Electronic Transactions on Numerical Analysis.
Volume 7, 1998, pp. 124-140.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

FAST LEJA POINTS*

J. BAGLAMA†, D. CALVETTI‡, AND L. REICHEL§

Abstract. Leja points are used in several areas of scientific computing, including polynomial approximation and eigenvalue computation. Their determination requires the maximization of a sequence of polynomials over a compact set in the complex plane. These computations can be quite time consuming when the number of Leja points to be determined is large. This paper introduces a new set of points, referred to as fast Leja points, that are simpler and faster to compute. An interactive example that illustrates the computation and distribution of fast Leja points is available at web site http://etna.mcs.kent.edu/vol.7.1998/pp124-140.html.

Key words. Leja points, polynomial interpolation, iterative methods, eigenvalue computation.

AMS subject classifications. 65D05, 65E05, 65F15, 65N25.

1. Introduction. Let $\mathbb{K}$ be a compact set in the complex plane $\mathbb{C}$ with a connected complement, and assume that the boundary of $\mathbb{K}$ satisfies some mild regularity conditions to be specified below. Edrei [7] and Leja [12] studied the following recursively defined sequence $\{z_j\}_{j=0}^{\infty}$ of points in $\mathbb{K}$. Let z_0 be a point such that

$$w(z_0)|z_0| = \max_{z\in\mathbb{K}} w(z)|z|, \qquad z_0 \in \mathbb{K}, \tag{1.1}$$

and let for $j = 1, 2, \ldots$, the points z_j satisfy

$$w(z_j)\prod_{k=0}^{j-1} |z_j - z_k| = \max_{z\in\mathbb{K}}\ w(z)\prod_{k=0}^{j-1} |z - z_k|, \qquad z_j \in \mathbb{K}, \tag{1.2}$$

where $w(z)$ is a given real-valued positive function on $\mathbb{K}$. We refer to $w(z)$ as the weight function. The points z_j determined by (1.1)-(1.2) might not be unique. We will call any sequence of points $\{z_j\}_{j=0}^{\infty}$ that satisfies (1.1)-(1.2) a sequence of Leja points for $\mathbb{K}$. Edrei [7] and Leja [12] studied these points for $w(z) := 1$ on $\mathbb{K}$.

Leja points are used in several areas of scientific computing, e.g., for polynomial interpolation [14], in iterative methods for the solution of large linear systems of equations [4, 5] and in eigenvalue computations [1, 2, 6].

The computation of Leja points requires the maximization of a sequence of products over a compact set $\mathbb{K}$, see (1.2), and these computations can be quite cumbersome when the number of Leja points generated is large. It is the purpose of the present paper to introduce a new set of points, referred to as *fast Leja points*, which are simpler and faster to compute than Leja points.

The interest in Leja points stems from the fact that when $z_0, z_1, z_2, \ldots,$ are Leja points for the set $\mathbb{K}$, the polynomials

$$p_j(z) := \prod_{k=0}^{j-1}(z - z_k), \qquad j = 1, 2, \ldots, \tag{1.3}$$

* Received March 2, 1998. Accepted June 30, 1998. Recommended by R. Lehoucq.
† Department of Mathematics, Texas Tech University, Lubbock, TX 79409. (baglama@math.ttu.edu). Research supported in part by NSF grant F377 DMR-8920147 ALCOM.
‡ Department of Mathematics, Case Western Reserve University, Cleveland, OH 44106. (dxc57@po.cwru.edu). Research supported in part by NSF grant DMS-9896073.
§ Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242. (reichel@mcs.kent.edu). Research supported in part by NSF grants DMS-9404706 and ASC-9720221.

ETNA
Kent State University
etna@mcs.kent.edu

can be evaluated fairly accurately for $z \in \mathbb{K}$ also when their degree j is large; see [14] for illustrations. Products of the form (1.3) arise, e.g., in the Newton formula for polynomial interpolation [3, 14], in the Implicitly Restarted Lanczos and Arnoldi methods for the computation of a few eigenvalues of a large matrix [1, 2, 6, 9, 10, 17], and in Richardson iteration for the solution of linear systems of equations [4, 5]. In the applications to eigenvalue computation and the solution of linear systems, each iteration increases the degree of polynomials of the form (1.3) by one, and, in general, it is not known a priori how many iterations are required until the desired eigenvalues, or the approximate solution, have been determined to sufficient accuracy. Leja points are convenient to use in these applications, because the Leja points z_j are independent of the Leja points z_k for $j < k$. Thus, the polynomial $p_k(z)$ is divisible by $p_j(z)$ for $1 \leq j \leq k$.

It is illuminating to compare polynomial interpolation at Leja points for an interval $[a, b]$ and at zeros of Chebyshev polynomials for the same interval. Assume that the Newton form of the interpolation polynomial is used. Polynomial interpolation of a smooth function f at the zeros of the Chebyshev polynomial of the first kind of degree k is known to yield a near-best polynomial approximant of degree at most $k-1$ of f on $[a, b]$; see, e.g., de Boor [3, Chapter 2] for a discussion and illustrative numerical examples. However, if we find that the achieved accuracy is insufficient and therefore we would like to interpolate f at the zeros of the Chebyshev polynomial of degree $k+1$ instead, it would be difficult to take advantage of the computations already carried out when determining the new interpolating polynomial. The difficulty stems from the fact that the Chebyshev polynomials of degree k and $k+1$ do not have common zeros. On the other hand, given the interpolating polynomial in Newton form of degree at most $k-1$ that interpolates f at the first k Leja points for $[a, b]$, it is simple and fast to update this polynomial to obtain the polynomial of degree at most k that interpolates f at the first $k+1$ Leja points for $\mathbb{K}$.

When the weight function is given by $w(z) := 1$, the maximum principle for analytic functions yields that the maxima in (1.1) and (1.2) are achieved for z on the boundary of $\mathbb{K}$. This also holds for other weight functions of interest to us. It therefore suffices to only maximize over the boundary of $\mathbb{K}$ in (1.1) and (1.2). We seek to reduce the computational work further by replacing the boundary of $\mathbb{K}$ by a discrete point set. Throughout this paper $i := \sqrt{-1}$.

Example 1.1. Let $\mathbb{K} := \{z : |z| \leq r\}$ for some constant $r > 0$, and let $w(z) := 1$. Then a sequence of Leja points can be constructed as follows. Let z_0 be an arbitrary point on the boundary of $\mathbb{K}$. The next Leja point z_1 is a point in $\mathbb{K}$ of largest distance to z_0. Thus, $z_1 := z_0 \exp(i\pi)$. Note that the point z_1 is uniquely determined. The Leja point z_2 maximizes the product of the distances $|z - z_1||z - z_0|$ for all $z \in \mathbb{K}$. This yields $z_2 = z_0 \exp(i\pi/2)$ or $z_2 = z_0 \exp(i3\pi/2)$. Note that either one of the two possible values of z_2 is acceptable. Having determined z_2, we have $z_3 := -z_2$. There are four possible choices for z_4.

Consider the binary representation of the nonnegative integers

$$j = \sum_{k=0}^{\infty} j_k 2^k, \qquad j_k \in \{0, 1\},$$

and define the points

$$z_j := z_0 \exp(i\pi \sum_{k=0}^{\infty} j_k 2^{-k}), \qquad j = 1, 2, \ldots . \tag{1.4}$$

It can be shown that the points (1.4) are Leja points for $\mathbb{K}$. Table 1.1 shows the arguments for z_j/z_0 for a few values of j.

