Padua points: genesis, theory, computation and applications *

Stefano De Marchi

Department of Mathematics
University of Padova

April 2, 2014

* Joint work with L. Bos (Verona), M. Caliari (Verona), A. Sommariva and M. Vianello (Padua), Y. Xu (Eugene)

Outline

1. Motivations
2. From Dubiner metric to Padua points
3. Padua points: properties
4. Interpolation: formula and computational issues
5. Cubature: formula and computational issues
6. Examples and numerical tests
7. Applications

Motivations

- Well-distributed nodes: there exist various nodal sets for polynomial interpolation of even degree n in the square $\Omega = [-1,1]^2$ (C.DeM.V., AMC04), which turned out to be equidistributed w.r.t. Dubiner metric (D., JAM95) and which show optimal Lebesgue constant growth.
- Efficient interpolant evaluation: the interpolant should be constructed without solving the Vandermonde system whose complexity is $\mathcal{O}(N^3)$, $N = \binom{n+2}{2}$ for each pointwise evaluation. We look for compact formulae.
- Efficient cubature: in particular computation of cubature weights for non-tensorial cubature formulae.

Main references

1. M. Caliari, S. De Marchi and M. Vianello: *Bivariate polynomial interpolation on the square at new nodal sets*, Applied Math. Comput. vol. 165/2, pp. 261-274 (2005).
2. L. Bos, S. De Marchi, M. Caliari, M. Vianello and Y. Xu: *Bivariate Lagrange interpolation at the Padua points: the generating curve approach*, J. Approx. Theory 143 (2006), 15–25.
3. L. Bos, S. De Marchi, M. Vianello and Y. Xu: *Bivariate Lagrange interpolation at the Padua points: the ideal theory approach*, Numer. Math., 108(1) (2007), 43-57.
4. M. Caliari, S. De Marchi, and M. Vianello: *Bivariate Lagrange interpolation at the Padua points: computational aspects*, J. Comput. Appl. Math., Vol. 221 (2008), 284-292.
5. M. Caliari, S. De Marchi and M. Vianello: *Algorithm 886: Padua2D: Lagrange Interpolation at Padua Points on Bivariate Domains*, ACM Trans. Math. Software, Vol. 35(3), Article 21, 11 pages (2008).
6. L. Bos, S. De Marchi and S. Waldron: *On the Vandermonde Determinant of Padua-like Points* (on Open Problems section), Dolomites Res. Notes on Approx. 2(2009), 1–15.
7. M. Caliari, S. De Marchi, A. Sommariva and M. Vianello: *Padua2DM: fast interpolation and cubature at Padua points in Matlab/Octave*, Numer. Algorithms 56(1) (2011), 45–60.

The Dubiner metric

The Dubiner metric in the 1D:

$$\mu_{[-1,1]}(x,y) = |\arccos(x) - \arccos(y)|, \ \forall x,y \in [-1,1] \ .$$

By using the Van der Corput-Schaake inequality (1935) for trig. polys. $T(\theta)$ of degree m and $|T(\theta)| \le 1$,

$$|T'(\theta)| \le m\sqrt{1 - T^2(\theta)} \, .$$

$$\mu_{[-1,1]}(x,y) := \sup_{\|P\|_{\infty,[-1,1]} \le 1} \frac{1}{m} |\arccos(P(x)) - \arccos(P(y))| \, ,$$

with $P \in \mathbb{P}_n([-1,1])$.
This metric generalizes to compact sets $\Omega \subset \mathbb{R}^d$, $d > 1$:

$$\mu_\Omega(\mathbf{x},\mathbf{y}) := \sup_{\|P\|_{\infty,\Omega} \le 1} \frac{1}{m} |\arccos(P(\mathbf{x})) - \arccos(P(\mathbf{y}))| \, .$$

The Dubiner metric

Conjecture(C.DeM.V.AMC04):

Nearly optimal interpolation points on a compact Ω are asymptotically equidistributed w.r.t. the Dubiner metric on Ω.

Once we know the Dubiner metric on a compact Ω, we have at least a method for producing "good" points.
For $d = 2$, let $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2)$

- Dubiner metric on the square, $[-1, 1]^2$:

$$\max\{|\arccos(x_1) - \arccos(y_1)|, |\arccos(x_2) - \arccos(y_2)|\} \ ;$$

- Dubiner metric on the disk, $|\mathbf{x}| \le 1$:

$$\left|\arccos\left(x_1y_1 + x_2y_2 + \sqrt{1 - x_1^2 - x_2^2}\sqrt{1 - y_1^2 - y_2^2}\right)\right| \ ;$$

The Dubiner metric

Conjecture(C.DeM.V.AMC04):

Nearly optimal interpolation points on a compact Ω are asymptotically equidistributed w.r.t. the Dubiner metric on Ω.

Once we know the Dubiner metric on a compact Ω, we have at least a method for producing "good" points.
For $d = 2$, let $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2)$

- Dubiner metric on the square, $[-1, 1]^2$:

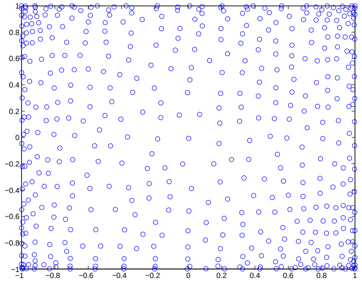
$$\max\{|\arccos(x_1) - \arccos(y_1)|, |\arccos(x_2) - \arccos(y_2)|\} \; ;$$

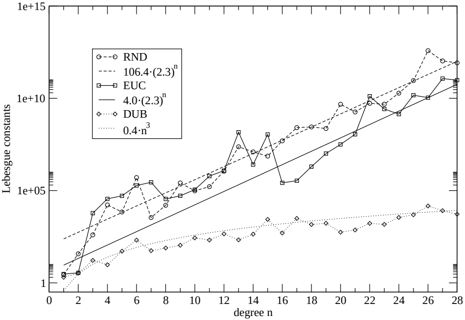
- Dubiner metric on the disk, $|\mathbf{x}| \le 1$:

$$\left|\arccos\left(x_1 y_1 + x_2 y_2 + \sqrt{1 - x_1^2 - x_2^2}\sqrt{1 - y_1^2 - y_2^2}\right)\right| \; ;$$

Dubiner points and Lebesgue constant

496 Dubiner nodes (i.e. deg. $n = 30$) and the comparison of Lebesgue constants for Random (RND), Euclidean (EUC) and Dubiner (DUB) points.





Euclidean pts, are Leja-like points, given by $\max_{\mathbf{x}\in\Omega} \min_{\mathbf{y}\in X_n} \|\mathbf{x}-\mathbf{y}\|_2$.

Morrow-Patterson points

- Let n be a positive even integer. The Morrow-Patterson points (MP) (cf. M.P. SIAM JNA 78) are the points

$$x_m = \cos\left(\frac{m\pi}{n+2}\right), \quad y_k = \begin{cases} \cos\left(\frac{2k\pi}{n+3}\right) & \text{if } m \text{ odd} \\ \cos\left(\frac{(2k-1)\pi}{n+3}\right) & \text{if } m \text{ even} \end{cases}$$

$1 \le m \le n+1$, $1 \le k \le n/2+1$. Note: they are $N = \binom{n+2}{2}$.

Extended Morrow-Patterson points

The Extended Morrow-Patterson points (EMP) (C.DeM.V. AMC 05) are the points

$$x_m^{EMP} = \frac{1}{\alpha_n} x_m^{MP}, \quad y_k^{EMP} = \frac{1}{\beta_n} y_k^{MP}$$

$\alpha_n = \cos(\pi/(n+2))$, $\beta_n = \cos(\pi/(n+3))$.

Note: the MP and the EMP points are equally distributed w.r.t. Dubiner metric on the square $[-1, 1]^2$ and unisolvent for polynomial interpolation of degree n on the square.

Padua points

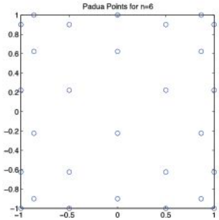
The Padua points (PD) can be defined as follows (C.DeM.V. AMC 05):

$$x_m^{PD} = \cos\left(\frac{(m-1)\pi}{n}\right), \quad y_k^{PD} = \begin{cases} \cos\left(\frac{(2k-1)\pi}{n+1}\right) & \text{if } m \text{ odd} \\ \cos\left(\frac{2(k-1)\pi}{n+1}\right) & \text{if } m \text{ even} \end{cases}$$

$$1 \le m \le n+1,\ 1 \le k \le n/2+1,\ N = \binom{n+2}{2}.$$

Some properties

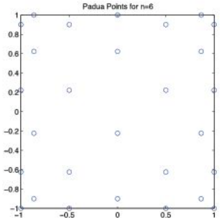
- The PD points are equispaced w.r.t. Dubiner metric on $[-1, 1]^2$.
- They are modified Morrow-Patterson points discovered in Padua in 2003 by B.DeM.V.&W. Actually the interior points are the MP points of degree $n-2$ while the boundary points are "natural" points of the grid.



