Approximating the approximant: a numerical code for polynomial compression of discrete integral operators

Stefano De Marchi

Dipartimento di Matematica e Informatica
University of Udine, Italy
e-mail: demarchi@dimi.uniud.it

Marco Vianello

Dipartimento di Matematica Pura e Applicata
University of Padova, Italy
e-mail: marcov@math.unipd.it

Abstract

The action of various one-dimensional integral operators, discretized by a suitable quadrature method, can be compressed and accelerated by means of Chebyshev series approximation. Our approach has a different conception with respect to other well-known fast methods: its effectiveness rests on the "smoothing effect" of integration, and it works in linear as well as nonlinear instances, with both smooth and nonsmooth kernels. We describe a `Matlab toolbox` which implements Chebyshev-like compression of discrete integral operators, and we present several numerical tests, where the basic $\mathcal{O}(n^2)$ complexity is shown to be reduced to $\mathcal{O}(mn)$, with $m \ll n$.

Keywords: Linear and nonlinear discrete integral operators, Chebyshev series expansion, compression, fast evaluation.

2000 AMS subject classification: 65D05, 65D10, 65R10, 65R20.

1 Polynomial compression of discrete integral operators

The efficient evaluation of discrete integral operators like

$$(A_n\mathbf{u})_i = \sum_{j=1}^{n} w_j\, K(x_i,t_j,u_j) \approx Au(x_i) = \int_\Omega K(x_i,t,u(t))\, dt\ ,\ \ 1 \le i \le q\ , \tag{1}$$

where $\Omega \subset \mathbb{R}^d$, is a key point in the numerical simulation of linear and nonlinear integral models of applied sciences. In (1), $\mathbf{w} = \{w_j\}$ and $\mathbf{t} = \{t_j\}$ are the n-dimensional vectors of quadrature weights and nodes, $\mathbf{u}$ the n-dimensional vector of the (possibly approximated) values of $u(t)$ at the quadrature nodes, $u_j \approx u(t_j)$, and $\mathbf{x} = \{x_i\}$ a q-dimensional vector of "target" points in Ω. Indeed, due to the intrisincally *dense structure* of such operators, the cost of direct computation of (1) is of $\mathcal{O}(nq)$ flops, which becomes prohibitive when n and q are "large". Within the framework of integral equations, for example, we have typically that $q = n$, so we face a *quadratic complexity* at each step of any iterative solver of the underlying system.

The case of linear operators, $K(x,t,u) = H(x,t)u$, has received most of the attention till now, starting from the basic work of Rohklin and Greengard, who in the 80's developed the celebrated *fast multipole method* [10, 26], which represented the "turning point" in the simulation of *large scale* integral models. In the last fifteen years, a number of *fast algorithms* has then been conceived, all sharing the task of accelerating the evaluation of dense discrete linear operators, i.e. of dense matrix-vector products: with no pretence of exhaustivity we may quote e.g. [1, 6, 7, 8, 9, 10, 11, 12, 15, 16, 17, 18, 31] with references therein.

Roughly speaking, all these methods are able to approximate the action of the (integral) operator, at a cost of $\mathcal{O}(n\log^p n)$, $p = 1, 2$, or even of $\mathcal{O}(n)$ flops (where n is the dimension of the discretization), to be compared with the $\mathcal{O}(n^2)$ cost of direct application. We may also group these methods into FFT based ones (e.g. when convolution operators on regular grids are involved), that is "algebra-based" fast algorithms, and "analysis-based" fast algorithms, which rest on *approximation theory* techniques (e.g. on suitable expansions of the kernel), achieving impressive speedups in computing the target vector at a specified precision.

In the recent paper [30], a first exploration of a quite simple idea has been proposed, still in the frame of "analysis-based" fast algorithms, but in the more general setting of *nonlinear operators*. This idea is based on the following almost trivial observation: computing the discrete integral transform $A_n\mathbf{u}$ in (1) is ultimately nothing else than evaluating the scalar function

$$\Phi_n(x) := \sum_{j=1}^{n} w_j\, K(x,t_j,u_j) \approx \Phi(x) := \int_\Omega K(x,t,u(t))\, dt\ ,\quad x \in \Omega\ , \qquad (2)$$

at the target points $\{x_i\}$, $1 \le i \le q$. We recall that in the discretization of second-kind integral equations the function Φ_n in (2) is termed the *Nyström interpolant* [2, 3, 14]. Irrespectively of the fact that the kernel K be linear or nonlinear in the argument u, we may try to *approximate globally* Φ_n in Ω with a function, say $S_{m,n}(x)$, whose construction and evaluation at the target points costs $\mathcal{O}(m(n+q))$ flops, with $m \ll n, q$.

In this paper we focus on the one-dimensional bounded case, i.e.

$$\Omega = [a,b]\ ,\quad -\infty < a < b < +\infty\ , \qquad (3)$$

and on global *polynomial approximation*. Observe that we are approximating the (discrete) integral transform not acting on the kernel, as it is usual in linear instances, but "after" integration. This allows to exploit the *smoothing effect* of integration, a feature which has been often overlooked concerning the construction of fast evaluation algorithms (at least outside the specific frame of integral equations solving). Indeed, the number n of quadrature nodes might be large, especially in nonsmooth instances of $K(x,\cdot,u(\cdot))$, unless special quadrature formulas are used (when practical). On the other hand, due to the smoothing effect, we can have $m \ll n$ when the approximating function $S_{m,n}(x) \approx \Phi_n(x)$ is *suitably* chosen. The second remarkable feature of the present approach consists in the *natural treatment of nonlinearities*. In fact, the approximation works on the *action* of the operator, and thus does not distinguish between linearity and nonlinearity. It is worth observing that in the linear case with globally smooth kernels, rather general polynomial approximations can be applied also within $\mathcal{H}$-matrix theory [17], generating global low rank approximations of the kernel matrix. Working on the action of the operator rather than on the kernel, however, is also convenient since one deals with a univariate rather than with a bivariate approximation problem.

