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
2011

Содержание

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 $x\in R^n$ с симметричной положительно определенной матрицей A размерности $n\times n$ и правой частью вектором $b\in R^n$. Точное решение системы (4.1) обозначим через x^*.

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

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

где $x^{(s)}$ – приближение, полученное на итерации с номером s, $x^{(s-1)}$ – приближение, полученное на предыдущей итерации. Параметр *точности метода* ε_1 задает исследователь. Если условие (4.3) выполнено, метод завершает работу и вектор $x^{(s)}$ трактуется как приближенное решение системы (4.1). Величина ε_2, определяемая как

$$\varepsilon_2=\left\|x^{(s)}-x^{(s-1)}\right\| \tag{4.4}$$

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 ε_2 также сообщается исследователю.

При изучении сходимости итерационных методов и при их реализации в качестве векторной нормы $\| \ \|$ обычно используют евклидову норму $\| \ \|_2$, норму $\| \ \|_1$, определяемую суммой модулей всех компонент вектора, норму $\| \ \|_\infty$, определяемую максимальным модулем компоненты, а также энергетическую норму $\| \ \|_A$, 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.

4.1. Метод простой итерации

4.1.1. Последовательный алгоритм

Метод простой итерации относится к классу явных стационарных одношаговых итерационных методов решения линейных систем вида $Ax=b$ с симметричной положительно определенной матрицей A. Метод 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 $x^{(s+1)}$:

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

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

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

где λ_{min} и λ_{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 справедлива следующая оценка:

$$\left\| z^{(s+1)} \right\|_2 \le \left(\frac{\mu_A - 1}{\mu_A + 1} \right)^{s+1} \left\| z^{(0)} \right\|_2 .$$

Здесь через $\| \ \|_2$ обозначена среднеквадратичная норма вектора, а через μ_A – число обусловленности матрицы A, согласованное с векторной нормой $\| \ \|_2$ 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.

4.1.2. 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 алгоритма 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.

4.2. Метод верхней релаксации

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

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

Метод верхней релаксации является представителем стационарных одношаговых итерационных методов линейной алгебры и записывается в виде

$$\frac{(D+\omega L)(x^{(s+1)}-x^{(s)})}{\omega}+Ax^{(s)}=b. \tag{4.6}$$

Здесь $x^{(s)}$ – приближение, полученное на итерации с номером s, $x^{(s+1)}$ – следующее приближение, ω – 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 опреде-

лена, 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 ω. Известно [4], что при решении некоторых классов разреженных систем уравнений, метод Зейделя требует $O(n^2)$ итераций, а при надлежащем выборе итерационного параметра ω метод будет сходиться за $O(n)$ итераций.

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

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

где h – шаг сетки, на которой проводилась дискретизация.

В некоторых случаях можно более точно оценить оптимальный параметр. Известной оценкой [17] является выражение

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

где ρ – 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).

4.2.1. Последовательный алгоритм

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

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

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

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

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 $x^{(s+1)}$:

$$a_{ii}x_i^{(s+1)} = -\omega\sum_{j=1}^{i-1} a_{ij}x_j^{(s+1)} + (1-\omega)a_{ii}x_i^{(s)} - \omega\sum_{j=i+1}^{n} a_{ij}x_j^{(s)} + \omega b_i \qquad (4.7)$$

Как следует из формулы (4.7), 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 нигде 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 (4.7) для всех компонент по порядку и постепенно обновлять вектор $x^{(s)}$.

4.2.2. 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 (4.7).

4.2.3. 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.

