SET_B_VALUES2;

    cout << "A: " << endl;
    cout << A << endl << endl;

    cout << "B: " << endl;
    cout << B << endl;
    cout << endl;

    // Gauss
    cout << endl;
    x = gauss_algorithm (A, B, 0);
    cout << "L'ush me Gauss [apl'oc]: \n[" << *x << "]\n";
    delete x;


    // Gauss me merik'h od'hghsh
    SET_A_VALUES2;
    SET_B_VALUES2;

    cout << endl;
    x = gauss_algorithm (A, B, 1);
    cout << "L'ush me Gauss [merik'h od'hghsh]: \n[" << *x << "]\n";
    delete x;

    // Crammer
    SET_A_VALUES2;
    SET_B_VALUES2;

    cout << endl;
    cout << "L'ush me Crammer: \n";
    x = crammer_algorithm (A, B);
    cout << "[" << *x << "]\n";
    delete x;

}
```

2.3 gauss.hpp

```
#include <vector.hpp>
#include <matrix.hpp>

void test_gauss ();
void test_gauss2 ();
vector *gauss_algorithm (matrix & A, vector & B, uint type);
```

3 Μέθοδος SOR

3η Εργαστηριακή Ασκηση στην Αριθμητική γραμμική Αλγεβρα

3.1 Ερώτημα II

Οι λύσεις που δίνονται απο το πρόγραμμα είναι:

```
Λύση με ω=0.9
Επαναλήψεις: 17
X: [0.618034 0.854102 0.944272 0.978714 0.99187 0.996894 0.998813 0.999545
    0.99982 0.999912 0.999912 0.999821 0.999547 0.998816 0.996897 0.991872
    0.978715 0.944273 0.854102 0.618034 ]

Λύση με ω=1
Επαναλήψεις: 15
X: [0.618034 0.854102 0.944272 0.978714 0.991869 0.996894 0.998813 0.999545
    0.999819 0.999911 0.999911 0.999819 0.999545 0.998814 0.996895 0.99187
    0.978714 0.944272 0.854102 0.618034 ]

Λύση με ω=1.1
Επαναλήψεις: 13
X: [0.618034 0.854102 0.944272 0.978714 0.991869 0.996894 0.998814 0.999545
    0.999819 0.99991 0.999909 0.999818 0.999543 0.998812 0.996894 0.991869
    0.978714 0.944272 0.854102 0.618034 ]

Λύση με ω=1.2
Επαναλήψεις: 13
X: [0.618034 0.854102 0.944272 0.978714 0.991869 0.996894 0.998814 0.99954
    0.999818 0.999909 0.999909 0.999817 0.999543 0.998812 0.996894 0.991869
    0.978714 0.944272 0.854102 0.618034 ]

Λύση με ω=1.3
Επαναλήψεις: 17
X: [0.618034 0.854102 0.944272 0.978713 0.991871 0.996893 0.998813 0.999543
    0.999817 0.999909 0.999909 0.999817 0.999543 0.998812 0.996894 0.991869
    0.978714 0.944272 0.854102 0.618034 ]

Λύση με ω=1.4
Επαναλήψεις: 21
X: [0.618035 0.8541 0.944274 0.978713 0.991869 0.996894 0.998812 0.999543
    0.999817 0.999909 0.999909 0.999817 0.999543 0.998812 0.996894 0.991869
    0.978714 0.944272 0.854102 0.618034 ]
```

```
Λύση με ω=1.5
Επαναλήψεις: 25
X: [0.618041 0.854103 0.944272 0.978715 0.99187 0.996894 0.998812 0.999543
    0.999817 0.999909 0.999909 0.999817 0.999543 0.998812 0.996894 0.991869
    0.978714 0.944272 0.854102 0.618034 ]

Λύση με ω=1.6
Επαναλήψεις: 27
X: [0.618059 0.854131 0.944282 0.978719 0.991878 0.996902 0.998817 0.999545
    0.999819 0.99991 0.99991 0.999818 0.999544 0.998813 0.996894 0.991869
    0.978714 0.944272 0.854102 0.618034 ]

Λύση με ω=1.7
Επαναλήψεις: 27
X: [0.618509 0.854528 0.944394 0.978824 0.992068 0.997042 0.998874 0.999572
    0.999853 0.999951 0.999941 0.999833 0.999548 0.998816 0.996899 0.991875
    0.978717 0.944274 0.854103 0.618035 ]

Λύση με ω=1.8
Επαναλήψεις: 29
X: [0.617217 0.855251 0.946349 0.979505 0.992551 0.998081 1.00016 1.00071
    1.00062 1.00046 1.00044 1.0004 1.00006 0.999159 0.997111 0.992041 0.978877
    0.944417 0.854202        0.61813 ]

Λύση με ω=1.9
Επαναλήψεις: 29
X: [0.61418 0.867575 0.972191 0.983575 1.00345 1.0156 1.0163 1.01718 1.01363
    1.0081 1.00834 1.0112 1.0097 1.00458 1.00029 0.995989
    0.983739 0.948568 0.856604 0.620341 ]

