Zadanie D
Skalowany Gauss

Język programowania: C++

Autorzy: Marek Śmieja i Tomasz Kapela

Do rozwiazywania układów równań liniowych na ogół stosuje się metodę eliminacji Gaussa. Z uwagi na niedokładność reprezentacji i błędy zaokrąglań wyniki uzyskiwane z użyciem jej podstawowej wersji mogą nie być dokładne. Dlatego rozważa się wiele modyfikacji zwiazanych z wyborem elementów głównych, a także stosuje się iteracyjne poprawianie wyniku.

Zadanie

Zadanie polega na zaimplementowaniu funkcji o sygnaturze

```
Vector solveEquations(
       const Matrix & A,   // Macierz
       const Vector & b,   // Wektor
       double  eps         // dopuszczalny błąd
);
```

która metodą eliminacji Gaussa rozwiąże układ równań równań liniowych

```
   A x = b
```

gdzie *A* jest macierzą kwadratową *n x n*, *x* i *b* są wektorami rozmiaru *n*. Maksymalny rozmiar to n=3000. (Rozmiar n zwracają metody *size* zarówno wektora jak i macierzy). Zakładamy, że podany układ ma dokładnie jedno rozwiązanie.

Aby zaliczyć wszystkie testy należy zaimplementować wariant metody eliminacji Gaussa:

- ze skalowanym wyborem elementów głównych,
- skale wierszy powinny być liczone tylko raz na początku algorytmu,
- rozwiązanie jest poprawiane iteracyjnie aż do otrzymania żądanej dokładności.

Sposób przesyłania rozwiązania

Do systemu BaCa należy przesłać plik o nazwie source.cpp zawierający definicję funkcji

```
solveEquations
```

i ewentualnych funkcji pomocniczych, ale nie zawierający funkcji main. Programy testujące będą kompilowały się w następujący sposób

```
g++ test.cpp source.cpp -O2 -std=c++11 -o test
```

Przykładowy program testujący znajduje się poniżej razem z przykładowymi danymi wejściowymi i spodziewanym wyjściem.

Ocena poprawności wyniku

Funkcja solveEquations powinna zwrócić wektor x zawierający rozwiązanie układu równań z dokładnością eps (liczonej według normy maksimum) tzn. powinien być spełniony warunek

```
residual_vector(A, b, x).max_norm() < eps
```

Zwracany wynik, jak również wektor rezydualny wypisywany przez przykładowy test mogą się różnić od przykładowego wyjścia. Ważne jest aby powyższy warunek był spełniony.

Reprezentacja wektorów i macierzy

Dla Państwa wygody przygotowałem prostą implementację class Vector i Matrix. Ich definicje znajdują się w pliku nagłówkowym vectalg.h. Nie należy go przesyłać, będzie dostępny na BaCy.
Plik vectalg.h

Przykłady użycia

```
#include <iostream>
#include "vectalg.h"
using namespace std;
int main() {
    Matrix A(4);    // Niezainicjowana macierz 4x4
    Matrix B {{1., 2.},{1.4, 1.23}}; // Macierz 2x3 wypełniona
danymi
    A = B;          // A jest równe B
    Matrix C(A);    //
    A(0,0) = 5;     // zmieniamy element o współrzędnych (0,0)
    cout << A << endl;
    cout << C << C.size() << endl;

    Vector v(3);
    v[0] = -1; v[1] = 9; v[2] = 10;  // dostęp do składowych
    Vector w = {1,2,3,5};
    cout << v << endl;
    cout << w << endl;
    cout << v.max_norm() << v.size() << endl;
    // v[3] = 4; // ERROR!
    return 0;
}
```

Przykładowe testy

Program testujący

```
#include "source.cpp"
```

```
#include <iostream>
#include "vectalg.h"
using namespace std;

Vector solveEquations(
       const Matrix & A,
       const Vector & b,
       double  eps
);

int main(){
   cout.precision(17);
   int n = 0;
   double eps = 0;

   // wczytywanie danych
   cin >> n;
   Matrix a(n);
   Vector b(n);
   cin >> a >> b >> eps;

   Vector x = solveEquations(a, b, eps);

   auto residual = residual_vector(a, b, x);
   cout << "rozwiazanie = " << x << endl;
   cout << "rezydualny = " << residual << endl;
   cout << "blad = " << residual.max_norm()
        << " limit = " << eps << endl ;
   cout << "Test " << (residual.max_norm() < eps ? "":"nie ") <<
"zaliczony" << endl;
   return 0;
}
```

Wejście programu ma format

```
n
A
b
eps
```

Przykład 1

```
2
1 1
1 -1
1
1
1e-15
```