Les Cahiers du GERAD ISSN: 0711–2440

A multi-precision quadratic regularization method for unconstrained optimization with rounding error analysis

D. Monnet, D. Orban

G–2023–18

June 2023
Revised: March 2025

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2023
– Bibliothèque et Archives Canada, 2023

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2023
– Library and Archives Canada, 2023

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A multi-precision quadratic regularization method for unconstrained optimization with rounding error analysis

Dominique Monnet [a, b]

Dominique Orban [a, b]

[a] *GERAD, Montréal (Qc), Canada, H3T 1J4*

[b] *Polytechnique Montréal, Montréal (Qc), Canada, H3T 1J4*

`dominique.monnet@polymtl.ca`
`dominique.orban@polymtl.ca`

June 2023
Revised: March 2025
Les Cahiers du GERAD
G–2023–18

Abstract : We propose a multi-precision extension of the Quadratic Regularization (R2) algorithm that enables it to take advantage of low-precision computations, and by extension to decrease energy consumption during the solve. The lower the precision in which computations occur, the larger the errors induced in the objective value and gradient, as well as in all other computations that occur in the course of the iterations. The Multi-Precision R2 (MPR2) algorithm monitors the accumulation of rounding errors with two aims: to provide guarantees on the result of all computations, and to permit evaluations of the objective and its gradient in the lowest precision possible while preserving convergence properties. MPR2's numerical results show that single precision offers enough accuracy for more that halve of objective evaluations and most of gradient evaluations during the algorithm's execution. However, MPR2 fails to converge on several problems of the test set for which double precision does not offer enough precision to ensure the convergence conditions before reaching a first order critical point. That is why we propose a practical version of MPR2 with relaxed conditions which converges for almost as many problems as R2 and potentially enables to save about 50 % time and 60% energy for objective evaluation and 50 % time and 70% energy for gradient evaluation.

Keywords: Unconstrained optimization, quadratic regularization, multi precision, rounding error analysis

1 Introduction

Over the recent years, the rise of Deep Neural Networks (DNN) and the continuous increase of the size of the networks has motivated researches to reduce the training cost. This cost includes, for instance, the training time and the energy dissipation. A natural way to do that is to perform the computations in low precision [25, 26], since dividing by two the number of digits roughly divide by two the computation time and by four the energy consumption [15]. Low-precision hardware architectures have been developed to serve the purpose of DNN fast training at lower energy cost. In this context, it becomes possible to use several precision levels, and to extend the DNN training methods to multi-precision.

DNN training, *i.e.*, minimizing the loss function, is a non-linear optimization problem. Using low-precision computations come at the price of evaluating the objective function the gradient, and possibly higher derivative orders, with errors. Convergence of non-linear algorithms under such evaluation errors has received attention in the recent years. For the Trust Region (TR) methods, the state-of-the-art reference [14] already includes the convergence conditions to inexact objective and gradient evaluations with deterministic error bounds. These results were recently extended to the multi-precision case in [18], which proves the convergence of TR if infinite precision evaluations can be performed. [24] consider TR methods with deterministic error bounds, and proposes to relax the acceptance step condition by including the error bounds. This enables to keep the TR radius large, but the convergence only guaranteed to the neighborhood of a critical point. TR convergence is also analyzed with probabilistic error bounds. In this context, the convergence is provided in the sense of a probability which increases with the number of iteration. TR methods are proved to converge with a probability 1 with exact objective function evaluation and probabilistic fully-linear model (which includes inexact order 1 and 2 derivative evaluations) in [4, 16]. More recent works study convergence of TR by further including probabilistic error bounds on the objective function [9, 10].

In the multi-precision context, one can assume that infinite precision is not available. That is, it is not possible to access the exact values of the objective and its derivatives. This raises the question of, given an optimality criterion, what level of optimality can be reached with the available precision. The question is studied in the TR context in [7], where errors are bounded by some absolute constant.

In the recent years, regularization methods have emerged as an alternative to TR. These methods are based on local Taylor expansion models, to which is added a regularization parameters that controls the step size [11, 12], and have the same complexity as TR methods. This complexity is $O(\epsilon^{-2})$ with ϵ the tolerance on the first order condition. Recent work extends regularization methods' proof of convergence to inexact evaluation of objective function and gradient with deterministic and probabilistic error bounds [5], and deterministic bounds in the context of least-squares problems [6]. In particular, [5] can be viewed as the multi-precision regularization counterpart of [18]. In [17], a composite problem incorporating a non-smooth term in the objective function is considered with deterministic error bounds on objective and gradient evaluations and shows that the complexity becomes $O(|\log(\epsilon)|\epsilon^{-2})$ for this class of problems.

In the context of multiple precision computations, mentioned in the first paragraph, considering only objective function and gradient (and possibly further derivative orders) is not enough to ensure convergence. Indeed, running an algorithm with finite precision computations, and in particular with (very) low-precision computations, implies that any computed value is inexact. The purpose of this paper is to propose a comprehensive convergence analysis of a multi-precision algorithm based on first order regularization methods, MPR2, by taking into account all the possible sources of error due to finite precision computations. To the best of the author's knowledge, such a study has not been proposed for non-linear methods. Note that [7] and [18] studied multi-precision extension of TR algorithm but only consider inexact evaluations of the objective function and its derivative. In the present paper, we also consider the effect of finite-precision arithmetic on operations carried out at each iteration of an algorithm, and that compound evaluation errors on the objective and its gradient.

More generally, our aim is to point out certain pitfalls of low-precision computations, and provide a method with guaranteed accuracy.

This paper is organized as follows. Section 2 introduces the problem we are aiming to solve, how the inexact evaluations of the objective function and gradient are modelled and the R2 algorithm in this context of inexact evaluations. Section 3 introduces Floating Points (FP) computations. It lists the sources of errors induced by finite precision computations occurring in R2 and provides bounds on these errors. Section 4.1 presents MPR2, the multi-precision variant of R2 that takes all the errors listed in Section 3 into account. Section 5 shows the numerical results obtain with MPR2. Section 6 recaps the results presented in this paper and devises on future research directions. Finally, the proof of convergence of MPR2 is given in Appendix A.

2 Problem statement and background

We consider the problem

$$\min_{x\in\mathbb{R}^n} \; f(x), \tag{1}$$

with $f : \mathbb{R}^n \to \mathbb{R}$ continuously differentiable. We assume that evaluations and computations are performed in finite precision, and that a set of FP systems is available. For example, modern hardware may offer access to half, single and double precision units. As a consequence, it is not possible to evaluate f nor ∇f exactly, but only finite precision approximations, denoted $\widehat{f}$ and $\widehat{g}$. For more generality, we assume that models ω_f and ω_g of the evaluation errors are provided as functions of x:

$$|f(x) - \widehat{f}(x)| \le \omega_f(x), \quad \|\nabla f(x) - \widehat{g}(x)\| \le \omega_g(x)\|\widehat{g}(x)\|, \tag{2}$$

for all $x \in \mathbb{R}^n$. Note that ω_f and ω_g can account for any kind of evaluation errors such as, for example, gradient approximation in a derivative-free context, and are not limited to finite precision computation errors.

We make the following assumption.

Assumption 1. There exists $L > 0$ such that $\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|$ for all $x, y \in \mathbb{R}^n$.

The quadratic regularization method [3], called here R2, is based on the first-order model

$$T(x, s) = f(x) + \nabla f(x)^T s. \tag{3}$$

Under Assumption 1 [8],

$$|f(x + s) - T(x, s)| \le \tfrac{1}{2}L\|s\|^2. \tag{4}$$

Since f and $\nabla f(x)$ cannot be computed exactly, neither can T. We must rely instead on the approximate Taylor series

$$\overline{T}(x, s) = \widehat{f}(x) + \widehat{g}(x)^T s. \tag{5}$$

The regularized approximate Taylor series,

$$m(x, s) = \overline{T}(x, s) + \tfrac{1}{2}\sigma\|s\|^2, \tag{6}$$

is used as local model in the algorithm, where $\sigma > 0$ is the regularization parameter that controls the step size. At each iteration, the step is chosen as the minimizer of the local model:

$$s \in \arg\min_{s\in\mathbb{R}^n} \; m(x, s) = \left\{ -\frac{1}{\sigma}\widehat{g}(x) \right\}. \tag{7}$$

Algorithm 1 is an adaptation of the standard R2 algorithm that takes objective and gradient errors into account. It is proved to converge to a first order critical point [5] under the assumption that all other computations in the algorithm are performed in exact arithmetic. In the algorithm we use $\omega_g(x_k) \leq \kappa_\mu$ to control the error on gradient evaluation instead of $\omega_g(x_k) \leq 1/\sigma_k$, which is employed in [5]. Doing so avoids demanding excessive accuracy on the gradient when σ_k is large.

The stopping condition in Step 1 of Algorithm 1 ensures that $\|\nabla f(x_k)\| \leq \epsilon$ while taking the error on the gradient evaluation into account. In Step 2, if the convergence condition $\omega_g(x_k) \leq \kappa_\mu$ is not satisfied, the evaluation precision on the gradient is increased and the gradient is recomputed with a smaller $\omega_g(x_k)$ in Step 1. In Step 3, a suitable evaluation precision for the objective function must be chosen such that $\omega_f(x_k)$ and $\omega_f(c_k)$ are lower than $\eta_0\overline{\Delta T}_k$ to ensure convergence.