Λύση με ω=1.142
Επαναλήψεις: 13
X: [0.618034 0.854102 0.944272 0.978714 0.991869 0.996894 0.998814 0.999543
    0.999817 0.999909 0.999909 0.999817 0.999543 0.998812 0.996894 0.991869
    0.978714 0.944272 0.854102 0.618034 ]
```

Ενω λύνοντας το σύστημα με Crammer, βρίσκουμε τη λύση:

```
X: [0.618034 0.854102 0.944272 0.978714 0.991869 0.996894 0.998812 0.999543
    0.999817 0.999909 0.999909 0.999817 0.999543 0.998812 0.996894 0.991869
    0.978714 0.944272 0.854102 0.618034]
```

Ετσι λύσεις πιο κοντά στις πραγματικές με τις λιγότερες ανακυκλώσεις δίνονται για ω στο διάστημα $[1, 1.2]$. Η Gauss-Seidel ($w = 1$) δίνει τη λύση με 15 ανακυκλώσεις

ενώ η SOR (με $w = 1.142$) δίνει την ίδια λύση με 13 ανακυκλώσεις. Αρα η SOR για τον συγκεκριμένο πίνακα είναι προτιμότερη.

3.2 epan.cpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// To arqe'io (maz'i me to epan.hpp), peri'eqoun sunart'hseic koin'ec gia tic
// epanal'hpthk'ec mej'odouc ('opwc SOR). Sunart'hseic 'opwc gia upologism'o m'eshc
// taq'uthtac, di'aspashc tou p'inaka klp.

#include <common.hpp>
#include <epan.hpp>
#include <sor.hpp>
#include <math.h>               // for log()
#include <crammer.hpp>
#include <gauss.hpp>

#ifdef USE_REAL
# define VAL(x) x.value()
#else
# define VAL(x) x
#endif

// upolog'izei thn taq'uthta s'ugklishc qrhsimopoi'wntac
// 'ena p'inaka T kai tic epanal'hyeic (rot)
real
calc_speed (const matrix & T, uint rot)
{
   uint i;
   matrix A (T);
   real norm;

   for (i = 0; i < rot; i++)
     {
        A = A * T;              // or A *= T;
     }
   // now A==A^k

   // R(A^k) = -ln||A^k||/k
   norm = A.norm1 ();
   return (real) ((-1) * log (VAL (norm)) / rot);
```

```
}

// Spaei ton p'inaka A, ston diag'wnio D, ston k'atw trigwnik'o L kai ston 'anw
// trigwnik'o U.
void
matrix_break (const matrix & A, matrix & _D, matrix & _L, matrix & _U)
{
   uint n = A.getn ();
   matrix D (n), L (n), U (n);
   uint i, j;

   for (i = 0; i < n; i++)
      D.set (i, i, A.get (i, i));

   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
        {
           if (i > j)
              L.set (i, j, A.get (i, j));
           else if (i < j)
              U.set (i, j, A.get (i, j));
        }

   _D = D;
   _L = L;
   _U = U;

   return;

}

#define DIM 20
#define SET_A_VALUES \
for (i=0;i<DIM;i++) \
for (j=0;j<DIM;j++) { \
A.set(i, j, 0); \
if (i==j) A.set( i, j, 3); \
if (i==(j-1)) A.set( i, j, -1); \
if (i==(j+1)) A.set( i, j, -1); \
}

#define SET_B_VALUES \
for (i=0;i<DIM;i++) { \
B.set( i, 1); \
}
```

```
void
test_sor ()
{
   matrix A (DIM);
   vector B (DIM);
   vector *x;
   uint i, j;

   // Put the matrix and vector values
   SET_A_VALUES;
   SET_B_VALUES;

   cout << "A: " << endl;
   cout << A << endl << endl;

   cout << "B: " << endl;
   cout << B << endl;
   cout << endl;

   // SOR
   cout << endl;

   for (i = 1; i <= 19; i++)
     {
        cout << "L'ush me SOR[w=" << 0.1 * i << "]" << endl;

        x = sor_algorithm_fast (A, B, 0.1 * i, 0.5 * 0.00001);
        cout << "X: \n[" << *x << "]\n";
        delete x;
        cout << endl;
     }

   cout << "L'ush me SOR[w=" << 1.142 << "]" << endl;

   x = sor_algorithm_fast (A, B, 1.142, 0.5 * 0.00001);
   cout << "X: \n[" << *x << "]\n";
   delete x;
   cout << endl;

   // Crammer
   cout << "L'ush me Crammer: " << endl;
   x = gauss_algorithm (A, B, 1);
   cout << "X: \n" << *x << endl;
   delete x;
}
```

3.3 epan.hpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// Header arqe'io tou epan.cpp

#include <matrix.hpp>

#ifndef EPAN_H
# define EPAN_H
void matrix_break (const matrix & A, matrix & _D, matrix & _L, matrix & _U);
void test_sor ();
real calc_speed (const matrix & T, uint rot);