- There are 4 families of PD pts: take rotations of 90 degrees, clockwise for even degrees and counterclockwise for odd degrees.

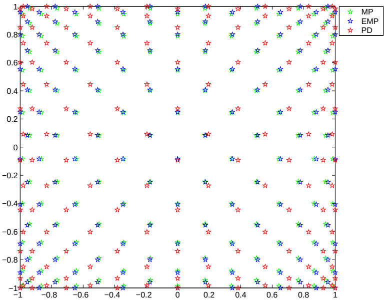
Some properties

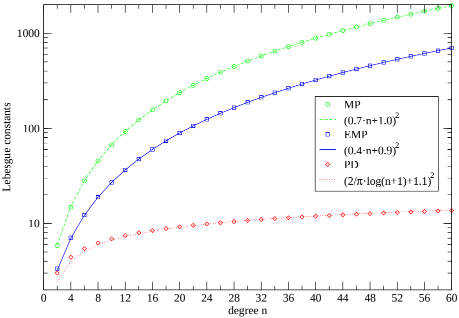
- The PD points are equispaced w.r.t. Dubiner metric on $[-1,1]^2$.
- They are modified Morrow-Patterson points discovered in Padua in 2003 by B.DeM.V.&W. Actually the interior points are the MP points of degree $n-2$ while the boundary points are "natural" points of the grid.



- There are 4 families of PD pts: take rotations of 90 degrees, clockwise for even degrees and counterclockwise for odd degrees.

Graphs of MP, EMP, PD pts and their Lebesgue constants





Left: the graphs of MP, EMP, PD for $n = 8$. **Right**: the growth of the corresponding Lebesgue constants.

Bivariate interpolation problem and Padua points

Let $\mathbb{P}_n^2$ be the space of bivariate polynomials of total degree $\leq n$.

Question: is there a set $\Xi \subset [-1,1]^2$ of points such that:

- $\mathrm{card}(\Xi) = \dim(\mathbb{P}_n^2) = \frac{(n+1)(n+2)}{2}$;
- the problem of finding the interpolation polynomial on Ξ of degree n is unisolvent;
- the Lebesgue constant Λ_n behaves like $\log^2 n$ for $n \to \infty$.

Answer: yes, it is the set $\Xi = \mathrm{Pad}_n$ of Padua points.

Bivariate interpolation problem and Padua points

Let $\mathbb{P}_n^2$ be the space of bivariate polynomials of total degree $\leq n$.
Question: is there a set $\Xi \subset [-1, 1]^2$ of points such that:

- $\mathrm{card}(\Xi) = \dim(\mathbb{P}_n^2) = \frac{(n+1)(n+2)}{2}$;
- the problem of finding the interpolation polynomial on Ξ of degree n is unisolvent;
- the Lebesgue constant Λ_n behaves like $\log^2 n$ for $n \to \infty$.

Answer: yes, it is the set $\Xi = \mathrm{Pad}_n$ of Padua points.

Bivariate interpolation problem and Padua points

Let $\mathbb{P}_n^2$ be the space of bivariate polynomials of total degree $\leq n$.
Question: is there a set $\Xi \subset [-1,1]^2$ of points such that:

- $\mathrm{card}(\Xi) = \dim(\mathbb{P}_n^2) = \frac{(n+1)(n+2)}{2}$;
- the problem of finding the interpolation polynomial on Ξ of degree n is unisolvent;
- the Lebesgue constant Λ_n behaves like $\log^2 n$ for $n \to \infty$.

Answer: yes, it is the set $\Xi = \mathrm{Pad}_n$ of Padua points.

Bivariate interpolation problem and Padua points

Let $\mathbb{P}_n^2$ be the space of bivariate polynomials of total degree $\leq n$.
Question: is there a set $\Xi \subset [-1,1]^2$ of points such that:

- $\mathrm{card}(\Xi) = \dim(\mathbb{P}_n^2) = \frac{(n+1)(n+2)}{2}$;
- the problem of finding the interpolation polynomial on Ξ of degree n is unisolvent;
- the Lebesgue constant Λ_n behaves like $\log^2 n$ for $n \to \infty$.

Answer: yes, it is the set $\Xi = \mathrm{Pad}_n$ of Padua points.

Bivariate interpolation problem and Padua points

Let $\mathbb{P}_n^2$ be the space of bivariate polynomials of total degree $\leq n$.
Question: is there a set $\Xi \subset [-1,1]^2$ of points such that:

- $\mathrm{card}(\Xi) = \dim(\mathbb{P}_n^2) = \frac{(n+1)(n+2)}{2}$;
- the problem of finding the interpolation polynomial on Ξ of degree n is unisolvent;
- the Lebesgue constant Λ_n behaves like $\log^2 n$ for $n \to \infty$.

Answer: yes, it is the set $\Xi = \mathrm{Pad}_n$ of Padua points.

Bivariate interpolation problem and Padua points

Let $\mathbb{P}_n^2$ be the space of bivariate polynomials of total degree $\leq n$.
Question: is there a set $\Xi \subset [-1,1]^2$ of points such that:

- $\mathrm{card}(\Xi) = \dim(\mathbb{P}_n^2) = \frac{(n+1)(n+2)}{2}$;
- the problem of finding the interpolation polynomial on Ξ of degree n is unisolvent;
- the Lebesgue constant Λ_n behaves like $\log^2 n$ for $n \to \infty$.

Answer: yes, it is the set $\Xi = \mathrm{Pad}_n$ of Padua points.

Padua points

Let us consider $n+1$ Chebyshev–Lobatto points on $[-1,1]$

$$C_{n+1} = \left\{ z_j^n = \cos\left(\frac{(j-1)\pi}{n}\right),\ j = 1,\ldots,n+1 \right\}$$

and the two subsets of points with odd or even indexes

$$C_{n+1}^{\mathrm{O}} = \left\{ z_j^n,\ j = 1,\ldots,n+1,\ j \text{ odd} \right\}$$
$$C_{n+1}^{\mathrm{E}} = \left\{ z_j^n,\ j = 1,\ldots,n+1,\ j \text{ even} \right\}$$

Then, the Padua points are the set

$$\mathrm{Pad}_n = C_{n+1}^{\mathrm{O}} \times C_{n+2}^{\mathrm{E}} \cup C_{n+1}^{\mathrm{E}} \times C_{n+2}^{\mathrm{O}} \subset C_{n+1} \times C_{n+2}$$

Padua points

Let us consider $n+1$ Chebyshev–Lobatto points on $[-1,1]$

$$C_{n+1} = \left\{ z_j^n = \cos\left(\frac{(j-1)\pi}{n}\right),\ j = 1,\ldots,n+1 \right\}$$

and the two subsets of points with odd or even indexes

$$C_{n+1}^{\mathrm{O}} = \left\{ z_j^n,\ j = 1,\ldots,n+1,\ j \text{ odd} \right\}$$
$$C_{n+1}^{\mathrm{E}} = \left\{ z_j^n,\ j = 1,\ldots,n+1,\ j \text{ even} \right\}$$

Then, the Padua points are the set

$$\mathrm{Pad}_n = C_{n+1}^{\mathrm{O}} \times C_{n+2}^{\mathrm{E}} \cup C_{n+1}^{\mathrm{E}} \times C_{n+2}^{\mathrm{O}} \subset C_{n+1} \times C_{n+2}$$

Padua points

Let us consider $n+1$ Chebyshev–Lobatto points on $[-1,1]$

$$C_{n+1} = \left\{ z_j^n = \cos\left(\frac{(j-1)\pi}{n}\right),\ j = 1,\ldots,n+1 \right\}$$

and the two subsets of points with odd or even indexes

$$C_{n+1}^{\mathrm{O}} = \left\{ z_j^n,\ j = 1,\ldots,n+1,\ j \text{ odd} \right\}$$
$$C_{n+1}^{\mathrm{E}} = \left\{ z_j^n,\ j = 1,\ldots,n+1,\ j \text{ even} \right\}$$