Algorithm 1 Quadratic regularization algorithm with inexact evaluations

Step 0: Initialization: Choose an initial point $x_0 \in \mathbb{R}^n$, initial $\sigma_0 > 0$, minimal value $\sigma_{\min} > 0$ for σ, final gradient tolerance $\epsilon > 0$, and constants

$$0 < \eta_1 \leq \eta_2 < 1, \quad 0 < \gamma_1 < 1 < \gamma_2 \leq \gamma_3, \quad \eta_0 < \tfrac{1}{2}\eta_1, \quad \eta_0 + \tfrac{1}{2}\kappa_\mu \leq \tfrac{1}{2}(1-\eta_2). \tag{8}$$

Set $k = 0$, compute $f_0 = \widehat{f}(x_0)$. Select a precision level for gradient evaluation.
Step 1: Check for termination: If $k = 0$ or $\rho_{k-1} \geq \eta_1$ (previous iteration successful), compute $g_k = \widehat{g}(x_k)$. Terminate if

$$\|g_k\| \leq \frac{\epsilon}{1+\omega_g(x_k)}. \tag{9}$$

Step 2: Step calculation: Compute $s_k = -g_k/\sigma_k$.
If $\omega_g(x_k) > \kappa_\mu$, increase gradient evaluation precision and go to Step 1.
Compute $c_k = x_k + s_k$.
Compute predicted reduction $\overline{\Delta T}_k := \overline{T}(x_k, s_k) - \widehat{f}(x_k) = g_k^T s_k$.
Step 3: Evaluate the objective function: Choose the objective function evaluation precision such that $\omega_f(x_k) \leq \eta_0\overline{\Delta T}_k$, and $\omega_f(c_k) \leq \eta_0\overline{\Delta T}_k$.
Compute $f_k^+ = \widehat{f}(c_k)$ and recompute $f_k = \widehat{f}(x_k)$ if evaluation precision different that $\widehat{f}(c_{k-1})$.
Step 4: Acceptance of the trial point: Define the ratio

$$\rho_k = \frac{f_k - f_k^+}{\overline{\Delta T}_k} \tag{10}$$

If $\rho_k \geq \eta_1$, set $x_{k+1} = c_k$, $f_{k+1} = f_k^+$. Otherwise, set $x_{k+1} = \widehat{x}_k$, $f_{k+1} = f_k$.
Step 5: Regularization parameter update:

$$\sigma_{k+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1\sigma_k), \sigma_k] & \text{if } \rho_k \geq \eta_2 \\ [\sigma_k, \gamma_2\sigma_k] & \text{if } \rho_k \in [\eta_1, \eta_2) \\ [\gamma_2\sigma_k, \gamma_3\sigma_k] & \text{if } \rho_k < \eta_1. \end{cases} \tag{11}$$

Increment $k \leftarrow k+1$, and go to Step 1.

3 Finite precision preliminaries

Finite precision computations induce not only inexact evaluations of the objective function and the gradient, but also other errors that must be taken into account. In this section, the sources of errors due to finite precision computations are analyzed. Mechanisms are added to Algorithm 1 to ensure the convergence in spite of these errors. Algorithm 2, introduced later in Section 4.1, is the adapted version of Algorithm 1 robust to finite precision errors.

3.1 IEEE 754 norm and dot product error

Let $\mathbb{F}$ denote a FP system in base 2 with mantissa of length p. The *machine epsilon* is $\epsilon_M := 2^{1-p}$. If the rounding strategy is round-to-nearest, the *unit round-off* is $u := 2^{-p} = \frac{1}{2}\epsilon_M$. For $x \in \mathbb{R}$, $\mathrm{fl}(x) \in \mathbb{F}$ is the representation of x in $\mathbb{F}$. Provided no overflow occurs when representing x, $\mathrm{fl}(x) = x(1+\delta)$ where $|\delta| \leq u$. The IEEE standard [2] requires that arithmetic operations produce a result such that

$$\mathrm{fl}(x \text{ op } y) = (x \text{ op } y)(1+\delta), \quad |\delta| \leq u, \quad \text{op} \in \{+,-,\times,/\},\ x, y \in \mathbb{F}. \tag{12}$$

We make the following assumption.

Assumption 2. Finite precision computations comply with the IEEE 754 norm, underflow and overflow do not occur during algorithm execution and the rounding mode is round to nearest (RN).

Successions of n arithmetic operations incur multiple errors of the form $1+\delta_i$ each, with $|\delta_i| \le u$. The notation used to study rounding error propagation is that of [19]:

$$\prod_i^n (1+\delta_i)^{\pm 1} = (1+\theta_n), \quad |\theta_n| \le \xi_n, \tag{13}$$

where several expressions for ξ_n, summarized in Table 1, were given over time. Bounds such as (13) can be overly pessimistic because they account for the worst-case propagation of errors.

Table 1: Formulas for ξ_n from the literature.

ξ_n	$\dfrac{nu}{1-nu}$	$\dfrac{nu}{1-nu/2}$	nu
Reference	[19]	[13]	[20]

One particular use of ξ_n is to bound the error on the computation of the dot product of x and $y \in \mathbb{R}^n$ [19]:

$$|\operatorname{fl}(x^T y) - x^T y| \le \xi_n |x|^T |y|, \tag{14}$$

where $|x|$ is the vector whose components are the absolute values of those of x.

We make the following further assumption.

Assumption 3. The unit roundoff of the lowest precision FP format u_{max} is such that $\xi_{n+2} = (n+2)u_{max} < 1$.

Assumption 3 is necessary to ensure the convergence of the multi-precision extension of Algorithm 1, and constrains the size of the problems that we consider in a given a FP format. For example, $u = \mathcal{O}(10^{-8})$ for single precision, which limits the size of the problem to $n = \mathcal{O}(10^8)$ with the least pessimistic bound $\xi_n = nu$ of Table 1.

3.2 Notation and conventions

Throughout, we use the following notation. The reader can refer to [19] for the notions related to rounding error accumulations.

1. We denote $\Pi = \{1, \dots, \pi_{\max}\}$ the set of indices of available FP formats. We assume that Π is ordered by decreasing unit round-off of the corresponding FP formats. For example, $\Pi = \{1,2,3\}$ corresponds to $\{\text{Float16}, \text{Float32}, \text{Float64}\}$.
2. δ denotes a rounding error. An index is added if there is a need to keep track of a particular rounding error or to clarify the calculations. However, because we mostly focus on error bound $|\delta| < u$, there is often no need to keep track of particular rounding error and we abusively drop the index. For example, in the analysis $\operatorname{fl}(c/\operatorname{fl}(a+b)) = \operatorname{fl}(\dfrac{c}{(a+b)(1+\delta)}) = \dfrac{c}{a+b}\dfrac{1+\delta}{1+\delta}$, δ is employed to denotes two different rounding error bounded as $|\delta| \le u$. Therefore, $\dfrac{c}{a+b}\dfrac{1+\delta}{1+\delta} \neq c/(a+b)$ but we write, following (13), $\dfrac{c}{a+b}\dfrac{1+\delta}{1+\delta} = (a+b)/c\,(1+\theta_2)$ with $|\theta_2| < 2u$. There is no need here to know which δ corresponds to $+$ or $/$ operators to perform the rounding error analysis.
3. FP numbers are denoted with a hat, $\widehat{x} \in \mathbb{F}$.
4. The notation tilde is used to denote a quantity that includes a rounding error. For example $\operatorname{fl}(\widehat{x}+\widehat{y}) = (\widehat{x}+\widehat{y})(1+\delta) = \widetilde{x}+\widetilde{y}$ with $\widetilde{x} = \widehat{x}(1+\delta)$ and $\widetilde{y} = \widehat{y}(1+\delta)$. Note that in general, $\widetilde{x} \notin \mathbb{F}$.

5. θ_n models any rounding error propagation as in (13).
6. ϑ_n denotes a real number bounded as $|\vartheta_n| \leq \xi_n$. It is important to note that ϑ_n is not the result of accumulated rounding errors and is not expressed as θ_n. As a consequence, $(1+\theta_n)(1+\delta) = (1+\theta_{n+1})$ but $(1+\vartheta_n)(1+\delta) \neq (1+\vartheta_{n+1})$.
7. $\pi_k^{f^+}$ denotes the index of the FP format used to evaluate $\widehat{f}$ at $\widehat{c}_k$, and π_k^f the one used to evaluate $\widehat{f}$ at $\widehat{x}_k$. Similarly, π_k^g denotes the index of the FP format used to evaluate $\widehat{g}$.
8. π_k^x denotes the index of the FP format of the current incumbent $\widehat{x}_k$, and π_k^c the index of the FP format of the candidate $\widehat{c}_k$.
9. We use the notation $\mathrm{fl}(x)$ when the FP system in which we wish to represent x is obvious from the context. Wherever it may be ambiguous, we write $\mathrm{fl}(x, \pi)$, where $\pi \in \Pi$.
10. u_k^c denotes the unit round-off related to the FP format corresponding to π_k^c, and u_k^g the unit round-off related to the FP format corresponding to π_k^g.
11. u_{max} (respectively u_{min}) denotes the largest (resp. the smallest) unit roundoff among the available FP formats, i.e., that associated with the lowest (resp. the highest) precision.
12. ξ_n, α_n, and β_n (introduced later) are quantities that depend on the unit roundoff and accounts for rounding error of dot product and norm computations with finite precision. Since several FP formats are dealt with, the unit roundoff might be added as an argument to avoid confusion. For example, we denote $\xi_n(u_k^x)$ the quantity ξ_n defined with the unit roundoff corresponding to the FP format of $\widehat{x}_k$.
13. Given a vector $\widehat{x} \in \mathbb{F}^n$, we denote abusively,

$$\widehat{x}(1+\delta_x) = (\widehat{x}_1(1+\delta_x), \ldots, \widehat{x}_n(1+\delta_x))^T, \quad \widehat{x}\delta_x = (\widehat{x}_1\delta_x, \ldots, \widehat{x}_n\delta_x)^T, \tag{15}$$

and vice-versa, where the same δ_x is "broadcast" to every components of $\widehat{x}$. This abusive notation is also extended to δ, θ_n and ϑ_n.

Multi-format operations: When performing computations with FP numbers in different formats, we assume that the operands are implicitly cast into the highest precision format among theirs and then the operation is performed. The resulting FP number is therefore represented in this last format and the rounding errors are bounded by its unit round-off. For example, if $\widehat{x}_k$ is a vector of Float32 elements and $\widehat{s}_k$ a vector of Float64 elements, the computation of $\widehat{c}_k := \mathrm{fl}(\widehat{x}_k + \widehat{s}_k)$ first consists in casting each element of $\widehat{x}_k$ to Float64 and then performing the addition, which results in $(\widehat{x}_k + \widehat{s}_k)(1+\delta)$, i.e., a vector of Float64 elements where δ is bounded by the unit round-off of Float64. The implicit casting into another FP format of the arguments is done only in the context of the operation, $\widehat{x}_k$ still has Float32 format after computing $\widehat{c}_k$ in the above example.

Representation of $\widehat{\sigma}_k$: We assume that $\widehat{\sigma}_k := \mathrm{fl}(\sigma_k)$ is representable exactly in any of the available FP formats. This can be achieved by setting σ_0 to a power of 2 and making sure that $\widehat{\sigma}_k$ is multiplied by a power of 2 (positive of negative) at Step 5 of Algorithm 2. In this way, we can safely ignore the FP format of $\widehat{\sigma}_k$, and we assume that the FP format of $\widehat{s}_k$ is that of $\widehat{g}_k$ when computed at Step 2, as indicated in Table 2.

