A connection of series approximations and the basis of the Krylov space in block algorithms of Coppersmith and Montgomery.

Mikhail Cherepniov[1]

Moscow State University, Moscow Leninskiye gory 1, mechanics-mathematics faculty, number theory department, Russia

Abstract. In this paper some properties of Wiedemann-Coppersmith algorithm are studied. In particular, when matrix of solving linear system is symmetric, orthogonal bases of Krylov space is constructed with the help of approximations of formal series from this algorithm. Here some modifications, that explore this properties is proposed.

This paper studies the problem of solving large sparse linear systems over $\mathbb{F} = GF(2)$, that is a part of integer factorization algorithm. Now block Montgomery 12, and Wiedemann-Coppersmith 11 algorithms are used for this purpose. The next of them was speeding up by Thomé 1 with the help of additional memory volume. This volume is too big for parameters that give fastest implementation 2.

In this paper we examine common principles of Montgomery and Wiedemann-Coppersmith algorithms to explore advantages of each other.

Mention that in both Wiedemann-Coppersmith and Montgomery's algorithms solution constructs with the help of its coefficients in Krylov space. It's easy to see that solution coefficients in Krylov space is a coefficients of some rather good approximation of the series with positive degrees of formal value. They may be found by sequentially increasing of the order of the approximations as it was made by Coppersmith in 11, or with comparatively small number of approximations with orders equals to degrees of two as in [9,10]. Mention that in these works constructs a bases of all approximations with the help of asymptotically faster algorithm with complexity bound $O(Nlog^2NloglogN)$, contrary to $O(N^2)$ for Coppersmith's one. However constant in O in the first bound is rather big and increase with block factor, that decrease time for the whole algorithm when parallelization is used. In record calculations (12.12.2009 was factorized RSA-768 with 232 decimal digits or 768 bits) was utilized the procedure 2 same to the Coppersmith's, but that constructs only "good" approximations with degrees equals to the degrees of two. Weak of both procedures is necessity to utilize an algorithm of matrix polynomial product 4, 11 that have a bad parallel properties and demands to increase memory.

Another remark is that in all algorithms except Montgomery's, sequence $A^iY, A \in F^{N\times N}, Y \in F^{n\times N}, F = GF(2), i = 0, \ldots, \approx \frac{2N}{n}$ constructs twice and product on matrix A is hard.

1 Block factor

Utilization of the block factor that is replacement of n by $n_1 = ns$, is very effective for algorithms like Wiedemann-Coppersmith algorithm, because parallelization on s sites may s times decrease time for construction A^iY.

It should be mentioned that matrix polynomial multiplication algorithm, that was utilize in 2, demands additional memory (1TB total, when $s = 8$). So restriction of cluster memory make it impossible to take optimal value s to make total time minimal.

2 Reduction to the symmetric case

For reduction of solving nonsymmetric linear system to symmetric one, a replacement of A by A^TA is usual. So calculations A^iB for some B obtain the form $(A^TA)^iB$. How harder is it?

Because of the big size, matrix A divides into vertical blocks

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

that are stored on the different parts of memory.

Let calculation node with number j stores A_j, and corresponding parts of some previous calculations. Let

$$(A^TA)^iB = \begin{bmatrix} V_{i1} \\ V_{i2} \\ \cdots \\ V_{ik} \end{bmatrix}, \tag{1}$$

where V_{ij} stores on node with number j too, $j = 1, \cdots, k$. It's clear that inner products $V_i^TV_i$ and products V_ig will have good parallel properties, because of the minimal communication between such nods.

Matrix $A_jV_{ij} \in \mathbb{F}^{N\times n}$ is calculated on nod with number j, that stores V_{ij} and A_j. A summarizing of $AV_i = \sum_{j=1}^{k} A_jV_{ij}$ demands total exchange of A_jV_{ij}. This is realized with the help of cyclic sending algorithm 5. Time for sending in this algorithm doesn't depend on k. Mention that if we separate vectors stored on the each nod in two parts and each part sum in different directions, then common time of summation decreases in two times in modern duplet sets. The result AV_i is multiplied on A_j^T by node number j, so we may start the next iteration. So addition multiplication by A^T increases time for calculation in two times, but doesn't increase time for sending, that is defining. Memory demands is the same. Moreover, calculation time may be decreased with the help of increasing the number of calculation nodes k.