Then, the Padua points are the set

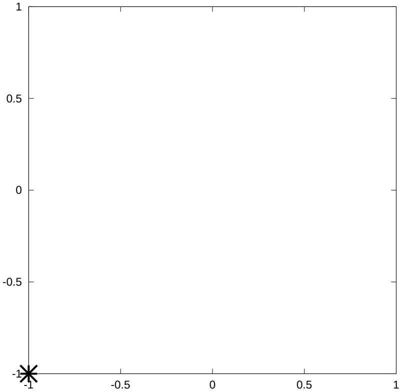
$$\mathrm{Pad}_n = C_{n+1}^{\mathrm{O}} \times C_{n+2}^{\mathrm{E}} \cup C_{n+1}^{\mathrm{E}} \times C_{n+2}^{\mathrm{O}} \subset C_{n+1} \times C_{n+2}$$

The generating curve

There exists an alternative representation as self-intersections and boundary contacts of the (parametric and periodic) generating curve:

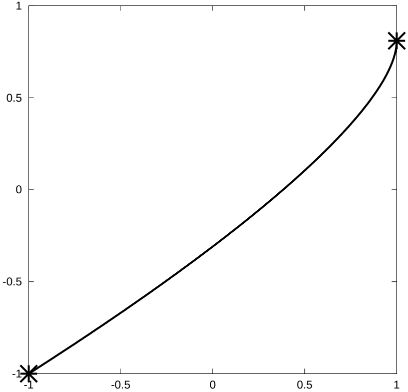
$$\gamma(t) = (-\cos((n+1)t), -\cos(nt)), \quad t \in [0, \pi]$$

The generating curve $\gamma(t)$ $(n = 4)$



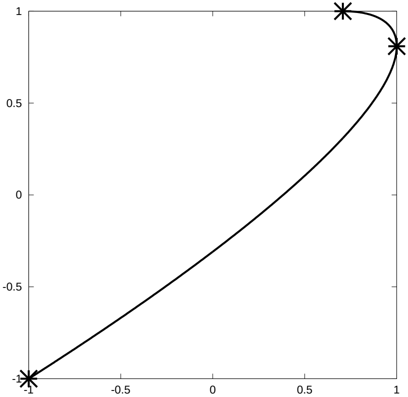
$t = 0$

The generating curve $\gamma(t)$ $(n = 4)$



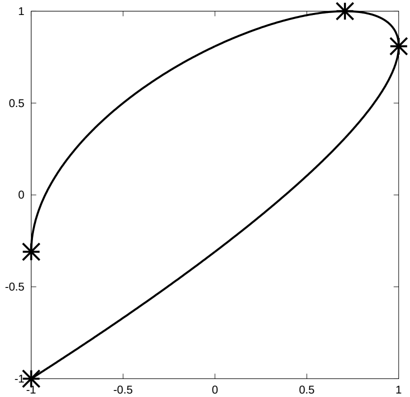
$$t \in \left[0, \frac{4\pi}{(n(n+1))}\right]$$

The generating curve $\gamma(t)$ $(n = 4)$



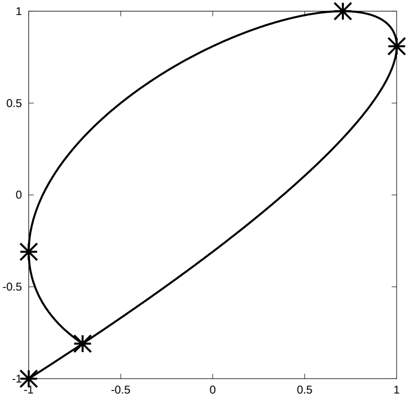
$$t \in \left[\frac{4\pi}{(n(n+1))}, \frac{5\pi}{(n(n+1))}\right]$$

The generating curve $\gamma(t)$ $(n = 4)$



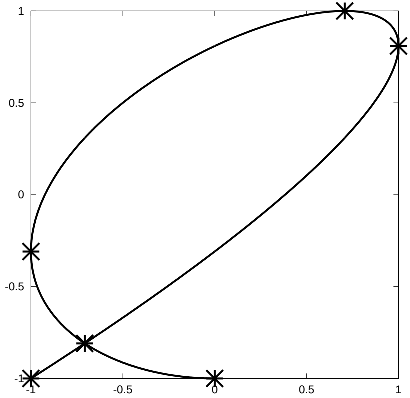
$$t \in \left[\frac{5\pi}{(n(n+1))}, \frac{8\pi}{(n(n+1))} \right]$$

The generating curve $\gamma(t)$ $(n = 4)$



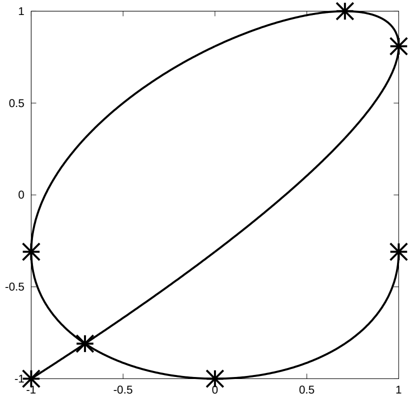
$$t \in \left[\frac{8\pi}{(n(n+1))}, \frac{9\pi}{(n(n+1))}\right]$$

The generating curve $\gamma(t)$ $(n = 4)$



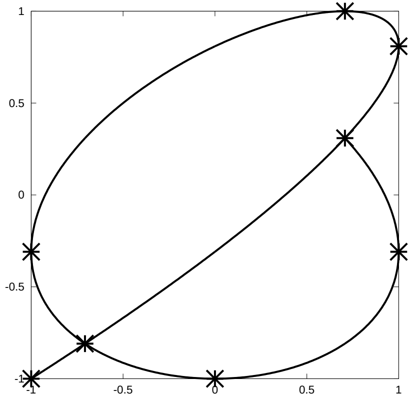
$$t \in \left[\frac{9\pi}{(n(n+1))}, \frac{10\pi}{(n(n+1))} \right]$$

The generating curve $\gamma(t)$ $(n = 4)$



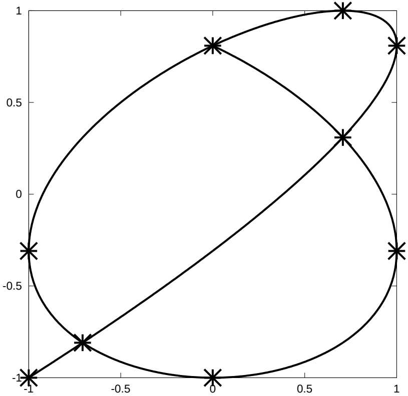
$$t \in \left[\frac{10\pi}{(n(n+1))}, \frac{12\pi}{(n(n+1))} \right]$$

The generating curve $\gamma(t)$ $(n = 4)$



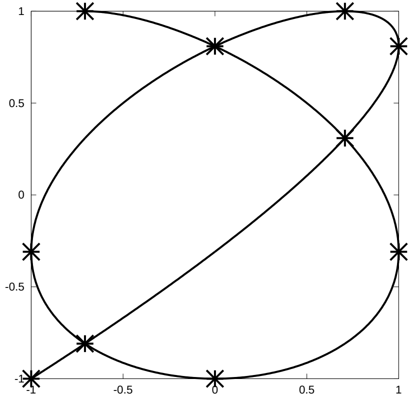
$$t \in \left[\frac{12\pi}{(n(n+1))}, \frac{13\pi}{(n(n+1))}\right]$$

The generating curve $\gamma(t)$ $(n = 4)$



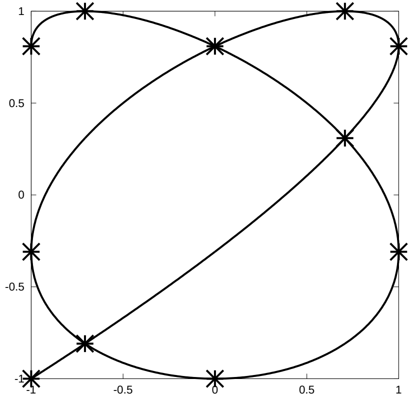
$$t \in \left[\frac{13\pi}{(n(n+1))}, \frac{14\pi}{(n(n+1))}\right]$$

