



МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



**НГТУ
НЭТИ** | **Факультет прикладной
математики и информатики**

Кафедра прикладной математики

Лабораторная работа №4
по дисциплине «Уравнения математической физики»

РЕШЕНИЕ НЕСИММЕТРИЧНЫХ СЛАУ

Студенты БЕГИЧЕВ АЛЕКСАНДР

ШИШКИН НИКИТА

Группа ПМ-92

Преподаватели ЗАДОРОВЫЙ А. Г.

ПАТРУШЕВ И. И.

ПЕРСОВА М. Г.

Новосибирск, 2022

Цель работы

Изучить особенности реализации итерационных методов BCG, BCGSTAB, GMRES для СЛАУ с несимметричными разреженными матрицами. Исследовать влияние предобуславливания.

Вариант: Реализовать решение СЛАУ методом BCGSTAB с LU - предобуславливанием.

Практическая часть

1. Реализовать программу решения СЛАУ большой размерности в разреженном строчном формате в соответствии с заданием.
2. Протестировать разработанную программу на небольших матрицах.
3. Сравнить реализованный метод по вычислительным затратам с методами, используемыми в предыдущей лабораторной работе, на матрицах большой размерности, полученных в результате конечноэлементной аппроксимации в предыдущей лабораторной работе.

Тестирование

Тестирование метода на небольших матрицах

$$\begin{bmatrix} 6 & 2 & 0 & 5 & 8 \\ 2 & 7 & 3 & 0 & 12 \\ 0 & 1 & 9 & 6 & 13 \\ 4 & 0 & 2 & 10 & 15 \\ 7 & 9 & 10 & 14 & -5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 70 \\ 85 \\ 118 \\ 125 \\ 86 \end{bmatrix}$$

Метод	Время (мс)	Кол-во итераций	Невязка
LOS (LU)	4	10	$4.31E - 16$
BCGSTAB (LU)	4	3	$4.27E - 21$

$$\begin{bmatrix} 10 & -4 & 0 & 0 & 0 & 0 & -2 & -3 & 0 & 0 & 0 & 0 \\ -2 & 17 & -6 & 0 & 0 & 0 & 0 & -5 & -4 & 0 & 0 & 0 \\ 0 & -7 & 13 & -2 & 0 & 0 & 0 & 0 & -3 & -1 & 0 & 0 \\ 0 & 0 & -3 & 6 & -1 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & -4 & 10 & -2 & 0 & 0 & 0 & 0 & -1 & -3 \\ 0 & 0 & 0 & 0 & 0 & 8 & -4 & 0 & 0 & 0 & 0 & -4 \\ -5 & 0 & 0 & 0 & 0 & -3 & 10 & -2 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & -2 & 6 & -3 & 0 & 0 & 0 \\ 0 & -2 & -1 & 0 & 0 & 0 & 0 & -1 & 4 & 0 & 0 & 0 \\ 0 & 0 & -4 & -2 & 0 & 0 & 0 & 0 & -3 & 13 & -4 & 0 \\ 0 & 0 & 0 & -3 & -4 & 0 & 0 & 0 & 0 & -5 & 13 & -1 \\ 0 & 0 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & -2 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{bmatrix} = \begin{bmatrix} -36 \\ -62 \\ -20 \\ -10 \\ -25 \\ -28 \\ 31 \\ 6 \\ 21 \\ 39 \\ 49 \\ 22 \end{bmatrix}$$

Метод	Время (мс)	Кол-во итераций	Невязка
LOS (LU)	5	23	$4.39E - 15$
BCGSTAB (LU)	5	8	$4.89E - 15$

Тестирование метода на матрицах большой размерности

Возьмем матрицу, сгенерированную в прошлой лабораторной работе (количество узлов ≈ 500).

Метод	Время (мс)	Кол-во итераций	Невязка
LOS (LU)	20	16	$3.26E - 15$
BCGSTAB (LU)	16	11	$3.53E - 16$

Матрица, сгенерированная в прошлой лабораторной работе (количество узлов ≈ 1400).

Метод	Время (мс)	Кол-во итераций	Невязка
LOS (LU)	54	20	$2.44E - 15$
BCGSTAB (LU)	51	14	$5.56E - 16$

Проведенные исследования и выводы

Из результатов проделанной работы можно сделать вывод, что метод BCGSTAB сходится за меньшее количество итераций, чем LOS, при этом время работы методов близки (разница в несколько миллисекунд). Для решения СЛАУ больших размерностей рекомендуется использовать BCGSTAB.

