Armenian Journal of Mathematics
Volume 1, Number 1, 2008, 50-63

The Convergence Acceleration of Two-Dimensional Fourier Interpolation

Anry Nersessian* and Arnak Poghosyan**

Institute of mathematics
of National Academy of Sciences of Armenia
Bagramian ave.24B, 0019 Yerevan, Armenia

*nerses@instmath.sci.am, ** arnak@instmath.sci.am

Abstract. Hereby, the convergence acceleration of two-dimensional trigonometric interpolation for a smooth functions on a uniform mesh is considered. Together with theoretical estimates some numerical results are presented and discussed that reveal the potential of this method for application in image processing. Experiments show that suggested algorithm allows acceleration of conventional Fourier interpolation even for sparse meshes that can lead to an efficient image compression/decompression algorithms and also to applications in image zooming procedures.

Key words: Convergence Acceleration, Interpolation, Local Fourier Analysis, Image Processing, Image Interpolation, Image Compression, Image Zooming
Mathematics Subject Classification 2000: 41A60, 42-XX, 65Txx, 68U10, 94A08

1 Introduction

It is well known that Fourier series and Fourier interpolation are powerful tools for theoretical and applied investigations. The main drawback that diminishes their strength is the Gibbs phenomenon near the points of singularities of the approximated function.

Let f be a piecewise smooth function on $[-1,1]$ with jump points $\{a_k\}$, $-1=a_0<a_1<\cdots<a_{l-1}<1$, $2\le l<\infty$. **S**uppose that $f\in C^{q+1}$, $q\ge 0$ on each segment $[a_k,a_{k+1}]$, $k=1,\cdots,l-2$ and also on the segments $[-1,a_1]$, $[a_{l-1},1]$.

Denote by

$$A_{sk}=f^{(k)}(a_s+0)-f^{(k)}(a_s-0),\ k=0,\cdots,q;\ s=1,\cdots,l-1,$$

$$A_{0k}=f^{(k)}(-1)-f^{(k)}(1),\ k=0,\cdots,q$$

the jumps of f and its derivatives at the points $\{a_s\}$. By $\{f_n\}$ we denote Fourier coefficients of f

$$f_n=\frac{1}{2}\int_{-1}^{1}f(t)e^{-i\pi nt}dt,\ n=0,\pm1,\cdots. \tag{1}$$

By means of integration by parts we get for $n \neq 0$

$$f_n = \sum_{s=0}^{l-1} e^{-i\pi n a_s} \sum_{k=0}^{q} \frac{A_{sk}}{2(i\pi n)^{k+1}} + \frac{1}{2(i\pi n)^{q+1}} \int_{-1}^{1} f^{(q+1)}(t) e^{-i\pi n t} dt. \tag{2}$$

Consider now Bernoulli polynomials $\{B_k\}$, $k = 1, 2, \cdots$ with Fourier coefficients $\{B_{k,n}\}$

$$B_{k,n} = \begin{cases} 0, & n = 0 \\ \dfrac{(-1)^{n+1}}{2(i\pi n)^{k+1}}, & n = \pm 1, \pm 2, \dots \end{cases}$$

On the real line Bernoulli polynomials are considered as 2-periodic piecewise-smooth functions with "jump points" $a_k = 2k + 1$, $k = 0, \pm 1, \pm 2, \cdots$.

Expansion (2) leads to the representation

$$f(x) = \sum_{n=-\infty}^{\infty} f_n e^{i\,\pi\,n\,x} = U(x) + V(x) \tag{3}$$

where

$$U(x) = -\sum_{s=0}^{l-1} \sum_{k=0}^{q} A_{sk} \sum_{n=-\infty}^{\infty} e^{i\,\pi\,n\,(x - a_s + 1)} B_{k,n}$$

is a piecewise-polynomial function consisting of "shifted" Bernoulli polynomials and $V \in C^q(\mathbf{R})$. Hence, the sequence

$$f_N(x) = U(x) + V_N(x),\ V_N(x) = \sum_{n=-N}^{N} V_n e^{i\pi n x} \tag{4}$$

converges to f by the rate $o(N^{-q})$, $N \to \infty$ as coefficients $\{V_n\}$ of V are tending to zero by the order $o(n^{-q-1})$, $n \to \infty$.

This idea of convergence acceleration was suggested by A. Krylov as far back as in 1905 [1] but it was widely known only since 1960 after publications of C. Lanczos [2]. Note, that this method is exact for a finite sum of piecewise-polynomials when q and N are rather big and exact values of $\{A_{sk}\}$ are used. However, for a practical realization of the corresponding method one should know not only $(2N + 1)$ Fourier coefficients $\{f_n\}$, $n = 0, \pm 1, \cdots, \pm N$, but also the location of singularities $\{a_s\}$ and corresponding jump values $\{A_{sk}\}$.

