MATHEMATICS OF COMPUTATION
Volume 89, Number 326, November 2020, Pages 2735–2752
https://doi.org/10.1090/mcom/3553
Article electronically published on June 24, 2020

# DÖRFLER MARKING WITH MINIMAL CARDINALITY IS A LINEAR COMPLEXITY PROBLEM

CARL-MARTIN PFEILER AND DIRK PRAETORIUS

Abstract. Most adaptive finite element strategies employ the Dörfler marking strategy to single out certain elements $\mathcal{M} \subseteq \mathcal{T}$ of a triangulation $\mathcal{T}$ for refinement. In the literature, different algorithms have been proposed to construct $\mathcal{M}$, where usually two goals compete. On the one hand, $\mathcal{M}$ should contain a minimal number of elements. On the other hand, one aims for linear costs with respect to the cardinality of $\mathcal{T}$. Unlike expected in the literature, we formulate and analyze an algorithm, which constructs a minimal set $\mathcal{M}$ at linear costs. Throughout, pseudocodes are given.

## 1. Introduction

In the last decade, the mathematical understanding of adaptive finite element methods (AFEM) has matured. For many elliptic model problems, one can mathematically prove that AFEM leads to optimal convergence behavior; see, e.g., [Dör96, MNS00, BDD04, Ste07, CKNS08] for some of the seminal works for symmetric problems, [MN05, CN12, FFP14] for the extension to nonsymmetric problems, or [CFPP14] for a recent review on the state of the art.

Starting from an initial mesh $\mathcal{T}_0$, the usual AFEM algorithms iterate the loop

$$\boxed{\texttt{solve}} \quad \to \quad \boxed{\texttt{estimate}} \quad \to \quad \boxed{\texttt{mark}} \quad \to \quad \boxed{\texttt{refine}}. \tag{1}$$

The latter generates a sequence $(\mathcal{T}_\ell)_{\ell\in\mathbb{N}_0}$ of successively refined meshes together with the associated FEM solutions $u_\ell$ and a posteriori error estimators $\eta_\ell = [\sum_{T\in\mathcal{T}_\ell} \eta_\ell(T)^2]^{1/2}$, where the index $\ell$ is the step counter of the adaptive loop. Formally, the algorithm reads as follows: For all $\ell = 0, 1, 2, \dots$, iterate the following steps:

| | |
|---|---|
| `solve` | Compute the FEM solution $u_\ell$ corresponding to $\mathcal{T}_\ell$. |
| `estimate` | Compute certain refinement indicators $\eta_\ell(T)$ for all $T \in \mathcal{T}_\ell$. |
| `mark` | Determine a subset of elements $\mathcal{M}_\ell \subseteq \mathcal{T}_\ell$ for refinement. |
| `refine` | Generate a new mesh $\mathcal{T}_{\ell+1}$ by refinement of (at least) all marked elements. |

Received by the editor July 31, 2019, and, in revised form, February 25, 2020.
2010 *Mathematics Subject Classification.* Primary 65N50, 65N30, 68Q25.
*Key words and phrases.* Dörfler marking criterion, adaptive finite element method, optimal complexity.
The authors thankfully acknowledge support by the Austrian Science Fund (FWF) through the doctoral school *Dissipation and dispersion in nonlinear PDEs* (grant W1245) and through the research project *Optimal adaptivity for BEM and FEM-BEM coupling* (grant P27005).
The first author is the corresponding author.

Usually, the set $\mathcal{M}_\ell$ from `mark` then contains the elements with the largest contributions $\eta_\ell(T)$. Often (and, in particular, for the analysis of rate optimality [CFPP14]), the Dörfler marking criterion [Dör96] is used: Given $0 < \theta \leq 1$, construct $\mathcal{M}_\ell \subseteq \mathcal{T}_\ell$ such that

$$\theta \eta_\ell^2 \leq \sum_{T \in \mathcal{M}_\ell} \eta_\ell(T)^2, \tag{2}$$

i.e., the marked elements control a fixed percentage $\theta$ of the overall error estimator. Clearly, one aims to choose the set $\mathcal{M}_\ell$ with as few elements as possible.

As far as convergence of AFEM is concerned, also other marking criteria can be considered [MSV08, Sie11]. Current proofs of rate optimality of AFEM, however, rely on the (quasi-) minimal Dörfler marking (2), where the set $\mathcal{M}_\ell$ has to be chosen with minimal cardinality (at least up to some $\ell$-independent generic constant); see [CFPP14]. Moreover, when the focus comes to the overall computational cost of AFEM, it is important that all steps of the adaptive algorithm can be performed at linear cost with respect to the number of elements $\#\mathcal{T}_\ell$; see [Ste07, GHPS18, FHPS19]. This is usually a reasonable assumption if `solve` employs iterative solvers like PCG [FHPS19] or multigrid [Ste07], and it requires appropriate data structures for `estimate` and `refine`.

If `mark` aims for a set $\mathcal{M}_\ell$, which satisfies (3) with minimal cardinality, then linear cost is less obvious: The work [Dör96] notes that a possible strategy is to sort the indicators, which, however, results in log-linear costs. Instead, the work [Ste07] employs an approximate sorting by binning. While this leads to linear costs, the resulting set $\mathcal{M}_\ell$ has only minimal cardinality up to a multiplicative factor 2, and [Ste07, Section 5] notes:

> *Selecting $\mathcal{M}_\ell$ that satisfies* (2) *with true minimal cardinality would require sorting all $T \in \mathcal{T}_\ell$ by the values of $\eta_\ell(T)$, which takes $\mathcal{O}(N \log N)$ operations.*

The present work bridges the approaches of [Dör96, Ste07] and proves that the latter statement is wrong: Based on ideas of the (Quick-) Selection algorithm [Hoa61], we present a linear-cost algorithm for `mark`, which provides a set $\mathcal{M}_\ell \subseteq \mathcal{T}_\ell$, which satisfies the Dörfler criterion (2) with minimal cardinality.

The outline of the present work reads as follows: In Section 2, we formulate the Dörfler marking and briefly discuss the algorithms from [Dör96, Ste07]. In Section 3, we present and analyze our new approach for `mark` named ***QuickMark***. Section 3.4 concludes with a `C++11` STL-based implementation of the new algorithm.

## 2. Dörfler marking

2.1. **Setting.** Let $0 < \theta < 1$ and $\mathcal{I} := \{1, \dots, N\}$. Given a vector $x \in \mathbb{R}^N_\star := \big\{x \in \mathbb{R}^N \backslash \{0\} \colon x_j \geq 0 \text{ for all } j \in \mathcal{I}\big\}$, an index set $\mathcal{M} \subseteq \mathcal{I}$ satisfies the *Dörfler criterion*, if

$$\theta \sum_{j \in \mathcal{I}} x_j \leq \sum_{j \in \mathcal{M}} x_j. \tag{3}$$

By $\#\mathcal{M}$, we denote the number of elements in $\mathcal{M}$. Let $N_{\min} := \min\big\{\#\mathcal{M} \colon \mathcal{M} \subseteq \mathcal{I} \text{ satisfies (3)}\big\}$ denote the minimal number of indices which are required to satisfy

the Dörfler criterion (3). We note that the minimizing set is not unique in general, e.g., if $x_i = x_j$ for all $i, j \in \mathcal{I}$ and $0 < \theta \leq (N-1)/N$.

*Remark* 1. For $\theta = 1$, the set $\mathcal{M} \subseteq \mathcal{I}$ of minimal cardinality satisfying (3) is unique and given by $\mathcal{M} := \{j \in \mathcal{I} \colon x_j > 0\}$. Clearly, this set can be determined at linear costs.

We say that an algorithm realizes the *minimal Dörfler marking*, if, for all $0 < \theta < 1$, for all $N \in \mathbb{N}$, and for all $x \in \mathbb{R}^N_\star$, the algorithm constructs a set $\mathcal{M} \subseteq \mathcal{I}$, which satisfies (3) with $\#\mathcal{M} = N_{\min}$. We say that an algorithm realizes the *quasi-minimal Dörfler marking*, if, for all $0 < \theta < 1$, there exists a constant $C \geq 1$ such that, for all $N \in \mathbb{N}$ and for all $x \in \mathbb{R}^N_\star$, the algorithm constructs a set $\mathcal{M} \subseteq \mathcal{I}$, which satisfies (3) with $\#\mathcal{M} \leq C\, N_{\min}$.

For current proofs of rate optimality of AFEM, the marking algorithm has to realize the quasi-minimal Dörfler marking [CFPP14], while available results on optimal computational costs require also that the marking step has linear costs [Ste07, GHPS18, FHPS19].

2.2. **Minimal Dörfler marking based on sorting.** It is already noted in [Dör96] that a set $\mathcal{M} \subseteq \mathcal{I}$, which satisfies (3) as well as $\#\mathcal{M} = N_{\min}$, can easily be constructed by sorting.

**Algorithm 2.** *For the setting from Section* 2.1*, perform the following steps* (i)–(iii)*:*

(i) *Determine a permutation* $\pi \colon \mathcal{I} \to \mathcal{I}$ *such that* $x_{\pi(1)} \geq x_{\pi(2)} \geq \cdots \geq x_{\pi(N)}$.
(ii) *Compute* $v := \theta \sum_{j=1}^N x_j$.
(iii) *Determine the minimal index* $n \in \{1, \dots, N\}$ *such that* $v \leq \sum_{i=1}^n x_{\pi(i)}$.

***Output:*** $\mathcal{M} := \{\pi(1), \dots, \pi(n)\}$.

In practice, step (i) of Algorithm 2 will be performed by sorting the vector $x \in \mathbb{R}^N_\star$. This leads to $\mathcal{O}(N \log N)$ operations for, e.g., the Introsort algorithm [Mus97].