Табл. 4.1. Результаты экспериментов (метод верхней релаксации)

n	1 поток	Параллельный алгоритм							
		2 потока		4 потока		6 потоков		8 потоков	
		T	S	T	S	T	S	T	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

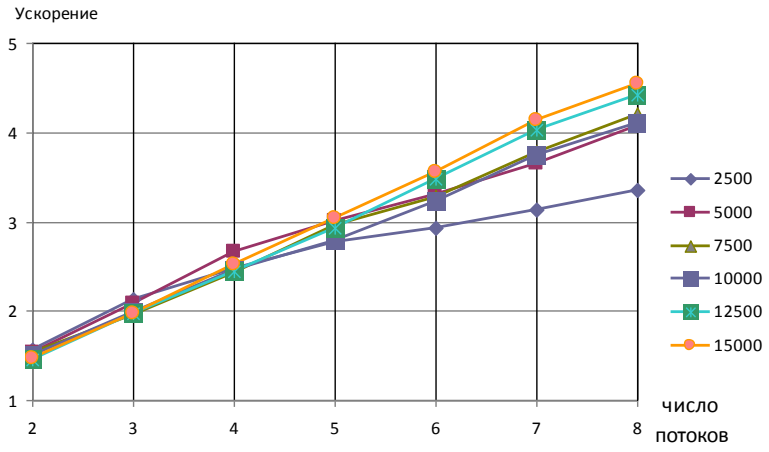

Рис. 4.1. Ускорение параллельного метода верхней релаксации

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

4.3. Метод сопряженных градиентов

Рассмотрим систему линейных уравнений (4.1) с симметричной, положительно определенной матрицей A размера $n \times n$. Основой *метода сопряженных градиентов* является следующее свойство: решение системы линейных уравнений (4.1) с симметричной положительно определенной матрицей A эквивалентно решению задачи минимизации функции

$$F(x) = \frac{1}{2}(Ax, x) - (b, x). \tag{4.8}$$

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

$$\nabla F(x) = Ax - b \tag{4.9}$$

обращается в ноль. Таким образом, решение системы (4.1) можно искать как решение задачи безусловной минимизации (4.8).

4.3.1. Последовательный алгоритм

С целью решения задачи минимизации (4.8) организуется следующий итерационный процесс.

Подготовительный шаг (s=0) определяется формулами

$$x^{(1)} = x^{(0)} + \alpha_0 h^{(0)},\ r^{(0)} = h^{(0)} = b - Ax^{(0)}.$$

где $x^{(0)}$ – произвольное начальное приближение; а коэффициент α_0 вычисляется как

$$\alpha_0 = \frac{(r^{(0)}, r^{(0)})}{(Ah^{(0)}, h^{(0)})}$$

Основные шаги (s=1, 2,…, *n*–1) определяются формулами

$$\alpha_s = \frac{(r^{(s)}, r^{(s)})}{(Ah^{(s)}, h^{(s)})},\quad r^{(s+1)} = r^{(s)} - \alpha_s Ah^{(s)},$$

$$\beta_s = \frac{(r^{(s+1)}, r^{(s+1)})}{(r^{(s)}, r^{(s)})},\ \ h^{(s+1)} = r^{(s+1)} + \beta_s h^{(s)},$$

$$x^{(s+1)} = x^{(s)} + \alpha_s h^{(s)}.$$

Здесь $r^{(s)} = b - Ax^{(s)}$ – невязка *s*-го приближения, коэффициент β_s находят из условия сопряженности

$$(Ah^{(s)}, h^{(s-1)}) = 0$$

направлений $h^{(s)}$ и $h^{(s-1)}$; а α_s является решением задачи минимизации функции *F* по направлению h_s

$$F(x_s + \alpha h_s) \to \min.$$

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 произведение $Ah^{(s)}$ 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

$$T_1 = L(2n^2 + 13n) \qquad (4.10)$$

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

$$\left\|r^{(k)}\right\| / \|b\| \le \varepsilon .$$

4.3.2. 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 h_{s-1} и h_s. 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 трудоемкость последовательного метода сопряженных градиентов определяется соотношением (4.10). Определим время выполнения параллельной реализации метода сопряженных градиентов. 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, δ – 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.

4.3.3. 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. 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 таблице 2.2 (время работы алгоритмов указано в секундах).

