Numer Algor (2019) 81:547–559
https://doi.org/10.1007/s11075-018-0560-9

ORIGINAL PAPER

Fast factorization of rectangular Vandermonde matrices with Chebyshev nodes

Mykhailo Kuian[1] · Lothar Reichel[1] · Sergij V. Shiyanovskii[2]

Received: 15 February 2018 / Accepted: 10 June 2018 / Published online: 28 June 2018

Abstract The polynomial interpolation problem with distinct interpolation points and the polynomial represented in the power basis gives rise to a linear system of equations with a Vandermonde matrix. This system can be solved efficiently by exploiting the structure of the Vandermonde matrix with the aid of the Björck–Peyrera algorithm. We are concerned with polynomial least-squares approximation at the zeros of Chebyshev polynomials. This gives rise to a rectangular Vandermonde matrix. We describe fast algorithms for the factorization of these matrices. Both QR and QR-like factorizations are discussed. The situations when the nodes are extreme points of Chebyshev polynomials or zeros of some classical orthogonal polynomial also are considered.

Keywords Vandermonde matrix · Fast factorization · Chebyshev nodes

✉ Lothar Reichel
reichel@math.kent.edu

Mykhailo Kuian
mkuian@kent.edu

Sergij V. Shiyanovskii
sshiyano@kent.edu

[1] Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA

[2] Liquid Crystal Institute, Kent State University, Kent, OH 44242, USA

1 Introduction

We consider fast QR factorization of rectangular Vandermonde matrices

$$V = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^{n-1} \end{bmatrix} \in \mathbb{R}^{N\times n}, \quad N \geq n, \tag{1.1}$$

whose nodes x_i are the zeros of a Chebyshev polynomial of the first kind of degree N for the interval $[-1, 1]$,

$$x_i = \cos\left(\frac{2i-1}{2N}\pi\right), \quad i = 1, 2, \ldots, N. \tag{1.2}$$

We also will discuss the situation when the nodes are extreme points of a Chebyshev polynomial of the first kind of degree $N-1$. These nodes are given by

$$x_i = \cos\left(\frac{i-1}{N-1}\pi\right), \quad i = 1, 2, \ldots, N. \tag{1.3}$$

Rectangular Vandermonde matrices[1] arise in the polynomial least-squares approximation problem

$$\min_{c\in\mathbb{R}^n} \sum_{i=1}^{N} (p(c, x_i) - y_i)^2, \quad p(c, x) = \sum_{j=0}^{n-1} c_j x^j, \quad c = [c_0, c_1, \ldots, c_{n-1}]^T. \tag{1.4}$$

where the vector $y = [y_1, y_2, \ldots, y_N]^T$ represents data that is to be fitted in the least-squares sense. The minimization problem (1.4) can be written in the form

$$\min_{c\in\mathbb{R}^n} \|Vc - y\|_2, \tag{1.5}$$

where $\|\cdot\|_2$ denotes the Euclidean vector norm. The use of the power form representation of the least-square polynomial p is convenient when the polynomial is to be differentiated or integrated.

Although Vandermonde matrices with real nodes are known to generally be quite ill-conditioned, the use of Chebyshev nodes (1.2) reduces this difficulty somewhat. The condition number of a Vandermonde matrix is defined as

$$\kappa(V) = \|V\|\|V^\dagger\|,$$

where $V^\dagger$ denotes the Moore–Penrose pseudoinverse of V, and $\|\cdot\|$ stands for a matrix norm. The condition number indicates how errors in the data vector y in (1.5) as well as round-off errors introduced during the solution process are propagated to the computed solution. It is desirable that the condition number not be too large. Gautschi [4] showed that for the Frobenius matrix norm, the condition number of a square Vandermonde matrix $V \in \mathbb{R}^{n\times n}$, with real nodes allocated to minimize the

[1] In the literature, the transpose of the matrix (1.1) is sometimes referred to as a rectangular Vandermonde matrix.

condition number, grows exponentially at a rate slightly less than $(1+\sqrt{2})^n$. When the matrix norm is induced by the uniform vector norm and the nodes are given by (1.2) with $N = n$, Gautschi [5] proved the condition number for square Vandermonde matrices to be asymptotically $\frac{3^{\frac{3}{4}}}{4}(1+\sqrt{2})^n$ as $n \to \infty$. This result indicates that it is beneficial to choose the nodes (1.2) for Vandermonde matrices. Moreover, it is known that the approximation of functions on the interval $[-1, 1]$ by polynomial interpolation at Chebyshev nodes gives near-optimal approximants in the sense that the polynomial interpolant is close to the best polynomial approximant of the same degree in the uniform norm; see Mason and Handscomb [10]. The nodes (1.3) also are known to be suitable interpolation points. If a polynomial approximant of a function is desired on a bounded interval $[a, b]$ different from $[-1, 1]$, then it is convenient to map this interval to $[-1, 1]$ by a linear transformation and compute a polynomial interpolant or least-squares approximant on the latter interval.

