Interpolation points and interpolation formulae on the square*

Stefano De Marchi

Department of Computer Science, University of Verona

Zaragoza, 11 December 2008

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

Motivations and aims

- 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.

Outline

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.

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

with $P \in \mathbb{P}_n([-1,1])$.

This metric generalizes to compact sets $\Omega \subset \mathbb{R}^d$, $d > 1$:

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

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.

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

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

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

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 we have at least a method for producing candidate points. Letting $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2)$

- Dubiner metric on the square:

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

- Dubiner metric on the disk:

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

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 we have at least a method for producing candidate points. Letting $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2)$

- Dubiner metric on the square:

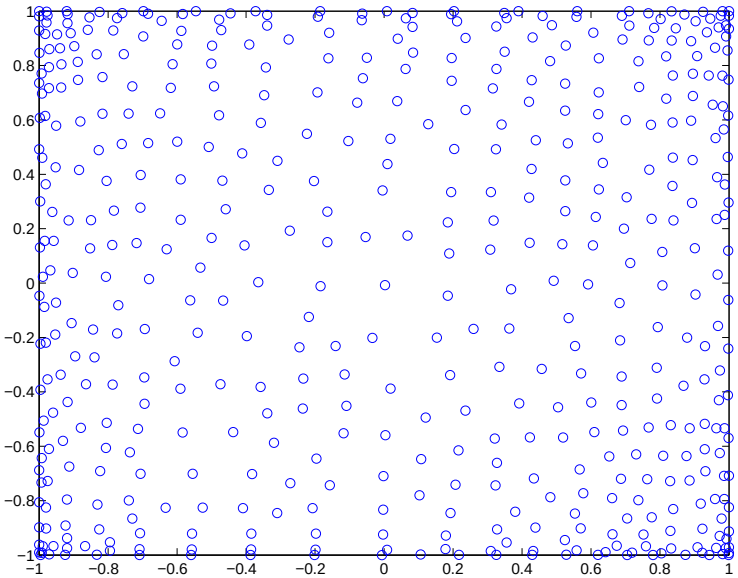
$$\max\{|\arccos(x_1) - \arccos(y_1)|, |\arccos(x_2) - \arccos(y_2)|\} ;$$

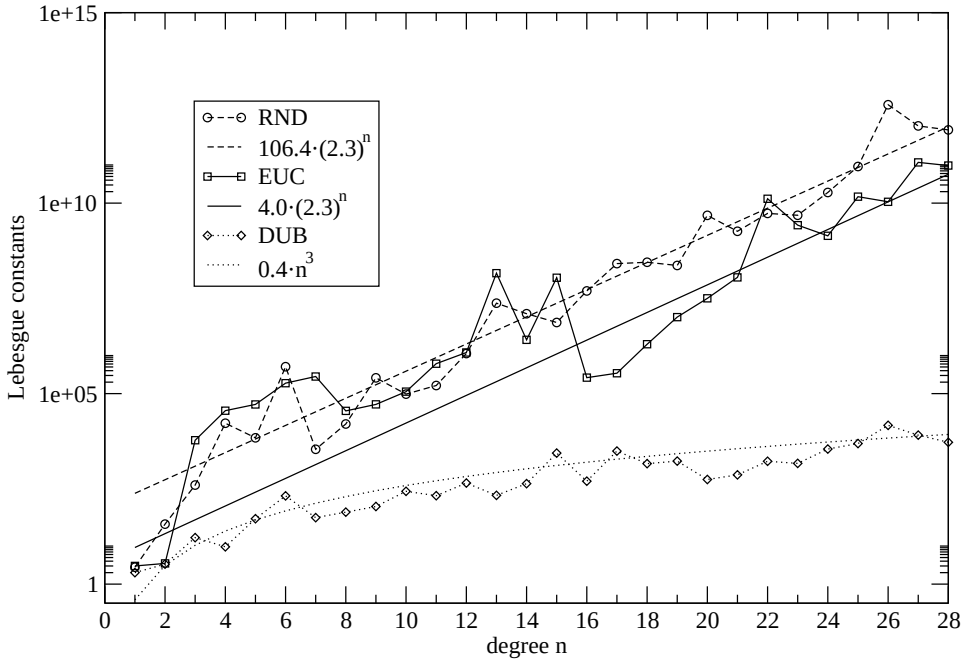
- Dubiner metric on the disk:

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

Dubiner points and Lebesgue constant

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





Note: Euclidean pts, $\max_{\mathbf{x}\in\Omega} \min_{\mathbf{y}\in X_n} \|\mathbf{x}-\mathbf{y}\|_2$.

Morrow-Patterson points

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

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

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

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 $[-1,1]^2$ and unisolvent for polynomial interpolation of degree n.

Padua points

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

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

$1 \le m \le n+1$, $1 \le k \le n/2+1$, $N = (n+2)(n+1)/2$.

- 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.
- Moreover, there are four families which correspond to successive rotations of 90 degrees, clockwise for even degrees and counterclockwise for odd degrees.

Padua points

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

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

$1 \le m \le n+1$, $1 \le k \le n/2+1$, $N = (n+2)(n+1)/2$.

- 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.
- Moreover, there are four families which correspond to successive rotations of 90 degrees, clockwise for even degrees and counterclockwise for odd degrees.

Padua points

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

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

$1 \le m \le n+1$, $1 \le k \le n/2+1$, $N = (n+2)(n+1)/2$.

- 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.
- Moreover, there are four families which correspond to successive rotations of 90 degrees, clockwise for even degrees and counterclockwise for odd degrees.

Padua points

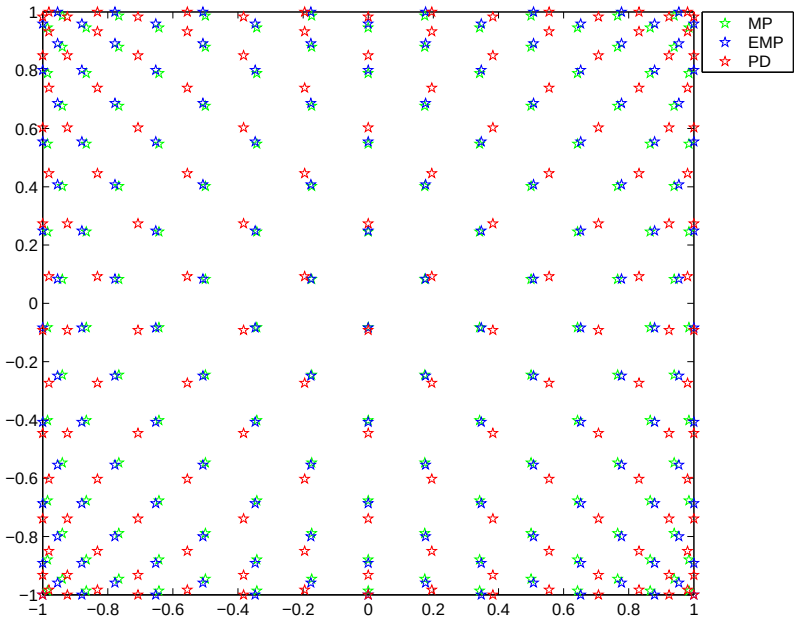
- The Padua points (PD) can be defined as follows (C.DeM.V. AMC 05):

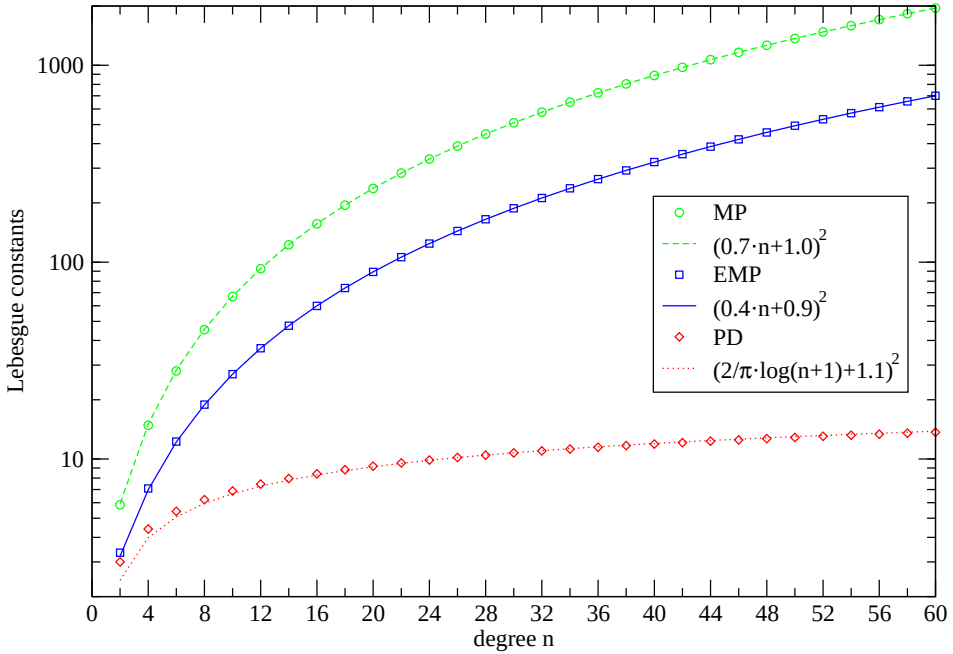
$$x_m^{PD} = \cos\left(\frac{(m-1)\pi}{n}\right), \quad y_k^{PD} = \begin{cases} \cos\left(\frac{(2k-1)\pi}{n+1}\right) & \text{if } m \text{ odd} \\ \cos\left(\frac{2(k-1)\pi}{n+1}\right) & \text{if } m \text{ even} \end{cases}$$

$1 \le m \le n+1$, $1 \le k \le n/2+1$, $N = (n+2)(n+1)/2$.

- 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.
- Moreover, there are four families which correspond to successive 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.

An interpolation formula for MP points

L. Bos in a note described an interpolation formula for MP points. Letting,

$$L_{m,k}(x,y) := \sum_{j=0}^{n} \mathbf{P}_j^T(x_m^{MP}, y_k^{MP})\mathbf{P}_j(x,y), \qquad 1\leq m\leq n+1,\ 1\leq k\leq n/2+1\ , \quad (1)$$

where

$$\mathbf{P}_j(x,y) = \begin{pmatrix} U_0(x)U_j(y) \\ U_1(x)U_{j-1}(y) \\ \vdots \\ U_j(x)U_0(y) \end{pmatrix}$$

$\mathbf{U_s}$ being the Chebyshev polynomials of second type, so that

$$L_{m,k}(x_s, y_r) = 0, \quad (s,r) \neq (m,k).$$

An interpolation formula for MP points

- $$\ell_{m,k}(x,y) := \frac{1}{\sum_{j=0}^{n} \mathbf{P}_j^T(x_m^{MP}, y_k^{MP})\mathbf{P}_j(x_m^{MP}, y_k^{MP})} L_{m,k}(x,y)\,, \quad (2)$$

  $m = 1, \ldots, n+1\,; k = 1, \ldots, n/2+1$, are the fundamental Lagrange polynomials; $\sum_{j=0}^{n} \mathbf{P}_j^T(x_m^{MP}, y_k^{MP})\mathbf{P}_j(x_m^{MP}, y_k^{MP}) \geq 1$. He also gave a naive upper bound (overestimate) for the Lebesgue constant

$$\Lambda_n^{MP} := \sum_{m=1}^{n+1} \sum_{k=1}^{n/2+1} |\ell_{m,k}(x,y)| \leq c_1 n^6,$$

  for appropriate $c_1 > 0$, while from our numerical results the growth is $\mathcal{O}(n^2)$ ($\approx (0.7n+1)^2$).

- The computational cost for evaluating the interpolant at any (x,y) is $\mathcal{O}(N^2)$.

An interpolation formula for MP points

- $$\ell_{m,k}(x,y) := \frac{1}{\sum_{j=0}^{n} \mathbf{P}_j^T(x_m^{MP}, y_k^{MP})\mathbf{P}_j(x_m^{MP}, y_k^{MP})} L_{m,k}(x,y)\,, \quad (2)$$

  $m = 1, \ldots, n+1\,; k = 1, \ldots, n/2+1$, are the fundamental Lagrange polynomials; $\sum_{j=0}^{n} \mathbf{P}_j^T(x_m^{MP}, y_k^{MP})\mathbf{P}_j(x_m^{MP}, y_k^{MP}) \geq 1$. He also gave a naive upper bound (overestimate) for the Lebesgue constant

  $$\Lambda_n^{MP} := \sum_{m=1}^{n+1} \sum_{k=1}^{n/2+1} |\ell_{m,k}(x,y)| \leq c_1 n^6,$$

  for appropriate $c_1 > 0$, while from our numerical results the growth is $\mathcal{O}(n^2)$ ($\approx (0.7n+1)^2$).

- The computational cost for evaluating the interpolant at any (x,y) is $\mathcal{O}(N^2)$.

Generating curves

- For MP points

$$\gamma_{\mathrm{MP}_n}(t) = \left[\cos\left(\frac{t\pi}{n+2}\right), \cos\left(\frac{(n+3-t)\pi}{n+3}\right)\right], \quad 0 \le t \le (n+2)(n+3)\,; \tag{3}$$

- For the PD points of the first family

$$\gamma_{\mathrm{PD}_n}(t) = \left[-\cos\left(\frac{t\pi}{n}\right), -\cos\left(\pi - \frac{t\pi}{n+1}\right)\right], \quad 0 \le t \le n(n+1)\,. \tag{4}$$

The interpolant of the PD pts

Let $A_j = (x_j, y_j) \in [-1,1]^2$ be a PD, it belongs to the curve,
$x = -\cos(n+1)t, \quad y = -\cos nt\,, \ 0 \le t \le \pi.$

$$\begin{aligned} K_n(\mathbf{x},\mathbf{y}) &= D_n(\theta_1+\phi_1,\theta_2+\phi_2) + D_n(\theta_1+\phi_1,\theta_2-\phi_2) + \\ &+ D_n(\theta_1-\phi_1,\theta_2+\phi_2) + D_n(\theta_1-\phi_1,\theta_2-\phi_2)\,, \\ \mathbf{x} &= (\cos\theta_1,\cos\theta_2), \quad \mathbf{y} = (\cos\phi_1,\cos\phi_2)\,, \end{aligned} \tag{5}$$

where the function D_n is defined by

$$D_n(\alpha,\beta) = \frac{1}{2}\,\frac{\cos((n+1/2)\alpha)\cos(\alpha/2) - \cos((n+1/2)\beta)\cos(\beta/2)}{\cos\alpha - \cos\beta}\,. \tag{6}$$

The **Lagrange polynomials** are

$$l_j(x,y) = w_j\,\{K_n((x,y),A_j) - T_n(x_j)\,T_n(x)\}$$

and the coefficients w_j are the corresponding cubature weights.