**Proposition 3.** *The set* $\mathcal{M}$ *generated by Algorithm* 2 *satisfies* (3) *as well as* $\#\mathcal{M} = N_{\min}$*, i.e., Algorithm* 2 *realizes the minimal Dörfler marking. Up to step* (i)*, the computational cost of Algorithm* 2 *is linear.*

*Proof.* Let $\mathcal{M}_{\min} \subseteq \mathcal{I}$ satisfy (3) with $\#\mathcal{M}_{\min} = N_{\min}$. By construction of $\mathcal{M} = \{x_{\pi(1)}, \dots, x_{\pi(n)}\}$, it holds that

$$\sum_{i=1}^{n-1} x_{\pi(i)} < v \leq \sum_{j \in \mathcal{M}_{\min}} x_j \leq \sum_{i=1}^{n} x_{\pi(i)}\,.$$

Hence, we see that $n - 1 < \#\mathcal{M}_{\min} = N_{\min} \leq n$. This implies that $n = N_{\min}$. It is obvious that steps (ii)–(iii) of Algorithm 2 have linear cost $\mathcal{O}(N)$. □

2.3. **Dörfler marking without sorting.** To avoid sorting, the work [Dör96] proposes (a variant of) the following algorithm; see [Dör96, Section 5.2].

**Algorithm 4.** *For the setting from Section* 2.1 *and given* $0 < \nu < 1$*, perform the following steps* (i)–(vi)*:*

(i) *Initialize* $n := 0$*, and* $\pi(j) := 0$ *for all* $j = 1, \dots, N$.
(ii) *Compute* $v := \theta \sum_{j=1}^N x_j$ *and* $M := \max_{i=1,\dots,N} x_i$.
(iii) *For* $k = 1, 2, 3, \dots, \lceil 1/\nu \rceil$*, iterate the following steps:*
  (iv) *For all* $i = 1, \dots, N$ *with* $i \notin \{\pi(j)\colon j = 1, \dots, n\}$*, iterate the following steps:*
    (v) *if* $x_i > (1-k\nu)M$*, then define* $\pi(n+1) := i$ *and update* $n \mapsto n+1$,
    (vi) *if* $v \le \sum_{j=1}^n x_{\pi(j)}$*, then terminate.*

***Output:*** $\mathcal{M} := \{\pi(1), \dots, \pi(n)\}$.

*Remark* 5. The algorithm proposed in [Dör96, Section 5.2] has the stopping criterion (vi) as part of step (iii), i.e., steps (iv)–(v) are iterated, until $v \le \sum_{j=1}^N x_{\pi(j)}$. If $x$ is constant, i.e., $x_j = c > 0$ for all $j \in \mathcal{I}$, then this variant leads to $\mathcal{M} = \mathcal{I}$ for all $0 < \theta \le 1$ and hence does not realize quasi-minimal Dörfler marking. Our formulation of Algorithm 4 excludes such a simple counterexample.

**Proposition 6.** *Algorithm* 4 *terminates after finitely many steps. The computational cost of Algorithm* 4 *is* $\mathcal{O}(N/\nu)$*. The set* $\mathcal{M}$ *generated by Algorithm* 4 *realizes* (3)*, but it is not quasi-minimal in general.*

*Proof.* Steps (i)–(ii) have linear costs $\mathcal{O}(N)$. Obviously, if in step (vi) the sum is rather updated than recomputed, steps (iii)–(vi) lead to total costs $\mathcal{O}(N/\nu)$ for Algorithm 4. To see that $\mathcal{M}$ satisfies (3), note that (at latest) for $k = \lceil 1/\nu \rceil$, it holds that $k\nu \ge 1$ and hence $x_i > (1 - k\nu)M$ is satisfied for all $x_i \neq 0$. It only remains to show that Algorithm 4 does not realize the quasi-minimal Dörfler marking.

Consider arbitrary $0 < \theta < 1$, $0 < \nu < 1$ and any constant $C \ge 1$. Then, there exist $N \in \mathbb{N}$ and $x \in \mathbb{R}^N_\star$ such that the set $\mathcal{M}$ generated by Algorithm 4 satisfies $\#\mathcal{M} > CN_{\min}$. An explicit, but rather technical construction of such an $x \in \mathbb{R}^N_\star$ of the form $x = (1, \varepsilon, \dots, \varepsilon, \delta, \dots, \delta) \in \mathbb{R}^N_\star$ where $0 < \varepsilon \ll \delta \ll 1$ is carried out in the extended preprint [PP19] of this work. □

2.4. **Quasi-minimal Dörfler marking with linear complexity by binning.** The following strategy has been proposed in the seminal work [Ste07], which gave the first optimality proof for a standard AFEM loop of type (1) for the 2D Poisson problem. The main observation is the following: If the reduction of the threshold in step (v) of Algorithm 4 is done by multiplication instead of subtraction, then the resulting algorithm satisfies the quasi-minimal Dörfler marking. While [Ste07, Section 5] outlines the proposed strategy for the choice $\nu = 1/2$, we work out all details in our proof of Proposition 8.

**Algorithm 7.** *For the setting from Section* 2.1 *and given* $0 < \nu < 1$*, perform the following steps* (i)–(v)*:*

(i) *Compute* $v := \theta \sum_{j=1}^N x_j$ *and* $M := \max_{j=1,\dots,N} x_j$.
(ii) *Determine the minimal* $K \in \mathbb{N}_0$ *with* $\nu^{K+1} M \le \frac{1-\theta}{\theta}\, v/N$.
(iii) *For* $k = 0, \dots, K$*, fill bins* $\mathcal{B}_k := \{j \in \mathcal{I}\colon \nu^{k+1} < x_j/M \le \nu^k\}$ *and define* $\mathcal{B}_{K+1} := \mathcal{I} \setminus \bigcup_{k=0}^K \mathcal{B}_k$.
(iv) *This yields a permutation* $\pi : \mathcal{I} \to \mathcal{I}$ *such that*
  • $x_{\pi(i)} > x_{\pi(j)}$ *for all* $i \in \mathcal{B}_I$ *and* $j \in \mathcal{B}_J$ *with* $I < J$.
(v) *Determine the minimal index* $n \in \{1, \dots, N\}$ *such that* $v \le \sum_{i=1}^n x_{\pi(i)}$.

***Output:*** $\mathcal{M} := \{\pi(1), \dots, \pi(n)\}$.

**Proposition 8.** *For arbitrary $0 < \nu < 1$, Algorithm 7 terminates after finitely many steps. The constructed set $\mathcal{M} \subseteq \mathcal{I}$ satisfies (3) with $\#\mathcal{M} \leq \lceil \nu^{-1} N_{\min} \rceil$. Moreover, a proper implementation of Algorithm 7 leads to a total computational cost of $\mathcal{O}\big(N + K\big)$ with $K = \mathcal{O}\big(\log_{1/\nu}(N/(1-\theta))\big)$.*

*Proof.* The only nonobvious statement is the bound $\#\mathcal{M} \leq \lceil \nu^{-1} N_{\min} \rceil$: For $j \in \mathcal{B}_{K+1}$, it holds that $x_j \leq \nu^{K+1} M \leq \frac{1-\theta}{\theta}\, v/N$ and hence

$$v + \frac{N}{\#\mathcal{B}_{K+1}} \sum_{j \in \mathcal{B}_{K+1}} x_j \leq \frac{v}{\theta} = \sum_{j \in \mathcal{I} \setminus \mathcal{B}_{K+1}} x_j + \sum_{j \in \mathcal{B}_{K+1}} x_j \,.$$

Since $\#\mathcal{B}_{K+1} \leq N$ and $x_j \geq 0$ for all $j \in \mathcal{I}$, it follows that $\bigcup_{k=0}^{k_0} \mathcal{B}_k = \mathcal{I} \setminus \mathcal{B}_{K+1}$ satisfies (3). Let $k_0 \in \mathbb{N}_0$ be the largest index such that $\mathcal{B}_{k_0} \subseteq \mathcal{M}$. If no such index exists, i.e., $\mathcal{B}_0 \supsetneq \mathcal{M}$, define $k_0 := -1$. Clearly, it holds that $k_0 \leq K$ and $\bigcup_{k=0}^{k_0} \mathcal{B}_k \subseteq \mathcal{M} \subseteq \bigcup_{k=0}^{k_0+1} \mathcal{B}_k$. Further, there exists $\mathcal{S} \subseteq \mathcal{B}_{k_0+1}$ such that $\mathcal{S} \cup \bigcup_{k=0}^{k_0} \mathcal{B}_k$ satisfies (3) with minimal cardinality $N_{\min}$.

To show $\#\mathcal{M} \leq \lceil \nu^{-1} N_{\min} \rceil$, it suffices to show that $\mathcal{R} := \mathcal{M} \cap \mathcal{B}_{k_0+1}$ satisfies $\#\mathcal{R} \leq \lceil \nu^{-1} \#\mathcal{S} \rceil$. Consider $\#\mathcal{R} > 0$. Then, $k_0 < K$, $\pi(n) \in \mathcal{R}$, and with $v' := v - \sum_{k=0}^{k_0} \sum_{j \in \mathcal{B}_k} x_j$, it holds that

$$\nu^{k_0+2} M (\#\mathcal{R} - 1) < \sum_{j \in \mathcal{R} \setminus \{\pi(n)\}} x_j < v' \leq \sum_{j \in \mathcal{S}} x_j \leq \nu^{k_0+1} M \#\mathcal{S} \,.$$

It immediately follows that $\#\mathcal{R} \leq \lceil \nu^{-1} \#\mathcal{S} \rceil$. Altogether, $\mathcal{M}$ satisfies (3) with $\#\mathcal{M} \leq \lceil \nu^{-1} N_{\min} \rceil$. □

## 3. Minimal Dörfler marking with linear complexity

This section constitutes the main contribution of this work.

**Theorem 9.** *Dörfler marking with minimal cardinality is a linear complexity problem. More precisely, a call of Algorithm 10 below with a vector $x \in \mathbb{R}^N_\star$ leads after $\mathcal{O}(N)$ operations to a set $\mathcal{M} \subseteq \{1, \dots, N\}$ with (3) and $\#\mathcal{M} = N_{\min}$.*

We prove this main theorem explicitly by introducing the ***QuickMark*** algorithm in Section 3.1. The correctness of the ***QuickMark*** algorithm is proved in Section 3.2 and the linear complexity of ***QuickMark*** is shown in Section 3.3. Section 3.4 concludes with some remarks on the implementation of the algorithm.

### 3.1. The *QuickMark* algorithm.

Adapting the divide-and-conquer strategy of efficient selection algorithms [Hoa61], we propose a new strategy to determine, at linear costs, a subset $\mathcal{M} \subseteq \{1, \dots, N\}$ with (3) and $\#\mathcal{M} = N_{\min}$. The proposed algorithm consists of an initial call (Algorithm 10) and the function ***QuickMark*** (Algorithm 11), which steers the divide-and-conquer strategy based on the subroutines ***Pivot*** (Algorithm 12) and ***Partition*** (Algorithm 13).