3 Algorithms which construct approximations fast

There are several algorithms, that construct approximations for formal series in recursive way 1, 9, 10. The main idea is: Let formal series $\alpha(\lambda) \in \mathbb{F}^{l\times m}[[\lambda]]$ has approximation $G_1(\lambda) \in \mathbb{F}^{m\times m}[\lambda]$ with order d:

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

Let G_2 is the same approximation for $\frac{\alpha(\lambda)G_1(\lambda)}{\lambda^d}$, and both of them satisfy some important properties for our algorithm. In several cases it is possible to demonstrate that product $G_1 \cdot G_2$, that is the approximation of order $2d$, satisfies this properties too. So we construct approximation with the order 2^k by construction of two approximations of order 2^{k-1} and multiplication of matrix polynomials. In considered cases, the degrees of these polynomials are no more than the order of corresponding approximations, and the mean value of degrees of columns is equal to the half of it. It's important, that when we start the calculation of G_2, the calculation of G_1 must be finished. So these calculations are not parallel. It's easy to see, that for the calculation of approximation of order 2^k by this algorithm, it is necessary to calculate 2^i approximations of order 2^{k-i}, $i = 0, 1, \ldots, k$. Total number of approximations became two times more, then in corresponding sequential construction. On the other hand, only 2^k approximations of the first order are calculated directly. The calculation of each of other are replaced with one matrix polynomial multiplication with total number of operations in the field of coefficients of considered polynomials bounded with $O(2^k \cdot k^c)$, where c is some absolute not great constant.

A work with matrix polynomials of the high degree demands a big memory volume. This volume considerably increase when we use fast algorithm of polynomial matrix multiplication, and this becomes critical. This additional increase expressed in some logarithmic factor to demanded memory. This factor escapes when sequential algorithm used. Running time of such algorithm in original is longer: $O(2^{2k})$, but, it may be reduced by every step optimization.

As it was mentioned above, Wiedemann-Coppersmith's algorithm constructs series twice. In this article we propose an economy in this place. Corresponding increase of number of steps may be compensate with simplification of each of them.

4 The correspondence between approximations of series with positive and negative powers of formal value.

Let

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

Let's consider a transformation "point" (sometimes it is named a "mirror", see *rev* in 11):

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

for series, and

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

for polynomials with degree t.

Let's consider Pade approximations to the series with positive powers of formal value:

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

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

Sometimes (9, 10) another notations are accepted:

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

(I_n is a unit matrix with the size n), that we will denote as $ord^+ G^{(t)} \geq 2t+1$, or $ord^+ Q^{(t)} \geq 2t+1$, where

$$G^{(t)} = \begin{pmatrix} Q^{(t)} \\ P^{(t)} \end{pmatrix}. \tag{3}$$

Let's denote an upper part of $G^{(t)}$ by $Q^{(t)}$.

It's easy to see that $\dot{Q}^{(t)}, \dot{P}^{(t)}$ will be Pade approximations for corresponding series with negative powers of formal value:

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

or

$$(\dot{\alpha}(\lambda)\|I_n) \begin{pmatrix} \dot{Q}^{(t)} \\ \dot{P}^{(t)} \end{pmatrix} = O(\lambda^{-(t+1)}), \tag{4}$$

that we will denote as $ord^- \dot{G}^{(t)} \geq t+1$, or $ord^- \dot{Q}^{(t)} \geq t+1$, where

$$\dot{G}^{(t)} = \begin{pmatrix} \dot{Q}^{(t)} \\ \dot{P}^{(t)} \end{pmatrix}. \tag{5}$$

It's clear by construction, that for arbitrary matrix polynomial with corresponding size: $deg G = deg \dot{G}$, $\dot{\dot{G}} = G$, $ord^+ \dot{G} = ord^- G + deg G$, or $ord^+ \dot{G} - 2deg \dot{G} = ord^- \dot{G} - deg G = c(G)$. So Pade approximations transform by "point" to Pade approximations, and the property $c(G) \geq 1$ is characteristic for them..

