CONVERGENCE RATE IMPROVEMENT OF RICHARDSON AND NEWTON-SCHULZ ITERATIONS

A PREPRINT

Alexander Stotsky
Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg SE - 412 96, Sweden
alexander.stotsky@chalmers.se
alexander.stotsky@telia.com

August 21, 2020

ABSTRACT

Fast convergent, accurate, computationally efficient, parallelizable, and robust matrix inversion and parameter estimation algorithms are required in many time-critical and accuracy-critical applications such as system identification, signal and image processing, network and big data analysis, machine learning and in many others.
This paper introduces new composite power series expansion with optionally chosen rates (which can be calculated simultaneously on parallel units with different computational capacities) for further convergence rate improvement of high order Newton-Schulz iteration. New expansion was integrated into the Richardson iteration and resulted in significant convergence rate improvement. The improvement is quantified via explicit transient models for estimation errors and by simulations. In addition, the recursive and computationally efficient version of the combination of Richardson iteration and Newton-Schulz iteration with composite expansion is developed for simultaneous calculations.
Moreover, unified factorization is developed in this paper in the form of tool-kit for power series expansion, which results in a new family of computationally efficient Newton-Schulz algorithms.

Keywords Least Squares Estimation · Efficient Parallel Iterative Solvers · Tool-Kit for Matrix Power Series Factorization · Computationally Efficient High Order Newton-Schulz and Richardson Algorithms · Simultaneous Calculations · Convergence Acceleration of Richardson Iteration

1 Introduction

Least squares method is widely used in control, system identification, signal processing, [1] - [4], statistics, [5] as well as in many computational applications such as emerging big data applications, [6], machine learning, [7] and in many other areas. For accurate solution many least squares problems (for example the problems related to data, signal and image processing, system identification, network analysis and many others) can be associated with calculation of the parameter vector θ_*, which satisfies the algebraic equation

$$A\theta_* = b \tag{1}$$

where b is the vector, and the matrix A is SPD (Symmetric and Positive Definite) matrix. For example, the matrix A is SPD for the systems with harmonic regressor, [2], [8] and multiplication of any invertible matrix A by its transpose transforms the system to the SPD case with the Gram matrix, [9]. The numerical stability problems associated with ill-conditioning of the Gram matrix can be solved using different types of preconditioning techniques, see for example, [10], [11] and references therein (see also Section 7 for simulations of the ill-conditioned matrices).
Iterative methods for solving (1) are often preferable (especially for large-scale systems) due to simplicity, better accuracy and robustness, less processor time and memory space compared to direct methods. The most general and

well-known method for iterative calculation of the matrix inverse is high order Newton-Schulz algorithm described in [12] - [21] and in many other publications. The second order version of Newton-Schulz iteration, see for example [22] - [24] is the most known.
High order Newton-Schulz algorithms are well-discussed in the literature. However, the questions associated with the relation between high order Newton-Schulz algorithms and power series expansions were not properly studied. The paper [25], which was the first paper with the description of the relation between second order Newton-Schulz algorithm and power series expansion does not provide complete description of this relation.
Reduction of the computational complexity of high order Newton-Schulz algorithm is one the most important challenges in this area. The computational complexity can be reduced via factorizations of the power series, see for example [17], [21], [26] - [35] and references therein. Practical applications of these factorizations (excepting Horner's rule) are hampered by the lack of general unified description.
Computational resources with high degree of parallelism (instead of single computing units) will be available in the future for implementation of numerical methods. The computational performance of iterative solvers can also be increased via parallel computing (especially for large scale systems), achieved for example, via multiprocessor and virtual systems, [36] - [38]. In order to improve the performance a serial algorithm is usually converted to parallel algorithm, see for example [39]. This paper proposes a new approach for convergence rate improvement where novel iterative algorithms are designed with high degree of parallelism (or enhanced parallelism). In other words, the iterative algorithm is designed as a number of independent computational parts (the number of parts is associated with the degree of parallelism) which can be executed simultaneously. The challenges associated with computational efficiency are addressed already on the design level in this case, providing new opportunities for high performance parallel processing.
This paper introduces new composite power series expansion with optionally chosen rates and high degree of parallelism for further convergence rate improvement in the unified framework described in [40]. New expansion applied to Richardson iteration resulted in significant improvement of the convergence rate. Simulation results are presented for quantification of the improvements of new algorithms compared to recent algorithms described in [40]. Moreover, explicit transient models are derived for all new algorithms described in this paper. In addition, the recursive and computationally efficient version of the combination of Richardson iteration and Newton-Schulz iteration with composite expansion is developed for simultaneous calculations.
Finally, factorization tool-kit is developed in this paper for general power series expansion, which allows nested applications and results in a family of new computationally efficient algorithms.
This paper is organized as follows. The paper starts with the representation of Newton-Schulz iteration as power series expansion in Section 3. A unified power series factorization for reduction of computational complexity is introduced in Section 4 and represented in the form of tool-kit in Section 9. New high order Newton-Schulz algorithms with composite polynomial are presented in Section 5. Richardson iteration with high order convergence accelerator is described in Section 6 and compared to existing algorithms by simulation in Section 7. The paper ends with brief conclusions in Section 8.

This paper was presented on the 21-st IFAC World Congress in Berlin, Germany, July 12-17, 2020, [41].

2 Splitting & Preconditioning

Numerical solution of the system of linear equations (1) using power series expansions requires splitting and preconditioning. Any positive definite and symmetric matrix A, whose inverse should be calculated can be split as follows, see for example [42] and references therein :

$$A = S - D \tag{2}$$
$$I - S^{-1}A = S^{-1}D \tag{3}$$
$$\rho(I - S^{-1}A) = \rho(S^{-1}D) < 1 \tag{4}$$

where the spectral radius $\rho(\cdot)$ defined in (4) is less than one for symmetric and positive definite matrices A and S (where S^{-1} is the preconditioner), provided that $2S - A$ is a positive definite matrix, [43].
For example, the matrix S can be chosen as a diagonal matrix, which contains the diagonal elements of SDD (Strictly Diagonally Dominant) and positive definite matrix A, see [44] for the general case and [45] for systems with harmonic regressor.
For positive definite (not SDD) matrix A the simplest preconditioner can be chosen as $S^{-1} = I/\alpha$ with $\alpha = \|A\|_\infty/2 + \varepsilon$, where $\|\cdot\|_\infty$ is the maximum row sum matrix norm, and $\varepsilon > 0$ is a small positive number, [11], [46].
Other types of preconditioning can be found in [42], [43], [47], [48], see also references therein.