Eisinberg et al. [3] were the first to describe a fast factorization of a rectangular Vandermonde matrix (1.1) determined by Chebyshev nodes (1.2). They presented the factorization

$$V = HUD, \tag{1.6}$$

where $H = [h_{ij}] \in \mathbb{R}^{N\times n}$ has the entries

$$h_{ij} = \cos\left[\frac{(2i-1)(j-1)}{2N}\right], \quad i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, n,$$

and the upper triangular matrix $U = [u_{ij}] \in \mathbb{R}^{n\times n}$ has the elements

$$\begin{aligned}
u_{2i,2j} &= C_{2j-1}^{(j-i)}, \quad i = 1, 2, \ldots, \left\lfloor \frac{n}{2} \right\rfloor, \quad j = i, i+1, \ldots, \left\lfloor \frac{n}{2} \right\rfloor,\\
u_{2i-1,2j-1} &= C_{2j-2}^{(j-i)}, \quad i = 1, 2, \ldots, \left\lceil \frac{n}{2} \right\rceil, \quad j = i, i+1, \ldots, \left\lceil \frac{n}{2} \right\rceil,\\
u_{1,2j-1} &= C_{2j-3}^{(j-1)}, \quad j = 1, 2, \ldots, \left\lceil \frac{n}{2} \right\rceil.
\end{aligned}$$

Here $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ denote the "floor" and "ceiling" functions, respectively, and the

$$C_k^{(\ell)} = \frac{k!}{(k-\ell)!\ell!} \tag{1.7}$$

are binomial coefficients. They arise in the expansion of the powers x^k in terms of Chebyshev polynomials

$$T_i(x) = \cos(i \arccos(x)), \quad -1 \le x \le 1, \quad i = 0, 1, \ldots, k; \tag{1.8}$$

see, for instance, Cody [2], Mason and Handscomb [10, Chapter 2], or (2.5) below. The matrix $D = \mathrm{diag}[d_{11}, d_{22}, \ldots, d_{nn}] \in \mathbb{R}^{n\times n}$ has the entries

$$d_{ii} = \frac{1}{2^{i-1}}, \quad i = 1, 2, \ldots, n.$$

The representation (1.6) allows Eisinberg et al. [3] to express the Moore–Penrose pseudoinverse of V in factored form,

$$V^\dagger = \frac{1}{N} D^{-1} Q B H^T, \tag{1.9}$$

where the matrix $Q = [q_{ij}] \in \mathbb{R}^{n\times n}$ is the inverse of U. It has the entries

$$q_{2i,2j} = (-1)^{i+j}\frac{2j-1}{2i-1}C_{i+j-2}^{(j-i)}, \quad i=1,2,\ldots,\left\lfloor\frac{n}{2}\right\rfloor, \quad j=i,i+1,\ldots,\left\lfloor\frac{n}{2}\right\rfloor,$$

$$q_{2i-1,2j-1} = (-1)^{i+j}\frac{j-1}{i-1}C_{2j-2}^{(j-i)}, \quad i=1,2,\ldots,\left\lceil\frac{n}{2}\right\rceil, \quad j=i,i+1,\ldots,\left\lceil\frac{n}{2}\right\rceil,$$

$$q_{1,2j-1} = (-1)^{j+1}, \quad j=1,2,\ldots,\left\lceil\frac{n}{2}\right\rceil.$$

Moreover,

$$B = \mathrm{diag}[1,2,2,\ldots,2] \in \mathbb{R}^{n\times n}.$$

Eisinberg et al. [3] compute the solution of (1.5) by evaluating

$$c = V^{\dagger}y,$$

using the factorization (1.9). The computation of the vector c in this manner requires $5Nn + 2n^2$ arithmetic floating point operations (flops), assuming that the required binomial coefficients (1.7) are explicitly known. The flop count does not include the evaluation of the entries h_{ij} of the matrix H. Here and throughout this paper, our flop counts only give the leading terms. In computations reported in Section 4, we precompute the coefficients (1.7) that will be used and store them in a file. We will refer to this fast solver due to Eisinberg et al. [3] as the *EFS factorization algorithm*.

It is the purpose of this paper to discuss alternative fast algorithms for the factorization of rectangular Vandermonde matrices and the solution of (1.5). These algorithms exploit the structure of the matrix (1.1) in a different manner than Eisinberg et al. [3]. We derive, in Section 2, a QR factorization

$$V = QR, \tag{1.10}$$

where the Vandermonde matrix (1.1) is determined by the nodes (1.2), the matrix $Q \in \mathbb{R}^{N\times n}$ has orthonormal columns, and the matrix $R \in \mathbb{R}^{n\times n}$ is upper triangular. The entries of the matrices Q and R are explicitly known, which makes fast solution of the least-squares problem (1.5) possible.