Forbidden evaluations: Evaluating $\widehat{f}$ at $\widehat{x}_k$ is forbidden if $\pi_k^x > \pi_k^f$, as that would imply casting $\widehat{x}_k$ to a lower precision format. A rounding error would hence occur, and $\widehat{f}$ would be evaluated at $\widehat{x}_k(1+\delta)$ instead of $\widehat{x}_k$. In order to keep the proof of convergence theoretically sound, such evaluation is forbidden. If $\pi_k^x < \pi_k^f$, casting induces no rounding error. As for multi-format operations, casting is implicit and the FP format of the argument of $\widehat{f}$ remains the same before and after the evaluation of $\widehat{f}$. The index of the FP format of the returned value by the evaluation of $\widehat{f}$ is π_k^f. All the above applies to $\widehat{g}$ similarly. Since the format index of $\widehat{s}_k$ is π_k^g, it follows that $\pi_k^g \geq \pi_k^x$ for all k, and the elements of $\widehat{x}_k + \widehat{s}_k$ are in a FP format with index π_k^g, as indicated in Table 2.

Defined values: We assume that some values in Algorithm 2 are *defined*, that is, "computed" in exact arithmetic. These values are ρ_k, ϕ_k, ξ_n, β_n, μ_k, and u'_k, which is why those quantities are denoted without a hat. Computing those values involve few operations, and therefore few rounding errors. In practice, they can be computed in high precision cheaply, which may be assimilated to exact arithmetic for the present purposes.

Table 2 summarizes what are the indices of the FP formats of the values returned by operations and evaluations of $\widehat{f}$ and $\widehat{g}$. Table 3 is limited to three FP formats but can be extended to any number of formats. Table 4 summarizes the indices of the FP formats of the computed values.

Table 2: Indices of FP format of values returned by operations. — denotes an operation not occurring in Algorithm 2.

$+, /, \widehat{f}, \widehat{g}$	$\widehat{x}_k$	$\widehat{g}_k$
$\widehat{s}_k$	π_k^g	—
$\widehat{\sigma}_k$	—	π_k^g

Table 3: Indices of FP format of values returned by evaluation of $\widehat{f}$ and $\widehat{g}$. A blank denotes a forbidden evaluation.

		$\pi_k^{f^+}/\pi_k^g$		
		1	2	3
π_k^x	1	1	2	3
/	2		2	3
π_k^c	3			3

Table 4: Indices of FP format of the computed values in Algorithm 2.

$\widehat{x}_k$	$\widehat{s}_k$	$\widehat{c}_k$	$\widehat{f}_k$	$\widehat{f}_k^+$	$\widehat{g}_k$	$\widehat{\Delta T}_k$	$\mathrm{fl}(\|\widehat{x}_k\|)$	$\mathrm{fl}(\|\widehat{s}_k\|)$	$\widehat{\|\widehat{g}_k\|}$	$\widehat{\phi}_k$
π_k^x	π_k^g	π_k^c	$\pi_{k-1}^{f^+}/\pi_k^f$	$\pi_k^{f^+}$	π_k^g	π_k^g	π_k^x	π_k^g	π_k^g	π_k^g

3.3 Inexact step and Backward-Error Taylor series

When computing the step $\widehat{s}_k$ at iteration k in finite precision arithmetic (Step 1 of Algorithm 1), rounding errors occur and $\widehat{s}_k$ is not in the direction $\widehat{g}_k = \widehat{g}(\widehat{x}_k)$ but instead in a direction $\widetilde{g}_k$ that includes those rounding errors. Indeed,

$$\widehat{s}_k = \mathrm{fl}(-\frac{1}{\widehat{\sigma}_k}\widehat{g}_k) = -\frac{1}{\widehat{\sigma}_k}\widetilde{g}_k, \tag{16}$$

with

$$\widetilde{g}_k = \widehat{g}_k \odot ((1+\delta_1), \ldots, (1+\delta_n)), \tag{17}$$

where $\odot$ is the element-wise multiplication operator.

Note that following the remarks of Section 3.2, the FP format index of $\widehat{g}_k$ is π_k^g since $\widehat{g}_k$ results from the evaluation of $\widehat{g}$, and π_k^g is also the FP format of $\widehat{s}_k$ since we assume that $\widehat{\sigma}_k$ is representable in any FP format (see Table 4).

Lemma 1. For all k, $\|\widehat{g}_k - \widetilde{g}_k\| \le u_k^g \|\widehat{g}_k\|$ and $\|\widetilde{g}_k\| \ge \|\widehat{g}_k\|(1-u_k^g)$.

Proof. Denote $\widehat{g}_k = (\widehat{g}_k^1, \ldots, \widehat{g}_k^n)$ and $\widetilde{g}_k = (\widetilde{g}_k^1, \ldots, \widetilde{g}_k^n)$. We have from (17)

$$\|\widehat{g}_k - \widetilde{g}_k\|^2 = \sum_{i=1}^n (\widehat{g}_k^i - \widehat{g}_k^i(1+\delta_i))^2 \le \sum_{i=1}^n (u_k^g)^2(\widehat{g}_k^i)^2 = (u_k^g)^2\|\widehat{g}_k\|^2,$$

which proves the first inequality. Similarly, because $-u \le \delta \le u \implies (1-u)^2 \le (1+\delta)^2 \le (1+u)^2$, (17) implies

$$\|\widetilde{g}_k\|^2 = \sum_{i=1}^n (\widehat{g}_k^i(1+\delta_i))^2 \ge (1-u_k^g)^2\|\widehat{g}_k\|^2. \qquad \square$$

We define the *Backward Error Taylor series* (BET) as

$$\widetilde{T}(\widehat{x}_k, s) = \widehat{f}(\widehat{x}_k) + \widetilde{g}^T s. \tag{18}$$

The terminology "backward error" [19] refers to the fact that $\widetilde{g}$ includes the backward error due to the finite precision computation of $\widehat{s}_k$. The BET series, being defined with $\widetilde{g}^T$ the direction of the computed step $\widehat{s}_k$, has to be considered in place of $\overline{T}$ to prove the convergence of MPR2.

Because $\widetilde{g}$ is unknown, so is $\widetilde{T}$. However, as Lemma 2 states, computing the decrease $\widehat{\Delta T}_k$ in the approximate Taylor series $\overline{T}$ with finite precision arithmetic provides the decrease $\widetilde{\Delta T}_k$ in the unknown BET series with a relative error.

Lemma 2. Let $\widehat{\Delta T}_k := \mathrm{fl}\left(\overline{T}(\widehat{x}_k, 0) - \overline{T}(\widehat{x}_k, \widehat{s}_k)\right)$ and $\widetilde{\Delta T}_k := \widetilde{T}(\widehat{x}_k, 0) - \widetilde{T}(\widehat{x}_k, \widehat{s}_k)$. Then,

$$\widehat{\Delta T}_k = \widetilde{\Delta T}_k(1+\vartheta_{n+1}) = -\widetilde{g}_k^T \widehat{s}_k(1+\vartheta_{n+1}) = \frac{\|\widetilde{g}_k\|^2}{\widehat{\sigma}_k}(1+\vartheta_{n+1}) \tag{19}$$

with $|\vartheta_{n+1}| \le \xi_{n+1}(u_k^g)$. Furthermore, $\widehat{\Delta T}_k \ge 0$.

Proof. Using (16), (17) and the fact that $\widehat{\sigma}_k$ is representable exactly, one has [19],

$$\begin{aligned}\widehat{\Delta T}_k = \mathrm{fl}(-\widehat{g}_k^T \widehat{s}_k) &= -\,\mathrm{fl}\left(\left(\widetilde{g}_k^1/(1+\delta_1), \dots, \widetilde{g}_k^n/(1+\delta_n)\right)^T \left(-\widetilde{g}_k/\widehat{\sigma}_k\right)\right) \\ &= \frac{(\widetilde{g}_k^1)^2}{\widehat{\sigma}_k}\frac{(1+\delta)^n}{(1+\delta_1)} + \frac{(\widetilde{g}_k^2)^2}{\widehat{\sigma}_k}\frac{(1+\delta)^n}{(1+\delta_2)} + \dots + \frac{(\widetilde{g}_k^n)^2}{\widehat{\sigma}_k}\frac{(1+\delta)^2}{(1+\delta_n)} \\ &= \frac{(\widetilde{g}_k^1)^2}{\widehat{\sigma}_k}(1+\theta_{n+1}) + \frac{(\widetilde{g}_k^2)^2}{\widehat{\sigma}_k}(1+\theta_{n+1}) + \dots + \frac{(\widetilde{g}_k^n)^2}{\widehat{\sigma}_k}(1+\theta_3).\end{aligned} \tag{20}$$

According to (13), the absolute value of each perturbation term θ in the sum above is bounded by ξ_{n+1}. Since $\widehat{\sigma}_k > 0$ and we assume $\xi_{n+1} < 1$, we have $1-\xi_{n+1} \le 1+\theta_j \le 1+\xi_{n+1}$ for $j = 3, \dots, n+1$. It follows that, on the one hand,

$$\begin{aligned}\widehat{\Delta T}_k &\ge \frac{(\widetilde{g}_k^1)^2}{\widehat{\sigma}_k}(1-\xi_{n+1}(u_k^g)) + \frac{(\widetilde{g}_k^2)^2}{\widehat{\sigma}_k}(1-\xi_{n+1}(u_k^g)) + \dots + \frac{(\widetilde{g}_k^n)^2}{\widehat{\sigma}_k}(1-\xi_{n+1}(u_k^g)) \\ &\ge \frac{1}{\widehat{\sigma}_k}\widetilde{g}_k^T \widetilde{g}_k (1-\xi_{n+1}(u_k^g)) \\ &\ge \widetilde{g}_k^T \widehat{s}_k (1-\xi_{n+1}(u_k^g)),\end{aligned} \tag{21}$$

and on the other hand,

$$\widehat{\Delta T}_k \le \widetilde{g}_k^T \widehat{s}_k (1+\xi_{n+1}(u_k^g)). \tag{22}$$

Therefore, one has

$$\widetilde{g}_k^T \widehat{s}_k (1-\xi_{n+1}(u_k^g)) \le \widehat{\Delta T}_k \le \widetilde{g}_k^T \widehat{s}_k (1+\xi_{n+1}(u_k^g)), \tag{23}$$

which proves that there exists ϑ_{n+1} such that $|\vartheta_{n+1}| \le \xi_{n+1}(u_k^g)$ satisfying

$$\widehat{\Delta T}_k = -\widetilde{g}_k^T \widehat{s}_k (1+\vartheta_{n+1}). \tag{24}$$

Since $\widehat{s}_k = -\widetilde{g}_k/\widehat{\sigma}_k$, one has $-\widetilde{g}_k^T \widehat{s}_k = \|\widetilde{g}_k\|^2/\widehat{s}_k$ and,

$$\widehat{\Delta T}_k = -\widetilde{g}_k^T \widehat{s}_k (1+\vartheta_{n+1}) = \frac{\|\widetilde{g}_k\|^2}{\widehat{\sigma}_k}(1+\vartheta_{n+1}), \tag{25}$$

which establishes the first statement of the lemma.

