М. А. Черепнёв, Связь приближений ряда и базиса пространства Крылова в блочных алгоритмах Копперсмита и Монтгомери, *Фундамент. и прикл. матем.*, 2012, том 17, выпуск 5, 211–223

Связь приближений ряда и базиса пространства Крылова в блочных алгоритмах Копперсмита и Монтгомери

М. А. ЧЕРЕПНЁВ
Московский государственный университет им. М. В. Ломоносова
e-mail: cherepniov@gmail.com

УДК 512.62

Ключевые слова: факторизация целых чисел, решение разреженных систем над $GF(2)$, решение линейных систем над $GF(2)$.

Аннотация

В данной статье изучаются свойства алгоритма Видемана—Копперсмита. В частности, для случая симметричной системы линейных уравнений из приближений формального ряда, которые строятся в этом алгоритме в шагах с нечётными номерами, построен ортогональный базис пространства Крылова. Предложены модификации алгоритма, использующие описанные свойства.

Abstract

M. A. Cherepniov, A connection of series approximations and the basis of the Krylov space in block algorithms of Coppersmith and Montgomery, Fundamentalnaya i prikladnaya matematika, vol. 17 (2011/2012), no. 5, pp. 211—223.

In this paper, some properties of the Wiedemann—Coppersmith algorithm are studied. In particular, when the matrix of a linear system is symmetric, an orthogonal basis of the Krylov space is constructed with the help of approximations of formal series from odd steps of this algorithm. We propose some modifications that use the described properties.

В настоящее время для решения больших разреженных систем линейных уравнений над полем из двух элементов чаще всего используются блочные алгоритмы Монтгомери и Видемана—Копперсмита.

Первоначальная последовательная версия алгоритма Видемана—Копперсмита была ускорена в работе [12] ценой увеличения объёма используемой оперативной памяти, что на практике (см. [10]) ограничивает возможности выбора параметров, обеспечивающих быстрейшее выполнение всего алгоритма.

В данной статье сделана попытка выявить некоторые общие принципы, по которым работают алгоритмы Монтгомери и Видемана—Копперсмита. Представляется, что это поможет воспользоваться преимуществами каждого из них.

Заметим, что в алгоритме Видемана—Копперсмита, как и в алгоритмах Ланцоша и Монтгомери, решение находится через его координаты в пространстве

Фундаментальная и прикладная математика, 2011/2012, том 17, № 5, с. 211—223.

Крылова. Нетрудно заметить, что коэффициенты решения в пространстве Крылова ищутся как коэффициенты некоторого достаточно хорошего приближения ряда по положительным степеням некоторой формальной переменной. Их можно искать, последовательно наращивая порядок приближения, как это сделано Копперсмитом в [6], или при помощи сравнительно небольшого количества приближений с порядками, равными степеням двойки, как предлагается в [5, 7]. Отметим, что в этих работах строится базис всех приближений при помощи асимптотически более быстрого алгоритма с количеством операций $O(N \log^2 N \log\log N)$ против $O(N^2)$ у Копперсмита. Однако константа в O в первой оценке достаточно большая, кроме того, она плохо зависит от так называемого блочного фактора, ускоряющего весь алгоритм при использовании параллельных вычислений. Для постановки последнего рекорда разложения чисел RSA на множители (12 декабря 2009 года разложено число RSA-768, содержащее 232 десятичные цифры, или 768 бит) была взята процедура, аналогичная процедуре в исходной версии Копперсмита, но строящая только «хорошие» приближения, степени которых равны степеням двойки. Слабым звеном обоих подходов является необходимость использования плохо параллелизуемого алгоритма умножения матричных полиномов [6,8], который, кроме того, требует увеличения используемой оперативной памяти.

Ещё одно важное замечание состоит в том, что во всех этих алгоритмах, кроме алгоритма Монтгомери, дважды последовательно строятся блоки A^iY, $A \in F^{N\times N}$, $Y \in F^{n\times N}$, $F = GF(2)$, $i = 0, \ldots, \approx 2N/n$, в то время как умножение на матрицу A — трудоёмкая операция.

1. Блочный фактор

Применение блочного фактора, т. е. замены числа используемых столбцов n на $n_1 = ns$, для алгоритмов типа Видемана—Копперсмита весьма эффективно, так как позволяет при помощи распараллеливания в s раз уменьшить время на построение A^iY.

Важно отметить, что алгоритм умножения матричных полиномов, применявшийся при постановке последнего рекорда, потребовал дополнительного, по сравнению с оригинальной версией Копперсмита, объёма оперативной памяти (всего около 1 ТБ при $s = 8$). Таким образом, ограничения оперативной памяти используемого кластера не позволили взять большее значение блочного фактора для уменьшения времени работы программы в целом.

2. Сведение к симметричному случаю

Для того чтобы свести нахождение решения линейной, вообще говоря несимметричной, системы $AX = 0$ к симметричному случаю, вместо матрицы A рассматривают матрицу $A^{\mathrm{T}}A$. В этом случае используемые в ланцошеподобных

алгоритмах последовательные произведения A^iB для некоторого начального вектора или блока векторов B превращаются в $(A^{\mathrm{T}}A)^iB$. Выясним, насколько такое вычисление сложнее.