In [30], Chebyshev-like polynomial approximation of $\phi_n(x)$ in (2) was succesfully tried, namely interpolation at Chebyshev-Lobatto nodes or alternatively truncated Chebyshev series, in view of their well-known "quasi-optimality" [24, 25]. Chebyshev approximation is a popular tool and is commonly used also in the framework of integral equations and transforms, see [20, 23] for two recent instances, but the present approach appears substantially unexplored. Differently from [30], here we chose to implement only the Chebyshev series approximation, since it does not present the stability drawbacks typical of interpolation at increasing degree.

In practice, we obtain a strong *compression* of the action of the discrete integral operator $A_n\mathbf{u}$, by approximating the function $\Phi_n(x)$ through its m-degree truncated Chebyshev series expansion in $[a,b]$,

$$\Phi_n(x) \quad \approx \quad S_{m,n}(x) = \frac{c_0}{2} + \sum_{s=1}^{m} c_s\, T_s\left(\frac{2x-a-b}{b-a}\right)\,, \tag{4}$$

where the T_s are the usual first-kind Chebyshev polynomials, and the coefficients c_s, are obtained as the discrete scalar products (Gauss-Lobatto quadrature)

$$c_s = \frac{2}{m\,\omega_s} \sum_{k=0}^{m} \frac{1}{\omega_k}\, \Phi_n(\xi_k^m)\, \cos\left(s\,\frac{\pi k}{m}\right)\,, \quad s = 0, ..., m\,, \tag{5}$$

where $\{\xi_k^m\}$ are the *Chebyshev-Lobatto* nodes in $[a,b]$,

$$\xi_k^m = \frac{2\cos\left(\pi k/m\right) - a - b}{b-a}\,, \quad k = 0, ..., m\,, \tag{6}$$

$\omega_0 = \omega_m = 2$ and $\omega_k = 1$ for $k = 2, ..., m-1$. Notice that the construction of the approximating polynomial $S_{m,n}(x)$ has a $\mathcal{O}(mn)$ cost, due to the computation of $\{\Phi_n(\xi_k^m)\}$, $k = 0, ..., m$, and that the evaluation of such a polynomial at the target points, i.e. of $S_{m,n}(x_i) \approx \Phi_n(x_i)$, $1 \le i \le q$, costs $\mathcal{O}(mq)$ flops.

From the point of view of convergence of the approximation $S_{m,n}$ as m increases, we may distinguish two different situations.

- (i) The kernel $K(x,t,u)$ is *smooth* in $x \in [a,b]$ for fixed $t \in [a,b]$ and $u \in D \subseteq \mathbb{R}$. In this case the function $\Phi_n(x)$ is smooth itself, with (at least) the same degree of regularity, and $S_{m,n} \to \Phi_n$ as $m \to \infty$.
- (ii) The kernel $K(x,t,u)$ is *nonsmooth* in x, as it happens, e.g., in the linear case $K(x,t,u) = H(x,t)u$ with weakly-singular linear kernels $H(x,t)$, or even with Green kernels. In this case $\Phi_n(x)$ can be much less smooth than $\Phi(x)$ in (2), and it happens that the *error* $\|S_{m,n} - \Phi_n\|$ exhibits a *stalling* at increasing m, correspondingly to the size of the

underlying discretization error $\|\Phi_n - \Phi\|$ (this latter being generated by the quadrature and the possible approximations $u_j \approx u(t_j)$, cf. (2)). This occurrence has been well documented in [30] through several numerical experiments.

The error stalling phenomenon quoted in (*ii*) calls for some explanation. First, observe that we can view the approximating polynomial $S_{m,n}(x)$ as a perturbation of $S_m(x)$, the m-degree truncated Chebyshev expansion of the "exact" result of the underlying integral operator, i.e. of the function $\Phi(x)$ in (2). Indeed, (5) can be considered as the evaluation of the coefficients of such an expansion, where $\Phi(x)$ is approximated by its discretization $\Phi_n(x)$. Thus, when $K(\cdot, t, u)$ is nonsmooth and Φ_n is less smooth than Φ, it can happen that $S_{m,n}$ does not really converge (or converges extremely slowly) to Φ_n, while S_m converges to Φ. In this case writing the following error splitting

$$S_{m,n} - \Phi_n = (S_{m,n} - S_m) + (\Phi - \Phi_n) + (S_m - \Phi) \;, \tag{7}$$

we have that the third summand on the r.h.s. is infinitesimal as $m \to \infty$, while as for the first one we have

$$\|S_{m,n} - S_m\| \leq \Lambda_m \|\Phi_n - \Phi\| \;, \tag{8}$$

where the Lebesgue constant Λ_m increases logarithmically with m, by well-known results [24, §2.4]. In (8), $\|\cdot\|$ can be either the functional max-norm when Φ_n is at least continuous, or the discrete q-dimensional max-norm corresponding to the evaluation of all the functions at the target nodes.

The considerations above give a qualitative explanation of the fact that in nonsmooth instances, for fixed n, starting from a certain m the error stalls just on the size of the underlying discretization error, namely $\|\Phi - \Phi_n\|$, which begins to dominate the whole approximation process. We stress that the stalling phenomenon may be considered an additional advantage of the method, first because one is not usually interested to go below the underlying integral discretization error, and secondly because one has at hand an independent tool to estimate such an error. Notice that the estimate can be obtained at a fixed level n of discretization, without the need to compare two different levels.