The interpolant of the PD pts

$$w_j = \frac{1}{n(n+1)} \begin{cases} 1/2 & \text{vertex pts} \\ 2 & \text{interior pts} \\ 1 & \text{boundary pts} \end{cases}$$

- K_n is the reproducing kernel of $\mathbb{P}_n([-1,1]^2)$ (Xu, JAT 95) equipped with the scalar product

$$< f, g >= \frac{1}{\pi^2} \int_{[-1,1]^2} f(x,y) g(x,y) \frac{dx}{\sqrt{1-x^2}} \frac{dy}{\sqrt{1-y^2}}$$

- holds the reproducing property

$$< p(x,y), K_n((x,y),A) >= p(A), \quad \forall p \in \mathbb{P}_n(\mathbb{R}^2)$$

and $A = (a,b) \in [-1,1]^2$.

The Xu points and the Xu interpolant

- Given the Chebyshev-Lobatto points on the interval $[-1, 1]$ (cf. Xu, JAT 95)

$$\xi_k = \xi_{k,n} = \cos \frac{k\pi}{n}, \quad k = 0, \ldots, n, \quad n = 2m \ ,$$

the **Xu interpolation points** on the square $Q = [-1, 1]^2$ are the two-dimensional Chebyshev array $X_N = \{\mathbf{z}_{r,s}\}$ of dimension $N = n(n+2)/2$

$$\begin{aligned} \mathbf{z}_{2i,2j+1} &= (\xi_{2i}, \xi_{2j+1}), \quad 0 \le i \le m, \ 0 \le j \le m-1 \ , \\ \mathbf{z}_{2i+1,2j} &= (\xi_{2i+1}, \xi_{2j}), \quad 0 \le i \le m-1, \ 0 \le j \le m \ . \end{aligned}$$

Note: the Xu points are exactly equally spaced w.r.t. the Dubiner metric.

The Xu points and the Xu interpolant

- Given the Chebyshev-Lobatto points on the interval $[-1, 1]$ (cf. Xu, JAT 95)

$$\xi_k = \xi_{k,n} = \cos \frac{k\pi}{n}, \quad k = 0, \ldots, n, \quad n = 2m \ ,$$

the **Xu interpolation points** on the square $Q = [-1, 1]^2$ are the two-dimensional Chebyshev array $X_N = \{\mathbf{z}_{r,s}\}$ of dimension $N = n(n+2)/2$

$$\begin{aligned} \mathbf{z}_{2i,2j+1} &= (\xi_{2i}, \xi_{2j+1}), \quad 0 \le i \le m, \ 0 \le j \le m-1 \ , \\ \mathbf{z}_{2i+1,2j} &= (\xi_{2i+1}, \xi_{2j}), \quad 0 \le i \le m-1, \ 0 \le j \le m \ . \end{aligned}$$

Note: the Xu points are exactly equally spaced w.r.t. the Dubiner metric.

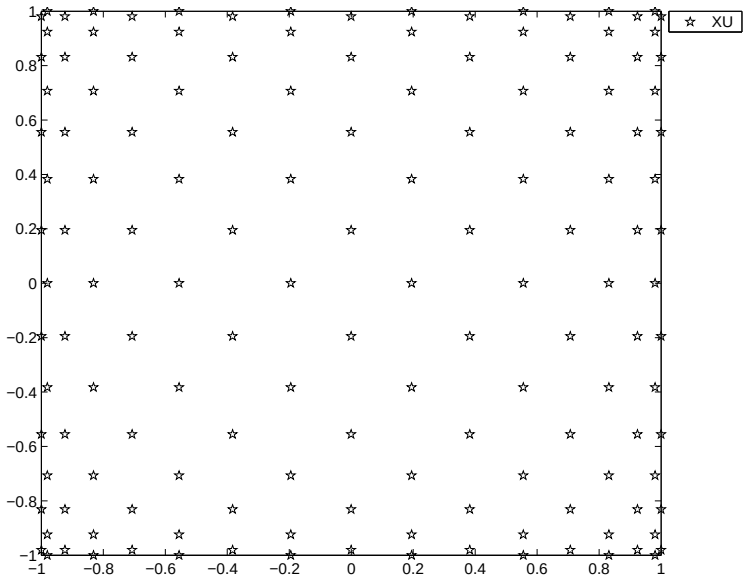
The Xu points and the Xu interpolant

The **Xu interpolant** of degree n in Lagrange form of a function f on Q is

$$L_n^{\mathrm{Xu}} f(\mathbf{x}) = \sum_{\mathbf{z}_{r,s} \in X_N} f(\mathbf{z}_{r,s})\, \ell_n(\mathbf{x}, \mathbf{z}_{r,s}), \quad \ell_n(\mathbf{x}, \mathbf{z}_{r,s}) := \frac{K_n^*(\mathbf{x}, \mathbf{z}_{r,s})}{K_n^*(\mathbf{z}_{r,s}, \mathbf{z}_{r,s})}\,, \qquad (7)$$

$$K_n^*(\mathbf{z}_{r,s}, \mathbf{z}_{r,s}) = \frac{1}{2}\left(K_{n-1}(\mathbf{z}_{r,s}, \mathbf{z}_{r,s}) + K_n(\mathbf{z}_{r,s}, \mathbf{z}_{r,s})\right) - 1\,, \qquad (8)$$

with K_n as defined for the PD points in (5).



The Xu points and the Xu interpolant

Remarks

- In the Xu interpolation formula, the ℓ_n in (7) are based on the K_n^* (cf. (8)), that make use of K_{n-1} and K_n: the interpolant based on the PD points only need the K_n (i.e no K_n^*).
- On the other hand, the dimension of the corresponding polynomial space, V_n, is
$\dim(\mathbb{P}_{n-1}(\mathbb{R}^2)) < \dim(V_n) := n(n+2)/2 < \dim(\mathbb{P}_n(\mathbb{R}^2))$.
- **Drawback:** numerical instability in computing $D_n(\alpha, \beta)$ when $\cos\alpha \approx \cos\beta$!

The Xu points and the Xu interpolant

Remarks

- In the Xu interpolation formula, the ℓ_n in (7) are based on the K_n^* (cf. (8)), that make use of K_{n-1} and K_n: the interpolant based on the PD points only need the K_n (i.e no K_n^*).
- On the other hand, the dimension of the corresponding polynomial space, V_n, is
$\dim(\mathbb{P}_{n-1}(\mathbb{R}^2)) < \dim(V_n) := n(n+2)/2 < \dim(\mathbb{P}_n(\mathbb{R}^2))$.
- **Drawback:** numerical instability in computing $D_n(\alpha, \beta)$ when $\cos\alpha \approx \cos\beta$!

The Xu points and the Xu interpolant

Remarks

- In the Xu interpolation formula, the ℓ_n in (7) are based on the K_n^* (cf. (8)), that make use of K_{n-1} and K_n: the interpolant based on the PD points only need the K_n (i.e no K_n^*).
- On the other hand, the dimension of the corresponding polynomial space, V_n, is
$\dim(\mathbb{P}_{n-1}(\mathbb{R}^2)) < \dim(V_n) := n(n+2)/2 < \dim(\mathbb{P}_n(\mathbb{R}^2))$.
- **Drawback:** numerical instability in computing $D_n(\alpha, \beta)$ when $\cos\alpha \approx \cos\beta$!

The Xu points and the Xu interpolant

- Stabilization

$$D_n(\alpha,\beta) = \frac{1}{4}\left[U_{n-1}(\cos\phi)U_{n-1}(\cos\psi) + U_{n-2}(\cos\phi)U_{n-2}(\cos\psi)\right],$$

where $\phi = (\alpha-\beta)/2$, $\psi = (\alpha+\beta)/2$, U_n Chebyshev polynomial of the second kind computed by the three-term recurrence, with overall computational cost $\approx 8nN \approx 11\,N^{3/2}$ flops.

- Hybrid stable formula for $U_n(\cos\theta)$: three-term recurrence whenever $|\theta - k\pi| \le \varepsilon$, otherwise $U_n(\cos\theta) = \sin(n+1)\theta/\sin\theta$. For $\varepsilon = 0.01$, $L_n^{\mathrm{Xu}} f(\mathbf{x})$ is computed at machine precision, the recurrence relation is used globally less than 1%, and for degrees n up to the hundreds, overall computational cost $\approx 32 c_{\sin} N$ flops, $c_{\sin}$ being the average evaluation cost of the sine function.
- In practical applications the computational cost becomes linear in the number N of Xu points
- We made a Fortran implementation of the Xu and PD interpolation

The Xu points and the Xu interpolant

- Stabilization

$$D_n(\alpha,\beta) = \frac{1}{4}\left[U_{n-1}(\cos\phi)U_{n-1}(\cos\psi) + U_{n-2}(\cos\phi)U_{n-2}(\cos\psi)\right],$$

where $\phi = (\alpha-\beta)/2$, $\psi = (\alpha+\beta)/2$, U_n Chebyshev polynomial of the second kind computed by the three-term recurrence, with overall computational cost $\approx 8nN \approx 11\,N^{3/2}$ flops.

- Hybrid stable formula for $U_n(\cos\theta)$: three-term recurrence whenever $|\theta - k\pi| \le \varepsilon$, otherwise $U_n(\cos\theta) = \sin(n+1)\theta/\sin\theta$. For $\varepsilon = 0.01$, $L_n^{\mathrm{Xu}} f(\mathbf{x})$ is computed at machine precision, the recurrence relation is used globally less than 1%, and for degrees n up to the hundreds, overall computational cost $\approx 32 c_{\sin} N$ flops, $c_{\sin}$ being the average evaluation cost of the sine function.
- In practical applications the computational cost becomes linear in the number N of Xu points
- We made a Fortran implementation of the Xu and PD interpolation

The Xu points and the Xu interpolant

- Stabilization

$$D_n(\alpha,\beta) = \frac{1}{4}\left[U_{n-1}(\cos\phi)U_{n-1}(\cos\psi) + U_{n-2}(\cos\phi)U_{n-2}(\cos\psi)\right] ,$$

where $\phi = (\alpha-\beta)/2$, $\psi = (\alpha+\beta)/2$, U_n Chebyshev polynomial of the second kind computed by the three-term recurrence, with overall computational cost $\approx 8nN \approx 11\,N^{3/2}$ flops.

- Hybrid stable formula for $U_n(\cos\theta)$: three-term recurrence whenever $|\theta - k\pi| \le \varepsilon$, otherwise $U_n(\cos\theta) = \sin(n+1)\theta/\sin\theta$. For $\varepsilon = 0.01$, $L_n^{\mathrm{Xu}} f(\mathbf{x})$ is computed at machine precision, the recurrence relation is used globally less than 1%, and for degrees n up to the hundreds, overall computational cost $\approx 32 c_{\sin} N$ flops, $c_{\sin}$ being the average evaluation cost of the sine function.
- In practical applications the computational cost becomes linear in the number N of Xu points
- We made a Fortran implementation of the Xu and PD interpolation

The Xu points and the Xu interpolant

Implementation details and performance (cf. B.C.DeM.V. '05):

- Comparison with the MPI software by T. Sauer (cf. S. AiCM 95, S. Xu Math.Comp.95). The MPI software is one of the most efficient and robust implementations of multivariate interpolation by polynomials.
- We compared the CPU times necessary to build the interpolant and the interpolation errors for both XU and MPI on many tests functions.

The Xu points and the Xu interpolant

Table 1: CPU times (secs.) and interpolation errors on $[0,1]^2$ for the Franke funct.

n	20	30	40	50	60
XU	2.1 7.3E-03	5.2 3.6E-04	10.3 3.1E-06	17.8 1.8E-08	28.4 2.5E-11
MPI	0.6 3.8E-02	Unsolv. * * *	Unsolv. * * *	Unsolv. * * *	Unsolv. * * *

Table 2: CPU times and interpolation errors of MPI for the Franke function on different domains by a change of variables and reordering the points as Leja sequences.

n	20	30	40	50	60
MPI $[-1,1]^2$	0.6 6.3E-03	4.3 3.5E-04	21.0 2.0E-01	75.6 3.8E-02	Unsolv. * * *
MPI $[-2,2]^2$	0.5 6.4E-03	3.7 1.0E-02	17.4 2.7E+02	62.3 1.3E+14	183.4 1.9E+35
MPI-Leja $[-1,1]^2$	0.6 6.4E-03	4.3 3.5E-04	21.0 1.1E-04	75.6 2.0E-03	Unsolv. * * *

The Lebesgue constant of the Xu points

Since, the maximum is attained at the four vertices of the square, the computation became "easy"

Table 1. Lebesgue constants size of different nodal sets on Q: Morrow-Patterson (MP), Extended Morrow-Patterson (EMP), Padua points (PD), Xu points (XU).