Assumption 3 ensures that $1+\vartheta_{n+1} \ge 1-\xi_{n+1}(u_k^g) \ge 0$. As a consequence,

$$\widehat{\Delta T}_k = \frac{\|\widetilde{g}_k\|^2}{\widehat{\sigma}_k}(1+\vartheta_{n+1}) \ge \frac{\|\widetilde{g}_k\|^2}{\widehat{\sigma}_k}(1-\xi_{n+1}(u_k^g)) \ge 0. \qquad \square$$

Although $\vartheta_{n+1}(u_k^g)$ is bounded by ξ_{n+1}, it is not, strictly speaking, the result of rounding error propagation since it is not expressed as in (13). Rather, it represents the relative error between the decrease of the BET series $\widetilde{\Delta T}_k$ and the computed value of the truncated Taylor series decrease $\widehat{\Delta T}_k$.

In order to streamline notation, we denote α_n an upper bound on $1/(1+\vartheta_{n+1})$ defined as,

$$\alpha_{n+1}(u_k^g) := \frac{1}{1-\xi_{n+1}(u_k^g)} \geq \frac{1}{1+\vartheta_{n+1}} > 0. \tag{26}$$

Note that Assumption 3 ensures that α_n is well-defined and that $\dfrac{1}{1+\vartheta_{n+1}} > 0$.

3.4 Actual candidate and search direction

With finite precision computations, the computed candidate is

$$\widehat{c}_k = \text{fl}(\widehat{x}_k + \widehat{s}_k) = (\widehat{x}_k + \widehat{s}_k)(1+\delta_k^+) = \widehat{x}_k + (\delta_k^+ \widehat{x}_k + (1+\delta_k^+)\widehat{s}_k), \tag{27}$$

with $|\delta_k^+| \leq u_k^g$ (see Section 3.2 for details). As a consequence, the computed step $\widehat{s}_k$ is not the actual step. The actual step is $\widehat{x}_k\delta_k^+ + \widehat{s}_k(1+\delta_k^+)$, and therefore the actual search direction is not $-\widetilde{g}_k$ but $\widehat{\sigma}_k(\widehat{c}_k - \widehat{x}_k) = -\widetilde{g}_k(1+\delta_k^+) + \widehat{\sigma}_k\widehat{x}_k\delta_k^+$. Figure 1 illustrates how rounding errors due to step and candidate computation affect the computation of the candidate $\widehat{c}_k$.

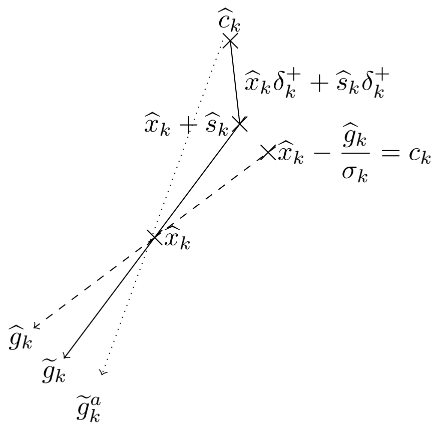

$\widehat{c}_k$
$\widehat{x}_k\delta_k^+ + \widehat{s}_k\delta_k^+$
$\widehat{x}_k + \widehat{s}_k$
$\widehat{x}_k - \dfrac{\widehat{g}_k}{\sigma_k} = c_k$
$\widehat{x}_k$
$\widehat{g}_k$
$\widetilde{g}_k$
$\widetilde{g}_k^a$

Figure 1: Difference between the exact candidate c_k and the computed candidate $\widehat{c}_k$.

In (27), following the remarks of Section 3.2, the index of the FP format of $\widehat{c}_k$ is π_k^g. We implement a mechanism that casts $\widehat{c}_k$ into a FP format of index π_k^c, which can be chosen freely. This enables to lower the precision of the FP format of $\widehat{c}_k$, and therefore those for $\widehat{f}$ and $\widehat{g}$. Indeed, $\widehat{f}$ and $\widehat{g}$ cannot be evaluated at $\widehat{x}_k$ or $\widehat{c}_k$ in a FP format index lower than π_k^x or π_k^c, respectively (see forbidden evaluations in Section 3.2), and $\widehat{x}_{k+1} = \widehat{c}_k$ if iteration k is successful. This mechanism is necessary to allow the precision for the evaluation of $\widehat{f}$ and $\widehat{g}$ to decrease over the iterations. The extra cast of $\widehat{c}_k$ is not represented in Figure 1 for simplicity.

Casting $\widehat{c}_k$ into $\pi_k^c < \pi_k^g$ induces the additional rounding error

$$\text{fl}(\widehat{c}_k, \pi_k^c) = \widehat{c}_k(1+\delta_k^c) = (\widehat{x}_k + \widehat{s}_k)(1+\delta_k^+)(1+\delta_k^c) = (\widehat{x}_k + \widehat{s}_k)(1+\delta_k'), \tag{28}$$

with $|\delta_k^c| \leq u_k^c$ and,

$$\delta_k' = \delta_k^+ + \delta_k^c + \delta_k^+\delta_k^c. \tag{29}$$

In the following, we consider δ_k' as a single virtual rounding error that is bounded by,

$$u_k' = u_k^g + u_k^c + u_k^g u_k^c. \tag{30}$$

Note that if $\pi_k^c \geq \pi_k^g$, no rounding error due to the casting of $\widehat{c}_k$ occurs and one simply has $\delta_k' = \delta_k^+$ and $u_k' = u_k^g$.

From (28), we define the *actual step* as,

$$\widetilde{s}_k^a := \widehat{s}_k(1+\delta_k') + \widehat{x}_k\delta_k', \tag{31}$$

which satisfies $\mathrm{fl}(\widehat{c}_k, \pi_k^c) = \widehat{x}_k + \widetilde{s}_k^a$. The *actual search direction* is

$$-\widetilde{g}_k^a := \widehat{\sigma}_k\widetilde{s}_k^a. \tag{32}$$

Since we have no relationship between $\widehat{g}_k$ and $\widehat{x}_k$, we do not have a relationship between $\widetilde{s}_k^a$ and $\widehat{g}_k$ as we did in the previous section with $\widehat{g}_k$ and $\widetilde{g}_k$. In order to deal with the inexact candidate computation, we introduce ϕ_k as,

$$\frac{\|\widehat{x}_k\|}{\|\widehat{s}_k\|} \leq \phi_k. \tag{33}$$

Lemma 3 provides an upper bound on $\|\widetilde{s}_k^a\|$ in terms of $\|\widehat{s}_k\|$ by taking advantage of ϕ_k, enabling us to get rid of $\widehat{x}_k$ in the expression that will prove useful later for the convergence analysis of MPR2.

Lemma 3. For all $k > 0$,

$$\|\widetilde{s}_k^a\| \leq \|\widehat{s}_k\|(1+\lambda_k), \tag{34}$$

with

$$\lambda_k := u_k'(\phi_k + 1), \tag{35}$$

where u_k' is defined in (30).

Proof. We combine (31), the triangle inequality and (33), and obtain

$$\begin{aligned}
\|\widetilde{s}_k^a\| &= \|\widehat{s}_k(1+\delta_k') + \widehat{x}_k\delta_k'\| \\
&\leq \|\widehat{s}_k\|(1+\delta_k') + \|\widehat{x}_k\|\delta_k' \\
&\leq \|\widehat{s}_k\|(1+u_k') + \|\widehat{x}_k\|u_k' \\
&\leq \|\widehat{s}_k\|(1+u_k') + \phi_k\|\widehat{s}_k\|u_k' \\
&= \|\widehat{s}_k\|(1+\lambda_k).
\end{aligned}$$

□

If ϕ_k is large, $\widehat{x}_k\delta_k'$ is not negligible compared to $\widehat{s}_k$ and the difference between $-\widehat{g}_k$ and the actual search direction can be large (see Figure 1). If ϕ_k is too large, convergence of the finite precision quadratic regularization algorithm cannot be ensured because there is no guarantee that the actual search direction $-\widetilde{g}_k^a$ is a descent direction.

3.5 Inexact norm computation

In Algorithm 2, it is necessary to compute $\|\widehat{g}_k\|$, $\|s_k\|$ and $\|\widehat{x}_k\|$. Due to rounding errors, norm computations are inexact. Lemma 4 states an upper bound on the error on norm computation, assuming that the computation of the norm is done in the same FP format as the argument.

Lemma 4. Let Assumption 3, be satisfied. The error on the computation of the norm of $\widehat{x} \in \mathbb{F}^n$ in finite precision is bounded as

$$|\,\mathrm{fl}(\|\widehat{x}\|) - \|\widehat{x}\|\,| \leq \mathrm{fl}(\|\widehat{x}\|)\beta_{n+2}(u_k^x), \tag{36}$$

where $\beta_{n+2}(u_k^x) := \max\left(|\sqrt{1-\xi_{n+2}(u_k^x)} - 1|, |\sqrt{1+\xi_{n+2}(u_k^x)} - 1|\right)$, and $\beta_{n+2}(u_k^x) \leq 1$.