The generating curve $\gamma(t)$ $(n = 4)$



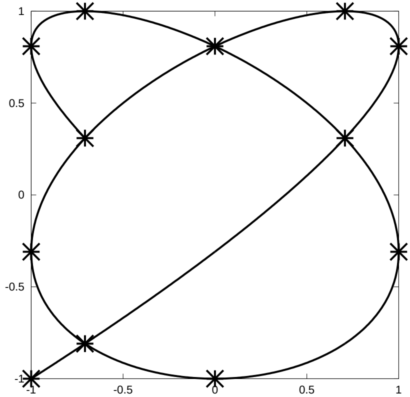
$$t \in \left[\frac{14\pi}{(n(n+1))}, \frac{15\pi}{(n(n+1))} \right]$$

The generating curve $\gamma(t)$ $(n = 4)$



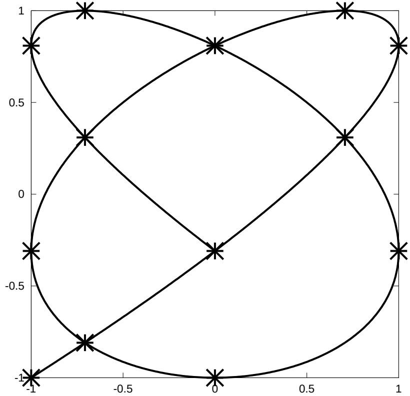
$$t \in \left[\frac{15\pi}{(n(n+1))}, \frac{16\pi}{(n(n+1))}\right]$$

The generating curve $\gamma(t)$ $(n = 4)$



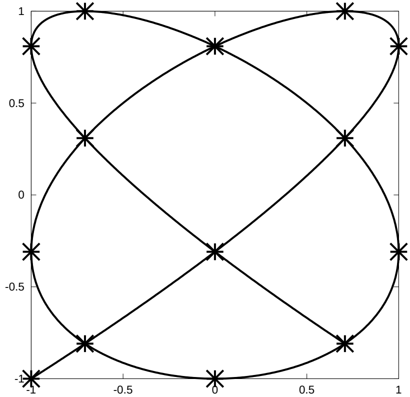
$$t \in \left[\frac{16\pi}{(n(n+1))}, \frac{17\pi}{(n(n+1))}\right]$$

The generating curve $\gamma(t)$ $(n = 4)$



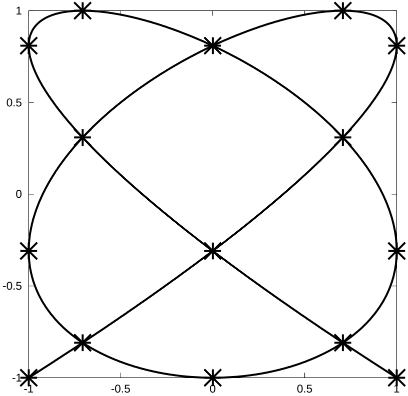
$$t \in \left[\frac{17\pi}{(n(n+1))}, \frac{18\pi}{(n(n+1))}\right]$$

The generating curve $\gamma(t)$ $(n = 4)$



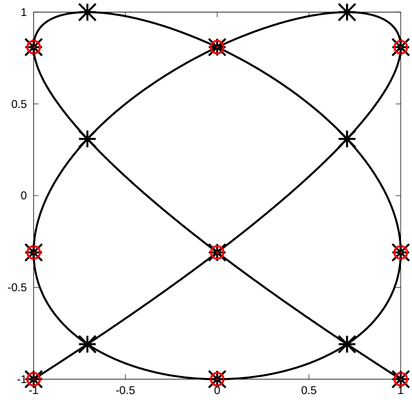
$$t \in \left[\frac{18\pi}{(n(n+1))}, \frac{19\pi}{(n(n+1))}\right]$$

The generating curve $\gamma(t)$ $(n = 4)$



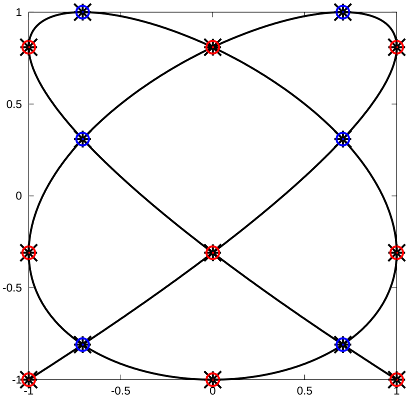
$$t \in \left[\frac{19\pi}{(n(n+1))}, \frac{20\pi}{(n(n+1))}\right]$$

The generating curve $\gamma(t)$ $(n = 4)$



$$C_{n+1}^{\text{odd}} \times C_{n+2}^{\text{even}}$$

The complete generating curve $\gamma(t)$ ($n = 4$)



$$\mathrm{Pad}_n = C^{\mathrm{O}}_{n+1} \times C^{\mathrm{E}}_{n+2} \cup C^{\mathrm{E}}_{n+1} \times C^{\mathrm{O}}_{n+2} \subset C_{n+1} \times C_{n+2}$$

The generating curve $\gamma(t)$ is a Lissajous curve

1. It is an algebraic curve: $T_{n+1}(x) = T_n(y)$ (for the first family!).
2. Lissajous curves are algebraic, their implicit equations can be found by using Chebyshev polynomials.
3. Chebyshev polynomials are Lissajous curves (cf. J.C. Merino, The Coll. Math. J. 34(2)2003).

Lagrange polynomials

The fundamental Lagrange polynomials of the Padua points are

$$L_{\boldsymbol{\xi}}(\mathbf{x}) = w_{\boldsymbol{\xi}} \left(K_n(\boldsymbol{\xi}, \mathbf{x}) - T_n(\xi_1) T_n(x_1) \right) \;, \quad L_{\boldsymbol{\xi}}(\boldsymbol{\eta}) = \delta_{\boldsymbol{\xi}\boldsymbol{\eta}}, \quad \boldsymbol{\xi}, \boldsymbol{\eta} \in \mathrm{Pad}_n \tag{1}$$

where

$$w_\xi = \frac{1}{n(n+1)} \cdot \begin{cases} \frac{1}{2} & \text{if } \xi \text{ is a vertex point} \\ 1 & \text{if } \xi \text{ is an edge point} \\ 2 & \text{if } \xi \text{ is an interior point} \end{cases}$$

$\{w_\xi\}$ are weights of cubature formula for the prod. Cheb. measure, exact "on almost" $\mathbb{P}^n_{2n}([-1,1]^2)$, i.e. pol. orthogonal to $T_{2n}(x_2)$

Lagrange polynomials

The fundamental Lagrange polynomials of the Padua points are

$$L_{\boldsymbol{\xi}}(\mathbf{x}) = w_{\boldsymbol{\xi}} \left(K_n(\boldsymbol{\xi}, \mathbf{x}) - T_n(\xi_1) T_n(x_1) \right) \ , \quad L_{\boldsymbol{\xi}}(\boldsymbol{\eta}) = \delta_{\boldsymbol{\xi}\boldsymbol{\eta}}, \quad \boldsymbol{\xi}, \boldsymbol{\eta} \in \mathrm{Pad}_n \tag{1}$$

where

$$w_{\boldsymbol{\xi}} = \frac{1}{n(n+1)} \cdot \begin{cases} \frac{1}{2} & \text{if } \boldsymbol{\xi} \text{ is a vertex point} \\ 1 & \text{if } \boldsymbol{\xi} \text{ is an edge point} \\ 2 & \text{if } \boldsymbol{\xi} \text{ is an interior point} \end{cases}$$

$\{w_{\boldsymbol{\xi}}\}$ are weights of cubature formula for the prod. Cheb. measure, exact "on almost" $\mathbb{P}^n_{2n}([-1,1]^2)$, i.e. pol. orthogonal to $T_{2n}(x_2)$

Reproducing kernel

$$K_n(\mathbf{x},\mathbf{y}) = \sum_{k=0}^{n}\sum_{j=0}^{k} \hat{T}_j(x_1)\,\hat{T}_{k-j}(x_2)\,\hat{T}_j(y_1)\,\hat{T}_{k-j}(y_2)\,, \quad \hat{T}_j = \sqrt{2}\,T_j,\ j \geq 1 \tag{2}$$

is the reproducing kernel of $\mathbb{P}_n^2([-1,1]^2)$ equipped with the inner product

$$\langle f, g\rangle = \int_{[-1,1]^2} f(x_1,x_2)g(x_1,x_2)\frac{\mathrm{d}x_1}{\pi\sqrt{1-x_1^2}}\frac{\mathrm{d}x_2}{\pi\sqrt{1-x_2^2}}\,,$$

with reproduction property

$$\int_{[-1,1]^2} K_n(\mathbf{x},\mathbf{y})p_n(\mathbf{y})w(\mathbf{y})\mathrm{d}\mathbf{y} = p_n(\mathbf{x}), \quad \forall p_n \in \mathbb{P}_n^2$$

$$w(\mathbf{x}) = w(x_1,x_2) = \frac{1}{\pi\sqrt{1-x_1^2}}\frac{1}{\pi\sqrt{1-x_2^2}}$$

Lebesgue constant

The Lebesgue constant

$$\Lambda_n = \max_{\mathbf{x} \in [-1,1]^2} \lambda_n(\mathbf{x}), \quad \lambda_n(\mathbf{x}) = \sum_{\boldsymbol{\xi} \in \mathrm{Pad}_n} |L_{\boldsymbol{\xi}}(\mathbf{x})|$$

is bounded by (cf. BCDeMVX, Numer. Math. 2006)

$$\Lambda_n \leq C \log^2 n \qquad (3)$$

(optimal order of growth on a square).

