A flexible block coordinate descent method for unconstrained optimization under Hölder continuity*

V. S. Amaral† R. Andreani‡ L. D. Secchin§ G. N. Silva†

April 3, 2026

Abstract

In this work, we propose a flexible block coordinate method for unconstrained optimization problems under Hölder continuity assumptions. The method guarantees convergence to stationary points and has worst-case complexity results comparable to those obtained by single-block methods that assume Lipschitz or Hölder continuity. The approach is based on quadratic models of the objective function combined with quadratic regularization. Flexible block selection strategies are allowed, and different sufficient descent conditions previously considered in the literature are unified within the same framework. The proposed method is fully implementable and incorporates a practical certificate of stationarity as a stopping criterion. Well-definiteness and convergence are established under the Hölder continuity of the objective function gradient, extending classical results that typically rely on Lipschitz assumptions. Illustrative numerical tests are performed and a freely available Julia implementation is provided.

Keywords: Block coordinate descent methods, quadratic regularization, Hölder continuity, complexity

1 Introduction

In this paper, we consider the unconstrained optimization problem

$$\min f(x) \quad \text{s.t.} \quad x \in \mathbb{R}^n \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a (possibly non-convex) continuously differentiable function. Many problems of the type of (1) arising from applications are large scale, and then their resolution requires low-cost practical methods. Depending on the structure of the problem, block coordinate descent (BCD) methods are suitable. In fact, in these methods only a few variables are used in an iteration, reducing the cost of computing function values and gradients, at the same time that guides the minimization process efficiently. Problems with a favorable structure to the application of BCD methods can be found in computational statistics [14, 15], machine learning [6], among others.

For a long time, BCD methods did not receive much attention from the researchers in optimization. However, in the last decades this scenario has changed, as it has been demonstrated that BCD methods are useful for solving problems with a large number of variables and where the required accuracy is moderate. Thus, nowadays many works devoted to BCD methods have emerged, see for instance [1, 2, 3, 4, 8, 13].

*This work has been partially supported by CEPID-CeMEAI (FAPESP 2013/07375-0), FAPESP (grant 2023/08706-1), and CNPq (grants 407147/2023-3, 306988/2021-6, 302520/2025-2, 401864/2022-7, and 306593/2022-0).

†Department of Mathematics, Federal University of Piauí, PI, Brazil. vitalianoamaral@ufpi.edu.br, gilson.silva@ufpi.edu.br

‡Department of Applied Mathematics, Universidade Estadual de Campinas, Campinas, SP, Brazil. andreani@ime.unicamp.br

§Department of Applied Mathematics, Federal University of Espírito Santo, ES, Brazil. leonardo.secchin@ufes.br

In [13], the convergence of different BCD methods are treated, specially for minimizing convex functions. Special attention is devoted to problems arising from machine learning applications, where randomly selection of a block of variables at each iteration is suitable. The author discuss in detail efficient parallel implementations for this case, showing the effectiveness of using BCD methods to solve such problems. Also, similar convergence properties to the deterministic cyclic BCD variant [3] are obtained.

In [2], BCD methods that employ high-order regularizations to minimize smooth possibly non-convex functions subject to box-constraints were introduced. Complexity results were established; the authors proved that the BCD method with a $p+1$-order regularization (which means that a term $\|x\|^{p+1}$ is added to the objective function of the subproblem) spends no more than $\mathcal{O}(\epsilon^{-(p+1)})$ outer iterations to reach an ϵ-stationary point in the sense that the 2-norm of the gradient of f is smaller than ϵ. In [4], BCD methods with high-order regularizations were extended to the case where general constraints for each block are present. The authors proved that an "ϵ-approximate" solution in a suitable sense can be reached in at most $\mathcal{O}(\epsilon^{-2})$ outer iterations. In [1], classical derivative-free methods were extended to BCD methods. It was proved that an ϵ-approximate solution can be achieved in at most $\mathcal{O}(\epsilon^{-\frac{\beta+1}{\beta}})$ outer iterations.

In this work, we investigate a BCD method for solving (1) that employs quadratic regularization. Unlike previous works that impose Lipschitz assumptions on the gradients [2, 4, 8, 13], our convergence/complexity analysis is developed under Hölder conditions. We consider two types of line searches: one based on the magnitude of the descent direction and another based on the optimality tolerance. These line searches were considered separately in the literature. Instead, in our method they can be combined in a variety of ways, increasing their adaptability to solve specific problems.

We prove that it reaches an ϵ-stationary point in at most $\mathcal{O}(\epsilon^{-\frac{2}{\beta}})$ outer iterations, where β is the Hölder constant. This encompasses previous results for the Lipschitz case ($\beta = 1$) [2, 4]. If only the line search based on the optimality tolerance is used, the slight better complexity $\mathcal{O}(\epsilon^{-\frac{\beta+1}{\beta}})$ is achieved, recovering the complexity of the "1-block" method with quadratic regularization under Hölder assumptions discussed in [9]. This is the same complexity for the BCD method stated in [1], which works under Hölder assumptions but is designed only for cyclic block selection. In contrast, our method allows for flexible the block selection rules, including, in particular, cyclic and random selections. Furthermore, the constant associated with the complexity in [1] depends on the number of blocks and variables, whereas this dependence does not exist in our results.

1.1 Summary of our Contributions

The main goal is to propose a flexible block coordinate framework for unconstrained optimization supposing only the Hölder continuity of the objective function gradient, with the ability to attest convergence to stationary points, and having strong complexity results compatible with other "1-block" methods that require Lipschitz continuity. Specifically, our work have the following features:

- the proposed method is implementable and the block selection rule is flexible. Additionally, they incorporate a legitimate certificate of stationarity as a stopping criterion, an unusual feature in previous block descent methods from the literature;
- well-definiteness and convergence are established under Hölder continuity, in contrast to the Lipschitz continuity assumptions in previous works;
- we present a method that uses quadratic approximations of the objective function plus a quadratic regularization. Complexity results comparable to other related methods that require Lipschitz continuity are provided;
- the sufficient descent criterion is flexible as it combines two types of decrease: one relative to the optimality tolerance and other relative to the magnitude of the descent direction. Both have been considered separately in the literature. Our theory unifies them, allowing for various combinations;

- we provide a freely available implementation in Julia. The code allows blocks to be defined arbitrarily, different block selection rules, and user-defined descent criteria.