In the next Section, we describe the algorithm with some implementation details, in particular concerning the detection of a possible error stalling. In Section 3, we present a number of numerical examples, which involve both linear and nonlinear kernels, with different degree of smoothness. Such examples confirm the effectiveness of Chebyshev-like compression of discrete integral operators. Finally, we discuss a few possible developments, like the extension of the present polynomial compression approach to integral operators on unbounded intervals, and to the multidimensional case (Fekete-like approximation).

2 Chebyshev-like compression: algorithm description and implementation details

In this Section we describe in detail the algorithm for Chebyshev-like compression of discrete integral operators and we provide some connections with the corresponding `Matlab toolbox`. With reference to the notation in the previous Section, fast evaluation of the discrete integral operator $A_n\mathbf{u} = \{\Phi_n(x_i)\},\ i = 1, ..., q$ (cf. (1)-(2)), by means of the polynomial approximation (truncated Chebyshev series) $S_{m,n}(x)$ of degree m (cf. (4)), requires as *inputs*:

- a, b: the interval end-points;
- n: the number of quadrature nodes;
- $\mathbf{t} = \{t_j\}$, $\mathbf{w} = \{w_j\}$: the arrays of the quadrature nodes and weights (of dimension n);
- $\mathbf{u} = \{u_j\}$: the array of the values, possibly approximated, of the function $u(t)$ at the nodes $t_j,\ j = 1, ..., n$;
- $\mathbf{x} = \{x_i\}$, the array of the target points (of dimension q). We point out that q should be large enough in order to appreciate the speed-up obtained by compression (e.g. typically $\mathbf{x} = \mathbf{t}$ and thus $q = n$ in the framework of integral equations solving);
- the generally nonlinear 3-variate kernel function $K(x, t, u)$, where $x, t \in [a, b]$ and $u \in D \subseteq \mathbb{R}$, which in the linear case has the form $K(x, t, u) = H(x, t)u$; notice that this function is defined by the user in the M-function, `kernel.m`, of the quoted Matlab toolbox, where we also provide some examples of kernels.

The algorithm can be briefly described as follows, by using a pseudo-code notation. The corresponding Matlab implementation can be found in the M-function `chebcompr.m` of the toolbox.

Algorithm: fast evaluation of a discrete integral operator by Chebyshev compression

1. Set the initial degree $m := m_0$, the degree increasing percentage α, the stalling detection parameter $\theta \in (0, 1)$, the relative tolerance tol, the threshold degree control parameter σ (by default $m_0 := 2$, $\alpha := 0.5$, $\theta := 0.25$, $tol := 10^{-6}$ and $\sigma := 1.0$), and the iteration counter $it := 0$. Compute the maximum number of iterations
$$itmax := \left\lfloor \frac{\log\left[1 + \sigma\alpha nq/(m_0(n+q))\right]}{\log(1+\alpha)} - 1 \right\rfloor .$$

2. $\mu := m$, $m := \lfloor (1+\alpha)m \rfloor + 1$, $it := it + 1$.

3. Compute $\{\Phi_n(\xi_k^m)\}$, where $\{\xi_k^m\}$, $k = 0, \ldots, m$, are the *Chebyshev-Lobatto* nodes in $[a, b]$, cf. (6). The function $\Phi_n(x)$ is implemented by the M-function `funPhi.m` of the toolbox.

4. Compute the coefficients $\{c_s\}$, $s = 0, \ldots, m$ of the $(m+1)$-term truncated Chebyshev series, $S_{m,n}(x)$ in (4), by the Gauss-Lobatto quadrature formula as in (5).

5. Compute the (relative) compression error indicator ind as
$$ind := \frac{\sum_{s=\mu+1}^{m} |c_s|}{\max_{k=0,\ldots,m} |\Phi_n(\xi_k^m)|} . \tag{9}$$

6. `If` $it = 1$ `then` $ind_{old} := 2\,ind/(1-\theta)$ `else` $ind_{old} := ind$.

7. `If` ($ind > tol$ `and` $((1-\theta) \cdot ind_{old}) > ind$ `or` $(ind > ind_{old} \cdot (1+\theta))$ `and` $it < itmax$) `goto` step 2. .

8. `If` ($it < itmax$) `then` compute $\mathbf{v} := \{S_{m,n}(x_i)\}$, $i = 1, \ldots, q$.

9. **output:** the final approximation $\mathbf{v} \approx A_n \mathbf{u}$, up to a relative error smaller than ind (unless $it \geq itmax$, which means that the result is not acceptable).

Some remarks on the implementation are now in order. A key point in the algorithm above is given by the *a posteriori* relative error indicator *ind* in (9). Indeed, given in general a sequence $\{m_j\}$, the relative error $E_{j,n}$ of the Chebyshev series approximation $S_{m_j,n}$ of Φ_n at the target nodes, can be estimated as

$$E_{j,n} := \frac{\|\mathbf{v}_j - A_n\mathbf{u}\|_\infty}{\|A_n\mathbf{u}\|_\infty} \approx \mathcal{E}_{j,n} := \frac{\sum_{s=m_j+1}^{m_{j+1}} |c_s|}{\max_{k=0,\dots,m_{j+1}} |\Phi_n(\xi_k^{m_{j+1}})|} \tag{10}$$

where

$$\mathbf{v}_j = \{S_{m_j,n}(x_i)\} \ , \quad i = 1, \dots, q \ , \quad j = 0, 1, 2, \dots \ . \tag{11}$$

As it is clear from the algorithm (see step 2.), the sequence $\{m_j\}$ is constructed by multiplying by a fixed factor $(1+\alpha)$, starting from m_0 (which by default is set to 2). This choice is different from the fixed step incremental adopted in [30], and is justified by the fact that the a posteriori estimate $\mathcal{E}_{j,n}$ in (10) might not be reliable in nonsmooth instances, when the increment $m_{j+1} - m_j$ is too "small". Notice that the estimate (10) is also different from that used in [30], namely $E_{j,n} \approx \|\mathbf{v}_{j+1} - \mathbf{v}_j\|_\infty / \|\mathbf{v}_{j+1}\|_\infty$, which requires the computation of all the vectors $\mathbf{v}_j$. Here, instead, we compute only the Chebyshev coefficients during the iterations, while the approximation at the target nodes is computed just once at the end.

