PADOVA UNIVERSITY PRESS

Dolomites Research Notes on Approximation

Special Issue for the "10 years of the Padua points", Volume 8 · 2015 · Pages 66–74

Application of modified Leja sequences to polynomial interpolation

L. P. Bos[a] · M. Caliari[a]

Abstract

We discuss several applications of the Leja point method for univariate polynomial interpolation. First we show how more or less arbitrary interpolation points sets can be stabilized by adding some points from the Leja sequence generated beginning with the given set. We then show how the Leja point idea can be used to generate good point sets for Hermite–Lagrange polynomial interpolation. Then we discuss a version of Leja sequences for the interval $[-1,1]$ that are constrained to be symmetric sets. Finally we discuss the extension of Leja stabilization to several variables.

1 Introduction

Given a compact set $K \subset \mathbb{C}$ and a point $\xi_0 \in K$, a Leja sequence $\{\xi_j\}_{j=0}^m$ is defined recursively by taking

$$\xi_m \in \arg\max_{\xi\in K} \prod_{j=0}^{m-1} |\xi - \xi_j|, \quad m \geq 1.$$

Leja points are not optimal; there are competitor point sets for which polynomial interpolation works better. Examples of such competitors are the points for which the associated Lebesgue function has as small a maximum as possible and also the so-called Fekete points, which maximize the associated Vandermonde determinant (and for which it is known that the Lebesgue function is at least bounded by the dimension of the underlying polynomial space). However, finding Leja points is computationally much less intense as for the next higher degree one must compute only one new point, as opposed to finding an entirely new set. Moreover, as it turns out, their performance in practice is (perhaps) surprisingly good. Also, the fact that they form a *sequence* is very useful computationally, especially when used in conjunction with the Newton form of the polynomial interpolant. Indeed, if it turns out, in some application, to be desirable to increase the accuracy of the interpolant by increasing the number of interpolation points one at a time, this may easily be done using the Newton form, making use of the lower degree interpolant previously calculated.

We also remark that Leja points are known to have the same asymptotic distribution as do the "near optimal" Fekete points. See for example [1] (and the references therein) for more details about their properties and other facts about polynomial interpolation. In case K is a finite set, what results is what is called the Leja re-ordering of the points of K and is useful (indeed necessary!) for a numerically stable evaluation of polynomial interpolants in Newton form (see e.g. [15]). In this brief note we illustrate some further practical applications of the Leja point idea.

2 Leja stabilization

We consider now the following problem. Suppose we are given a set of points $\{\xi_j\}_{j=0}^k$ in the interval $K = [a,b]$ at which we want to interpolate a given function $f : [a,b] \to \mathbb{R}$. For example, the set $\{\xi_j\}_{j=0}^k$ could be a set of equidistant or even random points, for which polynomial interpolation could be highly ill-conditioned.

We therefore try to stabilize the interpolation by considering the *extended* Leja sequence $\{\xi_j\}_{j=0}^m$ given by

$$\xi_m \in \arg\max_{\xi\in[a,b]} \prod_{j=0}^{m-1} |\xi - \xi_j|, \quad m \geq k+1. \tag{1}$$

Since we do not know in advance the required degree of interpolation, we calculate it in the Newton form as follows:

- sort the set $\{\xi_j\}_{j=0}^k$ à la Leja, that is

$$\zeta_0 \in \arg\max_{\zeta\in\{\xi_j\}_{j=0}^k} |\zeta|$$

$$\zeta_i \in \arg\max_{\zeta\in\{\xi_j\}_{j=0}^k} \prod_{j=0}^{i-1} |\zeta - \zeta_j| \quad i \geq 1$$

  and rename the set $\{\xi_j\}_{j=0}^k$.

[a] Department of Computer Science, University of Verona

- Compute the interpolating polynomial of degree k in the Newton form
$$p_k(x) = \sum_{j=0}^{k} \left(f[\xi_0, \ldots, \xi_j] \prod_{i=0}^{j-1} (x - \xi_i) \right)$$
- Set $m = k+1$ and compute the next point ξ_m in the sequence according to (1) and evaluate $|f[\xi_0, \ldots, \xi_m]\omega_m(\xi_m)|$, where $\omega_m(x) = \prod_{j=0}^{m-1}(x - \xi_j)$. If it is larger than a given tolerance, compute
$$p_m(x) = p_{m-1}(x) + f[\xi_0, \ldots, \xi_m]\omega_m(x)$$
set $m = m+1$ and repeat this step. Otherwise, stop.

The stopping criterion is based on the classical formula for the interpolation error
$$e_m(x) = f(x) - p_m(x) = f[\xi_0, \ldots, \xi_{m-1}, x]\omega_m(x)$$
taking into account that $|\omega_m(\xi_m)| = \max_{x\in[a,b]}|\omega_m(x)|$.