1.2 Notation and organization of the paper

Throughout the text, $\langle\cdot,\cdot\rangle$, $\|\cdot\|$, $\|\cdot\|_\infty$, and $\|\cdot\|_3$ denote the usual inner product, the Euclidean norm, the sup-norm, and the 3-norm in $\mathbb{R}^n$, respectively. The dimension of the i-th block of variables is denoted by n_i, $i = 1,\ldots,q$, so $n_1+n_2+\cdots+n_q = n$. The matrix $U_i \in \mathbb{R}^{n\times n_i}$ is used to describe the vector $v_i = U_i^T v \in \mathbb{R}^{n_i}$ composed by the components of $v \in \mathbb{R}^n$ within block i (note that the order of components in the i-th block is defined by U_i). In particular, U_i allows us to write the *partial gradient* $\nabla_{(i)}g(x) = U_i^T\nabla g(x)$ from the total gradient $\nabla g(x)$ of a function $g:\mathbb{R}^n\to\mathbb{R}$. Similarly, the *partial Hessian* of g at x is the $n_i\times n_i$ matrix $\nabla^2_{(i)}g(x) = U_i^T\nabla^2 g(x)U_i$.

In Section 2, we present the Block Coordinate Descent (BCD) method with quadratic regularization for solving (1), as well as its global and local convergence analysis. In the method, we aim to approximate a first-order stationary point x^* for (1) ($\nabla f(x^*) = 0$) or found an acceptable solution. A practical implementation and numerical tests are presented in Section 3. Finally, Section 4 brings our conclusions and future work.

2 BCD with quadratic regularization

The proposed BCD method with quadratic regularization (BCDQR) is presented in Algorithm 1. It is similar to that presented in [4], except for the stopping criterion in Step 2. The control on the choice of indices i performed in Steps 1 to 3 aims to guarantee that every block of variables is checked before stopping declaring success. In fact, the stopping criterion in Step 2 only takes place if $\|\nabla_{(i)}f(x^k)\|_\infty \le \epsilon$ for every $i\in\{1,\ldots,q\}$, which imply $\|\nabla f(x^k)\|_\infty \le \epsilon$. In this sense, the first-order stationary condition is approximately fulfilled. Notice that in [4], an exogenous assumption is imposed to ensure that every block is visited infinitely many times. Instead, we handle this directly in the algorithm via the set C, but with a high level of freedom in choosing index i in Step 2.

Algorithm 1 Block Coordinate Descent with Quadratic Regularization (BCDQR)

Let $x^0\in\mathbb{R}^n$, $\alpha\in(0,1)$, $\theta>0$, $\epsilon\in(0,1)$, $\sigma_0\ge 1$ and $f_{\text{est}}\in[-\infty,\infty)$. Initialize $k\leftarrow 0$.

Step 1: If $f(x^k)\le f_{\text{est}}$, stop declaring x^k an acceptable solution. Otherwise, take $C\leftarrow\{1,\ldots,q\}$.

Step 2: If $C=\emptyset$, stop declaring x^k an approximate stationary point. Otherwise, choose $i\in C$.

Step 3: If $\|\nabla_{(i)}f(x^k)\|_\infty\le\epsilon$, update $C\leftarrow C\backslash\{i\}$ and go to Step 2.

Step 4: Compute $x^{\text{trial}} = x^k + U_i s^k_{(i)}$ where $s^k_{(i)}\in\mathbb{R}^{n_i}$ is such that

$$\langle\nabla_{(i)}f(x^k), s^k_{(i)}\rangle + \frac{1}{2}\langle B_{(i)}(x^k)s^k_{(i)}, s^k_{(i)}\rangle + \sigma_k\|s^k_{(i)}\|^2 \le 0, \tag{2}$$

$$\|\nabla_{(i)}f(x^k) + B_{(i)}(x^k)s^k_{(i)} + 2\sigma_k s^k_{(i)}\| \le \theta\|s^k_{(i)}\| \tag{3}$$

and $B_{(i)}(x^k)\in\mathbb{R}^{n_i\times n_i}$ is symmetric.

Step 5: If

$$f(x^{\text{trial}}) \le f(x^k) - \min\left\{\frac{\alpha}{16\sigma_k}\epsilon^2\ ,\ \alpha\|s^k_{(i)}\|^2\right\}$$

holds, define $x^{k+1} = x^{\text{trial}}$ and $\sigma_{k+1}=\sigma_k$, take $k\leftarrow k+1$ and go to Step 1. Otherwise, define $\sigma_{k+1}=2\sigma_k$ and go to Step 4.

We now make some comments regarding the BCDQR method. First, the parameter f_{est} is an acceptable threshold for the value of f, which says that any x such that $f(x) \leq f_{\text{est}}$ is considered an acceptable solution for (1) (Step 1). Of course, this parameter depends on the problem and the application considered, and appears in the literature, see e.g. [9]. Taking $f_{\text{est}} = -\infty$ forces the algorithm to compute a genuine approximate stationary point (Step 2). Second, the parameter α controls the required level of decrease of f in Step 5; note that, as α approaches 1, the decrease becomes more aggressive, possibly causing the penalty parameter σ to grow unnecessarily. So, α should be tuned carefully in practice. Third, the descent test employed in Step 5 covers two types of criteria: one based on the penalty parameter σ_k and the tolerance ϵ, similar to what was done in [9], and another based on the magnitude of the direction computed in Step 4, similar to [4]. Finally, the trial point x^{trial} is accepted if at least one of these criteria is fulfilled, mitigating the growth of σ and bringing more numerical stability to the algorithm. Also, small σ's tend to generate larger steps, which hopefully makes the method converge faster.

In the sequel, we analyze the well-definiteness of Algorithm 1. The next lemma, whose proof is trivial, shows that Step 4 is well-defined.

Lemma 1. *Let $\varphi : \mathbb{R}^{n_i} \to \mathbb{R}$ be defined as*

$$\varphi(z) = \langle \nabla_{(i)} f(x^k), z \rangle + \frac{1}{2} \langle B_{(i)}(x^k) z, z \rangle + \sigma_k \|z\|^2. \tag{4}$$

There is an approximate minimizer $s_{(i)}$ of φ in the sense of $\varphi(s_{(i)}) \leq \varphi(0)$ and $\|\nabla \varphi(s_{(i)})\| \leq \theta \|s_{(i)}\|$. Also, any of such $s_{(i)}$ fulfills conditions (2) and (3).

Lemma 1 gives a practical way to compute $s^k_{(i)}$ in Step 4 of Algorithm 1, which may not be unique. Therefore, we conduct our theoretical analysis considering the general conditions (2) and (3), which allow the use of different strategies to compute $s^k_{(i)}$.

For the remaining analysis, we impose the Hölder-type continuity hypothesis below. Notice that in previous works, Lipschitz continuity on the gradient of the objective function is necessary, see e.g. [3, 4, 13].