Another fast solution method for (1.5) is obtained by applying the formula

$$V\tilde{R} = Q. \tag{1.11}$$

The matrices V and Q in (1.10) and (1.11) are the same, and the matrix $\tilde{R} \in \mathbb{R}^{n\times n}$ in (1.11) is the inverse of R in (1.10). Thus, $\tilde{R}$ is upper triangular. Its entries are explicitly known. It follows that (1.11) can be applied to derive a fast solution method for (1.5). The formula (1.11) was derived by Li [9], who applied it to investigate the conditioning of rectangular Vandermonde matrices with Chebyshev nodes; Li did not explore the application of the decomposition (1.11) in a fast solver for (1.5). Such a solver is described in Section 2.

Formulas analogous to (1.10) and (1.11) for the situation when the nodes are extreme points of a Chebyshev polynomial (1.3) also can be derived. The columns of the matrix Q in the formulas then are orthonormal with respect to a weighted discrete inner product and associated norm. These formulas are discussed in Section 3, where we also consider the situation when the nodes $x_1, x_2, \ldots, x_N$ are zeros of a classical orthogonal polynomial. Computed examples that compare the accuracy of

the computed solutions and the execution time required of the fast solvers outlined above and of a structure-ignoring QR factorization method are presented in Section 4. Concluding remarks can be found in Section 5.

2 Fast factorization methods for Vandermonde matrices defined by zeros of Chebyshev polynomials

This section discusses fast solvers for (1.5) based on the formulas (1.10) and (1.11) for Vandermonde matrices with the nodes (1.2). Introduce the normalized Chebyshev polynomials

$$\tilde{T}_k(x) = \begin{cases} \sqrt{\frac{1}{\pi}} T_k(x), & \text{if } k = 0, \\ \sqrt{\frac{2}{\pi}} T_k(x), & \text{if } k > 0. \end{cases} \tag{2.1}$$

Lemma 2.1 *The matrix*

$$Q = \sqrt{\frac{\pi}{N}} \begin{bmatrix} \tilde{T}_0(x_1) & \cdots & \tilde{T}_{n-1}(x_1) \\ \tilde{T}_0(x_2) & \cdots & \tilde{T}_{n-1}(x_2) \\ \vdots & \cdots & \vdots \\ \tilde{T}_0(x_N) & \cdots & \tilde{T}_{n-1}(x_N) \end{bmatrix} \in \mathbb{R}^{N \times n} \tag{2.2}$$

has orthonormal columns for any $1 \le n \le N$.

Proof The orthonormality of the columns follows from the property that for $0 \le j, k < N$,

$$\frac{\pi}{N} \sum_{i=1}^{N} \tilde{T}_j(x_i) \tilde{T}_k(x_i) = \begin{cases} 1, & j = k, \\ 0, & j \ne k, \end{cases}$$

which is a consequence of [10, eq. (6.12)]. □

Proposition 2.2 *The matrix (1.1) has a QR factorization (1.10) with* $Q \in \mathbb{R}^{N \times n}$ *given by (2.2) and the entries of the upper triangular matrix* $R = [r_{ij}] \in \mathbb{R}^{n \times n}$ *defined by*

$$r_{1j} = \begin{cases} 2^{1-j} \sqrt{N} C_{j-1}^{\left(\frac{j-i}{2}\right)}, & \text{if } j \text{ is odd}, \\ 0, & \text{if } j \text{ is even}, \end{cases} \tag{2.3}$$

and

$$r_{ij} = \begin{cases} 2^{2-j} \sqrt{\frac{N}{2}} C_{j-1}^{\left(\frac{j-i}{2}\right)}, & \text{if } j - i \text{ is even}, \\ 0, & \text{if } j - i \text{ is odd or } j < i. \end{cases} \tag{2.4}$$

Proof Results by Cody [2] or Mason and Handscomb [10, Chapter 2] can be used to show

$$x^j = 2^{1-j}\sqrt{\frac{\pi}{2}} \sum_{\substack{i=0 \\ j-i \text{ even}}}^{j}{}' \, C_j^{\left(\frac{j-i}{2}\right)} \tilde{T}_i(x), \tag{2.5}$$

where the ' indicates that the first term (for $i = 0$ and j even) is to be multiplied by $1/\sqrt{2}$. It follows from (1.10) that every element of the matrix V can be expressed as

$$x_k^j = \sum_{i=1}^{j+1} q_{ki} r_{i,j+1},$$

where the q_{ki} are the entries of the matrix (2.2). Using (2.2) and (2.5) gives (2.3) and (2.4). □

The QR factorization (1.10) can be used to solve the minimization problem (1.5) by first evaluating the vector $Q^T y$ and then solving

$$Rc = Q^T y$$

by back substitution. These computations are backward stable; see [12]. The flop count for this solution method is $2Nn + n^2$ when $n \ll N$: the evaluation of $Q^T y$ requires $2Nn$ flops, and back substitution costs $n^2/2$ flops. The latter count utilizes the zero structure of the matrix R. When $n \approx N$, the vector $Q^T y$ can be evaluated more rapidly (in $\mathcal{O}(N \log N)$ flops) by application of the fast Fourier transform; see, e.g., [10, Chapter 4]. We are primarily interested in the situation when $n \ll N$.