3 Newton-Schulz Iteration as Fast Power Series Expansion

The results presented in this Section introduce computationally efficient factorization of initial power series and show several steps (step by step) of fast matrix power series expansion which coincide with Newton-Schulz iteration. The relation between Newton-Schulz approach and power series expansion opens new opportunities for reduction of the computational complexity of Newton-Schulz algorithms.
The following initial power series factorization :

$$G_0 = \sum_{j=0}^{w-1}(S^{-1}D)^{(p+1)j}\ \{\sum_{d=0}^{p}(S^{-1}D)^d\}S^{-1} \tag{5}$$

$$= \sum_{j=0}^{h-1}(S^{-1}D)^j S^{-1} = (I-(S^{-1}D)^h)A^{-1} \tag{6}$$

$$F_0 = I - G_0 A = (S^{-1}D)^h \tag{7}$$

where (7) defines initial inversion error, and $p = 0, 1, 2, ...$, $w = 1, 2, 3, ...$, and $h = w(p+1) = 1, 2, 3...$, gives the starting point for the following steps of Newton-Schulz iteration:

Step 1.

$$G_1 = \{\sum_{j=0}^{n-1} F_0^j\}\ G_0 = \{\sum_{j=0}^{n-1}(S^{-1}D)^{hj}\}\ G_0$$
$$[I + (S^{-1}D)^h + (S^{-1}D)^{2h} + ... + (S^{-1}D)^{(n-1)h}]$$
$$[I + (S^{-1}D) + (S^{-1}D)^2 + ... + (S^{-1}D)^{h-1}]S^{-1}$$
$$= [I + (S^{-1}D) + ... + (S^{-1}D)^{(hn-1)}]S^{-1} \tag{8}$$
$$F_1 = I - G_1 A = (S^{-1}D)^{hn} \tag{9}$$

where G_1 in (9) is calculated via (8).

Step 2.

$$G_2 = \{\sum_{j=0}^{n-1} F_1^j\}\ G_1 = \{\sum_{j=0}^{n-1}(S^{-1}D)^{hnj}\}\ G_1$$
$$F_2 = I - G_2 A = (S^{-1}D)^{hn^2} \tag{10}$$

Further evaluation in Step k gives classical high order Newton-Schulz algorithm (11) and error model (12) :

$$G_k = \{\sum_{j=0}^{n-1} F_{k-1}^j\}\ G_{k-1} \tag{11}$$

$$F_k = I - G_k A = F_0^{n^k} = (S^{-1}D)^{hn^k} \tag{12}$$

where G_k is estimate of A^{-1}, $n = 2, 3, ...$ and $k = 1, 2, 3, ...$
Notice that the factorization similar to (5) can be applied to the power series (11) for improvement of computational efficiency. To this end the unified factorization method is developed in the next Section.

4 Reduction of Computational Complexity via Unified Factorization: Nested Algorithms

Consider the following matrix power series:

$$Z = \{\sum_{j=0}^{h-1} Y^j\}\ X \tag{13}$$

$$Y = I - XA \tag{14}$$

where X, Y, Z are matrices of corresponding dimensions, I is the identity matrix, $h = 2, 3, 4, ...$. Realization of the algorithm (13) requires h mmm (matrix-by-matrix multiplications) per iteration loop according to Horner's scheme, see

for example [21], [40].
Notice that Horner's rule is not optimal for evaluating matrix polynomials, [17] and for reduction of the computational complexity the power series (13) can be factorized as follows:

$$Y = I - XA \tag{15}$$

$$U = \{\sum_{d=0}^{p} Y^d\} X \tag{16}$$

$$Y^{p+1} = I - UA \tag{17}$$

$$Z = \{\sum_{j=0}^{w-1} Y^{(p+1)j}\} U \tag{18}$$

$$h = w\,(p+1),\ \ p = 0, 1, 2, ...,\ w = 1, 2, 3, ... \tag{19}$$

$$N_p = p + w + 1 \tag{20}$$

where N_p is the number of multiplications for realization of the algorithm (15) - (18) of the order h defined in (19). Indeed, realization of (15) - (17) requires $p+2$ multiplications, and realization of the series (18) which can be calculated as follows:

$$Z_i = Y^{p+1} Z_{i-1} + U,\ for\ i = 1 : (w-1),\ Z_0 = U \tag{21}$$

requires $w-1$ multiplications.
Notice that $N_p = p + 1$ and $N_p = w$ in (20) for the case where $w = 1$ and $p = 0$ respectively.
Notice that the idea of factorization (15) - (18) is associated with the Newton-Schulz iteration (11), where the sum $\sum_{j=0}^{w-1} Y^{(p+1)j}$ in (18) corresponds to $\sum_{j=0}^{n-1} F_{k-1}^{j}$ and U in (16) is associated with G_{k-1}.
Representation (13) and (14) includes Newton-Schulz algorithm (11), (12) with $X = G_{k-1}$, $Y = F_{k-1}$, and $Z = G_k$. In addition, equation (18) represents (5) with $Y = (S^{-1}D)$ and $X = S^{-1}$ and can be used for computationally efficient calculations of the initial power series. Algorithm (15)-(18) describes unified and systematic way for power series factorization, order reduction and improvement of the computational efficiency. Application of the algorithm (15) - (18) to factorization of Newton-Schulz iteration of orders $2-19$ is demonstrated in the form of tool-kit in Appendix, see Section 9.
The number of mmm for conventional recursive realization of high order Newton-Schulz algorithm is equal to the algorithm order. Factorization (15)-(18) reduces the number of mmm to (20) for the order (19). The reduction of computational complexity is quantified in Figure 1, where the order (which is equal to the number of mmm for conventional realization) is plotted with colored surface and the number of mmm for (15)-(18) is plotted with a white surface. The complexity can be essentially reduced for higher orders. The efficiency index introduced in [49] has the following form for the factorization (15)-(18):

$$EI = [w\,(p+1)]^{1/(p+w+1)} \tag{22}$$

Notice that sequential application of the factorization (15)-(18) implies further reduction of the computational complexity for higher orders. The sum (18) can also be calculated more efficiently for specific orders compared to (21). Nested application of (15)-(18) is illustrated by the following example.
Example: Nested algorithm as unification of the hyperpower iteration method (described in [35]) of order 45 that requires 10 mmm only.
Newton-Schulz iteration of the order 45 can be factorized in a step-wise way as follows:

$$Z = (I + Y^9 + ... + Y^{36})\ \ [I + Y + ... + Y^8]\ X \tag{23}$$

$$= (I + Y^9 + ... + (Y^9)^4)\ \ [I + Y^3 + Y^6]\ [I + Y + Y^2]\ X \tag{24}$$

$$= \{I + (I + (Y^9)^2)\ (Y^9 + (Y^9)^2)\}\ \ [I + Y^3 + Y^6]$$
$$[3I + (XA)\ (-3I + XA)]\ X \tag{25}$$

$$Y^3 = I - [\sum_{d=0}^{2} Y^d]\ X\ A,\ Y^9 = I - [\sum_{d=0}^{8} Y^d]\ X\ A$$

Algorithm (23) represents the algorithm (15) - (20) of the order $h = 45$ defined in (19) with $p = 8$ and $w = 5$ and requires $p + w + 1 = 14$ mmm. Further application of the algorithm (15) - (20) to the eighth order polynomial[1] in (23)