Поскольку матрица A занимает большой объём оперативной памяти, её режут на вертикальные блоки

$$A = (A_1 \parallel A_2 \parallel \ldots \parallel A_k)$$

и хранят на разных кусках оперативной памяти.

Пусть на j-м процессоре хранится A_j, а также j-й кусок предыдущих результатов, необходимых для текущих вычислений, т. е.

$$(A^{\mathrm{T}}A)^iB = \begin{bmatrix} V_{i1} \\ V_{i2} \\ \ldots \\ V_{ik} \end{bmatrix},$$

где часть V_{ij} блока $V_i = (A^{\mathrm{T}}A)^iB$ также лежит на вычислительном узле с номером j, $j = 1, \ldots, k$. Несложно понять, что операции вычисления «скалярных» произведений $V_i^{\mathrm{T}}V_i$ и умножения блока на «маленькую» матрицу V_ig при такой схеме хранения являются практически идеальными с точки зрения параллельных свойств и требуют минимальных обменов данными между вычислительными узлами.

Величина A_jV_{ij} — блок размера $N \times n$, который вычисляется на вычислительном узле с номером j, на котором хранится соответствующая часть V_{ij} блока V_i и часть A_j матрицы A. Вычисление суммы $AV_i = \sum\limits_{j=1}^{k} A_jV_{ij}$ формально требует полного обмена всеми A_jV_{ij}. Чтобы этого добиться, воспользуемся алгоритмом односторонней «циклической пересылки» [4]. Время на передачу данных при этом алгоритме является постоянной величиной, не зависящей от числа k вычислительных узлов. Кстати, если разделить хранимые на каждом узле векторы пополам и верхнюю часть циклически складывать в одну сторону, а нижнюю — в другую, то, поскольку современные коммутационные сети встречный поток данных обрабатывают параллельно, т. е. без дополнительных затрат по времени, общее время при такой двусторонней пересылке сократится вдвое. Результат вычисления AV_i на узле j умножается на A_j^{T}, затем можно переходить к следующей итерации. Как мы видим, наличие дополнительного умножения на A^{T} увеличивает примерно в два раза время на счёт, но не увеличивает время на пересылки, которое, как правило, является определяющим. Требования к оперативной памяти не меняются. Кроме того, время на счёт может быть уменьшено в разы путём дополнительного увеличения параметра k (числа вычислительных узлов), поскольку алгоритм циклической пересылки при этом не потребует дополнительного времени.

3. Быстрые алгоритмы построения приближений

В последнее время появилось несколько алгоритмов, которые строят приближения, удовлетворяющие нужным свойствам (например, невырожденны в некотором смысле), к формальным рядам рекурсивно [5,7,12]. Суть их в следующем. Пусть имеется приближение $G_1(\lambda) \in F^{m\times m}[\lambda]$ порядка d к формальному ряду $\alpha(\lambda) \in F^{l\times m}[[\lambda]]$:

$$\alpha(\lambda)G_1(\lambda) = O(\lambda^d).$$

Пусть также построено приближение G_2 порядка d к ряду $\alpha(\lambda)G_1(\lambda)/\lambda^d$ и оба эти приближения удовлетворяют некоторым нужным для алгоритма свойствам. Тогда в ряде случаев удаётся доказать, что произведение матриц $G_1 \cdot G_2$, являющееся, очевидно, приближением порядка $2d$, удовлетворяет тем же свойствам. Таким образом, задача построения приближения достаточно большого порядка 2^k сводится к построению двух приближений порядка 2^{k-1} и умножению матричных полиномов. Степени этих полиномов в рассматриваемых случаях не превосходят порядка соответствующего приближения и в среднем по столбцам равны его половине (см., например, [6]). Отметим, что для вычисления матрицы G_2 необходимо, чтобы матрица G_1 уже была вычислена, поэтому данные вычисления не являются параллельными. Легко понять, что для вычисления приближения порядка 2^k этим алгоритмом нам придётся вычислить 2^i приближений порядка 2^{k-i}, $i = 0, 1, \ldots, k$. Общее число необходимых приближений становится в два раза больше их числа при последовательном построении, однако непосредственно строятся только 2^k приближений первой степени. Построение каждого из остальных сводится к одному умножению матричных полиномов, при этом общее число операций в поле коэффициентов рассматриваемых многочленов оценивается величиной $O(2^k \cdot k^c)$, где c — некоторая небольшая абсолютная константа.

Однако для работы с матричными многочленами высокой степени требуется большая оперативная память, которая значительно возрастает из-за необходимости применения быстрых алгоритмов умножения матричных полиномов на одном кластере, что становится критическим для современной техники. Этот дополнительный рост выражается в некотором логарифмическом множителе к объёму оперативной памяти, необходимой для хранения рассматриваемых многочленов. Этого нет при использовании последовательных алгоритмов. Сложность этих алгоритмов несколько выше и составляет $O(2^{2k})$, однако она может быть уменьшена оптимизацией последовательных шагов.