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 1.1
Example 1.1: Arguments of z_j/z_0 for points z_j defined by (1.4)

j	$\arg(z_j/z_0)$
0	0
1	π
2	$\pi/2$
3	$3\pi/2$
4	$\pi/4$
5	$5\pi/4$
6	$3\pi/4$
7	$7\pi/4$

It follows from symmetry considerations that the maximum of each product in (1.2) is achieved at the midpoint of a boundary curve between two adjacent points (1.4). Moreover, the points in every set of 2^k Leja points $\{z_j\}_{j=0}^{2^k-1}$, $k = 1, 2, \ldots$, are equidistant on the boundary of $\mathbb{K}$. □

Example 1.2. Let $\mathbb{K}$ and $w(z)$ be the same as in Example 1.1. In order to generate the Leja points $\{z_j\}_{j=0}^{2^k-1}$, the set $\mathbb{K}$ can be replaced by the point set $\mathbb{K}_{2^\ell} := \{z_0 \exp(2\pi i k/2^\ell), k = 0, 1, \ldots, 2^\ell - 1\}$ for any integer $\ell \geq k$. □

In recent applications to eigenvalue computations and the solution of linear systems of equations discussed in [1, 2, 4], the set $\mathbb{K}$ is an interval $[a, b]$.

Example 1.3. Let $\mathbb{K} := [a, b]$, $0 \leq a < b < \infty$, $w(z) := 1$ and $z_0 := b$. Then $z_1 := a$ and $z_2 := \frac{1}{2}(a + b)$. Formulas for Leja points z_j for $j > 2$ are more complicated. However, we will see in Section 2 that Leja points for $\mathbb{K}$ have the same limit distributed as zeros of Chebyshev polynomials for $\mathbb{K}$. □

Example 1.4. Let $\mathbb{K}$ and $w(z)$ be the same as in Example 1.3, and let $\mathbb{K}'_m(a, b)$ consist of m points in $\mathbb{K}$. Introduce

$$\epsilon_m := \max_{z\in\mathbb{K}} \min_{z'\in\mathbb{K}'_m(a,b)} |z - z'|,$$

and assume that m increases rapidly enough in relation to the number of Leja points, n, so that

$$\epsilon_m = o(n^{-2}), \qquad n \to \infty. \tag{1.5}$$

This condition is satisfied for instance, when

$$\mathbb{K}'_m(a, b) := \left\{ a + (b - a)\frac{j - 1}{m - 1},\ j = 1, 2, \ldots, m \right\} \tag{1.6}$$

and $n = o(m^{1/2})$. Saff and Totik [16, Section V.1] show that if (1.5) holds, then the Leja points $\{z_j\}_{j=0}^{n-1}$ for $\mathbb{K}'_m(a, b)$ have the same limit distribution as $n \to \infty$ (and $m \to \infty$) as Leja points for $\mathbb{K}$. □

In order to reduce the computational work required for the generation of Leja points for an interval $[a, b]$, we proposed in [1, 2, 4] to instead compute Leja points for discrete sets

$$\mathbb{K}_m(a, b) := \left\{ \frac{a + b}{2} + \frac{a - b}{2} \cos\left(\frac{2j - 1}{2m}\pi\right),\ j = 1, 2, \ldots, m \right\} \tag{1.7}$$

consisting of zeros of the mth degree Chebyshev polynomial of the first kind for $\mathbb{K}$.

ETNA
Kent State University
etna@mcs.kent.edu

PROPOSITION 1.1. *Let* $\mathbb{K} := [a, b]$, $0 \leq a < b < \infty$ *and* $w(z) := 1$. *Let* $\mathbb{K}_m(a, b)$ *be defined by (1.7). Let* $n = n(m)$ *be a nondecreasing function of* m, *such that for all* $m > 0, 1 \leq n(m) \leq cm$ *for some constant* c *in the open interval* $(0, 1)$. *Then the Leja points* $\{z_j\}_{j=0}^{n-1}$ *for* $\mathbb{K}_m(a, b)$ *have the same limit distribution as* $n \to \infty$ *as Leja points for* $\mathbb{K}$.

Proof. The proposition follows from bounds by Ehlich and Zeller [8] for polynomials on an interval bounded at Chebyshev points for this interval. These bounds show that the inequality (1.14) in [16, p. 266] holds and the proposition can be shown by techniques discussed by Saff and Totik [16, Section V.1]. □

In some eigenvalue computations reported in [1], as well as in the iterative method for linear systems of equations with a symmetric indefinite matrix described in [5], the set $\mathbb{K}$ consists of two intervals $[a, b] \cup [a', b']$ with $a < b < 0 < a' < b'$. We conjecture that an analogue of Proposition 1.1 holds for this case. Here each interval is replaced by a discrete subset consisting of zeros of Chebyshev polynomials of degree m for the interval.

We note that use of the point sets (1.6) and (1.7) instead of the set $\mathbb{K} = [a, b]$ in (1.1)-(1.2) simplifies the computations, because maximization problems over a continuum are replaced by maximization problems over a finite point set. For fixed sets (1.6) or (1.7) with points $\{w_k\}_{k=1}^m$, the first n Leja points for $\mathbb{K}'_m(a, b)$ or $\mathbb{K}_m(a, b)$ can be computed in roughly $2mn$ arithmetic floating point operations by storing the products

$$\prod_{j=0}^{j-1} |w_k - z_j|, \qquad 1 \leq k \leq m, \tag{1.8}$$

for every $j = 1, 2, \ldots, n-1$. More details on operation counts are given in Section 3. Here, we only note that this operation count may increase significantly each time m is increased, because an increase in m yields a new point set $\{w_k\}_{k=1}^m$ and therefore new products (1.8) have to be evaluated.

This paper presents a progressive adaptive discretization of the boundary of $\mathbb{K}$ and replaces the maximization over $\mathbb{K}$ in (1.1)-(1.2) by maximization over discrete sets. Fast Leja points are obtained by maximization over these discrete sets. The discretization is carried out so that parts of the boundary with many fast Leja points are discretized by a fine mesh, while those parts of the boundary with few fast Leja points are discretized by a coarse mesh. The discretization is updated as new fast Leja points are computed. Advantages of this approach include:

(i) Computer codes for the generation of fast Leja points are simple and the computation of fast Leja points is rapid. The computation of n fast Leja points for $\mathbb{K}$ requires only roughly $\frac{1}{2}n^2$ arithmetic floating point operations.

(ii) The set $\mathbb{K}$ may be modified during the computation of fast Leja points without increasing the operation count. This is important in applications to eigenvalue computations and the solution of linear systems of equations; see below.

In eigenvalue computations, the set $\mathbb{K}$ should contain all or most of the undesired eigenvalues of the matrix; see Section 4. However, often such a set is not explicitly known a priori. In the algorithms described in [1, 2] sets $\mathbb{K}$ with this property are determined and updated during the computation. Analogously, when solving systems of linear equations, the set $\mathbb{K}$ should contain all or almost all eigenvalues of the matrix. The algorithms presented in [4, 5] determine such sets during the computation, and the sets $\mathbb{K}$ used during the iterations are generally updated several times as new spectral information about the matrix becomes available.

This paper is organized as follows. Section 2 reviews properties of Leja points. Fast Leja points are introduced in Section 3, and computed examples that illustrate properties of fast Leja points are presented in Section 4. Section 4 also displays timings that show that

ETNA
Kent State University
etna@mcs.kent.edu

fast Leja points indeed can be generated much faster than Leja points. An appendix contains MATLAB code for the generation of fast Leja points.

2. Some properties of Leja points. We review some properties of Leja points. These properties can be conveniently described in terms of a Green function for the complement of $\mathbb{K}$. In particular, Example 2.2 shows that the limit distribution of Leja points on an interval is the same as the limit distribution for zeros of Chebyshev polynomials for the interval.