To improve readability throughout this chapter, whenever a permutation $\pi$ on $\{1, \dots N\}$ would be altered by a function, that function instead is written to take the permutation as input $\pi_{\text{old}}$ and returns as output the new permutation $\pi_{\text{new}}$. If a permutation is not changed by a function, it is simply denoted by $\pi$. Moreover, let $\pi_{\text{id}}$ represent the identity permutation on $\{1, \dots, N\}$, i.e., $\pi_{\text{id}}(j) = j$ for all $j \in \{1, \dots, N\}$. For an index set $\mathcal{J} \subseteq \{1, \dots, N\}$ define $\pi(\mathcal{J}) := \{\pi(j) \colon j \in \mathcal{J}\}$.

**Algorithm 10** (Initial call of ***QuickMark***)**.** *For the setting from Section* 2.1*, we perform the following steps* (i)–(iv)*:*

(i) *Initialize the identity permutation* $\pi_{\text{old}} := \pi_{\text{id}}$.
(ii) *Define lower index* $\ell := 1$ *and upper index* $u := N$.
(iii) *Compute the goal value* $v := \theta \sum_{j=1}^{N} x_j$.
(iv) *Call* $[\pi_{\text{new}}, n] :=$ ***QuickMark***$(x, \pi_{\text{old}}, \ell, u, v)$

***Output:*** $\mathcal{M} := \pi_{\text{new}}(\{1, \dots, n\})$.

Analogously to selection algorithms [Hoa61], the ***QuickMark*** algorithm is based on the subroutine ***Partition***, where elements are essentially separated into two classes: Those elements with smaller value than the pivot element, and those with greater value than the pivot element. Then the algorithm decides which of the two classes is not to be inspected further.

**Algorithm 11** ($[\pi_{\text{new}}, n] =$ ***QuickMark***$(x, \pi_{\text{old}}, \ell, u, v)$)**.** ***Input:*** *Vector* $x \in \mathbb{R}^N$*, permutation* $\pi_{\text{old}}$ *on* $\{1, \dots, N\}$*, goal value* $v \in \mathbb{R}_{>0}$*, lower and upper indices* $1 \le \ell \le u \le N$.

(i) *Determine a pivot index* $[p] :=$ ***Pivot***$(x, \pi_{\text{old}}, \ell, u)$.
(ii) *Determine a new permutation via* $[\pi_{\text{new}}, g, s] :=$ ***Partition***$(x, \pi_{\text{old}}, \ell, u, p)$.
(iii) *Compute the sum of the greatest elements* $\sigma_g := \sum_{j=\ell}^{g} x_{\pi_{\text{new}}(j)}$.
(iv) *If* $\sigma_g \ge v$*, then return* ***QuickMark***$(x, \pi_{\text{new}}, \ell, g, v)$.
(v) *Else, if* $\sigma_g + (s-g-1)x_{\pi_{\text{old}}(p)} \ge v$*, then return* $[\pi_{\text{new}}, g + \lceil (v-\sigma_g)/x_{\pi_{\text{old}}(p)} \rceil]$.
(vi) *Else return* ***QuickMark***$(x, \pi_{\text{new}}, s, u, v - \sigma_g - (s-g-1)x_{\pi_{\text{old}}(p)})$.

***Output:*** *Permutation* $\pi_{\text{new}}$ *of* $\{1, \dots, N\}$ *and index* $n \in \{1, \dots, N\}$.

The ***Pivot*** subroutine should determine a feasible pivot element of a given (sub-) array. While the concrete choice of the pivot strategy is irrelevant for the correctness of the procedure, it is the decisive factor for the computational complexity of the divide-and-conquer strategy. For now, we consider an arbitrarily (e.g., randomly) chosen $p \in \{\ell, \dots, u\}$. While in Section 3.2 correctness of the algorithm is proved independently of the concrete pivot strategy, in Section 3.3 we propose a pivot strategy that leads — even in the worst case — to linear complexity $\mathcal{O}(N)$ of Algorithm 10.

**Algorithm 12** ($[p] =$ ***Pivot***$(x, \pi, \ell, u)$)**.** ***Input:*** *Vector* $x \in \mathbb{R}^N$*, permutation* $\pi$ *on* $\{1, \dots, N\}$*, lower and upper indices* $1 \le \ell \le u \le N$.

(i) *Use* $x_{\pi(\ell)}, x_{\pi(\ell+1)}, \dots, x_{\pi(u)}$ *to determine a pivot index* $p \in \{\ell, \dots, u\}$.

***Output:*** *Pivot index* $p \in \{\ell, \dots, u\}$.

For a given pivot element, the ***Partition*** subroutine reorganizes the elements of a (sub-) array depending on whether they are greater than, smaller than, or equal to the pivot.

**Algorithm 13** ($[\pi_{\text{new}}, g, s] =$ ***Partition***$(x, \pi_{\text{old}}, \ell, u, p)$)**.** ***Input:*** *Vector* $x \in \mathbb{R}^N$*, permutation* $\pi_{\text{old}}$ *on* $\{1, \dots, N\}$*, lower and upper indices* $1 \le \ell \le u \le N$*, pivot index* $\ell \le p \le u$.

(i) *Compute a permutation* $\pi_{\text{mod}}$ *on* $\{\ell, \dots, u\}$ *together with the unique indices* $g \in \{\ell - 1, \dots, u - 1\}$ *and* $s \in \{\ell + 1, \dots, u + 1\}$ *such that the following three implications hold true for all* $j \in \{\ell, \dots, u\}$*:*
- *If* $x_{\pi_{\text{old}}(\pi_{\text{mod}}(j))} > x_{\pi_{\text{old}}(p)}$*, then* $\ell \le j \le g$.

- *If* $x_{\pi_{\mathrm{old}}(\pi_{\mathrm{mod}}(j))} = x_{\pi_{\mathrm{old}}(p)}$, *then* $g < j < s$.
- *If* $x_{\pi_{\mathrm{old}}(\pi_{\mathrm{mod}}(j))} < x_{\pi_{\mathrm{old}}(p)}$, *then* $s \le j \le u$.

(ii) *Define* $\pi_{\mathrm{new}}(j) := \begin{cases} \pi_{\mathrm{old}}(\pi_{\mathrm{mod}}(j)) & \text{for } j \in \{\ell, \dots, u\}, \\ \pi_{\mathrm{old}}(j) & \text{else.} \end{cases}$

***Output:*** *Permutation* $\pi_{\mathrm{new}}$ *of* $\{1, \dots, N\}$ *together with indices* $g \in \{\ell-1, \dots, u-1\}$ *and* $s \in \{\ell+1, \dots, u+1\}$.

The following remark collects some important observations (4)–(5) about the state of $\pi_{\mathrm{old}}$ and $\pi_{\mathrm{new}}$ in Algorithm 13. The validity of (4) will be shown in Proposition 16 in Section 3.2. The properties (5) follow directly from Algorithm 13.

*Remark* 14. When ***Partition*** (Algorithm 13) is called in step (ii) of ***QuickMark*** (Algorithm 11), the permutation $\pi_{\mathrm{old}}$ and the indices $\ell, u$ satisfy

$$x_{\pi_{\mathrm{old}}(j)} > x_{\pi_{\mathrm{old}}(k)} \quad \text{for all} \quad 1 \le j < \ell \le k \le N\,, \tag{4a}$$

$$x_{\pi_{\mathrm{old}}(j)} > x_{\pi_{\mathrm{old}}(k)} \quad \text{for all} \quad 1 \le j \le u < k \le N\,. \tag{4b}$$

This is illustrated in Figure 1.

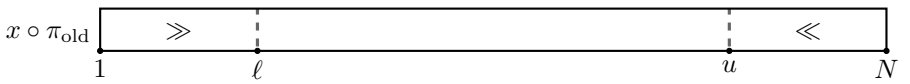


FIGURE 1. Ordering of $x \circ \pi_{\mathrm{old}}$ when ***Partition*** is called, cf. (4). The array $x \circ \pi_{\mathrm{old}}$ is partially sorted in descending order: The $\ell - 1$ strictly largest values in $x \circ \pi_{\mathrm{old}}$ are obtained by the indices $\{1, \dots, \ell - 1\}$. The $N - u$ strictly smallest values in $x \circ \pi_{\mathrm{old}}$ are obtained by the indices $\{u + 1, \dots, N\}$.

The permutation $\pi_{\mathrm{new}}$ defined in step (ii) of Algorithm 13 differs from $\pi_{\mathrm{old}}$ only at the indices $j \in \{\ell, \dots, u\} \subseteq \{1, \dots, N\}$. Consequently, (4a)–(4b) are preserved by $\pi_{\mathrm{new}}$. With the indices $g, s$ returned by Algorithm 13 and $p$ the pivot index, it additionally holds that

$$x_{\pi_{\mathrm{new}}(j)} > x_{\pi_{\mathrm{old}}(p)} \quad \text{for all} \quad \ell \le j \le g\,, \tag{5a}$$

$$x_{\pi_{\mathrm{new}}(j)} = x_{\pi_{\mathrm{old}}(p)} \quad \text{for all} \quad g < j < s\,, \tag{5b}$$

$$x_{\pi_{\mathrm{new}}(j)} < x_{\pi_{\mathrm{old}}(p)} \quad \text{for all} \quad s \le j \le u\,. \tag{5c}$$

This is illustrated in Figure 2.

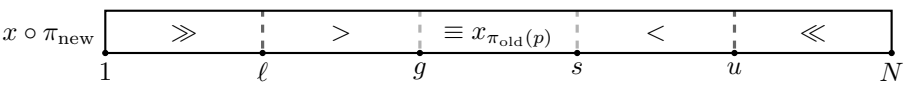


FIGURE 2. Ordering of $x \circ \pi_{\mathrm{new}}$ when ***Partition*** terminates, cf. (5). The properties (4) of $\pi_{\mathrm{old}}$ illustrated in Figure 1 are preserved. Additionally, the reordered array $x \circ \pi_{\mathrm{new}}$ is partially sorted for the indices $\{\ell, \dots, u\}$: Within the index range $\{\ell, \dots, u\}$, the $g - \ell + 1$ strictly largest values in $x \circ \pi_{\mathrm{new}}$ are obtained by the indices $\{\ell, \dots, g\}$, while the $u - s + 1$ strictly smallest values in $x \circ \pi_{\mathrm{new}}$ are obtained by the indices $\{s, \dots, u\}$. In particular, all indices $p'$ with $g < p' < s$ satisfy $x_{\pi_{\mathrm{new}}(p')} = x_{\pi_{\mathrm{old}}(p)}$ with the pivot-index $p$.