Proof. The error on the dot product computation can be modeled as [19],

$$\text{fl}(\widehat{x}^T\widehat{x}) = \widehat{x}_1^2(1+\delta)^n + \sum_{i=2}^{n} \widehat{x}_i^2(1+\delta)^{n+2-i}. \tag{37}$$

As a consequence, there exists ϑ_n such that

$$\text{fl}(\|\widehat{x}\|^2) = \|\widehat{x}\|^2(1+\vartheta_n), \quad |\vartheta_n| \le \xi_n(u_k^g). \tag{38}$$

Since the square root computation is exactly rounded [2], $\forall \hat{y} \in \mathbb{F}$, $\text{fl}(\sqrt{\hat{y}}) = \sqrt{\hat{y}}(1+\delta)$ and it follows that

$$\begin{aligned} \text{fl}(\sqrt{\text{fl}(\|\widehat{x}\|^2)}) &= \sqrt{\text{fl}(\|\widehat{x}\|^2)}(1+\delta) \\ &= \sqrt{\|\widehat{x}\|^2(1+\vartheta_n)}(1+\delta) \\ &= \|\widehat{x}\|\sqrt{(1+\vartheta_n)(1+\delta)^2}. \end{aligned} \tag{39}$$

Furthermore, because $\xi_n = nu$,

$$\begin{aligned} (1+\vartheta_n)(1+\delta)^2 &\le (1+\xi_n(u_k^x))(1+\xi_2(u_k^x)) \\ &= 1 + (n+2)u_k^x + 2n(u_k^x)^2 \\ &= 1 + \xi_{n+2}(u_k^g) + 2n(u_k^x)^2. \end{aligned} \tag{40}$$

Since $u \ll 1$, $2nu^2 = o((n+2)u)$ and we ignore this last term. As a consequence, there exists ϑ_{n+2} such that,

$$\text{fl}(\sqrt{\text{fl}(\|\widehat{x}\|^2)}) = \|\widehat{x}\|\sqrt{1+\vartheta_{n+2}}. \tag{41}$$

Due to Assumption 3, $\text{fl}(\sqrt{\text{fl}(\|\widehat{x}\|^2)}) \ge 0$. Therefore,

$$|\,\text{fl}(\|\widehat{x}\|) - \|\widehat{x}\|\,| = \text{fl}(\|\widehat{x}\|)\,|1 - \sqrt{1+\vartheta_{n+2}}| \tag{42}$$

and since $|1-\sqrt{1+\vartheta_{n+2}}| \le \max\left(|1-\sqrt{1-\xi_{n+2}(u_k^x)}|, |1-\sqrt{1+\xi_{n+2}(u_k^x)}|\right)$, we finally have,

$$|\,\text{fl}(\|\widehat{x}\|) - \|\widehat{x}\|\,| \le \text{fl}(\|\widehat{x}\|)\,\max\left(|1-\sqrt{1-\xi_{n+2}(u_k^x)}|, |1-\sqrt{1+\xi_{n+2}(u_k^x)}|\right). \qquad \square$$

4 Multi-precision algorithm

4.1 Algorithm description

Algorithm 2 is an adaptation of Algorithm 1 that implements strategies to deal with the errors due to finite precision computations details in Section 3.

At the initialization step, there is no modification on how the parameters should be chosen. The only change is that the initial FP format π_0^c of the candidate must be chosen.

At Step 1, the factor $1/(1+\beta_{n+2}(u_k^g))$ is added in Condition (44) to take into account the error due to norm computation of the approximated gradient $\widehat{g}_k$ (see Lemma 4). The new condition ensures $\|\nabla f(x_k)\| \le \epsilon$ despite the gradient evaluation and norm computation errors,

$$\begin{aligned} \|\nabla f(\widehat{x}_k)\| &\le \|\nabla f(\widehat{x}_k) - \widehat{g}_k\| + \|\widehat{g}_k\| \\ &\le (1+\omega_g(\widehat{x}_k))\|\widehat{g}_k\| \\ &\le (1+\omega_g(\widehat{x}_k))\,(|\,\|\widehat{g}_k\| - \text{fl}(\|\widehat{g}_k\|)| + \text{fl}(\|\widehat{g}_k\|)) \\ &\le (1+\omega_g(\widehat{x}_k))(1+\beta_{n+2}(u_k^g))\,\text{fl}(\|\widehat{g}_k\|). \end{aligned} \tag{49}$$

Step 2 consists in evaluating the gradient and computing the step with appropriate accuracy, and is given by Algorithm 3. The convergence condition $\omega_g(x_k) \le \kappa_\mu$ in Algorithm 1 becomes $\mu_k \le \kappa_\mu$. μ_k accounts for the error on gradient evaluation, and the errors due to finite precision computations. The

Algorithm 2 MPR2 : Multi-precision quadratic regularization algorithm

Step 0: Initialization: Initial point $\widehat{x}_0$, initial value $\widehat{\sigma}_0$, minimal value $\widehat{\sigma}_{\min}$ for $\widehat{\sigma}$, final gradient accuracy ϵ, formula for ξ_n. Constant values η_0, η_1, η_2, γ_1, γ_2, γ_3, κ_μ such that,

$$0 < \eta_1 \leq \eta_2 < 1 \quad 0 < \gamma_1 < 1 < \gamma_2 \leq \gamma_3, \quad \eta_0 < \tfrac{1}{2}\eta_1 \quad \eta_0 + \frac{\kappa_\mu}{2} \leq \tfrac{1}{2}(1-\eta_2) \tag{43}$$

Set $k=0$, compute $\widehat{f}_0 = \widehat{f}(x_0)$, select π_0^g and π_0^c.
Step 1: Check for termination: If $k=0$ or $\rho_{k-1} \geq \eta_1$ (previous iteration successful), compute $\widehat{g}_k = g(\widehat{x}_k)$. Compute $\widehat{\|\widehat{g}_k\|} = fl(\|\widehat{g}_k\|)$. Terminate if

$$\widehat{\|\widehat{g}_k\|} \leq \frac{1}{1+\beta_{n+2}(u_k^g)} \frac{\epsilon}{1+\omega_g(\widehat{x}_k)} \tag{44}$$

Step 2: Step calculation (Algorithm 3)
Step 3: Evaluate the objective function: Choose $\pi_k^{f^+} \geq \pi_k^c$ such that

$$\omega_f(\widehat{c}_k) \leq \eta_0 \widehat{\Delta T}_k. \tag{45}$$

Compute $f_k^+ = \widehat{f}(\widehat{c}_k)$. Terminates if no such precision is available.
If $\omega_f(\widehat{x}_k) > \eta_0 \widehat{\Delta T}_k$, choose $\pi_k^f > \pi_{k-1}^{f^+}$ as new format index for $\widehat{f}$ such that

$$\omega_f(\widehat{x}_k) \leq \eta_0 \widehat{\Delta T}_k. \tag{46}$$

Re-compute $f_k = \widehat{f}(\widehat{x}_k)$ with π_k^f. Terminates if no such precision is available.

Step 4: Acceptance of the trial point: Define the ratio

$$\rho_k = \frac{\widehat{f}_k - \widehat{f}_k^+}{\widehat{\Delta T}_k}. \tag{47}$$

If $\rho_k \geq \eta_1$, then $\widehat{x}_{k+1} = \widehat{c}_k$, $\widehat{f}_{k+1} = \widehat{f}_k^+$. Otherwise, set $\widehat{x}_{k+1} = \widehat{x}_k$, $\widehat{f}_{k+1} = \widehat{f}_k$.
Select $\pi_{k+1}^g \geq \pi_k^x$, select π_{k+1}^c.
Step 5: Regularization parameter update:

$$\widehat{\sigma}_{k+1} \in \begin{cases} [\max(\widehat{\sigma}_{\min}, \gamma_1\widehat{\sigma}_k), \widehat{\sigma}_k] & \text{if } \rho_k \geq \eta_2 \\ [\widehat{\sigma}_k, \gamma_2\widehat{\sigma}_k] & \text{if } \rho_k \in [\eta_1, \eta_2) \\ [\gamma_2\widehat{\sigma}_k, \gamma_3\widehat{\sigma}_k] & \text{if } \rho_k < \eta_1 \end{cases} \tag{48}$$

$k = k+1$, go to Step 1.

coefficient ϕ_k takes into account the norm computation errors that occur when computing $\widehat{\phi}_k$, and is a guaranteed upper bound on the ratio $\|\widehat{x}_k\|/\|\widehat{s}_k\|$. Indeed, due to the norm computation error (see Subsection Section 3.5),

$$\frac{\|\widehat{x}_k\|}{\|\widehat{s}_k\|} \leq \frac{\text{fl}(\|\widehat{x}_k\|)}{\text{fl}(\|\widehat{s}_k\|)} \frac{1+\beta_{n+2}(u_k^x)}{1-\beta_{n+2}(u_k^g)} \tag{50}$$

Since the format of $\text{fl}(\|\widehat{x}_x\|)$ is π_k^x and the one of $\text{fl}(\|\widehat{s}_x\|)$ is π_k^g, and $\pi_k^g \geq \pi_k^x$ (see introductory remarks), it follows that,

$$\begin{aligned} \frac{\|\widehat{x}_k\|}{\|\widehat{s}_k\|} &\leq \text{fl}\left(\frac{\text{fl}(\|\widehat{x}_k\|)}{\text{fl}(\|\widehat{s}_k\|)}\right)(1+u_k^g)\frac{1+\beta_{n+2}(u_k^x)}{1-\beta_{n+2}(u_k^g)} \\ &= \widehat{\phi}_k \frac{1+\beta_{n+2}(u_k^x)}{1-\beta_{n+2}(u_k^g)}(1+u_k^g) = \phi_k. \end{aligned} \tag{51}$$

The coefficient u'_k corresponds to the virtual rounding errors accounting for inexact computation of $\widehat{c}_k$ and the casting of $\widehat{c}_k$ into a lower FP format. By lowering the FP format precision, one enables to evaluate the objective at iteration k or the gradient at iteration $k+1$ (if k is successful: $\widehat{x}_{k+1} = \widehat{c}_k$) with a lower FP format (see details in Subsection Section 3.4).

