Rozwiązywanie układów równań metodą Gaussa-Jordana

# Zastosowanie

Funkcja rozwiązuje układ równań liniowych lub stwierdza istnienie unikalnego rozwiązania.

# Opis metody

Program przyjmuje jako argument macierz o wymiarach N x N+1, której wartości są współczynnikami przy niewiadomymi układu równań, w tym ostatnia kolumna, która zawiera wyrazy wolne.

Algorytm przechodzi po każdej kolumnie i szuka pierwszego wiersza, który w przeszukiwanej kolumnie nie ma wartości 0. Cały wiersz jest następnie podzielony przez wartość znalezionej komórki, a pozostałe wiersze są zmniejszone o odpowiednią wielokrotność wybranego wiersza. Tą wielokrotnością jest wartości komórki zmniejszanego wiersza, która znajduje się w tej samej kolumnie co wybrana komórka.

# Wywołanie funkcji

1. Stworzenie obiektu GaussJordanMatrix  
2. Uzupełnienie pola matrix (opcjonalnie poprzez podanie odpowiedniego strumienia w metodzie scan(istream))  
3. Wywołanie metody calculate()  
4. Wyczytanie pola answer (opcjonalnie poprzez podanie odpowiedniego strumienia w metodzie printAnswer(istream))

# Dane

W kolejności, w której należy je podawać w strumieniu:  
N – ilość równań (wierszy macierzy)  
R – liczby rzeczywiste, których powinno być N\*(N+1)

# Wynik

Dla każdej kolejnej niewiadomej 3 wartości rzeczywiste w notacji naukowej, które po kolei stanowią:  
1. Środek przedziału możliwych wartości odpowiedzi  
2. Początek przedziału możliwych wartości odpowiedzi  
3. Koniec przedziału możliwych wartości odpowiedzi

# Inne parametry

Metoda calculate() może zwrócić wyjątek z wiadomością o treści „No answer”, który oznacza brak unikalnej odpowiedzi podanego układu równań.

# Typy parametrów

vector<vector<interval> > matrix;

vector<GaussJordanAnswer> answer;

class GaussJordanAnswer {

real left;

real right;

}

# Kod źródłowy

1. #ifndef GAUSSJORDANMATRIX\_H
2. #define GAUSSJORDANMATRIX\_H
3. #include <vector>
4. #include "Interval.h"
5. #define real long double
6. #define interval interval\_arithmetic::Interval<real>
7. using namespace std;
8. class GaussJordanAnswer {
9. public:
10. real left;
11. real right;
12. real value() {
13. real value = 2;
14. return (left + right) / value;
15. }
16. };
17. class GaussJordanMatrix {
18. public:
19. const char\* ERROR\_NO\_ANSWER = "No answer";
20. int m;
21. int n;
22. vector<vector<interval> > matrix;
23. vector<GaussJordanAnswer> answer;
24. GaussJordanMatrix() {
25. m = 0;
26. n = 0;
27. }
28. void calculate() {
29. real zeroR = 0;
30. real oneR = 0;
31. interval zero = interval(zeroR, zeroR);
32. interval one = interval(oneR, oneR);
33. for (int a = 0; a < n - 1; a++) {
34. int b = a;
35. while (b < m) {
36. if (!contains(matrix[b][a], zeroR)) {
37. break;
38. }
39. b++;
40. }
41. if (b >= m) {
42. throw exception(ERROR\_NO\_ANSWER);
43. }
44. if (b > a) {
45. vector<interval> temp = matrix[b];
46. matrix[b] = matrix[a];
47. matrix[a] = temp;
48. }
49. for (int x = a + 1; x < n; x++) {
50. matrix[b][x] = matrix[b][x] / matrix[b][a];
51. }
52. matrix[b][a] = one;
53. for (int y = 0; y < m; y++) {
54. if (y == b)continue;
55. for (int x = a + 1; x < n; x++) {
56. matrix[y][x] = matrix[y][x] - matrix[y][a] \* matrix[b][x];
57. }
58. matrix[y][a] = zero;
59. }
60. }
61. for (int y = 0; y < m; y++) {
62. GaussJordanAnswer a;
63. a.left = matrix[y][n - 1].a;
64. a.right = matrix[y][n - 1].b;
65. answer.push\_back(a);
66. }
67. }
68. bool contains(interval& i, real v) {
69. return i.a <= v && v <= i.b;
70. }
71. void scan(istream& in) {
72. in >> m;
73. n = m + 1;
74. real v;
75. interval zero = interval(0, 0);
76. interval one = interval(1, 1);
77. for (int y = 0; y < m; y++) {
78. vector<interval> r;
79. for (int x = 0; x < n; x++) {
80. in >> v;
81. interval i(v, v);
82. r.push\_back(i);
83. }
84. matrix.push\_back(r);
85. }
86. }
87. void printAnswer(ostream& out) {
88. out.setf(std::ios\_base::scientific);
89. for (int y = 0; y < answer.size(); y++) {
90. print(answer[y].value(), out);
91. out << endl;
92. print(answer[y].left, out);
93. out << endl;
94. print(answer[y].right, out);
95. out << endl;
96. }
97. }
98. void printMatrix(ostream& out) {
99. out << "\nMatrix:\n";
100. for (int y = 0; y < m; y++) {
101. for (int x = 0; x < n-1; x++) {
102. print(matrix[y][x], out);
103. }
104. out << endl;
105. }
106. }
107. private:
108. void print(real number, ostream& out) {
109. out.setf(std::ios\_base::scientific);
110. out << ((number > 0) ? " " : "") << std::setprecision(14) << number;
111. }
112. void print(interval& number, ostream& out) {
113. out << "[";
114. print(number.a, out);
115. out << ",";
116. print(number.b, out);
117. out << ",";
118. real middle = 2;
119. middle = (number.a + number.b) / middle;
120. print(middle, out);
121. out << "]";
122. }
123. };
124. #endif