Нижегородский государственный университет им. Н.И. Лобачевского Факультет вычислительной математики и кибернетики

## Образовательный комплекс «Параллельные численные методы»

## Лекционные материалы

Баркалов К.А.

При поддержке компании Intel

Нижний Новгород 2012

# Содержание

4.	ИТЕРАЦИОННЫЕ МЕТОДЫ РЕШЕНИЯ СЛА	AY3
1. Ба	азовые итерационные методы	4
1.1.	Метод простой итерации	
1.2.	Методы Якоби и Зейделя	6
1.3.	Метод верхней релаксации	7
1.4.	ЧЕБЫШЕВСКОЕ УСКОРЕНИЕ ИТЕРАЦИОННОГО МЕТОДА	
2. П	редобуславливание	
2.1.	ПРЕДОБУСЛАВЛИВАТЕЛИ ЯКОБИ, ЗЕЙДЕЛЯ, SSOR	
2.2.	Общая схема <i>ILU</i> -разложения	
2.3.	<i>ILU</i> -РАЗЛОЖЕНИЕ БЕЗ ЗАПОЛНЕНИЯ ( <i>ILU</i> (0))	
2.4.	УРОВЕНЬ ЗАПОЛНЕНИЯ И ILU(P)-ФАКТОРИЗАЦИЯ	
3. M	етоды крыловского типа	
3.1.	Построение базиса подпространства Крылова	
3.2.	Обобщенный метод минимальных невязок	
3.3.	МЕТОД БИСОПРЯЖЕННЫХ ГРАДИЕНТОВ	
3.4.	Метод сопряженных градиентов	
4. Л	итература	
4.1.	ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ ИНФОРМАЦИИ	
4.2.	Дополнительная литература	
4.3.	Информационные ресурсы сети Интернет	

## 4. Итерационные методы решения СЛАУ

В данном разделе мы рассмотрим методы решения систем линейных уравнений, принципиально отличающиеся от прямых методов поиска точного решения, рассмотренных в предыдущем разделе. Пусть

 $Ax=b \tag{4.1}$ 

– система линейных уравнений относительно неизвестного вектора  $x \in \mathbb{R}^n$  с симметричной положительно определенной матрицей A размерности  $n \times n$  и правой частью вектором  $b \in \mathbb{R}^n$ . Точное решение системы (4.1) обозначим через  $x^*$ .

*Итерационный метод*, предназначенный для решения системы (4.1), генерирует последовательность векторов  $x^{(s)} \in \mathbb{R}^n$ , s=0,1,2,..., каждый из которых может рассматриваться исследователем как приближенное решение системы (4.1). Начальное приближение – вектор  $x^{(0)} \in \mathbb{R}^n$  – задает исследователь. Итерационный метод называется cxodsuumcs, если для любого начального приближения  $x^{(0)} \in \mathbb{R}^n$  последовательность  $x^{(s)} \in \mathbb{R}^n$ , s=0,1,2,..., сходится к точному решению  $x^*$ , т.е.

$$\lim_{s \to \infty} \left\| x^{(s)} - x^* \right\| = 0 \tag{4.2}$$

где через || || обозначена любая векторная норма.

На практике используют два критерия остановки итерационных методов одновременно: остановку по точности и остановку по числу итераций. Первый критерий определяется условием

$$\left\|\boldsymbol{x}^{(s)} - \boldsymbol{x}^{(s-1)}\right\| < \varepsilon_1 \tag{4.3}$$

где  $x^{(s)}$  – приближение, полученное на итерации с номером *s*,  $x^{(s-1)}$  – приближение, полученное на предыдущей итерации. Параметр *moчнocmu метода*  $\varepsilon_1$  задает исследователь. Если условие (4.3) выполнено, метод завершает работу и вектор  $x^{(s)}$  трактуется как приближенное решение системы (4.1). Величина  $\varepsilon_2$ , определяемая как

$$\mathcal{E}_2 = \left\| x^{(s)} - x^{(s-1)} \right\| \tag{4.4}$$

называется *достигнутой точностью метода* и сообщается исследователю.

Второй критерий определяется максимальным числом итераций N, на которое готов пойти исследователь. Если номер итерации, которую предстоит выполнить, превышает N, итерация не выполняется, метод завершает работу, вектор  $x^{(N)}$  трактуется как приближенное решение системы и достигнутая точность метода  $\varepsilon_2$  также сообщается исследователю.

При изучении сходимости итерационных методов и при их реализации в качестве векторной нормы || || обычно используют евклидову норму || ||<sub>2</sub>, норму || ||<sub>1</sub>, определяемую суммой модулей всех компонент вектора, норму || ||<sub>∞</sub>, определяемую максимальным модулем компоненты, а также энергетическую норму || ||<sub>A</sub>, порождаемую симметричной, положительно определенной матрицей *A*. Так как в конечномерных нормированных пространствах указанные нормы эквивалентны, для доказательства сходимости метода или для его реализации можно пользоваться любой из перечисленных норм.

## 1. Базовые итерационные методы

### 1.1. Метод простой итерации

#### 1.1.1. Последовательный алгоритм

Приведем систему (4.1) к виду

$$x=Gx+c$$

причем данное преобразование можно сделать не единственным образом, все будет зависеть от матрицы G и вектора c. Полученное соотношение подсказывает рекуррентное соотношение

$$x^{(s+1)} = Gx^{(s)} + c ,$$

которое является основой *метода простой итерации*. Сам метод заключается в построении последовательности  $x^{(s)}$  (используя данную рекуррентную формулу и начальное приближение  $x^{(0)}$ ).

Известно, что если ||G|| < 1, то последовательность  $x^{(s)}$ , порожденная методом простой итерации, сходится к решению задачи  $x^*$  при любом начальном приближении  $x^{(0)}$ , причем

$$\|x^* - x^{(s)}\| \le \|G\|^s \|x^* - x^{(0)}\|,$$

т.е. условие ||G|| < 1 является достаточным условием сходимости. Необходимым и достаточным условием сходимости является выполнение неравенства

$$\rho(G) < 1, \ \rho(G) = \max \left| \lambda_i(G) \right|.$$

Напомним, что величина  $\rho(G)$  называется спектральным радиусом матрицы G.

Частный случай метода простой итерации определяется формулой

$$\frac{x^{(s+1)} - x^{(s)}}{\tau} + Ax^{(s)} = b ,$$

где  $x^{(s)}$  – текущее приближение,  $x^{(s+1)}$  – следующее приближение,  $\tau \neq 0$  – фиксированное число, являющееся параметром метода. Формула для вычисления нового приближения по предыдущему может быть записана в виде

$$x^{(s+1)} = -\tau(Ax^{(s)} - b) + x^{(s)} = -\tau \cdot r^{(s)} + x^{(s)},$$

где  $r^{(s)} = Ax^{(s)} - b$  – невязка приближения с номером s.

Далее нетрудно записать явные формулы для отыскания компонент нового вектора *x*<sup>(*s*+1)</sup>:

$$x_i^{(s+1)} = -\tau \left( \sum_{j=1}^n a_{ij} x_j^{(s)} - b_i \right) + x_i^{(s)}.$$

Справедлива следующая теорема о сходимости: если матрица A симметрична и положительно определена и параметр метода  $\tau$  лежит в интервале  $(0, \lambda_{max})$ , где  $\lambda_{max}$  – максимальное собственное число матрицы A, метод сходится к точному решению системы (4.1) с любого начального приближения. Известно, что оптимальным значением параметра  $\tau$  является значение

$$\tau^* = \frac{2}{\lambda_{\min} + \lambda_{\max}},$$

где  $\lambda_{min}$  и  $\lambda_{max}$  – минимальное и максимальное собственные числа матрицы *A* соответственно.

Напомним, что погрешностью приближения с номером *s* называют вектор  $z^{(s)}$ , определяемый как

$$z^{(s)} = x^* - x^{(s)}$$

где  $x^*$  – точное решение системы Ax=b. Для метода простой итерации с оптимальным параметром  $\tau^*$  справедлива оценка

. . 1

$$\left\|z^{(s+1)}\right\|_{2} \leq \left(\frac{\mu_{A}-1}{\mu_{A}+1}\right)^{s+1} \left\|z^{(0)}\right\|_{2}.$$

Здесь через || ||<sub>2</sub> обозначена среднеквадратичная норма вектора, а через  $\mu_A$  – число обусловленности матрицы *A*, согласованное с векторной нормой || ||<sub>2</sub> и в силу этого вычисляемое как  $\mu_A = \lambda_{max} / \lambda_{min}$ .

Оценим трудоемкость предложенного алгоритма. Анализ расчетных формул показывает, что они включают одну операцию умножения матрицы на вектор и три операции над векторами (сложение, вычитание, умножение на константу). Общее количество операций, выполняемых на одной итерации, составляет

 $t_1 = 2n(n+1)$ .

Таким образом, выполнение L итераций метода потребует

 $T_1=2n(n+1)L$ 

операций.

#### 1.1.2. Параллельный алгоритм

При распараллеливании итерационных методов линейной алгебры (в частности, метода простой итерации) в первую очередь следует учесть, что выполнение итераций метода осуществляется последовательно и, тем самым, наиболее целесообразный подход состоит в распараллеливании вычислений, реализуемых в ходе выполнения итераций.

Анализ последовательного метода простой итерации показывает, что основные затраты на *s*-й итерации – порядка  $O(n^2)$  операций – состоят в умножении матрицы *A* на вектор текущего приближения  $x^{(s)}$ . Как результат, при организации параллельных вычислений могут быть использованы известные методы параллельного умножения матрицы на вектор (например, параллельный алгоритм матрично-векторного умножения при ленточном горизонтальном разделении матрицы). Дополнительные вычисления, имеющие порядок сложности O(n), представляют собой операции сложения и умножение на скаляр для векторов. Организация таких вычислений также может быть выполнена в многопоточном режиме.

Несмотря на простоту распараллеливания, данный метод редко применяется, т.к. он обладает существенно меньшей скоростью сходимости по сравнению с иными методами, к рассмотрению которых мы переходим.

## 1.2. Методы Якоби и Зейделя

Рассмотрим систему (4.1) с симметричной, положительно определенной матрицей A размера  $n \times n$ . Обозначим через D диагональную матрицу  $n \times n$  такую, что ее главная диагональ совпадает с главной диагональю матрицы A. Через L обозначим нижнюю треугольную матрицу  $n \times n$  такую, что ее ненулевые (поддиагональные) элементы также совпадают с элементами A, а главная диагональ является нулевой. Аналогично обозначим через R верхнюю треугольную матрицу  $n \times n$ , ненулевые (наддиагональные) элементы которой совпадают с элементами A, а главная диагональ также является нулевой. В этом случае для A справедливо представление в виде

$$A = L + D + R. \tag{4.5}$$

Используя данные обозначения, сформулируем два метода, являющихся частными случаями метода простой итерации.

Метод Якоби определяется в соответствии с выражением

$$Dx^{(s+1)} = (-L - R)x^{(s)} + b,$$

что равносильно

$$x^{(s+1)} = D^{-1}(-L-R)x^{(s)} + D^{-1}b.$$

В покомпонентной форме это будет соответствовать

$$x_{i}^{(s+1)} = \left( b_{i} - \sum_{j=1}^{i-1} a_{ij} x_{j}^{(s)} - \sum_{j=i+1}^{n} a_{ij} x_{j}^{(s)} \right) \middle/ a_{ii} .$$
(4.6)

Данные формулы получаются из следующих соображений: *i*-я компонента (s+1)-го приближения выражается через остальные (n-1)-у компоненты текущего *s*-го приближения из уравнения системы.

Известно, что если матрица А удовлетворяет условию строгого диагонального преобладания, т.е. выполняется

$$|a_{ii}| > \sum_{j=1}^{i-1} |a_{ij}| + \sum_{j=i+1}^{n} |a_{ij}|$$

то метод Якоби сходится при любом начальном приближении  $x^{(0)}$ .

*Метод Зейделя* получается из метода Якоби следующим образом: будем использовать в правой части формулы (4.6) уже посчитанные компоненты вектора следующего приближения  $x^{(s+1)}$ . Получим рекуррентное соотношение

$$x_{i}^{(s+1)} = \left(b_{i} - \sum_{j=1}^{i-1} a_{ij} x_{j}^{(s+1)} - \sum_{j=i+1}^{n} a_{ij} x_{j}^{(s)}\right) / a_{ii} .$$
(4.7)

#### 1.3. Метод верхней релаксации

Метод верхней релаксации (SOR – Successive Over Relaxation) является представителем стационарных одношаговых итерационных методов линейной алгебры и записывается в виде

$$\frac{(D+\omega L)(x^{(s+1)}-x^{(s)})}{\omega} + Ax^{(s)} = b.$$
(4.8)

Здесь  $x^{(s)}$  – приближение, полученное на итерации с номером *s*,  $x^{(s+1)}$  – следующее приближение,  $\omega$  – число (параметр метода), матрицы *A*, *L*, *D* и вектор *b* определены выше.

Необходимым условием сходимости метода релаксации с любого начального приближения  $x^{(0)}$  к точному решению задачи  $x^*$  является выполнение условия  $\omega \in (0,2)$ . Если же матрица A симметрична и положительно определена, то выполнение данного условия является также и достаточным. При этом если  $\omega \in (0,1)$ , то говорят о *методе нижней релаксации*, а при  $\omega \in (1,2)$  – о *методе верхней релаксации*, при  $\omega = 1$  метод релаксации будет совпадать с известным *методом Зейделя*.

Скорость сходимости метода верхней релаксации определяется выбором параметра  $\omega$ . Известно [4], что при решении некоторых классов разреженных систем уравнений, метод Зейделя требует  $O(n^2)$  итераций, а при надлежащем выборе итерационного параметра  $\omega$  метод будет сходиться за O(n) итераций.

В общем случае нет аналитической формулы для вычисления оптимального параметра  $\omega_{opt}$ , обеспечивающего наилучшую сходимость. Например, для решения систем уравнений, возникающих при аппроксимации дифференциальных уравнений в частных производных, можно использовать эвристические оценки вида

$$\omega_{ont} \approx 2 - O(h)$$
.

где *h* – шаг сетки, на которой проводилась дискретизация.