Identify $\mathbb{C}$ with the real plane $\mathbb{R}^2$. Throughout this paper $x, y \in \mathbb{R}$ and $z \in \mathbb{C}$. If $z = x + iy$, $i := \sqrt{-1}$, then z and (x, y) denote the same point. Let $\mathbb{K} \subset \mathbb{C}$ be a compact set, whose complement $\Omega := (\mathbb{C} \cup \{\infty\})\backslash\mathbb{K}$ is connected and possesses a Green function $G(x, y)$ with a logarithmic singularity at infinity. In particular, we assume that $\mathbb{K}$ is such that the boundary of $\mathbb{K}$, denoted by $\partial\mathbb{K}$ also is the boundary of Ω, denoted by $\partial\Omega$. This Green function is uniquely determined by the requirements i) $\Delta G(x, y) = 0$ in $\Omega\backslash\{\infty\}$, ii) $G(x, y) = 0$ on $\partial\Omega$, and iii)

$$\frac{1}{2\pi}\int_{\partial\Omega}\frac{\partial}{\partial n}G(x,y)d\sigma = 1,$$

where $\frac{\partial}{\partial n}$ denotes the normal derivative directed into Ω and $d\sigma$ stands for the element of arc length; see, e.g., [19, Chapter 4.1]. The non-negative constant κ defined by

$$\kappa := \lim_{|z|\to\infty} |z| \exp(-G(x,y)), \qquad z = x + iy,$$

is called the *capacity* of $\mathbb{K}$. The capacity depends on the size of $\mathbb{K}$. If $\mathbb{K}$ has capacity κ and α is a positive constant, then the set $\alpha\mathbb{K} := \{\alpha z : z \in \mathbb{K}\}$ has capacity $\alpha\kappa$.

Leja [12] showed that Leja points for $\mathbb{K}$ are uniformly distributed with respect to the density function

$$\frac{1}{2\pi}\frac{\partial}{\partial n}G(x,y), \qquad z = x + iy \in \partial\Omega \tag{2.1}$$

on the boundary of $\mathbb{K}$. Thus, each subarc γ of the boundary of $\mathbb{K}$ contains the fraction $\frac{1}{2\pi}\int_\gamma \frac{\partial}{\partial n}G(x,y)d\sigma$ of the Leja points $\{z_j\}_{j=0}^n$ for $\mathbb{K}$ as $n \to \infty$.

Example 2.1. Let $\mathbb{K}$ be the set in Example 1.1. Then the Green function is given by

$$G(x,y) := \ln|\frac{z}{r}|, \qquad z = x + iy,$$

and therefore $\mathbb{K}$ has capacity r. Since $\frac{\partial}{\partial n}G(x,y) = 1/r$ on $\partial\mathbb{K}$, Leja points are uniformly distributed on $\partial\mathbb{K}$. □

Example 2.2. Let $\mathbb{K} = [a, b]$ be a real interval, c.f. Example 1.3. Then the Green function is given by

$$G(x,y) := \ln\left|\frac{2}{b-a}(z - \frac{b+a}{2} + (z^2 - z(b+a) + ba)^{1/2})\right|, \qquad z := x + iy,$$

where the branch of the square root is chosen so that $|\frac{2}{b-a}(z-\frac{b+a}{2}+(z^2-z(b+a)+ba)^{1/2})| > 1$ for $z \in \Omega$. The capacity of $\mathbb{K}$ is $\frac{1}{4}(b-a)$. Leja points for $\mathbb{K}$ are uniformly distributed with respect to the density function

$$\frac{1}{2\pi}\frac{\partial}{\partial n}G(x,0) = \frac{2}{\pi}(b-a)^{-1}(b-x)^{-1/2}(x-a)^{-1/2}, \qquad a < x < b.$$

ETNA
Kent State University
etna@mcs.kent.edu

It is well known that zeros of Chebyshev polynomials for $[a, b]$ are uniformly distributed with respect to this density function, too. Thus, Leja points and zeros of Chebyshev polynomials have the same limit distribution. □

The following theorem shows some properties of Leja points that are easy to verify numerically. Numerical examples in Section 4 indicate that fast Leja points share these properties.

THEOREM 2.1. *Let $\{z_j\}_{j=0}^{\infty}$ be a sequence of Leja points for the set $\mathbb{K}$, i.e, the z_j satisfy (1.1)-(1.2). Assume that the weight function satisfies $b_1 \leq w(z) \leq b_2$ for all $z \in \mathbb{K}$ for some constants $0 < b_1 \leq b_2 < \infty$. Let κ denote the capacity for $\mathbb{K}$. Then*

$$\prod_{j=0}^{n-1} |z_n - z_j|^{1/n} \geq \kappa \left(\frac{b_1}{b_2}\right)^{1/n} \tag{2.2}$$

and

$$\lim_{n\to\infty} \prod_{j=0}^{n-1} |z_n - z_j|^{1/n} = \kappa. \tag{2.3}$$

Proof. The proof by Leja [12] for the case when $w(z) := 1$ extends in a straightforward manner to positive weight functions $w(z)$. □

It follows from results of Leja [12] that when the points $\{z_k\}_{k=0}^{j-1}$ in (1.3) are Leja points for a set $\mathbb{K}$ with capacity κ, the jth root of the magnitude of the products (1.3) converges to κ as $j \to \infty$ for any z in the interior of $\mathbb{K}$. This property is helpful for the accurate evaluation of products (1.3) of large degree for $z \in \mathbb{K}$.

3. Fast Leja points. Examples 1.1 and 1.2 are suggestive for the definition of fast Leja points. Assume for the moment that the boundary of $\mathbb{K}$ is a Jordan curve or a Jordan arc, and let $z(t)$, $0 \leq t \leq L$, be a continuous parametric representation of the boundary, such that $z(t) \neq z(s)$ for $0 \leq t < s < L$. Fast Leja points can now be defined recursively. Let $\{z(t_j)\}_{j=0}^{k-1}$ be a set of fast Leja points on the boundary of $\mathbb{K}$. If $\mathbb{K}$ is a Jordan arc, then we require that $t_0 = 0$ and $t_1 = L$. Let $\{z(s_j)\}_{j=0}^{l-1}$ be a set of candidate points that interlace the set of fast Leja points; between each pair of adjacent fast Leja points there is a candidate point. Thus, $l = k - 1$ when $\mathbb{K}$ is a Jordan arc, and $l = k$ when $\mathbb{K}$ is a Jordan curve. The next fast Leja point $z(t_k)$ is chosen among the candidate points, i.e., $t_k \in \{s_j\}_{j=0}^{l-1}$ satisfies

$$w(z(t_k)) \prod_{l=0}^{k-1} |z(t_k) - z(t_l)| = \max_{0\leq j<\ell} w(z(s_j)) \prod_{l=0}^{k-1} |z(s_j) - z(t_l)|. \tag{3.1}$$

Having determined $z(t_k)$, we remove this point from the set of candidate points and new candidate points (in general two) are introduced between $z(t_k)$ and the closest fast Leja points, so that the new set of fast Leja points $\{z(t_j)\}_{j=0}^{k}$ and the new set of candidate points interlace. The next fast Leja point, $z(t_{k+1})$, is now determined analogously.

Example 3.1. Let $\mathbb{K} := \{z : |z| \leq 1\}$ and consider the parametric representation $z(t) := \exp(it)$, $i = \sqrt{-1}$, $0 \leq t \leq 2\pi$, of the boundary curve. Let $w(z) := 1$ and introduce

$$t_j := j\pi, \quad s_j := t_j + \frac{\pi}{2}, \qquad j = 0, 1.$$

Then the fast Leja points $z(t_0)$ and $z(t_1)$ interlace the candidate points $z(s_0)$ and $z(s_1)$. Both candidate points yield the same value of the product on the right-hand side of (3.1). Therefore,

ETNA
Kent State University
etna@mcs.kent.edu

the next fast Leja point, $z(t_2)$, can be chosen to be either $z(s_0)$ or $z(s_1)$, say, $t_2 = s_0$. We choose the two new candidate points, $z(s_2)$ and $z(s_3)$, to be on the unit circle, equidistantly between $z(t_2)$ and adjacent fast Leja points. Continuing to allocate fast Leja points and candidate points in this manner, with new candidate points allocated at the midpoints between adjacent fast Leja points, yields a sequence of fast Leja points $\{z(t_j)\}_{j=0}^{\infty}$. This sequence is equivalent with the sequence of Leja points $\{z_j\}_{j=0}^{\infty}$ of Example 1.1 in the sense that

$$\max_{z\in\mathbb{K}} \prod_{j=0}^{n} |z - z(t_j)| = \max_{z\in\mathbb{K}} \prod_{j=0}^{n} |z - z_j|, \quad n = 0, 1, \ldots .$$