3.2. **Correctness of the *QuickMark* algorithm.** We consider $x \in \mathbb{R}^N_\star$, permutations $\pi$ on $\{1,\dots,N\}$, indices $\ell, u \in \{1,\dots,N\}$ with $1 \le \ell \le u \le N$, and a value $v \in \mathbb{R}_{>0}$. Proving the correctness of ***QuickMark*** (Algorithm 11) is organized into three steps: In Section 3.2.1 we verify some essential properties satisfied by the input parameters of calls to Algorithm 11. Section 3.2.2 introduces auxiliary subproblems generated and solved by Algorithm 11 and gives insight on the idea behind the ***QuickMark*** strategy. Termination of Algorithm 11 is investigated in Section 3.2.3, where the correctness is proved.

3.2.1. *Admissible calls to* ***QuickMark****.* We consider the following crucial properties, which will be shown to be always satisfied in Proposition 16.

**Definition 15.** A call $\boldsymbol{QuickMark}(x, \pi_{\mathrm{old}}, \ell, u, v)$ to Algorithm 11 is called **admissible**, if the inputs $x \in \mathbb{R}^N_\star, \pi_{\mathrm{old}}, \ell, u, v$ satisfy the following conditions (a)–(b):

(a) It holds that

$$x_{\pi_{\mathrm{old}}(j)} > x_{\pi_{\mathrm{old}}(k)} \quad \text{for all} \quad 1 \le j < \ell \le k \le N\,, \tag{6a}$$

$$x_{\pi_{\mathrm{old}}(j)} > x_{\pi_{\mathrm{old}}(k)} \quad \text{for all} \quad 1 \le j \le u < k \le N\,. \tag{6b}$$

(b) It holds that

$$0 < v = \theta \sum_{j=1}^{N} x_j - \sum_{j=1}^{\ell-1} x_{\pi_{\mathrm{old}}(j)} \le \sum_{j=\ell}^{u} x_{\pi_{\mathrm{old}}(j)}\,. \tag{7}$$

In fact, the following proposition shows that recursive calls of ***QuickMark*** preserve the admissibility conditions.

**Proposition 16.** *If* ***QuickMark*** *is initially called by Algorithm* 10(iv)*, then each subsequent recursive call* $\boldsymbol{QuickMark}(x, \pi, \ell, u, v)$ *from step* (iv) *or* (vi) *of Algorithm* 11 *is admissible.*

*Proof.* The statement follows directly by induction. First, we show that the initial call $\boldsymbol{QuickMark}(x, \pi_{\mathrm{old}}, \ell, u, v)$ of Algorithm 11 initiated by Algorithm 10(iv) with the inputs $x \in \mathbb{R}^N_\star$, $\pi_{\mathrm{old}} := \pi_{\mathrm{id}}$, $\ell := 1$, $u := N$, and $v := \theta \sum_{j=1}^N x_j$ is admissible: Since $\ell = 1$ and $u = N$, Definition 15(a) contains only statements about indices in the empty set and is therefore satisfied. Definition 15(b) follows from $x \in \mathbb{R}^N_\star$, $0 < \theta < 1$, and the definition of $v$.

For the induction step, consider an admissible call $\boldsymbol{QuickMark}(x, \pi_{\mathrm{old}}, \ell, u, v)$ of Algorithm 11. We show a potential subsequent call $\boldsymbol{QuickMark}(x, \pi_{\mathrm{new}}, \ell', u', v')$ initiated by either Algorithm 11(iv) (i.e., $\ell' = \ell$, $u' = g$, $v' = v$), or by Algorithm 11(vi) (i.e., $\ell' = s$, $u' = u$, $v' = v - \sum_{j=\ell}^{s-1} x_{\pi_{\mathrm{new}}(j)}$), is also admissible: By (a), (b) we refer to the assumption, i.e., the admissibility conditions of Definition 15 satisfied by $\boldsymbol{QuickMark}(x, \pi_{\mathrm{old}}, \ell, u, v)$. We aim to show the corresponding admissibility conditions of Definition 15 for the call $\boldsymbol{QuickMark}(x, \pi_{\mathrm{new}}, \ell', u', v')$, which will be denoted by (a′), (b′).

Recall that in either case (step (iv) or step (vi) in Algorithm 11), $\pi_{\mathrm{new}}$ differs from $\pi_{\mathrm{old}}$ only on the index set $\{\ell,\dots,u\} \subseteq \{1,\dots,N\}$. Therefore, in both cases (a′) follows from (5a)–(5c) and (a). If recursion relies on Algorithm 11(iv), then

$\ell' = \ell$, $u' = g$, and $v' := v \le \sigma_g$. Hence,

$$0 < v' = v \overset{\text{(b)}}{=} \theta \sum_{j=1}^{N} x_j - \sum_{j=1}^{\ell-1} x_{\pi_{\text{old}}(j)} = \theta \sum_{j=1}^{N} x_j - \sum_{j=1}^{\ell'-1} x_{\pi_{\text{new}}(j)} \le \sigma_g = \sum_{j=\ell'}^{u'} x_{\pi_{\text{new}}(j)}$$

proves (b′). If recursion relies on Algorithm 11(vi), then $\ell' = s$, $u' = u$, and

$$v > \sigma_g + (s - g - 1) x_{\pi_{\text{old}}(p)} \overset{\text{(5b)}}{=} \sum_{j=\ell}^{s-1} x_{\pi_{\text{new}}(j)} \,. \tag{8}$$

Combining (b) and the last estimate yields for $v' := v - \sum_{j=\ell}^{s-1} x_{\pi_{\text{new}}(j)}$ that

$$0 \overset{(8)}{<} v' \overset{\text{(b)}}{=} \theta \sum_{j=1}^{N} x_j - \sum_{j=1}^{\ell-1} x_{\pi_{\text{old}}(j)} - \sum_{j=\ell}^{s-1} x_{\pi_{\text{new}}(j)} = \theta \sum_{j=1}^{N} x_j - \sum_{j=1}^{\ell'-1} x_{\pi_{\text{new}}(j)} \overset{\text{(b)}}{\le} \sum_{j=\ell'}^{u'} x_{\pi_{\text{new}}(j)} \,.$$

This shows (b′). □

3.2.2. *Subproblems generated by* ***QuickMark****.* To analyze Algorithm 11, we introduce some auxiliary notation. In particular, the symbol $\mathcal{M}$ will be used differently than in Section 2.1. The connection between the two notations is clarified in Remark 17.

By $\mathfrak{P}(\{\ell, \dots, u\})$, we denote the power set of $\{\ell, \dots, u\}$. For any admissible call ***QuickMark***$(x, \pi, \ell, u, v)$ to Algorithm 11, let $\mathfrak{M}(x, \pi, \ell, u, v) \subseteq \mathfrak{P}(\{\ell, \dots, u\})$ consist of all $\mathcal{M} \in \mathfrak{P}(\{\ell, \dots, u\})$ such that

$$x_{\pi(j)} \ge x_{\pi(k)} \qquad \text{for all } j \in \mathcal{M} \text{ and all } k \in \{\ell, \dots, u\} \setminus \mathcal{M}, \tag{9a}$$

$$\sum_{j \in \mathcal{M}} x_{\pi(j)} \ge v > \sum_{j \in \mathcal{M} \setminus \{k\}} x_{\pi(j)} \qquad \text{for all } k \in \mathcal{M} \,. \tag{9b}$$

The following remark follows immediately from (9a)–(9b) and connects the introduced notation to the Dörfler marking criterion (3) from Section 2.1.

*Remark* 17. For arbitrary $\mathcal{M} \in \mathfrak{M}(x, \pi, 1, N, \theta \sum_{j=1}^{N} x_j)$, the set $\mathcal{M}' := \pi(\mathcal{M}) \in \mathfrak{M}(x, \pi_{\text{id}}, 1, N, \theta \sum_{j=1}^{N} x_j)$ satisfies (3) with minimal cardinality $\#\mathcal{M}' = N_{\min}$.

Later in Section 3.2.3, we will prove that ***QuickMark*** called by Algorithm 10 determines a set $\mathcal{M} \in \mathfrak{M}(x, \pi_{\text{id}}, 1, N, \theta \sum_{j=1}^{N} x_j)$. The core idea behind the proof is the observation that for an admissible call ***QuickMark***$(x, \pi, \ell, u, v)$, the set $\mathfrak{M}(x, \pi_{\text{id}}, 1, N, \theta \sum_{j=1}^{N} x_j)$ can be written as

$$\Big\{ \pi(\{1, \dots, \ell - 1\}) \cup \pi(\mathcal{M}') \colon \mathcal{M}' \in \mathfrak{M}(x, \pi, \ell, u, v) \Big\} \,.$$

Hence, an admissible call ***QuickMark***$(x, \pi_{\text{old}}, \ell, u, v)$ to Algorithm 11 either determines a set $\mathcal{M} \in \mathfrak{M}(x, \pi_{\text{new}}, \ell, u, v)$ and terminates in step (v), or it initiates another admissible recursive call denoted by ***QuickMark***$(x, \pi_{\text{new}}, \ell', u', v')$ in step (iv) or step (vi), where $\{\ell', \dots, u'\} \subsetneqq \{\ell, \dots, u\}$, i.e., the problem is reduced to a strict subproblem.

First, we will show, that all occurring subproblems of finding $\mathcal{M} \in \mathfrak{M}(x, \pi, \ell, u, v)$ are well-posed. In fact, for an admissible call ***QuickMark***$(x, \pi, \ell, u, v)$ the set $\mathfrak{M}(x, \pi, \ell, u, v)$ is always nonempty and all $\mathcal{M} \in \mathfrak{M}(x, \pi, \ell, u, v)$ attain the same minimum in $x \circ \pi$.