[1] Other type of factorization of the eighth order polynomial is presented in Appendix, Table 2 for h = 9

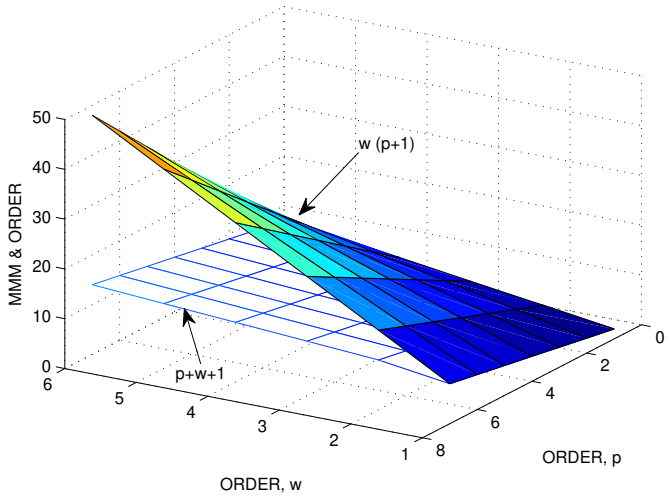

Figure 1: The order $h = w\ (p+1)$ (which is equal to the number of mmm for conventional realization), where $p = 1, 2, ..., 7,\ w = 1, 2, ..., 6$ of the factorization (15)-(18) is plotted with colored surface and the number of mmm $N_p = p + w + 1$ is plotted with a white surface.

and factorization of the fourth order polynomial (w.r.t. Y^9) results in algorithms (24), (25), which requires ten mmm only with EI $= 1.4633$.
Notice that nested method for derivation of the computationally efficient algorithms based on (15) - (20) is more simple compared to the method described in [35]. Moreover, the method is universal (compared to heuristic methods) and applicable to any order, see Section 9.
The efficiency index EI does not account for robustness with respect to error accumulation. Minimization of the number of mmm makes iteration more robust.
Notice also that the efficiency index of classical Newton-Schulz iteration of the second order, which is the most robust, is EI $= 1.4142$, see for example [29], [30] for comparisons of the efficiency indexes.

5 Novel Newton-Schulz Algorithm with Composite Polynomial and Enhanced Parallelism for Simultaneous Calculations

The unified framework for convergence rate improvement of high order Newton-Schulz matrix inversion algorithms was proposed in [40] . The following new composite power series expansion for Newton-Schulz iteration with different expansion rates for further convergence rate improvement extends this framework as follows :

$$G_k = \underbrace{T_c}_{\substack{\text{Composite}\\ \text{Polynomial}}} + \underbrace{\Gamma_c}_{\substack{\text{Composite}\\ \text{Residual}}} \underbrace{\{\sum_{j=0}^{n-1} F_{k-1}^j\}\, G_{k-1}}_{\substack{\text{Newton-Schulz}\\ \text{Iteration}}} \tag{26}$$

$$T_c = \sum_{i=1}^{w} \{\prod_{p=0}^{i-1} \Gamma_p\} T_i = T_1 + \Gamma_1 T_2 + ... + \prod_{p=1}^{w-1} \Gamma_p T_w \tag{27}$$

$$\Gamma_c = \prod_{p=1}^{w} \Gamma_p \tag{28}$$

where T_c is the composite power series expansion and the composite residual is defined as the product of the residual terms $\Gamma_c = \Gamma_1 \Gamma_2 \Gamma_3 ... \Gamma_w$ with the spectral radius $\rho(\Gamma_i) < 1$, and $\Gamma_0 = I$. Power series expansions T_i satisfy the

following relations:

$$T_i A = I - \Gamma_i \,, \quad i = 1, ..., w \tag{29}$$

$$T_c A = I - \Gamma_c \tag{30}$$

Multiplication of both sides of equation (26) by A together with the relation (30) results in the following error model:

$$F_k = \Gamma_c F_{k-1}^n \tag{31}$$

where the following spectral radius is less than one, $\rho(\Gamma_c) \leq \rho(\Gamma_1)...\rho(\Gamma_w) < 1$ according to Gelfand's formula provided that the matrices Γ_i commute.
The error model (31) with composite power series expansion is the same as the error model (22) in [40] for a single power series. The advantages of composite expansion are discussed below.
The power series expansions T_i (which can be calculated simultaneously on parallel computational units) can be taken as

$$T_i = \{\sum_{j=0}^{x_i-1} (S^{-1}D)^j\} S^{-1} = (I - \Gamma_i) A^{-1} \tag{32}$$

$$\Gamma_i = (S^{-1}D)^{x_i} \tag{33}$$

The rate of expansion x_i can be chosen using computational capacity of each parallel computational unit (fast power series expansion should be implemented on more powerful computational unit). For example x_i can be taken as a polynomial which is a function of step number k or as rapidly expanding power series associated with high order Newton-Schulz iteration with $x_i = m^k$, $m = 2, 3, ...$, see for example [40] for this and other choices.
Notice that the algorithm (26) - (28) has especially simple form for $n = w = m = 1$, $G_k = (S^{-1}D)G_{k-1} + S^{-1}$ which was derived in [50] directly from splitting.

