ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

ELSEVIER

Journal of Computational and Applied Mathematics ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

www.elsevier.com/locate/cam

Hyperinterpolation on the square☆

Marco Caliari[b], Stefano De Marchi[a], Marco Vianello[b,*]

[a] Department of Computer Science, University of Verona, Italy

[b] Department of Pure and Applied Mathematics, University of Padova, Via Trieste 63, 35121 Padova, Italy

Received 12 September 2005; received in revised form 24 January 2006

Abstract

We show that hyperinterpolation at (near) minimal cubature points for the product Chebyshev measure, along with Xu compact formula for the corresponding reproducing kernel, provide a simple and powerful polynomial approximation formula in the uniform norm on the square. The Lebesgue constant of the hyperinterpolation operator grows like $\log^2$ of the degree, as that of quasi-optimal interpolation sets recently proposed in the literature. Moreover, we give an accurate implementation of the hyperinterpolation formula with linear cost in the number of cubature points, and we compare it with interpolation formulas at the same set of points.

MSC: 65D05; 65D32

Keywords: Hyperinterpolation; Square; Xu points; Minimal cubature formulas; Lebesgue constant

1. Introduction

Hyperinterpolation of multivariate continuous functions on compact subsets or manifolds, originally introduced by Sloan in [14], is a discretized orthogonal projection on polynomial subspaces, which provides an approximation method more general (in some sense) than interpolation. Its main success up to now has been given by the application to polynomial approximation on the sphere; see, e.g., [11,15,9]. Indeed, the effectiveness of hyperinterpolation in the uniform norm requires three basic ingredients, which are seldom at disposal all together: a "good" cubature formula (i.e., positive weights and high algebraic degree of exactness), a "good" (i.e., accurate and efficient) formula for the reproducing kernel, and "slow" increase of the Lebesgue constant (the operator norm).

These requirements can be easily recognized, by summarizing briefly the structure of hyperinterpolation. Let $\Omega \subset \mathbb{R}^d$ be a compact subset (or lower dimensional manifold), and μ a positive measure such that $\mu(\Omega) = 1$ (i.e., a normalized positive and finite measure on Ω). For every function $f \in C(\Omega)$ the μ-orthogonal projection of f on $\Pi_n^d(\Omega)$ (the subspace of d-variate polynomials of degree $\leqslant n$ restricted to Ω) can be written as

$$S_n f(x) = \int_\Omega K_n(x, y) f(y)\, d\mu(y), \quad x \in \Omega, \quad \text{with } S_n p = p \text{ for } p \in \Pi_n^d(\Omega), \tag{1}$$

☆ Work supported by the ex-60% funds of the Universities of Padova and Verona, and by the GNCS-INdAM.

* Corresponding author. Tel.: +39 49 8271370; fax: +39 49 8271499.

E-mail address: marcov@math.unipd.it (M. Vianello).

0377-0427/$ - see front matter © 2006 Published by Elsevier B.V.

doi:10.1016/j.cam.2006.10.058

where the so-called reproducing kernel K_n is defined by

$$K_n(x,y)=\sum_{s=0}^{n}\mathbf{P}_s^t(x)\mathbf{P}_s(y),\quad x,y\in\mathbb{R}^d, \tag{2}$$

the sequence of polynomial arrays $(\mathbf{P}_0,\ldots,\mathbf{P}_n)$ being any μ-orthonormal basis of $\Pi_n^d(\Omega)$; cf. [8, Section 3.5].

Now, given a cubature formula for μ with $N=N(n)$ nodes $\xi\in X_N\subset\Omega$ and positive weights $\{w_\xi\}$, which is exact for polynomials of degree $\leqslant 2n$,

$$\int_\Omega p(x)\,\mathrm{d}\mu=\sum_{\xi\in X_N}w_\xi p(\xi),\quad \forall p\in\Pi_{2n}^d(\Omega), \tag{3}$$

we obtain from (1) the polynomial approximation of degree n

$$f(x)\approx L_nf(x)=\sum_{\xi\in X_N}w_\xi K_n(x,\xi)f(\xi)\quad \text{(hyperinterpolation)}. \tag{4}$$

It is known that necessarily $N\geqslant\dim(\Pi_n^d(\Omega))$, and that (4) is a polynomial interpolation at X_N whenever the equality holds; cf. [14,9].