В некоторых случаях можно более точно оценить оптимальный параметр. Известной оценкой [20] является выражение

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho^2 (D^{-1}(R+L))}},$$

где  $\rho$  – спектральный радиус матрицы, а *R*, *D*, *L* – из (4.5). Возникающий при этом вопрос об оценке спектрального радиуса может быть решен численно, например, при помощи степенного метода [2].

Также следует отметить, что для ряда задач, возникающих при решении задач математической физики методом сеток, оптимальные параметры метода верхней релаксации найдены аналитически (подробнее об этом см. в п. 4.3).

## 1.3.1. Последовательный алгоритм

Получим формулы для отыскания  $x^{(s+1)}$  по предыдущему приближению  $x^{(s)}$  в явном виде.

$$(D + \omega L)(x^{(s+1)} - x^{(s)}) + \omega A x^{(s)} = \omega b ,$$
  
$$Dx^{(s+1)} + \omega L x^{(s+1)} - Dx^{(s)} - \omega L x^{(s)} + \omega A x^{(s)} = \omega b ,$$
  
$$Dx^{(s+1)} = -\omega L x^{(s+1)} + Dx^{(s)} - \omega (A - L) x^{(s)} + \omega b .$$

 $\langle - \rangle$ 

 $\langle \cdot \rangle$ 

(---1)

С учетом того, что A-L=R+D, получаем

$$Dx^{(s+1)} = -\omega Lx^{(s+1)} + (1-\omega)Dx^{(s)} - \omega Rx^{(s)} + \omega b .$$

Далее нетрудно записать явные формулы для отыскания компонент нового вектора *x*<sup>(*s*+1)</sup>:

$$a_{ii}x_i^{(s+1)} = -\omega \sum_{j=1}^{i-1} a_{ij}x_j^{(s+1)} + (1-\omega)a_{ii}x_i^{(s)} - \omega \sum_{j=i+1}^n a_{ij}x_j^{(s)} + \omega b_i$$
(4.9)

Как следует из формулы (4.9), при подсчете *i*-й компоненты нового приближения все компоненты, индекс которых меньше *i*, берутся из нового приближения  $x^{(s+1)}$ , а все компоненты, индекс которых больше либо равен *i* – из старого приближения  $x^{(s)}$ . Таким образом, после того, как *i*-я компонента нового приближения вычислена, *i*-я компонента старого приближения вычислена, *i*-я компонента старого приближения компоненты с индексом, меньшим или равным *i*, будут использоваться не будет. Напротив, для подсчета следующих компонент вектора  $x^{(s+1)}$  компоненты с индексом, меньшим или равным *i*, будут использоваться «в новой версии». В силу этого обстоятельства для реализации метода достаточно хранить только одно (текущее) приближение  $x^{(s)}$ , а при расчете следующего приближения  $x^{(s+1)}$  использовать формулу (4.9) для всех компонент по порядку и постепенно обновлять вектор  $x^{(s)}$ .

#### 1.3.2. Организация параллельных вычислений

При распараллеливании метода верхней релаксации также применим подход, который состоит в распараллеливании вычислений, реализуемых в ходе выполнения итераций.

Анализ последовательного алгоритма показывает, что основные затраты на *s*-й итерации – порядка  $O(n^2)$  операций – заключаются в операциях матричного умножения  $Lx^{(s+1)}$  и  $Rx^{(s+1)}$ . Однако напрямую распараллелить эти операции не представляется возможным, т.к. в методе верхней релаксации не только итерации осуществляются последовательно, но и вычисление компонент вектора очередного приближения  $x^{(s+1)}$  также осуществляется последовательно, начиная с первой.

Применение ленточного разделения данных, аналогично методу простой итерации, приведет к изменению вычислительной схемы алгоритма. Поэтому одним из возможных способов распараллеливания, сохраняющем в точности последовательность действий метода, состоит в распараллеливании операций, необходимых для получения одной компоненты вектора нового приближения. При этом распараллелить можно вычисление сумм в формуле (4.9).

#### 1.3.3. Результаты вычислительных экспериментов

Вычислительные эксперименты для оценки эффективности параллельного варианта метода верхней релаксации проводились при условиях, указанных во введении. С целью формирования симметричной положительно определенной матрицы элементы подматрицы L генерировались в диапазоне от 0 до 1, значения элементов подматрицы R получались из симметрии матриц L и R, а элементы на главной диагонали (подматрица D) генерировались в диапазоне от n до 2n, где n – размер матрицы.

В качестве критерия остановки использовался критерий остановки по точности (4.3) с параметром  $\varepsilon = 10^{-6}$ ; а итерационный параметр  $\omega = 1.1$ . Во всех экспериментах метод нашел решение с требуемой точностью за 11 итераций. Как и для предыдущих экспериментов, ускорение будем фиксировать по сравнению с параллельной программой, запущенной в один поток.

		Параллельный алгоритм							
n	1 поток	2 пот	гока	4 пот	гока	6 пот	оков	8 пот	гоков
		Т	S	Т	S	Т	S	Т	S
2500	0,73	0,47	1,57	0,30	2,48	0,25	2,93	0,22	3,35
5000	3,25	2,11	1,54	1,22	2,67	0,98	3,30	0,80	4,08
7500	7,72	5,05	1,53	3,18	2,43	2,36	3,28	1,84	4,19
10000	14,60	9,77	1,50	5,94	2,46	4,52	3,23	3,56	4,10
12500	25,54	17,63	1,45	10,44	2,45	7,35	3,48	5,79	4,41
15000	38,64	26,36	1,47	15,32	2,52	10,84	3,56	8,50	4,54

Таблица 1. Результаты экспериментов (метод верхней релаксации)



Рис. 1. Ускорение параллельного метода верхней релаксации

Эксперименты демонстрируют неплохое ускорение (порядка 4 на 8-ми потоках).

### 1.4. Чебышевское ускорение итерационного метода

Пусть система *Ах*=*b* с помощью какого-либо метода (например, метода верхней релаксации) преобразована в итерационный процесс

$$x^{(s+1)} = Gx^{(s)} + c \tag{4.10}$$

Если данный процесс сходящийся, т.е.  $\rho(G) < 1$ , то получаемая последовательность векторов будет сходиться к истинному решению  $x^*$ 

$$\lim_{s\to\infty}x^{(s)}=x^*$$

Предположим теперь, что проведено *m* итераций метода и получены вектора  $x^{(0)}, x^{(1)}, ..., x^{(m)}$ , каждый из которых является приближением  $x^*$ . Поставим перед собой задачу: найти линейную комбинацию данных векторов

$$y^{(m)} = \sum_{i=0}^{m} \alpha_i x^{(i)}, \qquad (4.11)$$

которая приближает  $x^*$  лучше, чем  $x^{(m)}$ .

Так как  $y^{(m)}$  является приближением к  $x^*$ , то  $y^{(m)}$  должен совпадать с  $x^*$  в случае  $x^{(0)} = x^{(1)} = ... = x^{(m)} = x^*$ . Отсюда сразу следует, что коэффициенты  $\alpha_i$  должны удовлетворять условию

$$\sum_{i=0}^{m} \alpha_i = 1.$$

Рассмотрим теперь, как при этом ограничении выбрать  $\alpha_i$  так, чтобы минимизировать погрешность для  $y^{(m)}$ .

Погрешность приближения у<sup>(m)</sup> можно записать как

$$y^{(m)} - x^* = \sum_{i=0}^m \alpha_i x^{(i)} - x^* = \sum_{i=0}^m \alpha_i (x^{(i)} - x^*) =$$
$$= \sum_{i=0}^m \alpha_i G^i (x^{(0)} - x^*) = p_m(G)(x^{(0)} - x^*),$$

где  $p_m(G) = \sum_{i=0}^m \alpha_i G^i$  есть многочлен *m*-й степени такой, что

 $p_m(1) = \sum_{i=0}^{m} \alpha_i = 1$ . Таким образом, погрешность вектора  $y^{(m)}$  зависит от

спектрального радиуса матрицы  $p_m(G)$  – чем он будет меньше, тем меньше будет и погрешность.

Конечно, идеальной является ситуация  $\rho(p_m(G)) = 0$ , однако задача отыскания многочлена  $p_m$  с данным свойством является трудноразрешимой. Например, по теореме Гамильтона-Кэли [17] указанным свойством обладает характеристический многочлен матрицы R. Но для его нахождения нужно найти все собственные числа матрицы G, что можно сделать аналитически лишь в некоторых специальных случаях, а численное нахождение собственных значений едва ли проще исходной задачи. Поэтому мы не будем искать многочлен  $p_m$ , для которого  $\rho(p_m(G)) = 0$ , а вместо этого постараемся насколько возможно приблизить к нулю значение спектрального радиуса матрицы  $p_m(G)$ .

Предположим, что матрица перехода G обладает следующими свойствами (что естественным образом выполняется для многих итерационных методов):

- все собственные значения матрицы *G* вещественны;
- эти собственные значения находятся в отрезке  $[-\rho, \rho]$ , где  $0 < \rho < 1$ .

Тогда можно попытаться найти многочлен  $p_m$ , такой, что

- $p_m(1) = 1;$
- $\max_{-\rho < x < \rho} |p_m(x)|$  имеет наименьшее возможное значение среди всех полиномов *m*-й степени.

Поскольку собственными значениями матрицы  $p_m(G)$  являются числа  $p_m(\lambda(G))$  (по теореме об отображении спектра [17]), эти собственные значения будут малы, а потому будет мал и спектральный радиус (как наибольший из модулей собственных значений).

Построение многочлена  $p_m$ , удовлетворяющего сформулированным условиям – это классическая задача теории приближений, решение которой опирается на *многочлены* Чебышева.

Многочлен Чебышева  $T_m(x)$  степени *m* определяется с помощью рекуррентного соотношения

$$T_0(x) = 1, \ T_1(x) = x,$$
  
$$T_m(x) = 2xT_{m-1}(x) - T_{m-2}(x).$$
(4.12)

Многочлены Чебышева обладают замечательным свойством:  $T_m(x)$  является многочленом степени *m* со старшим коэффициентом  $2^{m-1}$ , который меньше всего отклоняется от нуля на интервале [-1,1] среди всех возможных многочленов с такой степенью и старшим коэффициентом.

Многочленом с теми свойствами, которые нам нужны, является

$$p_m(x) = T_m(x/\rho)/T_m(1/\rho)$$

Действительно, из свойств полиномов Чебышева следует, что  $p_m(1) = 1$ ; а если  $x \in [-\rho, \rho]$ , то  $|p_m(x)| \le 1/T_m(1/\rho)$ , что является малой величиной.

Для того чтобы предложенный механизм стал практической процедурой ускорения, для вычисления искомого приближения  $y^{(m)}$  нужно предложить более эффективный способ, чем по формуле (4.11) (мы предполагаем, что *m* большое, поэтому хранить и обращаться ко всем предыдущим приближениям  $x^{(m)}$ ,  $0 \le i \le m$ , будет крайне неэффективно).

Оказывается, трехчленное рекуррентное соотношение в определении полиномов Чебышева позволяет хранить и комбинировать только три вектора  $y^{(m)}, y^{(m-1)}, y^{(m-2)}$ , а не все предыдущие вектора  $x^{(m)}, 0 \le i \le m$ .

Введем в рассмотрение коэффициент  $\mu_m \equiv 1/T_m(1/\rho)$ , тогда  $p_m(G) = \mu_m T_m(G/\rho)$  и согласно рекуррентному соотношению (4.12)

$$\frac{1}{\mu_m} = \frac{2}{\rho\mu_{m-1}} - \frac{1}{\mu_{m-2}}.$$
(4.13)

Затем подставим все в выражение для невязки

$$y^{(m)} - x^* = p_m(G)(x^{(0)} - x^*) = \mu_m T_m \left(\frac{G}{\rho}\right)(x^{(0)} - x^*) =$$
$$= \mu_m \left[2\frac{G}{\rho}T_{m-1}\left(\frac{G}{\rho}\right) - T_{m-2}\left(\frac{G}{\rho}\right)\right](x^{(0)} - x^*) =$$

$$= \mu_m \left[ 2 \frac{G}{\rho} \frac{p_{m-1}(G/\rho)}{\mu_{m-1}} - \frac{p_{m-2}(G/\rho)}{\mu_{m-2}} \right] (x^{(0)} - x^*) =$$
$$= \mu_m \left[ 2 \frac{G}{\rho} \frac{y^{(m-1)} - x^*}{\mu_{m-1}} - \frac{y^{(m-2)} - x^*}{\mu_{m-2}} \right]$$

Отсюда получаем

$$y^{(m)} = \frac{2\mu_m}{\mu_{m-1}} \frac{G}{\rho} y^{(m-1)} - \frac{\mu_m}{\mu_{m-2}} y^{(m-2)} + d_m,$$

где

$$d_{m} = x^{*} - \frac{2\mu_{m}}{\mu_{m-1}} \frac{G}{\rho} x^{*} - \frac{\mu_{m}}{\mu_{m-2}} x^{*}$$

Так как G является переходной матрицей итерационного процесса, то

$$x^* = Gx^* + c \, .$$

Используя данное соотношение, получим

$$d_{m} = x^{*} - \frac{2\mu_{m}}{\mu_{m-1}} \frac{x^{*} - c}{\rho} - \frac{\mu_{m}}{\mu_{m-2}} x^{*} = x^{*} \mu_{m} \left( \frac{1}{\mu_{m}} - \frac{2}{\rho \mu_{m-1}} + \frac{1}{\mu_{m-2}} \right) + \frac{2\mu_{m}}{\rho \mu_{m-1}} c$$

Но по определению числа  $\mu_m$  имеем, что

$$\frac{1}{\mu_m} - \frac{2}{\rho \mu_{m-1}} + \frac{1}{\mu_{m-2}} = 0,$$

откуда получаем финальное соотношение

$$y^{(m)} = \frac{2\mu_m}{\mu_{m-1}} \frac{G}{\rho} y^{(m-1)} - \frac{\mu_m}{\mu_{m-2}} y^{(m-2)} + \frac{2\mu_m}{\rho\mu_{m-1}} c .$$

Таким образом, чебышевское ускорение итерационного процесса

$$x^{(s+1)} = Gx^{(s)} + c$$

состоит в следующем:

- положить  $\mu_0 = 1; \ \mu_1 = \rho; \ y^{(0)} = x^{(0)}; \ y^{(1)} = Gx^{(0)} + c$
- для всех последующих *m*=2, 3, ... вычислить

$$\mu_{m} = \left(\frac{2}{\rho\mu_{m-1}} - \frac{1}{\mu_{m-2}}\right)^{-1}$$
$$y^{(m)} = \frac{2\mu_{m}}{\rho\mu_{m-1}} \left(Gy^{(m-1)} + c\right) - \frac{\mu_{m}}{\mu_{m-2}} y^{(m-2)}$$
(4.14)

К сожалению, описанный алгоритм нельзя непосредственно применить к методу верхней релаксации при решении системы Ax=b. Дело в том, что матрица итерационного процесса метода верхней релаксации G в общем случае имеет комплексные собственные значения, а чебышевское ускорение требует, чтобы собственные значения матрицы G были вещественны и принадлежали отрезку  $[-\rho,\rho]$ . Однако ситуацию можно исправить с помощью симметричного метода верхней релаксации (SSOR – Simmetric Successive Over Relaxation).