Как мы указывали выше, в алгоритме Видемана—Копперсмита фактически приходится дважды вычислять ряд. В данной статье предложено экономичное решение путём указанного увеличения количества шагов в последовательном алгоритме построения приближений, которое может быть компенсировано упрощением каждого из них.

4. Связь между приближениями рядов по положительным и отрицательным степеням переменной

Пусть

$$\alpha(\lambda) = \sum_{i=0}^{\infty} \alpha_i \lambda^i \in F^{n\times n}[[\lambda]].$$

Рассмотрим преобразование «точка» (иногда его называют «зеркало»), которое действует на формальные ряды по правилу

$$\dot{\alpha}(\lambda) = \alpha(\lambda^{-1}),$$

а на многочлены фиксированной степени t следующим образом (см. также операцию rev в [6]):

$$\dot{Q} = \dot{Q}(\lambda) = \lambda^t Q(\lambda^{-1}).$$

Рассмотрим приближения типа Паде к ряду по положительным степеням:

$$\alpha(\lambda)Q^{(t)}(\lambda) + P^{(t)}(\lambda) = \sum_{i=2t+1}^{\infty} \rho_i^{(t)} \lambda^i,$$

$$Q^{(t)}(\lambda) \in F^{n\times 2n}[\lambda], \quad P^{(t)}(\lambda) \in F^{n\times 2n}[\lambda], \quad \deg Q^{(t)}, \deg P^{(t)} \leqslant t.$$

В некоторых работах (см., например, [5, 7]) приняты несколько другие обозначения:

$$(\alpha(\lambda) \parallel I_n) \begin{pmatrix} Q^{(t)} \\ P^{(t)} \end{pmatrix} = O(\lambda^{2t+1})$$

(I_n — единичная матрица размера n), что будем обозначать как $\operatorname{ord}^+ G^{(t)} \geqslant 2t+1$, где

$$G^{(t)} = \begin{pmatrix} Q^{(t)} \\ P^{(t)} \end{pmatrix}.$$

Матрицу $Q^{(t)}$ будем называть верхней частью $G^{(t)}$.

Нетрудно понять, что $\dot{Q}^{(t)}$, $\dot{P}^{(t)}$ будут матричными приближениями типа Паде к соответствующему ряду по отрицательным степеням переменной:

$$\dot{\alpha}(\lambda)\dot{Q}^{(t)}(\lambda) + \dot{P}^{(t)}(\lambda) = \sum_{i=t+1}^{\infty} \rho_i^{(t)} \lambda^{-i},$$

или

$$(\dot{\alpha}(\lambda) \parallel I_n) \begin{pmatrix} \dot{Q}^{(t)} \\ \dot{P}^{(t)} \end{pmatrix} = O(\lambda^{-(t+1)}),$$

что будем обозначать как $\operatorname{ord}^- \dot{G}^{(t)} \geqslant t+1$, где

$$\dot{G}^{(t)} = \begin{pmatrix} \dot{Q}^{(t)} \\ \dot{P}^{(t)} \end{pmatrix}.$$

По построению для произвольного матричного многочлена подходящего размера $\deg G = \deg \dot{G}$, $\dot{\dot{G}} = G$, $\operatorname{ord}^+ \dot{G} = \operatorname{ord}^- G + \deg G$, или $\operatorname{ord}^+ \dot{G} - 2\deg \dot{G} = \operatorname{ord}^- \dot{G} - \deg G = c(G)$. Значит, приближения Паде при преобразовании «точка» переходят в приближения Паде, а свойство $c(G) \geqslant 1$ является для них характеристическим.

Отметим здесь для дальнейшего, что если

$$\dot{\alpha}(\lambda)\tilde{Q}^{(t)}(\lambda) + \tilde{P}^{(t)}(\lambda) = \sum_{i=t+1-\delta}^{\infty} \tilde{\rho}_i^{(t)} \lambda^{-i}, \quad \delta \in \{0, 1, \ldots, t\}, \quad \deg \tilde{Q}^{(t)}, \deg \tilde{P}^{(t)} = t,$$

то

$$\alpha(\lambda)\dot{\tilde{Q}}^{(t)}(\lambda) + \dot{\tilde{P}}^{(t)}(\lambda) = \sum_{i=2t+1-\delta}^{\infty} \tilde{\rho}_i^{(t)} \lambda^{i},$$

или

$$\operatorname{ord}^+ \dot{\tilde{G}}^{(t)} = \operatorname{ord}^+ \begin{pmatrix} \dot{\tilde{Q}}^{(t)} \\ \dot{\tilde{P}}^{(t)} \end{pmatrix} \geqslant 2t + 1 - \delta.$$

При этом если при $2t \geqslant \delta$ свободный член многочлена $\dot{\tilde{Q}}^{(t)}$ равен нулю, то и свободный член многочлена $\dot{\tilde{P}}^{(t)}$ тоже равен нулю.