2.1 Numerical examples

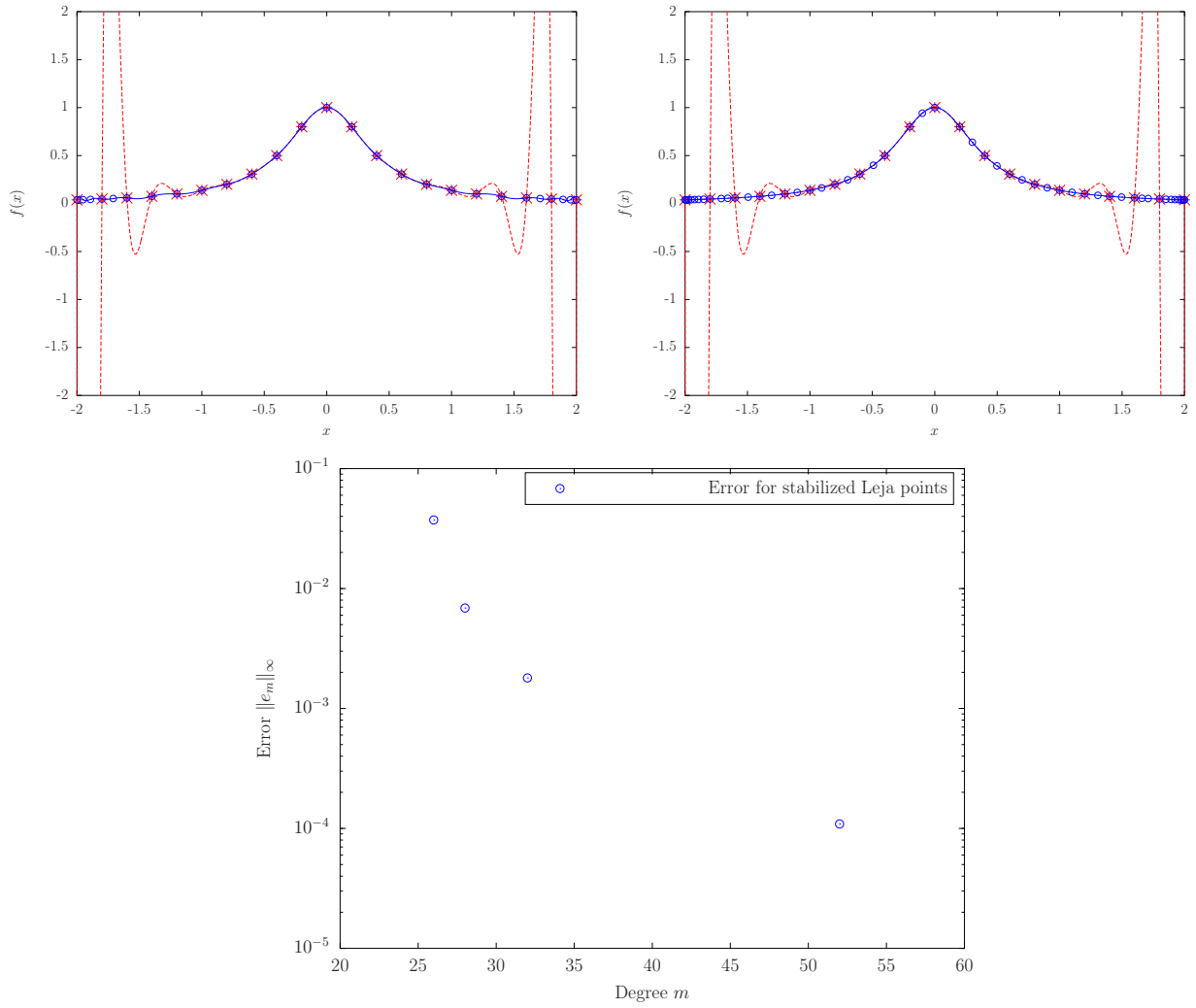

Figure 1: Leja stabilization procedure for the interpolation of the Runge function at an initial set of 21 (red stars) equidistant points (top). The interpolation at the original points is drawn with a red dashed line. The interpolation at 27 (left) and 53 (right) Leja stabilized points (blue circles) is drawn with a blue solid line. Behavior of the error with respect to the degree of interpolation (bottom).

We consider some examples of the above Leja stabilization. The first is interpolation on the interval $[-2, 2]$ of the Runge function,
$$f(x) = \frac{1}{1 + \left(\frac{5x}{2}\right)^2},$$

starting with 21 equispaced points. The prescribed tolerances were taken to be 10^{-1}, 10^{-2}, 10^{-3} and 10^{-4}, respectively. The final errors, measured on a set of 2001 equidistant points in $[-2,2]$, are $\|[\|\infty]e_m = 3.73\cdot 10^{-2}$, $\|[\|\infty]e_m = 6.87\cdot 10^{-3}$, $\|[\|\infty]e_m = 1.80\cdot 10^{-3}$ and $\|[\|\infty]e_m = 1.09\cdot 10^{-4}$, respectively with a total number of points equal to 27, 29, 33 and 53. The results corresponding to 27 and 53 points and the behavior of the error with respect to the degree of interpolation are shown in Figure 1.

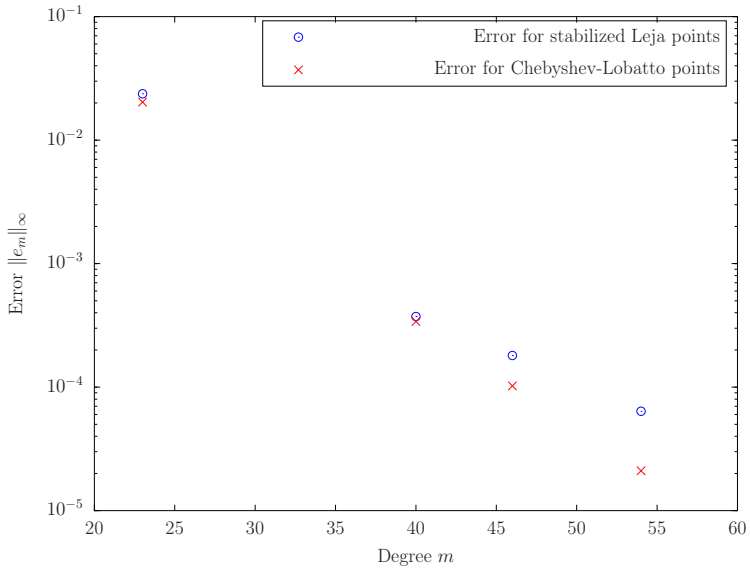

Figure 2: Behavior of the error with respect to the degree of interpolation in the Leja stabilization procedure for the interpolation of the Runge function at an initial set of Chebyshev–Lobatto points in comparison with pure Chebyshev–Lobatto points.