#endif
```

3.4 sor.cpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// To arqe'io (maz'i me to sor.hpp), peri'eqoun sunart'hseic gia thn SOR
// epanal'hpthk'h mej'odo. H kentrik'h e'inai h sor_algorithm(), h opo'ia
// ektele'i ton algorijmo SOR gia touc dedom'enouc p'inakec.

#include <sor.hpp>

// upolog'izei ton p'inaka Lw thc SOR mej'odou. Epistr'efei
// 0 an epit'uqei.
int
calc_Lw (matrix & _Lw, const real & w, const matrix & D, const matrix & L,
         const matrix & U)
{

   // h SOR sugkl'inei gia w < 2 && w > 0
   if (w < 0 || w > 2)
```

```
      return -1;                // error

   // Lw = (D-wL)^(-1) * [(1-w)*D+wU]
   _Lw = matrix_reverse (D - (L * w)) * ((D * ((real) 1 - w)) + (U * w));

   return 0;                    // finished

}

// upolog'izei ton p'inaka cw thc SOR mej'odou. Epistr'efei
// 0 an epit'uqei.
int
calc_cw (vector & _cw, const real & w, const matrix & D, const matrix & L,
         const vector & b)
{

   // h SOR sugkl'inei gia w < 2 && w > 0
   if (w < 0 || w > 2)
      return -1;                // error

   // cw = w(D-wL)^(-1) * b
   _cw = (matrix_reverse (D - (L * w)) * w) * b;

   return 0;                    // finished

}

// epistr'efei 0 an up'arqei h epijumhth diafor'a metax'u 'olwn twn stoiqe'iwn
// twn duo dianusm'atwn (a,b). To prec(ision) pr'epei na e'inai 0.5*10^(-m),
// 'opou m ta shmantik'a yhfi'a pou mac endiaf'eroun.
int
check_precision (const vector & x_prev, const vector & x, real prec)
{
   vector dif;

   //krit'hrio stamat'hmatoc me n'ormec
   dif = x - x_prev;
   if (dif.norm () / x.norm () < prec)
      return 0;
   else
      return 1;

}

// Ulopo'ihsh tou SOR me thn analutik'h ekfrash tou algor'ijmou.
vector *
```

```
sor_algorithm_fast (const matrix & A, const vector & B, const real & w,
                    const real & prec)
{
   int i, j;
   int n = B.getn ();
   vector x (n), old_x;
   uint loops = 0;
   vector *ret = new vector (x);
   real sum1, sum2, val;

   x.fill (1);
   do
     {
        old_x = x;
        for (i = 0; i < n; i++)
          {

             sum1 = 0;
             for (j = 0; j < i; j++)
               {
                  sum1 = sum1 + A.get (i, j) * x.get (j);
               }

             sum2 = 0;
             for (j = i + 1; j < n; j++)
               {
                  sum2 = sum2 + A.get (i, j) * old_x.get (j);
               }

             val =
                (1 - w) * old_x.get (i) +
                (w * (B.get (i) - sum1 - sum2)) / A.get (i, i);
             x.set (i, val);
          }
        loops++;
     }
   while (check_precision (x, old_x, prec) != 0);

   cout << "Loops in SOR algorithm: " << loops << endl;
   *ret = x;
   return ret;
}
```

3.5 sor.hpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// Header arqe'io tou sor.cpp

#ifndef SOR_H
# define SOR_H
#include <matrix.hpp>
#include <vector.hpp>
#include <epan.hpp>

int calc_Lw (matrix & _Lw, const real & w, const matrix & D, const matrix & L,
             const matrix & U);
int calc_cw (vector & _cw, const real & w, const matrix & D, const matrix & L,
             const vector & b);
vector *sor_algorithm_fast (const matrix & A, const vector & B,
                            const real & w, const real & prec);

#endif
```

4 Μέθοδος Συζυγών κλίσεων

1η Εργαστηριακή Ασκηση στην Αριθμητική γραμμική Αλγεβρα

4.1 Ερώτημα II (a)

Οι λύσεις που δίνονται απο το πρόγραμμα είναι:

Λύση με μέθοδο συζυγών κλίσεων (επαναλήψεις: 2)