Тексты основных модулей

Solvers.cs

```
1 namespace eMP_3;
2
3 public abstract class Solver {
4     protected TimeSpan _runningTime;
5     protected SparseMatrix _matrix = default!;
6     protected Vector<double> _vector = default!;
7     protected Vector<double>? _solution;
8     public int MaxIters { get; init; }
9     public double Eps { get; init; }
10    public TimeSpan? RunningTime => _runningTime;
11    public ImmutableArray<double>? Solution => _solution?.ToImmutableArray();
12
13    protected Solver(int maxIters, double eps)
14        => (MaxIters, Eps) = (maxIters, eps);
15
16    public void SetMatrix(SparseMatrix matrix)
17        => _matrix = matrix;
18
19    public void SetVector(Vector<double> vector)
20        => _vector = vector;
21
22    public abstract void Compute();
23
24    protected Vector<double> Direct(Vector<double> vector, double[] gglnew, double[]
25    ↪ dinew) {
26        Vector<double> y = new(vector.Length);
27        Vector<double>.Copy(vector, y);
28
29        double sum = 0.0;
30
31        for (int i = 0; i < _matrix.Size; i++) {
32            int i0 = _matrix.ig[i];
33            int i1 = _matrix.ig[i + 1];
34
35            for (int k = i0; k < i1; k++)
36                sum += gglnew[k] * y[_matrix.jg[k]];
37
38            y[i] = (y[i] - sum) / dinew[i];
39            sum = 0.0;
40        }
41
42        return y;
43    }
44
45    protected Vector<double> Reverse(Vector<double> vector, double[] ggunew) {
46        Vector<double> result = new(vector.Length);
47        Vector<double>.Copy(vector, result);
48
49        for (int i = _matrix.Size - 1; i >= 0; i--) {
50            int i0 = _matrix.ig[i];
51            int i1 = _matrix.ig[i + 1];
52
53            for (int k = i0; k < i1; k++)
```

Приведен единственный класс, который был реализован в предыдущей лабораторной работе.

```

53         result[_matrix.jg[k]] -= ggunew[k] * result[i];
54     }
55
56     return result;
57 }
58
59 protected void LU(double[] gglnew, double[] ggunew, double[] dinew) {
60     double suml = 0.0;
61     double sumu = 0.0;
62     double sumdi = 0.0;
63
64     for (int i = 0; i < _matrix.Size; i++) {
65         int i0 = _matrix.ig[i];
66         int i1 = _matrix.ig[i + 1];
67
68         for (int k = i0; k < i1; k++) {
69             int j = _matrix.jg[k];
70             int j0 = _matrix.ig[j];
71             int j1 = _matrix.ig[j + 1];
72             int ik = i0;
73             int kj = j0;
74
75             while (ik < k && kj < j1) {
76                 if (_matrix.jg[ik] == _matrix.jg[kj]) {
77                     suml += gglnew[ik] * ggunew[kj];
78                     sumu += ggunew[ik] * gglnew[kj];
79                     ik++;
80                     kj++;
81                 } else if (_matrix.jg[ik] > _matrix.jg[kj]) {
82                     kj++;
83                 } else {
84                     ik++;
85                 }
86             }
87
88             gglnew[k] -= suml;
89             ggunew[k] = (ggunew[k] - sumu) / dinew[j];
90             sumdi += gglnew[k] * ggunew[k];
91             suml = 0.0;
92             sumu = 0.0;
93         }
94
95         dinew[i] -= sumdi;
96         sumdi = 0.0;
97     }
98 }
99
100
101 public class LOS : Solver {
102     public LOS(int maxIters, double eps) : base(maxIters, eps) { }
103
104     public override void Compute() {
105         try {
106             ArgumentNullException.ThrowIfNull(_matrix, $"{nameof(_matrix)} cannot be
107             ↪ null, set the matrix");
108             ArgumentNullException.ThrowIfNull(_vector, $"{nameof(_vector)} cannot be
109             ↪ null, set the vector");
110
111             double alpha, beta;
112             double squareNorm;

```

```

111         _solution = new(_vector.Length);
112
113         Vector<double> r = new(_vector.Length);
114         Vector<double> z = new(_vector.Length);
115         Vector<double> p = new(_vector.Length);
116         Vector<double> tmp = new(_vector.Length);
117
118         Stopwatch sw = Stopwatch.StartNew();
119
120         r = _vector - (_matrix * _solution);
121
122         Vector<double>.Copy(r, z);
123
124         p = _matrix * z;
125
126         squareNorm = r * r;
127
128         for (int index = 0; index < MaxIters && squareNorm > Eps; index++) {
129             alpha = p * r / (p * p);
130             _solution += alpha * z;
131             squareNorm = (r * r) - (alpha * alpha * (p * p));
132             r -= alpha * p;
133
134             tmp = _matrix * r;
135
136             beta = -(p * tmp) / (p * p);
137             z = r + (beta * z);
138             p = tmp + (beta * p);
139         }
140
141         sw.Stop();
142
143         _runningTime = sw.Elapsed;
144     } catch (Exception ex) {
145         Console.WriteLine($"We had problem: {ex.Message}");
146     }
147 }
148
149 }
150
151 public class LOSLU : Solver {
152     public LOSLU(int maxIters, double eps) : base(maxIters, eps) { }
153
154     public override void Compute() {
155         try {
156             ArgumentNullException.ThrowIfNull(_matrix, $"{nameof(_matrix)} cannot be
157             ↪ null, set the matrix");
158             ArgumentNullException.ThrowIfNull(_vector, $"{nameof(_vector)} cannot be
159             ↪ null, set the vector");
160
161             double alpha, beta;
162             double squareNorm;
163
164             _solution = new(_vector.Length);
165
166             double[] gglnew = new double[_matrix.ggl.Length];
167             double[] ggunew = new double[_matrix.ggu.Length];
168             double[] dinew = new double[_matrix.di.Length];
169
170             _matrix.ggl.Copy(gglnew);