5.1 Double Newton-Schulz Algorithm with High Order Residual as Convergence Accelerator

The advantages of the framework described above are especially pronounced for case when choosing a number of the same rapid expansions T_i (high order Newton-Schulz iterations for example) with the expansion rate associated with the order n in (26). Fast and computationally efficient algorithms can be designed in this case.
Consider algorithm (26) with $T_1 = T_2 = ... = T_k$, where T_k and Γ_k are defined in (32),(33) with $w = n$ and $x_i = hn^k$, $h, n = 1, 2, ...$:

$$\Gamma_k = I - L_{k-1} A \tag{34}$$

$$L_k = \{\sum_{j=0}^{n-1} \Gamma_k^j\} \, L_{k-1} \tag{35}$$

$$\Gamma_k^n = I - L_k A \tag{36}$$

$$G_k = \underbrace{L_k}_{\substack{\text{Newton-Schulz} \\ \text{Iteration}}} + \underbrace{\Gamma_k^n}_{\substack{\text{High Order} \\ \text{Convergence} \\ \text{Accelerator}}} \underbrace{\{\sum_{j=0}^{n-1} F_{k-1}^j\} \, G_{k-1}}_{\substack{\text{Newton-Schulz} \\ \text{Iteration}}} \tag{37}$$

$$F_k = I - G_k A = \Gamma_k^n \, F_{k-1}^n \tag{38}$$

$$F_k = (S^{-1}D)^h \, (k \, n^{k+1} + \, n^k) \tag{39}$$

where $L_0 = \{\sum_{j=0}^{n-1} \Gamma_0^j\} \, T_0$, $\Gamma_0 = I - T_0 A$, and $T_0 = G_0$ (G_0 is calculated via (5)) are precalculated. The algorithm (34) - (39) has two Newton-Schulz loops (which can be calculated simultaneously) of the same order n associated with inversion errors (34) and (38). The sums $\sum_{j=0}^{n-1} \Gamma_k^j$ and $\sum_{j=0}^{n-1} F_{k-1}^j$ can be calculated recursively using Horner's scheme, [40] or factorizations, see Section 9.
Notice that the algorithm derived in [50] and the algorithm (15) in [40] are special cases of the algorithm (34) - (37) for $n = h = 1$ and $n = 1$ and $h \geq 1$ respectively.
Remark 1. Comparison of the error model (39) with the error model (12) of classical high order Newton-Schulz

algorithm shows that the algorithm (34) - (38) has significantly higher convergence rate due to the term $k\ n^{k+1}$.
Remark 2. The algorithm similar to (34) - (39) was proposed in [51]. The algorithm written in the following form:

$$Z_k = \sum_{j=0}^{p-1}(I - Z_{k-1}A)^j\ Z_{k-1} \tag{40}$$

$$G_k = G_{k-1} + (I - G_{k-1}A)\ Z_k \tag{41}$$

has also two Newton-Schulz loops, where both Z_k and G_k are the estimates of the matrix inverse and $p = 2, 3, ...$ is the order.
Algorithm (40), (41) has the following error model

$$L_k = L_{k-1}^p,\ \ L_k = I - Z_k A \tag{42}$$

$$F_k = F_{k-1}L_{k-1}^p,\ \ F_k = I - G_k A \tag{43}$$

where L_k and F_k are estimation errors.
The algorithm (34) - (39) has faster convergence due to the high order error F_{k-1}^n in the error model (38) compared to algorithm (40), (41) which has the error model (43) with the first order error F_{k-1}.

6 Richardson Iteration with High Order Convergence Accelerator

6.1 Algorithm Description

Combination of Richardson iteration, see [52] and matrix inversion algorithms was proposed first in [53] for improvement of the convergence rate of estimated parameters. A number of combinations of Richardson iteration with matrix inversion techniques has been developed in recent years, see for example [11] and [54] - [56] and references therein. Unified framework for many combinations was proposed recently in [40]. New matrix inversion algorithms described in the previous Section can be integrated into the Richardson iteration within this unified framework.
The parameter vector in (1) can be estimated via recursive algorithm as follows:

$$\theta_k = \theta_{k-1} - \underbrace{[\ L_k + \Gamma_k^n\ \{\sum_{j=0}^{q-1} F_k^j\}\ G_k]}_{\substack{\text{Fast Matrix Inversion}\\ \text{Algorithm}}}\ \underbrace{\{A\theta_{k-1} - b\}}_{\substack{\text{Parameter}\\ \text{Estimation Error}}} \tag{44}$$

where θ_k is the estimate of θ_* and Γ_k, L_k, F_k and G_k are calculated in (34) - (38) and $q = 1, 2, ...$ is the order of Neumann series.
The following model is valid for estimation error $\tilde{\theta}_k = \theta_k - \theta_*$:

$$\tilde{\theta}_k = \Gamma_k^n\ F_k^q\ \tilde{\theta}_{k-1} \tag{45}$$

$$\tilde{\theta}_k = (S^{-1}D)^h\ \{(k\ n^{k+1} +\ n^k)\ q + n^{k+1}\}\ \tilde{\theta}_{k-1} \tag{46}$$

The error model has especially simple form for the case where $q = n$:

$$\tilde{\theta}_k = \Gamma_k^n\ F_k^n\ \tilde{\theta}_{k-1} \tag{47}$$

$$\tilde{\theta}_k = (S^{-1}D)^h\ (k\ n^{k+2} +\ 2\ n^{k+1})\ \tilde{\theta}_{k-1} \tag{48}$$

$$\tilde{\theta}_k = (S^{-1}D)^{\gamma_k}\ \tilde{\theta}_0 \tag{49}$$

$$\gamma_k = h\ n^2\ \frac{(k\ n^{k+2} - (k-1)\ n^{k+1} - 2\ n^k - n + 2)}{(n-1)^2} \tag{50}$$

where $n > 1$, $\tilde{\theta}_0 = \theta_0 - \theta_*$ and $\theta_0 = L_0 b$.
The error model (49) shows significant improvement of the convergence rate of estimated parameters in the algorithm (44). This improvement is associated with introduction of the fast matrix inversion algorithms in the Richardson loop, and it is quantified in the next Section.
Notice that the algorithm (44) can also be seen as an extension of the unified framework of Richardson iteration, [40] where the multiplicative high order accelerator Γ_k^n and additional Newton-Schulz loop L_k were introduced for convergence rate improvement.

Remark 3. Stability analysis of combinations of Richardson iteration and matrix inversion methods described in [53] and [54], [56] is based on the residual error model $r_k = b - A\theta_k$, whereas the analysis in [11], [40] and [55] (including the analysis above) is presented in terms of the parameter mismatch $\tilde{\theta}_k$, where $A^{-1}r_k = \theta_* - \theta_k = -\tilde{\theta}_k$. Notice that the parameter mismatch is widely used in the area of system identification for stability analysis, [1] - [4] and allows simplified representation of the error models in unified Richardson and Newton-Schulz framework. Such error models simplify essentially the stability analysis, which allows integration of more sophisticated algorithms (which in turn could essentially improve convergence rate) into the framework.

6.2 Reduction of Computational Complexity via Recursive and Simultaneous Calculations

For development of the computationally efficient version the algorithm (44) is presented in the following form:

$$\theta_k = \theta_{k-1} - \omega_k \left(A\theta_{k-1} - b\right) \tag{51}$$

$$\omega_k = L_k + \Gamma_k^n \{\sum_{j=0}^{n-1} F_k^j\} G_k \tag{52}$$

Recursive algorithm for calculation of ω_k described below is divided in two independent computational parts for simultaneous calculations.