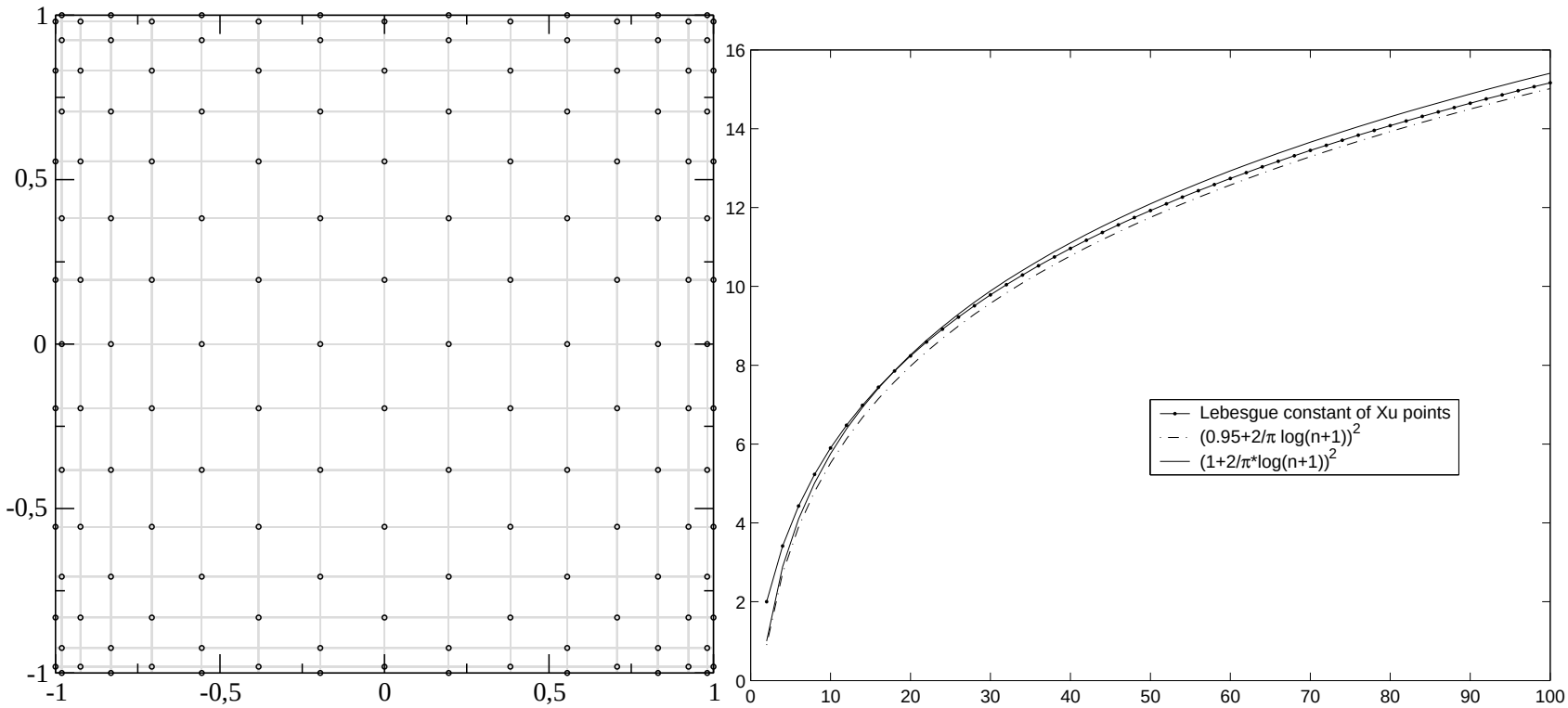
interp. pts.	Λ_{34}	Λ_{48}	Λ_{62}	Λ_{76}
MP	649	1264	2082	3102
EMP	237	456	746	1106
PD	11	13	14	15
XU	10	12	13	14

The Lebesgue constant of the Xu points

$$\begin{aligned} \Lambda_n^{Xu} &\leq 2\left\{2+4\left(\frac{2}{\pi}\log n+5\right)^2\right\} \\ &= 8\left(\frac{2}{\pi}\log n+5\right)^2+4\,. \end{aligned}$$

The Lebesgue constant of the Xu points

Figure 1. Left: the distribution of 144 Xu points on Q (i.e $n = 16$). Right: the behavior of the Lebesgue constant up to degree $n = 100$.



Applications of the Xu interpolant

We studied two main applications

1. We compressed a surface given as a large set of scattered data, i.e. by "interpolated interpolations".
2. Compression of a Finite Element PDE solution.

- Concerning 1. we adopted for sufficiently regular surfaces Xu-like interpolation of a cubic Shepard-like interpolant (cf. Renka TOMS99). The compression ratio obtained is

$$\text{compr. ratio} = 3\times \frac{\text{numb. of scatt. pts.}}{\text{numb. of Xu nodes}} \approx 6\times \frac{\text{numb. of scatt. pts.}}{n^2},$$

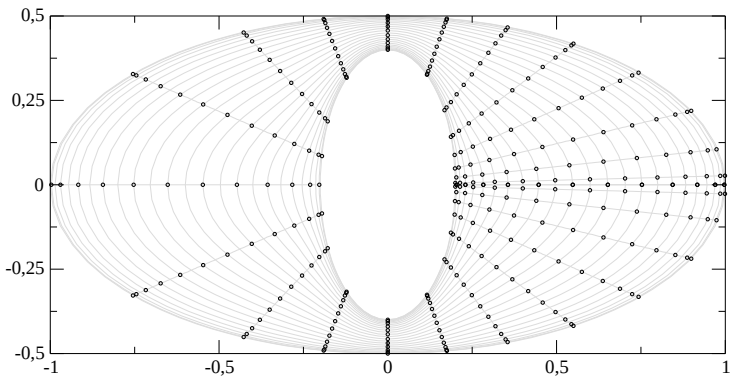
  where n is the polynomial degree.

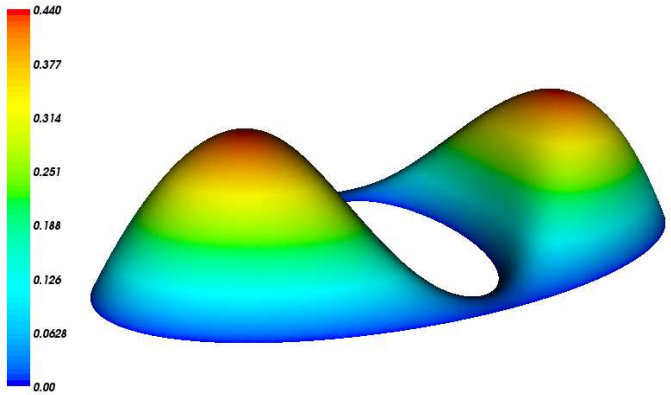
- Concerning 2. Given a FEM discretization (ex. Delaunay mesh) we used Xu-like interpolation of the Finite Element solution.

Applications of the Xu interpolant

Table 3. Compression errors (in the max-norm) for the Finite Element solution of the Poisson equation $\Delta f(\mathbf{x}) = -10\ ,\ \mathbf{x} \in \Omega\,;\ f(\mathbf{x}) = 0\ , \mathbf{x} \in \partial\Omega$, where Ω is the "lynx-eye" shaped domain in the Fig. below.

mesh size	$n = 8$	$n = 12$	$n = 16$	$n = 20$	$n = 24$	$n = 28$	$n = 32$
41402	1E-1	3E-2	1E-2	5E-3	2E-3	1E-3	1E-3

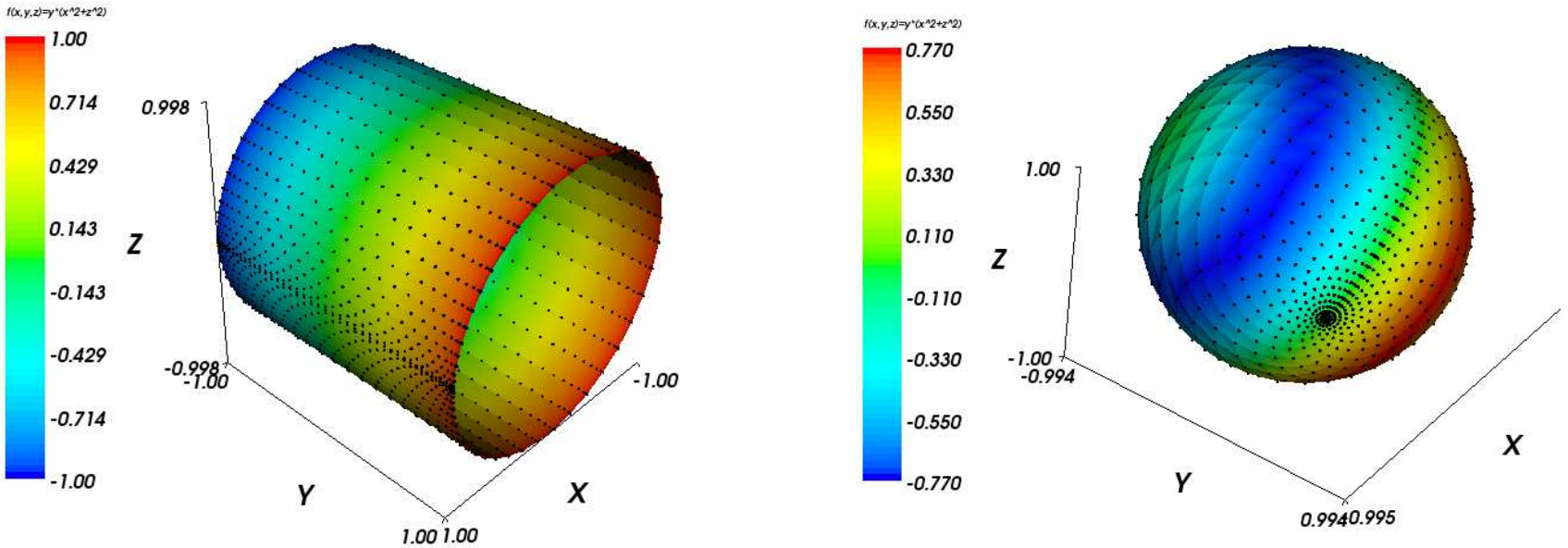




Left: the distribution of $N = 312$ Xu-like points (deg $n = 24$) in the "lynx-eye" shaped domain (generalized sector). Right: Plot of the Xu-like interpolated solution (deg $n = 24$: compression ratio $\approx$ 400:1, compression error $\approx 2 \cdot 10^{-3}$).

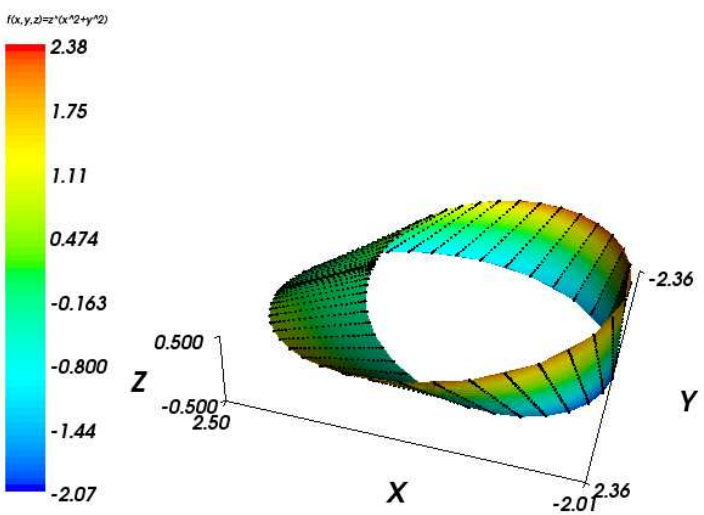
Applications of the Xu interpolant

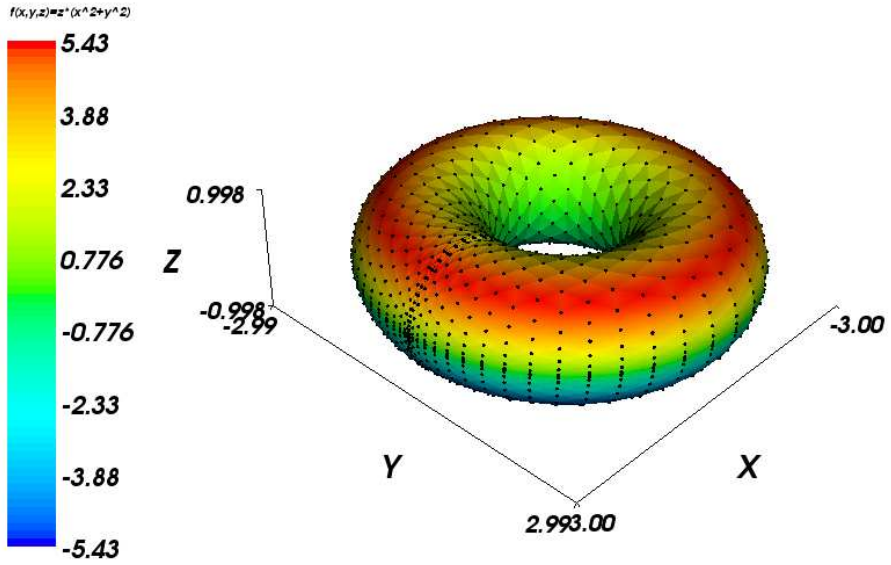
Recently we applied the Xu interpolation to functions in parametric form $f(x(u,v), y(u,v), z(u,v))$, where $u \in [a,b], \quad v \in [c,d]$. Here some interesting pictures



Left: Xu points over the cilinder and the function to be interpolated $f(x,y,z) = y(x^2+z^2)$. Right: The same for the sphere.

Applications of the Xu interpolant





Bivariate interpolation problem and Padua Pts

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 Pts

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 Pts

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 Pts

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 Pts

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 Pts

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 $[-1,1]$

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

and the two subsets of points with odd or even indexes

$$C_{n+1}^{\text{odd}} = \left\{ z_j^n,\ j = 1,\dots,n+1,\ j \text{ odd} \right\}$$
$$C_{n+1}^{\text{even}} = \left\{ z_j^n,\ j = 1,\dots,n+1,\ j \text{ even} \right\}$$

Then, the Padua points are the set

$$\text{Pad}_n = C_{n+1}^{\text{odd}} \times C_{n+2}^{\text{even}} \cup C_{n+1}^{\text{even}} \times C_{n+2}^{\text{odd}} \subset C_{n+1} \times C_{n+2}$$

Padua points

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

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

and the two subsets of points with odd or even indexes

$$C_{n+1}^{\text{odd}} = \left\{ z_j^n,\ j = 1,\dots,n+1,\ j \text{ odd} \right\}$$
$$C_{n+1}^{\text{even}} = \left\{ z_j^n,\ j = 1,\dots,n+1,\ j \text{ even} \right\}$$

Then, the Padua points are the set

$$\text{Pad}_n = C_{n+1}^{\text{odd}} \times C_{n+2}^{\text{even}} \cup C_{n+1}^{\text{even}} \times C_{n+2}^{\text{odd}} \subset C_{n+1} \times C_{n+2}$$

Padua points

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

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

and the two subsets of points with odd or even indexes

$$C_{n+1}^{\text{odd}} = \left\{ z_j^n,\ j = 1,\dots,n+1,\ j \text{ odd} \right\}$$
$$C_{n+1}^{\text{even}} = \left\{ z_j^n,\ j = 1,\dots,n+1,\ j \text{ even} \right\}$$