The hyperinterpolation error in the uniform norm, due to the exactness on $\Pi_{2n}^d(\Omega)$, can be easily estimated as

$$\|f-L_nf\|_\infty\leqslant(1+\Lambda_n)E_n(f),\quad \Lambda_n=\|L_n\|=\max_{x\in\Omega}\left\{\sum_{\xi\in X_N}w_\xi|K_n(x,\xi)|\right\}, \tag{5}$$

where Λ_n is the operator norm of $L_n:(C(\Omega),\|\cdot\|_\infty)\to(\Pi_n^d(\Omega),\|\cdot\|_\infty)$, usually termed the "Lebesgue constant" in the interpolation framework.

2. Hyperinterpolation at Xu points on the square

In the paper [18], Xu introduced a set of Chebyshev-like points in the square $[-1,1]^2$, which generate a (near) minimal degree cubature for the normalized product Chebyshev measure,

$$\mathrm{d}\mu=\frac{1}{\pi^2}\frac{\mathrm{d}x_1\,\mathrm{d}x_2}{\sqrt{1-x_1^2}\sqrt{1-x_2^2}},\quad \Omega=[-1,1]^2. \tag{6}$$

For even degrees such points and the corresponding minimal cubature appeared already in [10]; see also [6,5]. In addition, Xu proved that these points are also suitable for constructing polynomial interpolation, in a polynomial subspace $\mathscr{V}_n$, $\Pi_{n-1}^2\subset\mathscr{V}_n\subset\Pi_n^2$.

Interpolation at the Xu points, recently studied thoroughly in [1,2], exhibits some very appealing features: there is a compact formula for the Lagrange polynomials, which must be stabilized but nevertheless leads to linear complexity in the evaluation of the interpolant; the Lebesgue constant of the interpolation is $\mathcal{O}(\log^2 n)$, n being the degree, i.e., the polynomial approximation is "quasi-optimal" (cf. [3]).

Here we show that hyperinterpolation at the Xu points, even though is not interpolant, shares the same good computational features of Xu-like interpolation. In what follows we restrict, for simplicity's sake, to odd degrees n: the case of even degrees can be treated in a similar fashion, cf. [18].

Considering the $n+2$ Chebyshev–Lobatto points on the interval $[-1,1]$

$$z_k=z_{k,n+1}=\cos\frac{k\pi}{n+1},\quad k=0,\ldots,n+1,\quad n=2m-1,\ m\geqslant 1, \tag{7}$$

the Xu points on the square Ω are defined as the two dimensional Chebyshev-like set

$$X_N=A\cup B,\quad \text{of cardinality } N=(n+1)(n+3)/2,$$

where

$$A = \{(z_{2i}, z_{2j+1}),\ 0 \leqslant i \leqslant m,\ 0 \leqslant j \leqslant m-1\},$$
$$B = \{(z_{2i+1}, z_{2j}),\ 0 \leqslant i \leqslant m-1,\ 0 \leqslant j \leqslant m\}. \tag{8}$$

These points generate a minimal cubature formula, that is

$$\int_\Omega p(x)\,\mathrm{d}\mu = \sum_{\xi \in X_N} w_\xi p(\xi), \quad \forall p \in \Pi^2_{2n+1}, \tag{9}$$

where the weights are simply $w_\xi = 2(n+1)^{-2}$ for $\xi \in X_N \cap \mathring{\Omega}$ (interior points), $(n+1)^{-2}$ for $\xi \in X_N \cap \partial\Omega$ (boundary points); cf. [10,18]. Hence, in view of (3) we can construct the hyperinterpolation formula (4), which is not interpolant, since $N = (n+1)(n+3)/2 > \dim(\Pi^2_n) = (n+1)(n+2)/2$. In any case, its uniform approximation error can be estimated as in (5).