Calculations of both parts start with calculation of the $\sum_{j=0}^{n-1} \Gamma_k^j$, where $\Gamma_k = \Gamma_{k-1}^n$.

1) The first part is associated with the calculation of G_k in (37) via ω_{k-1} as follows:

$$G_k = [\sum_{j=0}^{n-1} \Gamma_k^j] \, [\omega_{k-1} - \sum_{j=0}^{n-1} F_{k-1}^j \, G_{k-1}] + \sum_{j=0}^{n-1} F_{k-1}^j \, G_{k-1} \tag{53}$$

which requires one matrix multiplication only and further calculation of $\sum_{j=0}^{n-1} F_k^j \, G_k$ with G_k defined in (53) which in turn can be further divided in independent parts (and calculated for example according to Horner's scheme or factorizations, see Section 9).

2) The second part is associated with calculations of L_k and Γ_k^n in (35) and (36) respectively using $\sum_{j=0}^{n-1} \Gamma_k^j$.

The results of both parts are merged in (52) to be included in the Richardson iteration (51).
Notice that the matrix-by-vector product $A\theta_{k-1}$ in (51) can be easily calculated in parallel via methods described for example in [36].

7 Comparisons & Quantification of the Performance

Numerical calculation of the parameter vector θ_* for the system (1) where the ill-conditioned SPD information matrix A associated with the system with harmonic regressor with three frequencies, [11], [45], [55] is chosen for comparisons. The performance evaluation is presented in the following three parts.
1) The convergence rate of new matrix inversion algorithm (34) - (39) is compared to the convergence rate of recent algorithm with improved convergence rate described in [40], see Figure 2. The Figure shows that convergence rate improvements are more pronounced for higher orders and larger step numbers.
2) Comparison of the convergence rate of the parameter estimation algorithm (44) and the Richardson iteration with improved convergence rate described in [40], is presented in Figure 3. The algorithm (44) with high order convergence accelerator improves essentially the convergence rate compared to existing algorithms even for lower orders and small step numbers. Indeed, comparison of the Figure 2 and Figure 3 shows that new algorithms are the most beneficial in the Richardson framework.
3) Finally, the performance evaluation of the algorithm (44) with respect to classical Newton-Schulz algorithm is presented in Figure 4. The Figure shows that the convergence rate of the Richardson iteration with convergence accelerator of the order three is comparable to the rate of classical Newton-Schulz algorithm (applied to the parameter estimation problem) of the order eight. Parameter estimation accuracy of the Richardson iteration (44) is about five times higher compared to the accuracy of classical Newton-Schulz algorithm in finite digit calculations.

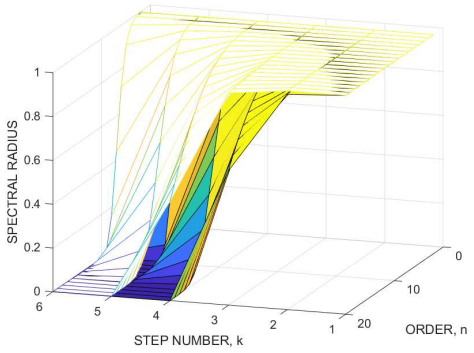

Figure 2: The Figure shows the spectral radius $\rho^{(k\, n^{k+1} + \, n^k)}$ for double Newton-Schulz matrix inversion algorithm defined in (39) (described in Section 5.1), plotted as colored surface. The spectral radius for Newton-Schulz iteration with improved convergence rate described in [40], $\rho^{(k+1)\, n^k}$ is plotted as white surface. Both surfaces are plotted for the spectral radius of ill-conditioned case as functions of the order n and step number k.

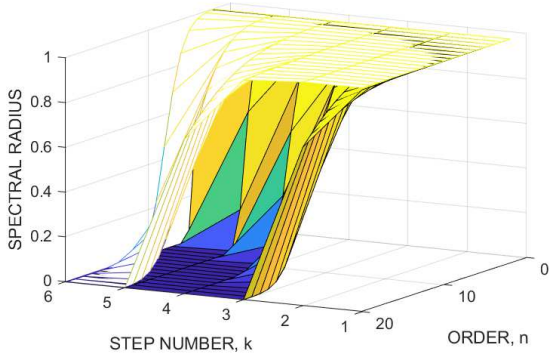

Figure 3: The Figure shows the spectral radius $\rho^{\,h\, n^2\, \frac{(k\, n^{k+2} - (k-1)\, n^{k+1} - 2\, n^k - n + 2)}{(n-1)^2}}$ for Richardson iteration (for parameter estimation) defined in (49) (plotted as colored surface). The spectral radius for Richardson iteration with improved convergence rate described in [40], $\rho^{2h\{\frac{n^{k+3} - n^4}{(n-1)^3} - (k-1)\{\frac{n^3}{(n-1)^2} + \frac{k}{2(n-1)}\} + k(n+2)\}}$ is plotted as white surface. Both surfaces are plotted for the spectral radius of ill-conditioned case as functions of the order n and step number k for $h = 1$.

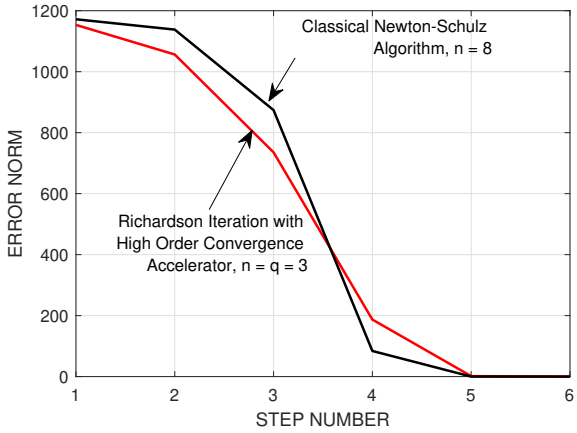

Figure 4: The Figure shows the error of the estimated parameters of Richardson iteration with convergence accelerator of the third order (plotted with a red line) and the parameter error of classical eight-order Newton-Schulz algorithm, plotted with a black line. Both algorithms converge in five steps.

8 Conclusion

This paper shows that the most general and well-known Newton-Schulz iteration is fast power series expansion and presents unified framework and tool-kit for power series factorization and reduction of the computational complexity. The framework allows reduction of complexity of many algorithms and factorization of the algorithm of the order 45 that requires 10 mmm only is presented as example.