The problem of jump reconstruction using only the finite number of Fourier coefficients was solved by K. Eckhoff ([4]-[6]). Namely, if jump points $\{a_s\}$ are known and N is rather big, then approximate values $\tilde{A}_{sk}$ can be found from a linear system arising from (2) by an appropriate choice of $n = n_s$, $s = 1, 2, \cdots, lq$, $|n_s| \leq N$

$$f_n = -\sum_{s=0}^{l-1} e^{-i\pi n (a_s - 1)} \sum_{k=0}^{q} \tilde{A}_{sk} B_{k,n},\ n = n_1, n_2, \cdots, n_{lq}. \tag{5}$$

It is natural (see [3]) to suppose that for some $0 < \alpha \le 1$

$$\alpha N \le |n_s| \le N, \quad s = 1, \dots, q+1. \tag{6}$$

Approximate locations of singularity points can be found using only finite number of Fourier coefficients $\{f_n\}$ by the method suggested, for example, in [8].

Similar method was developed for acceleration of conventional Fourier interpolation (see [5]). The corresponding scheme for a smooth function $f \in C^{q+1}[-1,1]$ is the following

$$I_{q,N}(f) = \sum_{k=0}^{q} A_k(f) B_k(x) + \sum_{n=-N}^{N} \check{V}_n e^{i\pi n x}, \tag{7}$$

where

$$A_k(f) = f^{(k)}(1) - f^{(k)}(-1),\ k = 0, \cdots, q-1.$$

$$\check{f}_n = \frac{1}{2N+1} \sum_{k=-N}^{N} f(x_k) e^{-i\pi n x_k}$$

and

$$\check{V}_n = \check{f}_n - \sum_{k=0}^{q} A_k(f) \check{B}_{k,n}. \tag{8}$$

Similar to (5), approximate values $\{\tilde{A}_k\}$ of the jumps $\{A_k\}$ can be found from the following system

$$\check{f}_n = \sum_{k=0}^{q} \widetilde{A}_k \check{B}_{k,n}, \quad n = n_1, n_2, \dots, n_{q+1}. \tag{9}$$

Note that pseudoinverses can be used while solving the systems (5) or (9) for singular matrices. In our experiments we do not use this way but recommend it for a safe approach.

A scheme for two-dimensional rectangles (when approximated function $f(x,y)$ is rather smooth) was suggested in [9] and [10]. Below we redescribe this approach with detailed analysis of some numerical experiments as well as with justification of possible applications in image processing problems.

2 Two-dimensional acceleration

2.1 Basic Notations

For $f \in C^{2q}[-1,1]^2$ denote

$$f^{(k,s)}(x,y) = \frac{\partial^{k+s} f(x,y)}{\partial x^k \partial y^s},\ k, s = 0, \cdots, q,$$

$$u_k(y) = f^{(k,0)}(1,y) - f^{(k,0)}(-1,y),\ k = 0, \cdots, q,$$
$$v_k(x) = f^{(0,k)}(x,1) - f^{(0,k)}(x,-1),\ k = 0, \cdots, q,$$

$$\Delta_{k,s} = f^{(k,s)}(1,1) - f^{(k,s)}(-1,1) - f^{(k,s)}(1,-1) + f^{(k,s)}(-1,-1),\ k,s = 0,\cdots,q.$$

By f_{nm}, $u_{k,m}$ and $v_{s,n}$ we denote the following Fourier coefficients

$$f_{nm} = \frac{1}{4}\int_{-1}^{1}\int_{-1}^{1} f(x,y)e^{-i\pi(nx+my)}dxdy,$$

$$u_{k,m} = \frac{1}{2}\int_{-1}^{1} u_k(y)e^{-i\pi my},\ v_{k,n} = \frac{1}{2}\int_{-1}^{1} v_k(x)e^{-i\pi nx},\ k = 0,\cdots,q.$$

The next two lemmas can easily be proved by means of integration by parts (see [10]).

Lemma 1. *For any $f \in C^{2q+2}[-1,1]^2$, $q \geq 0$ the following formula holds for $n,m \neq 0$*

$$f_{nm} = \frac{(-1)^{n+1}}{2}\sum_{k=0}^{q}\frac{u_{k,m}}{(i\pi n)^{k+1}} + \frac{(-1)^{m+1}}{2}\sum_{s=0}^{q}\frac{v_{s,n}}{(i\pi m)^{s+1}} -$$
$$-\frac{(-1)^{n+m}}{4}\sum_{s=0}^{q}\sum_{k=0}^{q}\frac{\Delta_{k,s}}{(i\pi n)^{k+1}(i\pi m)^{s+1}} +$$
$$+\frac{1}{4(i\pi n)^{q+1}(i\pi m)^{q+1}}\int_{-1}^{1}\int_{-1}^{1} f^{(q+1,q+1)}(t,z)e^{-i\pi(nt+mz)}dzdt. \tag{10}$$

Lemma 2. *For any $f \in C^{2q+2}[-1,1]^2$, $q \geq 0$ the following formula holds for $n,m \neq 0$*

$$f_{n0} = \frac{(-1)^{n+1}}{2}\sum_{k=0}^{q}\frac{u_{k,0}}{(i\pi n)^{k+1}} + \frac{1}{4(i\pi n)^{q+1}}\int_{-1}^{1}\int_{-1}^{1} f^{(q+1,0)}(t,z)e^{-i\pi nt}dzdt,\ n \neq 0. \tag{11}$$

$$f_{0m} = \frac{(-1)^{m+1}}{2}\sum_{k=0}^{q}\frac{v_{s,0}}{(i\pi m)^{s+1}} + \frac{1}{4(i\pi m)^{q+1}}\int_{-1}^{1}\int_{-1}^{1} f^{(0,q+1)}(t,z)e^{-i\pi mz}dzdt,\ m \neq 0. \tag{12}$$