Then, the Padua points are the set

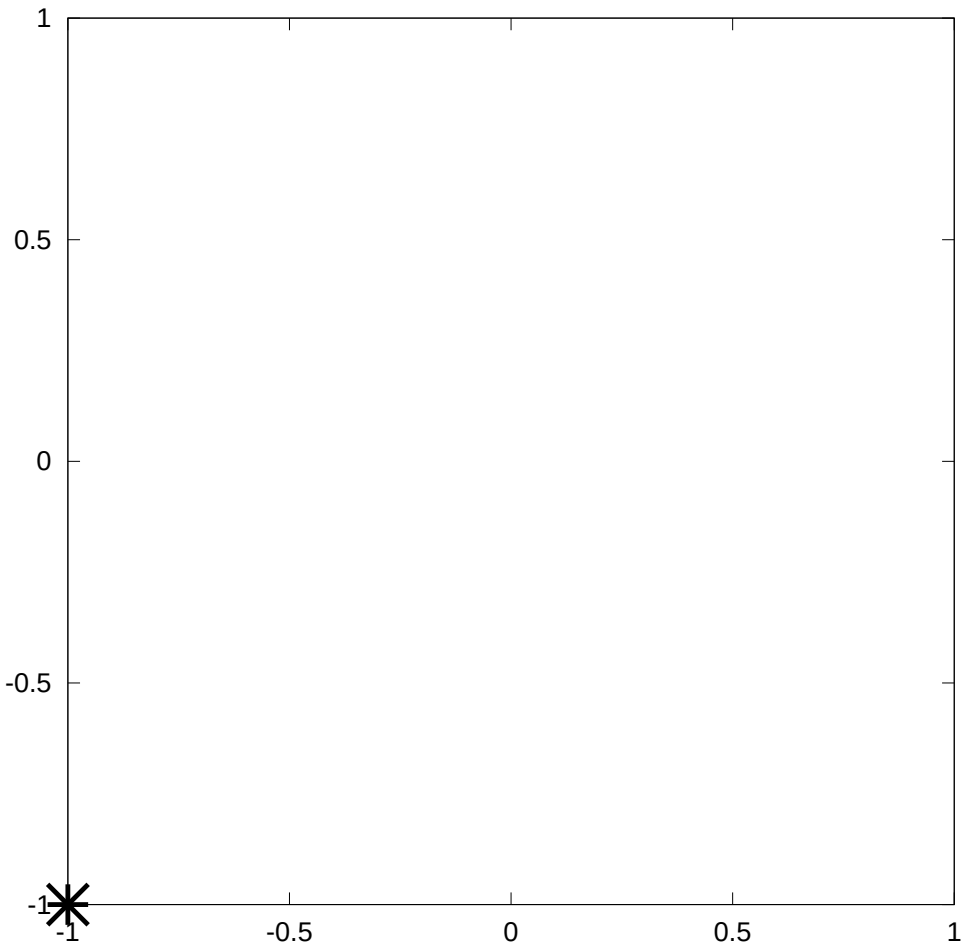
$$\text{Pad}_n = C_{n+1}^{\text{odd}} \times C_{n+2}^{\text{even}} \cup C_{n+1}^{\text{even}} \times C_{n+2}^{\text{odd}} \subset C_{n+1} \times C_{n+2}$$

The generating curve

There exists an alternative representation as self-intersections and boundary contacts of the generating curve

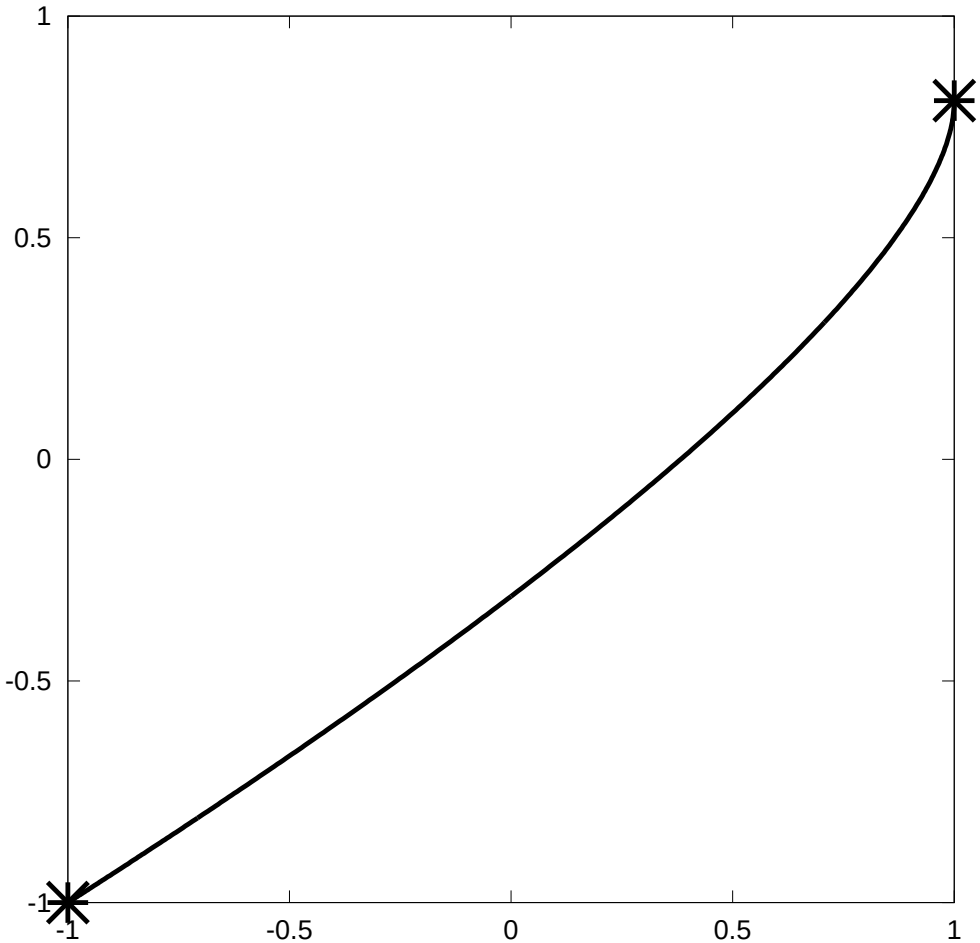
$$\gamma(t) = (-\cos((n+1)t), -\cos(nt)), \quad t \in [0, \pi]$$

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



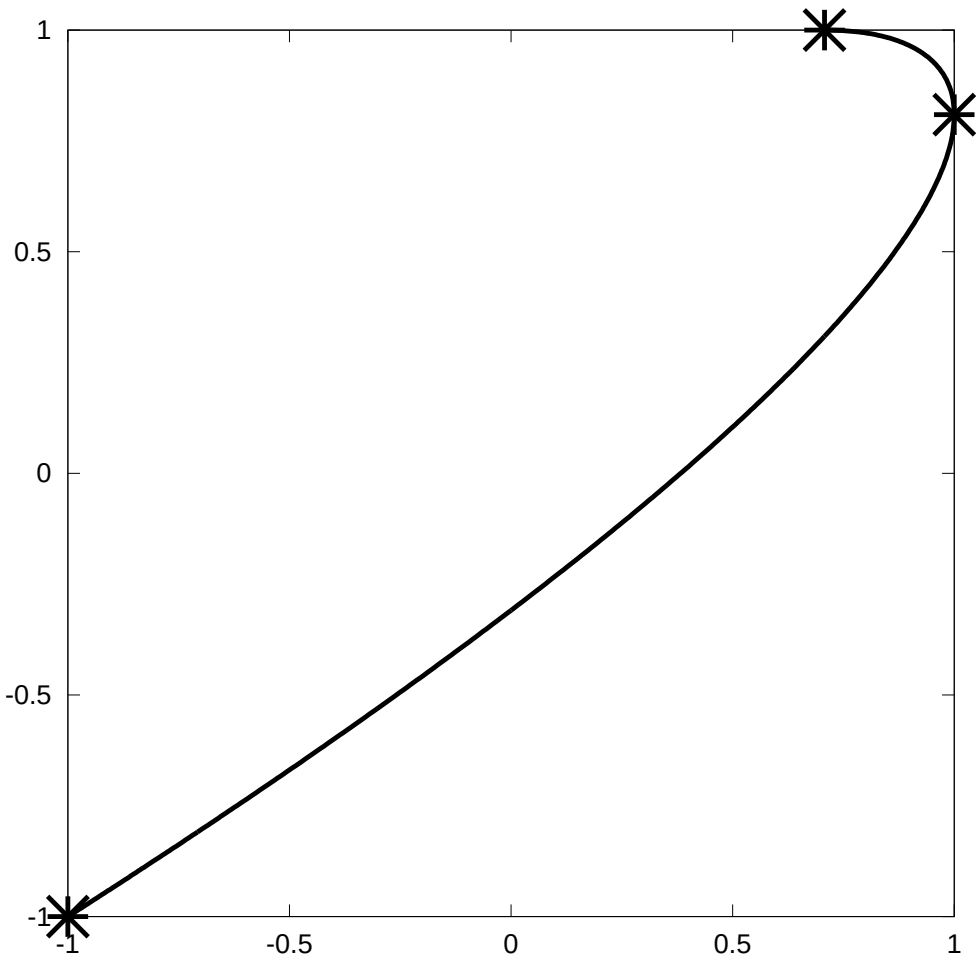
$t = 0$

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



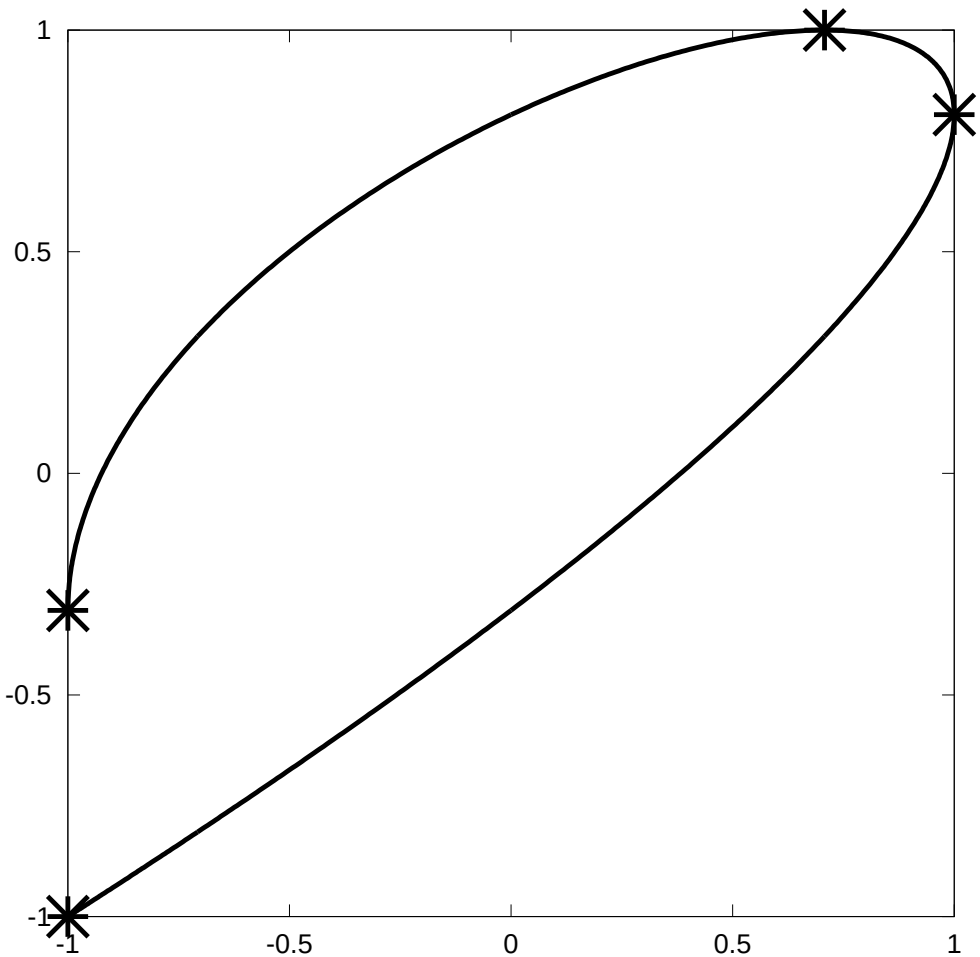
$$t \in \left[0, \frac{4\pi}{(n(n+1))}\right]$$

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



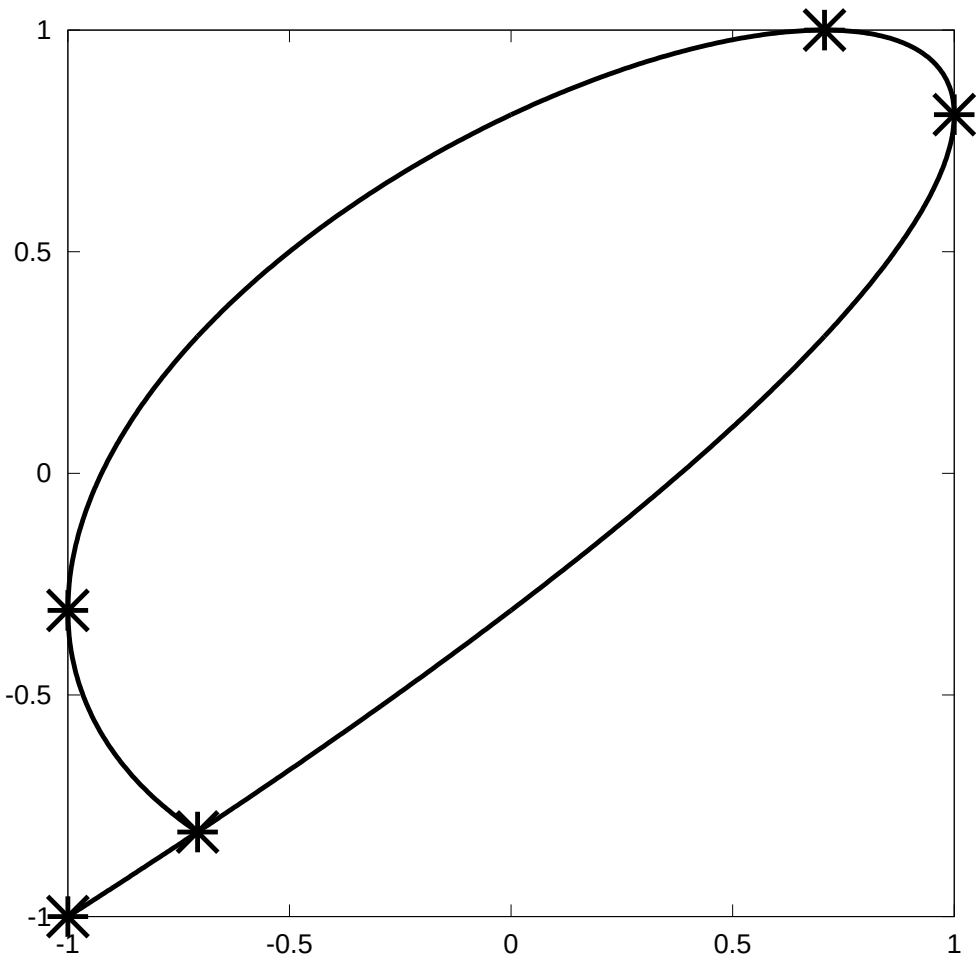
$$t \in \left[\frac{4\pi}{(n(n+1))}, \frac{5\pi}{(n(n+1))} \right]$$