Our MATLAB implementation of this solution method for (1.5) uses the standard MATLAB function `c = linsolve(R,Q'*y,opts)` with `opts.UT = true` for back substitution. This function does not exploit the zero structure of R. Back substitution therefore requires n^2 flops in our implementation. We refer to this solution method as the *explicit QR factorization algorithm*.

We turn to a solution method based on the formula (1.11). The entries of the upper triangular matrix

$$\tilde{R} = \begin{bmatrix} \tilde{r}_{1,1} & \tilde{r}_{1,2} & \tilde{r}_{1,3} & \cdots & \tilde{r}_{1,n} \\ & \tilde{r}_{2,2} & \tilde{r}_{2,3} & \cdots & \tilde{r}_{2,n} \\ & & \tilde{r}_{3,3} & \cdots & \tilde{r}_{3,n} \\ & & & \ddots & \vdots \\ & & & & \tilde{r}_{n,n} \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{2.6}$$

can be determined from the formula

$$T_i(x) = \sum_{k=0}^{\lfloor \frac{i}{2} \rfloor} (-1)^k \frac{i}{i-k} C_{i-k}^{(k)} 2^{i-2k-1} x^i;$$

see [10, Chapter 2] for a proof. We obtain $\tilde{r}_{1,1} = 1/\sqrt{N}$ and

$$\tilde{r}_{j,j-2i} = (-1)^i \sqrt{\frac{2}{N}} \frac{j-1}{j-i-1} C_{j-i-1}^{(i)} 2^{j-2i-2}, \tilde{r}_{j,j-2i-1} = 0$$

for $j = 2, 3, \ldots, n$ and $i = 0, 1 \ldots, \lfloor j/2 \rfloor$; see also Li [9].

It follows from (1.11) that $V^\dagger = \tilde{R}Q^T$. The solution of (1.5) can be computed by matrix-vector product evaluations with the matrices Q^T and $\tilde{R}$:

$$c = \tilde{R}(Q^T y). \tag{2.7}$$

Straightforward evaluation of (2.7) using the MATLAB command `R̃*(Q'*y)` requires $2Nn+n^2$ flops, assuming that the entries of the matrices $\tilde{R}$ and Q are known. Note that the matrix-vector product evaluation with $\tilde{R}$ does not use the zero-structure of $\tilde{R}$. We will refer to the expression (1.11) as a QR-like factorization of V, and to the solution method based on the computations (2.7) as the *explicit QR-like factorization algorithm*. The computations (2.7) are backward stable.

We will compare the fast algorithms discussed to the structure-ignoring QR factorization method that is implemented by the MATLAB function `[Q,R]=qr(V)`. This function computes a QR factorization of the matrix V by Householder triangularization. The computation of this factorization requires $2Nn^2 - \frac{2}{3}n^3$ flops; see, e.g., [12] for details. Evaluation of the solution vector c demands $2Nn + n^2$ additional flops. The total flop count therefore is $2N(n^2+n) - \frac{2}{3}n^3 + n^2$. We refer to this solution method as the *standard QR factorization algorithm*.

3 Fast factorization methods for Vandermonde matrices defined by extrema of Chebyshev polynomials and extensions

In this section, the nodes x_k are the extreme points (1.3) of the Chebyshev polynomial T_{N-1}.

Lemma 3.1 *Define the matrices*

$$Q = \sqrt{\frac{\pi}{N-1}} \begin{bmatrix} \tilde{T}_0(x_1) & \cdots & \tilde{T}_{n-1}(x_1) \\ \tilde{T}_0(x_2) & \cdots & \tilde{T}_{n-1}(x_2) \\ \vdots & \ldots & \vdots \\ \tilde{T}_0(x_N) & \cdots & \tilde{T}_{n-1}(x_N) \end{bmatrix} \in \mathbb{R}^{N\times n} \tag{3.1}$$

and

$$E = \operatorname{diag}\left[1/\sqrt{2}, 1, 1, \ldots, 1, 1/\sqrt{2}\right] \in \mathbb{R}^{N\times N}. \tag{3.2}$$

Introduce the inner product $(u,v)_E = u^T E^2 v$ and associated norm $\|v\|_E = (v,v)_E^{1/2}$ for $u, v \in \mathbb{R}^N$. Then the matrix Q has orthonormal columns with respect to this inner product and norm.