Moreover, the reproducing kernel $K_n(x, y)$ has an explicit and compact trigonometric representation (obtained by Xu in [17])

$$K_n(x, y) = D_n(\theta_1 + \phi_1, \theta_2 + \phi_2) + D_n(\theta_1 + \phi_1, \theta_2 - \phi_2)$$
$$+ D_n(\theta_1 - \phi_1, \theta_2 + \phi_2) + D_n(\theta_1 - \phi_1, \theta_2 - \phi_2), \tag{10}$$

where $x = (\cos\theta_1, \cos\theta_2)$, $y = (\cos\phi_1, \cos\phi_2)$, and the bivariate function D_n is defined for every $n > 0$ by

$$D_n(\alpha, \beta) = \frac{1}{2}\,\frac{\cos((n+1/2)\alpha)\cos(\alpha/2) - \cos((n+1/2)\beta)\cos(\beta/2)}{\cos\alpha - \cos\beta}. \tag{11}$$

(*note*: the definitions of K_n and D_n have been changed w.r.t. [18], in such a way that the index is exactly the degree of hyperinterpolation). This representation allows an efficient implementation (after some nontrivial stabilization), and the possibility of estimating analytically the Lebesgue constant, as we shall see in the following subsections.

2.1. Estimating the Lebesgue constant

First, it is convenient to rewrite $D_n(\alpha, \beta)$. By simple trigonometric manipulations, we obtain

$$D_n(\alpha, \beta) = \tfrac{1}{4}(U_n(\cos\phi)U_n(\cos\psi) + U_{n-1}(\cos\phi)U_{n-1}(\cos\psi)), \tag{12}$$

where $\phi = (\alpha - \beta)/2$, $\psi = (\alpha + \beta)/2$, and U_n denotes the usual Chebyshev polynomial of the second kind. This rewriting is also very useful for stabilizing the computation of D_n, as it is outlined in the next subsection.

With (12) at hand, it comes easy to bound the Lebesgue constant of hyperinterpolation linearly with N, the number of Xu points. Indeed, from the well-known bound for Chebyshev polynomials of the second kind $|U_n(\cos\theta)| \leqslant n+1$, we get immediately $w_\xi |K_n(x, \xi)| \leqslant 2((n+1)^2 + n^2)/(n+1)^2 \leqslant 4$, for any $x \in \Omega$, $\xi \in X_N$. Then, from (5) we get the estimate $\Lambda_n \leqslant 4N \sim 2n^2$. This already shows that hyperinterpolation at the Xu points is not a bad choice for approximation in the uniform norm. However, the latter is a substantial overestimate of the actual Lebesgue constant. In fact, we can prove the following:

Theorem 1. *The Lebesgue constant of hyperinterpolation at the Xu points can be bounded as*

$$\Lambda_n \leqslant 8\left(\frac{2}{\pi}\log(n+1) + 5\right)^2 + 5\left(\frac{2}{\pi}\log(n+1) + 5\right) + 2. \tag{13}$$

Proof. We give only the first step, because then the proof is very close to that in [2]. By using the trigonometric identity $U_{n-1}(\cos\theta) = U_n(\cos\theta)\cos\theta - \cos(n+1)\theta$, from the representation (12) we get immediately the estimate

Please cite this article as: M. Caliari, et al., Hyperinterpolation on the square, J. Comput. Appl. Math. (2006), doi: 10.1016/j.cam.2006.10.058

Table 1
Average use percentage η of the recurrence relation for U_n, in evaluating the hyperinterpolation polynomial at degree $n=19$, up to 10^6 uniform random points

# Of random points	Percentage η	
	$\varepsilon = 0.01$	$\varepsilon = 0.1$
1.0E + 02	0.75	6.25
1.0E + 03	0.69	6.27
1.0E + 04	0.63	6.34
1.0E + 05	0.64	6.36
1.0E + 06	0.64	6.37

Table 2
Average use percentage η of the recurrence relation for U_n, in evaluating the hyperinterpolation polynomial at different degrees

Degree	Percentage η	
n	$\varepsilon = 0.01$	$\varepsilon = 0.1$
19	0.64	6.37
39	0.64	6.37
79	0.64	6.37

$|D_n(\alpha, \beta)| \leqslant \frac{1}{2}|U_n(\cos\phi)U_n(\cos\psi)| + \frac{1}{4}(|U_n(\cos\phi)| + |U_n(\cos\psi)|) + \frac{1}{4}$. We can now proceed following the lines of [2], where the peculiar structure of the Xu points is nontrivially exploited, obtaining (13). □

2.2. *Implementing hyperinterpolation*

Rearranging (11) in the case that $\cos(\alpha) = \cos(\beta)$, allows us to give a version of the hyperinterpolation formula with pointwise evaluation cost $\mathcal{O}(N)$. However, the hyperinterpolant at the Xu points evaluated via (11) (which is like a first divided difference) turns out to be severely ill-conditioned, and must be stabilized.