Как следует из названия, данный метод является модификацией обычного метода верхней релаксации (SOR). Один шаг SSOR для вычисления (s+1)-го приближения  $x^{(s+1)}$  состоит из двух этапов:

- выполнить шаг метода SOR, вычисляя компоненты промежуточного приближения *x*<sup>(s+1/2)</sup> прямом порядке;
- выполнить шаг метода SOR, вычисляя компоненты нового приближения *x*<sup>(s+1)</sup> обратном порядке.

В матричной форме данные шаги можно записать как

$$(D + \omega L)x^{(s+1/2)} = (1 - \omega)Dx^{(s)} - \omega Rx^{(s)} + \omega b,$$
  
(D + \omega U)x^{(s+1)} = (1 - \omega)Dx^{(s+1/2)} - \omega Lx^{(s+1/2)} + \omega b. (4.15)

Исключая отсюда  $x^{(s+1/2)}$  можно получить матрицу *G* и вектор *c* для записи метода в форме (4.10)

$$G = (D + \omega U)^{-1} ((1 - \omega)D - \omega L)(D + \omega L)^{-1} ((1 - \omega)D - \omega R), \qquad (4.16)$$
$$c = (D + \omega U)^{-1} (E + ((1 - \omega)D - \omega L)(D + \omega L)^{-1})\omega b,$$

где E – единичная матрица. Известно [9], что в случае симметричной матрицы A матрица G также будет симметричной, а значит, будет иметь вещественные значения. Значит, для метода SSOR возможно применение чебышевского ускорения.

При практической реализации метода SSOR нет необходимости вычислять G и c явно. В самом деле, формула (4.14) содержит не саму матрицу G и вектор c, а их комбинацию  $Gy^{(m-1)} + c$ . Тогда вычисления по формуле (4.14) можно провести в два этапа:

• сначала, используя расчетную схему (4.15), вычислить промежуточное значение

$$\overline{y} = G y^{(m-1)} + c;$$

• затем, используя найденное значение  $\overline{y}$ , получить новое приближение

$$y^{(m)} = \frac{2\mu_m}{\rho\mu_{m-1}} \,\overline{y} - \frac{\mu_m}{\mu_{m-2}} \, y^{(m-2)} \, .$$

Еще раз отметим, что для выполнения одного шага метода потребуется хранить только два предыдущих приближения, что допустимо с точки зрения объема используемой памяти.

#### 1.4.1. Результаты вычислительных экспериментов

Вычислительные эксперименты для оценки эффективности параллельного варианта метода верхней релаксации проводились при условиях, указанных во введении. В отличие от п. 1.3.3, выполним данную серию экспериментов на разреженных системах.

Разреженные матрицы систем линейных уравнений были взяты из коллекции матриц университета Флориды [30]. Характеристики используемых в экспериментах матриц (название, размер *n*, число ненулевых элементов *nz*, оценка число обусловленности  $\mu_A$ ) приведены в таблица 2. (все матрицы являются симметричными положительно определенными).

Название	п	nz	$\mu_{A}$
mesh1em6	48	306	6.1
bcsstk04	132	3648	$2.3*10^{6}$
bcsstk05	153	2423	$1.4*10^{4}$
bcsstk09	1083	18437	9.5*10 <sup>3</sup>
chem97ZtZ	2541	7361	$2.5*10^2$

Таблица 2. Характеристики используемых матриц

Для примера на рис. 2 приведен портрет одной из матриц.



Рис. 2. Портрет матрицы bcsstk05

Системы уравнений с конкретной матрицей А формировались следующим образом:

- 1. Выбиралось точное решение (например, единичный вектор  $x^* = \{x_i = 1, 1 \le i \le n\}$ ).
- 2. На основе точного решения вычислялась правая часть  $b = Ax^*$ .
- 3. Получалась система уравнений Ax=b с известным решением  $x^*$ .

В качестве критерия остановки итерационного метода использовался критерий остановки по точности (4.3) с параметром  $\varepsilon = 10^{-6}$ . Итерационный параметры выбрать оптимальным образом для матрицы произвольной структуры очень сложно (кроме некоторых задач специального вида, речь о которых пойдет ниже), поэтому для определения наилучшей сходимости метода параметры  $\omega$  и  $\rho$  определялись экспериментально, подбором. Все методы достигли требуемой точности  $10^{-6}$ , число выполненных при этом итераций метода *s* для SOR и SSOR-Cheb приведены в таблица 3.

Название	SO	R	SOR-Cheb		
задачи	ω	S	ω	ρ	S
mesh1em6	1.9	146	1.9	0.9	35

Таблица 3. Результаты работы методов SOR и SSOR-Cheb на матрицах из коллекции университета Флориды

bcsstk04	1.9	341	1.09	0.99	229
bcsstk05	1.87	986	1.0	0.998	251
bcsstk09	1.95	885	1.11	0.999	537
chem97ZtZ	1.9	144	1.9	0.9	125

Результаты экспериментов показывают, что число итераций метода, требуемых для достижения заданной точности, снижается (от 1.5 до 3.5 раз) при применении методов верхней релаксации и симметричной верхней релаксации с чебышевским ускорением.

Заключительную серию экспериментов проведем на задачах большого размера, которые соответствует решению системы уравнений, возникающей при дискретизации уравнения Пуассона. Подробнее о численном решении дифференциальных уравнений в частных производных см. Раздел 6, здесь же кратко отметим следующие основные моменты:

- можно сформировать тестовую задачу с заранее известным точным решением;
- матрица СЛАУ имеет пятидиагональный портрет (см. рис. 3); •





для данного типа задач известна [4] формула для оптимального па-• раметра  $\omega$ 

$$\omega_{opt} = \frac{2}{1 + 2\sin(\pi h/2)},$$
(4.17)

где h – шаг сетки;

для данного типа задач известна [9] формула для оптимального параметра *ρ* как спектрального радиуса матрицы *G* из (4.16)

$$\rho_{opt} = 1 - \frac{\pi h}{2} \,, \tag{4.18}$$

где *h* – шаг сетки;

В качестве критерия остановки итерационного метода использовался критерий остановки по точности (4.3) с параметром  $\varepsilon = 10^{-6}$ , параметр  $\omega$  выбирался оптимальным образом по формуле (4.17), параметр чебышевского метода был оценен по формуле (4.18) как  $\rho = 0.99$ . Все методы достигли требуемой точности  $10^{-6}$ , число выполненных при этом итераций метода *s* для SOR, SSOR и SSOR-Cheb приведены в таблица 4.

S nz/n п ω SOR SSOR SSOR-Cheb  $4,9.10^{-6}$ 10000 1.9397 286 342 53 22500  $9.8 \cdot 10^{-6}$ 1.9592 428 512 65  $3,1.10^{-7}$ 40000 1.9692 569 682 72 62500  $1.2 \cdot 10^{-7}$ 1.9753 711 852 123 90000  $6.1 \cdot 10^{-8}$ 1.9793 1022 91 853 122500  $3.3 \cdot 10^{-8}$ 1.9823 995 1192 85  $1.9 \cdot 10^{-8}$ 160000 1.9845 1137 1362 97 143 202500  $1,2.10^{-8}$ 1.9862 1278 1532  $7,9.10^{-9}$ 250000 1.9875 1420 1702 276

Таблица 4. Результаты работы методов SOR, SSOR и SSOR-Cheb на пятидиагональной матрице уравнения Пуассона

Результаты экспериментов показывают, что использование метода с чебышевским ускорением позволяет значительно уменьшить число итераций, необходимых для достижения одной и той же точности, по сравнению с исходным методом верхней релаксации.

## 2. Предобуславливание

Предобуславливание – явная или неявная модификация системы линейных уравнений *Ах=b*, которая упрощает решение данной системы с помощью того или иного метода. Например, домножение строк матрицы *А* на значения, обратные элементам матрицы на главной диагонали, с целью получения единичной главной диагонали – явная форма предобуславливания.

Указанное масштабирование может снизить число итераций метода (хотя и не гарантирует подобное снижение).

Другим примером предобуславливания является домножение исходной системы некоторую матрицу  $M^{-1}$ , т.е. переход к системе вида

$$M^{-1}Ax = M^{-1}b. (4.19)$$

Матрицу *М* называют *матрицей предобуславливателя*, или же просто *предобуславливателем*. Данный способ предобуславливания можно отнести к неявным, т.к. на практике явного изменения системы линейных уравнений не производится, а матрица предобуславливателя используется внутри итерационного метода, для коррекции некоторых шагов алгоритма.

Анализируя соотношение (4.19), можно сформулировать неформальное требование к матрице предобуславливателя: M должна быть близка к A (чтобы  $M^{-1}A$  была бы близка к единичной матрице E). Отметим, что выбор M=A сразу же дает решение  $Ex = A^{-1}b$ , однако не имеет практического смысла, т.к. требует вычисления  $A^{-1}$ .

Вычисление матричного произведения  $M^{-1}A$  в общем случае приведет к изменению портрета матрицы (что для разреженных систем является крайне нежелательными), поэтому для использования предобуславливателя применяется другой подход. В схему метода вводятся некоторые дополнительные операции, которые позволяют учесть влияния предобуславливателя. Например, невязка исходной системы r = b - Ax связана с невязкой предобусловленной системы  $\bar{r} = M^{-1}b - M^{-1}Ax$  очевидным соотношением

$$M\bar{r} = r \,. \tag{4.20}$$

Аналогичное соотношение справедливо и для произведения матрицы на вектор. Пусть z = Ax, и  $\bar{z} = M^{-1}Ax$ . Тогда

$$M\overline{z} = z$$
.

Сводя вместе все требования к эффективному предобуславливанию, получаем, что матрица М:

- должна быть близка к А;
- должна быть легко вычислима;
- должна допускать быстрое решение систем вида (4.20).

Так как данные три критерия являются вообще говоря противоречивыми (как мы уже убедились на примере M=A), выбрать способ предобуславливания, удовлетворяющий всем критериям сразу является непростой задачей.

Одним из наиболее эффективных способов построения предобуславливателя является *неполное* LU-разложение (ILU-факторизация) исходной матрицы A. Неполное разложение предполагает представление матрицы A в форме

$$A=LU+R$$
,

где матрицы L и U являются нижней и верхней треугольной матрицами с портретами, совпадающими (или же почти совпадающими) с нижним и верхним треугольниками матрицы A, а матрица R является невязкой факторизации. После проведения *ILU*-факторизации в качестве предобуславливателя можно использовать

$$M=LU$$
,

данная матрица удовлетворяет (или же почти удовлетворяет) всем трем требованиям к предобуславливателю.

Ниже мы рассмотрим несколько способов предобуславливания, начиная от простейших (Якоби, Зейделя) и заканчивая более эффективными *ILU*(0) и *ILU*(*p*).

### 2.1. Предобуславливатели Якоби, Зейделя, SSOR

В п.1.1 был сформулирован общий вид метода простой итерации

$$x^{(s+1)} = Gx^{(s)} + c \tag{4.21}$$

для решения системы вида (4.1). Используя представление матрицы A в виде разности двух матриц M и N, т.е. A=M-N, можно записать

$$G = M^{-1}N = M^{-1}(M - A) = E - M^{-1}A, \qquad (4.22)$$
$$c = M^{-1}b.$$

Например, для методов Якоби и Гаусса-Зейделя матрица перехода G представляется в виде

$$G_J(A) = E - D^{-1}A,$$
  
 $G_{GS}(A) = E - (D + L)^{-1}A,$ 

где *E* – единичная матрица, *D* – диагональная матрица, совпадающая с диагональю матрицы *A*, а *L* – нижняя треугольная матрица, совпадающая с нижним треугольником матрицы *A* без главной диагонали.

Решение, полученное методом простой итерации (4.21), который был сформирован нами для решения системы (4.1), может быть проинтерпретировано как решение системы

$$(E-G)x = c,$$

которая, с учетом (4.22), может быть преобразована к виду

$$M^{-1}Ax = M^{-1}b$$

В итоге получаем систему, которая получается из исходной с помощью предобуславливателя *M*, а итерационный метод есть ни что иное, как метод простой итерации, примененный к системе уравнений с предобуславливателем *M*. Для рассмотренных нами методов Якоби, Гаусса-Зейделя, верхней и симметричной верхней релаксации данные матрицы равны соответственно

$$M_{J} = D,$$

$$M_{GS} = D + L,$$

$$M_{SOR} = \frac{1}{\omega} (D + \omega L),$$

$$M_{SSOR} = \frac{1}{\omega(2 - \omega)} (D + \omega L) D^{-1} (D + \omega R),$$

Здесь *D* – диагональ *A*, *L* – нижний треугольник *A* без главной диагонали, *R* – верхний треугольник *A* без главной диагонали.

В случае использования SOR- и SSOR-предобуславливателя нет необходимости выбирать параметр  $\omega$  так же тщательно, как и для соответствующих итерационных методов. Можно использовать, например,  $\omega=1$ , что в первом случае приводит нас к предобуславливателю Гаусса-Зейделя  $M_{GS}$ , а во втором – к симметричному предобуславливателю Гаусса-Зейделя

$$M_{SGS} = (D+L)D^{-1}(D+R)$$
.