Proof Introduce the discrete inner product

$$[f,g] = \frac{1}{2}f(x_1)g(x_1) + \sum_{i=2}^{N-1} f(x_i)g(x_i) + \frac{1}{2}f(x_N)g(x_N)$$

for polynomials f and g of degree at most $N-1$. It is well-known that the Chebyshev polynomials (1.8) satisfy the orthogonality relations, for $0 \leq j, k < N$,

$$[T_j, T_k] = \begin{cases} 0, & j \neq k, \\ \frac{N-1}{2}, & j = k \neq 0, \\ N-1, & j = k = 0; \end{cases} \tag{3.3}$$

see, e.g., [10, p. 87] for a proof. Rescaling according to (2.1) shows the lemma. □

The following result can be shown similarly as Proposition 2.2. We refer to a factorization $V = QR \in \mathbb{R}^{N\times n}$ as a QR-type factorization if the columns of $Q \in \mathbb{R}^{N\times n}$ are orthonormal with respect to a weighted inner product and associated norm, and $R \in \mathbb{R}^{n\times n}$ is upper triangular.

Proposition 3.2 *Let the nodes of the Vandermonde matrix (1.1) be Chebyshev extreme points (1.3) and let E be defined by (3.2). Then the matrix V has a QR-type factorization $V = QR$, where Q, defined by (3.1), has orthonormal columns with respect to the inner product $(\cdot,\cdot)_E$ and associated norm, and the entries of the upper triangular matrix $R = [r_{ij}] \in \mathbb{R}^{n\times n}$ are given by*

$$r_{1j} = \begin{cases} 2^{1-j}\sqrt{N-1}\,C_{j-1}^{\left(\frac{j-i}{2}\right)}, & \text{if } j \text{ is odd}, \\ 0, & \text{if } j \text{ is even}, \end{cases}$$

and

$$r_{ij} = \begin{cases} 2^{2-j}\sqrt{\frac{N-1}{2}}\,C_{j-1}^{\left(\frac{j-i}{2}\right)}, & \text{if } j-i \text{ is even}, \\ 0, & \text{if } j-i \text{ is odd or } j < i. \end{cases}$$

The matrices of Proposition 3.2 can be applied to solve the weighted least-squares problem

$$\min_{c\in\mathbb{R}^n} \|Vc - y\|_E. \tag{3.4}$$

Its solution can be computed as

$$Rc = Q^T E^2 y.$$

The vector c is evaluated by back substitution. A QR-like factorization, analogous to (1.11), also can be derived. We omit the details.

Let $d\mu$ be a nonnegative measure with support on the real axis and let $p_0, p_1, p_2, \ldots$ denote a family of orthonormal polynomials associated with this measure. One can derive a QR-type factorization of a Vandermonde matrix $V = QR \in \mathbb{R}^{N\times n}$, whose nodes $x_1, x_2, \ldots, x_N$ are the zeros of p_N, by using the orthogonality of the polynomials $p_0, p_1, \ldots, p_{n-1}$ with respect to a discrete inner product defined by the N-point Gauss quadrature rule associated with $d\mu$. Thus, the columns of $Q \in \mathbb{R}^{N\times n}$ are orthonormal with respect to a weighted inner product and associated norm determined by the Gauss quadrature rule; the matrix $R \in \mathbb{R}^{n\times n}$ is upper triangular. The computation of this factorization requires the evaluation of the nodes and weights of this Gauss rule. This can be carried out rapidly in several ways when the recursion coefficients of the polynomials p_j are known or easily computable; see

[1, 6–8]. One obtains a factorization $V = QR$, where $R \in \mathbb{R}^{n\times n}$ is upper triangular, and the columns of $Q \in \mathbb{R}^{N\times n}$ are orthonormal with respect to a discrete inner product and associated norm determined by the N-point Gauss rule. The matrix R is analogous to the upper triangular matrix of Proposition 3.2. Its entries are explicitly known for many families of classical orthogonal polynomials; see, e.g., Szegő [11].

4 Numerical experiments

We report numerical experiments that shed light on the performance of the EFS, explicit QR, and explicit QR-like factorization algorithms for Vandermonde matrices $V \in \mathbb{R}^{N\times n}$ given by (1.1) with the nodes (1.2). For each pair (n, N) with $N \geq n$, we generate 10000 vectors y with uniformly distributed entries in the interval $[-1, 1]$, and compute for each one of these vectors the solution c of the least-squares problem (1.5) so defined by using the fast algorithms as well as the "slow" standard structure-ignoring QR factorization algorithm based on the use of Householder matrices. The exact solution c^* is calculated with Mathematica using high-precision arithmetic; these computations are carried out with 50 significant decimal digits. All other computations are carried out in MATLAB with about 15 significant decimal digits on a Dell computer with an i7-4770 processor running at 3.44 GHz.