A few more comments have to be made about the stopping criterion of the iteration (step 7.). Indeed, the compression process stops either when the relative error indicator *ind* goes below a fixed tolerance *tol* (that is when convergence occurs), or when the stalling phenomenon takes place, or finally when the degree of the approximating polynomials exceeds a given threshold, which says that the method is considered no more convenient with respect to direct evaluation. This latter occurrence is checked by a threshold for the number of iterations, *itmax*, which is computed a priori (see below). The possible stalling is detected when

$$\left| \frac{\mathcal{E}_{j,n}}{\mathcal{E}_{j-1,n}} - 1 \right| \leq \theta \ , \tag{12}$$

cf. (10), where the parameter $\theta \in (0, 1)$ is defined by the user (by default it is set to 0.25).

Moreover, the choice of the maximum number of iterations *itmax* is justified as follows. At a given degree $m = m_0(1 + \alpha)^{it}$, we know that the total cost of the next compression step will be of $\mathcal{O}(\rho m(n + q))$, with

$\rho = m_0((1+\alpha)^{it+2} - 1)/\alpha$, to be compared with the $\mathcal{O}(nq)$ cost of direct evaluation, which means a *speed-up* of the order of $nq/(\rho m(n+q))$. Therefore, when $\rho \geq \sigma nq/(n+q)$, in the next step we would exceed a user-defined fraction σ of the complexity of direct evaluation. If, for example, the user considers acceptable a speed-up of at least one order of magnitude, then he will set $\sigma = 0.1$; our default choice, $\sigma := 1$, forces the exit when it is realized that the operator can not be compressed at all.

3 Numerical examples

In this Section, we present the results of the application of the Chebyshev compression algorithm, to discrete integral operators with kernels in three different classes. The first one concerns a *nonlinear smooth* kernel taken from the literature on Uryshon integral equations, where convergence of the compression process occurs. The second example, which comes from the framework of second-kind linear integral equations, involves a classical *weakly singular* kernel of *logarithmic* type. Here the quoted *stalling* phenomenon takes place. The last example concerns the evaluation of a discrete linear transform which has received some attention in the recent literature on fast methods, namely the *discrete Gauss transform*.

Notice that in all the examples we choose as target vector $\mathbf{x} = \mathbf{t}$, that means $n = q$ and thus that the direct evaluation of $A_n\mathbf{u}$ in (1) has a cost of $\mathcal{O}(n^2)$ flops. The tables are organized as follows, with reference to the algorithm description in the previous Section. The first row contains the number n of discretization points, while the second row exhibits the exit value of the approximating polynomial degree m, corresponding to the given values of n and to a relative error tolerance $tol := 10^{-6}$. Moreover, the symbol DE denotes the underlying discretization error at the discretization level n,

$$DE := \|\{Au(x_i)\} - A_n\mathbf{u}\|_\infty / \|\{Au(x_i)\}\|_\infty \ , \tag{13}$$

cf. (1). The compression error indicator, *ind*, is given by the a posteriori estimate (9), and AE denotes the actual compression error at the previous step, that is

$$AE := \|\mathbf{v}_j - A_n\mathbf{u}\|_\infty / \|A_n\mathbf{u}\|_\infty \ , \tag{14}$$

cf. (10) and (11), which is estimated by *ind*. All these errors are given with two significant figures. When *ind* is greater than the relative tolerance $tol := 10^{-6}$, it means that either a stalling has occurred, in which case both *ind* and AE should be of the order DE, or that the number of iterations has exceeded the prescribed threshold and thus the compression process is not successful.

Finally ESU and SU denote, respectively, the *estimated speed-up* and the true *speed-up* obtained in the evaluation of the discrete operator by the Chebyshev compression process. While SU is computed by using the `Matlab` function `cputime`, the estimate ESU is given by

$$ESU = \frac{n^2}{\sum_j (m_j\, n)} = \frac{n}{\sum_j m_j}\ , \tag{15}$$

where $\{m_j\}$ is the degrees sequence of the Chebyshev partial sums, cf. (4) and (11). In fact, the bulk of our algorithm is step 3., which is iterated until convergence or stalling occur, requires evaluating the kernel and costs $\mathcal{O}(m_j\, n)$ flops at the j-th step.