**Lemma 18.** *Let* ***QuickMark***$(x, \pi, \ell, u, v)$ *be an admissible call to Algorithm* 11. *Then,* $\mathfrak{M}(x, \pi, \ell, u, v) \neq \emptyset$. *Moreover, the definition*

$$x^*(x, \pi, \ell, u, v) := \min_{j \in \mathcal{M}} x_{\pi(j)} \tag{10}$$

*is independent of the concrete choice of* $\mathcal{M} \in \mathfrak{M}(x, \pi, \ell, u, v)$.

*Proof.* To show $\mathfrak{M}(x, \pi, \ell, u, v) \neq \emptyset$, we explicitly construct some $\mathcal{M} \in \mathfrak{M}(x, \pi, \ell, u, v)$: Starting with $\mathcal{M}_0 := \{\ell, \dots, u\}$, for $i = 0, \dots, u - \ell$ define

$$m_i := \min\{j \in \mathcal{M}_i \colon x_{\pi(j)} = \min_{k \in \mathcal{M}_i} x_{\pi(k)}\} \quad \text{and} \quad \mathcal{M}_{i+1} := \mathcal{M}_i \setminus \{m_i\}\,,$$

i.e., $\mathcal{M}_{i+1}$ is generated by extracting the index with the smallest value in $x \circ \pi$ from $\mathcal{M}_i$. By construction, (9a) holds for all $\mathcal{M}_i$, $i = 0, \dots, u - \ell + 1$. Further, the values $\sum_{j \in \mathcal{M}_i} x_{\pi(j)}$ are monotonically decreasing in $i = 0, \dots, u - \ell + 1$. Since $\mathcal{M}_{u-\ell+1} = \emptyset$, the admissibility (7) of $v$ implies that

$$\sum_{j \in \mathcal{M}_{u-\ell+1}} x_{\pi(j)} = 0 < v \leq \sum_{j=\ell}^{u} x_{\pi(j)} = \sum_{j \in \mathcal{M}_0} x_{\pi(j)}\,.$$

Consequently, there exists a unique $i' \in \{0, \dots, u - \ell\}$ such that

$$\sum_{j \in \mathcal{M}_{i'+1}} x_{\pi(j)} < v \leq \sum_{j \in \mathcal{M}_{i'}} x_{\pi(j)}\,.$$

By construction, for all $i = 0, \dots, u - \ell$ (and in particular for $i = i'$) it holds that

$$\sum_{j \in \mathcal{M}_i \setminus \{k\}} x_{\pi(j)} \leq -m_i + \sum_{j \in \mathcal{M}_i} x_{\pi(j)} = \sum_{j \in \mathcal{M}_{i+1}} x_{\pi(j)} \quad \text{for all } k \in \mathcal{M}_i\,.$$

Hence, combining the last two estimates shows that $\mathcal{M}_{i'}$ also satisfies (9b) and thus $\mathcal{M}_{i'} \in \mathfrak{M}(x, \pi, \ell, u, v)$. This proves $\mathfrak{M}(x, \pi, \ell, u, v) \neq \emptyset$.

To show that the definition (10) is independent of $\mathcal{M} \in \mathfrak{M}(x, \pi, \ell, u, v)$, we claim that

$$x_1^* := \min_{j \in \mathcal{M}_1} x_{\pi(j)} = \min_{j \in \mathcal{M}_2} x_{\pi(j)} =: x_2^* \quad \text{for all} \quad \mathcal{M}_1, \mathcal{M}_2 \in \mathfrak{M}(x, \pi, \ell, u, v)\,.$$

To prove this claim, we argue by contradiction and assume $x_1^* \neq x_2^*$ and, without loss of generality, $x_1^* < x_2^*$. Hence, we have $\mathcal{M}_1 \setminus \mathcal{M}_2 \neq \emptyset$ and

$$x_1^* < x_2^* \leq x_{\pi(k)} \quad \text{for all } k \in \mathcal{M}_2\,.$$

If there exists $k \in \mathcal{M}_2 \setminus \mathcal{M}_1$, then (9a) gives that $x_1^* \geq x_{\pi(k)}$. This contradicts the last estimate and hence proves that $\mathcal{M}_2 \setminus \mathcal{M}_1 = \emptyset$. Therefore, we deduce that $\mathcal{M}_2 \subsetneq \mathcal{M}_1$. Using the second inequality in (9b) for $\mathcal{M}_1$ and then using the first inequality in (9b) for $\mathcal{M}_2$ , we see that

$$v > -x_1^* + \sum_{j \in \mathcal{M}_1} x_{\pi(j)} \geq \sum_{j \in \mathcal{M}_2} x_{\pi(j)} \geq v\,.$$

This contradiction implies that $x_1^* = x_2^*$ and concludes the proof. □

3.2.3. *Termination of* ***QuickMark****.* For any admissible call of Algorithm 11 denoted by ***QuickMark***$(x, \pi_{\mathrm{old}}, \ell, u, v)$, exactly one of Algorithm 11, exactly one of three cases — recursion by step (iv), termination by step (v), or recursion by step (vi) — applies. The next lemma connects the termination in step (v) directly to the pivot index chosen in step (i).

**Lemma 19.** *Let* ***QuickMark****$(x, \pi_{\mathrm{old}}, \ell, u, v)$ be an admissible call to Algorithm* 11*. Then Algorithm* 11 *terminates with step* (v) *if and only if the pivot index* $p \in \{\ell, \dots, u\}$ *from step* (i) *satisfies* $x_{\pi_{\mathrm{old}}(p)} = x^*(x, \pi_{\mathrm{old}}, \ell, u, v)$.

*Proof.* After step (ii) of Algorithm 11, it holds that $\pi_{\mathrm{new}}(\{\ell, \dots, u\}) = \pi_{\mathrm{old}}(\{\ell, \dots, u\})$ and hence $x^*(x, \pi_{\mathrm{new}}, \ell, u, v) = x^*(x, \pi_{\mathrm{old}}, \ell, u, v)$.

First, suppose that Algorithm 11 terminates with step (v), i.e.,

$$\sum_{j=\ell}^{g} x_{\pi_{\mathrm{new}}(j)} = \sigma_g < v \le \sigma_g + (s-g-1)x_{\pi_{\mathrm{old}}(p)} \overset{(5b)}{=} \sum_{j=\ell}^{s-1} x_{\pi_{\mathrm{new}}(j)}\,.$$

Now (5a)–(5c) imply that

$$\{\ell, \dots, g\} \subsetneqq \mathcal{M} \subseteq \{\ell, \dots, s-1\} \quad \text{for all } \mathcal{M} \in \mathfrak{M}(x, \pi_{\mathrm{new}}, \ell, u, v)\,.$$

By definition (10) and (5a)–(5b), it follows that $x_{\pi_{\mathrm{old}}(p)} = x^*(x, \pi_{\mathrm{new}}, \ell, u, v)$.

Conversely, suppose $x_{\pi_{\mathrm{old}}(p)} = x^*(x, \pi_{\mathrm{new}}, \ell, u, v)$ and let $\mathcal{M} \in \mathfrak{M}(x, \pi_{\mathrm{new}}, \ell, u, v)$ be arbitrary. Then, (5a)–(5c) and $x_{\pi_{\mathrm{old}}(p)} = \min_{j\in\mathcal{M}} x_{\pi_{\mathrm{new}}(j)}$ imply that

$$\{\ell, \dots, g\} \subsetneqq \mathcal{M} \subseteq \{\ell, \dots, s-1\}\,.$$

Therefore, (9b) leads to

$$\sigma_g = \sum_{j=\ell}^{g} x_{\pi_{\mathrm{new}}(j)} < v \le \sum_{j=\ell}^{s-1} x_{\pi_{\mathrm{new}}(j)} \overset{(5b)}{=} \sigma_g + (s-g-1)x_{\pi_{\mathrm{old}}(p)}\,.$$

Consequently, Algorithm 11 terminates in step (v). □

Whenever an admissible call of Algorithm 11 terminates in step (v), a solution to the corresponding auxiliary subproblem is provided.

**Lemma 20.** *Let* ***QuickMark****$(x, \pi_{\mathrm{old}}, \ell, u, v)$ be an admissible call to Algorithm* 11*. If* ***QuickMark****$(x, \pi_{\mathrm{old}}, \ell, u, v)$ terminates in step* (v)*, then the output* $[\pi_{\mathrm{new}}, n]$ *guarantees that* $\mathcal{M} := \{\ell, \dots, n\} \in \mathfrak{M}(x, \pi_{\mathrm{new}}, \ell, u, v)$.

*Proof.* With $p, \pi_{\mathrm{new}}, g, s, \sigma_g$ from steps (i)–(iii), the termination in Algorithm 11(v) implies that

$$\sigma_g < v \le \sigma_g + (s-g-1)x_{\pi_{\mathrm{old}}(p)}\,.$$

Obviously, $x_{\pi_{\mathrm{old}}(p)} > 0$. Together with (5), this shows that $n := g + \lceil (v - \sigma_g)/x_{\pi_{\mathrm{old}}(p)} \rceil$ returned in Algorithm 11(v) satisfies that $g < n < s$. Again, (5) implies that $\mathcal{M} = \{\ell, \dots, n\}$ satisfies (9a). It remains to show (9b): The definition of $\sigma_g := \sum_{j=\ell}^{g} x_{\pi_{\mathrm{new}}(j)}$ and the choice of $n$ show that for all $k \in \mathcal{M}$ it holds

$$\sum_{j\in\mathcal{M}\setminus\{k\}} x_{\pi_{\mathrm{new}}(j)} \le -x_{\pi_{\mathrm{old}}(p)} + \sum_{j\in\mathcal{M}} x_{\pi_{\mathrm{new}}(j)} \overset{(5b)}{=} \sum_{j=\ell}^{n-1} x_{\pi_{\mathrm{new}}(j)} = \sigma_g + \sum_{j=g+1}^{n-1} x_{\pi_{\mathrm{new}}(j)}$$

$$\overset{(5b)}{=} \sigma_g + (n-g-1)\, x_{\pi_{\mathrm{old}}(p)} = \sigma_g + (\lceil (v-\sigma_g)/x_{\pi_{\mathrm{old}}(p)} \rceil - 1)\, x_{\pi_{\mathrm{old}}(p)} < \sigma_g + v - \sigma_g = v\,.$$

Similarly, we see that

$$\begin{aligned} v = v - \sigma_g + \sigma_g &\leq \lceil (v-\sigma_g)/x_{\pi_{\mathrm{old}}(p)} \rceil x_{\pi_{\mathrm{old}}(p)} + \sigma_g \\ &= (n-g) x_{\pi_{\mathrm{old}}(p)} + \sigma_g \overset{(5b)}{=} \sum_{j=g+1}^{n} x_{\pi_{\mathrm{new}}(j)} + \sigma_g = \sum_{j \in \mathcal{M}} x_{\pi_{\mathrm{new}}(j)} \,. \end{aligned}$$

Consequently, $\mathcal{M}$ satisfies (9b) and we conclude $\mathcal{M} := \{\ell, \dots, n\} \in \mathfrak{M}(x, \pi_{\mathrm{new}}, \ell, u, v)$. □

Algorithm 10 always terminates and provides a set of minimal cardinality satisfying the Dörfler marking criterion.