Assumption 1. *For each $i \in \{1, \ldots, q\}$, there are $L_i > 0$ and $\beta_i \in (0, 1]$ such that*

$$f(x + U_i s_{(i)}) \leq f(x) + \langle \nabla_{(i)} f(x), s_{(i)} \rangle + L_i \|s_{(i)}\|^{\beta_i + 1}$$

holds for all $x \in \mathbb{R}^n$ and $s_{(i)} \in \mathbb{R}^{n_i}$.

Assumption 1 holds if the partial gradients of f satisfy the Hölder condition with constants β_i and $(\beta_i + 1)L_i$ for each block, that is, if for every $i \in \{1, \ldots, q\}$,

$$\|\nabla_{(i)} f(x + U_i s_{(i)}) - \nabla_{(i)} f(x)\| \leq (\beta_i + 1) L_i \|s_{(i)}\|^{\beta_i}, \quad \forall x \in \mathbb{R}^n, \ \forall s_{(i)} \in \mathbb{R}^{n_i}.$$

See e.g. [16, Lemma 1].

In Algorithm 1, the matrix $B_{(i)}(x^k)$ does not need to be an approximation of the partial Hessian $\nabla^2_{(i)} f(x^k)$. This matrix can play different roles: in situations where gradients alone are sufficient, one may set it as the zero matrix; in other cases, it is possible to use second-order approximations, either the exact partial Hessian or simplified versions obtained, for instance, through quasi-Newton updates. This flexibility of choice, which may even vary from block to block, is what provides adaptability to the scheme. Thus, the formulation of the subproblem can rely solely on first-order information or, alternatively, on richer descriptions that incorporate second-order data. The only requirement is that $B_{(i)}(x^k)$ remains uniformly bounded, as stated in Assumption 2. This is consistent with the block descent algorithm considered in [4].

Assumption 2. *There is $\overline{B} \geq 1$ such that $\|B_{(i)}(x^k)\| \leq \overline{B}$ for all $i \in \{1, \ldots, q\}$ and k.*

The next two lemmas are useful to prove that the sufficient descent condition in Step 5 of Algorithm 1 holds for all σ_k large enough, which is stated in Theorem 4.

Lemma 2. *If Assumptions 1 and 2 hold and (2) is valid, then*

$$f(x^{\text{trial}}) \leq f(x^k) - \sigma_k \|s_{(i)}^k\|^2 + \frac{\overline{B}}{2}\|s_{(i)}^k\|^2 + L_i\|s_{(i)}^k\|^{\beta_i+1}.$$

Proof. Using Assumption 1 and the expression for x^{trial} given in Step 4, we have

$$\begin{aligned} f(x^{\text{trial}}) &\leq f(x^k) + \langle \nabla_{(i)} f(x^k), s_{(i)}^k\rangle + L_i\|s_{(i)}^k\|^{\beta_i+1} \\ &\leq f(x^k) - \sigma_k\|s_{(i)}^k\|^2 - \frac{1}{2}\langle B_{(i)}(x^k)s_{(i)}^k, s_{(i)}^k\rangle + L_i\|s_{(i)}^k\|^{\beta_i+1} \\ &+ \left[\langle \nabla_{(i)} f(x^k), s_{(i)}^k\rangle + \frac{1}{2}\langle B_{(i)}(x^k)s_{(i)}^k, s_{(i)}^k\rangle + \sigma_k\|s_{(i)}^k\|^2\right]. \end{aligned}$$

Now, using (2) and Assumption 2 we arrive at

$$\begin{aligned} f(x^{\text{trial}}) &\leq f(x^k) - \sigma_k\|s_{(i)}^k\|^2 - \frac{1}{2}\langle B_{(i)}(x^k)s_{(i)}^k, s_{(i)}^k\rangle + L_i\|s_{(i)}^k\|^{\beta_i+1} \\ &\leq f(x^k) - \sigma_k\|s_{(i)}^k\|^2 + \frac{1}{2}\|B_{(i)}(x^k)\|\|s_{(i)}^k\|^2 + L_i\|s_{(i)}^k\|^{\beta_i+1} \\ &\leq f(x^k) - \sigma_k\|s_{(i)}^k\|^2 + \frac{\overline{B}}{2}\|s_{(i)}^k\|^2 + L_i\|s_{(i)}^k\|^{\beta_i+1}, \end{aligned}$$

as we want to prove. □

Lemma 3. *Suppose that Assumption 2 holds. If* $\|\nabla_{(i)} f(x^k)\|_\infty > \epsilon$ *and* $\sigma_k \geq \dfrac{\overline{B}+\theta}{2}$ *then* $\sigma_k\|s_{(i)}^k\| \geq \epsilon/4$.

Proof. By (3) and Assumption 2 we have

$$\|\nabla_{(i)} f(x^k)\|_\infty \leq \|\nabla_{(i)} f(x^k)\| \leq \|B_{(i)}(x^k)s_{(i)}^k + 2\sigma_k s_{(i)}^k\| + \theta\|s_{(i)}^k\| \leq (\overline{B} + 2\sigma_k + \theta)\|s_{(i)}^k\|.$$

The statement follows by applying the hypotheses to the inequality above. □

As we already mentioned, the descent condition in Step 5 of Algorithm 1 is satisfied for σ_k large enough. To this end, we define, for each $i = 1, \ldots, q$,

$$\underline{\sigma}_i(t) = \max\left\{\frac{\overline{B}+\theta}{2}, \left[\frac{\overline{B}}{2(1-\alpha)} + \frac{2L_i}{(1-\alpha)}\left(\frac{t}{4}\right)^{\beta_i-1}\right]^{\frac{1}{\beta_i}}\right\}. \tag{5}$$

By straightforward calculations it is possible to see that

$$\underline{\sigma}_i(1) \leq \underline{\sigma}_i(\epsilon) \leq \epsilon^{(\beta_i-1)/\beta_i}\underline{\sigma}_i(1).$$

The next result shows that $\sigma_k \geq \underline{\sigma}_i(\epsilon)$ is sufficient to guarantee the well-definiteness of Step 5. Note that Steps 4 and 5 of Algorithm 1 only refer to a previously chosen block index i, so σ_k needs to be increased with respect to only one block per iteration to achieve the sufficient decrease in f.