Our `Matlab toolbox` for Chebyshev compression of discrete integral operators, consists of the files `chebcompr.m` (the implementation of the compression algorithm), `funPhi.m` (the implementation of the discrete operator), and `kernel.m` (the kernel definition, containing some predefined sample kernels), downloadable from the web site:
`ftp://ftp.dimi.uniud.it/pub/demarchi/chebcoint.tar`

Example 1. Consider the discrete operator (1) with the *nonlinear* smooth kernel

$$K(x,t,u) = \frac{1}{x+t+0.3u},\quad x,t \in [0,1],\quad u > 0\,, \tag{16}$$

which also appears in the survey paper [2]. The input vectors $\mathbf{t}$ and $\mathbf{w}$ correspond to the Simpson quadrature rule at n (odd) equispaced nodes. Moreover, we have $\mathbf{u} = \{u_j\} = \{u(t_j)\}$ with $u(t) = 1/(1+t)$: in this case, the reference function $\Phi(x)$ in (2), which is used to evaluate the underlying discretization errors DE, can be explicitly computed

$$\begin{aligned}\Phi(x) \;&=\; \frac{1}{2}\left(\log\left(20x+23\right) - \log\left(10x+3\right)\right) + \frac{5(x-1)}{r(x)} \;\times \\ &\times\;\; \left(\tanh^{-1}\left(\frac{5(x+3)}{r(x)}\right) - \tanh^{-1}\left(\frac{5(x+1)}{r(x)}\right)\right)\ ,\end{aligned}$$

where $r(x) = (25x^2 - 50x - 5)^{1/2}$.

Due to the x-smoothness of the kernel, here the compression process always converges to the smooth function $\Phi_n(x)$, so that the exit values of

$ind \approx AE$ are all below the prescribed tolerance (see Table 1). Observe the manifest precision of ind as an estimator of AE, and moreover the "mesh-independence" of m and of the compression error, two features that are typical of smooth linear and nonlinear instances, cf. [30].

We stress that the choice of a regular operator argument $u(t)$ is not the point here. In fact, due to the "smoothing effect" of integration, even with a discontinuos argument like $u(t) = 2 + \mathrm{sign}(t - 0.5)$, we obtain a compression process very similar to that of the previous instance (compare Tables 1 and 2; clearly, with this $u(t)$ the quadrature is much less precise). The reference function is now

$$\Phi(x) = \log\left(\frac{50x^2 + 115x + 76}{50x^2 + 85x + 21}\right) . \tag{17}$$

On the other hand, troubles should be expected when $u(0) \approx 0$, since $K(x, 0, u(0))$ becomes nearly singular at $x = 0$. This has been confirmed by all our numerical experiments (not reported for brevity), where the compression process fails to converge just (and only) in the latter case.

We also point out the fact that here usual fast evaluation methods are not readily applicable, since they are conceived for linear operators: indeed, many are based on suitable expansions of the corresponding kernel, a technique that is hardly reproducible in nonlinear instances. On the contrary, our compression process works on the action of the operator, and thus it does not really distinguish between linearity and nonlinearity.

n	129	513	2049
m	17	17	17
DE	$1.8\,10^{-9}$	$6.9\,10^{-12}$	$3.0\,10^{-14}$
ind	$7.2\,10^{-7}$	$7.2\,10^{-7}$	$1.4\,10^{-9}$
AE	$1.1\,10^{-6}$	$1.1\,10^{-6}$	$1.1\,10^{-6}$
ESU	3.3	13.2	52.5
SU	2.1	8.5	34.0

Table 1: Chebyshev compression of the discrete Uryshon operator in Example 1 with $u(t) = 1/(1 + t)$.

Example 2. We deal with a linear instance of (1), in the case of a classical

n	129	513	2049
m	17	17	17
DE	$3.0\,10^{-4}$	$7.4\,10^{-5}$	$1.8\,10^{-5}$
ind	$6.7\,10^{-7}$	$6.7\,10^{-7}$	$6.7\,10^{-7}$
AE	$1.0\,10^{-6}$	$1.0\,10^{-6}$	$1.0\,10^{-6}$
ESU	3.3	13.2	52.5
SU	2.0	8.5	34.2

Table 2: Chebyshev compression of the discrete Uryshon operator in Example 1 with $u(t) = 2 + \mathrm{sign}(t - 0.5)$.

logarithmic *weakly-singular kernel*

$$K(x,t,u) = H(x,t)u \ , \quad H(x,t) = \log\left(|x-t|\right) \ , \quad x,t \in [0,1] \ . \tag{18}$$

Here the input vectors $\mathbf{t}$ and $\mathbf{w}$ are the equispaced nodes and the weights of a trapezoidal-like quadrature formula: in practice, the diagonal has been discarded to avoid the singularity, which is essentially the same discretization technique adopted in [1] for the purpose of illustration. This might be considered a poor choice in view of the singularity of the kernel, the error DE being reduced only by about a factor $1/3$ by quadrupling the nodes. In spite of this, it produces a good illustration of the stalling phenomenon (as discussed below), and above all shows clearly that the Chebyshev compression process makes utilizable in practice even a rough "general purpose" discretization formula, without requiring more sophisticated theoretical or numerical tools, tailored on the specific kernel (like, e.g., product integration, when applicable).

In this example an error stalling occurs, as it is clear from Table 3, where the exit values of AE are all greater than 10^{-6} (the prescribed tolerance), and close to the underlying discretization error DE; it is worth observing that the error indicator ind, which is still of the size of DE, is less consistent with AE than in the previous (smooth kernel) example.