□

Example 3.2. Let $\mathbb{K} := [a, b]$, $0 \le a < b < \infty$, and consider the parametric representation $z(t) := t$, $a \le t \le b$. Let $w(z) := 1$. Allocate candidate points equidistantly between adjacent fast Leja points. Define

$$t_0 := b, \qquad t_1 := a, \qquad s_0 := \frac{a+b}{2}.$$

This corresponds to the fast Leja points $z(t_0) := a$ and $z(t_1) := b$, and candidate point $z(s_0) := \frac{a+b}{2}$. The next fast Leja point clearly is $z(t_2)$ with $t_2 := s_0$. We reuse s_0 to store a parameter value for a new candidate point. Thus, the new candidate points $z(s_0)$ and $z(s_1)$ correspond to the parameter values $s_0 = \frac{3a+b}{2}$ and $s_1 = \frac{a+3b}{2}$. We can choose t_3 to be either s_0 or s_1. This defines the fast Leja point $z(t_3)$. MATLAB code for generating fast Leja points for an interval in this manner is displayed in the Appendix. □

Example 3.3. We illustrate the use of a parametric representation that is not arc length. Let $\mathbb{K} := [-2, 2]$ and consider the parametric representation $z(t) := 2\cos(t)$, $0 \le t \le \pi$. We remark that zeros of Chebyshev polynomials of the first kind for $\mathbb{K}$ correspond to equidistant parameter values. Let $w(z) := 1$. Allocate candidate points equidistantly with respect to the parameter t between adjacent fast Leja points. Thus, define

$$t_0 := 0, \qquad t_1 := \pi, \qquad s_0 := \frac{\pi}{2}.$$

This corresponds to the fast Leja points $z(t_0) := 2$ and $z(t_1) := -2$, and candidate point $z(s_0) := 0$. The next fast Leja point clearly is $z(t_2)$ with $t_2 := s_0$, and the new candidate points $z(s_0)$ and $z(s_1)$ correspond to the parameter values $s_0 = \frac{\pi}{4}$ and $s_1 = \frac{3\pi}{4}$. Then we can choose t_3 to be either s_0 or s_1. This defines the fast Leja point $z(t_3)$. This point is different from the fast Leja point $z(t_3)$ in Example 3.2.

We will see in Example 4.2 below, that the fast Leja points of the present example are inferior to the fast Leja points of Example 3.2. This may depend on that $|z'(t)|$ is not bounded away from zero for $0 \le t \le \pi$. □.

This method of generating fast Leja points generalizes in a straightforward manner to sets $\mathbb{K}$ with a boundary that consists of several components: allocate fast Leja points and candidate points on every boundary curve and boundary arc using a parametric representation for each curve or arc.

We turn to the operation count for generating fast Leja points on a real interval as described in Example 3.2. Assume that the fast Leja points $\{z(t_j)\}_{j=0}^{k}$, the interlacing candidate points $\{z(s_j)\}_{j=0}^{k-1}$, and the products

$$\pi_{j,k-1} := w(z(s_j)) \prod_{q=0}^{k-1} |z(s_j) - z(t_q)|, \qquad 0 \le j \le k-3, \tag{3.2}$$

ETNA
Kent State University
etna@mcs.kent.edu

are known. We evaluate the products

$$\pi_{j,k} := \pi_{j,k-1}|z(s_j) - z(t_k)|, \qquad 0 \le j \le k-3,$$

as well as

$$\pi_{j,k} := w(z(s_j)) \prod_{q=0}^{k} |z(s_j) - z(t_q)|, \qquad j = k-2, k-1.$$

The next fast Leja point is given by $z(t_{k+1}) := z(s_q)$, where the index q is determined by the condition

$$\pi_{q,k} = \max_{0 \le j < k} \pi_{j,k}. \tag{3.3}$$

The point $z(s_q)$ is removed from the set of candidate points, and we let $s_j := s_{j-1}$ for $j = q, q+1, \ldots, k-1$. This yields the candidate points $\{z(s_j)\}_{j=0}^{k-1}$. To this set we add two new candidate points, $z(s_k)$ and $z(s_{k+1})$, adjacent to $z(t_{k+1})$. We are now in a position to determine the next point $z(t_{k+2})$ in the sequence of fast Leja points. The computations can be simplified by working with linked lists of fast Leja points and candidate points. In the MATLAB code in the Appendix such linked lists are simulated by using index arrays.

The count of arithmetic floating point operations $(+, -, *, /)$ for the computation of the fast Leja point $z(t_{k+1})$ as outlined above is $4k + \mathcal{O}(1)$. Thus, the set of fast Leja points $\{z(t_j)\}_{j=0}^{k}$ can be computed in $2k^2 + \mathcal{O}(k)$ arithmetic operations when the products (3.2) are stored. This operation count ignores the evaluation of the absolute values.

4. Numerical examples. Computed examples of this section indicate that fast Leja points can be valuable for polynomial approximation by interpolation and for eigenvalue computation. All numerical experiments were carried out on an HP 9000/735 work station using double precision arithmetic, i.e., with approximately 16 significant digits.

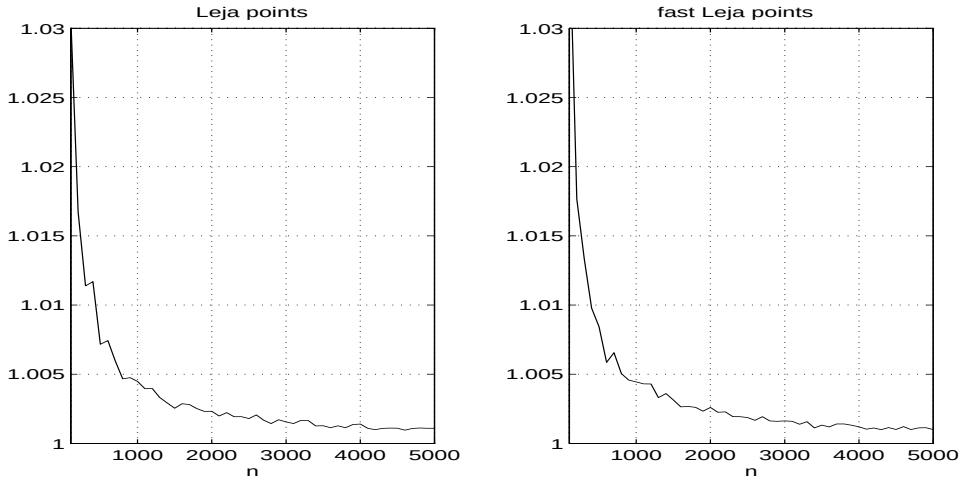

FIG. 4.1. *Example 4.1: Graphs for* $\prod_{j=0}^{n-1} |z_n - z_j|^{1/n}$ *as* n *increases.*

ETNA
Kent State University
etna@mcs.kent.edu

4.1. Polynomial interpolation. We present examples that illustrate the behavior of the products in (2.2) and (2.3) in Theorem 2.1 and display approximation errors obtained by polynomial interpolation at Chebyshev points, Leja points and fast Leja points. Polynomial interpolation is an important application of fast Leja points. This application, moreover, reveals how similar the distribution of fast Leja points is to the distribution of Chebyshev points already for a fairly small number of points.