To this purpose it is convenient to use the rewriting (12) of (11), and to compute the polynomials U_n by their three-term recurrence relation. The evaluation of $D_n(\alpha, \beta)$ becomes stable, paying the price of a computational cost $\mathcal{O}(n)$ instead of $\mathcal{O}(1)$. Then, it is not difficult to see that the dominant term in the final complexity for the pointwise evaluation of the hyperinterpolation polynomial $L_n f(x)$, is $(2n \times 4)N \sim 8\sqrt{2}N^{3/2} \sim 4n^3$ flops.

An effective way to reduce the computational cost of the stabilized formula (12), still preserving high accuracy, is to compute the Chebyshev polynomials of the second kind U_n by the three-term recurrence relation only when the trigonometric representation $U_n(\cos\theta) = \sin(n+1)\theta/\sin\theta$ (whose cost is $\mathcal{O}(1)$ in n and θ) is ill-conditioned, say when $|\theta - k\pi| \leqslant \varepsilon$ for a "small" value of ε. In this case, it is important to estimate the average use percentage of the recurrence relation in evaluating the hyperinterpolation polynomial.

As in [1] concerning interpolation at the Xu points, we can resort to some probabilistic considerations. Indeed, taking random, uniformly distributed evaluation points in the square, such a percentage becomes a random variable (function of a uniform random variable), whose expectation, say η, depends on the threshold ε but not on the degree n. This is clearly seen in Tables 1 and 2, where it is shown that the averages up to one million random points converge to a value, that does not depend on the degree n.

Now, the evaluation of $K_n(x, \xi)$ using only the trigonometric representation of $U_n(\cos\theta)$ costs about $6 \times 4 = 24$ evaluations of the sine function. Denoting by $c_{\sin}$ the average evaluation cost of the sine function (which actually depends on its internal implementation), the average complexity for the evaluation of the hyperinterpolation polynomial $L_n f(x)$ at the Xu points is of the order of

$$C(n, \varepsilon) := 8n\tau N + 24c_{\sin}(1-\tau)N \sim 4n^3\tau + 12c_{\sin}(1-\tau)n^2 \text{ flops}, \tag{14}$$

where $\tau = \eta/100$. Using the experimental value $c_{\sin} = 10$ (obtained with GNU Fortran, but consistent with usual implementations, cf. [16]), we can conclude that, for $\varepsilon \leqslant 0.01$ (i.e., $\tau \leqslant 0.0064$), the size of the ratio $C(n, \varepsilon)/N$ remains constant up to degrees of the order of hundreds, that is in practical applications the computational cost can be considered linear in the number N of Xu points.

2.3. Comparison with Xu-like interpolation

It is worth comparing interpolation with hyperinterpolation at the same set of Xu points. Given $X_N = A \cup B$ defined as in (8), we have two choices. On one hand, we can use Xu interpolation formula [18,1], which gives a polynomial of degree $n + 1$, say $p_{n+1}^{\mathrm{Xu}} \in \mathscr{V}_{n+1}$, where $\Pi_n^2 \subset \mathscr{V}_{n+1} \subset \Pi_{n+1}^2$. As shown in [1], the dominant cost in the pointwise evaluation of such a polynomial is $32c_{\sin}N$ flops (since both K_n and K_{n+1} are involved in the definition of the Lagrange polynomials), where $c_{\sin}$ represents the average evaluation cost of the sine function. The uniform approximation error can be estimated as $\|f - p_{n+1}^{\mathrm{Xu}}\|_\infty \leqslant (1 + \Lambda_{n+1}^{\mathrm{Xu}}) \inf_{p\in\mathscr{V}_{n+1}} \|f - p\|_\infty \leqslant (1 + \Lambda_{n+1}^{\mathrm{Xu}}) E_n(f)$, where $\Lambda_{n+1}^{\mathrm{Xu}}$ denotes the Lebesgue constant of Xu-like interpolation. Then, using the estimate of $\Lambda_{n+1}^{\mathrm{Xu}}$ given in [2], we get