Main result of the paper is new composite power series expansion for Newton-Schulz iteration with high degree of parallelism for the convergence rate improvement and computational efficiency. Comparative analysis of the convergence rates of new algorithms and exiting ones is performed via explicit transient models. New algorithms have faster convergence than known Newton-Schulz iterations. Moreover, new expansion resulted in significant improvement of the convergence rate of Richardson iteration for which recursive and computationally efficient version was developed. The results were also confirmed by simulations.

The paper opens new opportunities for convergence rate improvement of Newton-Schulz and Richardson iterations via computationally efficient composite expansions to be implemented on parallel machines with different computational performance.

References

[1] Ljung, L. & Söderström T., Theory and Practice of Recursive Identification. The MIT press series in signal processing, optimization, and control; Vol. 4, MIT Press, (1983).

[2] Fomin V., Fradkov A. and Yakubovich V., Adaptive Control of Dynamic Objects, Nauka, Moscow (1981) (in Russian).

[3] Ljung L., System Identification: Theory for the User, Prentice-Hall, Upper Saddle River, NJ. (1999).

[4] Gustafsson F., Adaptive Filtering and Change Detection, John Wiley & Sons, Ltd, (2000).

[5] Bates, D. & Watts D. Nonlinear Regression Analysis and its Applications. New York: Wiley, (1988).

[6] Wolberg J., Data Analysis Using the Method of Least Squares: Extracting the Most Information from Experiments. Berlin: Springer, (2005).

[7] Lagoudakis M., Parr R. and Littman M., Least-Squares Methods in Reinforcement Learning for Control, Methods and Applications of Artificial Intelligence, Proceeding of the Second Hellenic Conference on AI, SETN 2002, Thessaloniki, Greece, pp. 249-260, April (2002).

[8] Bayard, D., A General Theory of Linear Time-Invariant Adaptive Feedforward Systems with Harmonic Regressors. IEEE Trans. Autom. Control vol. 45, N 11, pp. 1983-1996, (2000).

[9] Björck Å., Numerical Methods for Least Squares Problems, SIAM, First edition, April 1, (1996).

[10] Benzi M., Preconditioning Techniques for Large Linear Systems: A Survey, Journal of Computational Physics vol. 182, pp. 418-477, (2002).

[11] Stotsky A., Accuracy Improvement in Least-Squares Estimation with Harmonic Regressor: New Preconditioning and Correction Methods, 54-th CDC, Dec. 15-18, Osaka, Japan, pp. 4035-4040, (2015).

[12] Isaacson E. and Keller H. , Analysis of Numerical Methods, John Wiley & Sons, New York, (1966).

[13] Petryshyn W., On Generalized Inverses and on the Uniform Convergence of $(I^{\smile}\beta K)^n$ with Application to Iterative Methods, J. Math. Anal. Appl., vol. 18, pp. 417-439, (1967).

[14] Zlobec S., On Computing the Generalized Inverse of a Linear Operator, Glasnik Mat-Fiz. Astronom. Ser. II Drushtvo Mat. Fiz. Hrvatske vol. 22, pp. 265-271, (1967).

[15] Garnett J., Ben-Israel A., Yau S., A Hyperpower Iterative Method for Computing Matrix Products Involving the Generalized Inverse, SIAM J. Numer. Anal., N 8, pp. 104-109, (1971).

[16] Sen S. and Prabhu S., Optimal Iterative Schemes for Computing Moore-Penrose Matrix Inverse, Int. J. Sys. Sci. vol. 8, pp. 748-753, (1976).

[17] Stickel E., On a Class of High Order Methods for Inverting Matrices, ZAMM Z. Angew. Math. Mech. 67, pp. 331-386, (1987).

[18] Climent J., Thome N. and Wei Y. , A Geometrical Approach on Generalized Inverses by Neumann-type Series, Linear Algebra Appl., vol. 332–334 pp. 533-540, (2001).

[19] Li W. and Li Z., A Family of Iterative Methods for Computing the Approximate Inverse of a Square Matrix and Inner Inverse of a Non-Square matrix, Applied Mathematics and Computation, vol. 215, N 9, pp. 3433-3442, (2010).

[20] Chen H. and Wang Y., A Family of Higher-order Convergent Iterative Methods for Computing the Moore-Penrose Inverse, Applied Mathematics and Computation vol. 218, pp. 4012-4016, (2011).

[21] Pan V., Soleymani F. and Zhao L., Highly Efficient Computation of Generalized Inverse of a Matrix, arXiv:1604.07893v1 [math.RA], (2016).

[22] Schulz G., Iterative Berechnung Der Reziproken Matrix, Zeitschrift für Angewandte Mathematik und Mechanik, vol. 13, pp. 57-59, (1933).

[23] Demidovich B., Maron I., Basics of Numerical Mathematics, Moscow, Fizmatgiz, 660 pages (in Russian), (1963).

[24] Ben-Israel A. , A Note on an Iterative Method for Generalized Inversion of Matrices, Math. Comput. vol. 20, pp. 439–440, (1966).

[25] Janiszowski K., Inversion of Square Matrices in Processors with Limited Calculation Abillities, International Journal of Applied Mathematics and Computer Science, AMSC, vol. 13, N 2, pp. 199-204, (2003).

[26] Sharifi M., Arab M., Haghani F., Finding Generalized Inverses by a Fast and Efficient Numerical Method, Journal of Computational and Applied Mathematics, vol. 279, pp. 187-191, (2015).

[27] Soleymani F., Stanimirovic P.,Ullah M., An Accelerated Iterative Method for Computing Weighted Moore-Penrose Inverse, Appl. Math. Comput. vol. 222, pp. 365-371, (2013).

[28] Soleymani, F., An Efficient and Stable Newton-type Iterative Method for Computing Generalized Inverse, $A^{(2)}_{T,S}$, Numer. Algorithms vol. 69, N3, pp. 569-578, (2015).

[29] Soleimani F., Stanimirovic P., Soleymani F., Some Matrix Iterations for Computing Generalized Inverses and Balancing Chemical Equations, Algorithms vol. 8, pp. 982 - 998, (2015).

[30] Soleymani, F., Stanimirovic, P., Haghani, F., On Hyperpower Family of Iterations for Computing Outer Inverses Possessing High Efficiencies, Linear Algebra Appl., vol. 484,pp. 477-495, (2015).

[31] Buranay S., Subasi, D. and Iyikal O., On the Two Classes of High Order Convergent Methods of Approximate Inverse Preconditioners for Solving Linear Systems, Numer. Linear Algebra Appl., vol. 24, N , Article ID e2111, (2017).