Example 4.1. Let $\mathbb{K} := [-2, 2]$ and $w(z) := 1$. According to Example 2.2, $\mathbb{K}$ has capacity $\kappa = 1$. Let $\{z_j\}_{j=0}^{\infty}$ be a sequence of Leja points for $\mathbb{K}$ and define the mapping

$$h(n) := \prod_{j=0}^{n-1} |z_n - z_j|^{1/n}, \qquad n = 1, 2, \ldots .$$

By Theorem 2.1, $h(n) \geq 1$ and $\lim_{n\to\infty} h(n) = 1$. The graph on the left-hand side of Figure 4.1 displays $h(n)$. We remark that in computations for generating the graph, we replaced the set $\mathbb{K}$ by the discrete set $\mathbb{K}_m(-2, 2)$ defined by (1.7) with $m = 6000$, and generated Leja points for this set. An increase of the number of points m in the set $\mathbb{K}_m(-2, 2)$ did not result in a significantly different graph.

The graph on the right-hand side of Figure 4.1 displays the mapping $h(n)$ when the points $z_j := z(t_j)$ are fast Leja points generated as described in Example 3.2. Comparing the graphs of Figure 4.1 indicates that the mapping $h(n)$ behaves in a similar way for Leja points and fast Leja points. □

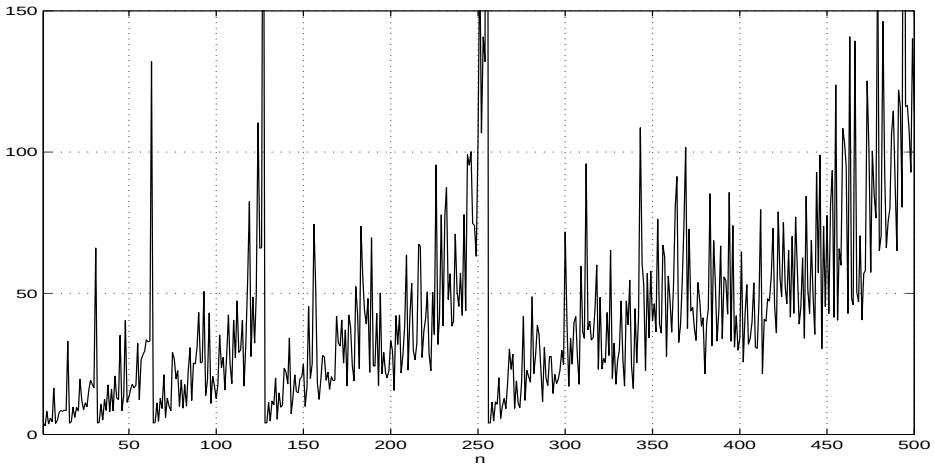

FIG. 4.2. *Example 4.2: Graph of* $\max_{z\in\mathbb{K}} \prod_{j=0}^{n-1} |z - z_j|$ *for* $1 \leq n \leq 500$, z_j *fast Leja points generated as described in Example 3.3.*

Example 4.2. We examine the products $\prod_{j=0}^{n-1} |z - z_j|$, $n = 1, 2, \ldots$, for $z \in \mathbb{K} := [-2, 2]$ and $w(z) := 1$. Figures 4.2-4.4 display the maximum value of these products on $\mathbb{K}$ when the points z_j are determined in three different ways. In Figure 4.2 the points z_j are generated as described in Example 3.3. During the computation of the z_j the interval $[0, \pi]$ for the parameter t is replaced by a set of 5000 equidistant points. We obtain $\max_{1\leq n\leq 500}\{\max_{z\in\mathbb{K}} \prod_{j=0}^{n-1} |z - z_j|\} = 528$.

The points z_j for Figure 4.3 are Leja points for $\mathbb{K}_m(-2, 2)$ with $m = 5000$. The value of m is chosen large enough so that the behavior of the products does not change when m is increased. We therefore refer to the computed points z_j as Leja points for $\mathbb{K}$. We note that

ETNA
Kent State University
etna@mcs.kent.edu

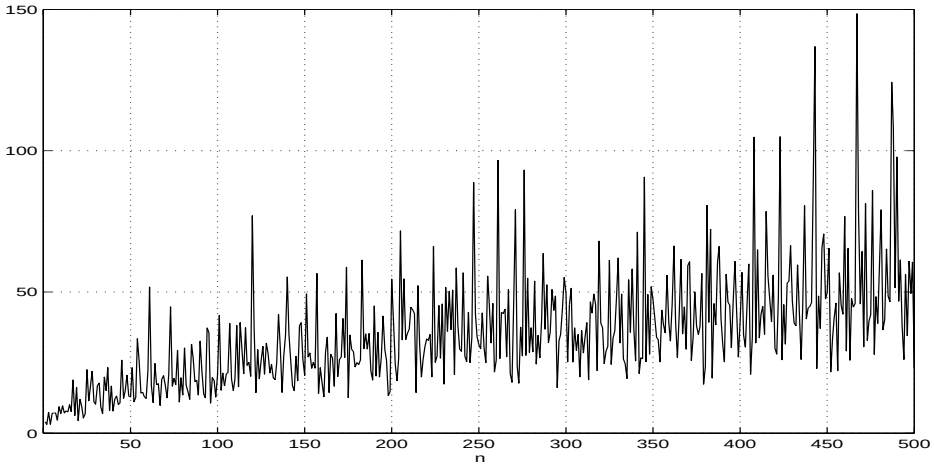

FIG. 4.3. *Example 4.2: Graph of* $\max_{z\in\mathbb{K}} \prod_{j=0}^{n-1} |z - z_j|$ *for* $1 \le n \le 500$, z_j *Leja points for* $\mathbb{K}_{5000}(-2, 2)$.

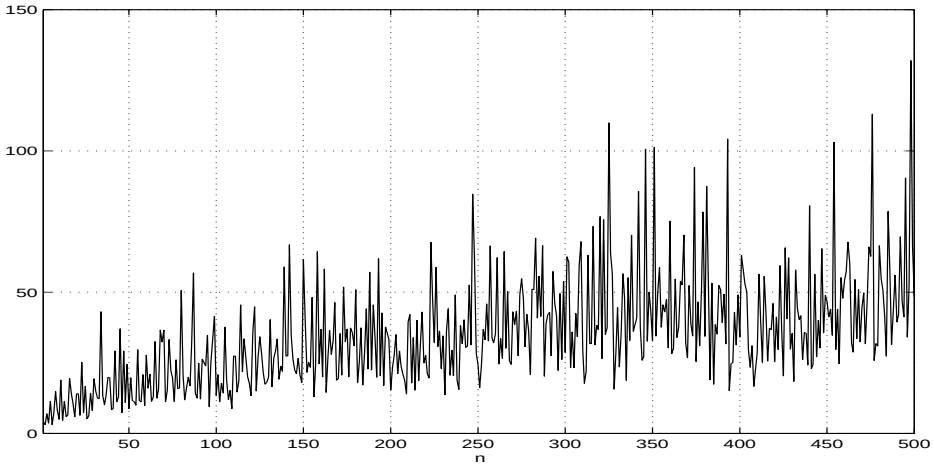

FIG. 4.4. *Example 4.2: Graph of* $\max_{z\in\mathbb{K}} \prod_{j=0}^{n-1} |z - z_j|$ *for* $1 \le n \le 500$, z_j *fast Leja points generated as described in Example 3.2.*

the expression $\max_{0\le k<n}\{\max_{z\in\mathbb{K}} \prod_{j=0}^{k-1} |z - z_j|\}$ grows less rapidly with n for the Leja points for $\mathbb{K}$ than for the points z_j used for Figure 4.2.

The points for Figure 4.4 are fast Leja points for $\mathbb{K}$ generated as described in Example 3.2. The expression $\max_{0\le k<n}\{\max_{z\in\mathbb{K}} \prod_{j=0}^{k-1} |z - z_j|\}$ grows slower with n than for any of the other sets of points z_j in this example. Slow growth is advantageous when approximating functions f on $[-2, 2]$ by polynomials that interpolate f at the points z_j; see Walsh [19, Chapter 4], de Boor [3, Chapter 2] or [14] for discussions. □

The following two examples compute polynomials in Newton form that interpolate given functions at Chebyshev, Leja and fast Leja points. The stability of the Newton form of the interpolation polynomial when interpolating at Leja points has previously been demonstrated in [14]. The examples below suggest that the stability properties and rate of convergence are the same when interpolating at fast Leja points.