Let's mention here, that if

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

then

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

or

$$ord^{+}\dot{\tilde{G}}^{(t)}=ord^{+}\begin{pmatrix}\dot{\tilde{Q}}^{(t)}\\ \dot{\tilde{P}}^{(t)}\end{pmatrix}\geq 2t+1-\delta. \tag{6}$$

If $2t\geq\delta$, and constant term of the polynomial $\dot{\tilde{Q}}^{(t)}$ is equal to zero, then constant term of the polynomial $\dot{\tilde{P}}^{(t)}$ is equal to zero too.

So $ord^{+}\lambda^{\delta}\dot{\tilde{G}}^{(t)}(\lambda)\geq 2t+1$, and δ lowest terms of this matrix polynomial are equal to zero. Approximations, with these properties, evidently, formes a linear subspace over $\mathbb{F}[\lambda]$.

As it was mentioned, we consider symmetric case: $A=A^{T}$.

Let's define for arbitrary polynomial $Q(\lambda)=\sum_{i=0}^{degQ}\lambda^{i}Q_i\in\mathbb{F}^{n\times 2n}[\lambda]$, matrix $A\in\mathbb{F}^{N\times N}$ and start block $B\in\mathbb{F}^{N\times 2n}$ blocks of vectors $Q(A,B)^{-}$ and $Q(A,B)^{+}$ by formulas

$$Q(A,B)^{-}=\sum_{i=0}^{degQ}A^{i}BQ_i, \tag{7}$$

$$Q(A,B)^{+}=\sum_{i=0}^{degQ}A^{degQ-i}BQ_i=\dot{Q}(A,B)^{-}. \tag{8}$$

Let's mention, that if we consider polynomial Q, as the polynomial of degree $degQ+1$ with zero constant term, then a defined block may be obtained from the previous by left multiplying by A. So, corresponding definition depends on formal degree of polynomial, that we define, if it is not clear, as a degree of non zero monomial of considered polynomial with a maximal degree. We will append zero terms, if it will be necessary to consider this polynomial as polynomial with bigger degree, which is not clear by the context.

Let's mention, that if $B\in\mathbb{F}^{N\times n},Q^{(i)}(\lambda)\in\mathbb{F}^{n\times n}[\lambda],degQ^{(i)}(\lambda)=i$, and constant terms of this polynomials are nonsingular, then the equality of linear spaces over $\mathbb{F}$ is realized:

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

Let's define a scalar product $(Q_1, Q_2)^+$ of matrix polynomials $Q_1(\lambda), Q_2(\lambda) \in \mathbb{F}^{n\times 2n}[\lambda]$ as a coefficient at $\lambda^{degQ_1+degQ_2+1}$ in the series

$$Q_1^T(\lambda)\alpha(\lambda)Q_2(\lambda).$$

It's easy to see, that this scalar product is equal to the coefficient at λ^{-1} in the series

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

that is equal to scalar product $(\dot{Q}_1, \dot{Q}_2)^-$, defined in 8, and linear in both arguments.

Transformation "point" preserves product:

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

and

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

where $\oplus$ is a sum of polynomials, when main terms are summed and a degree of this sum is a maximum of degrees of items. So, from (8) we obtain that defined $(Q_1, Q_2)^+$ will be linear relative to $\oplus$. As it was proved in 8,

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

so,

$$\begin{aligned}(Q_1(\lambda), Q_2(\lambda))^+ &= (\dot{Q}_1(\lambda), \dot{Q}_2(\lambda))^- = (\dot{Q}_1(A, B)^-)^T A\dot{Q}_2(A, B)^- \\ &= (Q_1(A, B)^+)^T AQ_2(A, B)^+.\end{aligned}$$

5 Sequential algorithm

We will show, how to construct solution with the help of sequential technic of 11 and a defined scalar product. Approximations $G^{(i)}(\lambda) \in \mathbb{F}^{2n\times 2n}[\lambda]$ of the series $(\alpha(\lambda) \parallel I_n) \in \mathbb{F}^{n\times 2n}[[\lambda]]$ construct as follows:

Columns of matrix

$$G^{(1)}(\lambda) = \begin{pmatrix} I_n & O_n \\ \alpha_0 & \lambda I_n \end{pmatrix}, \tag{9}$$

evidently forme a bases of approximations of order one. Let for some i:

$$(\alpha(\lambda) \parallel I_n)G^{(i)}(\lambda) = C_i\lambda^i(1 + O(\lambda)), C_i \in \mathbb{F}^{n\times 2n}. \tag{10}$$

Let τ_i is a nonsingular matrix, that gives matrix $C_i\tau_i$ in lower triangular form, and we will denote a number of nonzero columns in it as n_i. Then

$$G^{(i+1)}(\lambda) = G^{(i)}(\lambda)\tau_i \begin{pmatrix} \lambda I_{2n-n_i} & O \\ O & I_{n_i} \end{pmatrix}. \tag{11}$$

Since linear space of approximations of order $i+1$ is inserted in linear space of approximations of order i with columns of $G^{(i)}(\lambda)\tau_i$ as bases, then we obtain, that columns $G^{(i+1)}(\lambda)$ form bases of approximations of order $i+1$ from (10) .

If we denote

$$G^{(i)}(\lambda) = \begin{pmatrix} Q^{(i)}(\lambda) \\ P^{(i)}(\lambda) \end{pmatrix}, Q^{(i)}(\lambda), P^{(i)}(\lambda) \in \mathbb{F}^{n\times 2n}[\lambda], \tag{12}$$

then under construction $degQ^{(i)}(\lambda) \leq i-1$, and

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

where $O_{2n} \in \mathbb{F}^{n\times 2n}$ is a zero matrix, and $\tau_{i1}, \tau_{i2} \in \mathbb{F}^{2n\times 2n}$. So,

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

Since $degQ^{(i)}(\lambda) = i-1$, then scalar product

$$(Q^{(1)}(A,B)^+)^T AQ^{(i+1)}(A,B)^+ = (Q^{(1)}(\lambda), Q^{(i+1)}(\lambda))^+ = {Q_0^{(1)}}^T C_{i+1} = \begin{pmatrix} C_{i+1} \\ O_n \end{pmatrix} \tag{13}$$

with the help of previous equality, gives C_{i+1}, and $Q^{(i+1)}(A,B)^+, AQ^{(i+1)}(A,B)^+, \ldots, A^{k-1}Q^{(i+1)}(A,B)^+$ may be calculated from $Q^{(i)}(A,B)^+, AQ^{(i)}(A,B)^+, A^2Q^{(i)}(A,B)^+, \ldots, A^kQ^{(i)}(A,B)^+$, without using coefficients of the series $\alpha(\lambda)$, a calculation of which is unnecessary. So only one multiplication of matrix A by block $A^{k-1}Q^{(i+1)}(A,B)^+ \in \mathbb{F}^{N\times 2n}$ exists in each step of our algorithm, and it gives us $Q^{(i+1)}(A,B)^+, AQ^{(i+1)}(A,B)^+, \ldots, A^kQ^{(i+1)}(A,B)^+$.

For the construction of the whole Krylov space as linear space of blocks, it is necessary to construct the sequence of approximations

$$\tilde{\tilde{G}}^{(r)}(\lambda) = \begin{pmatrix} \tilde{\tilde{Q}}^{(r)}(\lambda) \\ \tilde{\tilde{P}}^{(r)}(\lambda) \end{pmatrix} \in \mathbb{F}^{2n\times n}[\lambda], deg\tilde{\tilde{G}}^{(r)} = r, r = 0, 1, \ldots, \approx \frac{N}{n}, \tag{14}$$

that have nonsingular matrix in upper parts of constant terms. We will use approximations $G^{(2r+1)}(\lambda)$, that was constructed in the steps with odd numbers, and will do transformations as follows:

Let $d(j, 2r+1)$ is a degree of column with a number j of $G^{(2r+1)}(\lambda)$, as matrix polynomial.

Iteration 1: Consider columns with degrees $d(j, 2r+1) \leq r$. If upper parts of their constant terms forme a matrix with rang n, than $\tilde{\tilde{G}}^{(r)}(\lambda)$ may be formed by those of them, that have linear independent upper parts of constant terms.