Interpolant

From the representations (1) (Lagrange poly.) and (2) (reproducing kernel) the interpolant of a function $f\colon [-1,1]^2 \to \mathbb{R}$ is

$$\mathcal{L}_n f(\mathbf{x}) = \sum_{\boldsymbol{\xi} \in \mathrm{Pad}_n} f(\xi) L_{\boldsymbol{\xi}}(\mathbf{x}) = \sum_{\boldsymbol{\xi} \in \mathrm{Pad}_n} f(\xi) \left[w_{\boldsymbol{\xi}} \left(K_n(\boldsymbol{\xi}, \mathbf{x}) - T_n(\xi_1) T_n(x_1) \right) \right] =$$

$$= \sum_{k=0}^{n} \sum_{j=0}^{k} c_{j,k-j} \hat{T}_j(x_1) \hat{T}_{k-j}(x_2) - \frac{c_{n,0}}{2} \hat{T}_n(x_1) \hat{T}_0(x_2) \,,$$

where the coefficients

$$c_{j,k-j} = \sum_{\boldsymbol{\xi} \in \mathrm{Pad}_n} f(\xi) w_{\boldsymbol{\xi}} \hat{T}_j(\xi_1) \hat{T}_{k-j}(\xi_2), \quad 0 \le j \le k \le n$$

can be computed once and for all.

Interpolant

From the representations (1) (Lagrange poly.) and (2) (reproducing kernel) the interpolant of a function $f\colon [-1,1]^2 \to \mathbb{R}$ is

$$\mathcal{L}_n f(\mathbf{x}) = \sum_{\boldsymbol{\xi} \in \mathrm{Pad}_n} f(\boldsymbol{\xi}) L_{\boldsymbol{\xi}}(\mathbf{x}) = \sum_{\boldsymbol{\xi} \in \mathrm{Pad}_n} f(\boldsymbol{\xi}) \left[w_{\boldsymbol{\xi}} \left(K_n(\boldsymbol{\xi}, \mathbf{x}) - T_n(\xi_1) T_n(x_1) \right) \right] =$$

$$= \sum_{k=0}^{n} \sum_{j=0}^{k} c_{j,k-j} \hat{T}_j(x_1) \hat{T}_{k-j}(x_2) - \frac{c_{n,0}}{2} \hat{T}_n(x_1) \hat{T}_0(x_2) \, ,$$

where the coefficients

$$c_{j,k-j} = \sum_{\boldsymbol{\xi} \in \mathrm{Pad}_n} f(\boldsymbol{\xi}) w_{\boldsymbol{\xi}} \hat{T}_j(\xi_1) \hat{T}_{k-j}(\xi_2), \quad 0 \le j \le k \le n$$

can be computed once and for all.

Coefficient matrix

Let us define the $n+1 \times n+1$ coefficient matrix

$$\mathbb{C}_0 = \begin{pmatrix} c_{0,0} & c_{0,1} & \dots & \dots & c_{0,n} \\ c_{1,0} & c_{1,1} & \dots & c_{1,n-1} & 0 \\ \vdots & \vdots & \iddots & \iddots & \vdots \\ c_{n-1,0} & c_{n-1,1} & 0 & \dots & 0 \\ \frac{c_{n,0}}{2} & 0 & \dots & 0 & 0 \end{pmatrix}$$

and for a vector $S = (s_1, \dots, s_m)$, $S \in [-1,1]^m$, the $(n+1) \times m$ Chebyshev collocation matrix

$$\mathbb{T}(S) = \begin{pmatrix} \hat{T}_0(s_1) & \dots & \hat{T}_0(s_m) \\ \vdots & \dots & \vdots \\ \hat{T}_n(s_1) & \dots & \hat{T}_n(s_m) \end{pmatrix}$$

Coefficient matrix factorization

Letting C_{n+1} the vector of the Chebyshev-Lobatto pts

$$C_{n+1} = \left(z_1^n, \ldots, z_{n+1}^n\right)$$

we construct the $(n+1) \times (n+2)$ matrix

$$\mathbb{G}(f) = (g_{r,s}) = \begin{cases} w_\xi f(z_r^n, z_s^{n+1}) & \text{if } \xi = (z_r^n, z_s^{n+1}) \in \mathrm{Pad}_n \\ 0 & \text{if } \xi = (z_r^n, z_s^{n+1}) \in (C_{n+1} \times C_{n+2}) \setminus \mathrm{Pad}_n \end{cases}.$$

Then $\mathbb{C}_0$ is essentially the upper-left triangular part of

$$\mathbb{C}(f) = \mathbb{P}_1 \, \mathbb{G}(f) \mathbb{P}_2^{\mathrm{T}}$$

$\mathbb{P}_1 = \mathbb{T}(C_{n+1}) \in \mathbb{R}^{(n+1)\times(n+1)}$ and $\mathbb{P}_2 = \mathbb{T}(C_{n+2}) \in \mathbb{R}^{(n+1)\times(n+2)}$.

Coefficient matrix factorization

Letting C_{n+1} the vector of the Chebyshev-Lobatto pts

$$C_{n+1} = \left(z_1^n, \ldots, z_{n+1}^n\right)$$

we construct the $(n+1) \times (n+2)$ matrix

$$\mathbb{G}(f) = (g_{r,s}) = \begin{cases} w_{\xi} f(z_r^n, z_s^{n+1}) & \text{if } \xi = (z_r^n, z_s^{n+1}) \in \mathrm{Pad}_n \\ 0 & \text{if } \xi = (z_r^n, z_s^{n+1}) \in (C_{n+1} \times C_{n+2}) \setminus \mathrm{Pad}_n \end{cases} .$$

Then $\mathbb{C}_0$ is essentially the upper-left triangular part of

$$\mathbb{C}(f) = \mathbb{P}_1 \, \mathbb{G}(f) \mathbb{P}_2^{\mathrm{T}}$$

$\mathbb{P}_1 = \mathbb{T}(C_{n+1}) \in \mathbb{R}^{(n+1)\times(n+1)}$ and $\mathbb{P}_2 = \mathbb{T}(C_{n+2}) \in \mathbb{R}^{(n+1)\times(n+2)}$.

Coefficient matrix factorization

Letting C_{n+1} the vector of the Chebyshev-Lobatto pts

$$C_{n+1} = \left(z_1^n, \ldots, z_{n+1}^n\right)$$

we construct the $(n+1) \times (n+2)$ matrix

$$\mathbb{G}(f) = (g_{r,s}) = \begin{cases} w_{\boldsymbol{\xi}} f(z_r^n, z_s^{n+1}) & \text{if } \boldsymbol{\xi} = (z_r^n, z_s^{n+1}) \in \mathrm{Pad}_n \\ 0 & \text{if } \boldsymbol{\xi} = (z_r^n, z_s^{n+1}) \in (C_{n+1} \times C_{n+2}) \setminus \mathrm{Pad}_n \end{cases} .$$

Then $\mathbb{C}_0$ is essentially the upper-left triangular part of

$$\mathbb{C}(f) = \mathbb{P}_1 \, \mathbb{G}(f) \mathbb{P}_2^{\mathrm{T}}$$

$\mathbb{P}_1 = \mathbb{T}(C_{n+1}) \in \mathbb{R}^{(n+1)\times(n+1)}$ and $\mathbb{P}_2 = \mathbb{T}(C_{n+2}) \in \mathbb{R}^{(n+1)\times(n+2)}$.