**Theorem 21.** *If initially called by Algorithm* 10*, then* ***QuickMark*** *terminates after finitely many operations and the output* $[\pi_{\mathrm{new}}, n]$ *guarantees that* $\pi_{\mathrm{new}}(\{1, \dots, n\})$ *satisfies the Dörfler criterion* (3) *with minimal cardinality.*

*Proof.* At latest the $(N-1)$st recursive call of ***QuickMark*** terminates in step (v) of Algorithm 11: Proposition 16 shows that all (subsequent) calls of ***QuickMark*** are admissible. For any recursive call ***QuickMark***$(x, \pi_{\mathrm{new}}, \ell', u', v')$ initiated by step (iv) or step (vi) of ***QuickMark***$(x, \pi_{\mathrm{old}}, \ell, u, v)$, it holds that $\{\ell', \dots, u'\} \subsetneqq \{\ell, \dots, u\}$. Therefore, if none of the first $N-2$ recursive calls of ***QuickMark*** terminates in step (v) of Algorithm 11, for the $(N-1)$st recursive call denoted by ***QuickMark***$(x, \bar\pi, \bar\ell, \bar u, \bar v)$ it holds that $\bar\ell = \bar u$. Consequently, for this call the pivot index is chosen as $\bar p = \bar\ell = \bar u$ in step (i) of Algorithm 11. Using Lemma 18, the admissibility of ***QuickMark***$(x, \bar\pi, \bar\ell, \bar u, \bar v)$ implies that $\mathfrak{M}(x, \bar\pi, \bar\ell, \bar u, \bar v) \neq \emptyset$. We infer that $\{\bar p\} \in \mathfrak{M}(x, \bar\pi, \bar\ell, \bar u, \bar v)$ and thus

$$x^*(x, \bar\pi, \bar\ell, \bar u, \bar v) \overset{(10)}{=} \min_{j \in \{\bar p\}} x_{\bar\pi(j)} = x_{\bar\pi(\bar p)} \,.$$

Hence, termination of ***QuickMark***$(x, \bar\pi, \bar\ell, \bar u, \bar v)$ in Algorithm 11(v) is implied by Lemma 19.

It remains to show that $\mathcal{M}' := \pi_{\mathrm{new}}(\{1, \dots, n\})$ satisfies (3) with minimal cardinality. By Remark 17, we will show $\mathcal{M} := \{1, \dots, n\} \in \mathfrak{M}(x, \pi_{\mathrm{new}}, 1, N, \theta \sum_{j=1}^{N} x_j)$: Suppose that $[\pi_{\mathrm{new}}, n]$ are obtained by Algorithm 10(iv). Denote the last recursive call of Algorithm 11 by ***QuickMark***$(x, \bar\pi_{\mathrm{old}}, \bar\ell, \bar u, \bar v)$. By Proposition 16, this call is admissible and $\pi_{\mathrm{new}} (= \bar\pi_{\mathrm{new}})$ differs from $\bar\pi_{\mathrm{old}}$ only for the indices $\{\bar\ell, \dots, \bar u\} \subseteq \{1, \dots, N\}$.

By Lemma 20, it holds that $\{\bar\ell, \dots, n\} \in \mathfrak{M}(x, \pi_{\mathrm{new}}, \bar\ell, \bar u, \bar v)$. Thus, the partial ordering (6a)–(6b) shows that

$$x_{\bar\pi_{\mathrm{new}}(j)} \geq x_{\bar\pi_{\mathrm{new}}(k)} \quad \text{for all } j \in \mathcal{M} \text{ and all } k \in \{1, \dots, N\} \setminus \mathcal{M} \,. \tag{11}$$

By Definition 15(b), it holds that

$$\bar v = \theta \sum_{j=1}^{N} x_j - \sum_{j=1}^{\bar\ell - 1} x_{\bar\pi_{\mathrm{old}}(j)} = \theta \sum_{j=1}^{N} x_j - \sum_{j=1}^{\bar\ell - 1} x_{\pi_{\mathrm{new}}(j)} \,.$$

Since $\{\bar\ell,\dots,n\}\in\mathfrak{M}(x,\pi_{\mathrm{new}},\bar\ell,\bar u,\bar v)$, condition (9b) reads

$$\sum_{j=\bar\ell}^{n} x_{\pi_{\mathrm{new}}(j)} \ge \bar v > -x_{\pi_{\mathrm{new}}(k)} + \sum_{j=\bar\ell}^{n} x_{\pi_{\mathrm{new}}(j)} \qquad \text{for all } \bar\ell\le k\le n\,.$$

Using the partial ordering (6a) and adding $\sum_{j=1}^{\bar\ell-1} x_{\pi_{\mathrm{new}}(j)}$ to the last estimate, we get

$$\sum_{j=1}^{n} x_{\pi_{\mathrm{new}}(j)} \ge \theta\sum_{j=1}^{N} x_j > -x_{\pi_{\mathrm{new}}(k)} + \sum_{j=1}^{n} x_{\pi_{\mathrm{new}}(j)} \qquad \text{for all } 1\le k\le N\,. \tag{12}$$

Consequently, (11)–(12) show that $\mathcal{M}\in\mathfrak{M}(x,\pi_{\mathrm{new}},1,N,\theta\sum_{j=1}^{N}x_j)$. □

3.3. **Computational complexity of the *QuickMark* algorithm.** Exploiting the fact that selection problems can always be solved in linear time [BPT+73], we show that the pivoting strategy in Algorithm 12 can be chosen such that, for any $x\in\mathbb{R}^N_\star$ and any $0<\theta<1$, Algorithm 10 always terminates after $\mathcal{O}(N)$ operations. Consider choosing the median of $\{x_{\pi(j)}\colon j=\ell,\dots,u\}$ as the pivot element.

**Algorithm 22** ($[p]=$ ***Median***$(x,\pi,\ell,u)$)**.** ***Input:*** *Vector $x\in\mathbb{R}^N$, permutation $\pi$ on $\{1,\dots,N\}$, lower and upper index $1\le\ell\le u\le N$.*

(i) *Determine an index $p\in\{\ell,\dots,u\}$ such that*

$$\#\{j\in\{\ell,\dots,u\}\colon x_{\pi(j)}<x_{\pi(p)}\}\le(u-\ell+1)/2\,, \tag{13a}$$

$$\#\{j\in\{\ell,\dots,u\}\colon x_{\pi(j)}>x_{\pi(p)}\}\le(u-\ell+1)/2\,. \tag{13b}$$

***Output:*** *Median index $p$.*

According to [BPT+73], Algorithm 22 can be implemented such that it always terminates in linear time $\mathcal{O}(u-\ell+1)$. This leads to the following theorem.

**Theorem 23.** *If **Pivot** is replaced by **Median** in Algorithm* 11(i)*, then, for any $x\in\mathbb{R}^N_\star$ and any $0<\theta<1$, Algorithm* 10 *terminates after $\mathcal{O}(N)$ operations. In particular, the multiplicative constant hidden in the Landau notation is generic and independent of $\theta$ and $N$.*

*Proof.* Obviously, steps (i)–(iii) of Algorithm 10 can be realized using $\mathcal{O}(N)$ operations. Moreover, the permutation $\pi$ can be represented by additionally storing an array containing $N$ indices. It remains to show that the call to ***QuickMark*** in step (iv) terminates at linear costs $\mathcal{O}(N)$.

Consider a (possibly recursive) call of ***QuickMark***$(x,\pi_{\mathrm{old}},\ell,u,v)$. The median (-index) of $x\circ\pi$ with respect to the indices $\{\ell,\dots,u\}$ of Algorithm 11(i) can be determined at linear cost $\mathcal{O}(u-\ell+1)$; see [BPT+73, Theorem 1]. The partition in Algorithm 11(ii) can be determined at linear cost $\mathcal{O}(u-\ell+1)$. In particular, this can easily be implemented by temporarily storing not more than $u-\ell+1$ additional indices $\pi_{\mathrm{mod}}$. Algorithm 11(iii) is of cost $g-\ell+1<u-\ell+1$ and steps (iv)–(vi) of Algorithm 11 are of constant cost $\mathcal{O}(1)$ plus, in the case of step (iv) or step (vi), the cost of the recursive call on at most $(u-\ell+1)/2$ indices; see (13). We have shown that for a generic constant $C\ge1$, the costs for an iteration of Algorithm 11 are bounded by $C(u-\ell+1)$ plus the costs of a potential recursive call.

Now, denote the computational costs of a call of ***QuickMark***$(x, \pi, \ell, u, v)$ by $T(m)$, where $m = \#\{\ell, \dots, u\} = u - \ell + 1$ is the number of elements under consideration. Then, due to the choice ***Pivot*** $:=$ ***Median***, using (13b) in Algorithm 11(iv), or (13a) in Algorithm 11(vi), respectively, it follows inductively that

$$T(N) \leq CN + T(N/2) \leq \cdots \leq CN \sum_{j=0}^{\infty} 2^{-j} = 2CN\,.$$

For the choice ***Pivot*** $:=$ ***Median***, we conclude that Algorithm 11, and hence Algorithm 10, always terminates at linear costs. □

*Remark* 24. (i) In the complexity estimate of Theorem 23 the dependency on $0 < \theta < 1$ is avoided due to the choice of ***Median*** as pivoting strategy. Other pivoting strategies may lead to a hidden constant depending on $0 < \theta < 1$.