In this example, due to the small number of given equidistant points, the initial reordering of the points is not necessary (we get exactly the same result with or without this step), but it is anyways a recommended procedure (see [13, Example 4.1]). If we try to interpolate the same function with a set of (pure) Leja points, the stopping criterion is triggered at 15, 21, 39 and 53 points, respectively.

We also tried to add Leja points to an initial set of 21 Chebyshev–Lobatto points and prescribed the tolerances 10^{-2}, 10^{-3}, 10^{-4} and 10^{-5}, respectively. In Figure 2 we compare the the error we obtained with the interpolation error at a pure set of Chebyshev–Lobatto points of corresponding degree. Although the latter is slightly smaller, we recall that the possibility given by the Newton interpolation form of adding points if necessary is not effective for the sets of Chebyshev points, since they all change for different degrees.

As a second example, we consider an initial set of 19 uniform random points with the addition of $\{-2,2\}$ and prescribed tolerances 10^{-1}, 10^{-2}, 10^{-3} and 10^{-4}, respectively. The behavior of the error with respect to the degree of interpolation is shown in Figure 3.

As a third example we consider the Titanium Heat data (see [6, p. 197]), rescaled to $[-2,2]$. We pick the same set of 12 data points. Then we add Leja points in order to stabilize the interpolation. Since the underlying analytic function is not known, we consider a simple piecewise linear interpolant at the original 49 data points. Moreover, since it is not possible to compute the error with respect to the exact function, we consider the addition of 10, 20, 30 and 40 points, respectively. The results are shown in Figure 4. It is therefore possible to construct a smooth curve passing through the given set of data.

The final example uses a set of discrete data similar to that of [14, Figures 1.–3.], namely

```
data =
 0.04 0.21 0.26 0.34 0.53 0.61 0.70 0.84 1.00
 0.19 0.28 0.58 0.66 0.62 0.18 0.13 0.09 0.05
```

The abscissas are then rescaled to $[-2,2]$. The missing values are reconstructed by means of piecewise linear interpolation. Again, the result is a smooth curve passing through the given set of data.

3 Leja–Hermite points

It is also possible to use Leja points for Hermite interpolation, that is interpolation which also matches the derivatives of the function at some interpolation points. In order to do this, and use the Newton interpolation form, it is sufficient that the points at which derivatives have to be reconstructed appear consecutively and with the right multiplicity. For instance, if the k-th derivative is required at 0, it is possible to define the following Leja sequence: $\{\xi_j\}_{j=0}^{k} = \{0\}$ and

$$\xi_m \in \arg\max_{\xi\in[a,b]} \prod_{j=0}^{m-1} |\xi-\xi_j| = \arg\max_{\xi\in[a,b]} |\xi|^{k+1} \prod_{j=k+1}^{m-1} |\xi-\xi_j|, \quad m \ge k+1.$$

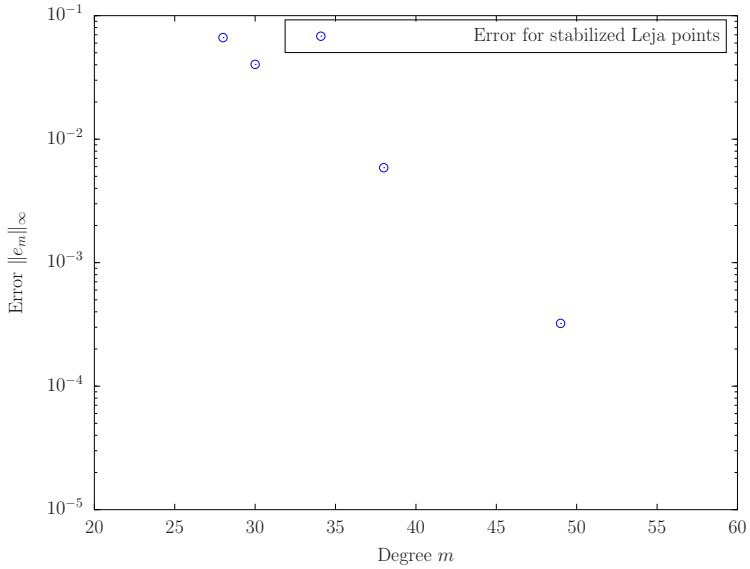

Figure 3: Behavior of the error with respect to the degree of interpolation in the Leja stabilization procedure for the interpolation of the Runge function at an initial set of uniform random points.