Note that for every $f \in L_2(-1,1)^2$

$$f(x,y) = \sum_{n,m=-\infty}^{\infty} f_{nm}e^{i\pi(nx+my)} = \sum_{nm=-\infty}^{\infty}{}' f_{nm}e^{i\pi(nx+my)} +$$
$$+\sum_{n=-\infty}^{\infty}{}' f_{n0}e^{i\pi nx} + \sum_{m=-\infty}^{\infty}{}' f_{0m}e^{i\pi my} + f_{00}, \tag{13}$$

where primes indicate that zero terms are omitted. Substitution of (10), (11) and (12) into (13) leads to the following expansion of f

$$f(x,y) = U(x,y) + V(x,y), \tag{14}$$

where 2-periodic function $V \in C^{2q+2}(R^2)$ for $f \in C^{2q+4}[-1,1]^2$ and

$$U(x,y) = \sum_{k=0}^{q} u_k(y)B_k(x) + \sum_{s=0}^{q} v_s(x)B_s(y) - \sum_{s=0}^{q}\sum_{k=0}^{q}\Delta_{ks}B_k(x)B_s(y). \tag{15}$$

Expansion (14) leads to the following decomposition

$$S_{q,N}(f) = U(x,y) + \sum_{n,m=-N}^{N} (f_{nm} - U_{nm}) e^{i\pi(nx+my)} \tag{16}$$

and interpolation

$$I_{q,N}(f) = U(x,y) + \sum_{n,m=-N}^{N} (\check{f}_{nm} - \check{U}_{nm}) e^{i\pi(nx+my)}, \tag{17}$$

where

$$\check{f}_{nm} = \frac{1}{(2N+1)^2} \sum_{k,s=-N}^{N} f(x_k, x_s) e^{-i\pi(nx_k+mx_s)}, \; x_k = \frac{2k}{2N+1}.$$

In the next two theorems the asymptotic behavior of $S_{q,N}$ and $I_{q,N}$ **is** revealed.

Theorem 1 [10]. *If $f \in C^{2q+4}([-1,1]\times[-1,1])$, $q \geq 0$ then*

$$\lim_{N\to\infty} (2N+1)^{q+2} ||f - S_{q,N}(f)||^2 = \frac{2^{2q+3}}{(2q+3)\pi^{2q+4}} \int_{-1}^{1} \left(|\widetilde{\varphi}_q(x)|^2 + \left|\widetilde{\psi}_q(x)\right|^2 \right) dx,$$

where

$$\widetilde{\varphi}_q(y) = \int_{-1}^{1} B_q(t)\varphi_q(y-t)dt, \; \widetilde{\psi}_q(x) = \frac{1}{2}\int_{-1}^{1} B_q(t)\psi_q(x-t)dt.$$

$$\varphi_q(y) = f^{(q+1,q+1)}(1,y) - f^{(q+1,q+1)}(-1,y),$$

$$\psi_q(x) = f^{(q+1,q+1)}(x,1) - f^{(q+1,q+1)}(x,-1)$$

and

$$\|f\| = \left(\int_{-1}^{1} |f(x)|^2 dx \right)^{1/2}.$$

Theorem 2 [10]. *If $f \in C^{2q+4}([-1,1]\times[-1,1])$, $q \geq 0$ then*

$$\lim_{N\to\infty} (2N+1)^{2q+3} ||f - I_{q,N}(f)||^2 =$$

$$= \frac{1}{\pi^{2q+4}} \left(\frac{2^{2q+3}}{2q+3} + \frac{1}{2} \int_{-1/2}^{1/2} \left| \sum_{s\in\mathbf{Z}}{}' \frac{(-1)^s}{(x+s)^{q+2}} \right|^2 dx \right) \int_{-1}^{1} (|\widetilde{\varphi}_q(x)|^2 + |\widetilde{\psi}_q(x)|^2) dx.$$

2.2 The function $U(x, y)$

In this section we describe the procedure of approximation of the function U. To this end approximations of functions $u_k(x)$, $v_k(x)$ as well as of numbers $\Delta_{k,s}$ must be realized.

From expansion (14) we derive

$$\frac{1}{2N+1}\sum_{m=-N}^{N} f(x_m, y_n)e^{-i\pi x_m \ell} = \sum_{k=0}^{q} u_k(y_n)\check{B}_{k,\ell} + \sum_{s=0}^{q} B_s(y_n)\left(\check{v}_{s,\ell} - \sum_{k=0}^{q}\Delta_{k,s}\check{B}_{k,\ell}\right) +$$

$$+\frac{1}{2N+1}\sum_{m=-N}^{N} V(x_m, y_n)e^{-i\pi x_m \ell},\ |n| \le N,\ \ell = \ell_1, \cdots, \ell_{q+1}.$$

Taking into account that the last two terms on the right hand side can be discarded if $\alpha N \le |\ell_s| \le N$, $N \to \infty$, $s = 1, \ldots, q+1$ for some $0 < \alpha \le 1$ we get the following systems of linear equations for determining unknown values $u_k(y_n)$

$$\frac{1}{2N+1}\sum_{m=-N}^{N} f(x_m, y_n)e^{-i\pi x_m \ell} = \sum_{k=0}^{q} u_k(y_n)\check{B}_{k,\ell},\ |n| \le N,\ \ell = \ell_1, \cdots, \ell_{q+1}. \tag{18}$$