Figure 1 displays the CPU time required by the algorithms as a function of N for $n = 20$. The graphs show the total time required by each method for 10000

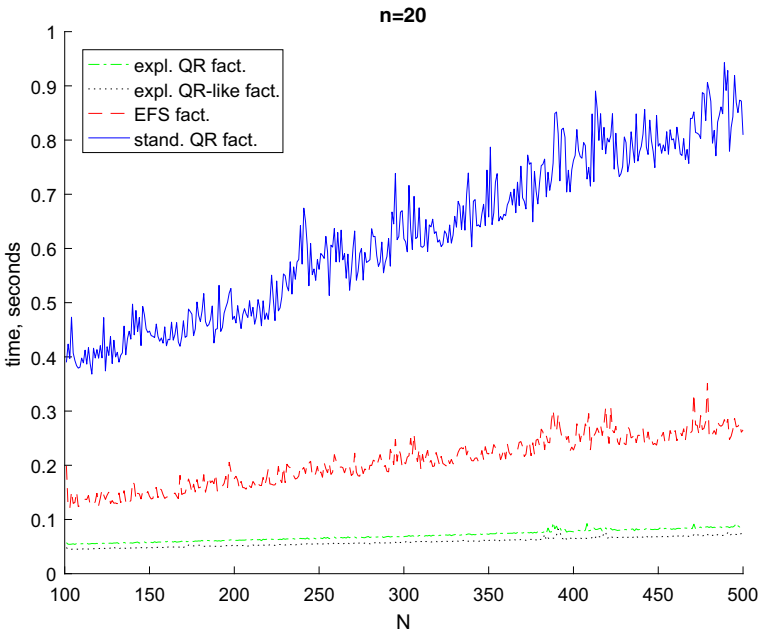

Fig. 1 Computing time as a function of N for $n = 20$ for the explicit QR, explicit QR-like, EFS, and standard QR factorization algorithms for Vandermonde matrices with Chebyshev nodes

experiments. The CPU time is seen to grow linearly with N for all algorithms in agreement with the flop counts reported in Sections 1 and 2. The CPU time for the explicit QR and QR-like algorithms grows the slowest with N.

Figure 2 shows how the CPU time depends on the parameter n for $N = 500$. The times reported are for 10000 experiments. The CPU time required by the explicit QR, explicit QR-like, and EFS factorization algorithms is seen to grow roughly linearly with n, while the CPU time for the standard QR factorization algorithm grows quadratically with n, in agreement with the flop counts for these methods. Figures 1 and 2 show the structure exploiting explicit QR and explicit QR-like factorization algorithms of Section 2 to determine the solutions the fastest.

Figure 3 displays the mean relative error $\|c - c^*\|_2/\|c^*\|_2$ over 10000 samples for each pair (n, N) for $2 \leq n \leq 40$ and $N = 100$ for the four algorithms in our comparison. The figure shows all the algorithms to give roughly the same accuracy for all n-values; for small n-values the slow structure-ignoring QR factorization algorithm gives slightly higher accuracy than the fast algorithms, but for n close to 40 the fast algorithms yield more accurate solutions. The accuracy achieved with the algorithms of Section 2 is almost indistinguishable. Figure 4 differs from Fig. 3 only in that $N = 1000$. The relative performance of the methods compared is quite similar in Figs. 3 and 4. The average errors in the computed solutions can be seen to grow exponentially with n for fixed N. Comparing Figs. 3 and 4 shows that the error does not grow significantly with N for fixed n. This behavior is in agreement with the result by Eisinberg et al. [3] that the spectral condition number of a rectangular