Очевидно, что рассмотренные предобуславливатели  $M_J$ ,  $M_{GS}$ ,  $M_{SGS}$  удовлетворяют двум из трех требований к предобуславливателям: они являются легко вычислимыми (т.к. являются подматрицами или же произведением подматриц матрицы A) и обратимыми (т.к. являются диагональной, треугольной и произведением треугольных матриц соответственно). Однако они не в полной мере удовлетворяют третьему свойству, т.е. не являются «близкими» к A, и, как следствие, спектральные характеристики матрицы  $M^{-1}A$  улучшаются не очень сильно.

В качестве примера рассмотрим предобуславливание матрицы A, получающейся при численном решении уравнения Пуассона на сетке 5×5 (подробнее о дискретизации дифференциальных уравнений в частных производных см. Раздел 6). Здесь размер матрицы – n=25, число ненулевых эле-

ментов – nz=105, число обусловленности  $\mu_A = 20.7$ . На рис. 4 приведен портрет матрицы A.



Рис. 4. Портрет матрицы А для уравнения Пуассона

Будем использовать симметричное предобуславливание Гаусса-Зейделя в соответствии с формулой  $M_{SGS} = (D+L)D^{-1}(D+R)$ . Портрет получившегося предобуславливателя приведен на рис. 5.



Рис. 5. Предобуславливатель *M*<sub>SGS</sub>

Применение предобуславливателя Гаусса-Зейделя улучшает спектральное число обусловленности примерно в 4 раза:

Исходная система: cond(A)=20.7;

 $M_{SGS}$ : cond( $M^{-1}A$ )=5.1.

Схожая картина наблюдается и при использовании предобуславливателя Гаусса-Зейдля для задачи Пуассона на сетках большего размера. Например, на сетке  $40 \times 40$  матрица *A* имеет размер *n*=1600, число ненулевых элементов *nz*=7840, а обусловленность изменяется следующим образом:

Исходная система: cond(A)=989;

 $M_{SGS}$ : cond( $M^{-1}A$ )=210.

Итак, применение предобуславливателей, основанных на идеях метода SSOR, улучшает характеристики решаемой задачи в разы, но не на порядки. Более хорошие результаты показывают методы, основанные на неполном *LU*-разложении, к рассмотрению которых мы и переходим.

#### 2.2. Общая схема ILU-разложения

Рассмотрим разреженную матрицу  $A = \{a_{i,j} : i, j = 1,...,n\}$ , и найдем ее представление в виде

A=LU-R,

где матрицы L и U являются нижней и верхней треугольной матрицей соответственно, а невязка R удовлетворяет некоторым дополнительным условиям (например, определенные элементы в R равны 0). В этом случае приближенное представление  $A \approx LU$  называется *неполным* LU*разложением* матрицы A (или ILU-*факторизацией*, incomplete LU). Сформулируем алгоритм ILU-разложения матрицы A, основываясь на полном LU-разложении (методе Гаусса). Полученный нами алгоритм может применяться для матриц любого вида, однако гарантировать существование ILU-разложения можно лишь для некоторого класса матриц – Mматриц.

**Определение**. Матрица *А* называется *М*-матрицей, если она удовлетворяет следующим четырем свойствам:

- 4.  $a_{ii} > 0$  для i = 1, ..., n.
- 5.  $a_{ii} \leq 0$  для  $i, j = 1, \dots, n$ .
- 6. Матрица A невырожденная (т.е. существует обратная матрица  $A^{-1}$ )
- 7. Все элементы  $A^{-1}$  неотрицательные.

Известно, что для данного класса матриц выполняется следующее утверждение.

**Утверждение**. [21] Пусть *А* – *М*-матрица, а матрица *A*<sub>1</sub> получена на первом шаге метода исключения Гаусса для матрицы *A*. Тогда *A*<sub>1</sub> также является *М*-матрицей.

Из данного утверждения следует, что матрица размера  $(n-1) \times (n-1)$ , полученная удалением первой строки и первого столбца в  $A_1$ , также является *М*-матрицей.

Предположим теперь, что некоторые элементы вне главной диагонали матрицы  $A_1$  были отброшены в процессе гауссова исключения (т.е. были приравнены к нулю). В результате мы получим матрицу  $\overline{A_1}$ ,

$$\overline{A}_{1} = A_{1} + R$$
,

где элементы матрицы R удовлетворяют условиям

$$r_{ii} = 0, r_{ij} \ge 0, i, j = 1, \dots, n.$$

Таким образом, элементы матрицы  $\overline{A}_1$  не меньше, чем элементы  $A_1$ , и  $\overline{A}_1$  также является *М*-матрицей. Указанный процесс можно повторить для матрицы  $\overline{A}_1$  (2:*n*, 2:*n*), и т.д., в итоге получим искомое неполное *LU* разложение матрицы *A*.

При описании алгоритма элементы, которые отбрасываются, не конкретизированы. Это можно сделать заранее, задав некоторый портрет отбрасываемых элементов. Данный портрет может быть произвольным, с одним лишь ограничением: не могут быть отброшены диагональные элементы. Таким образом, для произвольного портрета *P* нулевых элементов

$$P \subset \{(i, j) : i \neq j, 1 \le i, j \le n\}$$

*ILU*-разложение (*ILU*<sub>*P*</sub>) может быть получено в соответствии со следующим алгоритмом.

Алгоритм. Общая схема *ILU<sub>P</sub>* -разложения

1. for *k*=1,..., *n*−1 do

- 2. for i=k+1,...,n and if  $(i,k) \notin P$  do
- 3.  $a_{ik} = a_{ik}/a_{kk}$
- 4. for j=k+1,...,n and for  $(i,j) \notin P$  do

$$5. a_{ij} = a_{ij} - a_{ik} * a_{kj}$$

6. end 
$$j$$

7. enf *i* 

8. end *k* 

**Утверждение**. Пусть *А* – *М*-матрица, а *Р* – заданный портрет обнуляемых (отбрасываемых) элементов. Тогда описанный алгоритм является корректным, а результатом его работы является *ILU*-разложение матрицы *А* 

$$A=LU-R$$
.

Доказательство. В соответствии с алгоритмом, на каждом шаге мы получаем

$$\overline{A}_k = A_k + R_k, \ A_k = L_k \overline{A}_{k-1}$$

где

$$L_k = E - \frac{1}{a_{kk}^{(k)}} \begin{pmatrix} 0_k \\ A(k+1:n,k) \end{pmatrix} e_k^T.$$

Здесь использованы обозначения  $0_k$  для нулевого вектора из k компонент, A(m:n, j) для вектора, состоящего их компонент  $a_{ij}$ , i=m,...n, матрицы A.

Отсюда следует соотношение

$$\overline{A}_k = L_k \overline{A}_{k-1} + R_k ,$$

Применяя данное соотношение начиная с первого шага алгоритма, получим

$$\overline{A}_{n-1} = L_{n-1} \dots L_1 A + L_{n-1} \dots L_2 R_1 + \dots + L_{n-1} R_{n-2} + R_{n-1}$$

Обозначив

$$L = (L_{n-1}...L_1)^{-1}, U = \overline{A}_{n-1},$$

получим

$$U = L^{-1}A + S ,$$

где

$$S = L_{n-1} \dots L_2 R_1 + \dots + L_{n-1} R_{n-2} + R_{n-1}.$$

Отметим, что на шаге k алгоритма элементы отбрасываются только в правой нижней подматрице размером  $(n-k)\times(n-k)$  матрицы  $A_k$ . Следовательно, первые k строк и столбцов матрицы  $R_k$  будут нулевыми, и можно записать равенство

$$L_{n-1}...L_{k+1}R_k = L_{n-1}...L_1R_k$$
.

Значит, матрица S может быть представлена в виде

$$S = L_{n-1}...L_2(R_1 + ... + R_{n-2} + R_{n-1}).$$

Обозначив

$$R = R_1 + \ldots + R_{n-2} + R_{n-1},$$

получаем искомое разложение

$$A=LU-R$$
,

где  $(LU)^{-1} = U^{-1}L^{-1}$  и *R* являются матрицами с неотрицательными элементами. Утверждение доказано.

Итак, нами сформирован общий алгоритм *ILU*-разложения. Отметим, что при реализации на компьютере алгоритм в форме, рассмотренной выше, не будет эффективным, т.к. на k-м шаге алгоритма должны быть модифицированы все строки с номерами от (k+1) до n. Однако можно взять в качестве основы другой вариант метода исключения Гаусса, который обеспечивает построчное формирование искомого разложения.

Алгоритм. Общая схема *ILU<sub>P</sub>* -разложения

1. for *i*=2,..., *n* do

- 2. for  $k=1,\ldots, i-1$  and if  $(i,k) \notin P$  do
- 3.  $a_{ik} = a_{ik}/a_{kk}$
- 4. for j=k+1,...,n and for  $(i,j) \notin P$  do

$$5. a_{ij} = a_{ij} - a_{ik} * a_{kj}$$

- 6. end j
- 7. enf *k*

8. end *j* 

В соответствии с алгоритмом 2, на каждом шаге в *i*-ю строку матрицы A записываются значения *i*-х строк факторов L и U (главную диагональ L можно не хранить, т.к. по построению она – единичная). Предыдущие строки матриц L и U с номерами 1, 2, ..., *i*–1 используются при вычислениях, но не модифицируются. Алгоритм 2 может быть легко применен к разреженным матрицам в формате CSR, т.к. вычисление L и U происходит построчно.

Утверждение. [21] Матрицы *L* и *U*, полученные в результате работы алгоритма 2, удовлетворяют условию

$$A=LU-R$$
,

где – R есть матрица, состоящая из элементов, отброшенных в процессе неполной факторизации. Если  $(i, j) \in P$ , тогда элемент  $r_{ij}$  совпадает со значением – $a_{ij}$ , полученном по завершении цикла по k в алгоритме. Иначе элемент  $r_{ij}$  равен нулю.

## 2.3. ILU-разложение без заполнения (ILU(0))

Для удобства последующего изложения введем следующие обозначения:  $a_{i*} - i$ -я строка матрицы A; NZ(A) – множество индексов (i, j),  $1 \le i, j \le n$ , таких, что  $a_{ii} \ne 0$ .

Неполное LU разложение без заполнения (обозначаемое так же ILU(0)) состоит в использовании в качестве портера P нулевых элементов разложения портрет нулевых элементов матрицы A. Иными словами, надо найти разложение

$$A=LU-R$$
,

где матрицы в правой части удовлетворяют следующим свойствам:

- *L* и *U* являются нижнетреугольной с единичной диагональю и верхнетреугольной соответственно;
- *L* и *U* не содержат новых (по сравнению с матрицей *A*) ненулевых элементов;
- Элементы матрицы *R* равны 0 на множестве индексов *NZ*(*A*), т.е. матричное произведение *LU* точно воспроизводит матрицу *A* на множестве индексов *NZ*(*A*).

Данные требования не определяют ILU(0)-разложение однозначно. Вообще говоря, существует бесконечное множество различных пар матриц L и U, удовлетворяющих данным трем условиям. Одно из таких разложений может быть получено из общего алгоритма с использованием портрета нулевых элементов матрицы A в качестве портера P нулевых элементов разложения.

Алгоритм 3. ILU(0)-факторизация

1. for *i*=2,..., *n* do 2. for *k*=1,..., *i*-1 and for (*i*,*k*) $\in$ NZ(A) do 3.  $a_{ik} = a_{ik}/a_{kk}$ 4. for *j*=*k*+1,..., *n* and for (*i*,*j*) $\in$ NZ(A) do

 $5. a_{ij} = a_{ij} - a_{ik} * a_{kj}$ 

6. 
$$\operatorname{end} j$$

7. end *k* 

8. end *j* 

В качестве конкретного примера работы алгоритма рассмотрим факторизацию матрицы

$$A = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix}.$$

Сначала выполним полное *LU*-разложение, получим (с точностью до третьего знака)

$$A = LU = \begin{bmatrix} 1 & & & \\ -0.25 & 1 & & \\ -0.25 & -0.067 & 1 & \\ 0 & -0.0267 & -0.286 & 1 \end{bmatrix} \begin{bmatrix} 4 & -1 & -1 & 0 \\ 3.75 & -0.25 & -1 \\ 3.733 & -1.067 \\ 3.429 \end{bmatrix}$$

Затем выполним *ILU*(0)-разложение, получим (с точностью до третьего знака)

$$A \approx IL^*IU$$
,

где

$$IL = \begin{bmatrix} 1 & & & \\ -0.25 & 1 & & \\ -0.25 & 0 & 1 & \\ 0 & -0.267 & -0.267 & 1 \end{bmatrix} IU = \begin{bmatrix} 4 & -1 & -1 & 0 \\ 3.75 & 0 & -1 \\ & 3.75 & -1 \\ & & 3.467 \end{bmatrix}.$$

При этом ошибка разложения будет равна

$$A - IL * IU = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -0.25 & 0 \\ 0 & -0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Пример показывает, что значения элементов матриц L и U отличаются от соответствующих элементов матриц IL и IU.

В качестве примера большего масштаба вернемся к предобуславливанию матрицы *A*, получающейся при численном решении уравнения Пуассона на сетке 5×5. Напомним, что размер матрицы – n=25, число ненулевых элементов – nz=105, число обусловленности  $\mu_A = 20.7$ , портрет матрицы *A* приведен на рис. 4.

Как и в предыдущем примере, выполним сначала полное *LU*-разложение (подробнее об алгоритмах полной факторизации разреженных матриц см. Раздел 3), результирующие портреты приведены на рис. 6. Как и следовало ожидать, матрицы *L* и *U* оказались ленточными (произошло заполнение).



**Рис. 6.** Портреты матриц *L* и *U* (полное *LU*-разложение)

Затем проведем ILU(0)-разложение, получим матрицы IL и IU; их портреты представлены на рис. 7. В соответствии с алгоритмом портреты матриц L и U в точности совпадают с портретами соответствующих подматриц матрицы A.



**Рис. 7.** Портреты матриц *IL* и *IU* (*ILU*(0)-разложение)

Портрет матрицы невязки ILU(0)-разложения R = LU - A представлен на рис. 8. В данном случае портрет является симметричным, т.к. исходная матрица A была симметричной, и неполное LU-разложение будет совпадать с неполным разложением Холецкого.