Таким образом, $\operatorname{ord}^+ \lambda^{\delta} \dot{\tilde{G}}^{(t)}(\lambda) \geqslant 2t + 1$ и δ младших коэффициентов этого матричного многочлена равны нулю. Приближения, обладающие этим свойством, очевидно, образуют линейное подпространство над $F[\lambda]$.

Как уже отмечалось, мы рассматриваем в данной работе только симметричный случай: $A = A^{\mathrm{T}}$.

Определим для произвольного многочлена

$$Q(\lambda) = \sum_{i=0}^{\deg Q} \lambda^i Q_i \in F^{n \times 2n}[\lambda],$$

матрицы $A \in F^{N \times N}$ и начального блока $B \in F^{N \times 2n}$ блоки векторов $Q(A, B)^-$ и $Q(A, B)^+$ по следующим формулам:

$$Q(A, B)^- = \sum_{i=0}^{\deg Q} A^i B Q_i, \tag{1}$$

$$Q(A, B)^+ = \sum_{i=0}^{\deg Q} A^{\deg Q - i} B Q_i = \dot{Q}(A, B)^-. \tag{2}$$

Отметим сразу, что если рассматривать многочлен Q как многочлен степени $\deg Q + 1$ с нулевым старшим коэффициентом, то определяемый блок получится из предыдущего умножением слева на A, т. е. данное определение зависит от формальной степени многочлена, под которой мы в дальнейшем, если это не ясно из контекста, будем понимать степень ненулевого монома максимальной степени данного многочлена, добавляя в случае необходимости рассмотрения

многочлена как многочлена большей степени явно указанные нулевые мономы больших степеней.

Важно отметить, что если $B \in F^{N\times n}$, $Q^{(i)}(\lambda) \in F^{n\times n}[\lambda]$, $\deg Q^{(i)}(\lambda) = i$ и свободные члены этих многочленов являются невырожденными матрицами, то имеет место равенство линейных пространств над F:

$$\langle A^i B,\ i = 0, \ldots, t\rangle = \langle Q^{(i)}(A,B)^+,\ i = 0, \ldots, t\rangle.$$

Определим скалярное произведение $(Q_1, Q_2)^+$ матричных полиномов $Q_1(\lambda), Q_2(\lambda) \in F^{n\times 2n}[\lambda]$ как коэффициент при $\lambda^{\deg Q_1 + \deg Q_2 + 1}$ в ряде

$$Q_1^{\mathrm{T}}(\lambda)\alpha(\lambda)Q_2(\lambda).$$

Легко убедиться, что это скалярное произведение совпадает с коэффициентом при λ^{-1} в ряде

$$\dot{Q}_1^{\mathrm{T}}(\lambda)\dot{\alpha}(\lambda)\dot{Q}_2(\lambda),$$

т. е. со скалярным произведением $(\dot{Q}_1, \dot{Q}_2)^-$, определённым в [2] и линейным по обоим аргументам. Нетрудно понять, что преобразование «точка» сохраняет произведение,

$$(Q_1 \dot{Q}_2) = \dot{Q}_1\dot{Q}_2,$$

и

$$(Q_1 \dot{+} Q_2) = \dot{Q}_1 \oplus \dot{Q}_2,$$

где $\oplus$ означает сложение полиномов путём сложения коэффициентов при старших степенях, затем при степенях, на единицу меньших, и т. д. с присвоением формальной степени суммы максимума формальных степеней слагаемых. Поэтому в силу (2) определённое нами скалярное произведение $(Q_1, Q_2)^+$ будет линейным относительно операции $\oplus$. Поскольку, как было доказано в [2],

$$\big(\dot{Q}_1(\lambda), \dot{Q}_2(\lambda)\big)^- = (\dot{Q}_1(A,B)^-)^{\mathrm{T}} A \dot{Q}_2(A,B)^-,$$

то

$$\begin{aligned}\big(Q_1(\lambda), Q_2(\lambda)\big)^+ &= \big(\dot{Q}_1(\lambda), \dot{Q}_2(\lambda)\big)^- = \\ &= (\dot{Q}_1(A,B)^-)^{\mathrm{T}} A \dot{Q}_2(A,B)^- = (Q_1(A,B)^+)^{\mathrm{T}} A Q_2(A,B)^+.\end{aligned}$$

5. Последовательный алгоритм

Покажем, как при помощи скалярного произведения матричных полиномов можно последовательно строить решение разреженной симметричной системы линейных уравнений при использовании техники [6]. В этой статье приближения $G^{(i)}(\lambda) \in F^{2n\times 2n}[\lambda]$ ряда $(\alpha(\lambda) \parallel I_n) \in F^{n\times 2n}[[\lambda]]$ строятся последовательно.