```
X: [0.0121951 0.0121951 0.0121951 0.0121951 0.0121951
    0.0121951 0.0121951 0.0121951 0.0121951 0.0121951
    0.0121951 0.0121951 0.0121951 0.0121951 0.0121951
    0.0121951 0.0121951 0.0121951 0.0121951 0.0121951]
```

4.2 Ερώτημα II (b)

Η μέθοδος των συζυγών κλίσεων είναι καλύτερη απο την Gauss-Seidel στους συγκεκριμένους πίνακες, αφού χρειάζεται μόνο 2 επαναλήψεις, έναντι των 120 της Gauss-Seidel και η πολυπλοκότητα των επαναλήψεων είναι περίπου η ίδια.

Η μέθοδος του Cholesky θέλει εφαρμόγη του αλγορίθμου του Cholesky για να χωριστεί ο A σε LL^T, μια προς τα εμπρός αντικατάσταση (για το σύστημα $Ly = b$), και μία προς τα πίσω αντικατάσταση (για το σύστημα $L^T x = b$).

Μιας και οι επαναλήψεις της μεθόδου των συζυγών κλίσεων είναι μόνο 2, είναι προτιμότερο να χρησιμοποιηθεί αντί της μεθόδου του Cholesky.

4.3 conjurate-gradients.cpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// To arqe'io (maz'i me to conjurate-gradients.hpp), peri'eqoun sunart'hseic gia
// thn M'ejodo suzhg'wn kl'isewn.

#include <conjurate-gradients.hpp>
#include <crammer.hpp>
#include <gauss.hpp>

// Epistr'efei !=0 an to dedom'eno dianusma e'inai to mhdenik'o.
static inline int
```

```
vector_zero (vector & r)
{
   uint i;
   for (i = 0; i < r.getn (); i++)
      if (r.get (i) != 0)
         return 0;

   return 1;

}

// Implementation of the Conjurate Gradients Algorithm.
// Input parameters are a matrix A, and a vector B.
// A heap allocated vector is returned.
vector *
cnj_grad_algorithm (const matrix & A, const vector & B)
{
   int k, n = A.getn ();        // n*n
   vector *x = neW vector (n);
   vector r_prev (n);
   vector r_next (n);
   vector p (n);
   real bk, a = 0;
   uint loops = 0;

   // elegqoc an o p'inakac A e'inai summetrik'oc
   if (matrix_invert (A) != A)
      return NULL;

   // o alg'orijmoc
   x->fill (0);
   for (k = 0; k < n; k++)
     {
        loops++;
        r_prev = r_next;
        if (k == 0)
          {                     //thn pr'wth for'a den up'arqei to a
             r_next = B - (A * (*x));
          }
        else
           r_next = r_prev - (A * p) * a;

        if (vector_zero (r_next))
          {
             goto finish;
          }
```

```
        else
          {
             if (k == 0)
               {
                  p = r_next;
               }
             else
               {
                  bk = (r_next * r_next) / (r_prev * r_prev);
                  p = r_next + p * bk;
               }
             a = (r_next * r_next) / ((A * p) * p);
             *x = (*x) + p * a;
          }
     }

 finish:
   cout << "Loops in Conjurate gradients algorithm: " << loops << endl;
   return x;

}

#define DIM 20
#define SET_A_VALUES \
for (i=0;i<DIM;i++) \
for (j=0;j<DIM;j++) { \
if (i==j) A.set( i, j, 6); \
else A.set( i, j, 4); \
}

#define SET_B_VALUES \
for (i=0;i<DIM;i++) { \
B.set( i, 1); \
}

// Tests the conjurate gradients algorithm, using the
// given matrix and vectors, for the exercise.
void
test_cg ()
{
   matrix A (DIM);
   vector B (DIM);
   vector *x;
   uint i, j;

   // Put the matrix and vector values
```

```
   SET_A_VALUES;
   SET_B_VALUES;

   cout << "A: " << endl;
   cout << A << endl << endl;

   cout << "B: " << endl;
   cout << B << endl;
   cout << endl;

   // CONJURATE GRADIENTS
   cout << endl;

   cout << "L'ush me m'ejodo suzug'wn kl'isewn" << endl;

   x = cnj_grad_algorithm (A, B);
   cout << "X: \n[" << *x << "]\n";
   delete x;
   cout << endl;

   cout << "L'ush me Gauss: " << endl;
   x = gauss_algorithm (A, B, 1);
   cout << "X: \n" << *x << endl;
   delete x;
}
```

4.4 conjurate-gradients.hpp

```
#ifndef CG_HPP
# define CG_HPP

#include <matrix.hpp>
#include <vector.hpp>

void test_cg ();

#endif
```

5 Μέθοδος Crammer

Η μέθοδος του Crammer χρησιμοποιήθηκε για να επιβεβαιωθούν οι λύσεις των παραπάνω ασκήσεων.

5.1 crammer.cpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// To arqe'io peri'eqei sunart'hshc gia l'ush tou sust'hmatoc pin'akwn
// Aq=B, me th m'ejodo Crammer. Douleuei ikanopoihtik'a gia n<20

#include <common.hpp>
#include <vector.hpp>
#include <matrix.hpp>
#include <crammer.hpp>

matrix
matrix_replace_column (uint col, const matrix & A, const vector & B)
{
   uint i;
   uint n = A.getn ();
   matrix RET = A;

   for (i = 0; i < n; i++)
     {
        RET.set (i, col, B.get (i));
     }
   return RET;
}

vector *
crammer_algorithm (const matrix & A, const vector & B)
{
   uint n = A.getn ();          // n*n
   vector *x = new vector (n);
   uint i;
   matrix C;
   real val, det;

   det = matrix_det (A);
```

```
   for (i = 0; i < n; i++)
     {
        C = matrix_replace_column (i, A, B);
        val = matrix_det (C) / det;
        x->set (i, val);
     }

   return x;
}
```

5.2 crammer.hpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// Header arqe'io tou crammer.cpp

#ifndef CRAMMER_H
# define CRAMMER_H

vector *crammer_algorithm (const matrix & A, const vector & B);
matrix matrix_replace_column (uint col, const matrix & A, const vector & B);

#endif
```