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



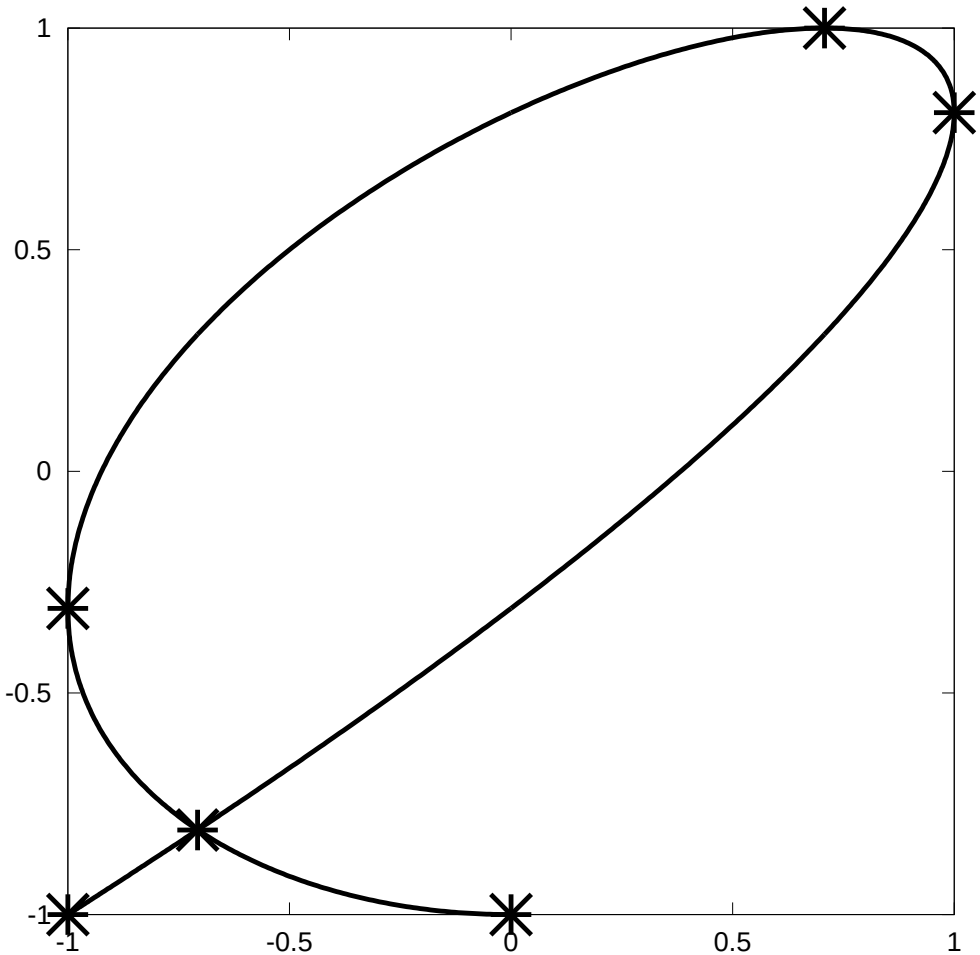
$$t \in \left[\frac{5\pi}{(n(n+1))}, \frac{8\pi}{(n(n+1))} \right]$$

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



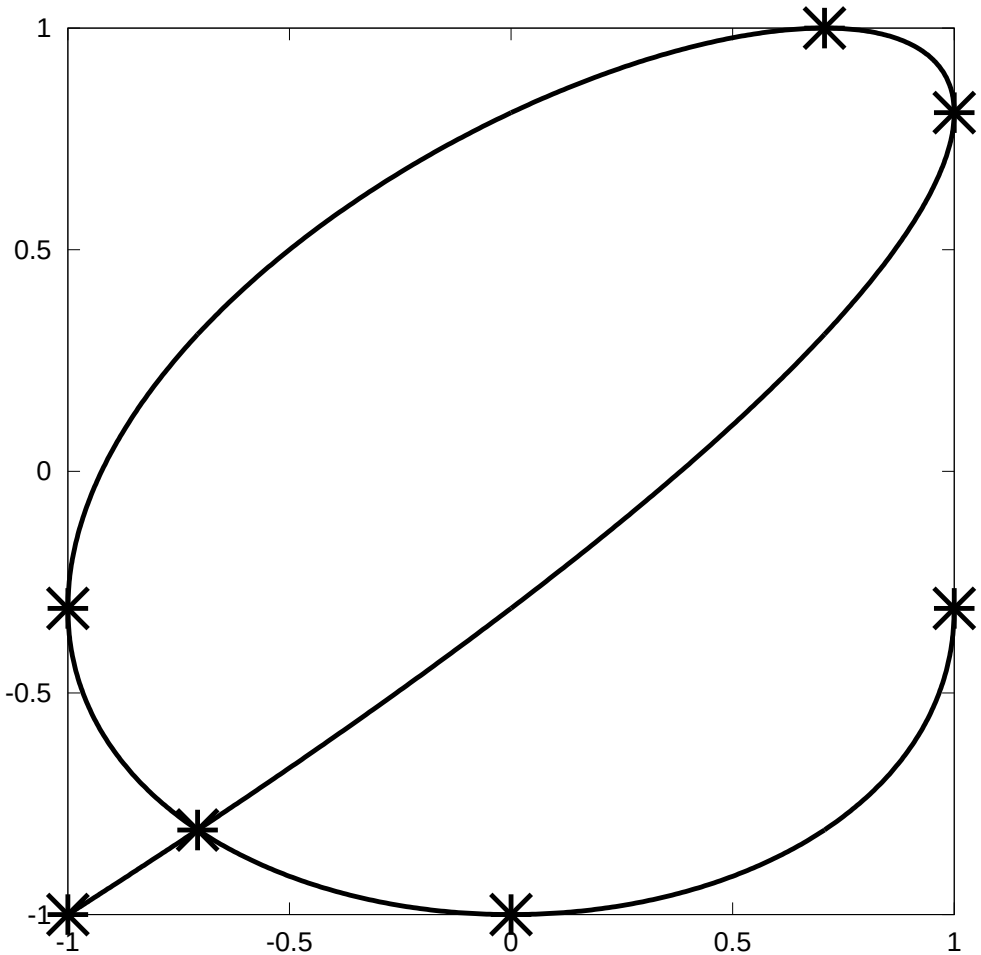
$$t \in \left[\frac{8\pi}{(n(n+1))}, \frac{9\pi}{(n(n+1))} \right]$$

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



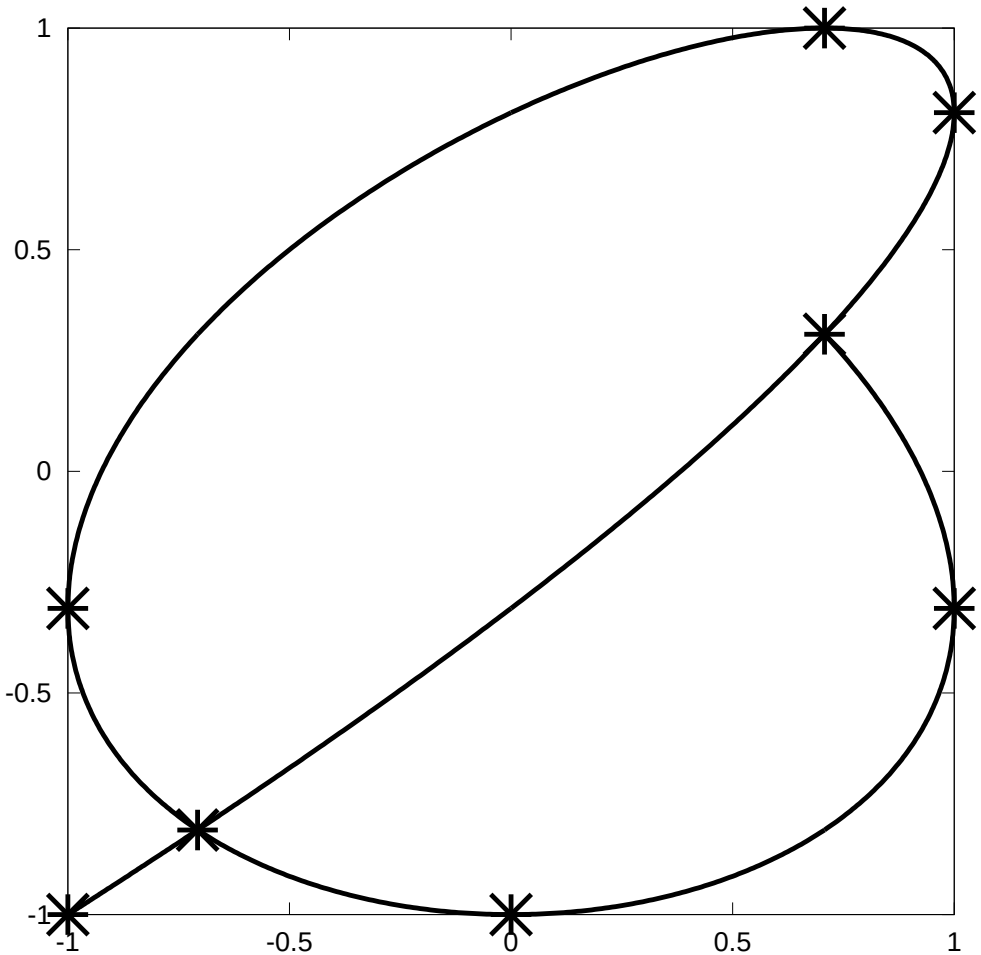
$$t \in \left[\frac{9\pi}{(n(n+1))}, \frac{10\pi}{(n(n+1))} \right]$$

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



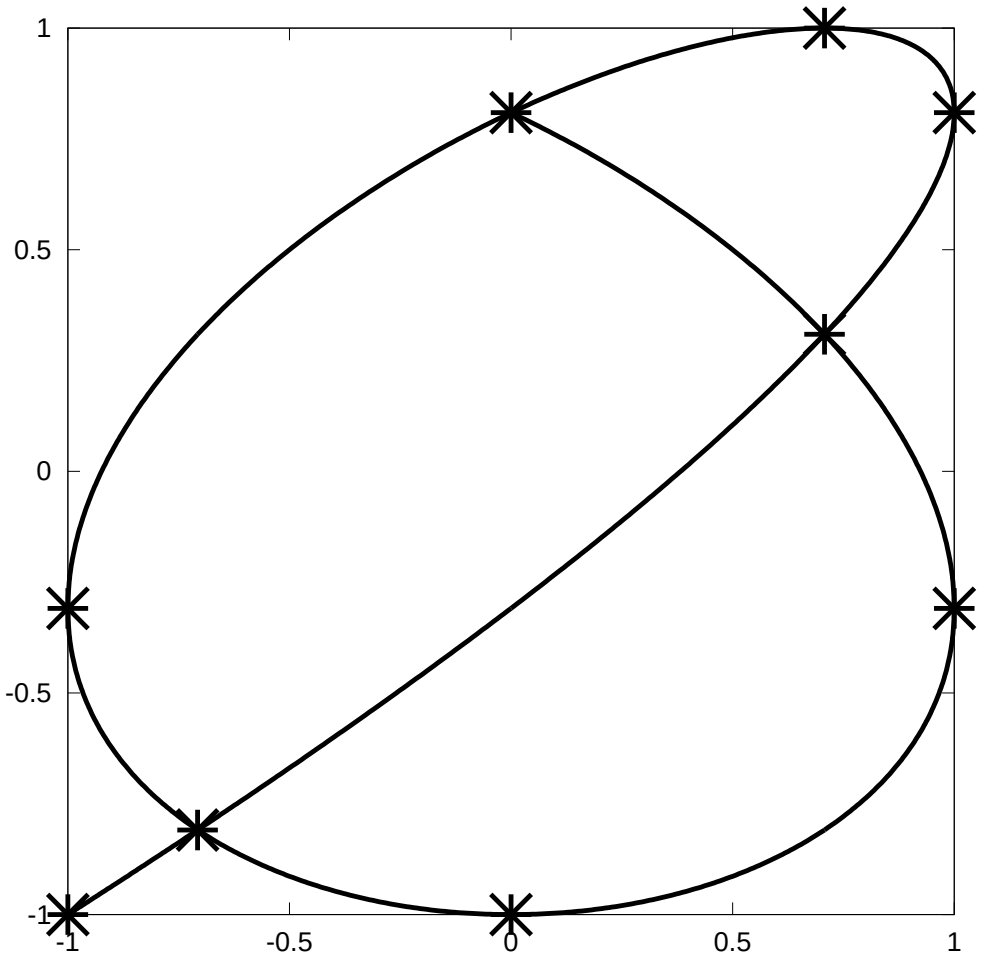
$$t \in \left[\frac{10\pi}{(n(n+1))}, \frac{12\pi}{(n(n+1))} \right]$$