Столбцы матрицы

$$G^{(1)}(\lambda) = \begin{pmatrix} I_n & O_n \\ \alpha_0 & \lambda I_n \end{pmatrix}, \tag{3}$$

очевидно, образуют базис приближений порядка 1. Пусть для некоторого i

$$(\alpha(\lambda) \parallel I_n)G^{(i)}(\lambda) = C_i\lambda^i\big(1+O(\lambda)\big), \quad C_i \in F^{n\times 2n}, \tag{4}$$

и столбцы матрицы $G^{(i)}(\lambda)$ образуют базис всех приближений порядка i. Выбирается невырожденная матрица τ_i, при которой $C_i\tau_i$ имеет нижнетреугольный вид; число нулевых столбцов в ней обозначим n_i. Тогда

$$G^{(i+1)}(\lambda) = G^{(i)}(\lambda)\tau_i \begin{pmatrix} \lambda I_{2n-n_i} & O \\ O & I_{n_i} \end{pmatrix}.$$

Поскольку линейное пространство приближений порядка $i+1$ вложено в линейное пространство приближений порядка i, порождённое столбцами $G^{(i)}(\lambda)\tau_i$, то из (4) следует, что столбцы $G^{(i+1)}(\lambda)$ образуют базис приближений порядка $i+1$.

Если обозначить

$$G^{(i)}(\lambda) = \begin{pmatrix} Q^{(i)}(\lambda) \\ P^{(i)}(\lambda) \end{pmatrix}, \quad Q^{(i)}(\lambda),\, P^{(i)}(\lambda) \in F^{n\times 2n}[\lambda],$$

то по построению $\deg Q^{(i)}(\lambda) \leqslant i-1$ и

$$Q^{(i+1)}(\lambda) = \lambda Q^{(i)}(\lambda)\tau_{i1} + Q^{(i)}(\lambda)\tau_{i2} = \lambda Q^{(i)}(\lambda)\tau_{i1} \oplus \big(Q^{(i)}(\lambda)\tau_{i2} + O_{2n}\lambda^{\deg Q^{(i)}+1}\big)$$

для некоторых матриц $\tau_{i1}, \tau_{i2} \in F^{2n\times 2n}$, где $O_{2n} \in F^{n\times 2n}$ — нулевая матрица. Поэтому

$$Q^{(i+1)}(A,B)^+ = Q^{(i)}(A,B)^+\tau_{i1} + AQ^{(i)}(A,B)^+\tau_{i2}.$$

Поскольку $\deg Q^{(i)}(\lambda) = i-1$, то скалярное произведение

$$\big(Q^{(1)}(A,B)^+\big)^{\mathrm{T}} AQ^{(i+1)}(A,B)^+ = \big(Q^{(1)}(\lambda), Q^{(i+1)}(\lambda)\big)^+ = {Q_0^{(1)}}^{\mathrm{T}} C_{i+1} = \begin{pmatrix} C_{i+1} \\ O_n \end{pmatrix} \tag{5}$$

с учётом предыдущего равенства позволяет вычислить сам коэффициент C_{i+1}, а также

$$Q^{(i+1)}(A,B)^+,\ AQ^{(i+1)}(A,B)^+, \ldots,\ A^{k-1}Q^{(i+1)}(A,B)^+$$

непосредственно из блоков

$$Q^{(i)}(A,B)^+,\ AQ^{(i)}(A,B)^+,\ A^2Q^{(i)}(A,B)^+, \ldots,\ A^kQ^{(i)}(A,B)^+$$

без использования коэффициентов ряда $\alpha(\lambda)$, которые в данной версии вычислять не нужно. Таким образом, в шаге алгоритма используется только одно умножение на матрицу A блока $A^{k-1}Q^{(i+1)}(A,B)^+ \in F^{N\times 2n}$ для получения очередной последовательности блоков

$$Q^{(i+1)}(A,B)^+,\ AQ^{(i+1)}(A,B)^+, \ldots,\ A^kQ^{(i+1)}(A,B)^+.$$

Для построения всего пространства Крылова в виде описанных выше блоков нам необходимо построить последовательность приближений

$$\tilde{\tilde{G}}^{(r)}(\lambda) = \begin{pmatrix} \tilde{\tilde{Q}}^{(r)}(\lambda) \\ \tilde{\tilde{P}}^{(r)}(\lambda) \end{pmatrix} \in F^{2n\times n}[\lambda], \quad \deg \tilde{\tilde{G}}^{(r)} = r, \quad r = 0,1,\ldots,\approx \frac{N}{n}, \tag{6}$$

верхняя часть свободных членов которых представляет невырожденные матрицы. Для этого будем использовать приближения $G^{(2r+1)}(\lambda)$, построенные в шагах с нечётными номерами, над которыми будем производить следующие преобразования.

Пусть $d(j, 2r+1)$ — степень j-го столбца $G^{(2r+1)}(\lambda)$ как матричного многочлена.

Шаг 1. Рассмотрим столбцы, для степеней которых выполняется неравенство $d(j, 2r+1) \leqslant r$. Если верхние части их свободных членов образуют матрицу ранга n, то $\tilde{\tilde{G}}^{(r)}(\lambda)$ можно составить из тех из них, верхние части свободных членов которых линейно независимы.