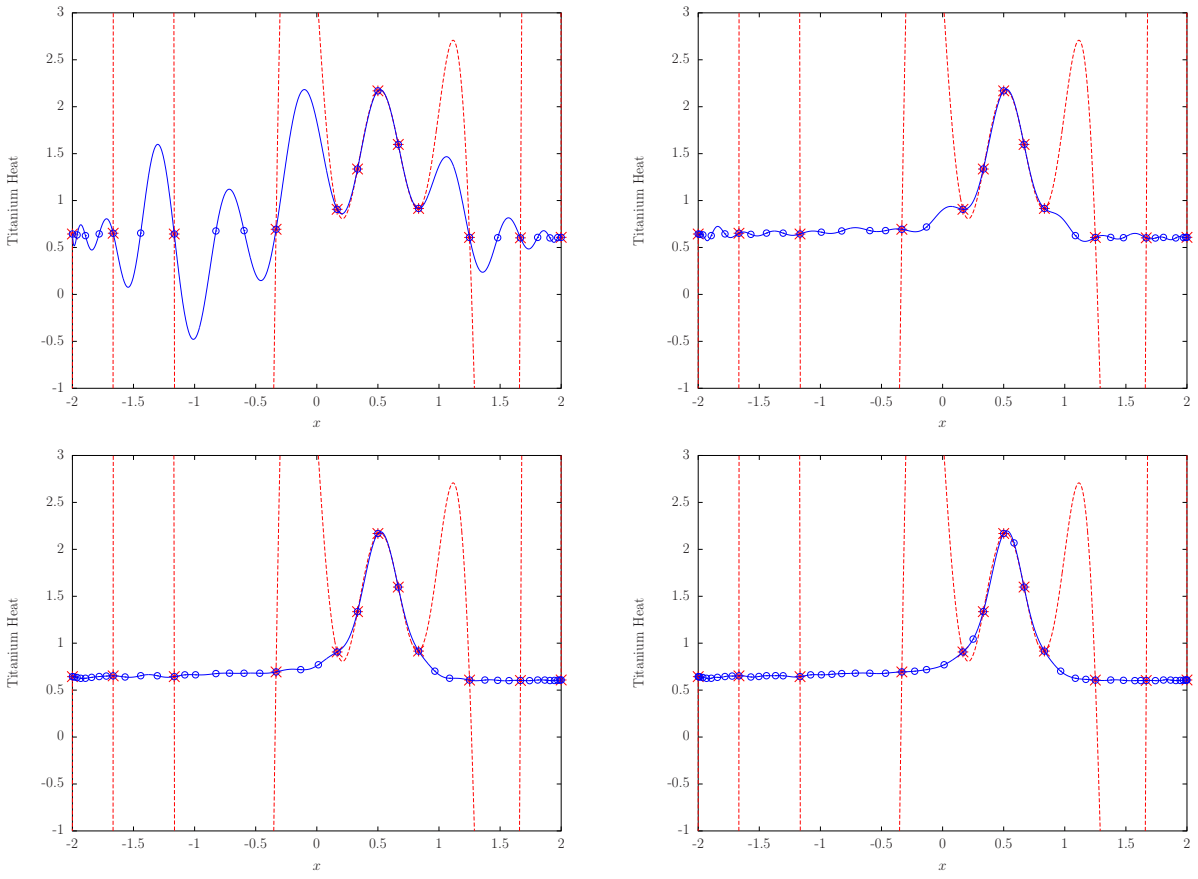

Figure 4: Leja stabilization procedure for the interpolation of certain Titanium Heat data (red stars). The interpolation at the original points is drawn with a red dashed line. The interpolation at 22, 32, 42, 52 Leja stabilized points (blue circles), respectively, is drawn with a blue solid line.

When $[a,b]$ is a symmetric interval $[-a,a]$, it is possible to analytically compute the first three points after the initial set of zeros, which are $\xi_{k+1} \in \{-a,a\}$, $\xi_{k+2} \in \{-a,a\} \setminus \{\xi_{k+1}\}$ and $\xi_{k+3} \in \{\pm\sqrt{a^2(k+1)/(k+3)}\}$. For the Newton interpolation form, it is