ETNA
Kent State University
etna@mcs.kent.edu

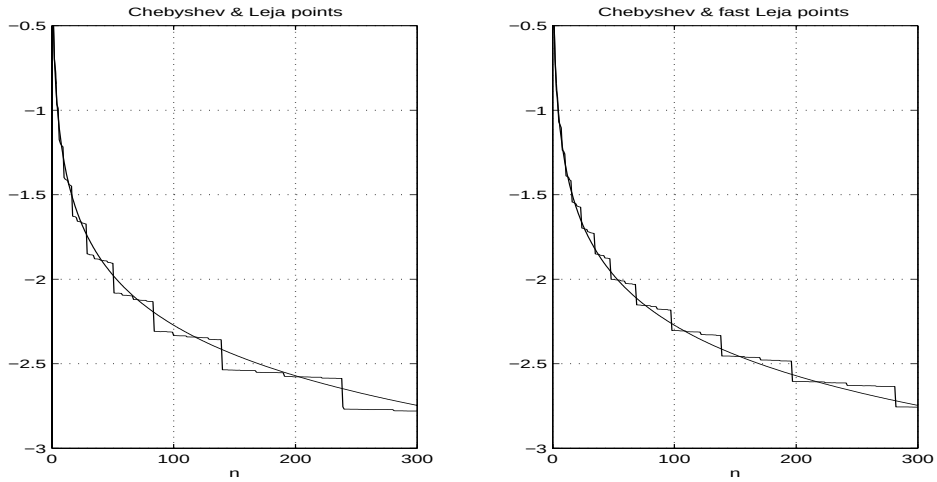

FIG. 4.5. *Example 4.3:* $\log_{10}(\max_{z\in\mathbb{K}}|f(z)-p_n(z)|)$ *for* $1 \le n \le 300$, *where the polynomial* p_n *of the form (4.1) interpolates* $f(z) := (1+\frac{z}{2})^{1/2}$ *at Chebyshev points (4.3) (smooth curves in both graphs), at Leja points for* $\mathbb{K}_{3000}(-2,2)$ *(staircase curve on the left), and at fast Leja points (staircase curve on the right).*

Example 4.3. Let $\mathbb{K} := [-2,2]$ and $w(z) := 1$. We consider polynomial approximation of the function $f(z) := (1+\frac{z}{2})^{1/2}$ on $\mathbb{K}$ by interpolating polynomials p_n in Newton form

$$p_n(z) := f[z_0] + \sum_{j=1}^{n} f[z_0, z_1, z_2, ..., z_j] \prod_{k=0}^{j-1} (z - z_k), \tag{4.1}$$

where the divided differences are defined recursively by

$$f[z_j, z_{j+1}, ..., z_k] := \frac{f[z_{j+1}, z_{j+2}, ..., z_k] - f[z_j, z_{j+1}, ..., z_{k-1}]}{z_k - z_j} \tag{4.2}$$

and $f[z_j] := f(z_j)$. It is convenient to use the Newton form of the interpolation polynomial, because it easily can be updated when interpolation at additional points is required, e.g., when new fast Leja points have been generated.

This example compares the approximation error for sequences of interpolation polynomials generated in three different ways. We first determine polynomials p_n that interpolate f at the scaled zeros of the $(n+1)$st degree Chebyshev polynomial for $\mathbb{K}$. The zeros are scaled to make the endpoints of $\mathbb{K}$ interpolation points, i.e., we interpolate at the points

$$\hat{\mathbb{K}}_{n+1} := \left\{ \frac{2\cos(\frac{2j+1}{2(n+1)}\pi)}{\cos(\frac{\pi}{2(n+1)})} : 0 \le j \le n \right\}. \tag{4.3}$$

We refer to the points in (4.3) as Chebyshev points. Thus, for each value of n, the polynomial p_n interpolates f at Chebyshev points. The approximation error

$$\max_{z\in\mathbb{K}} |f(z) - p_n(z)| \tag{4.4}$$

is close to the error achieved with the best polynomial approximant of degree at most n; see, e.g., de Boor [3, Chapter 2]. The smooth curves in Figure 4.5 display the 10-logarithm of the

ETNA
Kent State University
etna@mcs.kent.edu

error (4.4) for polynomials of degree between zero and 300. The rate of convergence is quite slow due to the branch point of f at $z = -2$.

In order to be able to evaluate polynomials of high degree, we ordered the points in each set $\hat{\mathbb{K}}_{n+1}$ like Leja points before using them in (4.1) and (4.2). This ordering is determined by replacing the set $\mathbb{K}$ in (1.1) and (1.2) by the set (4.3), and it reduces the propagation of round-off errors; see [14]. We remark that the interpolation polynomial has to be recomputed for each point set $\hat{\mathbb{K}}_{n+1}$, because the intersection $\hat{\mathbb{K}}_n \cap \hat{\mathbb{K}}_{n+1}$ only contains few points.

The second sequence of interpolation polynomials p_n is obtained by using Leja points for the discrete set $\mathbb{K}_{3000}(-2, 2)$ defined by (1.7) as interpolation points. The computed interpolation points approximate Leja points for $\mathbb{K}$, and we refer to the interpolation points as Leja points. The staircase curve on the left-hand side of Figure 4.5 shows the approximation error (4.4) obtained with these interpolation polynomials. The error is fairly close to the error achieved when interpolating at Chebyshev points. Note that interpolation in Leja points can give a smaller error than interpolation at Chebyshev points.

Finally, we interpolate f at fast Leja points for $\mathbb{K}$ generated as described in Example 3.2. The staircase curve on the right-hand side of Figure 4.5 displays the approximation error (4.4). The approximation error achieved is close to the error obtained when interpolating at Chebyshev points, however, the arithmetic work required for generating the sequence of polynomials that interpolate at fast Leja points is much smaller. □

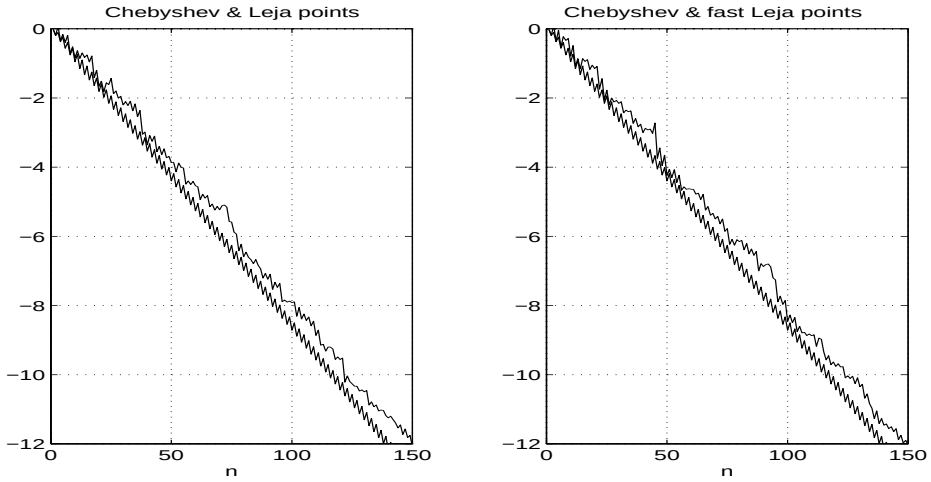

FIG. 4.6. *Example 4.4:* $\log_{10}(\max_{z\in\mathbb{K}} |f(z) - p_n(z)|)$ *for* $1 \le n \le 300$, *where the polynomial* p_n *of the form (4.1) interpolates* $f(z) := (1 + \frac{25}{4}z^2)^{-1}$ *at Chebyshev points (4.3) (jagged curve in both graphs), at Leja points for* $\mathbb{K}_{3000}(-2, 2)$ *(top curve on the left), and at fast Leja points (top curve on the right).*

Example 4.4. Consider the approximation of the Runge function $f(z) := (1 + \frac{25}{4}z^2)^{-1}$ on the interval $\mathbb{K} := [-2, 2]$ by interpolating polynomials p_n given by (4.1). We interpolate f at the same sequences of interpolation points as in Example 4.3. Figure 4.6 displays the computed errors $\max_{z\in\mathbb{K}} |f(z) - p_n(z)|$ and is analogous to Figure 4.5. In particular, the figure shows that interpolation at fast Leja points gives the same rate of convergence as interpolation at Chebyshev points as the degree of the polynomial increases. The convergence is geometric, because f is analytic in an open set containing $\mathbb{K}$. □

The fact that polynomial interpolation at fast Leja points in the Examples 4.3 and 4.4 gives about the same error as interpolation in Chebyshev points, suggests that fast Leja points and Chebyshev points are distributed similarly already for a fairly small number of points.