(ii) If Algorithm 11(i) chooses the pivot index $p \in \{\ell, \dots, u\}$ always randomly, then the algorithm might perform faster on average. However, this would lead to quadratic worst-case performance $\mathcal{O}(N^2)$ of Algorithm 10.

(iii) Theorem 23 is proved for choosing the 50%-quantile, i.e., the median element is the pivot (Algorithm 22). If any other fixed quantile is chosen as the pivot, then Theorem 23 still holds true.

(iv) If for fixed $q \in (0, 1)$ one chooses pivoting by the $q$-quantile rather than by the median in Theorem 23, then a call of ***QuickMark***$(x, \pi_{\mathrm{old}}, \ell, u, v)$ potentially leads to a recursive call in step (iv) or step (vi) of Algorithm 11 on up to $\max\{q, 1-q\}(u - \ell + 1)$ indices. Hence, the computational costs of Algorithm 11 with this pivoting strategy called on $N$ indices can then be estimated by

$$T(N) \leq CN + T(\max\{q, 1-q\}N) \leq \ldots \leq CN \sum_{j=0}^{\infty} \max\{q, 1-q\}^j = \frac{CN}{\min\{q, 1-q\}}\,.$$

Obviously, choosing the median as pivot (i.e., $q = 1/2$) optimizes this estimate.

3.4. **Remarks on the implementation of *QuickMark*.** Up to now, we focused on the idea and the theoretical aspects of the ***QuickMark*** algorithm, namely verifying Theorem 9. We conclude this section by discussing some adaptions to the algorithm as it is presented in Section 3.1, in order to arrive at an efficient competitive `C++11` implementation using routines provided by the standard library. Ultimately, we compare the performance of our implementation to an implementation of Algorithm 2 based on the sorting routine provided by the standard library.

The following observations lead to an efficient ***QuickMark*** implementation relying on routines provided by the standard library.

*Remark* 25. (i) The data structure for given refinement indicators $\eta_\ell(T)$ for all $T \in \mathcal{T}_\ell$ is usually a vector `eta`, where `eta[j]` refers to the estimated error for the $j$th element in the data structure representing the mesh $\mathcal{T}_\ell$. To preserve this relation, one aims to avoid manipulating (i.e., reordering) `eta`.

(ii) ***QuickMark*** as formulated in Algorithm 11 avoids manipulation of `eta` by operating on a permutation $\pi$ only. Hence, in a straightforward implementation of Algorithm 11, which uses a permutation $\pi$ to access elements of the array $x \circ \pi$, data is not accessed contiguously and a considerable performance penalty is introduced.

(iii) Hence, to achieve a more efficient implementation of ***QuickMark***, one would rather alter the algorithm to operate on (and modify) a temporary copy `x` of `eta` to

determine the value $x^* := x^*(\texttt{eta}, \pi_{\mathrm{id}}, 1, N, \theta \sum_{j=1}^{N} x_j)$. The desired set $\mathcal{M}$ is then given by the union of $\{j\colon \texttt{eta[j]} > x^*\}$ and a proper subset of $\{j\colon \texttt{eta[j]} = x^*\}$.

(iv) For the ease of presentation, in ***Partition*** (Algorithm 13) a partition into three subarrays — elements strictly greater than, equal to, and strictly smaller than the pivot element — is demanded. In view of using standard library partition implementations, we note that this is not necessary: It suffices to partition into two subarrays: One with elements greater than or equal to the pivot element, the pivot element itself, and one with elements smaller than or equal to the pivot element. Then, as long as it is ensured, that other elements with the same value as the pivot element are distributed evenly among the two subarrays, Theorem 23 holds true.

(v) When using a partition based algorithm to determine a quantile, e.g., the median element, as the pivot element, the subarray is already partitioned after Algorithm 11(i). Hence, Algorithm 11(ii) can be skipped.

Using headers `<vector>`, `<iterator>`, `<algorithm>`, `<functional>`, and `<numeric>`, a `C++11` implementation of ***QuickMark*** adapted to the observations of Remark 25 relying on routines from the standard library could read as follows.

```
using Iterator_t = std::vector<double>::iterator;
const double xStarKernel
(Iterator_t subX_begin, Iterator_t subX_end, double goal)
{
  // QuickMark, step (i)-(ii): partition by median element
  auto length {std::distance(subX_begin, subX_end)};
  auto subX_middle {subX_begin + length/2};
  std::nth_element(subX_begin, subX_middle, subX_end, std::greater<double>());
  auto pivot_val {*subX_middle};

  // QuickMark, step (iii)
  auto sigma_g {std::accumulate(subX_begin, subX_middle, (double)0.0)};

  // QuickMark, step (iv), (v) and (vi)
  if (sigma_g >= goal)
    return xStarKernel(subX_begin, subX_middle, goal);
  if (sigma_g + pivot_val >= goal)
    return pivot_val;
  return xStarKernel(++subX_middle, subX_end, goal - sigma_g - pivot_val);
}
```

Passing refinement indicators $\eta_\ell(T)$ for all $T \in \mathcal{T}_\ell$ (`eta`) and an adaptivity parameter $0 < \theta < 1$ (`theta`) to the following adaption of Algorithm 10, then yields the desired value $x^*$, such that the set $\mathcal{M}$ is readily obtained; see Remark 25(iii).

```
const double compute_xStar (const std::vector<double>& eta, double theta)
{
  std::vector<double> x {eta};
  double goal {theta * std::accumulate(x.cbegin(), x.cend(), (double)0.0)};
  return xStarKernel(x.begin(), x.end(), goal);
}
```

*Remark* 26. While ***QuickMark*** can be implemented such that its complexity is linear even in the worst case, the worst-case complexity of the given `C++` function `xStarKernel` is (standard library-) implementation dependent:

The `C++` standard requires `std::nth_element` to be of linear complexity only on average, while lacking any worst-case restriction [ISO17]. A quality introspective selection implementation of `std::nth_element` could be realized as proposed in [Mus97]: As fast as the Quickselect algorithm [Hoa61] in practice, maintaining linear worst-case complexity by relying on the median of medians algorithm from [BPT+73] as fallback strategy.

We conclude by comparing the performance of the `C++` standard library implementation `std::sort` to our implementation `xStarKernel` above. This is reasonable, since those two routines are the core components of Algorithms 2 and 10 (adapted to the observations of Remark 25), respectively. The completing components of Algorithms 2 and 10 are very similar for both approaches and in particular, make up for only a small fraction of the overall computational cost of the respective algorithm.

We consider adaptivity parameters $\theta \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$ and vectors of length $N \in \{10^j : j = 3, \dots, 9\}$. For each combination of $\theta$ and $N$ we generate 30 vectors `eta` of length $N$ filled with uniformly distributed pseudorandom double-precision values between 0 and 1. The core routines `std::sort` and `xStarKernel` are called on (copies of) each of these vectors and the computational times are measured. The sources were compiled with GNU compiler `g++` version `5.5.0`, optimization flag `-O3`, and `-std=c++11` enabled. All computations were performed on a machine with 32 GB of RAM and an Intel Core i7-6700 CPU with a base frequency of 3.4 GHz.

For all test cases $(\theta, N) \in \{0.1, 0.25, 0.5, 0.75, 0.9\} \times \{10^j : j = 3, \dots, 9\}$, the measured times for the fastest (Table 1), average (Table 2), and slowest (Table 3) run out of 30 runs is given. To emphasize the improved complexity of Algorithm 10 over Algorithm 2, the measurements for $\theta = 0.5$ are visualized in Figure 3: While the computational time spent per element increases logarithmically with the problem size for `std::sort`, it remains constant for `xStarKernel`. Hence, as expected, the ***QuickMark*** strategy clearly outperforms the approach of Algorithm 2 based on sorting. Moreover, while the measured time behaves like $\mathcal{O}(N \log N)$ for sorting, it only grows linearly with respect to the problem size for ***QuickMark*** as predicted by Theorem 23. In accordance with Theorem 23, different values of $0 < \theta < 1$ do not influence the performance of the algorithm.

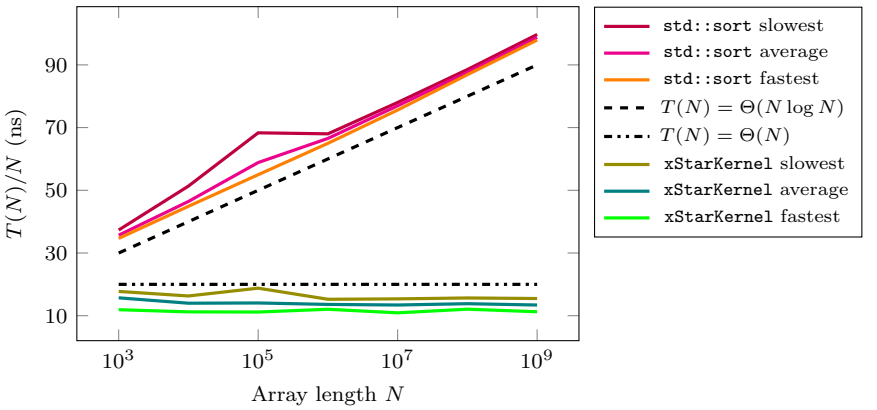


FIGURE 3. Visualization of the data from Tables 1–3 for $\theta = 0.5$. Performance comparison of `std::sort` versus `xStarKernel`: Computational time spent per element $T(N)/N$ in nanoseconds versus the array length $N$.