Coefficient matrix factorization

Exploiting the fact that the Padua points are union of two Chebyshev subgrids, we may define the two matrices

$$\mathbb{G}_1(f) = \left(w_{\boldsymbol{\xi}} f(\boldsymbol{\xi})\,,\, \boldsymbol{\xi} = (z_r^n, z_s^{n+1}) \in C_{n+1}^{\mathrm{E}} \times C_{n+2}^{\mathrm{O}}\right)$$

$$\mathbb{G}_2(f) = \left(w_{\boldsymbol{\xi}} f(\boldsymbol{\xi})\,,\, \boldsymbol{\xi} = (z_r^n, z_s^{n+1}) \in C_{n+1}^{\mathrm{O}} \times C_{n+2}^{\mathrm{E}}\right)$$

then we can compute the coefficient matrix as

$$\mathbb{C}(f) = \mathbb{T}(C_{n+1}^{\mathrm{E}})\, \mathbb{G}_1(f)\, (\mathbb{T}(C_{n+2}^{\mathrm{O}}))^t + \mathbb{T}(C_{n+1}^{\mathrm{O}})\, \mathbb{G}_2(f)\, (\mathbb{T}(C_{n+2}^{\mathrm{E}}))^t$$

We term this approach as MM, Matrix-Multiplication.

Coefficient matrix factorization by FFT

$$c_{j,l} = \sum_{\boldsymbol{\xi} \in \mathrm{Pad}_n} f(\boldsymbol{\xi}) w_{\boldsymbol{\xi}} \hat{T}_j(\xi_1) \hat{T}_l(\xi_2) = \sum_{r=0}^{n} \sum_{s=0}^{n+1} g_{r,s} \hat{T}_j(z_r^n) \hat{T}_l(z_s^{n+1})$$

$$= \beta_{j,l} \sum_{r=0}^{n} \sum_{s=0}^{n+1} g_{r,s} \cos \frac{jr\pi}{n} \cos \frac{ls\pi}{n+1} = \beta_{j,l} \sum_{s=0}^{M-1} \left(\sum_{r=0}^{N-1} g^0_{r,s} \cos \frac{2jr\pi}{N} \right) \cos \frac{2ls\pi}{M}$$

where $N = 2n$, $M = 2(n+1)$ and

$$\beta_{j,l} = \begin{cases} 1 & j = l = 0 \\ 2 & j \neq 0, \ l \neq 0 \\ \sqrt{2} & \text{otherwise} \end{cases} \qquad g^0_{r,s} = \begin{cases} g_{r,s} & 0 \le r \le n \text{ and } 0 \le s \le n+1 \\ 0 & r > n \text{ or } s > n+1 \end{cases}$$

Coefficient matrix factorization by FFT

The coefficients $c_{j,l}$ can be computed by a double Discrete Fourier Transform.

$$\hat{g}_{j,s} = \mathrm{REAL}\left(\sum_{r=0}^{N-1} g^0_{r,s}\mathrm{e}^{-2\pi \mathrm{i} jr/N}\right), \quad 0 \le j \le n,\ 0 \le s \le M-1$$

$$\frac{c_{j,l}}{\beta_{j,l}} = \hat{\hat{g}}_{j,l} = \mathrm{REAL}\left(\sum_{s=0}^{M-1} \hat{g}_{j,s}\mathrm{e}^{-2\pi \mathrm{i} ls/M}\right), \quad 0 \le j \le n,\ 0 \le l \le n-j \tag{4}$$

MATLAB® code for the FFT approach

Input: G $\leftrightarrow \mathbb{G}(f)$

```
Gfhat = real(fft(G,2*n));
Gfhat = Gfhat(1:n+1,:);

Gfhathat =real(fft(Gfhat,2*(n+1),2));

C0f = Gfhathat(:,1:n+1);
C0f =2*C0f; C0f(1,:) = C0f(1,:)/sqrt(2);
C0f(:,1) = C0f(:,1)/sqrt(2);
C0f = fliplr(triu(fliplr(C0f)));
C0f(n+1,1) = C0f(n+1,1)/2;
```

Output: C0 $\leftrightarrow \mathbb{C}_0$

Linear algebra approach vs FFT approach

- The construction of the coefficients is performed by a matrix-matrix product.
- It has been easily and efficiently implemented in FORTRAN77 (by, eventually optimized, BLAS) (cf. CDeMV, TOMS 2008) and in MATLAB® (based on optimized BLAS).
- The coefficients are approximated Fourier–Chebyshev coefficients, hence they can be computed by FFT techniques.
- FFT is competitive and more stable than the MM approach at high degrees of interpolation (see later).

Linear algebra approach vs FFT approach

- The construction of the coefficients is performed by a matrix-matrix product.
- It has been easily and efficiently implemented in FORTRAN77 (by, eventually optimized, BLAS) (cf. CDeMV, TOMS 2008) and in MATLAB® (based on optimized BLAS).
- The coefficients are approximated Fourier–Chebyshev coefficients, hence they can be computed by FFT techniques.
- FFT is competitive and more stable than the MM approach at high degrees of interpolation (see later).

Linear algebra approach vs FFT approach

- The construction of the coefficients is performed by a matrix-matrix product.
- It has been easily and efficiently implemented in FORTRAN77 (by, eventually optimized, BLAS) (cf. CDeMV, TOMS 2008) and in MATLAB® (based on optimized BLAS).
- The coefficients are approximated Fourier–Chebyshev coefficients, hence they can be computed by FFT techniques.
- FFT is competitive and more stable than the MM approach at high degrees of interpolation (see later).

Linear algebra approach vs FFT approach

- The construction of the coefficients is performed by a matrix-matrix product.
- It has been easily and efficiently implemented in FORTRAN77 (by, eventually optimized, BLAS) (cf. CDeMV, TOMS 2008) and in MATLAB® (based on optimized BLAS).
- The coefficients are approximated Fourier–Chebyshev coefficients, hence they can be computed by FFT techniques.
- FFT is competitive and more stable than the MM approach at high degrees of interpolation (see later).

Evaluating the interpolant (in Matlab)

- Given a point $\mathbf{x} = (x_1, x_2)$ and the coefficient matrix $\mathbb{C}_0$, the polynomial interpolation formula can be evaluated by a double matrix-vector product

$$\mathcal{L}_n f(\mathbf{x}) = \mathbb{T}(x_1)^{\mathrm{T}} \mathbb{C}_0(f) \mathbb{T}(x_2)$$

- If $\mathbf{X} = (X_1, X_2)$ ($X_{1,2}$ column vectors) is a set of target points, then

$$\mathcal{L}_n f(\mathbf{X}) = \operatorname{diag}\left((\mathbb{T}(X_1))^t \ \mathbb{C}_0(f) \ \mathbb{T}(X_2) \right) \tag{5}$$

  The result $\mathcal{L}_n f(\mathbf{X})$ is a (column) vector.

- If $\mathbf{X} = X_1 \times X_2$ is a Cartesian grid then

$$\mathcal{L}_n f(\mathbf{X}) = \left((\mathbb{T}(X_1))^t \ \mathbb{C}_0(f) \ \mathbb{T}(X_2) \right)^t \tag{6}$$

  The result $\mathcal{L}_n f(\mathbf{X})$ is a matrix whose i-th row and j-th column contains the evaluation of the interpolant as the built-in function `meshgrid` of MATLAB®.