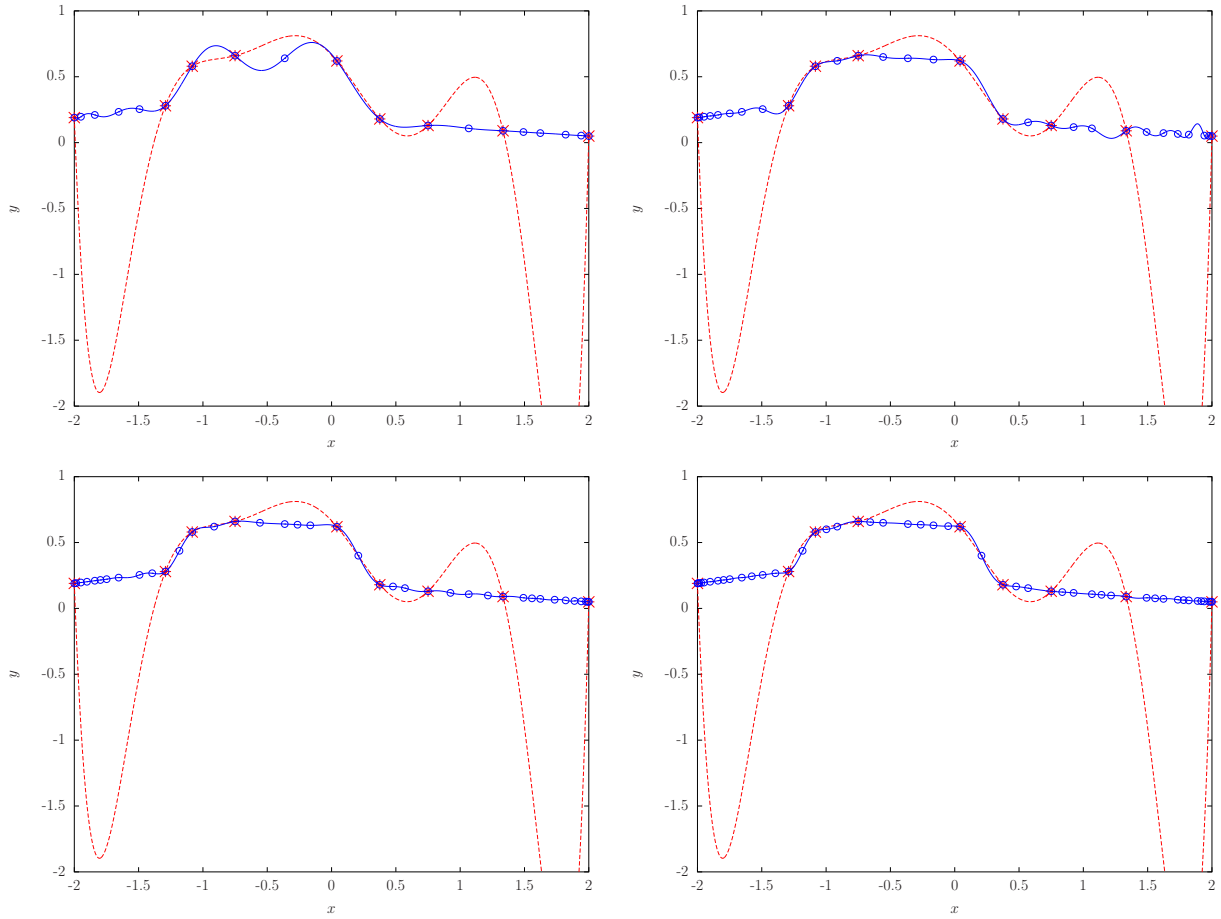

Figure 5: Leja stabilization procedure for the interpolation of the discrete data described on page 68 (red stars). The interpolation at the orginal points is drawn with a red dashed line. The interpolation with 19, 29, 39, 49 Leja stabilized points (blue circles), respectively, is drawn with a blue solid line.

possible to use the notion of confluent divided differences (see [10]): we define

$$\begin{aligned} f[\xi_i,\dots,\xi_j] &= \frac{f^{(j-i)}(\xi_0)}{(j-i)!} && 0 \le i \le j \le k \\ f[\xi_m] &= f(\xi_m) && m \ge k+1 \\ f[\xi_i,\dots,\xi_k,\xi_{k+1},\dots,\xi_m] &= \frac{f[\xi_{i+1},\dots,\xi_m]-f[\xi_i,\dots,\xi_{m-1}]}{\xi_m-\xi_i} && 0 \le i \le k < m \end{aligned}$$

Then, it is possible to define the polynomial p_m of degree m

$$p_m(x) = \sum_{j=0}^{m}\left(f[\xi_0,\dots,\xi_j]\prod_{i=0}^{j-1}(x-\xi_j)\right)$$

which satisfies

$$\begin{aligned} p_m^{(j)}(\xi_0) &= f^{(j)}(\xi_0) \quad j = 0,\dots,k \\ p_m(\xi_j) &= f(\xi_j) \qquad j = k+1,\dots,m \end{aligned}$$

As an example, we consider the solution of a simple differential equation

$$\begin{cases} -u_\varepsilon''(x) = -\dfrac{1}{2\varepsilon}\chi_{(-\varepsilon,\varepsilon)} & x \in (-2,2) \\ u_\varepsilon(-2) = u_\varepsilon(2) = 0 \end{cases}$$

that is

$$u_\varepsilon = \begin{cases} -\dfrac{1}{2}(x+2) & -2 \le x \le -\varepsilon \\ \dfrac{1}{4\varepsilon}x^2 + \dfrac{\varepsilon-4}{4} & -\varepsilon < x < \varepsilon \\ -\dfrac{1}{2}(2-x) & \varepsilon \le x \le 2 \end{cases}$$

In Figure 6 we show the interpolation of degree 6 of the solution u_ε corresponding to $\varepsilon = 0.5$ with the point 0 not repeated, repeated once and twice, respectively. The errors, in infinity norm over a set of 2001 equidistant points, are 0.04, 0.03 and 0.02, respectively.

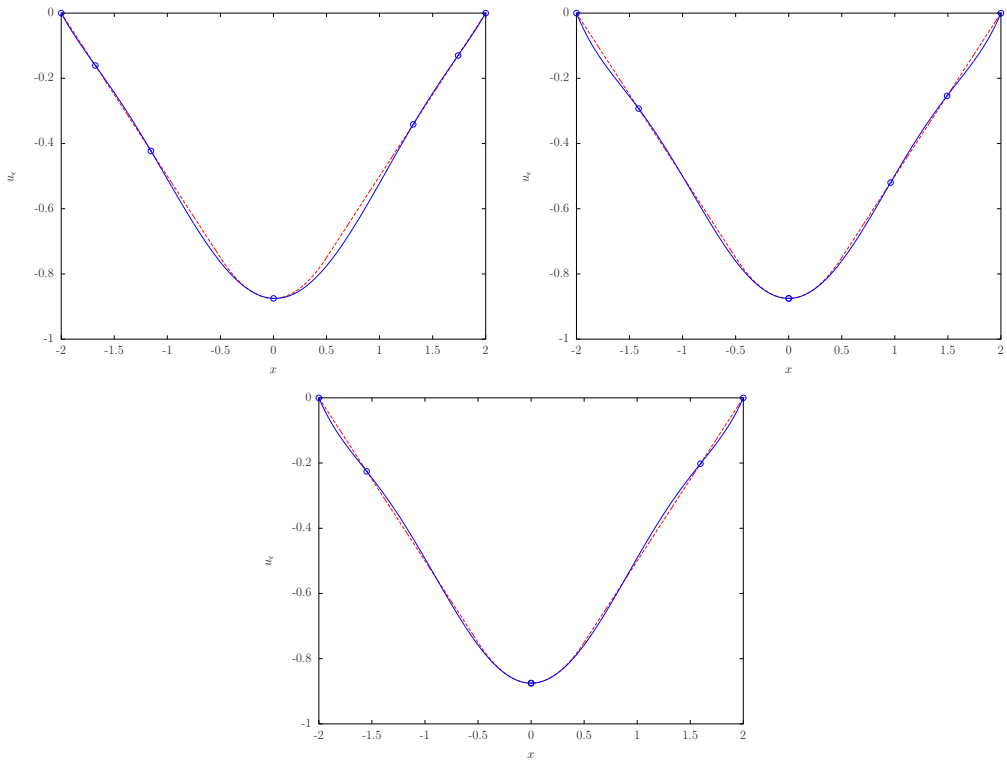

Figure 6: Interpolation (blue solid line) at Leja–Hermite points (blue circles) with 0 not repeated (upper left), repeated once (upper right) and repeated twice (lower). The exact solution is drawn with a red dashed line.