TABLE 1. Measured time (in seconds) for finding $x^*$ of a given double-precision vector of length $N$, versus the time it takes to sort it. Times for the fastest run out of 30 runs.

| $N$ | $\theta = 0.1$ | | $\theta = 0.25$ | | $\theta = 0.5$ | | $\theta = 0.75$ | | $\theta = 0.9$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | `sort` | `xStar` | `sort` | `xStar` | `sort` | `xStar` | `sort` | `xStar` | `sort` | `xStar` |
| $10^3$ | 3.4e−5 | 1.5e−5 | 3.5e−5 | 1.3e−5 | 3.5e−5 | 1.2e−5 | 3.4e−5 | 9.9e−6 | 3.4e−5 | 1.4e−5 |
| $10^4$ | 4.5e−4 | 1.3e−4 | 4.4e−4 | 1.2e−4 | 4.5e−4 | 1.1e−4 | 4.4e−4 | 8.5e−5 | 4.5e−4 | 1.1e−4 |
| $10^5$ | 5.5e−3 | 1.2e−3 | 5.4e−3 | 1.2e−3 | 5.5e−3 | 1.1e−3 | 5.5e−3 | 1.1e−3 | 5.5e−3 | 1.1e−3 |
| $10^6$ | 6.6e−2 | 1.2e−2 | 6.5e−2 | 1.1e−2 | 6.5e−2 | 1.2e−2 | 6.5e−2 | 1.1e−2 | 6.5e−2 | 1.1e−2 |
| $10^7$ | 7.6e−1 | 1.2e−1 | 7.6e−1 | 1.1e−1 | 7.6e−1 | 1.1e−1 | 7.6e−1 | 1.2e−1 | 7.6e−1 | 1.1e−1 |
| $10^8$ | 8.7e0 | 1.2e0 | 8.7e0 | 1.1e0 | 8.7e0 | 1.2e0 | 8.7e0 | 1.1e0 | 8.7e0 | 1.1e0 |
| $10^9$ | 9.8e+1 | 1.2e+1 | 9.8e+1 | 1.2e+1 | 9.8e+1 | 1.1e+1 | 9.8e+1 | 1.2e+1 | 9.8e+1 | 1.2e+1 |

TABLE 2. Measured time (in seconds) for finding $x^*$ of a given double-precision vector of length $N$, versus the time it takes to sort it. Average time for a run out of 30 runs.

| $N$ | $\theta = 0.1$ | | $\theta = 0.25$ | | $\theta = 0.5$ | | $\theta = 0.75$ | | $\theta = 0.9$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | `sort` | `xStar` | `sort` | `xStar` | `sort` | `xStar` | `sort` | `xStar` | `sort` | `xStar` |
| $10^3$ | 3.6e−5 | 1.7e−5 | 3.6e−5 | 1.6e−5 | 3.6e−5 | 1.6e−5 | 3.8e−5 | 1.6e−5 | 3.5e−5 | 1.5e−5 |
| $10^4$ | 4.6e−4 | 1.5e−4 | 4.6e−4 | 1.5e−4 | 4.6e−4 | 1.4e−4 | 4.7e−4 | 1.4e−4 | 4.8e−4 | 1.4e−4 |
| $10^5$ | 5.6e−3 | 1.4e−3 | 5.7e−3 | 1.4e−3 | 5.9e−3 | 1.4e−3 | 5.7e−3 | 1.4e−3 | 5.6e−3 | 1.3e−3 |
| $10^6$ | 6.7e−2 | 1.4e−2 | 6.7e−2 | 1.4e−2 | 6.7e−2 | 1.4e−2 | 6.6e−2 | 1.3e−2 | 6.6e−2 | 1.3e−2 |
| $10^7$ | 7.7e−1 | 1.4e−1 | 7.7e−1 | 1.4e−1 | 7.7e−1 | 1.3e−1 | 7.7e−1 | 1.4e−1 | 7.7e−1 | 1.4e−1 |
| $10^8$ | 8.8e0 | 1.4e0 | 8.8e0 | 1.4e0 | 8.8e0 | 1.4e0 | 8.8e0 | 1.4e0 | 8.8e0 | 1.3e0 |
| $10^9$ | 9.9e+1 | 1.4e+1 | 9.9e+1 | 1.3e+1 | 9.9e+1 | 1.3e+1 | 9.9e+1 | 1.3e+1 | 9.9e+1 | 1.3e+1 |

TABLE 3. Measured time (in seconds) for finding $x^*$ of a given double-precision vector of length $N$, versus the time it takes to sort it. Slowest time for a run out of 30 runs.

| $N$ | $\theta = 0.1$ | | $\theta = 0.25$ | | $\theta = 0.5$ | | $\theta = 0.75$ | | $\theta = 0.9$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | `sort` | `xStar` | `sort` | `xStar` | `sort` | `xStar` | `sort` | `xStar` | `sort` | `xStar` |
| $10^3$ | 6.0e−5 | 1.8e−5 | 3.7e−5 | 1.8e−5 | 3.7e−5 | 1.8e−5 | 6.7e−5 | 2.9e−5 | 3.7e−5 | 1.7e−5 |
| $10^4$ | 5.0e−4 | 1.8e−4 | 5.1e−4 | 1.9e−4 | 5.1e−4 | 1.6e−4 | 5.0e−4 | 1.7e−4 | 5.2e−4 | 1.7e−4 |
| $10^5$ | 6.2e−3 | 1.6e−3 | 6.1e−3 | 1.6e−3 | 6.8e−3 | 1.9e−3 | 6.1e−3 | 1.9e−3 | 5.7e−3 | 1.6e−3 |
| $10^6$ | 6.8e−2 | 1.6e−2 | 6.8e−2 | 1.6e−2 | 6.8e−2 | 1.5e−2 | 6.8e−2 | 1.5e−2 | 6.8e−2 | 1.6e−2 |
| $10^7$ | 7.9e−1 | 1.6e−1 | 7.8e−1 | 1.5e−1 | 7.8e−1 | 1.5e−1 | 7.8e−1 | 1.5e−1 | 8.1e−1 | 1.6e−1 |
| $10^8$ | 8.9e0 | 1.6e0 | 8.8e0 | 1.5e0 | 8.9e0 | 1.6e0 | 8.8e0 | 1.5e0 | 8.9e0 | 1.5e0 |
| $10^9$ | 1.0e+2 | 1.6e+1 | 1.0e+2 | 1.5e+1 | 1.0e+2 | 1.6e+1 | 1.0e+2 | 1.5e+1 | 1.0e+2 | 1.5e+1 |

## REFERENCES


[BDD04] P. Binev, W. Dahmen, and R. DeVore, *Adaptive finite element methods with convergence rates*, Numer. Math. **97** (2004), no. 2, 219–268. MR2050077

[BPT+73] M. Blum, V. Pratt, R. E. Tarjan, R. W. Floyd, and R. L. Rivest, *Time bounds for selection*, J. Comput. System Sci. **7** (1973), 448–461, DOI 10.1016/S0022-0000(73)80033-9. MR329916

[CFPP14] C. Carstensen, M. Feischl, M. Page, and D. Praetorius, *Axioms of adaptivity*, Comput. Math. Appl. **67** (2014), no. 6, 1195–1253, DOI 10.1016/j.camwa.2013.12.003. MR3170325

[CKNS08] J. M. Cascon, C. Kreuzer, R. H. Nochetto, and K. G. Siebert, *Quasi-optimal convergence rate for an adaptive finite element method*, SIAM J. Numer. Anal. **46** (2008), no. 5, 2524–2550, DOI 10.1137/07069047X. MR2421046

[CN12] J. M. Cascón and R. H. Nochetto, *Quasioptimal cardinality of AFEM driven by nonresidual estimators*, IMA J. Numer. Anal. **32** (2012), no. 1, 1–29, DOI 10.1093/imanum/drr014. MR2875241

[Dör96] W. Dörfler, *A convergent adaptive algorithm for Poisson's equation*, SIAM J. Numer. Anal. **33** (1996), no. 3, 1106–1124, DOI 10.1137/0733054. MR1393904

[FFP14] M. Feischl, T. Führer, and D. Praetorius, *Adaptive FEM with optimal convergence rates for a certain class of nonsymmetric and possibly nonlinear problems*, SIAM J. Numer. Anal. **52** (2014), no. 2, 601–625, DOI 10.1137/120897225. MR3176325

[FHPS19] T. Führer, A. Haberl, D. Praetorius, and S. Schimanko, *Adaptive BEM with inexact PCG solver yields almost optimal computational costs*, Numer. Math. **141** (2019), no. 4, 967–1008, DOI 10.1007/s00211-018-1011-1. MR3923519

[GHPS18] G. Gantner, A. Haberl, D. Praetorius, and B. Stiftner, *Rate optimal adaptive FEM with inexact solver for nonlinear operators*, IMA J. Numer. Anal. **38** (2018), no. 4, 1797–1831, DOI 10.1093/imanum/drx050. MR3867383

[Hoa61] C. A. R. Hoare, *Algorithm 65: Find*, Commun. ACM **4** (1961), no. 7, 321–322.

[ISO17] ISO, *ISO/IEC 14882:2017 Information technology — Programming languages — C++*, Fifth edition, ISO, 2017.

[MN05] K. Mekchay and R. H. Nochetto, *Convergence of adaptive finite element methods for general second order linear elliptic PDEs*, SIAM J. Numer. Anal. **43** (2005), no. 5, 1803–1827, DOI 10.1137/04060929X. MR2192319

[MNS00] P. Morin, R. H. Nochetto, and K. G. Siebert, *Data oscillation and convergence of adaptive FEM*, SIAM J. Numer. Anal. **38** (2000), no. 2, 466–488, DOI 10.1137/S0036142999360044. MR1770058

[MSV08] P. Morin, K. G. Siebert, and A. Veeser, *A basic convergence result for conforming adaptive finite elements*, Math. Models Methods Appl. Sci. **18** (2008), no. 5, 707–737, DOI 10.1142/S0218202508002838. MR2413035

[Mus97] D. R. Musser, *Introspective sorting and selection algorithms*, Software: Practice and Experience **27** (1997), no. 8, 983–993.

[PP19] C.-M. Pfeiler and D. Praetorius, *Dörfler marking with minimal cardinality is a linear complexity problem*, arXiv:1907.13078 (2019).

[Sie11] K. G. Siebert, *A convergence proof for adaptive finite elements without lower bound*, IMA J. Numer. Anal. **31** (2011), no. 3, 947–970, DOI 10.1093/imanum/drq001. MR2832786

[Ste07] R. Stevenson, *Optimality of a standard adaptive finite element method*, Found. Comput. Math. **7** (2007), no. 2, 245–269, DOI 10.1007/s10208-005-0183-0. MR2324418

TU Wien, Institute of Analysis and Scientific Computing, Wiedner Hauptstr. 8-10/E101/4, 1040 Vienna, Austria

*Email address*: carl-martin.pfeiler@asc.tuwien.ac.at

TU Wien, Institute of Analysis and Scientific Computing, Wiedner Hauptstr. 8-10/E101/4, 1040 Vienna, Austria

*Email address*: dirk.praetorius@asc.tuwien.ac.at