Рис. 8. Портрет матрицы невязки *ILU*(0)-разложения

После проведенного неполного разложения оценим влияние предобуславливания на решаемую систему уравнений. Для этого сравним обусловленность исходной матрицы A, и матрицы  $M^{-1}A$ :

Исходная система: cond(A)=20.7;

$$ILU(0): \operatorname{cond}(M^{-1}A) = 3.6.$$

Обусловленность системы уравнений улучшилась более чем в 5 раз.

Аналогичный эффект улучшения обусловленности наблюдается и для задачи Пуассона на сетках большего размера. На сетке  $40 \times 40$  матрица *A* имеет размер *n*=1600, число ненулевых элементов *nz*=7840, а обусловленность после применения *ILU*(0)-предобуславливателя улучшилась почти в 7 раз:

Исходная система: cond(A)=989;

 $ILU(0): \operatorname{cond}(M^{-1}A) = 143.$ 

#### 2.4. Уровень заполнения и *ILU(p)*-факторизация

Рассмотренный в предыдущем пункте метод ILU(0) обладает рядом достоинств: не требуется перераспределение памяти: портреты матриц L и U совпадают с портретами нижнего и верхнего треугольника матрицы A, заполнение отсутствует – как следствие, алгоритм имеет невысокую трудоемкость. Однако при этом применение ILU(0)-предобуславливателей может не давать нужного ускорения сходимости для итерационных методов. Неполное LU-разложение, более близкое к полному LU-разложению, может обеспечить более высокую скорость сходимости. Подобные методы отличаются от ILU(0)-факторизации тем, что допускают некоторое увеличение заполненности факторов L и U. Ниже будет рассмотрен один из подходов к построению подобного рода алгоритмов – ILU(p)-факторизация, здесь целое число p обозначает степень заполнения (в ILU(0) заполнение отсутствует).

Введем в рассмотрение понятие *уровня заполнения* для элементов  $a_{ij}$  разреженной матрицы A. Уровень заполнения должен изменяться для элементов в процессе факторизации, и отбрасываться (обнуляться) должны элементы с высоким уровнем заполнения. При этом уровень заполнения должен зависеть и от величины самого элемента  $a_{ij}$  – возникающим в процессе факторизации малым элементам (которые можно отбросить без внесения значительной погрешности) должен соответствовать большой уровень заполнения.

Очень простой моделью, соответствующей данным требованиям, является следующая: пусть элементу со значением порядка  $\varepsilon^k$ ,  $0 < \varepsilon < 1$ , соответствует уровень заполнения k.

Пи этом в начале факторизации ненулевым элементам матрицы A присваивается уровень заполнения 1, а нулевым элементам –  $\infty$ . Элементы матрицы обновляются в соответствии с формулой (шаг 5 алгоритма)

$$a_{ij} = a_{ij} - a_{ik} * a_{kj}$$

Пусть  $l_{ij}$  – текущий уровень заполнения для элемента  $a_{ij}$ . Тогда, в соответствии с принятой моделью, величина обновленного элемента должна быть порядка

$$\varepsilon^{l_{ij}} - \varepsilon^{l_{ik}} * \varepsilon^{l_{kj}} = \varepsilon^{l_{ij}} - \varepsilon^{l_{ik}+l_{kj}}$$

Немного огрубив данную оценку, можно сказать, что величину элемента можно оценить максимумом из величин  $\mathcal{E}^{l_{ij}}$  и  $\mathcal{E}^{l_k+l_{kj}}$ , следовательно, уровень заполнения обновленного элемента будет

$$l_{ij} = \min\{l_{ij}, l_{ik} + l_{kj}\}.$$

Чтобы данная формула соответствовала заполнению (точнее, отсутствию заполнения) в методе ILU(0), уровни заполнения должны быть уменьшены на единицу. В итоге получаем следующее определение.

**Определение**. Начальный уровень заполнения элемента  $a_{ij}$  разреженной матрицы *A* определяется как

$$l_{ij} = \begin{cases} 0, \text{если } a_{ij} \neq 0 \text{ или } i = j, \\ \infty, \text{ иначе.} \end{cases}$$

Каждый раз когда данный элемент изменяется на шаге 5 алгоритма, его уровень заполнения должен быть изменен в соответствии с формулой

$$l_{ii} = \min\{l_{ii}, l_{ik} + l_{ki} + 1\}.$$

В соответствии с определением, уровень заполнения является невозрастающей характеристикой. Таким образом, элемент  $a_{ij} \neq 0$  в начале будет иметь уровень заполнения 0 и в процессе факторизации. Введенное определение позволяет дать формальное правило отбрасывания элементов: в методе *ILU(p)*-факторизации все элементы с уровнем заполнения, большим *p*, должны быть отброшены (обнулены). Портрет нулевых элементов для фактора может быть задан в форме

$$P = \{(i, j) : l_{ii} > p\},\$$

где  $l_{ij}$  – уровень заполнения после всех преобразований. Случай p=0 соответствует алгоритму *ILU*(0).

При реализации ILU(p) обычно выделяют символическую часть алгоритма, во время которой происходит определение структуры матриц L и U. После выполнения символической части происходит численная факторизация, во время которой заполняются числовые значения матриц L и U. Ниже представлена схема алгоритма, в которой символическая и численная части объединены.

#### Алгоритм. *ILU*(*p*)-факторизация

1. Для всех  $a_{ii} \neq 0$  установить  $l_{ii} = 0$ 2. for *i*=2,..., *n* do 3. for k=1,...,i-1 and for  $l_{ik} \leq p$  do 4.  $a_{ik} = a_{ik}/a_{kk}$ for i=k+1,...,n and for  $(i,j)\in NZ(A)$  do 5. 6.  $a_{ij} = a_{ij} - a_{ik} * a_{kj}$  $l_{ij} = \min\{l_{ij}, l_{ik} + l_{kj} + 1\}$ 7. 7. end *j* 7. end k8. Обнулить все элементы в *i*-й строке, для которых  $l_{ii} > p$ 9. end *i* 

Можно выделить несколько трудностей, которые возникают при реализации данного алгоритма. Во-первых, степень заполнения и вычислительная сложность ILU(p)-факторизации не могут быть заранее предсказаны для p>0. Во-вторых, вычисление уровней заполнения может быть довольно трудоемкой операцией. И в-третьих, для знаконеопределенных матриц уровень заполнения может не являться хорошим эквивалентом величины

элементов матрицы, которые отбрасываются при факторизации. То есть, алгоритм может отбросить большие элементы, и как следствие, неполная факторизация станет неточной в том смысле, что невязка R=LU-A не будет достаточно малой. В конечном итоге это приводит к недостаточному ускорению сходимости итерационных методов с использованием ILU(p)-предобуславливателя.

В качестве примера использования ILU(p)-разложения вернемся к рассмотрению матрице A, получающейся при численном решении уравнения Пуассона на сетке 5×5. Напомним, что размер матрицы -n=25, число ненулевых элементов -nz=105, число обусловленности  $\mu_A = 20.7$ , портрет матрицы A приведен на рис. 4. Для матрицы A выполним ILU(1)-разложение, получим матрицы IL и IU; их портреты представлены на рис. 9.



**Рис. 9.** Портреты матриц *IL* и *IU* (*ILU*(1)-разложение)

Сравнение данных матриц с результатами ILU(0)-разложения, представленными на рис. 7, показывает, что в матрицах IL и IU возникла дополнительная диагональ. При этом матрица невязки также стала более заполненной, достаточно сравнить портрет невязки ILU(0)-разложения на рис. 8 с портретом невязки ILU(1) на рис. 10.

Увеличение коэффициента заполнения фактора позволяет ожидать лучших спектральных характеристик новой матрицы. И в самом деле, применение ILU(1)-предобуславливателя уменьшает обусловленности в 2 раза по сравнению с ILU(0):

Исходная система: cond(A)=20.7;

*ILU*(0): cond( $M^{-1}A$ )=3.6; *ILU*(1): cond( $M^{-1}A$ )=1.5.



Рис. 10. Портрет матрицы невязки *ILU*(1)-разложения

Аналогичный эффект улучшения обусловленности наблюдается и для задачи Пуассона на сетках большего размера. Например, на сетке  $40 \times 40$  матрица A имеет размер n=1600, число ненулевых элементов nz=7840, а обусловленность после применения ILU(1)-предобуславливателя улучшилась в 18 раз:

> Исходная система: cond(A)=989;  $ILU(0): cond(M^{-1}A)=143$ ;  $ILU(1): cond(M^{-1}A)=54$ .

Подводя итоги, отметим, что использование предобуславливателя улучшает спектральные характеристики решаемой задачи, при этом надо учитывать требования к памяти, использование ILU(p) разложения при больших значениях параметра р может привести к значительному заполнению фактора.

## 3. Методы крыловского типа

Итерационные методы, основанные на использовании подпространств Крылова (определение будет дано ниже), используются для решения систем вида (4.1). При этом предполагается, что сама матрица A не изменяется в процессе решения, а самая трудоемкая операция, которую можно использовать – это вычисление произведения матрицы на вектор, т.е. по заданному x можно определить y=Ax.

Существует множество различных методов, основанных на использовании крыловских подпространств. Некоторые из них пригодны для несимметричных матриц, другие требуют от матрицы симметрии или положитель-

ной определенности. Некоторые методы для несимметричных матриц предполагают, что могут вычисляться не только произведения типа Ax, но и произведения типа  $A^Tx$ .

**Определение**. Подпространством Крылова размерности *m*, порожденным вектором *v* и матрицей *A* называют линейное пространство

$$K_m = K_m(A, v) = span\{v, Av, A^2v, ..., A^{m-1}v\}.$$
(4.23)

В методах крыловского типа в качестве вектора *v* обычно выбирают нормированную невязку  $r_0/||r_0||$  начального приближения  $x_0$ , где  $r_0 = b - Ax_0$ .

### 3.1. Построение базиса подпространства Крылова

Найдем ортонормированный базис  $\{v_1, v_2, ..., v_m\}$  подпространства Крылова  $K_m(A, v)$ . Так как при построении подпространства использовалась нормированная невязка, то в качестве первого базисного вектора можно взять вектор v, т.е.  $v_1$ =v. Найдем на его основе все остальные базисные вектора. По определению (4.23)

$$K_m = span\{w_1, w_2, ..., w_m\},\$$

где

$$w_1 = v_1, \ w_2 = Aw_1, \dots, \ w_m = Aw_{m-1}.$$

Непосредственно использовать вектора  $\{w_1, w_2, ..., w_m\}$  в качестве базисных нельзя, т.к. они не являются ортонормированными. Требуется по системе векторов  $\{w_1, w_2, ..., w_m\}$  построить базис  $\{v_1, v_2, ..., v_m\}$ , т.е. провести *ортогонализацию*.

Предположим, что уже построены k базисных векторов (еще раз отметим, что нам известен первый базисный вектор  $v_1$ ). Тогда следующий базисный вектор  $v_{k+1}$  можно выразить как линейную комбинацию предыдущих базисных векторов и вектора  $Av_k$ :

$$v_{k+1} = Av_k - \sum_{i=1}^k \alpha_i v_i \; .$$

Так как новый базисный вектор  $v_{k+1}$  должен быть ортогонален ко всем предыдущим, то должны выполняться равенства

$$(v_{k+1}, v_i) = 0, \ 1 \le j \le k$$

Подставив в последнее равенство выражение для *v*<sub>k+1</sub>, получаем

$$(Av_k - \sum_{i=1}^k \alpha_i v_i, v_j) = 0, \ 1 \le j \le k,$$

или же

$$(Av_k, v_j) = \sum_{i=1}^k \alpha_i(v_i, v_j), \ 1 \le j \le k$$

Так как по предположению вектора  $\{v_1, v_2, ..., v_k\}$  – ортонормированны, то

$$(v_i, v_j) = \begin{cases} 0, i \neq j \\ 1, i = j \end{cases}, \ 1 \le i, j \le k ,$$

что позволяет легко получить выражения для коэффициентов  $\alpha_i$ 

$$\alpha_i = (Av_k, v_i), \ 1 \le i \le k$$

Данная вычислительная процедура, которая называется *ортогонализацией Арнольди*, может быть оформлена в виде следующего алгоритма.

#### Алгоритм (ортогонализация Арнольди).

Выбрать нормированный вектор  $v_1$  (например,  $v_1 = r_0 / || r_0 ||$ )

for j=2, 3, ..., m do  
for i=1, 2, ..., j do 
$$h_{ij} = (Av_i, v_j)$$
  
 $w_j = Av_j - \sum_{i=1}^j h_{ij} v_i$   
 $h_{j+1,j} = || w_j ||$   
if  $h_{j+1,j} = 0$  then стоп; else  $v_{j+1} = w_j / h_{j+1,j}$ 

end j

Запишем алгоритм ортогонализации в матричной форме. Обозначим

$$V_m = \begin{bmatrix} v_1, v_2, \dots, v_m \end{bmatrix}$$

матрицу, составленную из ортонормированных базисных векторов. Согласно алгоритму, для заданной размерности *m* вычисляется (*m*+1)-н вектор; припишем дополнительный вектор  $v_{m+1}$  к квадратной матрице  $V_m$ , получим расширенную прямоугольную матрицу  $V_{m+1}$ :

$$\overline{V}_m = [v_1, v_2, \dots, v_m, v_{m+1}]$$

Коэффициенты ортогонализации  $h_{ij}$  также объединим в виде одной квадратной матрицы  $H_m$ , дополнив в ней недостающие позиции нулями. Непосредственно из алгоритма Арнольди следует, что матрица  $H_m$  является верней матрицей Хессенберга<sup>1</sup>, соответственно, она имеет вид

$$H_{m} = \begin{vmatrix} h_{11} & h_{12} & h_{13} & h_{14} & \dots & h_{1m} \\ h_{21} & h_{22} & h_{23} & h_{24} & \dots & h_{2m} \\ 0 & h_{32} & h_{33} & h_{34} & \dots & h_{3m} \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & \dots & 0 & h_{m-1,m-2} & h_{m-1,m-1} & h_{m-1,m} \\ 0 & \dots & 0 & 0 & h_{m,m-1} & h_{m,m} \end{vmatrix}$$

Аналогично расширению матрицы  $V_m$ , матрицу  $H_m$  также можно расширить, добавив в нее последнюю строку с элементом  $h_{m+1,m}$ , получим матрицу  $\overline{H}_m$ .

$$\overline{H}_{m} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & \dots & h_{1m} \\ h_{21} & h_{22} & h_{23} & h_{24} & \dots & h_{2m} \\ 0 & h_{32} & h_{33} & h_{34} & \dots & h_{3m} \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & \dots & 0 & h_{m-1,m-2} & h_{m-1,m-1} & h_{m-1,m} \\ 0 & \dots & 0 & 0 & h_{m,m-1} & h_{m,m} \\ 0 & \dots & 0 & 0 & h_{m+1,m} \end{bmatrix}.$$

Можно показать, что справедливы следующие соотношения:

$$AV_m = V_m H_m + w_m e_m^T = \overline{V}_m \overline{H}_m, \qquad (4.24)$$
$$V_m^T A V_m = H_m.$$

которые впоследствии будут использованы при построении обобщенного метода минимальных невязкок.

