Metody numeryczne

Wojciech Chrobak

15 listopada 2017

Zadanie 2

$$||b1 - b2||,\ ||b3 - b4||\ -\ \textit{błędy bezwzględne}$$

$$\frac{||z1 - z2||}{||b1 - b2||},\ \frac{||z3 - z4||}{||b3 - b4||}\ -\ \textit{współczynniki uwarunkowania}$$

Współczynnik uwarunkowania określa w jakim stopniu błąd reprezentacji numerycznej danych wejściowych danego problemu wpływa na błąd wyniku. Współczynnik uwarunkowania definiuje się jako maksymalny stosunek błędu względnego rozwiązania do błędu względnego danych. Problem o niskim współczynniku uwarunkowania nazywamy dobrze uwarunkowanym zaś problemy o wysokim współczynniku uwarunkowania – źle uwarunkowanymi

Kod

```
#include <iostream>
#include <cmath>
#include <vector>
#include <iomanip>

using namespace std;

void printG(vector<vector<double>> G) {
  cout << endl;
  for (int i = 0; i < G.size(); i++) {
    for (int j = 0; j < G[i].size(); j++) {
      cout << "[ " << fixed << setprecision(6) << G[i][j] << " ]   ";
    }
    cout << endl;
  }
  cout << endl;
}

void printA(vector<double> A) {
  for (int i = 0; i < A.size(); i++) {
    cout << "[ " << showpoint << A[i] << " ]   ";
  }
  cout << endl;
}

vector<vector<double>> multi(vector<vector<double>> A, vector<vector<double>> B) {
  vector<vector<double>> res = A;

  for (int i = 0; i < A.size(); i++) {
    for (int j = 0; j < A[i].size(); j++) {
      double s = 0;
      for (int k = 0; k < B.size(); k++) {
```

```
          s += A[i][k] * B[k][j];
        }
        res[i][j] = s;
      }
    }

    return res;
  }

  vector<double> simply(vector<vector<double>> A, vector<double> B) {
    vector<double> res = B;

    for (int i = 0; i < A.size(); i++) {
      double s = 0;
      for(int k = 0; k < A.size(); k++) {
        s += A[i][k] * B[k];
      }
      res[i] = s;
    }

    return res;
  }


  vector<vector<double>> permutationMatrix(vector<vector<double>> A) {
    vector<vector<double>> permM = A;
    for (int i = 0; i < A.size(); i++) {
      for (int j = 0; j < A[i].size(); j++) {
        if (i == j) {
          permM[i][j] = 1;
        } else {
          permM[i][j] = 0;
        }
      }
    }

    for (int j = 0; j < A.size(); j++) {
      double max = A[j][j];
      int row = j;
      for (int i = j; i < A.size(); i++) {
        if (A[i][j] > max) {
          max = A[i][j];
          row = i;
        }
      }

      if (j != row) {
        vector<double> tmp = permM[j];
        permM[j] = permM[row];
        permM[row] = tmp;
      }
    }

    return permM;
  }


  void lu(vector<vector<double>> A, vector<vector<double>> &L, vector<vector<double>>
    &U) {
    for (int i = 0; i < A.size(); i++) {
      for (int j = 0; j < A.size(); j++) {
        L[i][j] = 0;
        U[i][j] = 0;
      }
```

```
    }


    for (int j = 0; j < A.size(); j++) {
      L[j][j] = 1;
      for (int i = 0; i <= j; i++) {
        double sum = 0;
        for (int k = 0; k < i; k++) {
          sum = sum + (U[k][j] * L[i][k]);
        }
        U[i][j] = A[i][j] - sum;
      }

      for (int i = j; i < A.size(); i++) {
        double sum2 = 0;
        for (int k = 0; k < j; k++) {
          sum2 = sum2 + (U[k][j] * L[i][k]);
        }
        L[i][j] = (A[i][j] - sum2) / U[j][j];
      }
    }
  }


  vector<double> forwardSubstitution(vector<vector<double>> M, vector<double> a) {
    vector<double> result;
    result.assign(a.size(), 0);
    for (int i = 0; i < M.size(); i++) {
      double val = 0;
      for (int c = 0; c < i; c++) {
        val = val + result[c] * M[i][c];
      }
      val = a[i] - val;
      result[i] = val / M[i][i];
    }
    return result;
  }

  vector<double> backSubstitution(vector<vector<double>> M, vector<double> a) {
    vector<double> result;
    result.assign(a.size(), 0);
    for (int i = M.size() - 1; i >= 0; i--) {
      double val = 0;
      for (int c = M[0].size() - 1; c > i; c--) {
        val = val + result[c] * M[i][c];
      }
      val = a[i] - val;
      result[i] = val / M[i][i];
    }
    return result;
  }

  vector<double> solveLU(vector<vector<double>> L, vector<vector<double>> U, vector<
    double> a) {
    vector<double> result;
    result = forwardSubstitution(L, a);
    result = backSubstitution(U, result);

    return result;
  }


  double norm(vector<double> u) {
    double a = 0;
```

```
  double norm;
  for (int i = 0; i < u.size(); ++i) {
    a += u[i] * u[i];
  }
  norm = sqrt(a);
  return norm;
}

vector<double> subtract(vector<double> A, vector<double> B) {
  vector<double> res;
  res.assign(A.size(), 0);

  for(int i = 0; i<res.size(); i++) {
    res[i] = A[i] - B[i];
  }

  return res;
}

int main() {
  vector<vector<double>> A = {{-116.66654, 583.33346,  -333.33308, 100.00012, 100.00012},
                {583.33346,  -116.66654, -333.33308, 100.00012,  100.00012},
                {-333.33308, -333.33308, 133.33383,  200.00025,  200.00025},
                {100.00012,  100.00012,  200.00025,  50.000125,  -649.99988},
                {100.00012,  100.00012,  200.00025,  -649.99988, 50.000125}};
  vector<vector<double>> L = A;
  vector<vector<double>> U = A;

  vector<vector<double>> z;
  z.assign(4,vector<double>(5));


  vector<vector<double>> right = {{-0.33388066, 1.08033290, -0.98559856, 1.31947922, -0.09473435},
                  {-0.33388066, 1.08033290, -0.98559856, 1.32655028, -0.10180541},
                  {0.72677951,  0.72677951, -0.27849178, 0.96592583, 0.96592583},
                  {0.73031505,  0.73031505, -0.27142071, 0.96946136, 0.96946136}};

  cout << "Macierz A: ";
  printG(A);

  vector<vector<double>> perm = permutationMatrix(A);
  cout << "Macierz permutacji: ";
  printG(perm);

  vector<vector<double>> permA = multi(perm, A);
  cout << "Macierz A po permutacji: ";
  printG(permA);

  lu(permA, L, U);
  cout << "Macierz L: ";
  printG(L);
  cout << "Macierz U: ";
  printG(U);

  cout << "\n--------------------------------------------------------------------------------\n\n";

  for(int i = 0; i < right.size(); i++) {
    cout << "b" << i+1 << ":    ";
    printA(right[i]);
    right[i] = simply(perm,right[i]);
    cout << "b" << i+1 << " po permutacji:    ";
    printA(right[i]);
```

```
    z[i] = solveLU(L,U,right[i]);
    cout << "z" << i+1 << ":    ";
    printA(z[i]);
    cout << "\n";
  }

  cout << "\n--------------------------------------------------------------------\n\n";

  cout << "|| b1 - b2 || = " << norm(subtract(right[0], right[1])) << endl;
  cout << "|| b3 - b4 || = " << norm(subtract(right[2], right[3])) << endl;
  cout << "\n|| z1 - z2 || / || b1 - b2 || = " << norm(subtract(z[0], z[1])) / norm(
  subtract(right[0], right[1])) << endl;
  cout << "|| z3 - z4 || / || b3 - b4 || = " << norm(subtract(z[2], z[3])) / norm(
  subtract(right[2], right[3])) << endl;
  return 0;
}
```