Beyond the square

The interpolation formula can be extended to other domains $\Omega \subset \mathbb{R}^2$, by means of a suitable mapping of the square (cf. CDeMV JCAM2008). Given

$$\begin{aligned} \boldsymbol{\sigma} : [-1,1]^2 &\to \Omega \\ \mathbf{t} &\mapsto \mathbf{x} = \boldsymbol{\sigma}(\mathbf{t}) \end{aligned}$$

it is possible to construct the (in general nonpolynomial) interpolation formula

$$\mathcal{L}_n f(\mathbf{x}) = \mathbb{T}(\sigma_1^{\leftarrow}(\mathbf{x}))^{\mathrm{T}} \mathbb{C}_0(f \circ \boldsymbol{\sigma}) \mathbb{T}(\sigma_2^{\leftarrow}(\mathbf{x}))$$

Beyond the square

The interpolation formula can be extended to other domains $\Omega \subset \mathbb{R}^2$, by means of a suitable mapping of the square (cf. CDeMV JCAM2008). Given

$$\boldsymbol{\sigma} : [-1,1]^2 \to \Omega$$
$$\mathbf{t} \mapsto \mathbf{x} = \boldsymbol{\sigma}(\mathbf{t})$$

it is possible to construct the (in general nonpolynomial) interpolation formula

$$\mathcal{L}_n f(\mathbf{x}) = \mathbb{T}(\sigma_1^{\leftarrow}(\mathbf{x}))^{\mathrm{T}} \mathbb{C}_0(f \circ \boldsymbol{\sigma}) \mathbb{T}(\sigma_2^{\leftarrow}(\mathbf{x}))$$

Cubature

Integration of the interpolant at the Padua points gives a nontensorial Clenshaw–Curtis cubature formula (cf. SVZ, Numer. Algorithms 2008)

$$\begin{aligned}\int_{[-1,1]^2} f(\mathbf{x})\mathrm{d}\mathbf{x} &\approx \int_{[-1,1]^2} \mathcal{L}_n f(\mathbf{x})\mathrm{d}\mathbf{x} = \sum_{k=0}^{n}\sum_{j=0}^{k} c'_{j,k-j}\, m_{j,k-j} \\ &= \sum_{j=0}^{n}\sum_{l=0}^{n} c'_{j,l}\, m_{j,l} = \sum_{j,\,\mathrm{even}}^{n}\ \sum_{l,\,\mathrm{even}}^{n} c'_{j,l}\, m_{j,l}\end{aligned}$$

Cubature

Where the *moments* $m_{j,l}$ are

$$m_{j,l} = \int_{-1}^{1} \hat{T}_j(t)\mathrm{d}t \int_{-1}^{1} \hat{T}_l(t)\mathrm{d}t$$

Since

$$\int_{-1}^{1} \hat{T}_j(t)\mathrm{d}t = \begin{cases} 2 & j = 0 \\ 0 & j \text{ odd} \\ \dfrac{2\sqrt{2}}{1-j^2} & j \text{ even} \end{cases}$$

The MATLAB® code for the cubature

Input: `C0f` $\leftrightarrow \mathbb{C}_0(f)$

```
j = [0:2:n];
mom = 2*sqrt(2)./(1-j.^2);
mom(1) = 2;
[M1,M2]=meshgrid(mom);
M = M1.*M2;
C0fM = C0f(1:2:n+1,1:2:n+1).*M;
Int = sum(sum(C0fM));
```

Output: `Int` $\leftrightarrow I_n(f)$

Cubature

It is often desiderable having a cubature formula involving the function values at the nodes and the corresponding cubature weights. Using the formula for the coefficients $c_{j,l}$, we can write

$$\begin{aligned} I_n(f) &= \sum_{\boldsymbol{\xi} \in \mathrm{Pad}_n} \lambda_{\boldsymbol{\xi}}\, f(\boldsymbol{\xi}) \\ &= \sum_{\boldsymbol{\xi} \in C_{n+1}^{\mathrm{E}} \times C_{n+2}^{\mathrm{O}}} \lambda_{\boldsymbol{\xi}}\, f(\boldsymbol{\xi}) + \sum_{\boldsymbol{\xi} \in C_{n+1}^{\mathrm{O}} \times C_{n+2}^{\mathrm{E}}} \lambda_{\boldsymbol{\xi}}\, f(\boldsymbol{\xi}) \end{aligned}$$

where

$$\lambda_{\boldsymbol{\xi}} = w_{\boldsymbol{\xi}} \sum_{j \text{ even}}^{n} \sum_{l \text{ even}}^{n} m'_{j,l}\, \hat{T}_j(\xi_1)\, \hat{T}_l(\xi_2) \qquad (7)$$

Cubature

Defining the Chebyshev matrix corresponding to even degrees

$$\mathbb{T}^{\mathrm{E}}(S) = \begin{pmatrix} \hat{T}_0(s_1) & \cdots & \hat{T}_0(s_m) \\ \hat{T}_2(s_1) & \cdots & \hat{T}_2(s_m) \\ \vdots & \cdots & \vdots \\ \hat{T}_{p_n}(s_1) & \cdots & \hat{T}_{p_n}(s_m) \end{pmatrix} \in \mathbb{R}^{([\frac{n}{2}]+1)\times m}$$

and the matrices of weights on the subgrids,
$\mathbb{W}_1 = \left(w_{\boldsymbol{\xi}},\, \boldsymbol{\xi} \in C^{\mathrm{E}}_{n+1} \times C^{\mathrm{O}}_{n+2}\right)^t$, $\mathbb{W}_2 = \left(w_{\boldsymbol{\xi}},\, \boldsymbol{\xi} \in C^{\mathrm{O}}_{n+1} \times C^{\mathrm{E}}_{n+2}\right)^t$, then the cubature weights $\{\lambda_{\boldsymbol{\xi}}\}$ can be computed in matrix form

$$\mathbb{L}_1 = \left(\lambda_{\boldsymbol{\xi}},\, \boldsymbol{\xi} \in C^{\mathrm{E}}_{n+1} \times C^{\mathrm{O}}_{n+2}\right)^t = \mathbb{W}_1 . \left(\mathbb{T}^{\mathrm{E}}(C^{\mathrm{E}}_{n+1})\right)^t \mathbb{M}_0\, \mathbb{T}^{\mathrm{E}}(C^{\mathrm{O}}_{n+2})\right)^t$$

$$\mathbb{L}_2 = \left(\lambda_{\boldsymbol{\xi}},\, \boldsymbol{\xi} \in C^{\mathrm{O}}_{n+1} \times C^{\mathrm{E}}_{n+2}\right)^t = \mathbb{W}_2 . \left(\mathbb{T}^{\mathrm{E}}(C^{\mathrm{O}}_{n+1})\right)^t \mathbb{M}_0\, \mathbb{T}^{\mathrm{E}}(C^{\mathrm{E}}_{n+2})\right)^t$$

where $\mathbb{M}_0 = \left(m'_{j,l}\right)$ (moment matrix) and the dot means that the final product is entrywise (Hadamard or Schur product).

Cubature

1. An FFT-based implementation is then feasible, in analogy to what happens in the univariate case with the Clenshaw-Curtis formula (cf. Waldvogel, BIT06). The algorithm is quite similar the one for interpolation (cf. CDSV, Numer. Alg. 2010)
2. The cubature weights are not all positive, but the negative ones are few and of small size and

$$\lim_{n\to\infty} \sum_{\boldsymbol{\xi}\in \mathrm{Pad}_n} |\lambda_{\boldsymbol{\xi}}| = 4$$

i.e. stability and convergence for every continuous f.

Numerical tests

Language: MATLAB® 7.6.0
Processor: Intel Core2 Duo 2.2GHz.
Similar results with Octave 3.2.3.

n	20	40	60	80	100	300	500	1000
FFT	0.001	0.001	0.001	0.002	0.003	0.034	0.115	0.387
MM	0.002	0.003	0.003	0.003	0.008	0.101	0.298	1.353

Table : CPU time (in seconds) for the computation of the interpolation coefficients at a sequence of degrees (average of 10 runs).

n	20	40	60	80	100	300	500	1000
FFT	0.001	0.001	0.002	0.002	0.004	0.028	0.111	0.389
MM	0.001	0.001	0.001	0.002	0.003	0.027	0.092	0.554

Table : CPU time (in seconds) for the computation of the cubature weights at a sequence of degrees (average of 10 runs).

Numerical tests

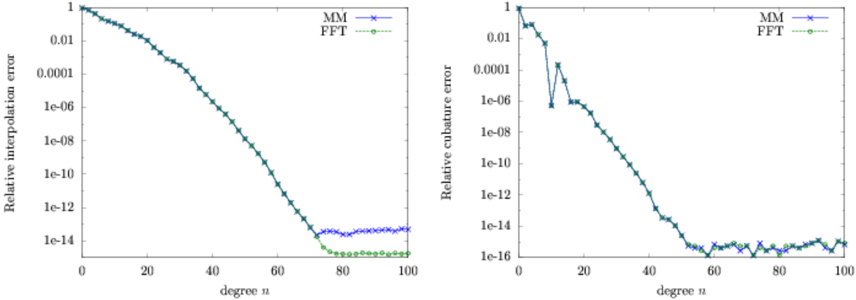


Figure : Relative interpolation errors (left) and cubature (right) versus the interpolation degree for the Franke test function in $[0,1]^2$, by the Matrix Multiplication (MM) and the FFT-based algorithms.