Алгоритм ортогонализации Арнольди, рассмотренный ранее, для построения каждого нового вектора  $v_k$  требует (k-1) вычисления скалярного произведения. Однако если отказаться от условия ортогональности базисных

<sup>&</sup>lt;sup>1</sup> Напомним, что матрицами Хессенберга называют «почти» треугольные матрицы; верхняя матрица Хессенберга – это квадратная матрица, у которой все элементы лежащие ниже первой поддиагонали равны нулю, т.е. *h<sub>i</sub>=*0 для всех *i>j*+1.

векторов в пользу некоторого более слабого условия биортогональности, можно построить менее трудоемкий алгоритм.

Определение. Системы векторов  $\{v_1, v_2, ..., v_m\}$  и  $\{w_1, w_2, ..., w_m\}$  называются биортогональными, если  $(v_i, w_i) = 0$  при  $i \neq j$ .

Очевидно, что выбор  $v_i = w_i$ ,  $1 \le i \le m$ , приводит нас к обычному условию ортогональности.

Алгоритм биортогонализации Ланцоша выполняет построение биортогональных базисов  $\{v_1, v_2, ..., v_m\}$  и  $\{w_1, w_2, ..., w_m\}$  для двух подпространств Крылова

$$K_m(A, v_1) = span\{v_1, Av_1, A^2v_1, \dots, A^{m-1}v_1\}$$

И

$$K_m(A^T, w_1) = span\{w_1, Aw_1, A^2w_1, \dots, A^{m-1}w_1\}$$

Алгоритм биортогонализации Ланцоша

Положить  $\delta_1 = \beta_1 = 0, v_0 = w_0 = 0$ 

Выбрать вектора  $v_1$ ,  $w_1$  такие, что  $(v_1, w_1) = 1$ 

for *j*=1,...,*m* do

$$\begin{aligned} \alpha_{j} &= (Av_{j}, w_{j}) \\ \overline{v}_{j+1} &= Av_{j} - \alpha_{j}v_{j} - \delta_{j}v_{j-1} \\ \overline{w}_{j+1} &= A^{T}w_{j} - \alpha_{j}w_{j} - \delta_{j}w_{j-1} \\ \delta_{j+1} &= \sqrt{|(\overline{v}_{j+1}, \overline{w}_{j+1})|} \text{. Если } \delta_{j+1} = 0 \text{ то Стоп.} \\ \beta_{j+1} &= (\overline{v}_{j+1}, \overline{w}_{j+1})/\delta_{j+1} \\ w_{j+1} &= \overline{w}_{j+1}/\beta_{j+1} \\ v_{j+1} &= \overline{v}_{j+1}/\delta_{j+1} \end{aligned}$$

end j

Отметим, что скалярные множители  $\delta_{j+1}$  и  $\beta_{j+1}$  могут быть выбраны не единственным способом: их выбор должен обеспечивать выполнение равенства  $(v_{j+1}, w_{j+1}) = 1$ . Поэтому, в соответствии с алгоритмом, данные параметры должны удовлетворять условию

$$\delta_{j+1}\beta_{j+1} = (\overline{v}_{j+1}, \overline{w}_{j+1}).$$

Чтобы записать основные соотношения, описывающие алгоритм, в матричной форме, рассмотрим трехдиагональную матрицу  $T_m$ 

$$T_{m} = \begin{bmatrix} \alpha_{1} & \beta_{2} & & \\ \delta_{2} & \alpha_{2} & \beta_{3} & \\ & \ddots & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_{m} \\ & & & \delta_{m} & \alpha_{m} \end{bmatrix}.$$
(4.25)

Аналогично методу Арнольди, можно показать, что выполняются следующие условия:

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^T;$$
  

$$AW_m = W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T;$$
  

$$W_m^T A V_m = T_m.$$

## 3.2. Обобщенный метод минимальных невязок

Рассмотрим систему линейных уравнений (4.1) с невырожденной матрицей A размера  $n \times n$  (в отличие от рассмотренных ранее методов матрица может быть несимметричной и знаконеопределенной).

Идея метода обобщенных минимальных невязок состоит в следующем. Рассмотрим подпространство Крылова  $K_m$ , построенное с использованием нормированного вектора начальной невязки  $v = r_0/||r_0||$ 

$$K_m = K_m(A, v) = span\{v, Av, A^2v, ..., A^{m-1}v\}.$$

Любой вектор x из подпространства  $x_0 + K_m$  может быть представлен в форме

$$x = x_0 + V_m y,$$

где  $V_m$  – матрица, составленная из векторов ортонормированного базиса подпространства Крылова  $K_m$ , а y – вектор размерности m. Тогда невязку уравнения можно определить как функцию вектора y

$$J(y) = \|b - Ax\| = \|b - A(x_0 + V_m y)\|.$$
(4.26)

А используя соотношение (4.24), получаем

$$b - Ax = r_0 - AV_m y = \beta v_1 - \overline{V}_m \overline{H}_m y = \overline{V}_m (\beta e_1 - \overline{H}_m y).$$

Так как столбцы матрицы  $\overline{V}_m$  составляют ортонормированную систему векторов, то

$$J(y) = \|b - A(x_0 + V_m y)\| = \|\beta e_1 - \overline{H}_m y\|.$$

Метод обобщенных минимальных невязок аппроксимирует точное решение системы (4.1) вектором  $x_m$  из подпространства  $x_0 + K_m$ , который минимизирует невязку (4.26). Вектор  $x_m$  может быть получен как

$$x_m = x_0 + V_m y_m,$$

где

$$y_m = \arg\min_{y} \left\| \beta e_1 - \overline{H}_m y \right\|.$$

Вектор  $y_m$  как решение задачи минимизации может быть найден как решение линейной задачи наименьших квадратов размера  $(m+1) \times m$ , где  $m \ll n$ . Сказанное позволяет сформировать алгоритм следующим образом.

Алгоритм (обобщенный метод минимальных невязок, GMRes)

Вычислить  $r_0 = b - Ax_0$ ,  $\beta = ||r_0||$  и  $v_1 = r_0/\beta$ .

Задать нулевую матрицу  $\overline{H}_m$  размера (*m*+1)×*m*.

for *j*=1,2,...,*m* do

Вычислить  $w_i = Av_i$ 

Для  $i=1,2,\ldots,j$  вычислить  $h_{ij}=(w_j,v_i), w_j=w_j-h_{ij}v_i$ 

Вычислить  $h_{j+1,j} = \|w_j\|$ .

if  $h_{i+1,i} = 0$  then *m*=*j* и перейти на шаг (\*)

Вычислить  $v_{i+1} = w_i / h_{i+1,i}$ 

end j

(\*) Вычислить  $y_m = \arg\min_{y} \left\| \beta e_1 - \overline{H}_m y \right\|$  и  $x_m = x_0 + V_m y_m$ .

Вычисление у<sub>т</sub> предполагает решение системы линейных уравнений

$$H_m y = \beta e_1$$
,

которая является переопределенной (число строк больше числа столбцов) и поэтому должна решаться в смысле наименьших квадратов. В соответствии с алгоритмом ортогонализации Арнольди матрица  $\overline{H}_m$  имеет верхнюю хессенбергову форму, поэтому для решения задачи наименьших квадратов проще всего будет привести матрицу к верхнему треугольному виду с помощью вращений Гивенса [18]. Пусть матрица вращения записана в виде

$$\Omega_i = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -s_i & s_i & & \\ & & -s_i & c_i & & \\ & & & & 1 & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix}$$

где  $c_i^2 + s_i^2 = 1$ . На шаге *m* алгоритма матрица  $\Omega_i$  имеет размеры  $(m+1) \times (m+1)$ , а коэффициенты  $c_i$ ,  $s_i$  выбираются таким образом, чтобы обнулить коэффициент  $h_{i+1,i}$ .

В качестве примера рассмотрим данную задачу при m=4. Матрица  $\overline{H}_4$  и правая часть  $\overline{g}_0$  системы линейных уравнений имеют вид

$$\overline{H}_{4} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ & h_{32} & h_{33} & h_{34} \\ & & h_{43} & h_{44} \\ & & & & h_{54} \end{bmatrix}, \ \overline{g}_{0} = \beta e_{1} = \begin{bmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Умножив их на матрицу

$$\Omega_{1} = \begin{bmatrix} c_{1} & s_{1} & & \\ -s_{1} & c_{1} & & \\ & & 1 & \\ & & & 1 \\ & & & & 1 \end{bmatrix}, \text{ где } s_{1} = \frac{h_{21}}{\sqrt{h_{11}^{2} + h_{21}^{2}}}, c_{1} = \frac{h_{11}}{\sqrt{h_{11}^{2} + h_{21}^{2}}}$$

получим матрицу  $\overline{H}_{4}^{(1)}$  и правую часть  $\overline{g}_{1}$ 

$$\overline{H}_{4}^{(1)} = \begin{bmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & h_{14}^{(1)} \\ & h_{22}^{(1)} & h_{23}^{(1)} & h_{24}^{(1)} \\ & h_{32} & h_{33} & h_{34} \\ & & h_{43} & h_{44} \\ & & & & h_{54} \end{bmatrix}, \ \overline{g}_{1} = \begin{bmatrix} c_{1}\beta \\ -s_{1}\beta \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

И так далее, до получения матрицы  $\overline{H}_4^{(4)}$  и правой части  $\overline{g}_4$ 

$$\overline{H}_{4}^{(4)} = \begin{bmatrix} h_{11}^{(4)} & h_{12}^{(4)} & h_{13}^{(4)} & h_{14}^{(4)} \\ & h_{22}^{(4)} & h_{23}^{(4)} & h_{24}^{(4)} \\ & & h_{33}^{(4)} & h_{34}^{(4)} \\ & & & h_{44}^{(4)} \\ & & & & 0 \end{bmatrix}, \ \overline{g}_{4} = \begin{bmatrix} \gamma_{1} \\ \gamma_{2} \\ \gamma_{3} \\ \gamma_{4} \\ \gamma_{5} \end{bmatrix}.$$

Таким образом, решение исходной задачи наименьших квадратов сводится к решению треугольной системы, получающейся удалением последней строки из матрицы  $\overline{H}_m^{(m)}$  и последнего элемента правой части  $\overline{g}_m$ . При этом очевидно, что для решения  $y_m$  невязка  $\|\beta e_1 - \overline{H}_m y_m\|$  совпадает с последним элементом правой части (в рассмотренном нами примере – с  $\gamma_5$ ). Можно показать, что данное значение совпадает с невязкой исходной системы уравнение, т.е.

$$\|r_m\| = \|b - Ax_m\| = |\gamma_{m+1}|.$$

Указанное значение можно использовать в критерии остановки метода, и организовать такую проверку на каждом шаге алгоритма.

Вернемся к рассмотренному примеру при m=4, и предположим, что уже применены 4 вращения  $\Omega_1$ ,  $\Omega_2$ ,  $\Omega_3$ ,  $\Omega_4$ . Тогда может быть проверен критерий малости невязки  $||r_4|| < \varepsilon$  (или же относительной невязки  $||r_4||/|b|| < \varepsilon$ ). Предположим, что критерий не выполнен, и требуется продолжать вычисления. Тогда на следующем шаге алгоритма будет вычислен пятый столбец матрицы  $\overline{H}_5$ . Приписав данный столбец к матрице  $\overline{H}_4^{(4)}$  (расширив ее предварительно нулевой строкой для соответствия размерностей), получим матрицу  $\overline{H}_5^{(1)}$ . Затем, применив к последнему столбцу все предыдущие вращения  $\Omega_1$ ,  $\Omega_2$ ,  $\Omega_3$ ,  $\Omega_4$ , получим систему

$$\overline{H}_{5}^{(4)} = \begin{bmatrix} h_{11}^{(4)} & h_{12}^{(4)} & h_{13}^{(4)} & h_{14}^{(4)} & h_{15}^{(4)} \\ & h_{22}^{(4)} & h_{23}^{(4)} & h_{24}^{(4)} & h_{25}^{(4)} \\ & & h_{33}^{(4)} & h_{34}^{(4)} & h_{35}^{(4)} \\ & & & h_{44}^{(4)} & h_{45}^{(4)} \\ & & & 0 & h_{55}^{(4)} \\ & & & 0 & h_{65}^{(4)} \end{bmatrix}, \ \overline{g}_{5} = \begin{bmatrix} \gamma_{1} \\ \gamma_{2} \\ \gamma_{3} \\ \gamma_{4} \\ \gamma_{5} \\ 0 \end{bmatrix}.$$

Далее алгоритм продолжается как и раньше – к системе применяется очередное вращение  $\Omega_5$  с целью обнуления элемента  $h_{65}^{(4)}$ , решается треугольная система, находится промежуточное решение  $y_5$ , затем через него выражается очередное приближение  $x_5 = x_0 + V_5 y_5$ , проверяется малость невязки  $||r_5|| < \varepsilon$  и т.д.