Theorem 4. *Suppose that Assumptions 1 and 2 hold and, at iteration k of Algorithm 1, we have chosen i in Step 2. Also, suppose that $\sigma_k \geq \underline{\sigma}_i(\epsilon)$. If $x^{\text{trial}} \in \mathbb{R}^n$ is the trial point computed in Step 5 then*

$$f(x^{\text{trial}}) \leq f(x^k) - \alpha\sigma_k\|s_{(i)}^k\|^2 \tag{6}$$

and

$$f(x^{\text{trial}}) \leq f(x^k) - \frac{\alpha}{16\sigma_k}\epsilon^2. \tag{7}$$

In particular, x^{trial} fulfills the sufficient descent condition in Step 5.

Proof. First note that Step 4 of Algorithm 1 only takes place if $\|\nabla_{(i)} f(x^k)\|_\infty > \epsilon$. So, Lemma 3 ensures that

$$\|s_{(i)}^k\| \geq \frac{\epsilon}{4\sigma_k}. \tag{8}$$

In view of Lemma 2, to prove (6) it is enough that

$$-\alpha\sigma_k\|s_{(i)}^k\|^2 \geq -\sigma_k\|s_{(i)}^k\|^2 + \frac{\overline{B}}{2}\|s_{(i)}^k\|^2 + L_i\|s_{(i)}^k\|^{\beta_i+1},$$

which, after straightforward calculations, is equivalent to

$$\sigma_k \geq \frac{\overline{B}}{2(1-\alpha)} + \frac{2L_i}{(1-\alpha)}\|s_{(i)}^k\|^{\beta_i-1}.$$

The inequality above follows from (8), $\sigma_k \geq 1$, (5), and $\beta_i \in (0,1]$, noting that

$$\begin{aligned}
\frac{\overline{B}}{2(1-\alpha)} + \frac{2L_i}{(1-\alpha)}\|s_{(i)}^k\|^{\beta_i-1} &\leq \frac{\overline{B}}{2(1-\alpha)} + \frac{2L_i}{(1-\alpha)}\left(\frac{\epsilon}{4\sigma_k}\right)^{\beta_i-1} \\
&\leq \left[\frac{\overline{B}}{2(1-\alpha)} + \frac{2L_i}{(1-\alpha)}\left(\frac{\epsilon}{4}\right)^{\beta_i-1}\right]\sigma_k^{1-\beta_i} \\
&\leq \underline{\sigma}_i(\epsilon)^{\beta_i}\sigma_k^{1-\beta_i} \leq \sigma_k^{\beta_i}\sigma_k^{1-\beta_i} = \sigma_k.
\end{aligned}$$

Thus, (6) holds. Finally, using (8) in (6) we obtain (7). □

2.1 Convergence and complexity analysis

Up to this point, we have established the well-definiteness of Algorithm 1. In this section, we address its complexity. We are interested in establishing upper bounds on the number of iterations and the number of evaluations of f required to achieve a prefixed optimality tolerance. The following additional assumption is necessary in this context, which is automatically satisfied if f is bounded below:

Assumption 3. *The sequence $\{f(x^k)\}$ is bounded below, let us say, $f(x^k) \geq \underline{f}$ for all k.*

Theorem 4 shows that the reduction in f required in Step 5 is achieved for every σ_k large enough. Next we prove that, indeed, f reduces by a constant factor only depending on the constants from the method and Assumptions 1 and 2.

Lemma 5. *Suppose that Assumptions 1 and 2 hold. Then, regarding Algorithm 1, we have*

$$f(x^{k+1}) \leq f(x^k) - \frac{\alpha}{\eta}\epsilon^{\frac{2}{\beta}},$$

where $\beta = \min\{\beta_i \mid i = 1,\ldots,q\} \in (0,1]$,

$$\eta = 64\max\{\underline{\sigma}_i^2(1) \mid i = 1,\ldots,q\}$$

(that does not depend on ϵ), and $\underline{\sigma}_i(1)$ given by (5).

Proof. Let us define

$$\sigma_{\max} = \epsilon^{\frac{\beta-1}{\beta}}\max\{\underline{\sigma}_i(1) \mid i = 1,\ldots,q\}.$$

As $0 < \epsilon < 1$ and $0 < \beta \leq \beta_i \leq 1$ for all i, we have $\epsilon^{\frac{\beta-1}{\beta}} \geq \epsilon^{\frac{\beta_i-1}{\beta_i}} \geq 1$ for all $i = 1,\ldots,q$, and thus

$$\sigma_{\max} \geq \max\{\underline{\sigma}_i(1)\epsilon^{\frac{\beta_i-1}{\beta_i}} \mid i = 1,\ldots,q\} \geq \max\{\underline{\sigma}_i(\epsilon) \mid i = 1,\ldots,q\}.$$

If $\sigma_k \geq \sigma_{\max}$ then $\{\sigma_k\}$ remains constant after a certain iteration by Theorem 4. Therefore, due to the way σ_k increases in Step 5, we have

$$\sigma_k \leq 2\sigma_{\max}. \tag{9}$$

Steps 4 and 5 only takes place if $\|\nabla_{(i)} f(x^k)\|_\infty > \epsilon$. In this case, as in the proof of Lemma 3, we have

$$\epsilon < \|\nabla_{(i)} f(x^k)\|_\infty \leq (\overline{B} + 2\sigma_k + \theta)\|s^k_{(i)}\| \leq 8\sigma_{\max}\|s^k_{(i)}\|. \tag{10}$$

By Step 5, x^{k+1} satisfies

$$f(x^{k+1}) \leq f(x^k) - \alpha\|s^k_{(i)}\|^2 \quad \text{or} \quad f(x^{k+1}) \leq f(x^k) - \frac{\alpha}{16\sigma_k}\epsilon^2.$$

In the first case, using (9), (10), and the definitions of η and $\sigma_{\max}$ we obtain

$$f(x^{k+1}) \leq f(x^k) - \alpha\frac{\epsilon^2}{64\sigma^2_{\max}} = f(x^k) - \alpha\frac{\epsilon^2}{\eta\epsilon^{2(\beta-1)/\beta}} = f(x^k) - \frac{\alpha}{\eta}\epsilon^{\frac{2}{\beta}}.$$

In the second case, using (9) and $\sigma_{\max} \geq 1$ we arrive at

$$f(x^{k+1}) \leq f(x^k) - \frac{\alpha}{32\sigma_{\max}}\epsilon^2 \leq f(x^k) - \alpha\frac{\epsilon^2}{64\sigma^2_{\max}},$$

which falls in the first case. This completes the proof. □

Considering Lemma 5, as β approaches zero, the reduction in f diminishes. This is coherent with the theory: small β leads to a less stringent Hölder continuity, therefore the expected local reduction along descent directions $s_{(i)}$ is smaller.