ETNA
Kent State University
etna@mcs.kent.edu

The distribution of fast Leja points is presently being investigated.

4.2. Eigenvalue computation. The determination of a few, say k, eigenvalues and associated eigenvectors of a large sparse symmetric matrix $A \in \mathbb{R}^{n\times n}$, $n \gg k$, is an important computational problem that arises in many applications. The difficulties associated with determining eigenvalues has spurred considerable research; see, e.g., Saad [15] for a survey and references. Recently, Sorensen [17] proposed the Implicitly Restarted Arnoldi (IRA) method for the computation of a few eigenvalues of a large sparse nonsymmetric matrix, and the closely related Implicitly Restarted Lanczos (IRL) method for the computation of a few eigenvalues of a large sparse symmetric matrix. Improvements and analyses of these methods have been presented by Lehoucq and Sorensen [9, 10]. ARPACK by Lehoucq et al. [11] implements these methods.

The IRA and IRL methods can be regarded as curtailed QR algorithms for the nonsymmetric and symmetric eigenvalue problems, respectively. Similarly as in the QR algorithms, the choice of shifts is important for the performance of the IRA and IRL methods. The IRA method generates a sequence of small Hessenberg matrices, whose spectral factorizations are computed. In the IRL method, the Hessenberg matrices generated are real and symmetric. Lehoucq and Sorensen [9, 10, 17] propose to use some of the computed eigenvalues as shifts. This approach has recently been analyzed by Morgan [13]; see also Calvetti et al. [6] and Stathopoulos et al. [18] for related discussions.

The convergence of the IRA and IRL methods can be studied by considering certain polynomial approximation problems. Specifically, one would like to determine an accelerating polynomial that is of large magnitude in areas of the complex plane that contain the k eigenvalues of interest and small elsewhere. For instance, when computing the k smallest eigenvalues of a large symmetric matrix, we would like the polynomial to be large in an interval that contains these k eigenvalues, and small on an interval that contains the remaining eigenvalues. The eigenvalues of the sequence of the small symmetric tridiagonal matrices generated by the IRL method help us determine a sequence of intervals $[a_j, b_j]$, $j = 1, 2, \ldots$, that do not contain any of the desired k smallest eigenvalues of the matrix, but many of the undesired ones. We describe in [1, 2] how a polynomial $p_q^{(j)}$ that is small on the interval $[a_j, b_j]$ can be determined by letting it have Leja points $\{z_k^{(j)}\}_{k=0}^{q-1}$ for a set $\mathbb{K}_m(a_j, b_j)$ as zeros. Thus,

$$p_q^{(j)}(z) := c_j \prod_{k=0}^{q-1} (z - z_k^{(j)}), \tag{4.5}$$

where c_j is a scaling factor of no significance for the convergence. When allocating the Leja points $\{z_k^{(j)}\}_{k=0}^{q-1}$, the presence of the points $\cup_{r=1}^{j-1}\{z_k^{(r)}\}_{k=0}^{q-1}$ for previous intervals $[a_r, b_r]$, $1 \le r < j$, is taken into account; see [1, 2] for details. The accelerating effect of all the polynomials is described by the product polynomial

$$p_{jq}(z) := \prod_{r=1}^{j} p_q^{(r)}(z). \tag{4.6}$$

Details on how Leja points can be applied to compute a few extreme or nonextreme eigenvalues can be found in [1, 2]. The points $z_k^{(j)}$ are analogous to the shifts in the QR algorithm. We therefore refer to the $z_k^{(j)}$ as Leja shifts in the context of eigenvalue computations.

Leja shifts are compared to Chebyshev shifts in [2, Example 5.2]. The latter are defined as follows. For each $r = 1, 2 \ldots$ the Chebyshev shifts $\{z_k^{(r)}\}_{j=0}^{q-1}$ are the zeros of the Chebyshev polynomial of the first kind of degree q for the interval $[a_r, b_r]$. Chebyshev shifts are

ETNA
Kent State University
etna@mcs.kent.edu

found to yield slower convergence than Leja shifts. This depends on that Chebyshev shifts for the interval $[a_r, b_r]$ are distributed independently of already allocated Chebyshev shifts for the intervals $[a_s, b_s]$ for $s < r$. The allocation of fast Leja points for $[a_r, b_r]$, on the other hand, takes the distribution of previously allocated fast Leja points into account; see [2] for further discussion. An interactive example that demonstrates this is available at web site http://etna.mcs.kent.edu/xxx.

We propose the use of fast Leja points for the sequence of intervals $[a_j, b_j]$, $j = 1, 2, \ldots$, to define the accelerating polynomials (4.5) and (4.6). The examples below compare Leja points for the sets $\mathbb{K}_m(a_j, b_j)$ with $m = 1000$ or $m = 3000$ with fast Leja points for the intervals $[a_j, b_j]$. The number of matrix-vector product evaluations with the matrices A are compared. We also tabulate the CPU time required for computing Leja points for $\mathbb{K}_{1000}(a_j, b_j)$ and $\mathbb{K}_{3000}(a_j, b_j)$, and fast Leja points for $[a_j, b_j]$. We remark that when computing Leja points for sets $\mathbb{K}_m(a_j, b_j)$, the polynomial has to be evaluated at each one of the m points of $\mathbb{K}_m(a_j, b_j)$ for every change of interval $[a_j, b_j]$. This evaluation can be expensive when the polynomial is of high degree. The degree equals the number of evaluations of matrix-vector products with the matrix A. Fast Leja points makes these evaluations unnecessary, and this contributes to making fast Leja points faster to determine than Leja points for the sets $\mathbb{K}_m(a_j, b_j)$.

TABLE 4.1
Example 4.5: $A = \mathrm{diag}(1, 2, 3, \ldots, 2500)$

Shifts	CPU time for shifts	# matrix-vector products
# Lanczos vectors = 5		
fast Leja points	0.20 sec.	510
Leja points for sets $\mathbb{K}_{1000}(a, b)$	12.29 sec.	551
Leja points for sets $\mathbb{K}_{3000}(a, b)$	37.23 sec.	553
# Lanczos vectors = 10		
fast Leja points	0.27 sec.	540
Leja points for sets $\mathbb{K}_{1000}(a, b)$	5.07 sec.	526
Leja points for sets $\mathbb{K}_{3000}(a, b)$	13.7 sec.	507
# Lanczos vectors = 15		
fast Leja points	0.17 sec.	508
Leja points for sets $\mathbb{K}_{1000}(a, b)$	3.05 sec.	510
Leja points for sets $\mathbb{K}_{3000}(a, b)$	8.58 sec.	497