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



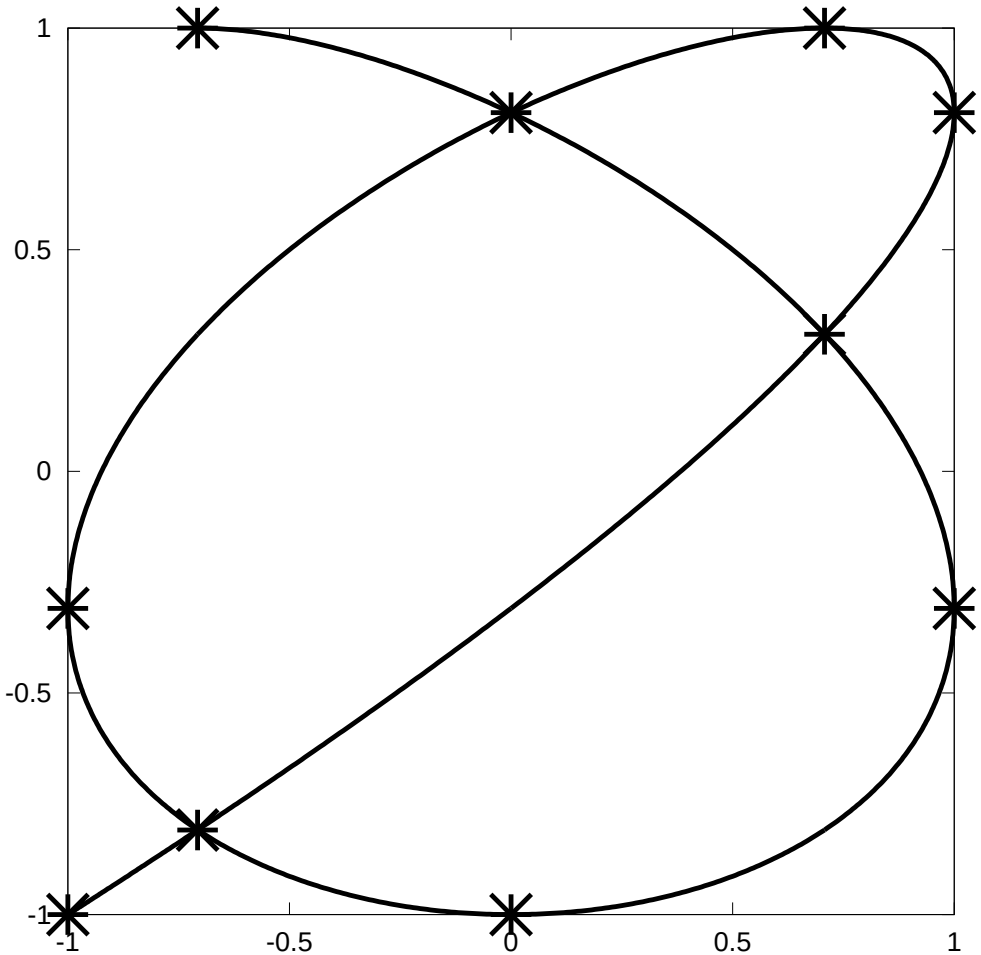
$$t \in \left[\frac{12\pi}{(n(n+1))}, \frac{13\pi}{(n(n+1))} \right]$$

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



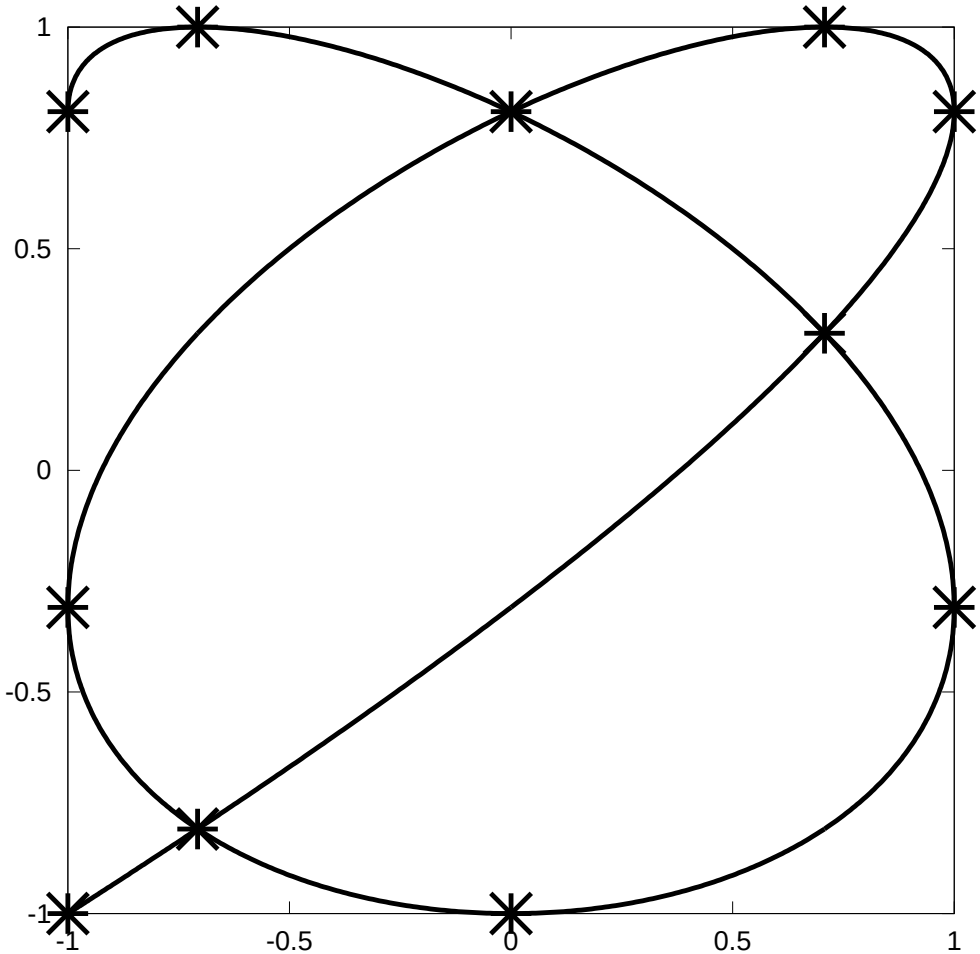
$$t \in \left[\frac{13\pi}{(n(n+1))}, \frac{14\pi}{(n(n+1))} \right]$$

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



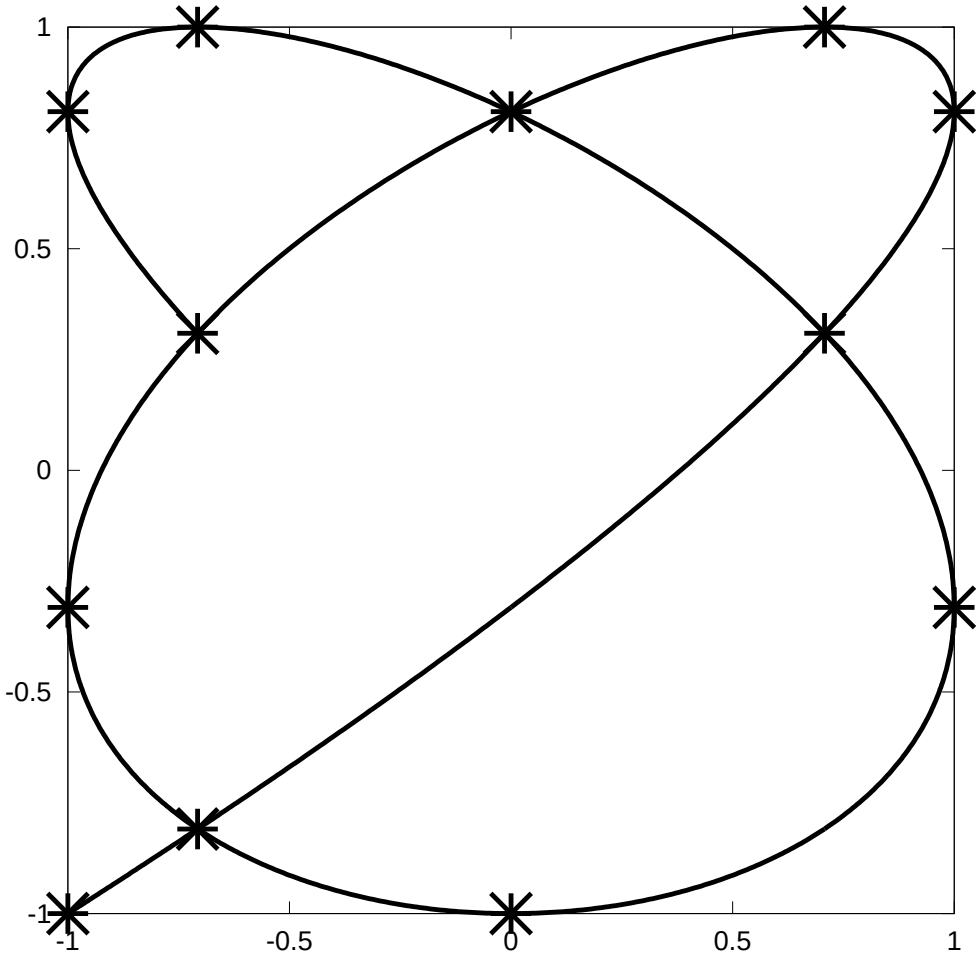
$$t \in \left[\frac{14\pi}{(n(n+1))}, \frac{15\pi}{(n(n+1))} \right]$$

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



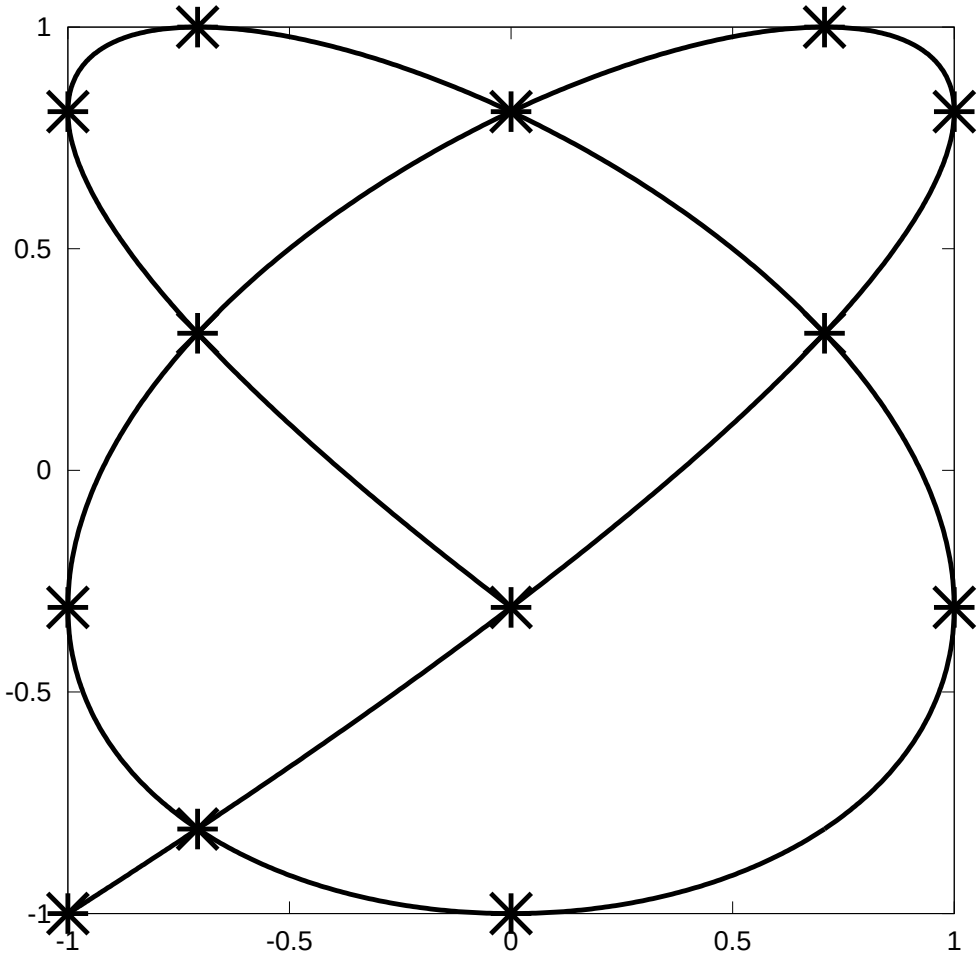
$$t \in \left[\frac{15\pi}{(n(n+1))}, \frac{16\pi}{(n(n+1))} \right]$$

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



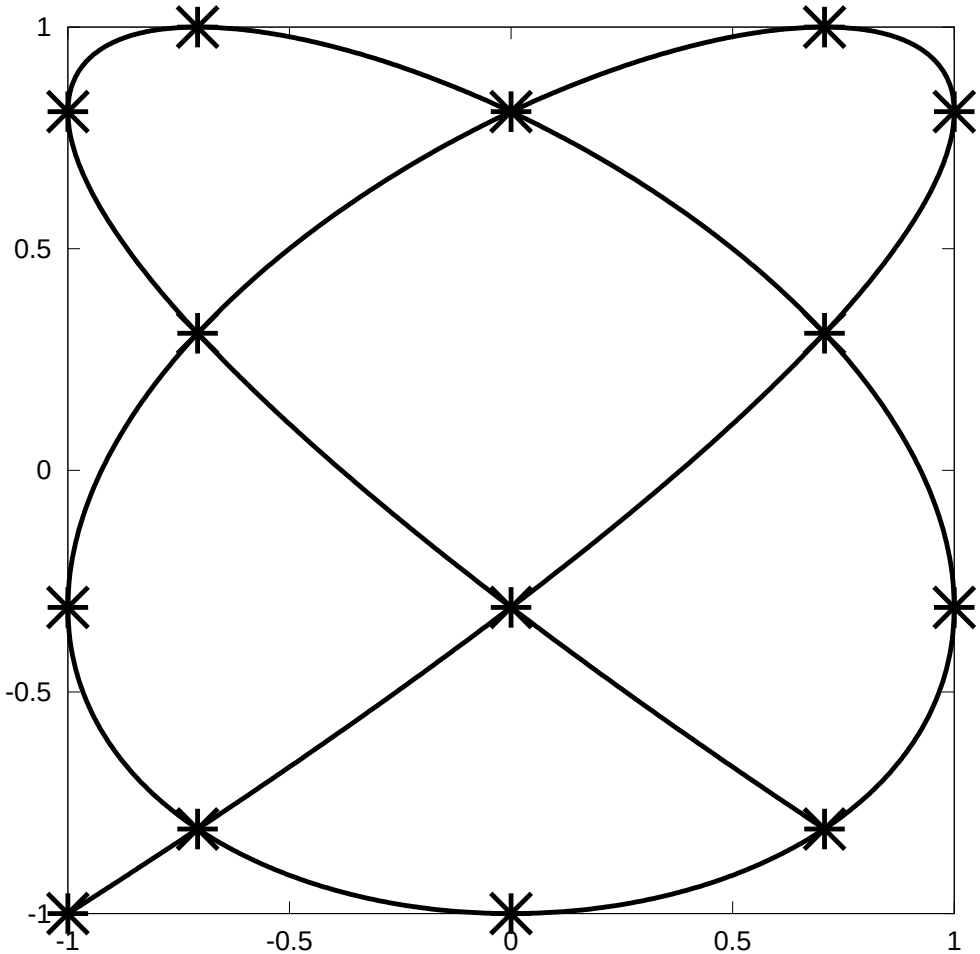
$$t \in \left[\frac{16\pi}{(n(n+1))}, \frac{17\pi}{(n(n+1))} \right]$$