Iteration 2: In other case we will consider linear space of columns over $\mathbb{F}[\lambda]$, with respect to ordinary addition, which degrees $d(j, 2r+1) \leq r+1$, and upper parts of constant terms are zeros. Since this columns are approximations their

constant terms are whole zeros. Columns, that was selected on the first iteration, multiplying by λ, will be inserted in this space. If they are not a bases of the whole considered subspace, we may select columns that forms bases of the complementation in it. If we divide this columns by λ, we will obtain degree r and order $2r$. Let's complement columns, selected on the first iteration, with them. If a number of selected columns is equal to n, than we obtain $\tilde{\tilde{G}}^{(r)}(\lambda)$.

Iteration 3: If on the 2 iteration we didn't obtain necessary columns, let's consider columns with the property $d(j, 2r+1) \leq r+2$ and upper parts of two lowest terms which are equal to zero and so on... .

Let's denote $v(r)$ a number of necessary iterations and consider it as a random value dependent on matrix coefficients of using approximations, which we will consider as independent random values.

Theorem 1. *Mean value* $v(r) \leq 1,76$. *If* $\log_2 \frac{N}{n} > 10$, *then probability that* $\max_s \delta(s) \leq 4 \log_2 \frac{N}{n}$ *is more than 0,99.*

Proof. In 8 it was shown, how to construct approximations $\tilde{Q}^{(s)}(\lambda) \in \mathbb{F}^{n\times n}[\lambda], deg\tilde{Q}^{(s)} = s$, to the series $\dot{\alpha}(\lambda) = \sum_{i=0}^{\infty} \alpha_i \lambda^{-i}$ with nonsingular principal terms $\tilde{Q}_s^{(s)}$, so that

$$\dot{\alpha}(\lambda)\tilde{Q}^{(s)}(\lambda) - \tilde{P}^{(s)}(\lambda) = \sum_{i=s+1-\delta(s)}^{\infty} \tilde{\alpha}_i^{(s)} \lambda^{-i}, \tag{15}$$

for some $\tilde{P}^{(s)}(\lambda) \in \mathbb{F}^{n\times n}[\lambda], deg\tilde{P}^{(s)} = s$, where $\delta(s)$ is not big. Nonzero columns of matrixes $\tilde{\alpha}_{s+1-l}^{(s)}, l > 0$, situated on the right, and with a quantity $u^{(s)}(l)$ that doesn't increase when l increases.

We have

$$\alpha(\lambda)\dot{\tilde{Q}}^{(s)}(\lambda) - \dot{\tilde{P}}^{(s)}(\lambda) = \sum_{i=2s+1-\delta(s)}^{\infty} \tilde{\alpha}_i^{(s)} \lambda^{i}. \tag{16}$$

Let's denote

$$\dot{\tilde{G}}^{(r)} = \begin{pmatrix} \dot{\tilde{Q}}^{(r)} \\ \dot{\tilde{P}}^{(r)} \end{pmatrix}. \tag{17}$$

This matrix polynomial has nonsingular upper part of constant term and may be taken as $\tilde{\tilde{G}}^{(r)}$. However iterations 1-3 construct such a polynomial directly with value no bigger then $\delta(r)$.

Columns of matrix polynomials $\lambda\dot{\tilde{G}}^{(r)}(\lambda)$, on positions

$$u^{(r)}(2) + 1, \ldots, u^{(r)}(1), \tag{18}$$

regarding from the right, will situate in the linear space, that was describe in iteration 2.

In the case of $\delta(r) = 1$ we have: $u^{(r)}(2) = 0$, and upper parts of columns of polynomial $\dot{\tilde{G}}^{(r)}(\lambda)$ on positions (18) complement upper parts of constant terms of columns, selected on iteration 1, to full rank matrix (nonsingular). This is because linear space over $\mathbb{F}[\lambda]$ of columns, selected on the 1 iteration, contains columns of $\dot{\tilde{G}}^{(r)}(\lambda)$ situated on positions $u^{(r)}(1)+1,\ldots,n$, regarding from the right.