6 Γενικές Κλάσεις

6.1 Matrix

6.1.1 matrix.hpp

```
// Iw'annina 2001
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// Header arqe'io tou matrix.cpp

#ifndef MATRIX_H
# define MATRIX_H

#include <vector.hpp>

class matrix
{
 public:
   matrix (const uint);
    ~matrix ();
     matrix (const matrix &);
     matrix ();

//functions
   uint getn () const;
   real get (const uint i, const uint j) const;
   real norm1 () const;
   void set (const uint i, const uint j, real val);
   void fill (const real fl);
   void diagfill (const real fl);
   void swaplines (const int line1, const int line2);

//operators
   bool operator == (const matrix &) const;
   bool operator != (const matrix &) const;
   matrix operator + (const matrix &) const;
   matrix operator - (const matrix &) const;
     matrix & operator = (const matrix &);
   matrix operator *(const real &) const;
   matrix operator *(const matrix &) const;
   vector operator *(const vector &) const;
```

```
   friend ostream & operator << (ostream &, const matrix &);

 private:
   void direct_set (const uint i, real val)
   {
      elem[i] = val;
   }
   real direct_get (const uint ij) const
   {
      return elem[ij];
   }

   uint itsn;
   real *elem;
   int *line_index;
};

//diloseis synartisewn
matrix *matrix_step (uint a, uint b, const matrix & A);
real matrix_det (const matrix & A);
matrix matrix_invert (const matrix & A);
matrix matrix_reverse (const matrix & A);

#endif
```

6.1.2 matrix.cpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// To arqe'io (maz'i me to matrix.hpp), peri'eqei thn kl'ash (matrix) gia th dhmiourg'ia
// kai qr'hsh (pollaplasiasm'o, pr'osjesh klp) pin'akwn $n*n$.
// Up'arqoun ep'ishc sunart'hseic gia ton upologism'o or'izousac klp.

#include "matrix.hpp"

static matrix NULL_MATRIX (0);

// returns the matrix's size
uint matrix::getn () const
{
   return itsn;
```

```
}

#define get_line_index( line) line_index[line]

// enall'asei d'uo gramm'ec tou p'inaka qwr'ic na tic antigr'ayei
// (qrhsimopoie'i index)
void
matrix::swaplines (const int line1, const int line2)
{
   uint tmp;

   if (line1 == line2)
      return;                   // no need to do anything

   tmp = line_index[line1];

   line_index[line1] = line_index[line2];
   line_index[line2] = tmp;

}

// upolog'izei thn ||A||1
real matrix::norm1 () const
{
   uint i, j;
   uint n = itsn;
   real max;
   real *sums = new real[n];
   real abs;

   for (i = 0; i < n; i++)
     {
        sums[i] = 0;
        for (j = 0; j < n; j++)
          {
             abs = this->get (i, j);
             if (abs < 0)
                abs = abs * (-1);
             sums[i] = sums[i] + abs;
          }
     }

   max = sums[0];
   for (i = 0; i < n; i++)
     {
        if (sums[i] > max)
```

```
            max = sums[i];
      }
   delete sums;

   return max;

}


void
matrix::set (const uint i, const uint j, real val)
{
   elem[(get_line_index (i) * itsn) + j] = val;
}


// returns a matrix element
real matrix::get (const uint i, const uint j) const
{
   return elem[(get_line_index (i) * itsn) + j];
}

// fills the matrix
void
matrix::fill (const real fl)
{
   for (uint i = 0; i < itsn; i++)
     {
        for (uint j = 0; j < itsn; j++)
          {
             elem[i * itsn + j] = fl;
          }
     }
}


// fills the diagonial
void
matrix::diagfill (const real fl)
{
   for (uint i = 0; i < itsn; i++)
     {
        for (uint j = 0; j < itsn; j++)
          {
             if (i == j)
               {
```

```
                elem[get_line_index (i) * itsn + j] = fl;
              }
            else
              {
                elem[get_line_index (i) * itsn + j] = 0;
              }
          }
     }
}

// constructor
matrix::matrix (const uint n)
{
   uint i;

   if (n == 0)
      line_index = NULL;
   else
      line_index = new int[n];
   for (i = 0; i < n; i++)
      line_index[i] = i;

   if (n == 0)
      elem = NULL;
   else
      elem = new real[n * n];

   itsn = n;
   for (i = 0; i < n; i++)
     {
        for (uint j = 0; j < n; j++)
          {
             elem[i * itsn + j] = 0;
          }
     }

}

// constructor
matrix::matrix ()
{
   line_index = NULL;
   elem = NULL;
   itsn = 0;

}
```

```
// destructor
matrix::~matrix ()
{
   itsn = 0;
   delete elem;
   delete line_index;
}

// print a matrix
ostream & operator << (ostream & output, const matrix & A)
{
   uint n = A.getn ();
   for (uint i = 0; i < n; i++)
     {
        for (uint j = 0; j < n; j++)
          {
             output << A.get (i, j) << "\t";
          }
        output << endl;
     }
   return output;
}