Next we prove one of the main results of this section, namely, that the number of (outer) iterations in Algorithm 1 required to achieve a prescribed tolerance ϵ is bounded by a multiple of $\epsilon^{-2/\beta}$. In this result, we discard inner iterations, that is, the number of trial points rejected in Step 5. Note, however, that the trial point is never rejected when σ_k is large enough by Theorem 4.

Theorem 6. *Suppose that Assumptions 1, 2, and 3 hold, and let β and η as in Lemma 5. Then, after at most*

$$\left\lceil \frac{\max\{0, f(x^0) - \max\{\underline{f}, f_{\text{est}}\}\}\eta}{\alpha} \right\rceil \epsilon^{-\frac{2}{\beta}}$$

outer iterations, Algorithm 1 stops.

Proof. If $f_{\text{est}} \geq f(x^0)$, the algorithm stops at the first iteration, and so there is nothing to prove. Suppose then that $f_{\text{est}} < f(x^0)$. In this case, we can neglect f_{est} since it can be interpreted as a lower bound for f, so it is sufficient to consider only $\underline{f}$.

Independently of the number of trial points rejected, by applying Lemma 5 recursively we obtain

$$\underline{f} < f(x^{k+1}) \leq f(x^k) - \frac{\alpha}{\eta}\epsilon^{\frac{2}{\beta}} \leq f(x^{k-1}) - 2\frac{\alpha}{\eta}\epsilon^{\frac{2}{\beta}} \leq \cdots \leq f(x^0) - (k+1)\frac{\alpha}{\eta}\epsilon^{\frac{2}{\beta}},$$

from which follows that

$$k + 1 < \left\lceil \frac{(f(x^0) - \underline{f})\eta}{\alpha} \right\rceil \epsilon^{-\frac{2}{\beta}}.$$

This concludes the proof. □

In Algorithm 1, partial gradients and Hessians approximations are evaluated one time per iteration. On the other hand, the number of evaluations of function f depends on the times that σ is increased at Step 5. In Theorem 4, we prove that this number is finite if σ is large enough. Next, we give a upper bound on the number of evaluations of f sufficient to achieve a predefined optimality tolerance.

Theorem 7. *Suppose that Assumptions 1, 2, and 3 hold, and let β and η as in Lemma 5. The number of evaluations of f employed in Algorithm 1 is bounded above by*

$$\left[\frac{\max\left\{0, f(x^0) - \max\{\underline{f}, f_{\text{est}}\}\right\}\eta}{\alpha}\right]\epsilon^{-\frac{2}{\beta}} + \log_2\left(\frac{2\sigma_{\max}}{\sigma_0}\right),$$

where $\sigma_{\max} = \epsilon^{\frac{\beta-1}{\beta}} \max\{\underline{\sigma}_i(1) \mid i = 1, \ldots, q\}$.

Proof. The number of evaluations of f is bounded by the maximum number of outer iterations, which is given in Theorem 6, plus an upper bound on the number of times that σ is increased in Step 5. So, let r the number of times σ increases throughout the entire minimization process. By Theorem 4, σ does not increase if $\sigma_k \geq \sigma_{\max}$, and by Step 5 we have

$$2^r \sigma_0 \leq 2\sigma_{\max}.$$

This implies

$$r = \log_2(2^r) \leq \log_2\left(\frac{2\sigma_{\max}}{\sigma_0}\right),$$

from which follows the statement. □

It is instructive to compare the complexities in the theorems above with other works from literature where a Lipschitz continuity on the gradients is imposed, which is equivalent to choose $\beta_i = 1$ for all i in Assumption 1. In this case, Theorem 6 gives the complexity $\mathcal{O}(\epsilon^{-2})$ for the number of outer iterations in Algorithm 1. This is the same complexity for the block descent method presented in [4] (although this method is designed also for constrained problems), see [4, Theorem 4.2]. In summary, Theorems 6 and 7 encompass such previous result from the literature, with the difference that in Algorithm 1 two types of descent conditions (Step 5) are covered.

There are only a few works in the literature that establish convergence rates of optimization methods assuming Hölder continuity assumptions rather than Lipschitz ones. In [16], a standard one block gradient method with complexity $\mathcal{O}(\epsilon^{-\frac{\beta+1}{\beta}})$ was provided under Hölder continuity of the gradient. Their method does not employ line searches, but depends on the (*a priori* unknown) Hölder constant. Algorithm 1 recovers a gradient-type method if we take $B_{(i)}(x^k) = 0$. An one-block method that employs high-order regularizations for constrained problems under Hölder conditions is discussed in [9]. It uses only the descent criterion based on the tolerance ϵ (see Step 5 of Algorithm 1) achieving the complexity $\mathcal{O}(\epsilon^{-\frac{\beta+1}{\beta}})$ on the number of outer iterations when a quadratic regularization is employed. Considering both decay rates, we obtain a complexity of $\mathcal{O}(\epsilon^{-\frac{2}{\beta}})$, which is slightly worse. However, next it will be shown that, by assuming only the decrease based on ϵ as in [9], we recover the complexity $\mathcal{O}(\epsilon^{-\frac{\beta+1}{\beta}})$, the same present in other works that use quadratic regularizations under Hölder assumptions; see for example [10, 16].

Theorem 8. *Suppose that Assumptions 1, 2, and 3 hold, and let β and η as in Lemma 5. Also, suppose that in Step 5 of Algorithm 1 we only consider the descent condition based on ϵ, that is,*

$$f(x^{\text{trial}}) \leq f(x^k) - \frac{\alpha}{16\sigma_k}\epsilon^2.$$

Then, after at most

$$\overline{T} = \left[\frac{\max\left\{0, f(x^0) - \max\{\underline{f}, f_{\text{est}}\}\right\}\tilde{\eta}}{\alpha}\right]\epsilon^{-\frac{\beta+1}{\beta}}$$

outer iterations, Algorithm 1 stops, where $\tilde{\eta} = 32\max\{\underline{\sigma}_i(1) \mid i = 1, \ldots, q\}$. Furthermore, the number of evaluations of f is bounded above by $\overline{T} + \log_2(2\sigma_{\max}/\sigma_0)$, where $\sigma_{\max} = \epsilon^{\frac{\beta-1}{\beta}} \max\{\underline{\sigma}_i(1) \mid i = 1, \ldots, q\}$.