The coefficient μ_k in Algorithm 3 aggregates most of the errors due to finite precision computation listed in Section 3, and is the counterpart of ω_g in Algorithm 1. The coefficient ξ_{n+1} and α_{n+1} account for the inexact approximated model decrease (see Subsection 3.3). The coefficient λ_k, given in (35), includes ϕ_k the upper bound on $\|\widehat{x}_k\|/\|\widehat{s}_k\|$ and takes into account the inexact candidate computation explained in Section 3.4. The denominator $1-u_k^g$ account for the difference in norm between $\widehat{g}(x_k)$ and

```
Algorithm 3 Step 2: Step calculation
  Compute $\widehat{s}_k = \mathrm{fl}(\widehat{g}_k/\widehat{\sigma}_k)$.
  Compute $\widehat{\phi}_k = \mathrm{fl}(\|\widehat{x}_k\|/\|\widehat{s}_k\|)$.
  Define $\phi_k = \widehat{\phi}_k \frac{1+\beta_{n+2}(u_k^x)}{1-\beta_{n+2}(u_k^g)}(1+u_k^g)$.
  Define $u'_k = u_k^g + u_k^c + u_k^g u_k^c$.
  Define
      $\mu_k = \frac{\alpha_{n+1}(u_k^g)\omega_g(\widehat{x}_k)(1+\lambda_k) + \alpha_{n+1}(u_k^g)\lambda_k + u_k^g + \xi_{n+1}(u_k^g)\alpha_{n+1}(u_k^g)}{1-u_k^g}.$    (52)
  If
      $\mu_k \le \kappa_\mu$    (53)
  is not satisfied: increase $\pi_k^g$ and go to Step 1 (at "compute $\widehat{g}_k$" step) or increase $\pi_k^c$ and redefine $u'_k$ and $\mu_k$.
  Terminate if no available FP formats $(\pi_k^c, \pi_k^g)$ ensure the inequality.
  Compute $\widehat{c}_k = \mathrm{fl}(\widehat{x}_k + \widehat{s}_k)$ and cast $\widehat{c}_k$ into the format of index $\pi_k^c$.
  Compute approximated Taylor series decrease $\widehat{\Delta T}_k = \mathrm{fl}(\widehat{g}_k^T \widehat{s}_k)$.
```

$\widetilde{g}_k$ (see Lemma 6). Note that if we assume that the computations are done with infinite precision, *i.e.* $u'_k = u_k^g = 0$, one has $\mu_k = \omega(\widehat{x}_k)$ and retrieve the condition in Algorithm 1. As such, all the errors due to finite precision computations aggregated in μ_k can simply be interpreted as noise on the gradient degrading the search direction. The condition $\mu_k \le \kappa_\mu$ ensures the convergence of Algorithm 2. If $\mu_k > \kappa_\mu$, one can lower μ_k by decreasing π_k^c, and by extension lower u'_k and therefore μ_k. The second option is to lower π_k^g and to recompute the gradient at Step 1 with a higher precision in order to lower $\omega_g(\widehat{x}_k)$.

Step 3 remains similar in Algorithm 1. A stopping criterion is added if the needed objective function evaluation accuracy cannot be achieved with the provided FP formats. This criterion is needed to ensure the convergence.

At Step 4, the only modification is the selection of the FP format for the gradient evaluation at the next step, and the FP format of the candidate. Step 5 remains the same, since we suppose that no rounding errors occur when computing $\widehat{\sigma}_{k+1}$.

4.2 Note on $\widehat{\rho}_k$

As indicated in Section 3.2, we assume that ρ_k is defined in Algorithm 2, that is, ρ_k is computed with infinite precision. This assumption is reasonable if a FP format with a precision higher than the formats used in Algorithm 2 is used to compute ρ_k. However, if such precision is not available, it is critical to take into account the rounding errors occurring when computing ρ_k. Indeed, if $\widehat{f}_k$ and $\widehat{f}_k^+$ are large and close to each other, a "catastrophic cancellation" might occur at the numerator, and the value computed for ρ_k is not relevant.

The rounding error occurring is such that,

$$\widehat{\rho_k} = \mathrm{fl}\left(\frac{\widehat{f}_k^+ - \widehat{f}_k^+}{\widehat{\Delta T}_k}\right) = \frac{\widehat{f}_k^+(1+\delta)^2 - \widehat{f}_k^+(1+\delta)^2}{\widehat{\Delta T}_k}, \tag{54}$$

with delta bounded as $|\delta| \le u_k^\rho = \min(u_k^f, u_{k-1}^f, u_k^g)$. The error between the exact value of f and $\widehat{f}_k(1+\delta)^2$ can be bounded as,

$$|f(\widehat{x}_k) - \widehat{f}_k(1+\theta_2)| \le |f(\widehat{x}_k) - \widehat{f}_k| + |\widehat{f}_k - \widehat{f}_k(1+\theta_2)| \le \omega_f(\widehat{x}_k) + \widehat{f}_k\xi_2(u_k^\rho), \tag{55}$$

and similarly,

$$|f(\widehat{c}_k) - \widehat{f}_k^+(1+\delta)^2| \le \omega_f(\widehat{c}_k) + |\widehat{f}_k^+|\xi_2(u_k^\rho). \tag{56}$$

Therefore, the inexact computation of ρ_k can be taken into account by adapting the conditions at Step 3 as,

$$\omega_f(\widehat{x}_k) + \xi_2(u_k^\rho)|\widehat{f}_k| \le \eta_0\widehat{\Delta T}_k, \quad \omega_f(\widehat{c}_k) + \xi_2(u_k^\rho)|\widehat{f}_k^+| \le \eta_0\widehat{\Delta T}_k. \tag{57}$$

4.3 Complexity analysis and level of optimality

In order to analyse the complexity, we assume that f is lower bounded.

Assumption 4. There exists f_{low} such that,

$$\forall x \in \mathbb{R}^n,\ f(x) \geq f_{low}. \tag{58}$$

We also assume that no underflow or overflow occur during the execution. Our implementation of Algorithm 2, briefly detailed in Section 5, includes many safeguards to avoid underflow and overflow.

Assumption 5. There is no underflow or overflow occurring during the execution Algorithm 2.

The complexity of Algorithm 2 is given by Theorem 1, whose proof is in Appendix A. We obtain the classical worst-case complexity of $\mathcal{O}(\epsilon^{-2})$. Note that the rounding errors due to finite precision are taken into account in κ_s, and indirectly in κ_μ as an upper bound on μ_k.

Theorem 1. Let Assumptions 1 to 5 be satisfied. If the stopping conditions at Steps 2 and 3 are not met, Algorithm 2 needs at most

$$\epsilon^{-2}\kappa_s(f(x_0) - f_{low}), \tag{59}$$

successful iterations, with

$$\kappa_s = \left(\frac{1+\beta_{n+2}(u_{max})}{1-\beta_{n+2}(u_{max})}\frac{1+\kappa_\mu}{1-u_{max}}\right)^2 \frac{\sigma_{\max}}{(\eta_1 - 2\eta_0)(1-\xi_{n+1}(u_{max}))}, \tag{60}$$

and at most

$$\epsilon^{-2}\kappa_s(f(x_0) - f_{low})\left(1 + \frac{|\log(\gamma_1)|}{\log(\gamma_2)}\right) + \frac{1}{\log(\gamma_2)}\log\left(\frac{\sigma_{\max}}{\widehat{\sigma}_0}\right) \tag{61}$$

iterations to provide an iterate $\widehat{x}_k$ such that $\|\nabla f(\widehat{x}_k)\| \leq \epsilon$, with $\sigma_{\max}$ given in Lemma 8.

Note that Theorem 1 is limited to the case where sufficient precision is available so that conditions (45), (46) and (53) can always be satisfied until the given first order criterion is reached. Such precision might not be available, which raises the question of what level of optimality can be achieved. Doing so requires to make some assumptions on the absolute evaluation errors on objective and gradient [7].

Assumption 6. There exist absolute bounds v_f on objective and v_g on gradient evaluation errors : $\forall k > 0$,

$$|f(\widehat{x}_k) - \widehat{f}(\widehat{x}_k)| \leq v_f, \quad \|\nabla f(\widehat{x}_k) - \widehat{g}(\widehat{x}_k)\| \leq \|\widehat{g}(\widehat{x}_k)\|\omega_g(\widehat{x}_k) \leq v_g. \tag{62}$$

Theorem 2 provides upper bound on the optimality criterion, based on the results provided by Theorem 1.

Theorem 2. Let Assumptions 1 to 3, 5 and 6 be satisfied. If Algorithm 2 stops because (53) is not satisfied, then

$$v_g\left(1 + \frac{\alpha_{n+1}(u_k^g)(1+\lambda_k)}{\mu_k(1-u_k^g) - \alpha_{n+1}(u_k^g)\lambda_k - u_k^g + \xi_{n+1}(u_k^g)\alpha_{n+1}(u_k^g)}\right) > \|\nabla(\widehat{x}_k)\|. \tag{63}$$

If Algorithm 2 stops because either (45) or (46) is not satisfied, then

$$\frac{1+\kappa_\mu}{(1-u_k^g)}\sqrt{\frac{v_f\eta_0}{\sigma_{\max}(1-\xi_{n+1}(u_k^g))}} > \|\nabla f(\widehat{x}_k)\|. \tag{64}$$

Proof. Consider first the case where Algorithm 2 stops because (53) fails. In this case,

$$\omega_g(\widehat{x}_k) > \frac{\mu_k(1-u_k^g) - \alpha_{n+1}(u_k^g)\lambda_k - u_k^g + \xi_{n+1}(u_k^g)\alpha_{n+1}(u_k^g)}{\alpha_{n+1}(u_k^g)(1+\lambda_k)}. \tag{65}$$

Multiplying by $\|\widehat{g}(\widehat{x}_k)\|$ both sides, and using Assumption 6, we obtain

$$v_g > \frac{\mu_k(1-u_k^g) - \alpha_{n+1}(u_k^g)\lambda_k - u_k^g + \xi_{n+1}(u_k^g)\alpha_{n+1}(u_k^g)}{\alpha_{n+1}(u_k^g)(1+\lambda_k)}\|\widehat{g}(\widehat{x}_k)\|. \tag{66}$$

Assumption 6 with triangle inequality further yields $\|\widehat{g}(\widehat{x}_k)\| \geq \|\nabla f(\widehat{x}_k)\| - v_g$. Injecting this last inequality into (66) yields the first statement of the theorem.