Hence, for any fixed value of n, $|n| \le N$ we need to solve a system of linear equations with $(q+1)$ unknowns $u_k(y_n)$. As it is well known this can be done by $2/3(q+1)^3$ arithmetic operations using Gaussian elimination, but taking into account that in our case we are solving Vandermonde linear system the number of operations can be reduced to $5/2(q+1)^2$ by the well known Bjorck - Pereyra algorithm (see [11]). Note also that this algorithm is not only faster but it is often more accurate than standard methods. As a result, the overall number of multiplications for calculation of numbers $u_k(y_n)$ is $O(q^2 N)$, $qN \to \infty$. It is worth to mention that in our algorithms we use the choice $\ell_1 = N, \ell_2 = -N, \ell_3 = N-1$ and so on. Further we suppose that $N > q/2$.

Calculation of the values $v_s(x_n)$ can be carried out similarly by solving the system

$$\frac{1}{2N+1}\sum_{n=-N}^{N} f(x_m, y_n)e^{-i\pi y_n \ell} = \sum_{s=0}^{q} v_s(x_m)\check{B}_{s,\ell},\ |m| \le N,\ \ell = \ell_1, \cdots, \ell_{q+1}.$$

Finally, for determination of $\Delta_{k,s}$ we derive from (14)

$$\check{f}_{nm} = \sum_{k=0}^{q}\check{u}_{k,m}\check{B}_{k,n} + \sum_{s=0}^{q}\check{v}_{s,n}\check{B}_{s,m} - \sum_{k=0}^{q}\sum_{s=0}^{q}\Delta_{k,s}\check{B}_{k,n}\check{B}_{s,m},$$

$$n = n_1, \cdots, n_{q+1},\ m = m_1, \cdots, m_{q+1}.$$

Now the functions $u_k(x)$ and $v_k(x)$ can be reconstructed according to (7).

As a result, the complexity of construction of the function U is similar to the complexity of one-dimensional interpolation and can be discarded while calculating the overall complexity of two-dimensional acceleration. Hence, the overall complexity of accelerated interpolation for fixed q is $O(N^2 \log N), N \to \infty$ as for Fourier interpolation.

2.3 Two examples

In this section a comparison of an accelerated interpolation with the classical Fourier interpolation is considered on the base of two synthetic examples. Experiments are carried out by *Wolfram Mathematica 6* package. In Tables below the value $a \times 10^{-b}$ is indicated as $a, -b$.

First, we consider the following analytic function with high oscillations and big edge values (see Figure 1)

$$f_1(x,y) = \left(x \sin(2y) + \frac{1}{2}\right) \sin\left(5x^2 + 3y^2 - y\right). \tag{19}$$

It is a complicated example for an interpolation on a sparse grid.

In *Table* 1 absolute errors of a corresponding interpolation are presented for different values of q and N. The column $q = -1$ corresponds to the classical Fourier interpolation. We see that even for small values of N the accelerated interpolation gives more precise results.

	q=-1	q=0	q=1	q=2	q=3	q=4
N=2	4,-1; 4,-1	3,-1; 3,-1	2,-1; 2,-1	2,-1; 2,-1	1.5,-1; 1.5,-1	—
N=4	4,-1; 2,-1	1.5,-1; 4,-2	1,-1; 3,-2	1,-1; 1.5,-2	1,-1; 1.5,-2	1,-1; 1,-2
N=8	4,-1; 8,-2	4,-2; 3,-3	2,-2; 5,-4	8,-3; 1,-4	4,-3; 8,-5	3,-3; 2,-5
N=16	3,-1; 2,-2	2,-2; 3,-4	4,-3; 4,-5	6,-4; 4,-5	2,-4; 3,-6	8,-5; 4,-5
N=32	3,-1; 2,-2	1,-2; 6,-5	1,-3; 4,-5	1,-4; 6,-5	1.5,-3; 4,-5	2,-3; 4,-5

Table 1. Absolute errors of an interpolation on the square $[-1 + 1/N, 1 - 1/N]^2$ *(after the semicolon - on the square* $[-1/2, 1/2]^2$*) for* $f_1(x, y)$ *using* $(2N + 1)^2$ *grid points. The column* $q = -1$ *corresponds to the Fourier classical interpolation. The value* $a \times 10^{-b}$ *is indicated as* $a, -b$

In *Figure* 1 the graph of $f_1(x, y)$ and graphs of the edge jump functions are presented, and in *Figure* 2 we have presented the U and V functions for this example when $N = 4$, $q = 0$. From *Figure* 3 we see that corresponding edge jumps are reduced compared with *Figure* 1 (right).

The next test function is not smooth. It has jumps in its first derivatives on some lines.

$$f_2(x,y) = xy \left| x^2 - 2y^2 - \frac{1}{16} \right| - \left| \sin\left(-x^3 + 2y + \frac{1}{2}\right) \right| \tag{20}$$

The graph of this test function is presented in *Figure* 4 together with the edge jumps. Note, that here *Theorem* 2 is not valid (as $\partial^2 f_2(x,y)/\partial x \partial y$ does not exist on mentioned lines) but from *Table* 2 we see that the algorithm works in this case also.