Шаг 2. В противном случае рассмотрим линейное подпространство над $F[\lambda]$ относительно обычного сложения столбцов, для которых $d(j, 2r+1) \leqslant r+1$ и верхние части свободных членов которых нулевые. Поскольку эти столбцы являются приближениями, их свободные члены полностью нулевые. В это пространство войдут умноженные на λ столбцы, выбранные на первом шаге. Если они не являются базисом всего рассматриваемого подпространства, выберем в нём столбцы, образующие базис дополнения. Разделив эти столбцы на λ, получим столбцы степени r порядка $2r$. Дополним ими столбцы, выбранные на первом шаге. Если число выбранных столбцов равно n, мы нашли $\tilde{\tilde{G}}^{(r)}(\lambda)$.

Шаг 3. Если на втором шаге набрать нужные столбцы не удалось, аналогично рассматриваем столбцы с условием $d(j, 2r+1) \leqslant r+2$ и верхними частями двух младших коэффициентов, равными нулю, и т. д.

Обозначим число необходимых итераций $v(r)$ и рассмотрим его как случайную величину, зависящую от матричных коэффициентов используемых здесь приближений, которые будем считать независимыми случайными величинами.

Теорема. *Математическое ожидание* $v(r)$ *не превосходит* 1,76. *При* $\log_2(N/n) > 10$ *с вероятностью* 0,99 $\max\limits_s v(s) \leqslant 4\log_2(N/n)$.

Доказательство. В работе [2] было показано, как построить приближения $\tilde{Q}^{(s)}(\lambda) \in F^{n\times n}[\lambda]$, $\deg \tilde{Q}^{(s)} = s$, с невырожденными старшими коэффициентами $\tilde{Q}_s^{(s)}$ ряда $\dot{\alpha}(\lambda) = \sum\limits_{i=0}^{\infty} \alpha_i \lambda^{-i}$ так, чтобы

$$\dot{\alpha}(\lambda)\tilde{Q}^{(s)}(\lambda) - \tilde{P}^{(s)}(\lambda) = \sum_{i=s+1-\delta(s)}^{\infty} \tilde{\alpha}_i^{(s)} \lambda^{-i} \tag{7}$$

для некоторого $\tilde{P}^{(s)}(\lambda) \in F^{n\times n}[\lambda]$, $\deg \tilde{P}^{(s)} = s$ при небольших значениях $\delta(s)$. При этом ненулевые столбцы в матрицах $\tilde{\alpha}_{s+1-l}^{(s)}$, $l > 0$, расположены справа, а их число $u^{(s)}(l)$ не возрастает с ростом l.

Имеем

$$\alpha(\lambda)\dot{\tilde{Q}}^{(s)}(\lambda) - \dot{\tilde{P}}^{(s)}(\lambda) = \sum_{i=2s+1-\delta(s)}^{\infty} \tilde{\alpha}_i^{(s)} \lambda^{i}.$$

Обозначим

$$\dot{\tilde{G}}^{(r)} = \begin{pmatrix} \dot{\tilde{Q}}^{(r)} \\ \dot{\tilde{P}}^{(r)} \end{pmatrix}.$$

Этот матричный многочлен имеет невырожденную верхнюю часть свободного члена и может быть взят в качестве $\tilde{\tilde{G}}^{(r)}$. Однако шаги 1—3 непосредственно строят подобный многочлен с заведомо не большим значением $\delta(r)$.

Столбцы многочленов $\lambda\dot{\tilde{G}}^{(r)}(\lambda)$, находящиеся на позициях

$$u^{(r)}(2)+1,\ldots,u^{(r)}(1), \tag{8}$$

считая справа, будут лежать в линейных пространствах, описанных в шаге 2.

В случае $\delta(r)=1$ имеем, что $u^{(r)}(2)=0$, и верхние части столбцов свободного члена многочлена $\dot{\tilde{G}}^{(r)}(\lambda)$ на позициях (8) дополняют верхние части свободных членов столбцов, выбранных на первом шаге до матрицы полного ранга n, так как линейное пространство над F столбцов, выбранных на первом шаге в этом случае, содержит столбцы многочлена $\dot{\tilde{G}}^{(r)}(\lambda)$, находящиеся на позициях $u^{(r)}(1)+1,\ldots,n$, считая справа.

Соответственно столбцы, находящиеся на позициях $u^{(r)}(3)+1,\ldots,u^{(r)}(2)$, в многочлене $\lambda^2\dot{\tilde{G}}^{(r)}(\lambda)$ находятся в линейном пространстве, описанном в шаге 3, и т. д. Как было отмечено выше, алгоритм Видемана—Копперсмита строит матричные приближения, столбцы которых являются базисом над $F[\lambda]$ относительно обычного сложения, всех приближений данного порядка. Поэтому рассматриваемые образы могут быть выбраны на шагах 2 и 3. Поэтому число итераций $v(r)$ не превосходит $\delta(r)$.

Как было вычислено в [3], при $\log_2(N/n) > 10$ с вероятностью 0,99 $\max\limits_s \delta(s) \leqslant 4\log_2(N/n)$, в то время как среднее значение $\delta(s)$ не превосходит 1,76. Теорема доказана. □

Общее число шагов построения приближений в нашем алгоритме примерно равно $2N/n$. Для построения пространства Крылова будет использована половина из них.