// constructor
matrix::matrix (const matrix & rhs)
{
   uint i, n;

   n = rhs.getn ();

   line_index = new int[n];
   for (i = 0; i < n; i++)
      line_index[i] = i;

   elem = new real[n * n];
   itsn = n;

   for (i = 0; i < n; i++)
     {
        for (uint j = 0; j < n; j++)
          {
             this->set (i, j, rhs.get (i, j));
          }
     }
```

```
}

// matrix == matrix
bool matrix::operator == (const matrix & rhs)
     const
     {

        if (itsn != rhs.getn ())
           return
              0;                // false

        for (uint i = 0; i < itsn; i++)
          {
             for (uint j = 0; j < itsn; j++)
                if (rhs.get (i, j) != this->get (i, j))
                   return
                      0;        // false
          }
        return
           1;                   // true
     }

// matrix != matrix
bool matrix::operator != (const matrix & rhs)
const
{

  if (*this == rhs)
    return 0;                // false
  else return 1;  //true
}


// matrix + matrix
matrix
matrix::operator + (const matrix & rhs) const
{
   matrix B (itsn);

   if (itsn != rhs.getn ())
      return NULL_MATRIX;

   for (uint i = 0; i < itsn; i++)
     {
        for (uint j = 0; j < itsn; j++)
```

```
            B.set (i, j, elem[get_line_index (i) * itsn + j] + rhs.get (i, j));
      }
   return B;
}

// matrix - matrix
matrix matrix::operator - (const matrix & rhs)
const
{
    matrix
    B (itsn);

    if (itsn != rhs.getn ())
       return NULL_MATRIX;

    for (uint i = 0; i < itsn; i++)
      {
          for (uint j = 0; j < itsn; j++)
            B.set (i, j, this->get (i, j) - rhs.get (i, j));
       }
     return B;
}

// matrix = other_matrix
matrix & matrix::operator = (const matrix & rhs)
{
   uint i;

   if (this == &rhs)
      return *this;
   delete elem;
   delete line_index;

   uint n = rhs.getn ();
   itsn = n;
   elem = new real[n * n];

   line_index = new int[n];
   for (i = 0; i < n; i++)
      line_index[i] = i;

   for (i = 0; i < n; i++)
     {
        for (uint j = 0; j < n; j++)
           this->set (i, j, rhs.get (i, j));
     }
```

```
   return *this;
}

// matrix * number
matrix matrix::operator * (const real & l) const
{
    matrix B (itsn);

    for (uint i = 0; i < itsn; i++)
     {
         for (uint j = 0; j < itsn; j++)
          {
            B.set (i, j, l * elem[get_line_index (i) * itsn + j]);
          }
      }
     return B;
}


// matrix * matrix
matrix matrix::operator * (const matrix & rhs) const
{
   matrix B (itsn);

   if (itsn != rhs.getn ())
      return NULL_MATRIX;

   uint n = itsn;
   real sum = 0;

   for (uint i = 0; i < itsn; i++)
    {
      for (uint j = 0; j < itsn; j++)
       {
          sum = 0;
          for (uint x = 0; x < n; x++)
           {
               sum =
                  sum + elem[get_line_index (i) * itsn + x] * rhs.get (x, j);
            }
           B.set (i, j, sum);
        }
     }
     return B;
}
```

```
// matrix * vector
vector matrix::operator * (const vector & rhs) const
{
     uint n = itsn;
     vector B(n);

     if (n != rhs.getn ())
        return _NULL_VECTOR;

     real sum;

     for (uint i = 0; i < n; i++)
      {
        sum = 0;
        for (uint x = 0; x < n; x++)
         {
            sum = sum + (this->get (i, x) * rhs.get (x));
         }
         B.set (i, sum);
      }

     return B;
}

//matrix functions

// kanei ton upop'inaka pou prok'uptei afair'wntac thn a
// gramm'h kai b st'hlh.
// H tim'h pou epistr'efetai pr'epei na g'inei delete.
matrix *
matrix_step (uint a, uint b, const matrix & A)
{
   uint n = A.getn ();
   long bi = -1;
   long bj = -1;
   uint j = 0, i = 0;
   matrix *
      B =
      new
      matrix (n - 1);

   for (i = 0; i < n; i++)
     {
        if (i != a) bi++;
        bj = -1;
        for (j = 0; j < n; j++)
```

```
          {
             if (j != b)
                bj++;
             if (i != a && j != b)
                B->set ((uint) bi, (uint) bj, A.get (i, j));
          }
     }
   return B;
}

//prosomo'iwsh tou (-1)^(i+j)
inline static int
matrix_pros (uint i, uint j)
{
   i += 1;
   j += 1;

   // % is expensive
   if ((i + j) % 2 == 0)
        return 1;
   else
        return -1;
}