Consider now the where Algorithm 2 stops at iteration k because (46) fails. The same reasoning applies if (45) fails, it suffices to replace x_k by c_k in the rest of the proof. If (46) fails, we have,

$$|f(x_k) - \widehat{f}(x_k)| > \eta_0\widehat{\Delta T}_k \implies v_f > \eta_0\widehat{\Delta T}_k. \tag{67}$$

Lemma 2 yields

$$v_f > \eta_0\frac{\|\widetilde{g}_k\|^2}{\widehat{\sigma}_k}(1+\vartheta_{n+1}), \tag{68}$$

and with Lemma 8, which provides an upper bound on $\widehat{\sigma}_k$, and since $|\vartheta_{n+1}| \leq \xi_{n+1}$,

$$v_f > \eta_0\frac{\|\widetilde{g}_k\|^2}{\sigma_{\max}}(1-\xi_{n+1}(u_k^g)) \implies \sqrt{\frac{v_f\eta_0}{\sigma_{\max}(1-\xi_{n+1}(u_k^g))}} > \|\widetilde{g}_k\|. \tag{69}$$

Lemma 1 yields

$$\sqrt{\frac{v_f\eta_0}{\sigma_{\max}(1-\xi_{n+1}(u_k^g))}} > \|\widehat{g}_k\|(1-u_k^g), \tag{70}$$

and with the error bound on gradient evaluation (2)

$$\sqrt{\frac{v_f\eta_0}{\sigma_{\max}(1-\xi_{n+1}(u_k^g))}} > \|\nabla f(\widehat{x}_k)\|\frac{(1-u_k^g)}{1+\omega_g(\widehat{x}_k)}. \tag{71}$$

Noting that κ_μ is an upper bound on $\omega_g(\widehat{x}_k)$, since in the present case (53) holds, and rearranging the inequality, we obtain the second statement. □

5 Numerical tests

MPR2 is implemented in the Julia MultiPrecisionR2.jl package [21] as part of JuliaSmoothOptimizer (JSO) organisation [22]. This implementation is robust to many pitfalls generated by the use of multiple FP formats and in particular low-precision, low-range FP formats (underflow, overflow, memory allocation for x, c and s, etc.). The detailed implementation of MPR2 is out of the scope of this paper, and is not described here. The reader can refer to the MultiPrecisonR2.jl package for implementation details.

The numerical results presented below can be reproduced from the dedicated file of the package.[1] The results are obtained by running MPR2 on a standard laptop that does not support half precision natively. Therefore, the computational effort savings presented in Section 5.3 are estimations. The numerical results are obtained over a set of 164 unconstrained problems implemented in OptimizationProblems.jl [22] package, with dimensions ranging from 1 to 100 variables. The FP formats employed are half, single and double. Assumption 3 is satisfied for all the problems, with $u_{\max} = 1/2048$ for half precision. The tolerance on the first-order criterion is set to $\epsilon = \epsilon_M(\texttt{Float64})^{1/4} \approx 1.2e^{-4}$. The maximum number of iteration of MPR2 is set to 20,000, and the maximum execution time is set to 15 min on a standard laptop. All the defined values are computed with quadruple precision.

[1]https://github.com/JuliaSmoothOptimizers/MultiPrecisionR2/blob/mpr2_paper/docs/src/mpr2_numerical_test.md

5.1 Precision selection strategy

The strategy implemented for selecting the evaluation FP formats and π_c are the following.

At Step 2 of Algorithm 2, the index π_k^g is selected as $\pi_k^g = \pi_k^c$. If $\mu_k > \kappa_\mu$, one can update either π_k^c to decrease u_k' or π_k^g to decrease $\omega_g(\widehat{x}_k)$ on which μ_k depends. The update rule is

$$\begin{cases} \pi_k^c = \pi_k^c + 1 & \text{if } \pi_k^c < \pi_k^g, \\ \pi_k^g = \pi_k^g + 1 & \text{otherwise,} \end{cases} \tag{72}$$

until $\mu_k < \kappa_\mu$.

At Step 3 of Algorithm 2, the index $\pi_k^{f^+}$ for objective function evaluation at $\widehat{c}_k$ is chosen as the index of the least accurate FP format whose unit roundoff u_k^f satisfies

$$\omega_f(\widehat{x}_k)\frac{\widehat{f}(\widehat{x}_k) - \widehat{\Delta T}_k}{\widehat{f}(\widehat{x}_k)}\frac{u_f^k}{u_f^{k-1}} \le \eta_0 \widehat{\Delta T}_k. \tag{73}$$

The left-hand side is a prediction of $\omega_f(\widehat{c}_k)$ based on the assumptions that $\widehat{f}(\widehat{c}_k) \approx \widehat{f}(\widehat{x}_k) - \widehat{\Delta T}_k$ and that ω_f grows linearly with the unit roundoff and is proportional to f. These two assumptions are consistent with our observations over the test set. If the convergence condition (45) is not satisfied with the above selection strategy, $\pi_k^{f^+}$ is increased by 1 until (45) is ensured.

If it is necessary to re-evaluate $\widehat{f}(\widehat{x}_k)$ to decrease $\omega_f(\widehat{x}_k)$ and enforce satisfy (46), the re-evaluation index π_k^f is selected as the index of least accurate FP format whose unit roundoff $u_k^{f^-}$ satisfies

$$\omega_f(\widehat{x}_k)\frac{u_k^{f^-}}{u_{k-1}^f} \le \eta_0 \widehat{\Delta T}_k. \tag{74}$$

The left-hand side is a prediction of $\omega_f(\widehat{x}_k)$ evaluated with $u_k^{f^-}$ based on the assumption that $\omega_f(\widehat{x}_k)$ is linearly dependent on the unit roundoff. This assumption is corroborated by our observations over the test set of problems. If (46) is still not satisfied with the above selection strategy, π_k^f is increased by 1 until (46) is satisfied.

At Step 4 of Algorithm 2, the index π_k^c is chosen as $\pi_c^c = \max(1, \pi_k^{f^+} - 1)$. This enables to lower the precision of the candidate and, by extension, to allow lower precision evaluation of the objective function and the gradient (see details in Section 3.4).

5.2 Guaranteed implementation

One challenge with Algorithm 2 is that the error bounds ω_f and ω_g on objective and gradient must be available. In the general case, obtaining analytical expression of these bounds might not be possible. That is why the standard, or guaranteed, implementation of Algorithm 2 relies on interval arithmetic provided by the JuliaIntervals.jl library [23] to compute the error bounds ω_f and ω_g. Interval arithmetic enables to compute guaranteed bounds on the exact values (which cannot be computed in finite-precision) of the objective and the gradient, by accounting for the worst possible case of rounding errors accumulating during the evaluations. From these bounds, it is possible to derive ω_f and ω_g. The results presented below are obtained with the guaranteed implementation of Algorithm 2, using interval arithmetic.

Table 5 lists the values of the parameters of MPR2. Over the test problems, the number of evaluations and *successful* evaluations are computed. An evaluation is successful if there is no need to re-evaluate the objective or the gradient with a higher precision FP format, and gives a metric to evaluate the precision selection strategy detailed above. Table 6 displays the number of problems for which Algorithm 2 reaches a first order critical point (FO), reaches the maximum number of iteration

Table 5: MPR2 parameters values

γ_1	γ_2	γ_3	η_0	η_1	η_2	κ_μ
1/2	1	2	0.05	0.1	0.7	0.2

or allowed time (MIT), and for which the convergence conditions at Step 2 and 3 are not ensured (F). Note that 18 problems out of the 164 were ignored due to operators not supported by JuliaInterval.jl or other issues linked to this package. Table 6 also displays the percentage of evaluations performed in a given FP format for the gradient and the objective, and the success rate for each of the FP. The success rate is the ratio between the number of successful evaluations and the total number of evaluations in a given format.

Table 6: Percentage of total evaluations of objective and gradient for the given FP formats

FO	MIT	F	% of obj. eval. (success rate in %)			% of grad. eval. (success rate in %)		
			16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
137	20	7	0.2 (53.6)	33.8 (93.4)	66 (100)	1.3 (84.8)	78 (100)	20.7 (100)

The results in Table 6 show that Float64 does not offer sufficient precision for objective or gradient evaluations to satisfy the convergence conditions at Step 2 and 3. Note that objective and gradient evaluations reach 100 % success rate with 64 bits. Indeed, MPR2 stops when this format does not offer enough precision (or the first order criterion is reached), and, as a consequence, every evaluations with 64 bits is successful until MPR2 stops.

Table 6 also shows that the objective function is evaluated with more precision than the gradient. The reason is that the error bounds provided by interval arithmetic are, to a certain extent, proportional to the exact value. Since the gradient tends to zero over the iterations, ω_g decreases accordingly and 32 bits offers in most case sufficient precision. However, the value of the objective function at a critical point is not 0 in the general case, and the convergence conditions at Step 3 requires the error on the objective to be lower than $\eta_0\widehat{\Delta T}_k$ which is proportional to the square of the norm of the gradient. As a consequence, the highest precision is employed to evaluate the objective, especially for the last iterations where $\widehat{\Delta T}_k$ is small.

16 bits evaluations are marginal. This is due to the fact that overflow often occurs, or the precision is not sufficient for objective evaluation which can happens after only few iterations if the model decrease gets too small (see conditions (45) and (46)). For gradient evaluation, the unit roundoff of 16 bits (1/2048) limits its use to value of ϕ_k lower than 2048 (see μ_k formula (52)). However, this value of ϕ_k can be reached in few iterations, after which only small step sizes (compare to the incumbent $\widehat{x}_k$) can ensure sufficient objective decrease.

5.3 Efficient implementation

In practice, interval evaluations are highly time consuming compared with classical evaluation, and are prohibitive for many problems. Furthermore, it is unlikely that the error bounds computed with interval evaluation, although guaranteed, are reliable estimates of the actual evaluation errors since they account for the worst-case rounding error accumulation. That is why we propose `r-`MPR2 , a relaxed and efficient version of Algorithm 2 that aims to increase the use of low-precision FP formats and reach convergence to a first-order critical point based on simple evaluation error models. `r-`MPR2 has the following features:

- the objective and gradient errors are estimated as

$$\omega_f(\widehat{c}_k) = |\widehat{f}(\widehat{c}_k)|2u, \quad \omega_g(\widehat{x}_k) = 2u, \tag{75}$$

  with u the corresponding unit roundoff associated to $\pi_k^{f^+}$ or π_k^g,

- $\mathtt{r}$-MPR2 does not stop if the conditions on μ_k and $\omega_f(\widehat{x}_k)$, $\omega_f(\widehat{c}_k)$, are not met with the highest precision FP format.