Działanie programu

$$A = \begin{bmatrix} -116.666540 & 583.333460 & -333.333080 & 100.000120 & 100.000120 \\ 583.333460 & -116.666540 & -333.333080 & 100.000120 & 100.000120 \\ -333.333080 & -333.333080 & 133.333830 & 200.000250 & 200.000250 \\ 100.000120 & 100.000120 & 200.000250 & 50.000125 & -649.999880 \\ 100.000120 & 100.000120 & 200.000250 & -649.999880 & 50.000125 \end{bmatrix}$$

$$p\ (macierz\ permutacji) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$pA = \begin{bmatrix} 583.333460 & -116.666540 & -333.333080 & 100.000120 & 100.000120 \\ 100.000120 & 100.000120 & 200.000250 & 50.000125 & -649.999880 \\ -116.666540 & 583.333460 & -333.333080 & 100.000120 & 100.000120 \\ -333.333080 & -333.333080 & 133.333830 & 200.000250 & 200.000250 \\ 100.000120 & 100.000120 & 200.000250 & -649.999880 & 50.000125 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.171429 & 1 & 0 & 0 & 0 \\ -0.200000 & 4.666664 & 1 & 0 & 0 \\ -0.571428 & -3.333327 & -0.500000 & 1 & 0 \\ 0.171429 & 1 & -0 & -1.999999 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 583.333460 & -116.666540 & -333.333080 & 100.000120 & 100.000120 \\ 0 & 120.000118 & 257.143120 & 32.857230 & -667.142775 \\ 0 & 0 & -1600.000047 & -33.333523 & 3233.331020 \\ 0 & 0 & 0 & 350.000245 & -349.996260 \\ 0 & 0 & 0 & 0 & 0.007970 \end{bmatrix}$$

$$z1 = \begin{bmatrix} 0.001983 \\ -0.000037 \\ -0.000220 \\ 0.000241 \\ -0.001780 \end{bmatrix}, z2 = \begin{bmatrix} 0.001983 \\ -0.000037 \\ -0.000220 \\ 0.000251 \\ -0.001790 \end{bmatrix}, z3 = \begin{bmatrix} 354.885181 \\ 354.885181 \\ 709.768198 \\ 354.883432 \\ 354.883432 \end{bmatrix}, z4 = \begin{bmatrix} 358.434025 \\ 358.434025 \\ 716.865884 \\ 358.432276 \\ 358.432276 \end{bmatrix}$$

$$||b1 - b2|| = 0.010000$$

$$||b3 - b4|| = 0.010000$$

$$\frac{||z1 - z2||}{||b1 - b2||} = 0.001429$$

$$\frac{||z3 - z4||}{||b3 - b4||} = 1003.764115$$

Wskaźnik uwarunkowania macierzy A:

$$\kappa = \frac{max|\lambda_i|}{min|\lambda_i|}$$

$$\kappa = 803011$$

Wniosek - duży wskaźnik uwarunkowania macierzy powoduje duże względne zaburzenia rozwiązania nawet dla małych zaburzeń wektora danych. Zadanie jest źle uwarunkowane.