$$\|f - p_{n+1}^{\mathrm{Xu}}\|_\infty \leqslant (8a_n^2 + 5) E_n(f) \quad \text{(interpolation)}, \tag{15}$$

where we have defined

$$a_n = \frac{2}{\pi} \log(n + 1) + 5. \tag{16}$$

On the other hand, hyperinterpolation at X_N gives a polynomial of degree n, which is not interpolant. The dominant cost in its pointwise evaluation is $24c_{\sin}N$ flops, and the uniform approximation error is estimated via (5) and (13), i.e.,

$$\|f - L_n f\|_\infty \leqslant (8a_n^2 + 5a_n + 3) E_n(f) \quad \text{(hyperinterpolation)}. \tag{17}$$

In view of the error estimates above we can expect, in practice, close approximation errors by the two methods, as is confirmed by the numerical tests of the next section.

3. Numerical tests

In order to show the efficiency and robustness of our implementation of hyperinterpolation at the Xu points [4], we made some comparisons with Xu-like interpolation (as implemented in [1,4]), and with the `MPI package` by Sauer, one of the most effective implementations of multivariate polynomial interpolation (via finite differences and the notion of blockwise interpolation, cf. [12,13]).

We compared the CPU times necessary to build and evaluate the interpolant, as well as the approximation errors, on a grid of 100×100 control points in the reference square, with hyperinterpolation at Xu points (HYP-XU), and interpolation at the same points (MPI, and Xu-like interpolation INT-XU). Clearly, both INT-XU and HYP-XU can be extended to arbitrary rectangles by an obvious change of variables. The tests were performed on a AMD Athlon 2800+ processor machine. Our numerical results on several test functions with different degree of regularity, some of which are collected in Tables 3 and 4, show that:

- MPI works quite well for small degrees, but becomes useless for higher degrees, even when one tries to stabilize it by a Leja-like reordering of the interpolation points (cf. [1,7]);
- both INT-XU and HYP-XU are accurate and robust, and can suitably manage very high degrees (up to the order of the hundreds, without problems);
- in practice, HYP-XU approximates like INT-XU, but has slightly lower computational cost.

From the observations above, we can draw the conclusion that hyperinterpolation at Xu points might be considered a valid alternative to interpolation, for polynomial approximation of bivariate functions that can be sampled without restrictions on rectangles.

Please cite this article as: M. Caliari, et al., Hyperinterpolation on the square, J. Comput. Appl. Math. (2006), doi: 10.1016/j.cam.2006.10.058

Table 3
CPU times (in seconds) and approximation errors on $[0,1]^2$ for the classical Franke test function $f(x_1,x_2)=\frac{3}{4}e^{-1/4((9x_1-2)^2+(9x_2-2)^2)}+\frac{3}{4}e^{-1/49(9x_1+1)^2-1/10(9x_2+1)}+\frac{1}{2}e^{-1/4((9x_1-7)^2+(9x_2-3)^2)}-\frac{1}{5}e^{-((9x_1-4)^2+(9x_2-7)^2)}$, using $N=(n+1)(n+3)/2$ Xu points with interpolation of degree $n+1$ (MPI, stabilized MPI, Xu interpolation formula) and hyperinterpolation of degree n

n	19	29	39	49	59
N	220	480	840	1300	1860
MPI	0.6	Unsolv.	Unsolv.	Unsolv.	Unsolv.
	3.8E−02	* * *	* * *	* * *	* * *
MPI-Leja	0.6	4.3	21.0	75.6	Unsolv.
	6.4E−03	3.5E−04	1.1E−04	2.0E−03	* * *
INT-XU	2.1	5.2	10.3	17.8	28.4
	7.3E−03	3.6E−04	3.1E−06	1.8E−08	2.5E−11
HYP-XU	1.9	4.7	9.5	16.6	26.5
	7.3E−03	3.6E−04	3.2E−06	1.8E−08	3.0E−11