```

```

169     _matrix.ggu.Copy(ggunew);
170     _matrix.di.Copy(dinew);
171
172     Vector<double> r = new(_vector.Length);
173     Vector<double> z = new(_vector.Length);
174     Vector<double> p = new(_vector.Length);
175     Vector<double> tmp = new(_vector.Length);
176
177     Stopwatch sw = Stopwatch.StartNew();
178
179     LU(gglnew, ggunew, dinew);
180
181     r = Direct(_vector - (_matrix * _solution), gglnew, dinew);
182     z = Reverse(r, ggunew);
183     p = Direct(_matrix * z, gglnew, dinew);
184
185     squareNorm = r * r;
186
187     for (int iter = 0; iter < MaxIters && squareNorm > Eps; iter++) {
188         alpha = p * r / (p * p);
189         squareNorm = (r * r) - (alpha * alpha * (p * p));
190         _solution += alpha * z;
191         r -= alpha * p;
192
193         tmp = Direct(_matrix * Reverse(r, ggunew), gglnew, dinew);
194
195         beta = -(p * tmp) / (p * p);
196         z = Reverse(r, ggunew) + (beta * z);
197         p = tmp + (beta * p);
198     }
199
200     sw.Stop();
201
202     _runningTime = sw.Elapsed;
203 } catch (Exception ex) {
204     Console.WriteLine($"We had problem: {ex.Message}");
205 }
206 }
207 }
208
209 public class BCGSTABLU : Solver {
210     public BCGSTABLU(int maxIters, double eps) : base(maxIters, eps) { }
211
212     public override void Compute() {
213         try {
214             ArgumentNullException.ThrowIfNull(_matrix, $"{nameof(_matrix)} cannot be
215             ↪ null, set the matrix");
216             ArgumentNullException.ThrowIfNull(_vector, $"{nameof(_vector)} cannot be
217             ↪ null, set the vector");
218
219             double alpha = 1.0;
220             double omega = 1.0;
221             double rho = 1.0;
222             double beta, temp;
223
224             double vectorNorm = _vector.Norm();
225
226             _solution = new(_vector.Length);
227
228             double[] gglnew = new double[_matrix.ggl.Length];

```

```

227     double[] ggunew = new double[_matrix.ggu.Length];
228     double[] dinew = new double[_matrix.di.Length];
229
230     _matrix.ggl.Copy(gglnew);
231     _matrix.ggu.Copy(ggunew);
232     _matrix.di.Copy(dinew);
233
234     Vector<double> r = new(_vector.Length);
235     Vector<double> r0 = new(_vector.Length);
236     Vector<double> z = new(_vector.Length);
237     Vector<double> p = new(_vector.Length);
238     Vector<double> v = new(_vector.Length);
239     Vector<double> s = new(_vector.Length);
240     Vector<double> t = new(_vector.Length);
241
242     Stopwatch sw = Stopwatch.StartNew();
243
244     LU(gglnew, ggunew, dinew);
245
246     r = Direct(_vector - (_matrix * _solution), gglnew, dinew);
247
248     Vector<double>.Copy(r, r0);
249
250     for (int iter = 0; iter < MaxIters && r.Norm() / vectorNorm >= Eps;
251         ↪ iter++) {
252         temp = rho;
253         rho = r0 * r;
254         beta = rho / temp * (alpha / omega);
255         p = r + (beta * (p - (omega * v)));
256         v = Direct(_matrix * Reverse(p, ggunew), gglnew, dinew);
257         alpha = rho / (r0 * v);
258         s = r - (alpha * v);
259         t = Direct(_matrix * Reverse(s, ggunew), gglnew, dinew);
260         omega = t * s / (t * t);
261         _solution += (omega * s) + (alpha * p);
262         r = s - (omega * t);
263     }
264
265     _solution = Reverse(_solution, ggunew);
266
267     sw.Stop();
268
269     _runningTime = sw.Elapsed;
270 } catch (Exception ex) {
271     Console.WriteLine($"We had problem: {ex.Message}");
272 }
273 }

```