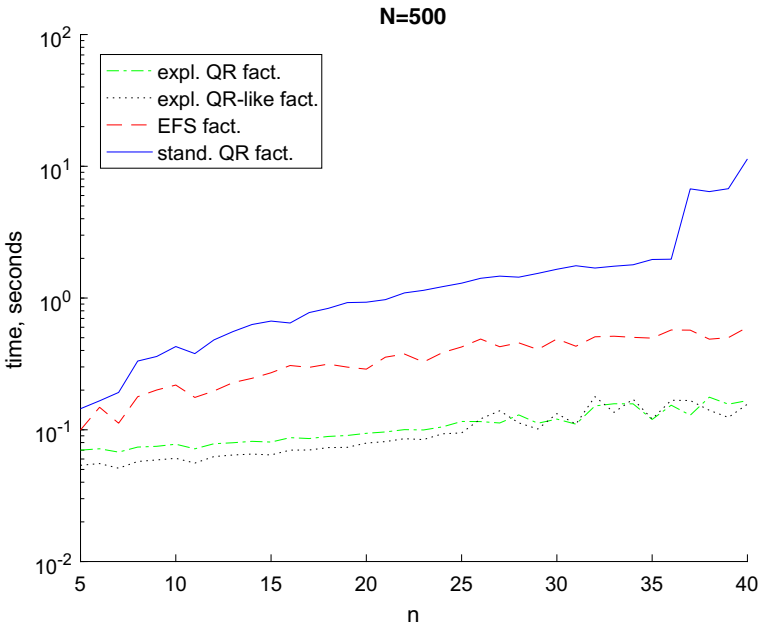

Fig. 2 Computing time as a function of n for $N = 500$ for the explicit QR, explicit QR-like, EFS, and standard QR factorization algorithms for Vandermonde matrices with Chebyshev nodes

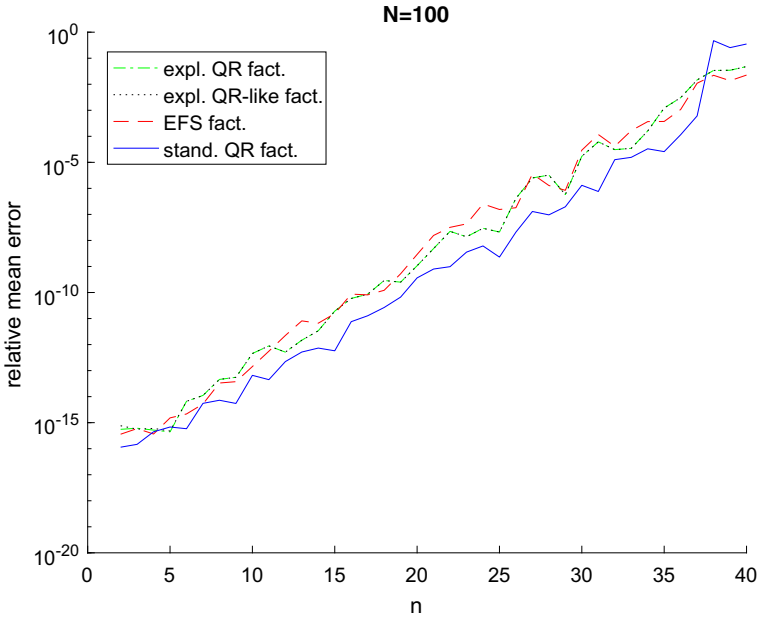

Fig. 3 Mean error as a function of n for $N = 100$ for the explicit QR, explicit QR-like, EFS, and standard QR factorization algorithms for Vandermonde matrices with Chebyshev nodes

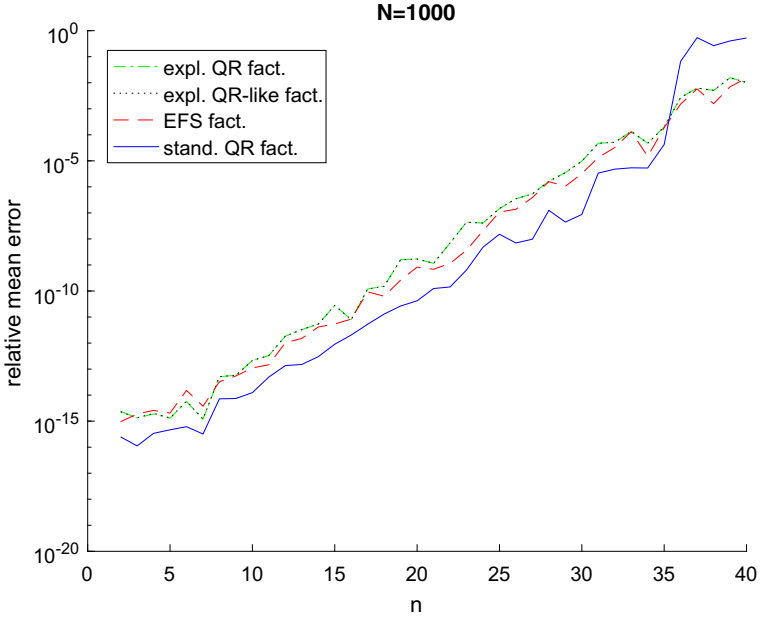

Fig. 4 Mean error as a function of n for $N = 1000$ for the explicit QR, explicit QR-like, EFS, and standard QR factorization algorithms for Vandermonde matrices with Chebyshev nodes