Table 4
As in Table 3 for the function $f(x_1,x_2)=(x_1^2+x_2^2)^{5/2}$ on $[-1,1]^2$

n	19	29	39	49	59
MPI-Leja	0.6	4.3	20.8	74.8	Unsolv.
	1.1E−04	1.3E−05	1.4E−05	6.8E−04	* * *
INT-XU	2.1	5.2	10.3	17.8	28.4
	1.1E−04	1.3E−05	3.1E−06	1.0E−06	4.0E−07
HYP-XU	1.9	4.7	9.5	16.6	26.5
	1.1E−04	1.3E−05	3.1E−06	1.0E−06	4.0E−07

Acknowledgments

We are grateful to Yuan Xu, for having introduced us in the fascinating field of multivariate orthogonal polynomials. We also wish to thank Tomas Sauer, who kindly provided us the Multivariate Polynomial Interpolation package.

References

[1] L. Bos, M. Caliari, S. De Marchi, M. Vianello, A numerical study of the Xu polynomial interpolation formula, Computing 76 (2005) 311–324.
[2] L. Bos, S. De Marchi, M. Vianello, On the Lebesgue constant for the Xu interpolation formula, J. Approx. Theory 141 (2006) 134–141.
[3] M. Caliari, S. De Marchi, M. Vianello, Bivariate polynomial interpolation on the square at new nodal sets, Appl. Math. Comput. 165 (2005) 261–274.
[4] M. Caliari, S. De Marchi, M. Vianello, Bivariate polynomial interpolation and hyperinterpolation at Xu points, Fortran codes available at www.math.unipd.it/~marcov/software.html.
[5] R. Cools, I.P. Mysovskikh, H.J. Schmid, Cubature formulae and orthogonal polynomials, Numerical Analysis 2000, vol. V, Quadrature and Orthogonal Polynomials, J. Comput. Appl. Math. 127 (2001) 121–152.
[6] R. Cools, H.J. Schmid, Minimal cubature formulae of degree $2k-1$ for two classical functionals, Computing 43 (1989) 141–157.
[7] S. De Marchi, On Leja sequences: some results and applications, Appl. Math. Comput. 152 (2004) 621–647.
[8] C.F. Dunkl, Y. Xu, Orthogonal polynomials of several variables, Encyclopedia of Mathematics and its Applications, vol. 81, Cambridge University Press, Cambridge, 2001.
[9] K. Hesse, I.H. Sloan, Hyperinterpolation on the sphere, UNSW School of Mathematics, preprint AMR05/23, 2005.
[10] C.R. Morrow, T.N.L. Patterson, Construction of algebraic cubature rules using polynomial ideal theory, SIAM J. Numer. Anal. 15 (1978) 953–976.
[11] M. Reimer, Multivariate polynomial approximation, International Series of Numerical Mathematics, vol. 144, Birkhäuser, Basel, 2003.
[12] T. Sauer, Computational aspects of multivariate polynomial interpolation, Adv. Comput. Math. 3 (1995) 219–238.
[13] T. Sauer, Y. Xu, On multivariate Lagrange interpolation, Math. Comput. 64 (1995) 1147–1170.
[14] I.H. Sloan, Polynomial interpolation and hyperinterpolation over general regions, J. Approx. Theory 83 (1995) 238–254.
[15] I.H. Sloan, R.S. Womersley, Constructive polynomial approximation on the sphere, J. Approx. Theory 103 (2000) 91–118.
[16] P.T.P. Tang, Some software implementations of the functions sine and cosine, ANL Report 90/3, Argonne National Laboratory, April 1990.
[17] Y. Xu, Christoffel functions and Fourier series for multivariate orthogonal polynomials, J. Approx. Theory 82 (1995) 205–239.
[18] Y. Xu, Lagrange interpolation on Chebyshev points of two variables, J. Approx. Theory 87 (1996) 220–238.

Please cite this article as: M. Caliari, et al., Hyperinterpolation on the square, J. Comput. Appl. Math. (2006), doi: 10.1016/j.cam.2006.10.058