4 Symmetric Leja sequences

In a symmetric interval $[-a, a]$, it would be desirable to have symmetric interpolation points, especially for the interpolation of even or odd functions. Or else we can consider the interpolation on purely complex intervals $i[-b, b]$: symmetric points allow one to use real arithmetic for the interpolation of functions satisfying $\overline{f(z)} = f(\overline{z})$ (see [15]). We consider here three sequences of points related to Leja sequences. The first is the real projection of a Leja sequence for the complex unit disk (see [5]). Then we consider the Leja points $\{\zeta_j\}_j$ for the interval $[0, a^2]$ (with $\zeta_0 = 0$ as starting point), and construct the sequence $\{\xi_j\}_j$ defined by

$$\xi_j = \begin{cases} 0 & j = 0 \\ \sqrt{\zeta_j} & j \text{ odd} \\ -\sqrt{\zeta_{j-1}} & j \text{ even}, j \neq 0 \end{cases}$$

The final set is computed as in [4], namely $\xi_0 = 0$ and, recursively,

$$\xi_m \in \arg\max_{\xi \in [0,a]} \prod_{j=0}^{m-1} |\xi - \xi_j|, \quad \xi_{m+1} = -\xi_m, \quad m \text{ odd}$$

Therefore, points with even index are forced to be symmetric with respect to the previous point. For the three sequences, we show the growth of the Lebesgue constant (see Figure 7), computed over a set of about 5000 points interleaved with the Leja points.

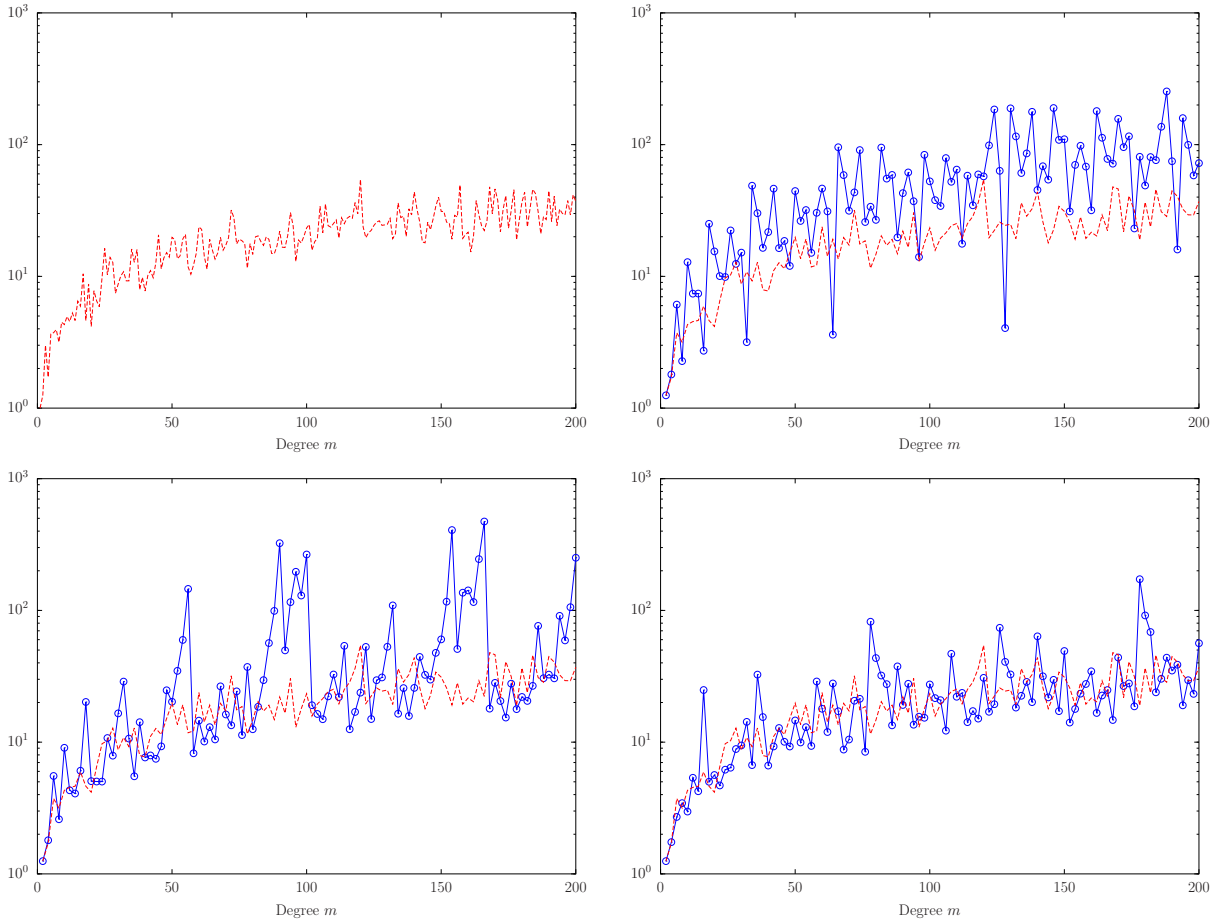

Figure 7: Lebesgue constant for the different sets of symmetric Leja points, namely real projection of Leja points (top-right), positive and negative square roots of Leja points on $[0,1]$ (bottom-left) and symmetric forced Leja points (bottom-right), respectively. The Lebesgue constant for the original Leja points is drwan with a red dashed line.