Columns, situated on positions $u^{(r)}(3)+1,\ldots,u^{(r)}(2)$ in matrix polynomial $\lambda^2\dot{\tilde{G}}^{(r)}(\lambda)$ situate in the linear space, described in iteration 3, and so on ... . As it was mentioned above, the Wiedemann-Coppersmith algorithm constructs matrix approximations which columns forms bases over $\mathbb{F}[\lambda]$, with respect to ordinary addition, of all approximations of corresponding order. So, considered images may be chosen in 2 and 3 iterations and so on ... . The whole number of iterations, that is necessary to construct approximation with nonsingular constant term is no more than $\delta(r)$.

As it was calculated in 13, if $\log_2 \frac{N}{n} > 10$, then with probability 0,99 $\max_s \delta(s) \leq 4\log_2 \frac{N}{n}$, and mean value $\delta(s)$ is no more than 1,76.

The whole number of steps of construction of approximations in our algorithm is approximately equal to $\frac{2N}{n}$. We use half of them for construction Krylov space.

Approximations $\tilde{\tilde{Q}}^{(i)}(\lambda)$, that we have constructed, have a degree i, nonsingular constant terms and order less than $2i+1$ by number $\delta'(i) \leq \delta(i)$:

$$\alpha(\lambda)\tilde{\tilde{Q}}^{(i)}(\lambda) - \tilde{\tilde{P}}^{(i)}(\lambda) = \sum_{l=2i+1-\delta'(l)}^{\infty} \tilde{\tilde{\alpha}}_l^{(i)}\lambda^l, \tag{19}$$

Therefore, if $j < i - \delta'(i)$, we have:

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

and when $i - \delta'(i) \leq j \leq i$, we have:

$$(\tilde{\tilde{Q}}^{(j)}(A,B)^+)^T A\tilde{\tilde{Q}}^{(i)}(A,B)^+ = (\tilde{\tilde{Q}}^{(j)}(\lambda), \tilde{\tilde{Q}}^{(i)}(\lambda))^+ = \sum_{k+l=j+i+1} \tilde{\tilde{Q}}_k^{(j)T}\tilde{\tilde{\alpha}}_l^{(i)}, \tag{20}$$

where $\tilde{\tilde{Q}}^{(j)}(\lambda) = \sum_{l=0}^{j} \tilde{\tilde{Q}}_l^{(j)}\lambda^l$, and sum to the right in (20) contains no more than $\delta'(l)+1$ terms.

Under construction, $\tilde{\tilde{Q}}^{(r)}(\lambda) = Q^{(2r+1)}(\lambda)(\tau_{r0} + \frac{1}{\lambda}\tau_{r1} + \ldots + \frac{1}{\lambda^{\delta'(r)}}\tau_{r\delta'(r)})$ for some $\tau_{ri} \in \mathbb{F}^{2n\times 2n}$. Therefore,

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

and if $k = \max_i \delta'(i)$ we can calculate $\tilde{\tilde{Q}}^{(r)}(A,B)^{+}$ from

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

in every step of our algorithm.

Not hard additional orthogonalization gives next term of A orthogonal bases W_i of Krylov space at the same time with the next term in sum:

$$X = \sum_{i=0}^{\approx \frac{N}{n}} W_i (W_i^T A W_i)^{-1} W_i^T B. \tag{21}$$

Then we construct solution with the help of this block and standard Montgomery's technic from 12.

Additional orthogonalization demands storing of some sequential blocks from orthogonal bases of Krylov space, and calculation of scalar product of them to the next constructed block. A number of these blocks is bounded with value $\delta'(s) \leq \delta(s)$.

It's not very hard to construct a procedure, that forms approximations, satisfying condition (15) with $\delta(s) = 1$. So, we have a good reason for supposing, that real running time of our algorithm is shorter, and iteration 3 is not necessary, sometimes iteration 2 is not necessary too.

The lower bound of memory is

$$\max_s \delta(s) \cdot Nn = O(nNlnN),$$

that is approximately equal to demands of algorithm Thomé 1. If $\delta(s) \in \{0,1\}$, these demands become much less.