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



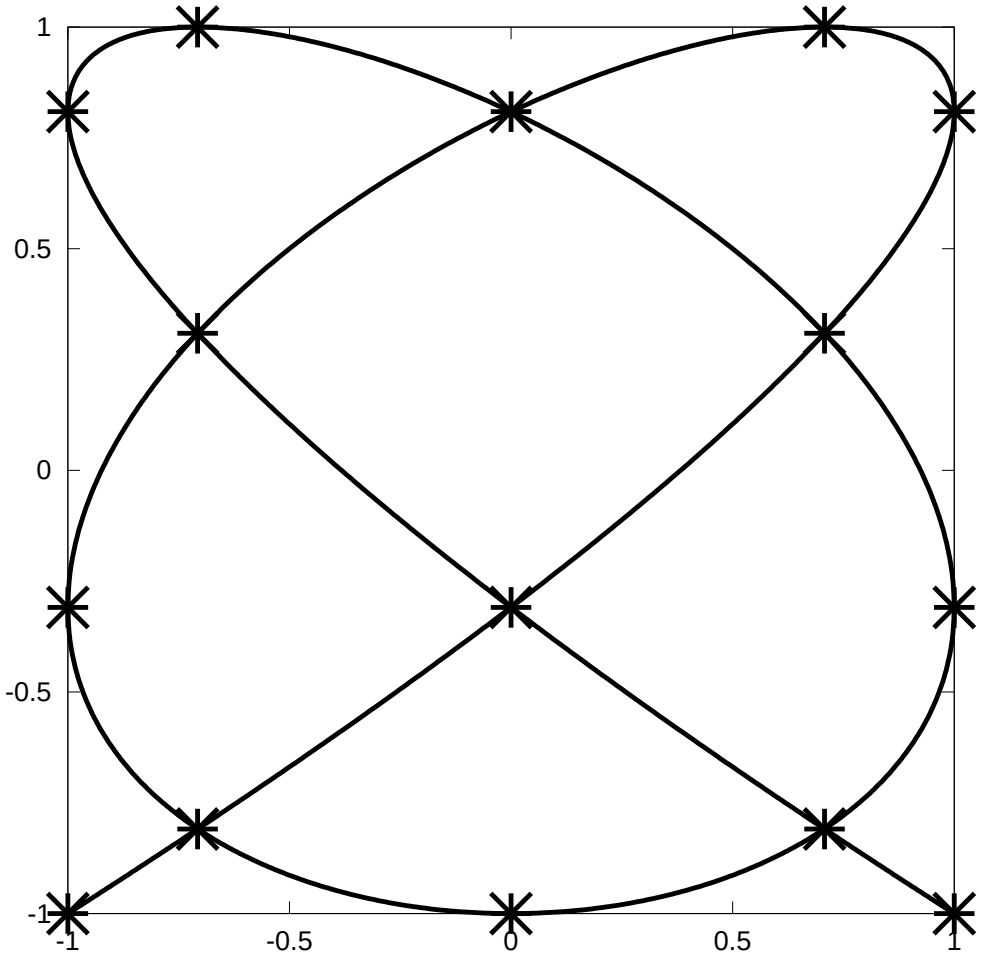
$$t \in \left[\frac{17\pi}{(n(n+1))}, \frac{18\pi}{(n(n+1))} \right]$$

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



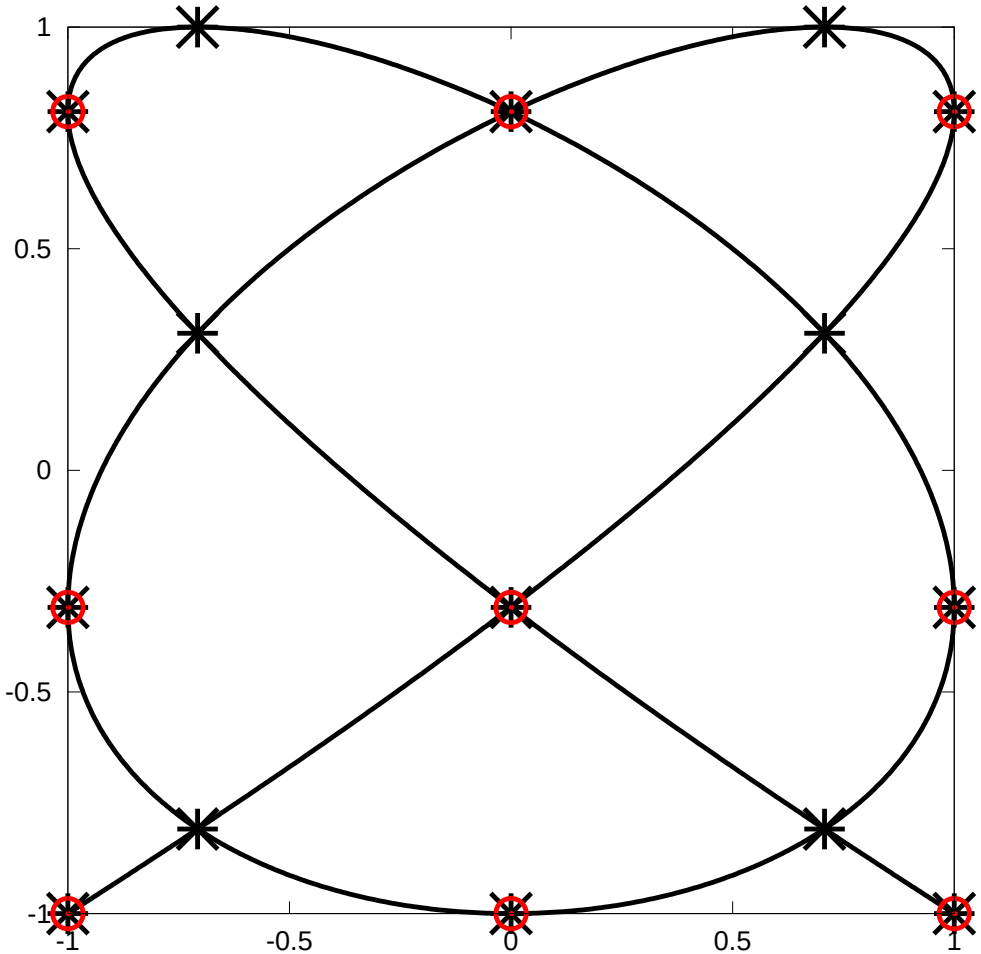
$$t \in \left[\frac{18\pi}{(n(n+1))}, \frac{19\pi}{(n(n+1))} \right]$$

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



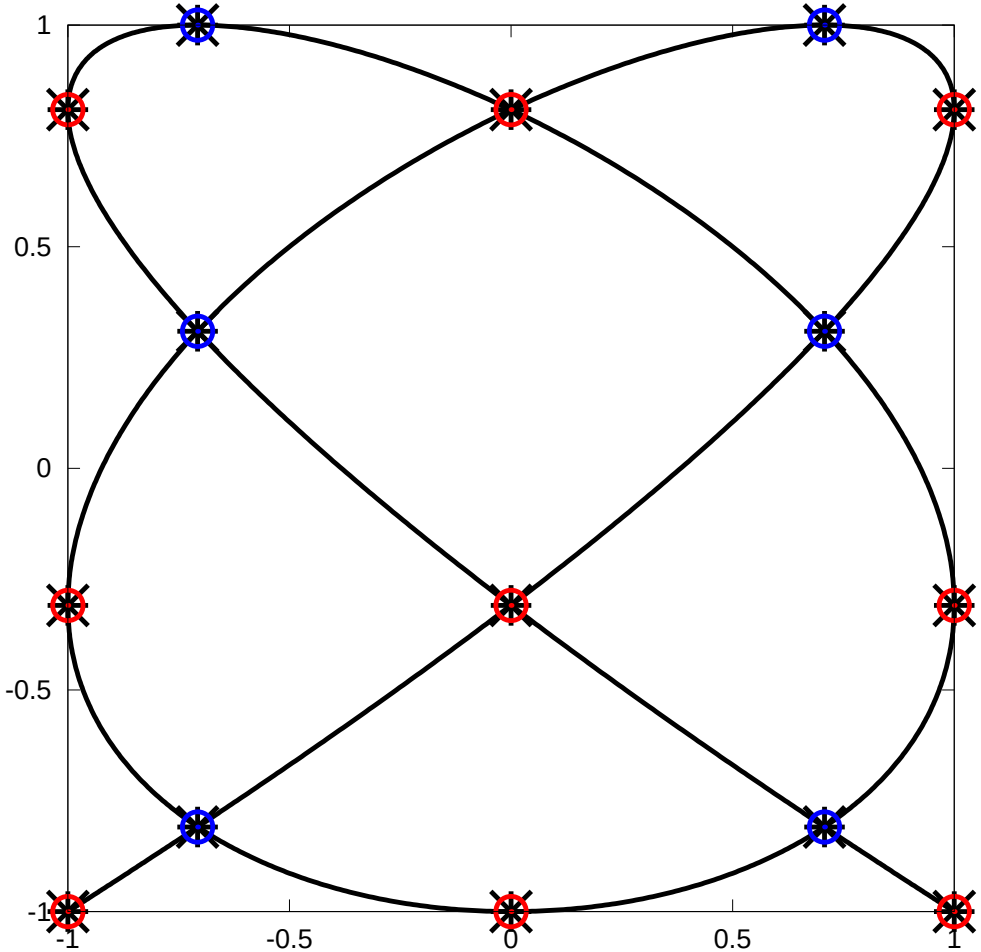
$$t \in \left[\frac{19\pi}{(n(n+1))}, \frac{20\pi}{(n(n+1))} \right]$$

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



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

The generating curve $\gamma(t)$ $(n = 4)$, is a Lissajous curve



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

Lagrange polynomials

The fundamental Lagrange polynomials of the Padua points are

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

where

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

(Note: $\{w_{\boldsymbol{\xi}}\}$ are weights of cubature formula for the prod. Cheb. measure, exact "on almost" $\Pi^n_{2n}([-1,1]^2)$), i.e. pol. orthogonal to $T_{2n}(x_1)$

$$K_n(\mathbf{x}, \mathbf{y}) = \sum_{k=0}^{n} \sum_{j=0}^{k} \hat{T}_j(x_1) \hat{T}_{k-j}(x_2) \hat{T}_j(y_1) \hat{T}_{k-j}(y_2) , \qquad (9)$$

$\hat{T}_j$ is the normalized Chebyshev polynomial of degree j.

Lagrange polynomials

The fundamental Lagrange polynomials of the Padua points are

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

where

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

(Note:$\{w_{\boldsymbol{\xi}}\}$ are weights of cubature formula for the prod. Cheb. measure, exact "on almost" $\Pi^n_{2n}([-1,1]^2)$), i.e. pol. orthogonal to $T_{2n}(x_1)$

$$K_n(\mathbf{x}, \mathbf{y}) = \sum_{k=0}^{n} \sum_{j=0}^{k} \hat{T}_j(x_1) \hat{T}_{k-j}(x_2) \hat{T}_j(y_1) \hat{T}_{k-j}(y_2) , \qquad (9)$$

$\hat{T}_j$ is the normalized Chebyshev polynomial of degree j.

Lagrange polynomials

The fundamental Lagrange polynomials of the Padua points are

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

where

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

(Note:$\{w_{\boldsymbol{\xi}}\}$ are weights of cubature formula for the prod. Cheb. measure, exact "on almost" $\Pi^n_{2n}([-1,1]^2)$), i.e. pol. orthogonal to $T_{2n}(x_1)$

$$K_n(\mathbf{x}, \mathbf{y}) = \sum_{k=0}^{n} \sum_{j=0}^{k} \hat{T}_j(x_1) \hat{T}_{k-j}(x_2) \hat{T}_j(y_1) \hat{T}_{k-j}(y_2) , \qquad (9)$$

$\hat{T}_j$ is the normalized Chebyshev polynomial of degree j.

Reproducing kernel

$K_n(\mathbf{x}, \mathbf{y})$ 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

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

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

Reproducing kernel

$K_n(\mathbf{x}, \mathbf{y})$ 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

$$\Lambda_n \leq C \log^2 n$$

(optimal order of growth on a square).

Interpolant

Given the representation (9) for the reproducing kernel, the interpolant of a function $f : [-1, 1]^2 \to \mathbb{R}$ is

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

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

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.

Interpolant

Given the representation (9) for the reproducing kernel, the interpolant of a function $f : [-1,1]^2 \to \mathbb{R}$ is

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

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

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 coefficient matrix

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

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

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

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

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

Linear algebra approach

- The construction of the coefficients is performed by a matrix-matrix product.
- It can be easily (and efficiently) implemented in FORTRAN77 (by, eventually optimized, BLAS) and in MATLAB® (based on optimized BLAS).

Linear algebra approach

- The construction of the coefficients is performed by a matrix-matrix product.
- It can be easily (and efficiently) implemented in FORTRAN77 (by, eventually optimized, BLAS) and in MATLAB® (based on optimized BLAS).

A new approach based on FFT

- Since the coefficients are approximated Fourier–Chebyshev coefficients, they can be computed also by FFT techniques.
- FFT is competitive and more stable than the matrix-matrix multiplication at high degree of interpolation.

A new approach based on FFT

- Since the coefficients are approximated Fourier–Chebyshev coefficients, they can be computed also by FFT techniques.
- FFT is competitive and more stable than the matrix-matrix multiplication at high degree of interpolation.