5 The computation of Leja points and extension to several variables

In order to numerically compute (1), we consider the function

$$L(\xi)=\prod_{j=0}^{m-1}(\xi-\xi_j)$$

and its derivative

$$L'(\xi)=\sum_{i=0}^{m-1}\prod_{\substack{j=0\\ j\neq i}}^{m-1}(\xi-\xi_j)$$

We therefore look for the $m-2$ (simple) zeros $\{\zeta_j\}_{j=0}^{m-2}$ of the polynomials $L'(\xi)_{|(\xi_j,\xi_{j+1})}$. Then, we select

$$\xi_m\in\arg\max_{\zeta_j}|L(\zeta_j)|$$

This approach, which we call *continuous*, works with repeated nodes, too.

In [2] the authors showed how to extract Leja sequences from admissible meshes using the LU factorization with partial (row) pivoting. We briefly sketch the algorithm here, in Matlab notation. If $V(a_1,\dots,a_m;p_1,\dots,p_n)$ is the Vandermonde matrix

$$V(\boldsymbol{a};\boldsymbol{p})=(p_j(a_i))_{i,j},\quad 1\le i\le m,\ 1\le j\le n$$

with $\{a_i\}_{i=1}^m$ a set of candidate points and $\{p_j\}_{j=1}^n$ a polynomial basis with p_j of degree $j-1$, then the set of Leja points $\{\xi_j\}$ can be extracted by

```
[L,U,p] = lu(V,'vector');
xi = a(p(1:n));
```

If an initial set of points is given and we want 1) to order them à la Leja and 2) to add Leja points, we have to construct an initial vector $\boldsymbol{a}$ with $a_i = \xi_{i-1}$, $i = 1,\dots,k+1$ and $a_i \in [-2,2] \setminus \{\xi\}_{j=0}^{k}$, $i = k+2,\dots,m$. Then, the first $k+1$ steps of the LU factorization have to be performed with a pivoting strategy restricted to the first $k+1$ rows of V. Finally, the standard pivoting strategy is applied till the end. It is possible to write a reasonable fast Matlab code by taking the so called jki (left-looking, see [8, Ch. 7.2]) implementation of the LU factorization, where `keep` denotes the number of initial points.

```
[m,n] = size (V);
p = (1:m);
for j = 1:n
  for k = 1:j-1
    V(k+1:m,j) = V(k+1:m,j) - V(k+1:m,k) * V(k,j);
  end
  endpoint = (j <= keep)*keep+(j > keep)*m;
  [~,idx] = max(abs(V(j:endpoint,j)));
  jp = j-1+idx;
% exchange rows
  temp = V(j,:); V(j,:) = V(jp,:); V(jp,:) = temp;
  temp = p(j); p(j) = p(jp); p(jp) = temp;
  i = j+1:m;
  V(i, j) = V(i, j) / V(j, j);
end
```

For instance, starting from an initial set of 21 equispaced points in $[-2,2]$, it was possible to reorder them and add 22 Leja points extracted from a set of 100001 points in less then three seconds on a modern laptop. The maximum error with respect to the set generated with the continuous approach is about $2 \cdot 10^{-5}$.

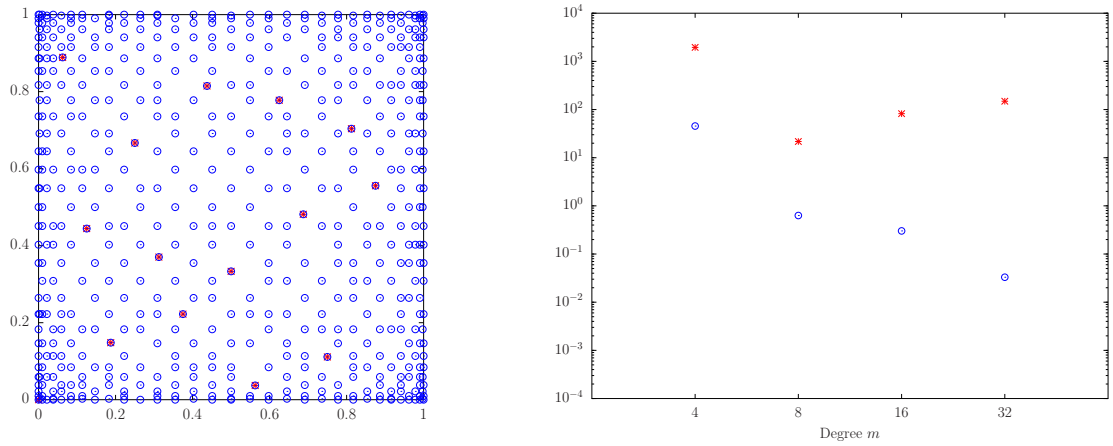

Figure 8: Leja points with the original 15 points (red stars) for degree 32 (left) and the behavior of the Lebesgue constant (red stars) and interpolation error (blue circles) (right).

Of course, this modified LU factorization can be applied to the Vandermonde matrix for a set of points in $\mathbb{C}^d$ and polynomials on $\mathbb{C}^d$. We show here an example in $\mathbb{R}^2$. Starting with the set of first 15 points from a Halton sequence, namely

```
data =
 0 1/2 1/4 3/4 1/8 5/8 3/8 7/8 1/16 9/16 5/16  13/16 3/16 11/16 7/16
 0 1/3 2/3 1/9 4/9 7/9 2/9 5/9 8/9  1/27 10/27 19/27 4/27 13/27 22/27
```

we extract Leja points from meshes for the degrees 8, 16, and 32 in the square $[0,1]^2$ provided by the software described in [7]. Then we interpolate the two-dimensional Runge function

$$f(x,y) = \frac{1}{1+(5(2x-1))^2+(5(2y-1))^2}$$

measure the interpolation error and estimate the Lebesgue constant. In Figure 8 we show the final set of Leja points for degree 32 (left) and the behavior of the interpolation error and the Lebesgue constant (right). Of course, degree 4 corresponds to interpolation at the original set of 15 points. In Figure 9 we show the interpolated functions at the original 15 points (left) and at 561 Leja points (corresponding to degree 32).