6 Parallel algorithm

Coefficients C_i can be constructed as in 11 with the help of convolution of polynomial $G^{(i)}$ and series α in every step of described algorithm. Because of (20), additional orthogonalization can be done for polynomials $\tilde{\tilde{Q}}^{(i)}(\lambda)$, and $\tilde{\tilde{\alpha}}_l^{(i)}(\lambda)$ are constructed with the help of convolution with the series α. As a result, we obtain orthogonal bases of Krylov space in the view of polynomials $W_i(\lambda)$, that give $W_i = W_i(A,B)^{+}$. So, formula (21) gives $X(\lambda), degX \approx \frac{N}{n}$, and $X = X(A,B)^{+}$, that is constructed when we calculate $A^i B, i = 0, 1, \ldots, \approx \frac{N}{n}$ repeatedly.

7 Conclusion

Proposed sequential algorithm do similar steps to algorithm P.Montgomery 12. The number of multiplications by A in it is two times more. Utilization of technic from 11 permit to decrease time for additional orthogonalization of constructed blocks relative to previous, with the help of reduction of their number from 3 to 1 or less. It's hard to utilize parallel systems here because of the necessity to gather matrix C_i (13) in every step.

If one construct sequential approximations in parts with calculation of corresponding parts of solution coordinates, then it is not necessary to do 3 and 4 steps. In that case we must store only some part of Krylov space that demands less memory. So not so big cluster is need. Besides one can remember only part of Krylov space and not construct it twice.

To compare with the Montgomery's algorithm, mention that left multiple in (13) is constant. So scalar products may be calculate on the computer nod that obtain right multiple. This eliminate time for transfers, that is dominant. Scalar product of vectors replaced with scalar product of polynomials and take less time and memory. Number of vector additions in two steps is approximately two times less then in Montgomery's algorithm that have two times less steps. Number of stored vectors is four times less.

Analogous algorithm can be made for σ-bases construction 10.

Of cause, our algorithm, like Wiedemann-Coppersmith algorithm, demands products on big sparse matrix two times more than in Montgomery's algorithm. Nevertheless this products are in procedure of construction of series coefficients. This procedure may be very good parallelized in the Internet.

References

Thomé E.: Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm. Journal of Symbolic Computation, 33:5, 757–775 (2002)

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, January 7, 2010. http://eprint.iacr.org/2010/006.pdf

Kaltofen E.: Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. Math. Comp., 64(210), 777–806 (1995)

Jeannerod C.-P. and Villard G. Asymptotically fast polynomial matrix algorithms for multivariable systems, Int. J. Control, 79(11):1359- 1367, (2006)

Barnett M., Littlefield R., Payne D.G., van de Geijn R.: Global Combine on Mesh Architectures with Wormhole Routing. Journal of Parallel and Distributed Computing archive, Volume 24, Issue 2 (Feb. 1, 1995)

Villard G.: A study of Coppersmith's block Wiedemann algorithm using matrix polynomials. RR 975-I-M IMAG Grenoble France, (April 1997)

Nesterenko Y.V., Cherepnev M.A., and others: Tecnical report "Investigation of algorithms of solving of systems of algebraic equations over finite fields on clusters", (2008)

Cherepnev M.A.: Block Lacos based algorithm of solvingsparce linear systems. Diskret. Math., v.20, no.1, p.145–150 (2008)

Giorgi P., Jannerod C-P., Villard G.: On Complexity of Polynomial Matrix Computations. ISSAC'03, August 3–6, Philadelphia, USA (2003)

Beckermann B. and Labahn G.: A uniform approach for the fast computation of Matrix-type Pade approximants. SIAM J., Matrix Analysis and Applications (1994)

Coppersmith D.: Solving homogeneous linear systems over GF(2) via block Widemann algorithm. Mathematics of Computation, vol. 62, no. 205 (1994)

Montgomery P.L.: A Block Lanczos Algorithm for Finding Dependencies over GF(2). Advances in Cryptology - EuroCrypt'95 / Louis C.Guillou and Jean-Jacques Quisquater, rditors. Berlin: Springer-Verlag (Lect. Notes in Comp. Sci. V.921) 106–120 (1995)

Astakhov V.V.: Estimstes of the running time and memory requirements of the new algorithm of solving large sparse linear systems over the field with two elements. A Journal of Tambov State University, The works of participants of International conference "ParCA" presented according to the results of reviewing by International Program Commettee, V.15, Iss.4, .1311-1327 (2010)