// or'izousa (at last!)
// upolog'izei thn or'izousa p'inaka. Qrhsimopoie'i anadromik'h diadikas'ia
// kai e'inai pol'u arg'h gia meg'alouc p'inakec (pou den 'eqoun poll'a mhdenik'a).
real
matrix_det (const matrix & A)
{
   uint n = A.getn ();
   real b = 0, sum = 0;
   real tmp;
   matrix *mtmp;

   if (A.getn () == 2)
     {
        return ((A.get (0, 0) * A.get (1, 1)) -
                (A.get (1, 0) * A.get (0, 1)));
     }
   else
     {
        if (A.getn () == 0 || A.getn () == 1)
          {
             return 0;
          }
```

```
        else
          {
             for (uint j = 0; j < n; j++)
               {
                  tmp = A.get (0, j);

                  // ean to stoiqe'io e'inai mhdenik'o, den upolog'izei touc
                  // upop'inakec gia na k'anei ton pollaplasiasm'o met'a.
                  if (tmp != 0)
                    {
                       mtmp = matrix_step (0, j, A);
                       b = matrix_det (*mtmp) * matrix_pros (0, j) * tmp;
                       delete mtmp;
                    }
                  else
                     b = 0;

                  sum = sum + b;
               }
             return sum;
          }
     }
}

//anastrofos pinakas
matrix
matrix_invert (const matrix & A)
{
   uint n = A.getn ();
   matrix B (n);

   for (uint i = 0; i < n; i++)
     {
        for (uint j = 0; j < n; j++)
          {
             B.set (j, i, A.get (i, j));
          }
     }
   return B;
}

//antistrofos pinakas!
matrix
matrix_reverse (const matrix & A)
{
   real a = 0, det = matrix_det (A);
```

```
   uint n = A.getn ();
   matrix B(n);
   matrix *mtmp;

   if (det == 0 || n <= 1)
      return NULL_MATRIX;

   if (n == 2)
     {
        B.set (0, 0, A.get (1, 1) / det);
        B.set (0, 1, (A.get (0, 1) * (-1)) / det);
        B.set (1, 0, (A.get (1, 0) * (-1)) / det);
        B.set (1, 1, A.get (0, 0) / det);
     }
   else
     {
        for (uint i = 0; i < n; i++)
          {
             for (uint j = 0; j < n; j++)
               {
                  mtmp = matrix_step (i, j, A);
                  a = ((matrix_det (*mtmp)) * matrix_pros (i, j)) / det;
                  delete mtmp;
                  B.set (j, i, a);
               }
          }
     }

   return B;

}
```

6.2 Vector

6.2.1 vector.hpp

```
// Iw'annina 2001
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// Header arqe'io tou vector.cpp
```

```
#ifndef VECTOR_H
# define VECTOR_H

class vector
{
 public:
   vector (const uint);
     vector ();
    ~vector ();
     vector (const vector &);

//functions
   uint getn () const;
   real get (const uint i) const;
   void set (const uint i, real val);
   void fill (const real fl);
   void swaplines (const int line1, const int line2);
   real norm1 () const;
   real norm () const;
//operators
   vector operator + (const vector &) const;
   vector operator - (const vector &) const;
     vector & operator = (const vector &);
   vector operator *(const real) const;
   real operator *(const vector &) const;
   friend ostream & operator << (ostream &, const vector &);

 private:
   // line_index is not kept, since it's too
   // easy to swap lines here.
     uint itsn;
   real *elem;
};