Notice that

$$\Phi(x) = (1-x)\log(1-x) + x\log(x) - 1 \ , \tag{19}$$

cf. (2), due to the "smoothing effect" of integration, is much more regular than $K(\cdot, t, u(t))$, and also of $\Phi_n(x)$, as it can be clearly seen in Figures 1 and 2. As already discussed, the Chebyshev compression process applied to the continuous function Φ would converge, as it can be proved by the well-known Dini-Lipschitz test, cf. [25, Thm. 3.4] (while it does not converge on Φ_n); using in practice the discretization Φ_n instead of Φ for the implementation, from a certain m the discretization error becomes dominant, producing the observed error stalling. It should be observed that in the case of linear integral operators with singular kernels, differently from the hierarchical methods of $\mathcal{H}$-matrix theory (cf. [15, Prop. 5.2]), global polynomial approximations cannot lead to exponential convergence. This, however, may not be considered a serious drawback of our approach, since, as already observed, in practical applications it is not necessary to go below the discretization error.

n	128	512	2048	8192
m	11	26	61	92
DE	$3.1\,10^{-2}$	$9.4\,10^{-3}$	$2.8\,10^{-3}$	$7.9\,10^{-4}$
ind	$8.2\,10^{-3}$	$2.9\,10^{-3}$	$7.1\,10^{-4}$	$2.4\,10^{-4}$
AE	$4.0\,10^{-2}$	$1.0\,10^{-2}$	$3.1\,10^{-3}$	$1.0\,10^{-3}$
ESU	5.8	7.9	12.3	31.8
SU	4.1	5.9	9.6	25.2

Table 3: Chebyshev compression of the discrete "weakly-singular" linear operator in Example 2 with $u(t) \equiv 1$.

Example 3. We consider the problem of efficiently evaluating the *discrete Gauss transform*, that is

$$(G_\delta f)_i = \sum_{j=1}^{n} f_j\, e^{-|x_i - t_j|^2/\delta}\ ,\quad 1 \leq i \leq q\ ,\quad \delta > 0\ , \tag{20}$$

which plays a crucial role in several applications of mollification techniques, like e.g. the numerical solution of direct and inverse heat conduction problems, vortex methods, tomography, nonparametric statistics, noisy data filtering, cf. [12, 22, 32] and references therein. In (20), the $\{t_j\}$ are usually termed the "source points", the $\{x_i\}$ the "target points", and the $\{f_j\}$ the "strengths".

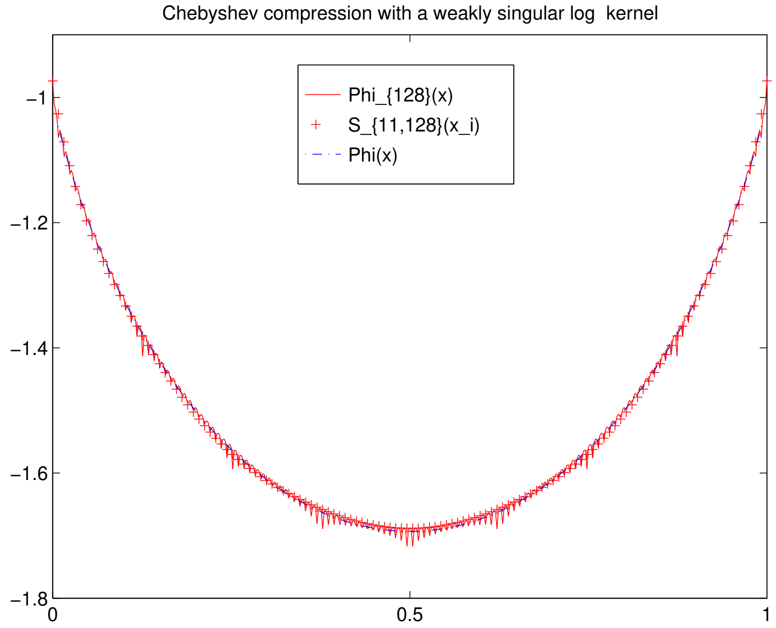

Figure 1: Comparing the graphs of the functions $\Phi_n(x)$, $S_{m,n}(x)$ and $\Phi(x)$ in Example 2., with $n = 128$ and $m = 11$.

In [12], a fast evaluation algorithm (termed the “Fast Gauss Transform”) was analyzed and implemented, which is tailored on the specific structure of the transform, i.e. on the Gaussian-like kernel, and turns out to be extremely efficient, being able to obtain impressive speed-ups (we recall that a further improvement of the method has been recently proposed in [13]). In practice, it reduces the computational cost from $\mathcal{O}(nq)$ to $\mathcal{O}(n+q)$, by means of a clever use of combined Hermite and Taylor expansions, and works in any dimension, with arbitrary distributions of sources and of targets in the unit box.

Here we apply for the first time the Chebyshev-like compression procedure to the discrete Gauss transform. In particular, we consider an example from [12], concerning a two-dimensional transform with sources and targets on a curve, which is in practice a one-dimensional problem. Indeed, Green-

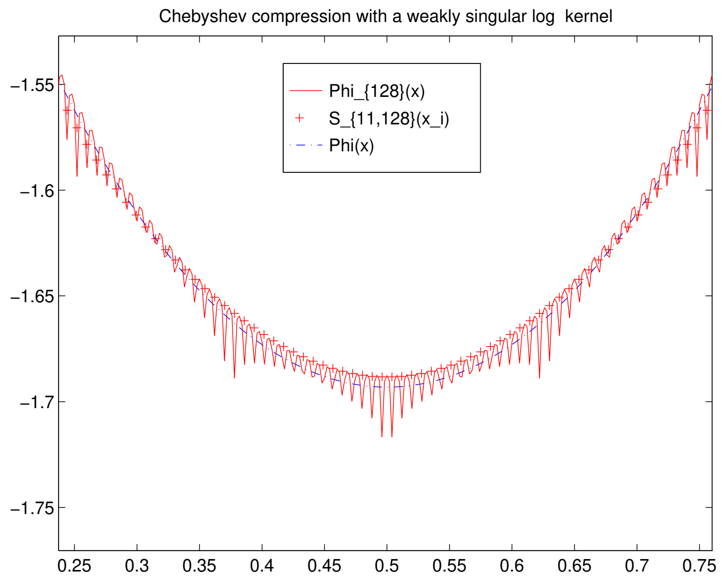

Figure 2: Zooming the graphs in Fig. 1.