On the rectangle $[-.4, 0] \times [-.75, -.35]$ this function is smooth ($f \in C^\infty$) and in *Table* 2 we have information about interpolation on this smoothness area also.

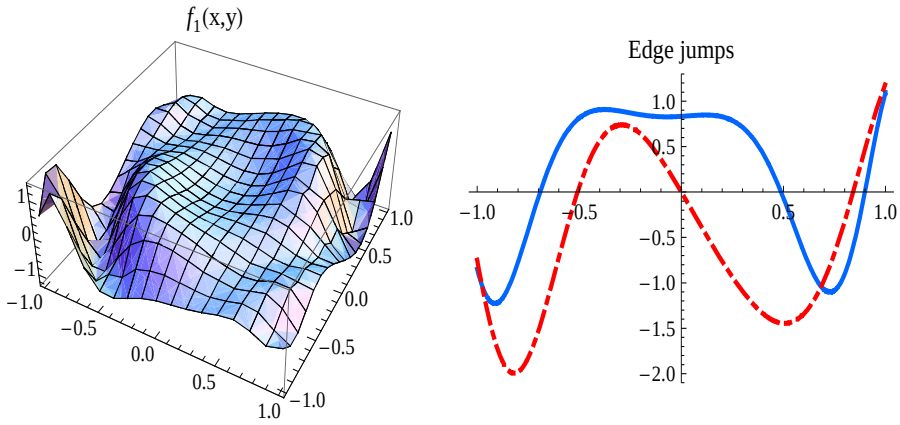

Figure 1: Graph of the function $f_1(x,y)$ values (left) as well as (right) values of $f_1(x,1) - f_1(x,-1)$ (blue) and $f_1(1,y) - f_1(-1,y)$ (red-dashing),$-1 \le x,y \le 1$

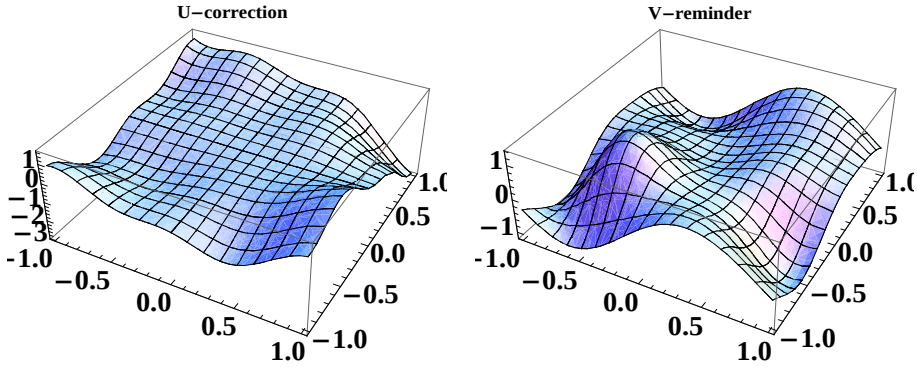

Figure 2: The graphs of U (left) and V (right) for the function $f_1(x,y)$ when $N = 4$,

	q=-1	q=0	q=1	q=2
N=2	3,-1; 2,-1	2,-1; 1.5,-1	4,-1; 2,-1	3,-1; 2,-1
N=4	4,-1; 2,-1	1.5,-1; 3,-2	1.5,-1; 1,-1	1.5,-1; 1,-1
N=8	4,-1; 1,-1	1.5,-1; 8,-2	1,-1; 3,-2	2,-1; 4,-2
N=16	4,-1; 6,-2	4,-2; 6,-3	3,-1; 1,-2	1,0; 1.5,-2
N=32	6,-1; 3,-2	2,-2; 3,-3	1.5,-1; 3,-3	6,0; 4,-3
N=64	6,-1; 1.5,-2	1,-2; 1,-3	1.5,0; 1,-3	8,1; 1.5,-3

Table 2. Absolute errors of interpolation on the square $[-1+1/N, 1-1/N]^2$ *(after the semicolon - on the rectangle* $[-.4,0] \times [-.75,-.35]$*) for* $f_2(x,y)$ *using* $(2N+1)^2$ *grid points.* $q = -1$ *corresponds to Fourier interpolation. The value* $a \times 10^{-b}$ *is indicated as* $a,-b$

In *Figure* 3 the graphs of absolute values of edge jumps for the last example are presented (compare with Figure 4).

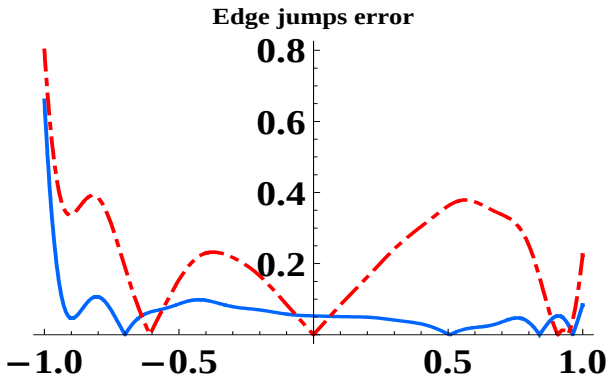

Figure 3: Graphs for absolute values of edge jumps for the function $f_1(s,y) - U(x,y)$ if $N = 4, q = 0$. Colors are the same as in Figure 1.