MATLAB® code for the FFT approach

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

```
C0 = G;
X = real(fft(C0,max(1,2*(size(G,2)-1)),2));
X = X(:,1:n+1);
Y = real(fft(X,max(1,2*(size(G,1)-1))));
C0 = Y(1:n+1,:);
C0 = 2*C0;
C0(1,:) = C0(1,:)/sqrt(2);
C0(:,1) = C0(:,1)/sqrt(2);
C0 = fliplr(triu(fliplr(C0)));
C0(n1,1) = C0(n1,1)/2;
```

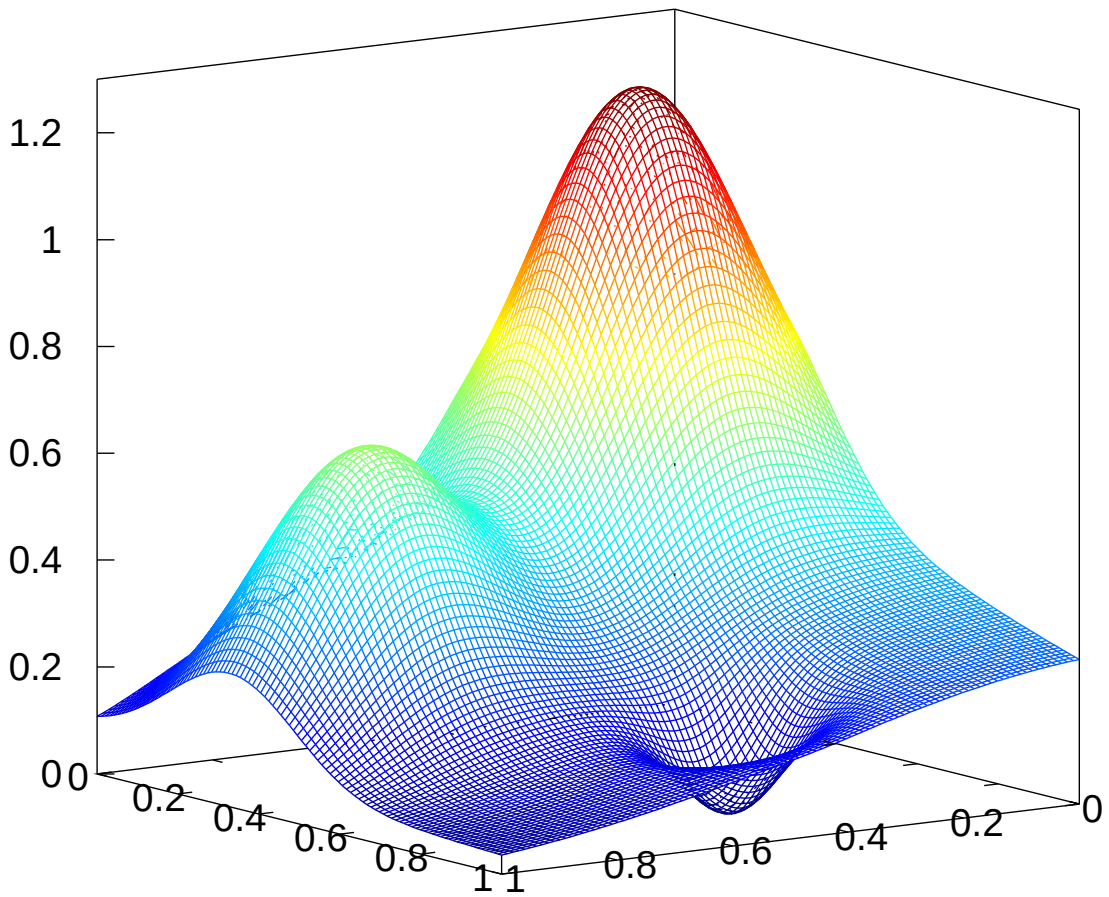
Output: $\mathtt{C0} \leftrightarrow \mathbb{C}_0$

Evaluation

Given the 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

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

Franke's function



Numerical results

Interpolation on Pad_n (total degree n) vs. tensor-product interpolation on $\mathrm{TCL}_n = C_{n+1} \times C_{n+1}$ (maximum degree n^2) for the Franke's function:

	TCL_n	Pad_n	TCL_n	Pad_n
degree n	25	34	35	48
points	625	630	1225	1225
error	$1.2 \cdot 10^{-3}$	$4.3 \cdot 10^{-5}$	$2.3 \cdot 10^{-6}$	$3.3 \cdot 10^{-8}$
	TCL_n	Pad_n	TCL_n	Pad_n
degree n	45	62	55	76
points	2025	2016	3015	3003
error	$1.5 \cdot 10^{-9}$	$5.4 \cdot 10^{-12}$	$1.9 \cdot 10^{-13}$	$1.9 \cdot 10^{-14}$

(number of points = number of function evaluations)

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. 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

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

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. 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

$$\int_{[-1,1]^2} f(\mathbf{x})\mathrm{d}\mathbf{x} \approx \int_{[-1,1]^2} \mathcal{L}_n f(\mathbf{x})\mathrm{d}\mathbf{x}$$

exact for $f \in \mathbb{P}_n^2$.

Cubature

Defining

$$c'_{j,k-j} = \begin{cases} \frac{1}{2}c_{j,k-j} = \frac{1}{2}\sum_{\boldsymbol{\xi}\in \mathrm{Pad}_n} f(\boldsymbol{\xi}) w_{\boldsymbol{\xi}} \hat{T}_j(\xi_1)\hat{T}_{k-j}(\xi_2), & j = k = n \\ c_{j,k-j} = \sum_{\boldsymbol{\xi}\in \mathrm{Pad}_n} f(\boldsymbol{\xi}) w_{\boldsymbol{\xi}} \hat{T}_j(\xi_1)\hat{T}_{k-j}(\xi_2), & \text{otherwise} \end{cases}$$

then

$$\mathcal{L}_n f(\mathbf{x}) = \sum_{k=0}^{n}\sum_{j=0}^{k} c'_{j,k-j}\hat{T}_j(x_1)\hat{T}_{k-j}(x_2)$$

Cubature

Defining

$$c'_{j,k-j} = \begin{cases} \dfrac{1}{2}c_{j,k-j} = \dfrac{1}{2}\displaystyle\sum_{\boldsymbol{\xi}\in \mathrm{Pad}_n} f(\boldsymbol{\xi}) w_{\boldsymbol{\xi}}\, \hat{T}_j(\xi_1)\, \hat{T}_{k-j}(\xi_2), & j = k = n \\ c_{j,k-j} = \displaystyle\sum_{\boldsymbol{\xi}\in \mathrm{Pad}_n} f(\boldsymbol{\xi}) w_{\boldsymbol{\xi}}\, \hat{T}_j(\xi_1)\, \hat{T}_{k-j}(\xi_2), & \text{otherwise} \end{cases}$$

then

$$\mathcal{L}_n f(\mathbf{x}) = \sum_{k=0}^{n} \sum_{j=0}^{k} c'_{j,k-j}\, \hat{T}_j(x_1)\, \hat{T}_{k-j}(x_2)$$

Moments

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

$$m_{j,k-j} = \left(\int_{-1}^{1} \hat{T}_j(t) \mathrm{d}t \right) \left(\int_{-1}^{1} \hat{T}_{k-j}(t) \mathrm{d}t \right)$$

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

Cubature weights

$$\int_{[-1,1]^2} \mathcal{L}_n f(\mathbf{x})\mathrm{d}\mathbf{x} = \sum_{k=0}^{n}\sum_{j=0}^{k} c'_{j,k-j} m_{j,k-j} = \sum_{\boldsymbol{\xi}\in \mathrm{Pad}_n} \lambda_{\boldsymbol{\xi}} f(\boldsymbol{\xi})$$

where

$$\lambda_{\boldsymbol{\xi}} = w_{\boldsymbol{\xi}} \sum_{k=0}^{n}\sum_{j=0}^{k} m'_{j,k-j}\, \hat{T}_j(\xi_1)\, \hat{T}_{k-j}(\xi_2), \qquad m'_{j,k-j} = \begin{cases} \dfrac{1}{2} m_{j,k-j}, & j = k = n \\ m_{j,k-j}, & \text{otherwise} \end{cases}$$

The cubature weights $\lambda_{\boldsymbol{\xi}}$ are not all positive. However they satisfy

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

and

$$\int_{[-1,1]^2} f(\mathbf{x})\mathrm{d}\mathbf{x} = \int_{[-1,1]^2} \mathcal{L}_n f(\mathbf{x})\mathrm{d}\mathbf{x} + o(n^{-p}), \qquad f \in \mathcal{C}^p([-1,1]^2)$$

Cubature weights: stability

$$\int_{[-1,1]^2} \mathcal{L}_n f(\mathbf{x})\mathrm{d}\mathbf{x} = \sum_{k=0}^{n}\sum_{j=0}^{k} c'_{j,k-j} m_{j,k-j} = \sum_{\boldsymbol{\xi}\in \mathrm{Pad}_n} \lambda_{\boldsymbol{\xi}} f(\boldsymbol{\xi})$$

where

$$\lambda_{\boldsymbol{\xi}} = w_{\boldsymbol{\xi}} \sum_{k=0}^{n}\sum_{j=0}^{k} m'_{j,k-j}\, \hat{T}_j(\xi_1)\, \hat{T}_{k-j}(\xi_2), \qquad m'_{j,k-j} = \begin{cases} \dfrac{1}{2} m_{j,k-j}, & j = k = n \\ m_{j,k-j}, & \text{otherwise} \end{cases}$$

The cubature weights $\lambda_{\boldsymbol{\xi}}$ are not all positive. However they satisfy

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

and

$$\int_{[-1,1]^2} f(\mathbf{x})\mathrm{d}\mathbf{x} = \int_{[-1,1]^2} \mathcal{L}_n f(\mathbf{x})\mathrm{d}\mathbf{x} + o(n^{-p}), \qquad f \in \mathcal{C}^p([-1,1]^2)$$

Cubature weights: stability and convergence

$$\int_{[-1,1]^2} \mathcal{L}_n f(\mathbf{x})\mathrm{d}\mathbf{x} = \sum_{k=0}^{n}\sum_{j=0}^{k} c'_{j,k-j} m_{j,k-j} = \sum_{\boldsymbol{\xi}\in \mathrm{Pad}_n} \lambda_{\boldsymbol{\xi}} f(\boldsymbol{\xi})$$

where

$$\lambda_{\boldsymbol{\xi}} = w_{\boldsymbol{\xi}} \sum_{k=0}^{n}\sum_{j=0}^{k} m'_{j,k-j}\, \hat{T}_j(\xi_1)\, \hat{T}_{k-j}(\xi_2), \qquad m'_{j,k-j} = \begin{cases} \dfrac{1}{2} m_{j,k-j}, & j = k = n \\ m_{j,k-j}, & \text{otherwise} \end{cases}$$

The cubature weights $\lambda_{\boldsymbol{\xi}}$ are not all positive. However they satisfy

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

and

$$\int_{[-1,1]^2} f(\mathbf{x})\mathrm{d}\mathbf{x} = \int_{[-1,1]^2} \mathcal{L}_n f(\mathbf{x})\mathrm{d}\mathbf{x} + o(n^{-p}), \qquad f \in \mathcal{C}^p([-1,1]^2)$$

MATLAB® code for the cubature

Input `C0` $\leftrightarrow \mathbb{C}_0$

```
k = [0:2:n];
mom = 2*sqrt(2)./(1-k.^2);
mom(1) = 2;
Mmom = mom'*mom;
CM = C0(1:2:n1,1:2:n1).*Mmom;
Int = sum(sum(CM));
```

Output: `Int` $\leftrightarrow \displaystyle\int_{[-1,1]^2} \mathcal{L}_n f(\mathbf{x}) \mathrm{d}\mathbf{x}$

Numerical results

Clenshaw–Curtis cubature on Pad_n (CCPad_n) vs. tensor-product Gauss–Legendre–Lobatto cubature (TGLL_n) for the Franke's function:

	TGLL_n	CCPad_n	TGLL_n	CCPad_n
degree n	6	7	8	10
points	36	36	64	66
error	$4.2 \cdot 10^{-3}$	$3.8 \cdot 10^{-4}$	$1.3 \cdot 10^{-4}$	$1.3 \cdot 10^{-5}$

	TGLL_n	CCPad_n	TGLL_n	CCPad_n
degree n	11	14	15	20
points	121	120	225	231
error	$5.7 \cdot 10^{-5}$	$9.4 \cdot 10^{-6}$	$1.1 \cdot 10^{-6}$	$1.1 \cdot 10^{-7}$

(number of points = number of function evaluations)

Numerical results

Clenshaw–Curtis cubature on Pad_n (CCPad_n) vs. tensor-product Gauss–Legendre–Lobatto cubature (TGLL_n) for the function $(x^2+y^2)^{3/2}$:

	TGLL_n	CCPad_n	TGLL_n	CCPad_n
degree n	6	7	8	10
points	36	36	64	66
error	$1.8 \cdot 10^{-3}$	$3.8 \cdot 10^{-4}$	$3.1 \cdot 10^{-4}$	$1.4 \cdot 10^{-7}$
	TGLL_n	CCPad_n	TGLL_n	CCPad_n
degree n	11	14	15	20
points	121	120	225	231
error	$7.7 \cdot 10^{-5}$	$2.8 \cdot 10^{-7}$	$1.5 \cdot 10^{-5}$	$9.8 \cdot 10^{-9}$

(number of points = number of function evaluations)

Conclusions

- We studied different families of point sets for polynomial interpolation on the square.
- The most promising, from theoretical purposes and computational cost both of the interpolant and Lebesgue constant growth are the Padua points.
- More on Padua points (papers, software, links) at the CAA research group:
  http://www.math.unipd.it/∼marcov/CAA.html
- http://en.wikipedia.org/wiki/Padua_points.

Conclusions

- We studied different families of point sets for polynomial interpolation on the square.
- The most promising, from theoretical purposes and computational cost both of the interpolant and Lebesgue constant growth are the Padua points.
- More on Padua points (papers, software, links) at the CAA research group:
  http://www.math.unipd.it/∼marcov/CAA.html
- http://en.wikipedia.org/wiki/Padua_points.

Conclusions

- We studied different families of point sets for polynomial interpolation on the square.
- The most promising, from theoretical purposes and computational cost both of the interpolant and Lebesgue constant growth are the Padua points.
- More on Padua points (papers, software, links) at the CAA research group:
  http://www.math.unipd.it/∼marcov/CAA.html
- http://en.wikipedia.org/wiki/Padua_points.

Conclusions

- We studied different families of point sets for polynomial interpolation on the square.
- The most promising, from theoretical purposes and computational cost both of the interpolant and Lebesgue constant growth are the Padua points.
- More on Padua points (papers, software, links) at the CAA research group:
  http://www.math.unipd.it/∼marcov/CAA.html
- http://en.wikipedia.org/wiki/Padua_points.

Main references

1. M. Caliari, S. De Marchi, A. Sommariva and M. Vianello: *Fast interpolation and cubature at Padua points*, In preparation (2008).
2. 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)
3. L. Bos, M. Caliari, S. De Marchi and M. Vianello: *A numerical study of the Xu polynomial interpolation formula in two variables*, Computing 76(3-4)(2006), 311–324 .
4. L. Bos, S. De Marchi and M. Vianello: *On the Lebesgue constant for the Xu interpolation formula* J. Approx. Theory 141 (2006), 134–141.
5. L. Bos, M. Caliari, S. De Marchi and M. Vianello: *Bivariate interpolation at Xu points: results, extensions and applications*, Electron. Trans. Numer. Anal. 25 (2006), 1–16.
6. 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.
7. 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.
8. 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.
9. 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.