Proof. The proofs of Theorems 6 and 7 can be adapted in a straightforward way if we are able to state Lemma 5 with $\epsilon^{\frac{\beta+1}{\beta}}$ instead of $\epsilon^{\frac{2}{\beta}}$ and $\tilde{\eta}$ instead of η, considering only the descent condition based on ϵ. In fact, note that the proof of Lemma 5 remains valid until (10). Now, suppose that

$$f(x^{k+1}) \leq f(x^k) - \frac{\alpha}{16\sigma_k}\epsilon^2.$$

As $\sigma_k \leq 2\sigma_{\max}$, we have

$$f(x^{k+1}) \leq f(x^k) - \frac{\alpha}{32\sigma_{\max}}\epsilon^2 = f(x^k) - \frac{\alpha}{\tilde{\eta}\epsilon^{\frac{\beta-1}{\beta}}}\epsilon^2 = f(x^k) - \frac{\alpha}{\tilde{\eta}}\epsilon^{\frac{\beta+1}{\beta}},$$

as we wanted. □

Remark 1. *Although the descent criterion based on ϵ leads to a better complexity result, we prefer to state Algorithm 1 using it together with the criterion based on the magnitude of direction, which is present in several papers in this context. However, note that all the results of this section allow us to use one of these criteria alone, taking the maximum between $\alpha\epsilon^2/(16\sigma_k)$ and $\alpha\|s_{(i)}^k\|^2$ in Step 5 instead of minimum, or using any convex combination of them:*

$$f(x^{k+1}) \leq f(x^k) - \left[(1-\lambda)\frac{\alpha}{16\sigma_k}\epsilon^2 + \lambda\alpha\|s_{(i)}^k\|^2\right],$$

where $\lambda \in [0,1]$ is fixed. We can even use different descent conditions depending on the stage of the minimization process based, for example, on whether x^k appears to be close to the solution or whether σ_k is large.

Remark 2. *It is possible to achieve the complexity of Theorem 8 with a criterion based on the magnitude of the direction if, for example, we change $\alpha\|s_{(i)}^k\|^2$ by $\alpha\sigma_k\|s_{(i)}^k\|^2$ in Step 5 of Algorithm 1. Note that Theorem 4 guarantees the well-definiteness of the algorithm in this case. However, we prefer to maintain the less stringent descent criterion without σ_k because it favors the acceptance of the directions computed in Step 4 more often, which hopefully maintains σ_k small and, as we already mentioned, it is used in previous papers. Also, note that under Lipschitz continuity assumptions ($\beta = 1$), the complexity of Theorems 6, 7, and 8 coincide.*

2.2 Practical aspects of implementation

Next, we discuss how Algorithm 1 can be implemented in practice. A code written in Julia is freely available at https://github.com/leonardosecchin/bcd.

Block selection rule

At each iteration, we have to select a block index i in Step 2 whenever $C \neq \emptyset$. Different strategies can be adopted in practice, as long as we ensure that all blocks are visit in each cycle. In our implementation, the user can provide their own selection rule. Three are pre-implemented: ascending cyclic selection, random selection, and selecting the block with the worst optimality measure $\|\nabla_{(i)}f(x^k)\|_\infty$ first.

Stopping criteria

Besides the stopping criteria in Steps 1 and 2 of Algorithm 1, we implemented two criteria associated with failure: stopping by "lack of progress", which takes place if the objective function does not improve during N consecutive iterations. This occurs when $\max_{j=k-N,\ldots,k-1} f(x^j) \leq f(x^k) + \varepsilon_f|f(x^k)|$ (suppose $k > N$ for simplicity), where $\varepsilon_f > 0$ is parameter. The second criterion is, as usually, the maximum number of outer iterations.

Computing directions

Lemma 1 gives a practical way for computing the direction $s^k_{(i)}$ in Step 4 of Algorithm 1. If $B_{(i)}(x^k)+2\sigma_k I$ is a positive definite matrix, then a minimizer of (4) can be obtained directly by solving the symmetric linear system

$$\Big(B_{(i)}(x^k) + 2\sigma_k I\Big) z = -\nabla_{(i)} f(x^k) \tag{11}$$

(remember that $\sigma_k \geq 1$). Julia has an efficient integrated solver to handle this system that uses a sparse LDL^t factorization. When the MA57 routine provided by HSL [7] is available, we use it instead. MA57 is suitable for sparse, symmetric (possibly indefinite) systems and can be accessed through the packages `HSL.jl` (https://github.com/JuliaSmoothOptimizers/HSL.jl) and `libHSL`, the latter available for academic use at https://licences.stfc.ac.uk/product/libhsl.

Descent criterion

As stated in Remark 1, different descent rules in Step 5 are possible by combining the quantities $E = \alpha\epsilon^2/(16\sigma_k)$ and $S = \alpha\|s^k_{(i)}\|^2$. The user can pass your own combination of E and S, which can be varies along iterations according to the data of the current iteration. Two simple rules are pre-implemented: the one presented in Algorithm 1, which uses $\min\{E, S\}$, and another defined similarly but using $\max\{E, S\}$. The former is the standard one.

About the cost of the line search

Algorithm 1 employs a line search in which f needs to be evaluated. It is known that the cost of computing gradients is proportional to the cost of computing function values for generic functions [11]. Thus, it appears that the gain of computing partial gradients is lost while using line searches that require the evaluation of f. However, there are situations where BCD methods with line searches make sense, for example, when minimizing composite functions [12], where the "expensive part" can be computed over each block separately. One natural example is LASSO, in which the term $\frac{1}{2}\|Ax-b\|^2$ appears in the objective function. Writing

$$f(x) = \frac{1}{2}\|Ax-b\|^2 = \sum_{i=1}^{q} \frac{1}{2}\|A_{(i)}x_{(i)} - b\|^2,$$

where $A_{(i)}$ is the $m \times n_i$ matrix formed by the columns of A with indices in block i. By storing the products $A_{(i)}x_{(i)}$, we can evaluate $f(x+U_i s_{(i)})$ efficiently only computing $A_{(i)}(x_{(i)}+s_{(i)})$, as long as $f(x)$ is known. In our numerical tests, we consider the related the problem where $\frac{1}{2}\|\cdot\|^2$ is changed to $\frac{1}{p}\|\cdot\|_p^p$, $p > 1$, for which the above discussion can be adapted straightforward.