Example 4.5. We wish to compute the three smallest eigenvalues of the matrix

$$A = \mathrm{diag}(1, 2, 3, \ldots, 2500), \tag{4.7}$$

by Algorithm 4.1 in [1] with fast Leja points as shifts. We set the tolerance of the algorithm to $1 \cdot 10^{-4}$. Table 4.1 reports the number of matrix-vector products required and the CPU time necessary to compute the fast Leja points. For comparison, we also used Leja points for sets $\mathbb{K}_m(a_j, b_j)$ as shifts, for $m = 1000$ and $m = 3000$. Table 4.1 shows the number of matrix-vector product evaluations to be about the same for the different choices of shifts, however, the computation of fast Leja points is much faster than the determination of Leja points for the sets $\mathbb{K}_m(a_j, b_j)$. □

Example 4.6. This example differs from Example 4.5 only in the choice of matrix A. Thus, we let $A = \mathrm{diag}(a_1, a_2, \ldots, a_{100})$ have entries $a_j := \frac{j^2}{100}$ and compute the three

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 4.2
Example 4.6: $A = \mathrm{diag}(a_1, a_2, \ldots, a_{100})$ *with entries* $a_i = \frac{i^2}{100}$, $1 \leq i \leq 100$

Shifts	CPU time for shifts	# matrix-vector products
# Lanczos vectors = 5		
fast Leja points	0.09 sec.	369
Leja points for sets $\mathbb{K}_{1000}(a, b)$	5.74 sec.	340
Leja points for sets $\mathbb{K}_{3000}(a, b)$	21.25 sec.	374
# Lanczos vectors = 10		
fast Leja points	0.15 sec.	378
Leja points for sets $\mathbb{K}_{1000}(a, b)$	2.69 sec.	368
Leja points for sets $\mathbb{K}_{3000}(a, b)$	6.93 sec.	344

smallest eigenvalues by Algorithm 4.1 of [1] with the tolerance set to $1 \cdot 10^{-3}$. Table 4.2 reports the number of matrix-vector products required and the CPU time for the computation of the shifts. □

Example 4.7. Let $A = \mathrm{diag}(a_1, a_2, \ldots, a_{500})$ with $a_{2j} := \sqrt{j}$ and $a_{2j-1} := -\sqrt{j}$, $1 \leq j \leq 250$. We want to compute the smallest positive and largest negative eigenvalues of A, as well as corresponding eigenvectors, by Algorithm 4.2 described in [1]. Since the desired eigenvalues are in the middle of the spectrum, the accelerating polynomials have to be small on a sequence of pairs of intervals $[a_j, b_j] \cup [a'_j, b'_j]$, such that $b_j < 0 < a'_j$. We store eight Lanczos vectors and set the tolerance in the algorithm to $1 \cdot 10^{-3}$. The computation of the desired eigenpairs with fast Leja points as shifts required 190 matrix-vector products with the matrix A. When, instead, we used Leja points for sequences of sets $\mathbb{K}_{4000}(a_j, b_j) \cup \mathbb{K}_{4000}(a'_j, b'_j)$ as shifts, 218 matrix-vector product evaluations are required.

5. Conclusion. Fast Leja points are introduced, and shown to have desirable properties for polynomial approximation and eigenvalue computation. The ease of their computation makes them attractive to use in these applications.

Acknowledgment. We would like to thank Norm Levenberg for discussions.

REFERENCES

[1] J. Baglama, D. Calvetti and L. Reichel, *Iterative methods for the computation of a few eigenvalues of a large symmetric matrix*, BIT, 36 (1996), pp. 400–421.
[2] J. Baglama, D. Calvetti, L. Reichel and A. Ruttan, *Computation of a few small eigenvalues of a large matrix with application to liquid crystal modeling*, J. Comput. Phys., 146 (1998), pp. 203–226.
[3] C. de Boor, *A Practical Guide to Splines*, Springer, New York, 1978.
[4] D. Calvetti and L. Reichel, *Adaptive Richardson iteration based on Leja points*, J. Comput. Appl. Math., 71 (1996), pp. 267–286.
[5] D. Calvetti and L. Reichel, *An adaptive Richardson iteration method for indefinite linear systems*, Numer. Algorithms, 12 (1996), pp. 125–149.
[6] D. Calvetti, L. Reichel and D. C. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electron. Trans. Numer. Anal., 2 (1994), pp. 1–21.
[7] A. Edrei, *Sur les déterminants récurrents et les singularités d'une fonction donnée par son developpement de Taylor*, Compositio Math., 7 (1939), pp. 20–88.
[8] H. Ehlich and K. Zeller, *Schwankung von polynomen zwischen gitterpunkten*, Math. Z., 86 (1964), pp. 41–44.
[9] R. B. Lehoucq, *Analysis and implementation of an implicitly restarted Arnoldi iteration*, Ph.D. Thesis, Rice University, Houston, 1995.
[10] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 789–821.

ETNA
Kent State University
etna@mcs.kent.edu

[11] R. B. Lehoucq, D. C. Sorensen and C. Yang,*ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.
[12] F. Leja, *Sur certaines suits liées aux ensemble plan et leur application à la representation conforme*, Ann. Polon. Math., 4 (1957), pp. 8–13.
[13] R. B. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65 (1996), pp. 1213–1230.
[14] L. Reichel, *Newton interpolation at Leja points*, BIT, 30 (1990), pp. 332–346.
[15] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Halstead Press, New York, 1992.
[16] E. B. Saff and V. Totik, *Logarithmic Potentials with Extremal Fields*, Springer, Berlin, 1997.
[17] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[18] A. Stathopoulos, Y. Saad and K. Wu, *Dynamic thick restarting of the Davidson and the implicitly restarted Arnoldi methods*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.
[19] J. L. Walsh, *Interpolation and Approximation by Rational Functions in the Complex Domain*, 5th ed., Amer. Math. Soc., Providence, RI, 1969.

ETNA
Kent State University
etna@mcs.kent.edu

6. Appendix. We present a MATLAB code for the generation of fast Leja points for an interval $[a, b]$ as described in Example 3.2. The notation used in the MATLAB code is similar to that of Example 3.2.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:                                                                    %
% a      right endpoint of the interval [a,b], presently set to -2              %
% b      left endpoint of the interval [a,b], presently set to 2                %
% nflp   number of fast Leja points to be computed, presently set               %
%        to 500                                                                 %
% zt(j)  fast Leja point                                                        %
% zs(j)  candidate points                                                       %
% index  pointers for candidate points                                          %
%        index(k,1) -> pointer to the fast Leja point to the left               %
%                      of zs(k)                                                 %
%        index(k,2) -> pointer to the fast Leja point to the right              %
%                      of zs(k)                                                 %
% zprod  product |z-zt(k)| over k. The product is evaluated for                 %
%        all candidate points z=zs(j) in the array zs.                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a = -2; b = 2; nflp = 500;
if abs(a) > abs(b), zt = [a,b]; else zt = [b,a]; end
zt(3) = (a+b)/2;
zs(1) = (zt(2)+zt(3))/2; zs(2) = (zt(3)+zt(1))/2;
zprod(1) = prod(zs(1)-zt); zprod(2) = prod(zs(2)-zt);
index(1,1) = 2; index(1,2) = 3; index(2,1) = 3; index(2,2) = 1;
for i = 4:nflp
  [maxval,maxi] = max(abs(zprod));
  zt(i) = zs(maxi);
  index(i-1,1) = i; index(i-1,2) = index(maxi,2); index(maxi,2) = i;
  zs(maxi) = (zt(index(maxi,1))+zt(index(maxi,2)))/2;
  zs(i-1) = (zt(index(i-1,1))+zt(index(i-1,2)))/2;
  zprod(maxi) = prod(zs(maxi)-zt(1:i-1));
  zprod(i-1) = prod(zs(i-1)-zt(1:i-1));
  zprod = zprod.*(zs-zt(i));
end
```

The determination of the candidate points $zs(j)$ is accomplished by using an array $index$, where $index(k, 1)$ points to the fast Leja point after the candidate point $zs(k)$, and $index(k, 2)$ points to the fast Leja point before the candidate point $zs(k)$. The use of the array $index$ makes it possible to determine new candidate points without explicitly ordering the fast Leja points.