gard and Strain applied their algorithm to the case when $\mathbf{x} = \mathbf{t}$ are n equispaced points on the unit circle, with strenghts $f_j = \cos(t_j)$, where t_j is the jth angle. By an obvious change of variables, the original two-dimensional discrete transform can be immediately rewritten as the one-dimensional one

$$\Phi_n(t_i) = \sum_{j=1}^{n} \cos(t_j)\, e^{-\left(|\cos(t_i)-\cos(t_j)|^2+|\sin(t_i)-\sin(t_j)|^2\right)/\delta}\ , \quad 1 \leq i \leq n\ , \tag{21}$$

where $t_j = 2\pi(j-1)/n$. This is of the form (1) with $w_j \equiv 1$ and $u_j = \cos(t_j)$, and strictly speaking is not a discretization of an integral transform, unless we would set $w_j = 2\pi/n$, which corresponds to a scaled integral and clearly does not affect the *relative* errors.

In Table 4 we report the results of the application of our compression algorithm to (21) with $\delta = 0.01$, correspondingly to three different values of

n, up to the very large problem $n = 102400$. Albeit we are not able to equal the Fast Gauss Transform [12], which is from three to five times faster, the performance of Chebyshev compression is quite satisfactory, especially if one considers that it has been conceived as a "general purpose" algorithm.

n	3200	12800	102400
m	92	92	61
ind	$3.7\,10^{-7}$	$9.3\,10^{-8}$	$8.2\,10^{-7}$
AE	$3.7\,10^{-7}$	$9.3\,10^{-8}$	$8.2\,10^{-7}$
ESU	12.4	49.6	616.8

Table 4: Chebyshev compression of the discrete Gauss transform in Example 3.

4 Remarks and future developments

Summarizing the results of the paper, we have produced a `Matlab toolbox` which implements Chebyshev-like polynomial compression of discrete integral operators, in the case of a one-dimensional bounded integration interval. The numerical code provides a robust "fast evaluator", which shows in the examples speed-up ratios of one and even two orders of magnitude. Moreover, it has been conceived to work both with linear or nonlinear operators, and to treat properly possible error stallings arising with nonsmooth kernels.

The present work represents only a first step in the application of polynomial compression to integral operators. Indeed, we conclude with a list of observations on some applications and possible extensions of the method.

I. It seems quite natural to apply polynomial compression in the numerical solution of first- and second-kind integral equations, namely in computing residuals within iterative solvers of the finite-dimensional systems, which are generated at each discretization level of Nyström method [2].

It is worth observing that the error stalling phenomenon arising with nonsmooth kernels, could be suitably exploited to obtain an indicator of the discretization error at the actual level. As it is well-known, this is a key point in all efficient implementations of "multilevel" solvers, in order to avoid iterating beyond the locally dominant error.

For second-kind equations, the resulting algorithm corresponds substantially to the composition of a suitable projection method (into the subspace of m-degree polynomials) with the Nyström method, where the subspace dimension can change dynamically along the levels; see [27] for a more complete discussion on this topic.

II. The present algorithm works on discrete integral operators, which are typically (but not necessarily) given discretizations of infinite-dimensional integral operators; see the r.h.s. of (1), (2). Alternatively, one might apply the compression technique directly on the function $\Phi(x)$, computing the required values $\Phi(\xi_k^m)$ (cf. step 3. of the algorithm description in Section 2) by means of an adaptive quadrature formula. This approach could be of interest, for example, within infinite-dimensional Picard-like and Newton-like solvers for nonlinear integral equations, cf. e.g. [19, 28].

III. As just observed, and made clear by Example 3, discrete operators that are discretizations of infinite-dimensional integral operators do not exhaust the applicability of Chebyshev compression. Indeed, restricting to the linear case think for example of any dense matrix-vector product, where the matrix coefficients are the values of some bivariate function at the nodes of some (also irregular) rectangular grid. We recall that the FFT is easily applicable only on regular grids with special (e.g. convolutional) kernels.

Our algorithm could be tried in such more general situations, in order to accelerate the evaluation. Even in these instances, however, when the mesh is regular in the t-direction we might rewrite the problem as the computation of a discretized (scaled) integral transform, simply by taking $w_j = \Delta t_j \equiv \Delta t$ (the constant step): this would clearly not affect the relative error of the compression process.

IV. Even within one-dimensional problems, polynomial compression is not the only possibility: rational compression, for example, could be tried. Moreover, the case of unbounded integration intervals is important in the applications (Wiener-Hopf operators, ...). Here, Laguerre or Hermite truncated expansions could be appropriate tools. The use of Fourier expansions in suitable weighted Sobolev-type spaces is suggested in the recent work [21], within numerical methods for linear integral equations with nonsmooth kernels. This kind of approach, however, differently from our "general purpose" algorithm, exploits explicitly a priori information on the specific structure of the solution (which is typically smooth apart from endpoint singularities of algebraic/logarithmic type).

V. The effectiveness of one-dimensional polynomial compression suggest extending the method to multi-dimensional (discrete) integral operators. This represents a challenge for a real impact of such a technique in the simulation of large-scale integral models; here, Chebyshev polynomials could play a role only for tensor-product domains.

With more general domains, interpolation at Fekete nodes on triangles is an appealing alternative, in view of its near optimal properties. The idea is to interpolate on a rough triangulation of the domain (much coarser than the underlying quadrature triangulation), in order to speed-up the evaluation process ("Fekete-like" compression). Again, the smoothing effect of integration would play a key role. We refer the reader, e.g., to [4, 5, 29] for the rapidly developing research topic concerning theory and applications of multivariate Fekete interpolation.