3 Numerical experiments

We consider the problem

$$\min\ f(x) = \frac{1}{p}\|Ax-b\|_p^p \quad \text{s.t.} \quad x \in \mathbb{R}^n, \tag{12}$$

where A is a $m \times n$ matrix. For $p \in (1,2)$, function $\|\cdot\|_p^p$ has Hölder continuous gradient, but not Lipschitz. As discussed in the previous section, $f(x)$ can be updated efficiently when only $x_{(i)}$ changes. Similarly, we can compute efficiently partial gradients

$$\nabla_{(i)} f(x) = A^t_{(i)} \sum_{i \in (i)} e_i |(Ax-b)_i|^{p-1} \mathrm{sgn}(Ax-b)_i$$

where the sum is taken over the indices of the block i, e_i is the canonical vector with appropriate dimension, and $\operatorname{sgn} z \in \{-1, 1\}$ denotes the sign of z. In fact, writing $Ax - b = [A_{(1)}x_{(1)} \cdots A_{(q)}x_{(q)}] - b$, it is possible to update $Ax - b$ only recomputing $A_{(i)}x_{(i)}$ when $x_{(i)}$ changes.

Test setup and parameters

For $p \in (1, 2)$, the partial Hessians of f are even not defined when some coordinate is zero. To overcome this, we take

$$B_{(i)}(x) = A_{(i)}^t \widetilde{D}_{(i)}(x) A_{(i)}, \quad [\widetilde{D}_{(i)}(x)]_{jj} = (p-1) \min\{|(Ax-b)_j|^{p-2}, 10^3\}.$$

This choice approximates the true Hessian for $p > 2$. Note that $B_{(i)}(x^k) + 2\sigma_k I$ is symmetric and positive definite, allowing us to compute descent directions as discussed in Section 2.2.

Tests were ran in a computer equipped with Intel© Xeon© Silver 4114 CPU 2.20GHz, 160 Gb RAM, under Ubuntu 24.04.3 LTS. Julia v1.11.5 is used. We selected $m \times n$ matrices A with $m > n$ from the University of Florida Sparse Matrix Collection (https://sparse.tamu.edu), as described in Table 1. The right-hand side b is randomly chosen. We fix $p = 1.5$ and apply Algorithm 1 starting from the origin. We set $\alpha = 10^{-4}$, $\theta = 1$, $\varepsilon = 10^{-3}$, $\sigma_0 = 1$, $f_{\text{est}} = -\infty$, and $\varepsilon_f = 10^{-8}$. Data are divided into q blocks of balanced size, where $q = \lceil n/n_b \rceil$ and n_b is the desired number of variables per block. Thus, the smaller n_b is, the greater the number of blocks q. The maximum number of iterations was set to $k_{\max} = \max\{5,000; 100q\}$ and N, the number of consecutive outer iterations to declare lack by progress, to $\lceil k_{\max}/5 \rceil$.

Varying the sizes of the blocks

We perform tests varying the number n_b of variables in each block. Here, blocks are formed by consecutive variables. We consider the values n_b equal to 0.5%, 1%, 5%, 10%, 15%, and 20% of the number n of variables. In our experiments, the three block selection rules described previously performed similarly. Therefore, we report only the results obtained with cyclic selection.

Figure 1 presents the performance profiles [5] for outer iterations and CPU time across the different block sizes considered. As expected, both the number of blocks and their sizes affect the behavior of the algorithm. On the one hand, larger blocks improve the quality of the directions towards the solution, as they carry information from more variables, but the cost of computing these directions is higher. On the other hand, smaller blocks make each iteration computationally cheaper, but more iterations are required to achieve convergence. This can be observed by comparing the performance profiles in Figure 1: although larger values of n_b lead to convergence in fewer iterations, they result in higher total CPU time. Therefore, the block size should be chosen by balancing the cost per iteration and the overall effort to solve the problem. In our tests, a block of size around 10% of n is the best overall choice in terms of CPU time.

Table 1 presents the detailed results for n_b equal to 10% of n. Column "opt" contains the maximum sup-norm of all partial gradients at the final iterate. Column "block size" corresponds to the approximate size of the blocks, that is, $n/10$. We observe that in almost all problems, optimality tolerance was achieved.

4 Conclusions

We proposed flexible block coordinate methods for unconstrained optimization under Hölder continuity assumptions, with the ability to ensure convergence to stationary points and to obtain strong complexity results compatible with those of "1-block" methods that require Lipschitz/Hölder continuity [9]. The proposed framework is fully implementable and features a flexible and transparent block visitation control, as well as a legitimate certificate of stationarity as a stopping criterion, an uncommon feature in previous

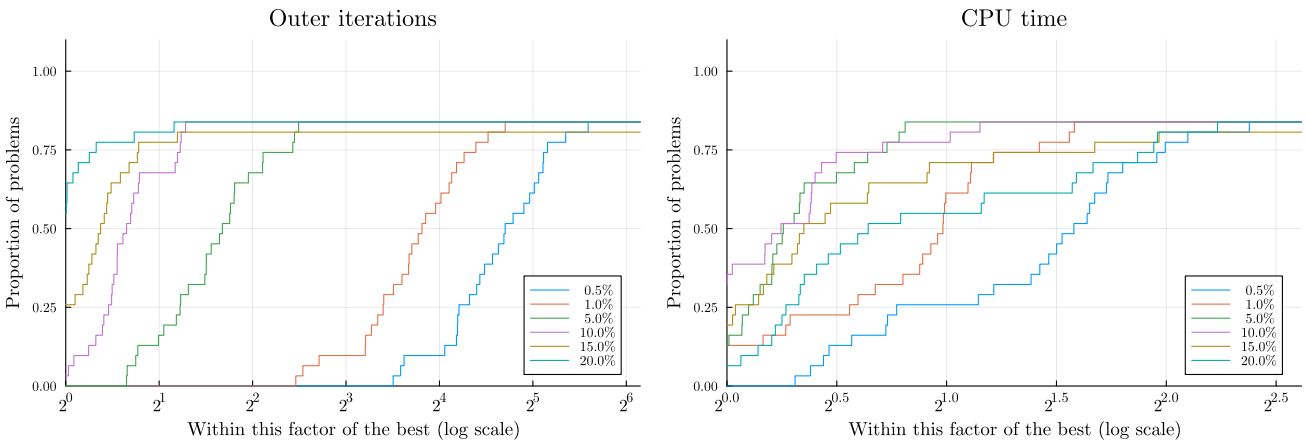

Figure 1: Performance profiles for n_b varying from 0.5% to 20% of n.

block descent methods. In particular, we extended the complexity results for the cyclic BCD method established in [1] to more general block selection rules.

The theoretical analysis establishes well-definiteness and convergence of the method under Hölder continuity, thus relaxing the classical Lipschitz continuity assumptions typically adopted in the literature [2, 4, 8, 13]. The algorithm is based on quadratic approximations of the objective function combined with a quadratic regularization. We derive complexity results comparable to those obtained under stronger smoothness assumptions, demonstrating the robustness of the approach in more general settings. From a computational perspective, we provide freely available implementations in Julia, including an interface with the well-established MA57 package from HSL [7]. In addition, the sufficient descent criteria adopted in this framework are flexible, simultaneously considering reductions related to the optimality tolerance and to the magnitude of the descent direction. These criteria, previously treated separately in the literature, are unified in our analysis, allowing multiple combinations and leading to more flexible numerical strategies.