[32] Esmaeilic H., Erfanifar R. and Rashidi M., A Fourth-Order Iterative Method for Computing the Moore-Penrose Inverse, Journal of Hyperstructures vol. 6, N1, pp. 52-67, (2017).

[33] Jebreen H. and Chalco-Cano Y., An Improved Computationally Efficient Method for Finding the Drazin Inverse, Discrete Dynamics in Nature and Society, vol. 2018, Article ID 6758302, 8 pages, (2018).

[34] Stanimirovic P., Kumar A. and Katsikis V., Further Efficient Hyperpower Iterative methods for the Computation of Generalized Inverses $A^2_{T,S}$, RACSAM, vol.113, pp. 3323-3339, (2019).

[35] Buranay S. and Iyikal O., A Predictor-Corrector Iterative Method for Solving Linear Least Squares Problems and Perturbation Error Analysis, Journal of Inequalities and Applications, vol. 203, pp.1-14, (2019).

[36] Saad Y., Iterative Methods for Sparse Linear Systems, 2-nd edition, SIAM, Philadelpha, PA, (2003).

[37] Ferronato M., Preconditioning for Sparse Linear Systems at the Dawn of the 21st Century: History, Current Developments, and Future Perspectives, ISRN Applied Mathematics, vol. 2012, Article ID 127647, 49 pages, (2012).

[38] Peng R. and Spielman D., An Efficient Parallel Solver for SDD Linear Systems, arXiv:1311.3286v1 [cs.NA], 13 Nov. (2013).

[39] Van der Vorst H. and Van Dooren P., Parallel Algorithms for Numerical Linear Algebra, Elsevier Science Ltd, Oxford, UK, 340 pages, (1990).

[40] Stotsky A., Unified Frameworks for High Order Newton-Schulz and Richardson Iterations: A Computationally Efficient Toolkit for Convergence Rate Improvement, Journal of Applied Mathematics and Computing, vol. 60, N 1 - 2, pp. 605-623, (2019).

[41] Stotsky A., Efficient Iterative Solvers in the Least Squares Method, Proc. of the 21-st IFAC World Congress, Berlin, Germany, July 12-17, 2020.

[42] Chen K., Matrix Preconditioning Techniques and Applications, Cambridge University Press, Cambridge, UK., (2005).

[43] Hackbusch W., Iterative Solution of Large Sparse Systems of Equations, Springer, New York, (1994).

[44] Horn R. and Johnson C., Matrix Analysis, Cambridge University Press, (1985).

[45] Stotsky A., Recursive Trigonometric Interpolation Algorithms, Journal of Systems and Control Engineering, vol. 224, N 1, pp. 65-77, (2010).

[46] Shadid J. and Tuminaro R., A Comparison of Preconditioned Nonsymmetric Krylov Methods on a Large MIMD Machine, SIAM J. Sci. Comput., vol. 15, N. 2, pp. 440-459, March (1994).

[47] Varga R., Matrix Iterative Analysis, Prentice Hall, Englewood Cliffs, NJ, (1962).

[48] Stotsky A., Towards Accurate Estimation of Fast Varying Frequency in Future Electricity Networks: The Transition from Model-Free Methods to Model-Based Approach, Journal of Systems and Control Engineering, vol. 230, N 10, pp. 1164-1175, (2016).

[49] Ehrmann H., Konstruktion und Durchführung von Iterationsverfahren höherer Ordnung. Arch. Ration. Mech. Anal., N. 4, pp. 65-88,(1959).

[50] Brezinski C. Variations on Richardson's Method and Acceleration, in: Numerical Analysis, A Numerical Analysis Conference in Honour of Jean Meinguet, Bull. Soc. Math. Belgium, pp. 33-44, (1996).

[51] Srivastava S. and Gupta D., A Higher Order Iterative Method for $A_{T,S}^{(2)}$, Journal of Applied Mathematics and Computing, vol.46, N 1/2, pp. 147 - 168, (2014).

[52] Richardson, L. The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam, Philosophical Transactions of the Royal Society A 210, pp. 307-357, (1910).

[53] Dubois D., Greenbaum A. and Rodrigue G., Approximating the Inverse of a Matrix for Use in Iterative Algorithms on Vector Processors, Computing, 22, pp. 257-268, (1979).

[54] Chen, Y.L., Iterative methods for solving restricted linear equations. Appl. Math. Comput. vol. 86, pp. 171-184, (1997).

[55] Stotsky A., Combined High-Order Algorithms in Robust Least-Squares Estimation with Harmonic Regressor and Strictly Diagonally Dominant Information Matrix, Proc. IMechE Part I: Journal of Systems and Control Engineering, vol. 229, N 2, pp. 184-190, (2015).

[56] Srivastava S., Stanimirovic P., Katsikis V. and Gupta D., A Family of Iterative Methods with Accelerated Convergence for Restricted Linear System of Equations, Mediterr. J. Math., vol. 14-222, pp 1-26, (2017).

[57] Traub J., Iterative Methods for Solution of Equations, Englewood Cliffs, NJ: Prentice-Hall, (1964).

[58] Table of Prime Factors, `https://en.wikipedia.org/wiki/Table_of_prime_factors#1_to_100`

9 Appendix. Factorization Tool-Kit: A Unified Approach

Some of known factorizations of the Newton-Schulz iteration are presented in the following unified framework (see the Tables 1 - 3 below):

$$Z = \{\sum_{j=0}^{h-1} Y^j\}\, X \tag{54}$$

$$Y = I - XA \tag{55}$$

$$Z = \{\sum_{j=0}^{w-1} Y^{(p+1)j}\}\, \{\sum_{d=0}^{p} Y^d\}\, X \tag{56}$$

$$h = w\,(p+1),\ \ p = 0, 1, 2, ...,\ w = 1, 2, 3, ... \tag{57}$$

$$Z_1 = (I + Y\{\sum_{j=0}^{w-1} Y^{(p+1)j}\}\, \{\sum_{d=0}^{p} Y^d\})\, X \tag{58}$$