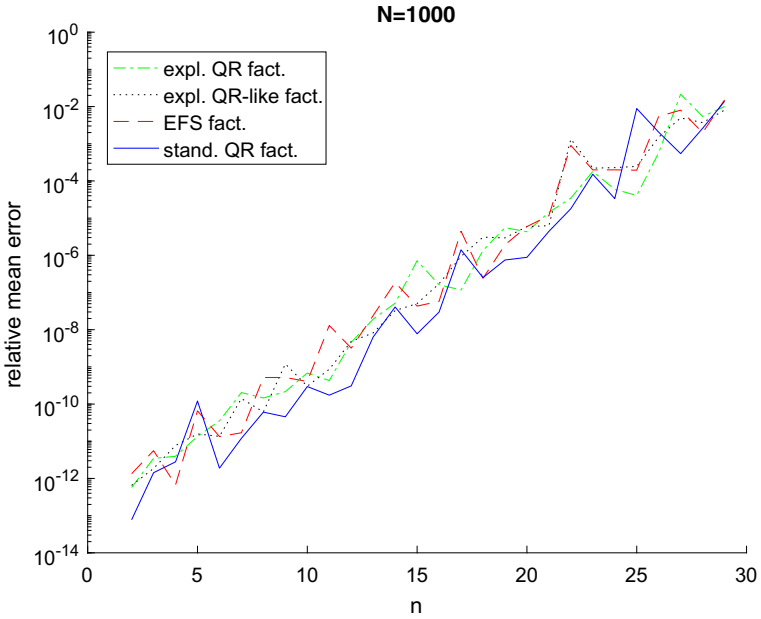

Fig. 5 Maximum component-wise relative error as a function of n for $N = 1000$ for the explicit QR, explicit QR-like, EFS, and standard QR factorization algorithms for Vandermonde matrices with Chebyshev nodes

Vandermonde matrix V with Chebyshev nodes is independent of the number of nodes $N \geq n$.

Fitting a straight line to logarithmically scaled data in the least-square sense, we find the error growth to be proportional to $(2.4)^n$. This holds for the data of both Figs. 3 and 4. This growth rate is close to the growth of the condition number of Vandermonde matrices with Chebyshev nodes; see Section 1.

Figure 5 shows the maximum component-wise relative error. Each point of each graph is the maximum relative error over 10000 least-squares problems (1.5) with data-vectors $y \in \mathbb{R}^N$ with uniformly distributed entries in $[-1, 1]$. This error is seen to be larger than the mean relative error shown in Fig. 4, but just like in the latter figure the errors in the computed solutions of the algorithms compared are of about the same size.

5 Conclusion

We presented new fast algorithms, referred to as explicit QR and QR-like factorization algorithms, for the factorization of rectangular Vandermonde matrices with Chebyshev nodes or Chebyshev extreme points. Such matrices arise in polynomial least-squares approximation problems and yield the polynomial in a particularly simple form for further processing, including differentiation and integration. The new algorithms are found to be faster and about as accurate as an available fast algorithm.

The computed examples show the accuracy of the fast algorithms to be about the same as that of the much slower standard QR factorization algorithm based on the use of Householder matrices.

Funding information This research was supported in part by NSF grants DMS-1729509 and DMS-1720259.

References

1. Bogaert, I.: Iteration-free computation of Gauss-Legendre quadrature nodes and weights. SIAM J. Sci. Comput. **36**, A1008–A1026 (2014)
2. Cody, W.J.: A survey of practical rational and polynomial approximation of functions. SIAM Rev. **12**, 400–423 (1970)
3. Eisinberg, A., Franzé, G., Salerno, N.: Rectangular Vandermonde matrices on Chebyshev nodes. Linear Algebra Appl. **338**, 27–36 (2001)
4. Gautschi, W.: Optimally scaled and optimally conditioned Vandermonde and Vandermonde-like matrices. BIT Numer. Math. **51**, 103–125 (2011)
5. Gautschi, W.: Norm estimates for inverses of Vandermonde matrices. Numer. Math. **23**, 337–347 (1975)
6. Gautschi, W.: The interplay between classical analysis and (numerical) linear algebra - a tribute to Gene H. Golub. Electron. Trans. Numer. Anal. **13**, 119–147 (2002)
7. Golub, G.H., Welsch, J.H.: Calculation of Gauss quadrature rules. Math. Comp. **23**, 221–230 (1969)
8. Laurie, D.P.: Computation of Gauss-type quadrature formulas. J. Comput. Appl. Math. **127**, 201–217 (2001)
9. Li, R.C.: Vandermonde matrices with Chebyshev nodes. Linear Algebra Appl. **428**, 1803–1832 (2008)
10. Mason, J.C., Handscomb, D.C.: Chebyshev Polynomials. CRC Press, Boca Raton (2003)
11. Szegő, G. Orthogonal Polynomials, 4th ed. American Mathematical Society, Providence (1975)
12. Trefethen, L.N., Bau, D.: Numerical Linear Algebra. SIAM, Philadelphia (1997)