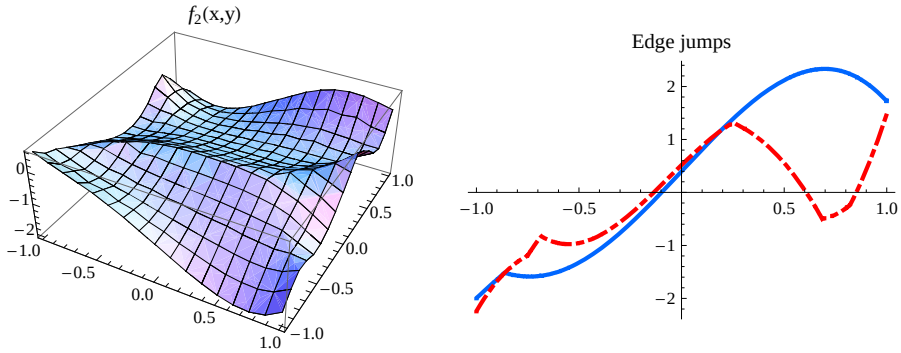

Figure 4: Graph of the function $f_2(x,y)$ values (left) as well as (right) values of $f_2(x,1) - f_2(x,-1)$ (blue) and $f_2(1,y) - f_2(-1,y)$ (red-dashing)

3 An Application in Image Analysis and Synthesis

3.1 The problem

It is well known that Fourier series expansion (or interpolation) is rather efficient for analysis and synthesis of smooth and periodic signals or images. In this case the rate of the decay of Fourier coefficients is rather big, that leads to very sparse representations in frequency array. Particularly, it allows making high order compression of an image with a low loss of the information as well as realizing approximation/interpolation with high precision.

This is not the case for non-periodic signals/images (see Introduction). It is obvious that real signals/images are usually non-periodic. Even if the original signal/image is periodic, its local segments (which are used for local analysis and synthesis) are non-periodic. Hence, an application of the accelerated interpolation is in place.

Note that a similar method of division of a smooth image $f(x,y), (x,y) \in [0,1]^2$ into two components $f = u + v$ (artificial u and residual v) was used in the interesting series of papers of Naoki Saito with co-authors ([12] - [15]) in the case when $u(x,y)$ is a solution of a discrete analogue of polyharmonic equation $\Delta^m u = 0$, $m \geq 1$, with boundary values

derived from corresponding values of f. For numerical implementation of this approach an algorithm suggested by Averbuch, Braverman, Israeli, and Vozovoi (ABIV method [16]) was used. Actually this algorithm is based on well known method of separation of variables. For the translation to the frequency array discrete Fourier transform (DFT) as well as discrete Cos/Sin (DCT/DST) transforms were applied. One can find descriptions of these algorithms and several applications to real images, as well as detail analysis of related works in [15].

Our method can be viewed as an alternative to the mentioned approach. We think that the acceleration on the base of DFT is the most preferable since (see [12]) it represents the image orientation patterns better than DCF and DST.

3.2 Numerical results

Possibilities of our algorithm, in general, were investigated above. Here we examine the accelerated interpolation for applications in image processing. We are interested in lossless/lossy compression/decompression possibilities as well as in the memory consumption.

Our experiments show that usually both for small and big N the accelerated interpolation is more precise compared with the Fourier classical interpolation. Moreover, when value of $|m \times n|$ relatively big then coefficients $\{V_{mn}\}$ of V usually are smaller compared with DFT coefficients $\{f_{mn}\}$ of f (see Table 3). Hence, after a quantization procedure more coefficients of V can be discarded compared with DFT of f ones. (see Figure 6, 7 also). This will lead to a serious memory consumption economy. An additional memory will be required only for saving the function U. For example, when $q = 0$, $N = 2$ and $f = f_1$, then U has the following simplified form

$$U(x,y) = -0.128\,x + 0.167\,y - 1.04\,x\,y + 0.615\,y\,\cos(\pi x) - 0.367\,y\ cos(2\pi x) + 0.55\,x\,\cos(\pi y) - 0.429\,x\,\cos(2\pi y) + 0.356\,y\,\sin(\pi x) - 0.376\,x\,\sin(\pi y).$$

and hence only 9 coefficients are necessary for describing this function in the memory.

0.103 0.167 0.088 0.088 0.031	0.015 0.053 0.084 0.053 0.015
0.091 0.031 0.158 0.034 0.061	0.07 0.264 0.172 0.09 0.07
0.038 0.051 0.074 0.051 0.038	0.015 0.053 0.074 0.052 0.015
0.061 0.034 0.158 0.031 0.091	0.07 0.09 0.172 0.264 0.07
0.031 0.088 0.088 0.167 0.103	0.015 0.052 0.084 0.053 0.015

Table 3. The matrix of absolute values of DFT ($q = 0$, $N = 2$) using the Fourier interpolation for f_2 (left) and the corresponding remainder matrix of absolute values of coefficients for $V(x,y)$ (right)

In applications the cases $N = 4$ and $N = 8$ are important and correspond to $JPEG$ algorithm, which scans the image with 8×8 grids, and to $JPEG2000$ algorithm, which scans the image with 16×16 grids (see [17], [18]). For these cases the economy can be much more as discrete coefficients of V tend to zero more rapid compared with DFT of f ones.