#### 3.2.1. Метод с перезапуском

Как следует из описания алгоритма, для его реализации требуется хранение матрицы  $\overline{H}_m$  размера  $(m+1) \times m$  и матрицы  $V_m$  размера  $n \times m$  (здесь m – размерность подпространства Крылова, n – размер матрицы A). Поэтому с ростом m возрастают расходы на хранение матрицы и решение соответствующей переопределенной системы. В качестве обхода данной проблемы можно предложить перезапуск метода с текущего приближения. Схема метода с перезапуском приведена ниже.

Пусть дано  $m \ll n$  и  $x_0$  – начальное приближение, тогда алгоритм с перезапуском GMRes(*m*) определяется следующими шагами:

1. Вычислить  $r_0 = b - Ax_0$ ,  $r_0 = b - Ax_0$  и  $\beta = r_0/\beta$ .

2. В соответствии с исходным алгоритмом GMRes вычислить матрицу  $H_m$  размера  $(m+1) \times m$ .

3. Вычислить  $y_m = \arg\min_{y} \left\| \beta e_1 - \overline{H}_m y \right\|$  и  $x_m = x_0 + V_m y_m$ 

4. Если выполнено условие  $||r_m|| < \varepsilon$ , то стоп;

Иначе  $x_0 = x_m$ , перейти на шаг 1.

Известной трудностью, которая возникает при использовании GMRes(m) является возможное нарушение сходимости в случае, если матрица A не является положительно определенной. Исходный алгоритм GMRes гарантирует (в точной арифметике, без ошибок округления) сходимость максимум за n итераций, где n – размер матрицы A. Для улучшения сходимости

метода может применяться предобуславливание, базовый алгоритм GMRes с использованием предобуславливателя *M* приведен ниже. Предобусловленный метод с перезапуском получается с использованием такой же схемы.

Алгоритм (обобщенный метод минимальных невязок с предобуславливанием, PGMRes)

Найти  $r_0$  из системы  $Mr_0 = b - Ax_0$ ,

Задать  $\beta = ||r_0||$  и  $v_1 = r_0/\beta$ .

Задать нулевую матрицу  $\overline{H}_m$  размера (*m*+1)×*m*.

for *j*=1,2,...,*m* do

Найти  $w_j$  из системы  $Mw_j = Av_j$ 

Для i=1,2,...,j вычислить  $h_{ij} = (w_i, v_i), w_j = w_j - h_{ij}v_i$ 

Вычислить  $h_{i+1, j} = \|w_i\|$ .

if  $h_{j+1,j} = 0$  then *m*=*j* и перейти на шаг (\*)

Вычислить  $v_{j+1} = w_j / h_{j+1,j}$ 

end j

(\*) Вычислить 
$$y_m = \arg\min_{y} \left\| \beta e_1 - \overline{H}_m y \right\|$$
 и  $x_m = x_0 + V_m y_m$ .

#### 3.2.1. Результаты вычислительных экспериментов

Проведем серию экспериментов по решению систем уравнений с разреженными матрицами методом обобщенных минимальных невязок. Разреженные матрицы систем линейных уравнений были взяты из коллекции матриц университета Флориды [30]. Характеристики используемых в экспериментах матриц (название, размер *n*, число ненулевых элементов *nz*, оценка число обусловленности  $\mu_A$ ) приведены в таблица 5. Так как метод GMRes не требует симметрии или положительной определенности матрицы, все матрицы являются вещественными квадратными матрицами общего вида.

Таблица 5. Характеристики используемых матриц

Название	п	nz	$\mu_{A}$
fs_183_1	183	998	$2,2 \cdot 10^{13}$
fs_541_1	541	4282	4467

sherman2	1080	23094	9,6·10 <sup>11</sup>
watt_1	1856	11360	$4, 4.10^{9}$
cage10	11397	150645	11

Для иллюстрации портрет одной из матриц (fs\_183\_1) изображен на рис. 11.



**Рис. 11.** Портрет матрицы fs\_183\_1

Системы уравнений с конкретной матрицей А формировались следующим образом:

- 8. Выбиралось точное решение (например, единичный вектор  $x^* = \{x_i = 1, 1 \le i \le n\}$ ).
- 9. На основе точного решения вычислялась правая часть  $b = Ax^*$ .
- 10. Получалась система уравнений Ax=b с известным решением  $x^*$ .

В качестве критерия остановки метода использовался критерий остановки по невязке с параметром  $\varepsilon = 10^{-8}$ , максимальное число итераций – 100, метод запускался как без предобуславливателя, так и с предобуславливателем (использовалось *ILU*(0)-разложение), число выполненных при этом итераций метода *s* приведено в таблица 6.

Таблица 6. Результаты работы методов GMRes и PGMRes на матрицах из коллекции университета Флориды

Название	n	S		
матрицы	11	GMRes	PGMRes	
fs_183_1	183	78	16	
fs_541_1	541	13	2	
sherman2	1080	100	18	
watt_1	1856	40	48	
cage10	11397	24	7	

Метод сошелся за нужное число итераций на всех матрицах, кроме одной. При этом метод с предобуславливанием сошелся во всех случаях, и число итераций, требуемое для достижения нужной точности, было для большинства задач меньше.

## 3.3. Метод бисопряженных градиентов

Метод бисопряженных градиентов основан на биортогонализации Ланцоша точно так же, как метод сопряженных градиентов основан на симметричной ортогонализации Ланцоша. Неявно рассматриваемый метод решает не только исходную задачу Ax = b, то также и двойственную систему  $A^T \bar{x} = \bar{b}$  (в формулировке алгоритма эта двойственная система отсутствует).

Пусть выполнены m шагов биортогонализации Ланцоша. Запишем LU разложение матрицы  $T_m$  из (4.25) в виде

$$T_m = L_m U_m$$
.

Тогда решение *x<sub>m</sub>* на шаге *m* алгоритма может быть записано как

$$x_m = x_0 + V_m T_m^{-1}(\beta e_1) = x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1) = x_0 + P_m L_m^{-1}(\beta e_1),$$

где

$$P_m = V_m U_m^{-1}.$$

Рассмотрим свойства матрицы *P<sub>m</sub>*. Для этого составим по аналогии матрицу

$$\overline{P}_m = W_m (L_m^T)^{-1}.$$

Тогда выполняется равенство

$$\overline{P}_{m}^{T}AP_{m} = L_{m}^{-1}W_{m}^{T}AV_{m}U_{m}^{-1} = L_{m}^{-1}T_{m}U_{m}^{-1} = E$$

Фактически, данное равенство означает выполнение свойства *А*бисопряженности (или же просто бисопряженности, если матрица *А* понятна из контекста) для векторов  $\{p_1,...,p_m\}$  и  $\{\overline{p}_1,...,\overline{p}_m\}$ :

$$(Ap_i, \overline{p}_j) = 0$$
 при  $i \neq j$ , (4.27)

где  $p_i - i$ -й столбец матрицы  $P_m$ ,  $\overline{p}_i - j$ -й столбец матрицы  $\overline{P}_m$ .

Очевидно, что бисопряженность эквивалентна биортогональности относительно скалярного А-произведения, поэтому для нахождения систем векторов  $\{p_1,...,p_m\}$  и  $\{\overline{p}_1,...,\overline{p}_m\}$  можно воспользоваться биортогонализацией Ланцоша, используя в ней скалярные А-произведения. Собирая все воедино, получаем следующий алгоритм.

Алгоритм (метод бисопряженных градиентов, BiCG).

Предварительный шаг – вычислить  $r_0 = b - Ax_0$ ; выбрать  $\bar{r}_0$  так, чтобы  $(r_0, \bar{r}_0) \neq 0$  (например, выбрать  $\bar{r}_0 = r_0$ ); положить  $p_0 = r_0$ ,  $\bar{p}_0 = \bar{r}_0$ 

Основные шаги (j=0,1,...,n) определяются формулами

$$\begin{split} \alpha_{j} &= \frac{(r_{j}, \bar{r}_{j})}{(Ap_{j}, \bar{p}_{j})}, \ x_{j+1} = x_{j} + \alpha_{j} p_{j}, \\ r_{j+1} &= r_{j} - \alpha_{j} Ap_{j}, \ \bar{r}_{j+1} = \bar{r}_{j} - \alpha_{j} A^{T} \bar{p}_{j} \\ \beta_{j} &= \frac{(r_{j+1}, \bar{r}_{j+1})}{(r_{j}, \bar{r}_{j})}. \text{ Если } \beta_{j} = 0 \text{ или } \| r_{j+1} \| < \varepsilon \text{ то Стоп.} \\ p_{j+1} &= r_{j+1} + \beta_{j} p_{j}, \ \bar{p}_{j+1} = \bar{r}_{j+1} + \beta_{j} \bar{p}_{j} \end{split}$$

Если требуется решить двойственную систему с матрицей  $A^T$ , то тогда в начале алгоритма надо выбирать  $\bar{r}_0 = \bar{b} - A^T x_0$ , а в теле алгоритма обновлять текущее приближение как  $\bar{x}_{j+1} = \bar{x}_j + \alpha_j \bar{p}_j$ . При этом нетрудно показать, что вектора  $\{r_1, ..., r_m\}$  и  $\{\bar{r}_1, ..., \bar{r}_m\}$  являются бисопряженными, т.е.

$$(r_i, \overline{r}_i) = 0$$
 при  $i \neq j$ .

Введение в схему метода сопряженных градиентов предобуславливателя *М* происходит по стандартной схеме.

Алгоритм (метод бисопряженных градиентов с предобуславливанием, PBiCG<sup>2</sup>)

Предварительный шаг – вычислить  $r_0 = b - Ax_0$ ; выбрать  $\bar{r}_0$  так, чтобы  $(r_0, \bar{r}_0) \neq 0$  (например, выбрать  $\bar{r}_0 = r_0$ );  $Mz_0 = r_0$ ,  $M^T \bar{z}_0 = \bar{r}_0$  положить  $p_0 = z_0$ ,  $\bar{p}_0 = \bar{z}_0$ 

Основные шаги (*j*=0,1,...,*n*) определяются формулами

$$\alpha_{j} = \frac{(z_{j}, \bar{r}_{j})}{(Ap_{j}, \bar{p}_{j})}, \ x_{j+1} = x_{j} + \alpha_{j} p_{j},$$
$$r_{j+1} = r_{j} - \alpha_{j} Ap_{j}, \ \bar{r}_{j+1} = \bar{r}_{j} - \alpha_{j} A^{T} \bar{p}_{j}$$
$$Mz_{j+1} = r_{j+1}, \ M^{T} \bar{z}_{j+1} = \bar{r}_{j+1}$$

 $\beta_j = \frac{(z_{j+1}, \bar{r}_{j+1})}{(z_j, \bar{r}_j)}$ . Если  $\beta_j = 0$  или  $||r_{j+1}|| < \varepsilon$  то Стоп.  $p_{j+1} = z_{j+1} + \beta_j p_j, \ \bar{p}_{j+1} = \bar{z}_{j+1} + \beta_j \bar{p}_j$ 

Отметим, что в алгоритме используются матрицы  $A^{T}$  (и, соответственно,  $M^{T}$ ), поэтому использование метода бисопряженных градиентов может оказаться затруднительным, если получение транспонированных матриц в силу специфики решаемой задачи является трудоемкой операцией. В этом случае можно применить метод Conjugate Gradient Squared [20], в котором не требуется проводить транспонирование матрицы A.

#### 3.3.1. Результаты вычислительных экспериментов

Проведем серию экспериментов по решению систем уравнений с разреженными матрицами методом бисопряженных градиентов. Так как метод BiCG не требует симметрии или положительной определенности матрицы, эксперименты проведем на тех же задачах, что и в методе GMRes (см. п. 3.2.1).

В качестве критерия остановки метода использовался критерий остановки по невязке с параметром  $\varepsilon = 10^{-4}$ , максимальное число итераций выбиралось равным порядку матрицы, метод запускался как без предобуславливателя, так и с предобуславливателем (использовалось *ILU*(0)-разложение). В обоих случаях методы достигли требуемой точности, число выполненных при этом итераций метода *s* приведено в таблица 7.

<sup>&</sup>lt;sup>2</sup> Preconditioned Biconjugate Gradient

Название	п	S		
матрицы	п	GMRes	PGMRes	
fs_183_1	183	183	10	
fs_541_1	541	6	1	
sherman2	1080	1080	20	
watt_1	1856	839	36	
cage10	11397	10	3	

Таблица 7. Результаты работы методов BiCG и PBiCG на матрицах из коллекции университета Флориды

Отметим, что при более грубой точности метод BiCG сделал больше итераций, чем метод GMRes (а с предобуславливанием – столько же). Однако при этом BiCG значительно менее требователен к объему хранимой информации, поэтому его можно применять к матрицам большего размера.

### 3.4. Метод сопряженных градиентов

Рассмотрим систему линейных уравнений (4.1) с симметричной, положительно определенной матрицей A размера  $n \times n$ . Все ранее рассмотренные методы крыловского типа не предполагали каких-либо дополнительных свойств матрицы A, и поэтому могут быть и к системе с симметричной матрицей напрямую. Однако учет симметрии позволяет упростить вычислительную схему метода, что ведет к его более легкой реализации и меньшим требованиям к ресурсам.

В первую очередь отметим, что для симметричной матрицы подпространства Крылова  $K_m(r_0, A)$  и  $K_m(r_0, A^T)$  будут совпадать. Это означает, что в методе бисопряженных градиентов при выборе  $\bar{r}_0 = r_0$  будут выполняться равенства

$$\overline{r}_i = r_i, \ \overline{p}_i = p_i. \tag{4.28}$$

Значит, отпадает необходимость в хранении и обработке векторов  $\bar{r}_i$  и  $\bar{p}_i$ . При этом условие (4.27) бисопряженности векторов  $p_i$  и  $\bar{p}_j$  превращается в условие сопряженности

$$(Ap_i, p_i) = 0$$
 при  $i \neq j$ .

Все это приводит к нас к следующему алгоритму.