$$h_1 = h + 1 \tag{59}$$

where equations (54) and (55) represent the Newton-Schulz iteration and equations (56), (58) represent the factorizations Z and Z_1 of orders h is h_1 respectively.
The factorization (56) is valid for the orders, which are presented as the composite numbers[2], $h = 2, 4, 6, 8, 9, 10, 12, 14, 15, 16, 18$ (excepting $h = 2$). For the orders which represent the prime numbers[3], $h_1 = 3, 5, 7, 11, 13, 17, 19$ the factorization (58) is valid. Notice that factorization for the order $h = 10$ is presented in both forms following [30], see Table 2.
Notice that the factorization defined in (56) is not unique. For example the Newton-Schulz iteration of order $h = 18$ can be factorized in the following four ways: a) $p = 5$, $w = 3$, b) $p = 2$, $w = 6$, c) $p = 8$, $w = 2$, d) $p = 1$, $w = 9$ which have different number of mmm in implementation.
Notice also that the factorization (58) is simple application of the idea known as Schröder–Traub sequence, [16], [57] to the polynomial factorized in (56). The factorization (58) increases the order of (56) by one, see (57) and (59).
Moreover, nested application of the factorizations (56) and (58) implies additional order reduction and improvement of the computational efficiency (mainly for high orders), see for example nested factorization for $h = 15$ with 7 mmm in Table 3.
Finally, the Tables 1 - 3 which end with the algorithm of the nineteenth order can be easily extended for higher orders (for any order) using nested applications of the factorizations (56), (58) and the tables of prime factors, [58], providing computationally efficient and implementable solutions for higher orders.
However, the factorizations which require computational efforts and additional memory may result in error accumulation in finite-digit calculations. The factorizations in high order Newton-Schulz iterations can be seen as the additional sub-steps (nested calculations for order reduction). The idea of order reduction is also associated with the Newton-Schulz iteration, see Section 4. Therefore Newton-Schulz algorithms of low orders ($h = 2, 3$) being iterated for a number of steps can be applied instead of factorized Newton-Schulz iterations of higher orders for the sake of robustness and efficiency.
Notice that two and three steps of the second order Newton-Schulz iteration are equivalent to one step of the fourth and eighth order iterations with 4 and 6 mmm respectively, see Table 1, $h = 4$ and Table 2, $h = 8$. Robustness, efficiency and accuracy arguments motivate application of the second order [22]- [24] and the third order [12] Newton-Schulz iterations instead of higher orders in some cases. However, application of the eleventh order algorithm, see Table 2 and [34] requires also 6 mmm and provides faster convergence than three steps of the second order Newton-Schulz iteration. Therefore the proper choice of the order and factorization that is made for each particular application should represent the trade-off between the robustness and convergence rate.

[2] A composite number is a positive integer that has at least one divisor other than one and itself or can be formed by multiplying two smaller positive integers

[3] A prime number is a positive integer that has exactly two distinct whole number factors (or divisors), namely one and the number itself

Order h	Factorization	Unified Factorization	Ref.
$h=2$	$(I+Y)\,X$	$\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d\, X$ $p=1,\, w=1$	[22]- [24]
$h=3$	$(I+Y(I+Y))\,X$	$(I+Y\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d)\, X$ $p=1,\, w=1$	[16], [19],[21]
$h=4$	$(I+Y^2)(I+Y)\,X$	$\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d\, X$ $p=1,\, w=2$	[32]
$h=5$	$(I+Y(I+Y^2)(I+Y))\,X$ $=(I+Y+Y^2+Y^2(Y+Y^2))\,X$	$(I+Y\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d)\, X$ $p=1,\, w=2$	[30]
$h=6$	$(I+Y^3)(I+Y+Y^2)\,X$	$\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d\, X$ $p=2,\, w=2$	
$h=7$	$(I+(Y+Y^4)(I+Y+Y^2))\,X$	$(I+Y\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d)\, X$ $p=2,\, w=2$	[28]

Table 1: Factorization of the algorithms of orders $h=2,...,7$

Order h	Factorization	Unified Factorization	Ref.
$h = 8$	$(I + Y^4)(I + Y^2)(I + Y)X$	$\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d X$ $p = 3, w = 2$	
$h = 9$	$(I + Y^3 + Y^6)(I + Y + Y^2)\ X$ $= (I + (I + Y^4)(I + Y^2)(Y + Y^2))\ X$	$\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d X$ $p = 2, w = 3$	
$h = 10$	$(I + Y(I + Y^3 + Y^6)$ $(I + Y + Y^2))\ X =$ $(I + Y^5)(I + (Y + Y^2)(I + Y^2))\ X$	$(I + Y \sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d)X$ $p = 2, w = 3$ $= \sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d X$ $p = 4, w = 2$	[30], [33]
$h = 11$	$(I + Y(I + Y^5)(I + (Y + Y^2)$ $(I + Y^2))\ X$ $(I + Y(I + (Y^2 + Y^4)(I + Y^4))$ $(I + Y))\ X$	$(I + Y \sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d)X$ $p = 4, w = 2$ $(I + Y \sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d)X$ $p = 1, w = 5$	[30] [34]
$h = 12$	$(I + Y^4 + Y^8)$ $(I + Y + Y^2 + Y^3)\ X$	$\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d\ X$ $p = 3, w = 3$	[30]
$h = 13$	$(I + Y(I + Y^4 + Y^8)$ $(I + Y + Y^2 + Y^3))\ X$	$(I + Y \sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d)\ X$ $p = 3, w = 3$	[30]

Table 2: Factorization of the algorithms of orders $h = 8, ..., 13$

Order h	Factorization	Unified Factorization	Ref.
$h = 14$	$(I + Y^7)$ $(I + Y + Y^2 + Y^3 + Y^4 + Y^5 + Y^6)\, X$	$\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d\, X$ $p = 6, w = 2$	[30]
$h = 15$	$(I + Y^3 + Y^6 + Y^9 + Y^{12})$ $(I + Y + Y^2)\, X$ $= (I + (I + (Y^3)^2)\, ((Y^3)^2 + Y^3))$ $(I + Y + Y^2)\, X$	$\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d\, X$ $p = 2, w = 5$	[30]
$h = 16$	$(I + Y^4 + Y^8 + Y^{12})$ $(I + Y + Y^2 + Y^3)\, X$	$\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d\, X$ $p = 3, w = 4$	[30]
$h = 17$	$(I + (Y + Y^2 + Y^3 + Y^4)$ $(I + Y^4 + Y^8 + Y^{12}))\, X$	$(I + Y \sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d)\, X$ $p = 3, w = 4$	[30]
$h = 18$	$(I + Y^6 + Y^{12})$ $(I + Y + Y^2 + Y^3 + Y^4 + Y^5)\, X$	$\sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d\, X$ $p = 5, w = 3$	[30]
$h = 19$	$(I + (Y + Y^2)(I + Y^2 + Y^4)$ $(I + Y^6 + Y^{12}))\, X$	$(I + Y \sum_{j=0}^{w-1} Y^{(p+1)j} \sum_{d=0}^{p} Y^d)\, X$ $p = 5, w = 3$	[30]

Table 3: Factorization of the algorithms of orders $h = 14, ..., 19$