It is also possible to extract Leja–Hermite points using the LU factorization. We describe the procedure for the computation of Leja points with ξ_0 repeated $k+1$ times. The initial vector $\boldsymbol{a}$ is $a_i = \xi_0$, $i = 1,\dots,k+1$, $a_i \in [-2,2] \setminus \{\xi_0\}$, $i = k+2,\dots,m$.

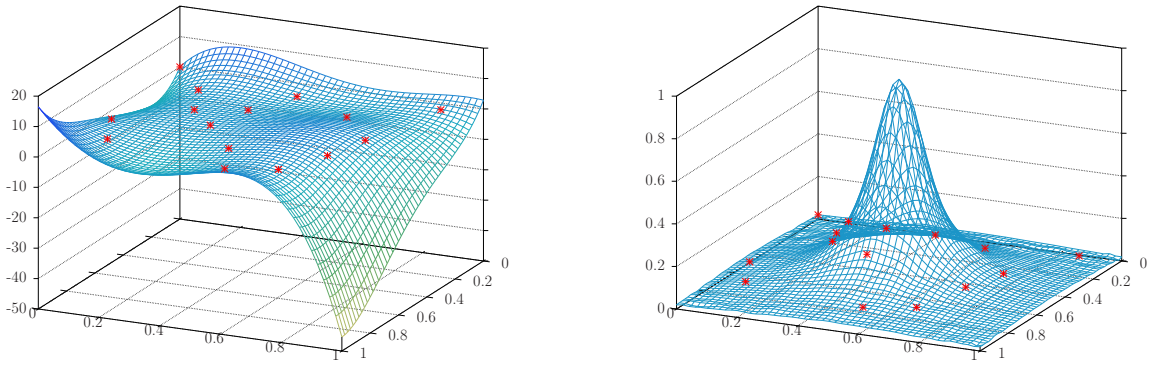

Figure 9: Interpolation of the two-dimensional Runge function at the original 15 points (left) and at 561 (degree 32) Leja points (right).

Then we need to apply the previous algorithm, with $\texttt{keep} = k+1$, to the *confluent* Vandermonde matrix

$$\begin{bmatrix} V(\xi_0; \boldsymbol{p}) \\ V(\xi_0; \boldsymbol{p}^{(1)}) \\ \vdots \\ V(\xi_0; \boldsymbol{p}^{(k)}) \\ V(a_{k+2}, \ldots, a_m; \boldsymbol{p}) \end{bmatrix}$$

where $\boldsymbol{p}^{(j)}$ denotes the ordered set of derivatives of basis polynomials. Note that this matrix is just the interpolation matrix corresponding to the associated Hermite-Lagrange interpolation matrix and its determinant can be easily computed (see [9, (6.1.34)]). In particular the minors (and hence the pivots) have factors $(a_j - \xi_0)^{k+1}$.

The computation of divided difference, especially for high degree of interpolation, may become unstable even for points sorted à la Leja. We refer the reader to [3, 12, 11] for possible alternatives.

References

[1] T. Bloom, L. Bos, J.-P. Calvi, and N. Levenberg. Polynomial interpolation and approximation in $\mathbb{C}^d$. *Ann. Polon. Math.*, 106:53–81, 2012.

[2] L. Bos, S. De Marchi, A. Sommariva, and M. Vianello. Computing multivariate Fekete and Leja points by numerical linear algebra. *SIAM J. Numer. Anal.*, 48(5):1984–1999, 2010.

[3] M. Caliari. Accurate evaluation of divided differences for polynomial interpolation of exponential propagators. *Computing*, 80(2):189–201, 2007.

[4] M. Caliari, A. Ostermann, and S. Rainer. Meshfree exponential integrators. *SIAM J. Sci. Comput.*, 35(1):A431–A452, 2013.

[5] J.-P. Calvi and P. V. Manh. Lagrange interpolation at real projections of Leja sequences for the unit disk. *Proc. Amer. Math. Soc.*, 140(12):4271–4284, 2012.

[6] C. de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer, revised edition, 2001.

[7] S. De Marchi, F. Piazzon, A. Sommariva, and M. Vianello. Polynomial Meshes: Computation and Approximation. In *Proceedings of the 15th International Conference on Computational and Mathematical Methods in Science and Engineering*, pages 414–425, 2015.

[8] G. Golub and J. M. Ortega. *Scientific Computing An Introduction with Parallel Computing*. Academic Press, Inc., 1993.

[9] R. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[10] B. Kalantari. Generalization of Taylor's theorem and Newton's method via a new family of determinantal interpolation formulas and its applications. *J. Comput. Appl. Math.*, 126:287–318, 2000.

[11] M. Lopez-Fernandez and S. Sauter. Fast and stable contour integration for high order divided differences via elliptic functions. *Math. Comp.*, 84:1291–1315, 2015.

[12] A. McCurdy, K. C. Ng, and B. N. Parlett. Accurate computation of divided differences of the exponential function. *Math. Comp.*, 43(168):501–528, 1984.

[13] L. Reichel. Newton interpolation at Leja points. *BIT*, 30:332–346, 1990.

[14] K. Salkauskas. C^1 splines for interpolation of rapidly varying data. *Rocky Mountain J. Math.*, 14(1), 1974.

[15] H. Tal-Ezer. High degree polynomial interpolation in Newton form. *SIAM J. Sci. Stat. Comput.*, 12(3):648–667, 1991.