Алгоритм (метод сопряженных градиентов, CG<sup>3</sup>)

Предварительный шаг – вычислить  $r_0 = p_0 = b - Ax_0$ 

Основные шаги (*i*=0, 1, 2,..., *n*) определяются формулами

$$\alpha_{i} = \frac{(r_{i}, r_{i})}{(Ap_{i}, p_{i})}, \ x_{i+1} = x_{i} + \alpha_{i}p_{i},$$
$$r_{i+1} = r_{i} - \alpha_{i}Ap_{i},$$
$$(r_{i} - r_{i})$$

$$\beta_i = \frac{(r_{i+1}, r_{i+1})}{(r_i, r_i)}, \ p_{i+1} = r_{i+1} + \beta_i p_i.$$

Здесь  $r_i$  – невязка *i*-го приближения, а коэффициент  $\beta_i$  выбирается так, чтобы выполнялось условие сопряженности

$$(Ap_i, p_{i-1}) = 0$$

направлений  $p_i$  и  $p_{i-1}$ .

Отметим, что метод сопряженных градиентов допускает следующую интерпретацию. Решение системы линейных уравнений (4.1) с симметричной положительно определенной матрицей *А* эквивалентно решению задачи минимизации функции

$$F(x) = \frac{1}{2}(Ax, x) - (b, x).$$
(4.29)

в пространстве  $R^n$ . В самом деле, функция F(x) достигает своего минимального значения тогда и только тогда, когда ее градиент

$$\nabla F(x) = Ax - b \tag{4.30}$$

обращается в ноль. Таким образом, решение системы (4.1) можно искать как решение задачи безусловной минимизации (4.29).

С этой точки зрения коэффициент  $\alpha_i$  является решением задачи минимизации функции *F* по направлению  $p_i$ 

$$\alpha_i = \arg\min F(x_i + \alpha p_i).$$

Анализ расчетных формул метода показывает, что они включают две операции умножения матрицы на вектор, четыре операции скалярного произведения и пять операций над векторами. Однако на каждой итерации про-

<sup>&</sup>lt;sup>3</sup> Conjugate Gradient

изведение *Ар<sub>i</sub>* достаточно вычислить один раз, а затем использовать сохраненный результат. Общее количество числа операций, выполняемых на одной итерации, составляет

$$t_1 = 2n^2 + 13n$$
.

Таким образом, выполнение L итераций метода потребует

$$T_1 = L(2n^2 + 13n) \tag{4.31}$$

операций. Можно показать, что для нахождения точного решения системы линейных уравнений с положительно определенной симметричной матрицей необходимо выполнить не более n итераций, тем самым, сложность алгоритма поиска точного решения имеет порядок  $O(n^3)$ . Однако ввиду ошибок округления данный процесс обычно рассматривают как итерационный, процесс завершается либо при выполнении обычного условия остановки (4.3), либо при выполнении условия малости относительной нормы невязки

$$\|r_i\|/\|b\| \leq \varepsilon$$
.

Введение в схему метода сопряженных градиентов предобуславливателя *М* происходит по стандартной схеме.

Алгоритм (метод сопряженных градиентов с предобуславливанием, PCG<sup>4</sup>)

Предварительный шаг – вычислить  $r_0 = b - Ax_0$ ,  $Mz_0 = r_0$ ,  $p_0 = z_0$ .

Основные шаги (*i*=0, 1, 2,..., *n*-1) определяются формулами

$$\alpha_{i} = \frac{(r_{i}, z_{i})}{(Ap_{i}, p_{i})}, \ x_{i+1} = x_{i} + \alpha_{i} p_{i},$$
$$r_{i+1} = r_{i} - \alpha_{i} Ap_{i}, \ z_{i+1} = M^{-1} r_{i+1},$$
$$\beta_{i} = \frac{(r_{i+1}, z_{i+1})}{(r_{i}, z_{i})}, \ p_{i+1} = z_{i+1} + \beta_{i} p_{i}.$$

Отметим еще раз, что использование предобуславливателя не предполагает вычисление матрицы  $M^{-1}$ , требуется лишь решить систему уравнений

$$Mz_{i+1} = r_{i+1}$$
,

<sup>&</sup>lt;sup>4</sup> Preconditioned Conjugate Gradient

относительно неизвестного вектора  $z_{i+1}$ . Приведенные ниже результаты экспериментов показывают значительное влияние предобуславливания на скорость сходимости метода.

#### 3.4.1. Организация параллельных вычислений

При разработке параллельного варианта метода сопряженных градиентов для решения систем линейных уравнений в первую очередь следует учесть, что выполнение итераций метода осуществляется последовательно и, тем самым, наиболее целесообразный подход состоит в распараллеливании вычислений, реализуемых в ходе выполнения итераций.

Анализ последовательного алгоритма показывает, что основные затраты на *s*-й итерации состоят в умножении матрицы *A* на вектора  $p_{i-1}$  и  $p_i$ . Как результат, при организации параллельных вычислений могут быть использованы известные методы параллельного умножения матрицы на вектор.

Дополнительные вычисления, имеющие меньший порядок сложности, представляют собой различные операции обработки векторов (скалярное произведение, сложение и вычитание, умножение на скаляр). Организация таких вычислений, конечно же, должна быть согласована с выбранным параллельным способом выполнения операция умножения матрицы на вектор.

Выберем для дальнейшего анализа эффективности получаемых параллельных вычислений параллельный алгоритм матрично-векторного умножения при ленточном горизонтальном разделении матрицы. При этом операции над векторами, обладающие меньшей вычислительной трудоемкостью, также будем выполнять в многопоточном режиме. Вычислительная сложность параллельной операции умножения матрицы на вектор при использовании схемы ленточного горизонтального разделения матрицы составляет

$$2n(2n-1)/p+\delta$$
,

где n – длина вектора, p – число потоков,  $\delta$  – накладные расходы на создание и закрытие параллельной секции.

Все остальные операции над векторами (скалярное произведение, сложение, умножение на константу) могут быть выполнены в однопоточном режиме, т.к. не являются определяющими в общей трудоемкости метода. Следовательно, общая вычислительная сложность параллельного варианта метода сопряженных градиентов может быть оценена как

$$T_p = L\left(\frac{2n^2}{p} + 13n + \delta\right),$$

где *L* – число итераций метода.

#### 3.4.2. Результаты вычислительных экспериментов

Вычислительные эксперименты для оценки эффективности параллельного варианта метода сопряженных градиентов для решения систем линейных уравнений с симметричной положительно определенной матрицей проводились при условиях, указанных во введении.

В первой серии экспериментов будем решать системы уравнений с плотной матрицей. Элементы на главной диагонали матрицы A) генерировались в диапазоне от n до 2n, где n – размер матрицы, остальные элементы генерировались симметрично в диапазоне от 0 до 1. В качестве критерия остановки использовался критерий остановки по точности (4.3) с параметром  $\varepsilon = 10^{-6}$ .

Результаты вычислительных экспериментов приведены в таблица 8 (время работы алгоритмов указано в секундах).

		Параллельный алгоритм							
n	1 поток	2 по	тока	4 по	тока	6 пот	оков	8 пот	оков
		t	S	t	S	t	S	t	S
500	0,02	0,01	1,64	0,01	2,56	0,01	2,56	0,01	2,56
1000	0,21	0,16	1,26	0,10	2,09	0,07	2,88	0,05	3,83
1500	0,65	0,48	1,36	0,27	2,41	0,19	3,42	0,15	4,20
2000	1,32	0,94	1,41	0,53	2,51	0,38	3,48	0,29	4,50
3000	3,42	2,34	1,46	1,33	2,58	0,96	3,56	0,74	4,63
4000	6,49	4,54	1,43	2,53	2,56	1,80	3,60	1,40	4,62
5000	11,02	7,41	1,49	4,17	2,65	2,98	3,70	2,31	4,78

Таблица 8. Результаты экспериментов (метод сопряженных градиентов)



Рис. 12. Ускорение параллельного метода сопряженных градиентов

Так как операция умножения матрицы на вектор – основная по трудоемкости операция на одной итерации метода – распараллеливается хорошо, то и эффективность распараллеливания метода сопряженных градиентов будет линейная, что подтверждается приведенным графиком.

Спад ускорения при N=500 объясняется недостаточной вычислительной нагрузкой, которая приходится на каждый процесс (этот эффект будет проиллюстрирован и в дальнейшем при решении дифференциальных уравнений в частных производных). Использовать более чем 4 потока для решения данной задачи при  $N \le 500$  – нецелесообразно.

Во второй серии экспериментов будут решаться системы уравнений с разреженными матрицами. Разреженные матрицы систем линейных уравнений были взяты из коллекции матриц университета Флориды [30]. Характеристики используемых в экспериментах матриц (название, размер *n*, число ненулевых элементов *nz*, оценка число обусловленности  $\mu_A$ ) приведены в таблица 9 (все матрицы являются симметричными положительно определенными).

Название	п	nz	$\mu_{A}$
bcsstk01	48	400	$8.82 \cdot 10^5$
bcsstk05	153	2 423	$1.43 \cdot 10^4$
bcsstk10	1086	22 070	$5.24 \cdot 10^5$
bcsstk13	2003	83 883	$1.06 \cdot 10^{10}$
parabolic_fem	525 825	3 674 625	$2.11 \cdot 10^{10}$
tmt_sym	726 713	5 080 961	5

Таблица 9. Характеристики используемых матриц

Для иллюстрации портрет одной из матриц (bcsstk13) изображен на рис. 13.

Системы уравнений с конкретной матрицей А формировались следующим образом:

- 1. Выбиралось точное решение (например, единичный вектор  $x^* = \{x_i = 1, 1 \le i \le n\}$ ).
- 2. На основе точного решения вычислялась правая часть  $b = Ax^*$ .
- 3. Получалась система уравнений Ax=b с известным решением  $x^*$ .

<sup>&</sup>lt;sup>5</sup> В коллекции матриц число обусловленности для матрицы tmt\_sym отсутствует



Рис. 13. Портрет матрицы bcsstk13

В качестве критерия остановки метода использовался критерий остановки по относительной невязке с параметром  $\varepsilon = 10^{-7}$ , метод запускался как без предобуславливателя, так и с предобуславливателем (использовалось *ILU*(0)-разложение). В обоих случаях методы достигли требуемой точности, число выполненных при этом итераций метода *s* приведено в таблица 10.

Название	п	S		
матрицы	п	CG	PCG	
bcsstk01	48	125	15	
bcsstk05	153	272	35	
bcsstk10	1086	2335	149	
bcsstk13	2003	20031	10016	
parabolic_fem	525 825	1690	1045	
tmt_sym	726 713	5356	1210	

Таблица 10. Результаты раб	оты методов СС и	PCG
на матрицах из коллекции	университета Фло	риды

Эксперименты показывают, что для достижения требуемой точности методу сопряженных градиентов требуется итераций больше, чем порядок матрицы *n*. Это связно с наличием ошибок округления, в точной арифметике метод сходится за число итераций, не превосходящее *n*. Подробнее данный эффект рассмотрен в лабораторной работе «Метод сопряженных градиентов». Также из результатов экспериментов видно, что использование предобуславливателя значительно сокращает число итераций, необходимых для достижения требуемой точности.

## 4. Литература

#### 4.1. Использованные источники информации

- Вержбицкий В.М. Численные методы (математический анализ и обыкновенные дифференциальные уравнения). – М.: Высшая школа, 2001.
- Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. М.: Наука, 1987.
- Тихонов А.Н., Самарский А.А. Уравнения математической физики. М.: Наука, 1977.
- 4. Самарский А.А., Гулин А.В. Численные методы. М.: Наука, 1989.
- 5. Самарский А.А. Введение в численные методы. СПб.: Лань, 2005.
- 6. Калиткин Н.Н. Численные методы. М.: Наука, 1978
- 7. Хамахер К., Вранешич З., Заки С. Организация ЭВМ. СПб.: Питер, 2003.
- 8. Голуб Дж., Ван Лоун Ч. Матричные вычисления. М.: Мир, 1999.
- 9. Деммель Дж. Вычислительная линейная алгебра. М.: Мир, 2001.
- 10. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений. М.: Мир, 1984.
- 11. Писсанецки С. Технология разреженных матриц. М.: Мир, 1988.
- 12. Соболь И.М. Численные методы Монте-Карло. М.: Наука, 1973.
- Соболь И.М. Точки, равномерно заполняющие многомерный куб. М.: Знание, 1985.
- 14. Д. Кнут. Искусство программирования. Том 2: Получисленные алгоритмы. М.: «Вильямс», 2007.
- Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. – Н.Новгород, Изд-во ННГУ, 2003.
- Гергель В.П. Теория и практика параллельных вычислений. М.: БИНОМ, 2007.
- 17. Гантмахер Ф. Р. Теория матриц. М.: Наука, 1966.

- 18. Белов С.А., Золотых Н.Ю. Численные методы линейной алгебры. Н.Новгород, Изд-во ННГУ, 2005.
- Баландин М.Ю., Шурина Э.П. Методы решения СЛАУ большой размерности. – Новосибирск: Изд-во НГТУ, 2000.
- 20. Dongarra J., et al. Templates for the solution of linear systems: building blocks for iterative methods. SIAM, 1994.
- 21. Saad Y. Iterative Methods for Sparse Linear Systems. SIAM, 2003.
- 22. Karniadakis G., Kirby R. Parallel scientific computing in C++ and MPI. Cambridge university press, 2003.
- 23. Quinn M. Parallel programming in C with MPI and OpenMP. McGraw-Hill, 2004.

## 4.2. Дополнительная литература

- 24. Ширяев А.Н. Вероятность, М.: Наука. 1989.
- Metropolis N., Ulam S. The Monte Carlo method // J. Amer. statistical assoc. 1949. 44, N 247, P. 335-341.
- Percus O., Kalos M. Random number generators for MIMD parallel processors // J. of Parallel and Distributed Computing. 1989. V.6. P. 477–479.
- Matsumoto M., Nishimura T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator // ACM Trans. on Modeling and Computer Simulations. 1998. V. 8(1).
- Mascagni M., Srinivasan A. Algorithm 806: SPRNG: A scalable library for pseudorandom number generation // ACM Transactions on Mathematical Software. 2000. V. 26, № 3. P. 436–461.
- Niederreiter H. Random Number Generation and Quasi-Monte Carlo Methods. – SIAM, 1992. – 247 p.

#### 4.3. Информационные ресурсы сети Интернет

- 30. The University of Florida Sparse Matrix Collection [http://www.cise.ufl.edu/research/sparse/matrices/]
- 31. Intel Math Kernel Library Reference Manual [http://software.intel.com/ sites/products/documentation/hpc/mkl/mklman.pdf].