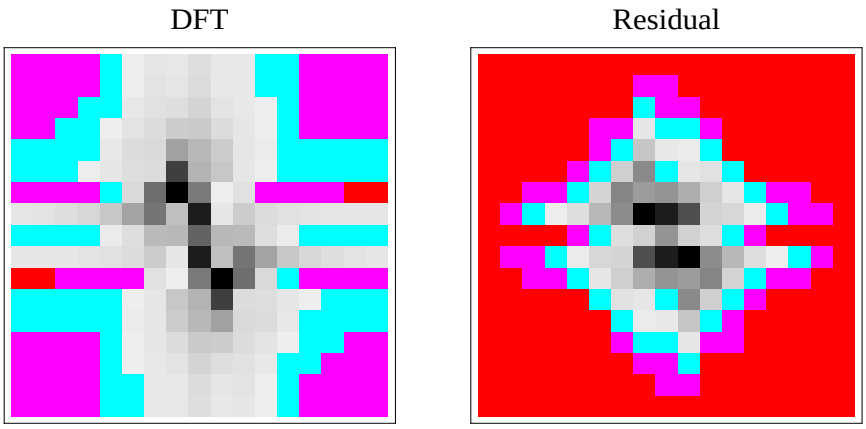

Figure 5: Data for compression when $f = f_1(x,t), N = 8, q = 0$. Quantization is applied to DFT and the residual using 101 steps. Pixel values: $Red = 0; 1 \leq Magenta \leq 3, 4 \leq Cyan \leq 6, others \geq 7$.

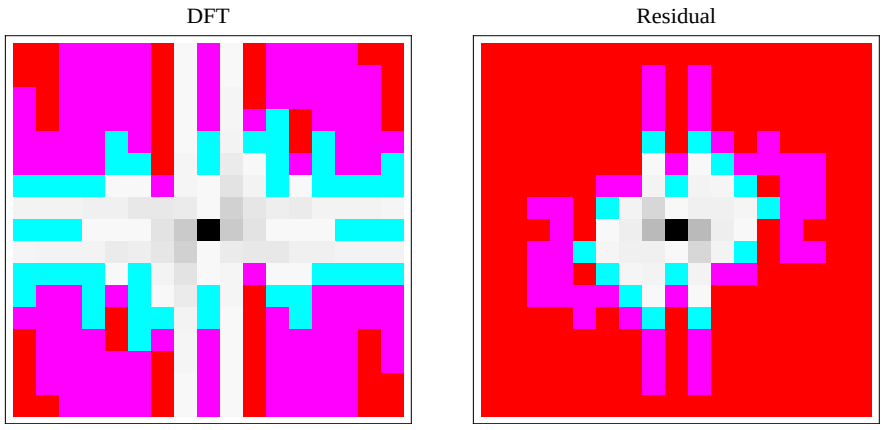

Figure 6: Data for compression when $f = f_2(x,t), N = 8, q = 0$. Quantization is applied to DFT and the residual using 201 steps. Pixel values: $Red = 0$, $1 \leq Magenta \leq 6$, $7 \leq Cyan \leq 12$, $others \geq 13$

Figure 6 shows a situation connected with f_1 when $N = 8$ and $q = 0$. In this case we must keep for function $U(x,y)$ (with correspondingly discarded coefficients) 26 correspondingly discarded coefficients during compression using discarded Red pixels, 20 coefficients using Magenta + Red pixels and only 15 coefficients using Magenta +Red + Cyan pixels. We can compress DFT file by discarding only two red pixels while we have 229 red pixels in the residual file (right). Thus total amount of requested memory using our method is 204 pixels instead of 289 and it is about lossless compression. Similar situation is observed for Magenta + Red and Magenta +Red + Cyan pixels although here DFT file can be only a bit compressed.

The function $f_2(x,y)$ is more acceptable as a test function for image applications (see Figure 7). In this case we must keep 31 coefficients for function $U(x,y)$ while discarding using Red pixels, 20 coefficients using Magenta + Red pixels and only 13 coefficients using Magenta +Red + Cyan pixels. Here we can compress DFT file by discarding 45

Red pixels so total amount of requested memory is 244 pixels instead of 289. At the same time in the residual file we have 204 red pixels. Thus total amount of requested memory for our method is 116 pixels. In that way we have discarded only .5% of the energy so it is practically lossless compression. One can easily make comparisons for other color pixels.

Note that for simplicity sake we are not use the procedure of multiplication by quantization matrix (see [17]) but it does not change the general situation.

In our other experiments with different synthetic images without jumps the situation was similar. Moreover the ratio of compression memory amounts $(using\, DFT)$: $(using\, our\, method)$ was stable grow with growing of N.

4 Conclusion

Described algorithm is a natural generalization of a method developed earlier for one-dimensional case (see [1]-[7]). The transformation of this method to more general $N_1 \times N_2$ grid is obvious. Generalization of this acceleration for Sin/Cos (DST/DCT) - based interpolations is also possible without great efforts.