Построенные многочлены $\tilde{\tilde{Q}}^{(i)}(\lambda)$ имеют степень i и невырожденные свободные члены. Порядок их будет меньше, чем $2i+1$, на число $\delta'(i) \leqslant \delta(i)$:

$$\alpha(\lambda)\tilde{\tilde{Q}}^{(i)}(\lambda) - \tilde{\tilde{P}}^{(i)}(\lambda) = \sum_{l=2i+1-\delta'(l)}^{\infty} \tilde{\tilde{\alpha}}_l^{(i)}\lambda^l.$$

Поэтому при $j < i-\delta'(i)$

$$\big(\tilde{\tilde{Q}}^{(j)}(A,B)^+\big)^{\mathrm{T}} A\tilde{\tilde{Q}}^{(i)}(A,B)^+ = 0,$$

а при $i-\delta'(i) \leqslant j \leqslant i$

$$\big(\tilde{\tilde{Q}}^{(j)}(A,B)^+\big)^{\mathrm{T}} A\tilde{\tilde{Q}}^{(i)}(A,B)^+ = \big(\tilde{\tilde{Q}}^{(j)}(\lambda),\tilde{\tilde{Q}}^{(i)}(\lambda)\big)^+ = \sum_{k+l=j+i+1} \tilde{\tilde{Q}}_k^{(j)T}\tilde{\tilde{\alpha}}_l^{(i)}, \tag{9}$$

где

$$\tilde{\tilde{Q}}^{(j)}(\lambda) = \sum_{l=0}^{j} \tilde{\tilde{Q}}_l^{(j)} \lambda^l$$

и сумма справа в (9) содержит не более $\delta'(l) + 1$ слагаемых.

По построению

$$\tilde{\tilde{Q}}^{(r)}(\lambda) = Q^{(2r+1)}(\lambda) \left(\tau_{r0} + \frac{1}{\lambda} \tau_{r1} + \ldots + \frac{1}{\lambda^{\delta'(r)}} \tau_{r\delta'(r)} \right)$$

для некоторых $\tau_{ri} \in F^{2n \times 2n}$. Поэтому

$$\tilde{\tilde{Q}}^{(r)}(A,B)^+ = \sum_{i=0}^{\delta'(r)} A^i Q^{(2r+1)}(A,B)^+ \tau_{ri},$$

и при $k = \max\limits_i \delta'(i)$ мы, используя имеющуюся на каждом шаге последовательность

$$Q^{(2r+1)}(A,B)^+,\ AQ^{(2r+1)}(A,B)^+, \ldots,\ A^k Q^{(2r+1)}(A,B)^+,$$

можем вычислить $\tilde{\tilde{Q}}^{(r)}(A,B)^+$.

Несложная доортогонализация приводит к последовательному построению A ортогонального базиса W_i пространства Крылова одновременно с очередным слагаемым в сумме

$$X = \sum_{i=0}^{\approx N/n} W_i (W_i^{\mathrm{T}} A W_i)^{-1} W_i^{\mathrm{T}} B, \tag{10}$$

при помощи которого решение системы строится так же, как в алгоритме Монтгомери [11].

Доортогонализация требует временного хранения последовательных блоков пространства Крылова, не ортогональных друг другу, и вычисления их скалярных произведений. Число этих блоков определяется величиной $\delta'(s) \leqslant \delta(s)$. Если ограничиться хранением в оперативной памяти лишь $k \approx 7$ последовательных блоков пространства Крылова, а предыдущие $O(\ln N)$ хранить в долговременной памяти на жёстком диске, то из-за редкого ($\approx N/(2k)$) к ней обращения, это не повлияет на скорость работы всего алгоритма.

Не очень сложно построить процедуру, дающую аппроксимации, удовлетворяющие условиям (7) с $\delta(s) = 1$. Поэтому есть все основания полагать, что реальное время работы нашего алгоритма значительно меньше: шаг 3 не нужен, а иногда не нужен и шаг 2.

Грубая оценка снизу требуемого объёма используемой оперативной памяти даёт оценку

$$\max_s \delta(s) \cdot Nn = O(nN \ln N),$$

что по порядку совпадает с требованиями для алгоритма Томэ [12]. Однако если $\delta(s) \in \{0, 1\}$, эти требования существенно снижаются.

6. Параллельный алгоритм

Коэффициенты C_i в шаге алгоритма можно строить, как в [6], при помощи свёртки полинома $G^{(i)}$ и ряда α. Согласно формулам (9) доортогонализацию тоже можно проводить для многочленов $\tilde{\tilde{Q}}^{(i)}(\lambda)$, а $\tilde{\tilde{\alpha}}_l^{(i)}(\lambda)$ строить при помощи свёрток с рядом α. В результате будем получать ортогональный базис пространства Крылова W_i в виде многочленов $W_i(\lambda)$, для которых $W_i = W_i(A,B)^+$. Поэтому формула (10) даст $X(\lambda)$, $\deg X \approx N/n$, $X = X(A,B)^+$, который вычисляется в результате повторного параллельного вычисления с использованием блочного фактора элементов A^iB, $i = 0, 1, \ldots, \approx N/n$.