Табл. 4.2. Результаты экспериментов (метод сопряженных градиентов)

n	1 поток	Параллельный алгоритм							
		2 потока		4 потока		6 потоков		8 потоков	
		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

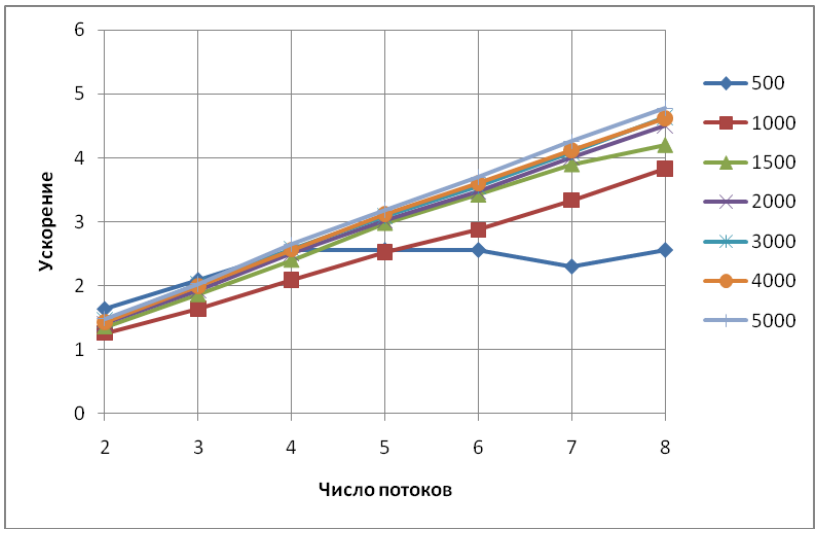

Рис. 4.2. Ускорение параллельного метода сопряженных градиентов

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.

Литература

Использованные источники информации

1. Вержбицкий В.М. Численные методы (математический анализ и обыкновенные дифференциальные уравнения). – М.: Высшая школа, 2001.
2. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. – М.: Наука, 1987.
3. 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 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. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений. – М.: Мир, 1984.
10. Писсанецки С. Технология разреженных матриц. — М.: Мир,1988.
11. Соболь И.М. Численные методы Монте-Карло. – М.: Наука, 1973.
12. Соболь И.М. Точки, равномерно заполняющие многомерный куб. – М.: Знание, 1985.
13. Д. Кнут. Искусство программирования. Том 2: получисленные алгоритмы. – М.: «Вильямс», 2007.
14. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. – Н.Новгород, Изд-во ННГУ, 2003.
15. Гергель В.П. Теория и практика параллельных вычислений. – М.: БИНОМ, 2007.
16. Белов С.А., Золотых Н.Ю. Численные методы линейной алгебры. – Н.Новгород, Изд-во ННГУ, 2005.
17. J. Dongarra et al. Templates for the solution of linear systems: building blocks for iterative methods. SIAM, 1994.

18. G. Karniadakis, R. Kirby. Parallel scientific computing in C++ and MPI. Cambridge university press, 2003.

19. M. Quinn. Parallel programming in C with MPI and OpenMP. McGraw-Hill, 2004.

Дополнительная литература

20. Ширяев А. Н. Вероятность, – М.: Наука. 1989.

21. Metropolis N., Ulam S. The Monte Carlo method, J. Amer. statistical assoc., 1949, 44, N247, 335-341.

22. O. Percus, M. Kalos. Random number generators for MIMD parallel processors// Journal of parallel and distributed computing, v.6, 1989. pp. 477–479.

23. M. Matsumoto, T. Nishimura (1998). «Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator». ACM Trans. on Modeling and Computer Simulations v. 8(1).

24. M. Mascagni, A. Srinivasan. Algorithm 806: SPRNG: A scalable library for pseudorandom number generation. ACM Transactions on Mathematical Software, v. 26, № 3, 2000. pp. 436–461.

25. Niederreiter H. Random Number Generation and Quasi-Monte Carlo Methods. – SIAM, 1992. – 247 p.

Информационные ресурсы сети Интернет

26. Intel Math Kernel Library Reference Manual.

[http://software.intel.com/sites/products/documentation/hpc/mkl/mklman.pdf].