Numerical tests

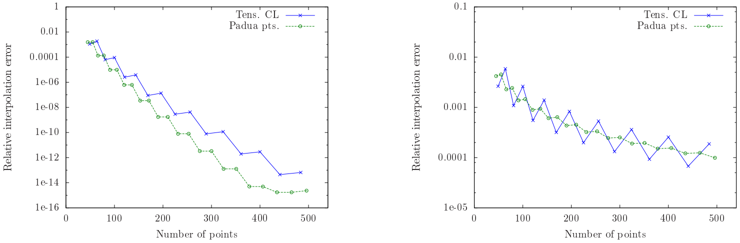


Figure : Relative interpolation errors versus the number of interpolation points for the Gaussian $f(\mathbf{x}) = \exp(-|\mathbf{x}|^2)$ (left) and the C^2 function $f(\mathbf{x}) = |\mathbf{x}|^3$ (right) in $[-1,1]^2$; Tens. CL = Tensorial Chebyshev-Lobatto interpolation.

Numerical tests

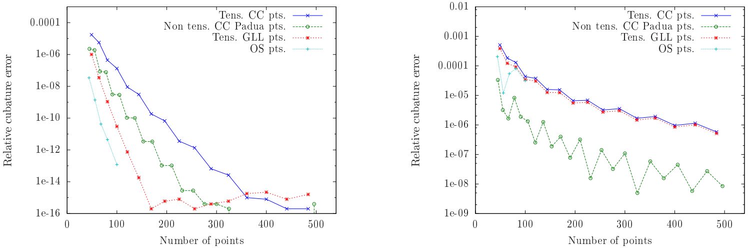


Figure : Relative cubature errors versus the number of cubature points (CC = Clenshaw-Curtis, GLL = Gauss-Legendre-Lobatto, OS = Omelyan-Solovyan) for the Gaussian $f(\mathbf{x}) = \exp(-|\mathbf{x}|^2)$ (left) and the C^2 function $f(\mathbf{x}) = |\mathbf{x}|^3$ (right); the integration domain is $[-1, 1]^2$, the integrals up to machine precision are, respectively: 2.230985141404135 and 2.508723139534059.

Padua points on triangle

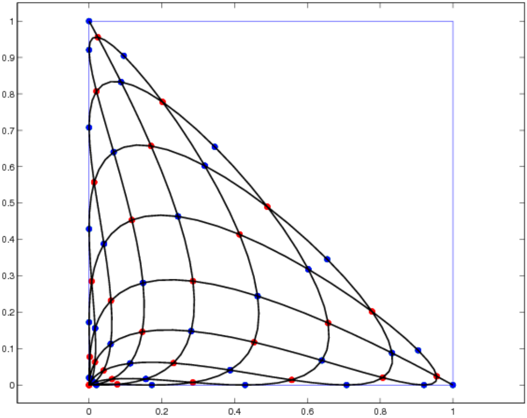
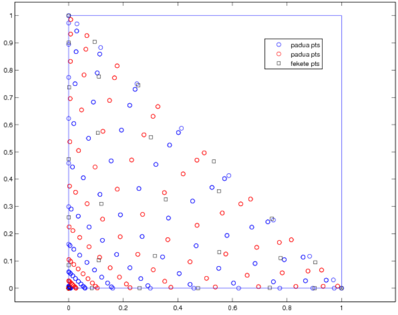


Figure : Padua points on the unit triangle for $n = 10$.

Approximated Fekete pts from Padua points on triangle



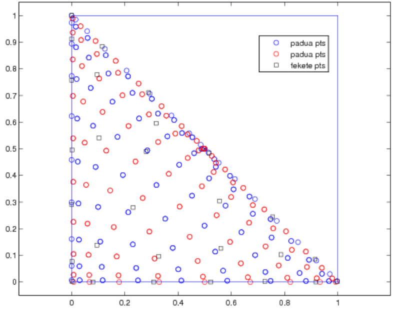


Figure : Fekete points for $n = 6$ extracted from a mesh of Padua points for $n = 24$. Left: the Padua points mapped on the lower vertex transformation. Right: Padua points on the triangle mapped along the diagonal.

Applications

1. Padua points are WAM (Weakly Admissible Meshes) for interpolation or extracting Fekete points on 2D domains (cf. BSV09)
2. Padua points can be used in 3D (tensor product) WAMs on different domains (Master's theses recently done at UniPD)
3. Vandermonde determinant of Padua Points has variables that separate: this was an open question (see BDeMW DRNA09) now solved (see DeMU13 also as arXiv:1311.6455)
4. Histogram Compression and Image Retrieval Through Padua Points Interpolation (cf. Montagna-Finlayson 2008) "Experiments show that our new compact Padua point representation supports excellent indexing and recognition."

Applications

1. Padua points are WAM (Weakly Admissible Meshes) for interpolation or extracting Fekete points on 2D domains (cf. BSV09)
2. Padua points can be used in 3D (tensor product) WAMs on different domains (Master's theses recently done at UniPD)
3. Vandermonde determinant of Padua Points has variables that separate: this was an open question (see BDeMW DRNA09) now solved (see DeMU13 also as arXiv:1311.6455)
4. Histogram Compression and Image Retrieval Through Padua Points Interpolation (cf. Montagna-Finlayson 2008) "Experiments show that our new compact Padua point representation supports excellent indexing and recognition."

Applications

1. Padua points are WAM (Weakly Admissible Meshes) for interpolation or extracting Fekete points on 2D domains (cf. BSV09)
2. Padua points can be used in 3D (tensor product) WAMs on different domains (Master's theses recently done at UniPD)
3. Vandermonde determinant of Padua Points has variables that separate: this was an open question (see BDeMW DRNA09) now solved (see DeMU13 also as arXiv:1311.6455)
4. Histogram Compression and Image Retrieval Through Padua Points Interpolation (cf. Montagna-Finlayson 2008) "Experiments show that our new compact Padua point representation supports excellent indexing and recognition."

Applications

1. Padua points are WAM (Weakly Admissible Meshes) for interpolation or extracting Fekete points on 2D domains (cf. BSV09)
2. Padua points can be used in 3D (tensor product) WAMs on different domains (Master's theses recently done at UniPD)
3. Vandermonde determinant of Padua Points has variables that separate: this was an open question (see BDeMW DRNA09) now solved (see DeMU13 also as arXiv:1311.6455)
4. Histogram Compression and Image Retrieval Through Padua Points Interpolation (cf. Montagna-Finlayson 2008) "Experiments show that our new compact Padua point representation supports excellent indexing and recognition."

Applications

1. Padua points are WAM (Weakly Admissible Meshes) for interpolation or extracting Fekete points on 2D domains (cf. BSV09)
2. Padua points can be used in 3D (tensor product) WAMs on different domains (Master's theses recently done at UniPD)
3. Vandermonde determinant of Padua Points has variables that separate: this was an open question (see BDeMW DRNA09) now solved (see DeMU13 also as arXiv:1311.6455)
4. Histogram Compression and Image Retrieval Through Padua Points Interpolation (cf. Montagna-Finlayson 2008) "Experiments show that our new compact Padua point representation supports excellent indexing and recognition."

Applications

1. New observations on the distribution of Padua points by Cuyt et al. NA2012

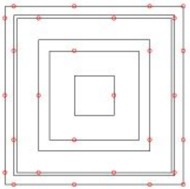


Figure : Padua pts for $n = 6$, they lie on n concentric squares with sides at the zeros of U_n and U_{n-1} (the inner) except the external and the center (just a dot!) .

2. For more applications see www.math.unipd.it/∼marcov/CAApadua.html

Some open problems

1. Still a conjecture the fact that the Lebesgue function attains its maximum in one of the vertices $(-1,1)$ or $(1,1)$ (for the first family)
2. Padua points in 3D ... open problem
3. Make the software more efficient (if there's any possibility), maybe by using ChebFun2 (Nick Trefethen's definition of efficiency: 10 digits, 5 sec. and 1 page!)

Some open problems

1. Still a conjecture the fact that the Lebesgue function attains its maximum in one of the vertices $(-1,1)$ or $(1,1)$ (for the first family)
2. Padua points in 3D ... open problem
3. Make the software more efficient (if there's any possibility), maybe by using ChebFun2 (Nick Trefethen's definition of efficiency: 10 digits, 5 sec. and 1 page!)

Some open problems

1. Still a conjecture the fact that the Lebesgue function attains its maximum in one of the vertices $(-1,1)$ or $(1,1)$ (for the first family)
2. Padua points in 3D ... open problem
3. Make the software more efficient (if there's any possibility), maybe by using ChebFun2 (Nick Trefethen's definition of efficiency: 10 digits, 5 sec. and 1 page!)

THANK YOU
FOR YOUR KIND ATTENTION