Acknowledgements. This work has been partially supported by the research project: "Numerical analysis of integral equations" (funds "ex 60%", 1998-1999) at the Department of Pure and Applied Mathematics of the University of Padova, by the research funds of the Department of Mathematics and Computer Science of the University of Udine, and by the GNIM-INdAM.

References

[1] B. Alpert, G. Beylkin, R. Coifman and V. Rokhlin, *Wavelet-like bases for the fast solution of second-kind integral equations*, SIAM J. Sci. Comput. **14** (1993), 159-184.

[2] K.E. Atkinson, *A survey of numerical methods for solving nonlinear integral equations*, J. Integral Equations Appl. **4** (1992), 15-46.

[3] K.E. Atkinson, "The Numerical Solution of Integral Equations of the Second Kind", Cambridge Monographs on Applied and Computational Mathematics 4, Cambridge University Press, Cambridge, 1997.

[4] L. Bos and S. De Marchi, *Fekete points for bivariate polynomials restricted to* $y = x^m$, East J. Approx. **6** (2000), 189-200.

[5] L. Bos, M.A. Taylor and B.A. Wingate, *Tensor product Gauss-Lobatto points are Fekete points for the cube*, Math. Comp., electronically published on April 19, 2000.

[6] A. Brandt and C.H. Venner, *Multilevel evaluation of integral transforms with asymptotically smooth kernels*, SIAM J. Sci. Comput. **19** (1998), 468-492.

[7] W. Dahmen, *Wavelet and multiscale methods for operator equations*, Acta Numer. **6** (1997), 55-228.

[8] S.A. Goreinov, E.E. Tyrtyshnikov and A.Yu. Yeremin, *Matrix-free iterative solution strategies for large dense linear systems*, Numer. Linear Algebra Appl. **4** (1997), 273-294.

[9] L. Greengard, *Fast algorithms for classical physics*, Science **265** (1994), 909-914.

[10] L. Greengard and V. Rohklin, *A fast algorithm for particle simulations*, J. Comput. Phys. **73** (1987), 325-348.

[11] L. Greengard and V. Rohklin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numer. **6** (1997), 229-269.

[12] L. Greengard and J. Strain, *The fast Gauss transform*, SIAM J. Sci. Stat. Comput. **12** (1991), 79-94.

[13] L. Greengard and X. Sun, *A new version of the fast Gauss transform*, Proceedings of the International Congress of Mathematicians, Vol. III (Berlin, 1998), Doc. Math. 1998, Extra Vol. III, 575-584.

[14] W. Hackbusch,"Integral Equations: Theory and Numerical Treatment", International Series of Numerical Mathematics, 120, Birkhäuser, Basel, 1995.

[15] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. I. Introduction to $\mathcal{H}$-matrices*, Computing **62** (1999), 89-108.

[16] W. Hackbusch and B.N. Khoromskij, *A sparse $\mathcal{H}$-matrix arithmetic. II. Application to multi-dimensional problems*, Computing **64** (2000), 21-47.

[17] W. Hackbusch and B.N. Khoromskij, *Towards $\mathcal{H}$-matrix approximation of the linear complexity*, in: Operator Theory: Advances and Applications, vol. 121, Birkhäuser, 2001, pp. 194-220.

[18] W. Hackbusch and Z. Nowak, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numer. Math. **54** (1989), 463-491.

[19] C.T. Kelley and J.I. Northrup, *A pointwise quasi-Newton method for integral equations*, SIAM J. Numer. Anal. **25** (1988), 1138-1155.

[20] S.R. Lustig, S. Rastogi and N. Wagner, *Telescoping fast multipole methods using Chebyshev economization*, J. Comp. Phys. **122** (1995), 317-322.

[21] G. Monegato and L. Scuderi, "Weighted Sobolev-type spaces and numerical methods for 1D integral equations", invited talk at the "*4th International Conference on Functional Analysis and Approximation Theory*", Acquafredda di Maratea (PZ, Italy), september 2000.

[22] D.A. Murio, "The Mollification Method and the Numerical Solution of Ill-Posed Problems", John Wiley, New York, 1993.

[23] R. Piessens, *Computing integral transforms and solving integral equations using Chebyshev polynomial approximations*, Numerical analysis in the 20th century, Vol. I, Approximation theory. J. Comput. Appl. Math. **121** (2000), 113-124.

[24] T.J. Rivlin, "An Introduction to the Approximation of Functions", Dover, New York, 1981.

[25] T.J. Rivlin, "Chebyshev Polynomials", John Wiley, New York, 1990.

[26] V. Rohklin, *Rapid solution of integral equations of classical potential theory*, J. Comp. Phys. **60** (1985), 187-207.

[27] A. Sommariva, *A fast Nyström-Broyden solver by Chebyshev compression*, in preparation.

[28] A. Sommariva and M. Vianello, *Computing positive fixed-points of decreasing Hammerstein operators by relaxed iterations*, J. Integral Equations Appl. **12** (2000), 95-112.

[29] M.A. Taylor, B.A. Wingate and R.E. Vincent, *An algorithm for computing Fekete points in the triangle*, SIAM J. Numer. Anal. **38** (2000), 1707-1720.

[30] M. Vianello, *Chebyshev-like compression of linear and nonlinear discretized integral operators*, Neural, Parallel and Sci. Comput. **8** (2000), 327-353.

[31] T. von Petersdorff, C. Schwab and R. Schneider, *Multiwavelets for second-kind integral equations*, SIAM J. Numer. Anal. **34** (1997), 2212-2227.

[32] S. Zhan and D.A. Murio, *Surface fitting and numerical gradient computations by discrete mollification*, Computers Math. Applic. **37** (1999), 85-102.