7. Заключение

Предложенный в этой работе последовательный алгоритм, по существу, выполняет те же действия, что и алгоритм Монтгомери [11]. Количество умножений на матрицу A в нём вдвое больше. Применение техники построения приближений из [6] позволяет уменьшить время на доортогонализацию получаемых блоков относительно уже построенных, сократив их число с трёх до не более чем одного. Применение параллельных систем в этом случае ограничено необходимостью собирать на каждом шаге матрицы C_i (5).

Если строить последовательные приближения кусками с вычислением части координат решения, то можно совсем отказаться от 3-го и 4-го шагов, запоминая только кусок пространства Крылова. В этом случае не будет необходимости иметь большую память и можно отказаться от использования кластеров большой мощности в решении рассматриваемой задачи. Кроме того, можно запоминать текущий кусок пространства Крылова и не строить его дважды.

Если сравнивать трудоёмкость этого алгоритма с трудоёмкостью алгоритма Монтгомери, то следует отметить, что левый множитель в формуле для вычисления скалярных произведений (5) не меняется и, значит, может храниться на вычислительном узле, вычисляющем правый множитель. Тем самым экономится время на пересылки при вычислении скалярных произведений, которое играет доминирующую роль. Скалярное произведение векторов заменено на скалярное произведение многочленов, которое проще и требует меньше оперативной памяти. Количество сложений векторов в каждых двух шагах предложенного алгоритма примерно в два раза меньше, чем в одном шаге алгоритма Монтгомери, имеющего в два раза меньше шагов. Число векторов, которые надо хранить, примерно в четыре раза меньше.

Аналогичные модификации могут быть применены и для алгоритма построения σ-базиса [5].

Конечно, рассматриваемый алгоритм, как и алгоритм Видемана—Копперсмита, использует вдвое большее число умножений на большую разреженную матрицу, чем алгоритм Монтгомери. Однако эти умножения используются в процедуре последовательного вычисления коэффициентов ряда, которая легко парал-

лелизуется на вычислительные ресурсы, не связанные быстрыми каналами, т. е. может быть проделана в Интернете.

Литература

[1] Нестеренко Ю. В., Черепнёв М. А. и др. Отчёт по хоздоговору «Исследование алгоритмических подходов к решению систем алгебраических уравнений над конечными полями на многопроцессорных кластерах. — Алгоритм-К, 2008.

[2] Черепнёв М. А. Блочный алгоритм типа Ланцоша решения разреженных систем линейных уравнений // Дискрет. мат. — 2008. — Т. 20, № 1. — С. 145—150.

[3] Astakhov V. V. Estimates of the running time and memory requirements of the new algorithm of solving large sparse linear systems over the field with two elements // J. Tambov State Univ. The works of participants of Int. conf. «ParCA» presented according to the results of reviewing by International Program Committee. — 2010. — Vol. 15, no. 4. — P. 1311—1327.

[4] Barnett M., Littlefield R., Payne D.G., van de Geijn R. Global combine on mesh architectures with wormhole routing // J. Parallel and Distributed Comput. Arch. — 1995. — Vol. 24, no. 2.

[5] Beckermann B., Labahn G. A uniform approach for the fast computation of matrix-type Padé approximants // SIAM J. Matrix Anal. Appl. — 1994. — Vol. 15, no. 3. — P. 804—823.

[6] Coppersmith D. Solving homogeneous linear equations over $GF(2)$ via block Widemann algorithm // Math. Comput. — 1994. — Vol. 62, no. 205. — P. 333—350.

[7] Giorgi P., Jeannerod C.-P., Villard G. On complexity of polynomial matrix computations // ISSAC'03. Proc. of the Int. Symp. on Symbolic and Algebraic Computation, August 3—6, Philadelphia, USA, 2003. — New York: ACM, 2003. — P. 135—142.

[8] Jeannerod C.-P., Villard G. Asymptotically fast polynomial matrix algorithms for multivariable systems // Int. J. Control. — 2006. — Vol. 79, no. 11. — P. 1359—1367.

[9] Kaltofen E. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems // Math. Comput. — 1995. — Vol. 64, no. 210. — P. 777—806.

[10] Kleinjung T., Aoki K., Franke J., Lenstra A. K., Thomé E., Bos J. W., Gaudry P., Kruppa A., Montgomery P. L., Osvik D. A., Riele H., Timofeev A., Zimmermann P. Factorization of a 768-bit RSA modulus. Version 1.0. — 2010. — `http://eprint.iacr.org/2010/006.pdf`.

[11] Montgomery P. L. A clock Lanczos algorithm for finding dependencies over $GF(2)$ // Adv. Cryptology — EuroCrypt'95 / L. C. Guillou, J.-J. Quisquater, eds. — Berlin: Springer, 1995. — (Lect. Notes Comput. Sci.; Vol. 921). — P. 106—120.

[12] Thomé E. Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm // J. Symbol. Comput. — 2002. — Vol. 33, no. 5. — P. 757—775.

[13] Villard G. A study of Coppersmith's block Wiedemann algorithm using matrix polynomials: RR 975-I-M IMAG Grenoble France. — 1997.