The precision of approximation by accelerated interpolation is directly connected with the smoothness of approximated function. For smooth functions the theoretical estimates are highly confirmed with the results of numerical experiments. For these functions the parameter q can be chosen as $q \leq 4$ without suffering from essential round-off errors on conventional computers. Note that $U(x, y)$ can usually be saved (before compression) by means of about $8(q + 1)N$ coefficients.

In image processing applications we have to deal with non-smooth and noisy examples, so the main recommendation is an application of this approaches only if $q = 0$. In this case when $f(x, y) \in C^1$ we can provide an interpolation with a high precision (acceptable in applications) and also a high level compression with a relatively low memory consumption. In the future this approach can be strengthened for piecewise smooth in the square functions with a preliminary singularity detection procedure.

There are many differences in our approach compared with the schemes suggested in $[12] - [15]$. First of all, we do not use an approximate calculation of normal derivatives on the boundary by any scheme of finite differences on the interpolation grid. We find this values by solving corresponding system of linear equations (see section 2.2) and as a result we can deal with sparse grids for relatively low smooth images. But the energy of our residual function V is usually not so small as in the algorithm PHLST5 [15], and this energy sharply increases together with q. Moreover, in this paper we consider the compressing and zooming possibilities of our algorithm while in $[12]-[15]$ image processing results are demonstrated on real images without any information about memory saving or zooming possibilities.

In our future investigations we will continue comparison of these two methods, particularly we will compare them for RAW files of real images taking into consideration the analysis carried out in [12] - [15]. According to our present experiments we recommend to use our approach in relatively smooth parts of real images for lossless/lossy compression/decompression as well as for zooming using virtual pixels restored from accelerated interpolation.

References

[1] A. Krylov, *On approximate calculations*, Lectures delivered in 1906 (in Russian), St. Petersburg 1907, Tipolitography of Birkenfeld.

[2] C. Lanczos, *Discourse of Fourier series*, Oliver and Boyd, Edinburgh, 1966.

[3] Barkhudaryan A., Barkhudaryan R., and Poghosyan A., Asymptotic behavior of Eckhoff's method for Fourier Series Convergence Acceleration. Analysis in Theory and Applications, Volume 23, Number 3 (2007), 228-242.

[4] K. S. Eckhoff, *Accurate and efficient reconstruction of discontinuous functions from truncated series expansions*, Math. Comp. **61** (1993), 745-763.

[5] K. S. Eckhoff, *Accurate reconstructions of functions of finite regularity from truncated Fourier series expansions*, Math. Comp. **64** (1995), 671-690.

[6] K. S. Eckhoff, *On a high order numerical method for functions with singularities*, Math. Comp. **67** (1998), 1063-1087.

[7] K. S. Eckhoff and C. E. Wasberg, *On the numerical approximation of derivatives by a modified Fourier collocation method*, Technical Report No 99, Dept. of Mathematics, University of Bergen, Norway, 1995

[8] Anne Gelb and Eitan Tadmor.: Detection of edges in spectral data II. Nonlinear enhansment. SIAM j. Numer. Anal., 38, no.4, 1389-1408 (2000)

[9] A. Nersessian and A. Poghosyan, *Bernoulli method in multidimensional case*, Preprint No 20 Ar-00 (in Russian), Deposited in ArmNIINTI 09.03.00 (2000), 1-40.

[10] A. Nersessian and A. Poghosyan, *Asymptotic errors of accelerated two-dimensional trigonometric approximations*, Proceedings of the ISAAC fourth Conference on Analysis, Yerevan, Armenia (G. A. Barsegian, H. G. W. Begehr, H. G. Ghazaryan, A. Nersessian eds), Yerevan, 2004, 70-78.

[11] A.Bjorck and V.Pereyra, Solution of Vandermonde Systems of Equations, Math. Comp., 24 (1970), 893-903.

[12] Saito, N., Remy, J.-F.: A new local sine transform without overlaps: A combination of computational harmonic analysis and PDE. In: M.A. Unser, A. Aldroubi, A.F. Laine (eds.) Wavelets: Applications in Signal and Image Processing X. Proc. SPIE, vol. 5207, pp. 495506 (2003)

[13] Saito, N., Remy, J.-F.: The polyharmonic local sine transform: A new tool for local image analysis and synthesis without edge effect. Appl. Comput. Harmon. Anal. 20(1), 4173 (2006)

[14] Yamatani, K., Saito, N.: Improvement of DCT-based compression algorithms using Poissons equation. IEEE Trans. Image Process. 15(12), 36723689 (2006)

[15] Jucheng Zhao, Naoki Saito and Yi Wang.: PHLST5: A practical and Improved version of Polyharmonic Local Sine Transform. J. Math. Imaging Vis., 30, 23-41 (2008)

[16] Averbuch, A., Israeli, M., Vozovoi, L.: A fast Poisson solver of arbitrary order accuracy in rectangular regions. SIAM J. Sci. Comput. 19(3), 933-952 (1998)

[17] G. K. Wallace, The JPEG Still Picture Compression Standard. IEEE Transactions on Consumer Electronics, Vol. 38, No 1 (1992).

[18] M. W. Marcellin, M. J. Gormish, A. Bilgin, M. P. Boliek, An Overview of JPEG-2000. Proc. of IEEE Data Compression Conference, pp. 523-541 (2000).