# ifndef NULL_OK
extern vector _NULL_VECTOR;
# endif

#endif
```

6.2.2 vector.cpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// Perigraf'h:
// Peri'eqei thn kl'ash vector h opo'ia qrhsimopoie'itai gia thn qr'hsh
// dianusm'atwn. Dhl'wnontai kai pr'axeic metax'u dianusm'atwn (+,-),
// pollaplasiasm'oc me arijm'o klp.


#define NULL_OK
#include "vector.hpp"

vector _NULL_VECTOR (0);

// returns the vector's size
uint vector::getn () const
{
   return itsn;
}

void
vector::swaplines (const int line1, const int line2)
{
   real tmp;

   if (line1 == line2)
      return;                   // no need to do anything

   tmp = elem[line1];

   elem[line1] = elem[line2];
   elem[line2] = tmp;

}

void
vector::set (const uint i, real val)
{
   elem[i] = val;
}


// returns a vector element
real vector::get (const uint i) const
{
```

```
   return elem[i];
}

real vector::norm1 () const
{
   real
      max,
      abs;
   uint
      i;

   max = elem[0];
   if (max < 0)
      max = max * (-1);

   for (i = 0; i < itsn; i++)
     {
        abs = elem[i];
        if (abs < 0)
           abs = abs * (-1);

        if (elem[i] > max)
           max = elem[i];
     }

   return max;

}

real vector::norm () const
{
   real
      max,
      abs;
   uint
      i;
   max = 0;

   for (i = 0; i < itsn; i++)
     {
        abs = elem[i];          // abs == |elem[i]|
        if (abs < 0)
           abs = abs * (-1);

        if (elem[i] > max)
           max = max + elem[i];
```

```
     }

   return max;

}

// fills the matrix
void
vector::fill (const real fl)
{
   for (uint j = 0; j < itsn; j++)
     {
        elem[j] = fl;
     }
}


// constructor
vector::vector (const uint n)
{
   uint i;

   if (n == 0)
      elem = NULL;
   else
      elem = new real[n];

   for (i = 0; i < n; i++)
     {
        elem[i] = 0;
     }
   itsn = n;

}

// constructor
vector::vector ()
{
   // this creates an unusable vector
   // you must initialize it later
   elem = NULL;
   itsn = 0;

}

// destructor
```

```
vector::~vector ()
{
   itsn = 0;
   delete elem;
}

// print a vector
ostream & operator << (ostream & output, const vector & A)
{
   uint n = A.getn ();
   for (uint i = 0; i < n; i++)
     {
        output << A.get (i) << "\t";
     }
   return output;
}

// constructor
vector::vector (const vector & rhs)
{
   uint i, n;

   itsn = n = rhs.getn ();

   elem = new real[n];

   for (i = 0; i < n; i++)
     {
        elem[i] = rhs.get (i);
     }
}


// vector + vector
vector vector::operator + (const vector & rhs)
     const
     {
        if (itsn != rhs.getn ())
           return
           vector (0);

        vector
        B (itsn);

        for (uint i = 0; i < itsn; i++)
          {
```

```
            B.set (i, elem[i] + rhs.get (i));
          }
        return
           B;
     }

// vector - vector
vector vector::operator - (const vector & rhs)
     const
     {
        if (itsn != rhs.getn ())
           return
           vector (0);

        vector
        B (itsn);

        for (uint i = 0; i < itsn; i++)
          {
            B.set (i, (elem[i] - rhs.get (i)));
          }
        return
           B;
     }

// vector = other_vector
vector & vector::operator = (const vector & rhs)
{
   if (this == &rhs)
      return *this;

   delete elem;

   uint n = rhs.getn ();
   itsn = n;
   elem = new real[n];

   for (uint i = 0; i < n; i++)
     {
        elem[i] = rhs.get (i);
     }
   return *this;
}

// vector * number
vector vector::operator * (const real l)
```

```
     const
     {
        vector
        B (itsn);
        for (uint i = 0; i < itsn; i++)
          {
             B.set (i, l * elem[i]);
          }
        return
           B;
     }


// vector * vector
// eswterik'o gin'omeno
real vector::operator * (const vector & l)
     const
     {
        real
           ret =
           0;

        for (uint i = 0; i < itsn; i++)
          {
             ret = ret + (this->get (i) * l.get (i));
          }
        return
           ret;
     }
```

7 Κοινά αρχεία

7.1 main.cpp

```
#include <gauss.hpp>
#include <epan.hpp>
#include <conjurate-gradients.hpp>
int
main ()
{
   test_cg ();
//      test_sor();
//      test_gauss();

   return 0;
}
```

7.2 common.hpp

```
// Iw'annina 2001
// Ergas'ia gia thn "Arijmhtik'h Grammik'h Algebra"
// Maurogiann'opouloc N'ikoc

// Header arqe'io tou project. Ed'w dhl'wnontai ta FLOAT, kai DOUBLE_FLOAT
// pou qrhsimopoio'untai gia tic pr'axeic kinht'hc upodiastol'hc.
// Ant'i gia float, double, mporo'un na qrhsimopoihjo'un ta double, long double
// (an uposthr'izontai apo to s'usthma kai ton compiler)

#ifndef COMMON_H
# define COMMON_H

typedef float FLOAT;
typedef double DOUBLE_FLOAT;
typedef unsigned int uint;

typedef FLOAT real;

#include <sys/types.h>
#include <iostream.h>

#endif
```

7.3 Makefile

```
CXX = g++-3.0
CCOPTS = -O3 -I. -Wall -ggdb3 -pedantic -march=i686 -mcpu=i686  #-DUSE_REAL

all: matrix

matrix.o: matrix.cpp matrix.hpp
$(CXX) -c matrix.cpp $(CCOPTS)

main.o: main.cpp
$(CXX) -c main.cpp $(CCOPTS)

epan.o: epan.cpp epan.hpp
$(CXX) -c epan.cpp $(CCOPTS)

sor.o: sor.cpp sor.hpp
$(CXX) -c sor.cpp $(CCOPTS)

vector.o: vector.cpp vector.hpp
$(CXX) -c vector.cpp $(CCOPTS)

gauss.o: gauss.cpp matrix.hpp
$(CXX) -c gauss.cpp $(CCOPTS)

crammer.o: crammer.cpp matrix.hpp
$(CXX) -c crammer.cpp $(CCOPTS)

conjurate-gradients.o: conjurate-gradients.cpp conjurate-gradients.hpp
$(CXX) -c conjurate-gradients.cpp $(CCOPTS)

matrix: matrix.o gauss.o vector.o crammer.o main.o epan.o \
sor.o conjurate-gradients.o
$(CXX) crammer.o matrix.o gauss.o vector.o main.o \
epan.o sor.o conjurate-gradients.o -o matrix

clean:
rm -f *~ matrix core *.o
```