As future research directions, we highlight: (i) the investigation of adaptive and stochastic block selection strategies; (ii) extensions to broader classes of problems, including constrained ones; (iii) the use of high-order regularizations, especially cubic regularization, for which there exists effective numerical strategies to compute directions [2]; and (iv) the development and incorporation of cheap and robust numerical strategies for solving large-scale problems, as done in [8], which may further expand the applicability of the proposed methodology to specific problems. This is possible, in particular, due to the flexibility in choosing the Hessian approximation and to the inexact computation of directions (Step 4 of Algorithm 1).

References

[1] V. Amaral. A partially derivative-free cyclic block coordinate descent method for nonseparable composite optimization. *Mathematical Modelling and Analysis*, 30(3):535–552, Sep. 2025.

[2] V. S. Amaral, R. Andreani, E. G. Birgin, D. S. Marcondes, and J. M. Martínez. On complexity and convergence of high-order coordinate descent algorithms for smooth nonconvex box-constrained minimization. *Journal of Global Optimization*, 84(3):527–561, Apr. 2022.

[3] A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, Jan. 2013.

[4] E. G. Birgin and J. M. Martínez. Block coordinate descent for smooth nonconvex constrained minimization. *Computational Optimization and Applications*, 83(1):1–27, July 2022.

[5] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[6] D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, Apr. 2006.

[7] HSL. A collection of fortran codes for large scale scientific computation. Disponível em http://www.hsl.rl.ac.uk.

[8] R. Lopes, S. A. Santos, and P. J. S. Silva. Accelerating block coordinate descent methods with identification strategies. *Computational Optimization and Applications*, 72(3):609–640, Jan. 2019.

[9] J. M. Martínez. On high-order model regularization for constrained optimization. *SIAM Journal on Optimization*, 27(4):2447–2458, Jan. 2017.

[10] J. M. Martínez and M. Raydan. Cubic-regularization counterpart of a variable-norm trust-region method for unconstrained minimization. *Journal of Global Optimization*, 68(2):367–385, Oct. 2017.

[11] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[12] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1–2):1–38, Dec. 2014.

[13] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, Mar. 2015.

[14] L. Wu and Y. Yang. Nonnegative elastic net and application in index tracking. *Applied Mathematics and Computation*, 227:541–552, Jan. 2014.

[15] L. Wu, Y. Yang, and H. Liu. Nonnegative-lasso and application in index tracking. *Computational Statistics & amp; Data Analysis*, 70:116–126, Feb. 2014.

[16] M. Yashtini. On the global convergence rate of the gradient descent method for functions with Hölder continuous gradients. *Optimization Letters*, 10(6):1361–1370, 2016.

Table 1: Computational tests with ℓ_p-least squares using blocks containing around 10% of variables each, thus totalizing 10 blocks.

Name	(m, n)	block size	iter	f	opt	time (s)
162bit	(3,606; 3,597)	359.7	*5,000*	*2.77e+01*	*1.77e+00*	*79.23*
176bit	(7,441; 7,431)	743.1	*5,000*	*5.55e+01*	*3.15e+00*	*318.23*
ch6-6-b3	(5,400; 2,400)	240.0	132	4.51e+02	9.64e−04	2.30
ch7-6-b3	(12,600; 4,200)	420.0	368	1.16e+03	9.91e−04	24.88
ch7-8-b2	(11,760; 1,176)	117.6	151	1.02e+03	9.95e−04	3.36
ch7-9-b2	(17,640; 1,512)	151.2	100	1.54e+03	9.97e−04	4.77
cis-n4c6-b3	(5,970; 1,330)	133.0	170	6.55e+02	8.35e−04	2.17
Franz4	(6,784; 5,252)	525.2	254	3.66e+02	9.28e−04	12.46
Franz5	(7,382; 2,882)	288.2	297	5.44e+02	9.43e−04	8.55
Franz6	(7,576; 3,016)	301.6	259	5.76e+02	9.97e−04	8.07
Franz7	(10,164; 1,740)	174.0	163	8.86e+02	9.93e−04	4.39
Franz8	(16,728; 7,176)	717.6	218	1.26e+03	9.48e−04	35.81
Franz9	(19,588; 4,164)	416.4	220	1.55e+03	9.81e−04	25.91
Franz10	(19,588; 4,164)	416.4	227	1.54e+03	9.72e−04	26.55
GL7d12	(8,899; 1,019)	101.9	295	2.13e+03	8.16e−04	4.84
Kemelmacher	(28,452; 9,693)	969.3	*5,000*	*3.94e+03*	*2.26e+01*	*1466.54*
Maragal_5	(4,654; 3,320)	332.0	*5,000*	*4.92e+02*	*7.20e−02*	*135.90*
Maragal_4	(1,964; 1,034)	103.4	*5,000*	*1.98e+02*	*9.85e−02*	*20.49*
mk10-b3	(4,725; 3,150)	315.0	302	2.87e+02	9.93e−04	4.90
mk11-b3	(17,325; 6,930)	693.0	274	1.44e+03	9.98e−04	37.81
mk12-b2	(13,860; 1,485)	148.5	107	1.20e+03	8.89e−04	3.07
n2c6-b4	(3,003; 1,365)	136.5	125	1.99e+02	8.22e−04	0.74
n2c6-b5	(4,945; 3,003)	300.3	145	3.81e+02	9.13e−04	2.74
n2c6-b6	(5,715; 4,945)	494.5	264	2.93e+02	9.96e−04	11.74
n3c6-b4	(3,003; 1,365)	136.5	96	1.97e+02	7.84e−04	0.56
n3c6-b5	(5,005; 3,003)	300.3	145	3.91e+02	9.90e−04	2.78
n3c6-b6	(6,435; 5,005)	500.5	222	3.40e+02	9.95e−04	11.63
n4c5-b4	(2,852; 1,350)	135.0	101	1.86e+02	8.92e−04	0.56
n4c5-b5	(4,340; 2,852)	285.2	168	3.18e+02	9.75e−04	2.61
n4c5-b6	(4,735; 4,340)	434.0	151	2.37e+02	9.99e−04	3.93
n4c6-b3	(5,970; 1,330)	133.0	131	6.56e+02	8.24e−04	1.48