We also propose to decrease μ by a factor $a \in]0,1]$ to relax the condition (53) into $a\mu \le \kappa_\mu$ to assess whether even more computational effort can be saved. We compare `r`-MPR2 run with Float16, Float32 and Float64 with R2, the generic single precision version of MPR2 that does not take any finite-precision error into account, implemented in the JSOSolvers.jl package [1] run with Float64. Both `r`-MPR2 and R2 use the parameters given in Table 5. For the sake of comparison, the stopping criterion for `r`-MPR2 is set to $\|\widehat{g_k}\| \le \epsilon$, which is the one used in R2. Table 7 (resp. Table 8) displays the number of problem solved by `r`-MPR2 and R2 as well as the percentage of total objective function (resp. gradient) evaluations over the whole set of problems in each FP format with the associated success rate, and the estimated computational effort ratio between `r`-MPR2 and R2 for time and energy. The lower the ratio, the higher the effort savings with `r`-MPR2 . The time and energy efforts are estimated from the following observation: dividing the computation precision by two divides the computation time by two and the energy consumption by four [15], *e.g.*, evaluating the objective function in Float16 is four times faster and requires sixteen times less energy than evaluating it in Float64.

Table 7: Estimated computational effort saved by `r`-MPR2 compared with R2 for objective evaluation.

Algo	a	% of obj. eval. (success rate in %) 16 bits	32 bits	64 bits	time ratio	energy ratio	pb solved(/164)
R2	-	-	-	1.0(100)	1.0	1.0	154
`r`-MPR2	1.0	0.5(91.8)	45.8(>99.9)	53.6(100)	0.704	0.598	153
`r`-MPR2	0.1	0.8(92)	51.9(>99.9)	47.3(100)	0.636	0.523	143
`r`-MPR2	0.01	2.8(99)	58.2(>99.9)	38(100)	0.94	0.73	132

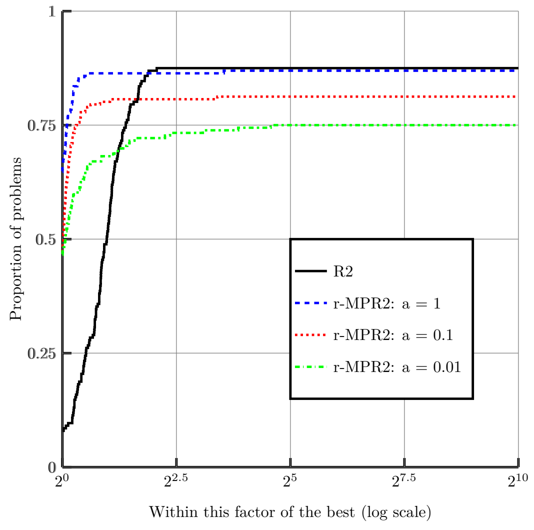
Table 8: Estimated computational effort saved by `r`-MPR2 compared with R2 for gradient evaluation.

Algo	a	% of grad. eval. (success rate in %) 16 bits	32 bits	64 bits	time ratio	energy ratio	pb solved(/164)
R2	-	-	-	1.0(100)	1.0	1.0	154
`r`-MPR2	1.0	1.1(89.7)	74.4(>99.9)	24.4(100)	0.598	0.417	153
`r`-MPR2	0.1	10.5(93.1)	71.9(>99.9)	17.6(100)	0.513	0.331	143
`r`-MPR2	0.01	14(92.8)	70.3(>99.9)	15(100)	0.65	0.405	132

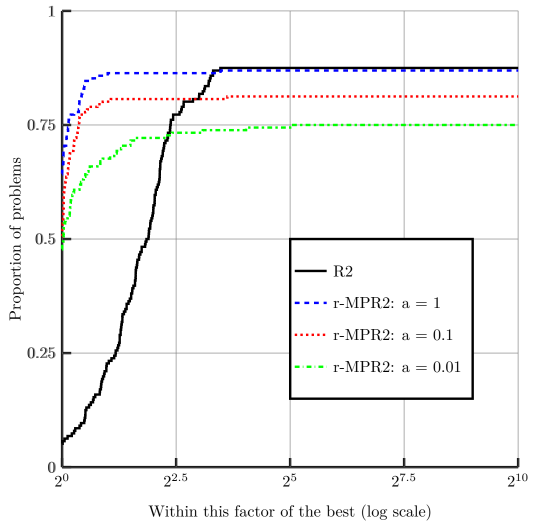
From Table 7 and Table 8, it appears that `r`-MPR2 enables significant time and energy savings, but lacks robustness for $a = 0.1$ and $a = 0.01$ as fewer problems are solved. Furthermore, the proportion of evaluations in lower FP format increases as a decreases. However, the worst (higher) time and energy ratios are obtained with $a = 0.01$. This is due to the fact that `r`-MPR2 performs many more objective and gradient evaluations with $a = 0.01$, due to overly optimistic evaluation bounds, leading to numerical instability. The results also show that the proposed strategy for `r`-MPR2 has a high success rate, therefore avoiding unnecessary re-evaluation with a higher precision FP format.

The performance profiles comparing R2 and `r`-MPR2 with respect to time and energy effort for the objective and the gradient are displayed in Figure 2. The performance profiles show that `r`-MPR2 outperforms R2, in accordance to the results displayed in Table 7 and Table 8.

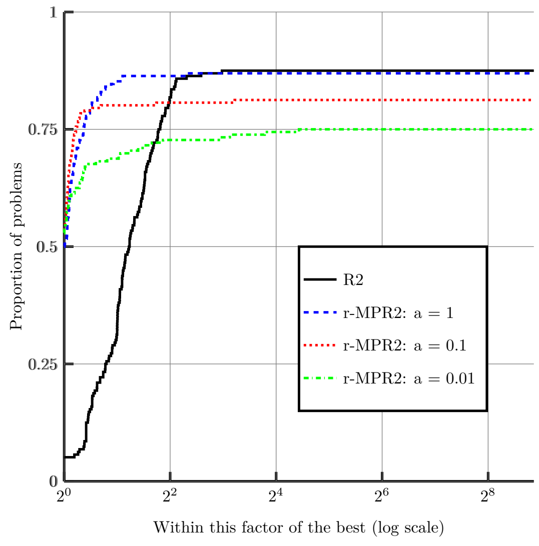
The limitation of `r`-MPR2 is that it does not solve as many problems as R2 for the lowest values of a. This is not necessarily intuitive: since `r`-MPR2 can perform evaluations with Float64 just as R2 and uses the same parameters (given in Table 5), one could expect `r`-MPR2 to be able to solve all the problems that R2 solves. From what we observed on the few problems that R2 solves and `r`-MPR2 does not, our interpretation is that the objective function error model is too optimistic, that is, the evaluation error is greater than what the model (75) estimates. This leads to computing $\widehat{f}(\widehat{x}_k)$ and $\widehat{f}(\widehat{c}_k)$ in FP formats that does not offer enough precision. As a consequence, ρ is not a reliable indicator for iteration success and can falsely be estimated lower than η_1 over several iterations, causing σ to



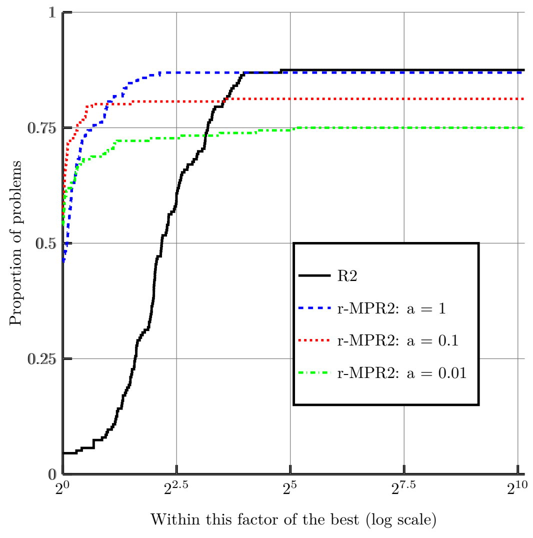
(a) Objective time effort perfomance profile.



(b) Objective energy effort perfomance profile.



(c) Gradient time effort perfomance profile.



(d) Gradient energy effort perfomance profile.

Figure 2: Performance profiles comparing R2 and Algorithm 2 with respect to time and energy effort for objective and gradient.

increase. This is typically the case if $f(\widehat{x}_k) > f(\widehat{c}_k)$ but $\widehat{f}(\widehat{x}_k) < \widehat{f}(\widehat{c}_k)$, which case should be avoided thanks to conditions (46) and (45) and reliable bounds $\omega_f(\widehat{x}_k)$, $\omega_f(\widehat{c}_k)$. In certain cases, $\widehat{x}_k$ is a local minimum for the noisy function $\widehat{f}$. The iterates become "trapped" in this neighbourhood since ρ can only be negative and σ increases, until $\widehat{c}_k = \widehat{x}_k$ because of candidate computation underflow. Note that such behaviour does happen with R2 and `r-`MPR2 with $a = 1$, but not on as many problems since employing only double precision helps avoid unreliable values of ρ.

One strategy to avoid this shortcoming would be to detect when the candidate is "trapped", and switching to a simple gradient descent, *i.e.*, accepting all the steps and not relying on the unreliable value of ρ. This however raises the question of the relevance of trying to reach a first-order point, based on a noisy measure of the gradient, with an accuracy such that the noise level on the objective evaluation makes it impossible to compare two close solutions.

6 Conclusion

In this paper, we proposedMPR2 (Algorithm 2), a multi-precision adaptation of R2 algorithm. We performed a comprehensive convergence analysis of Algorithm 2 that takes not only the objective function and gradient evaluation errors due to FP computations, but all the rounding errors (step and candidate computations, etc) expect for few defined variables that involve few operators. This convergence analysis provided convergence conditions that takes the rounding error into account, aggregated by the μ_k value.

We implemented rigorously Algorithm 2, using interval analysis to guarantee the evaluation errors of the objective function and the gradient. The numerical tests showed that the majority of the evaluations can be performed in Float32 on a bank of problems ranging between 1 and 100 variables, while Float16 is scarcely used. It also appeared that Float64 does not offer sufficient accuracy to guarantee the convergence of Algorithm 2 to a first order critical point.

We also proposed `r`-MPR2 algorithm, a version of Algorithm 2 based on relative error model for the objective and gradient evaluation, avoiding the time-consuming use of interval evaluation. The numerical results show that `r`-MPR2 enables significant computational savings compared with the original R2 algorithm, at the cost of lower robustness.

Future work could include extending the proposed analysis to second order methods. Such an analysis is challenging since the step is not simply computed from the gradient but from a process minimizing a quadratic subproblem. Such process (e.g. conjugate gradient) accumulates rounding errors that have to be taken into account.

Data availability

The results presented in this paper were obtained from the implementation of Algorithm 2 in MultiPrecisionR2.jl package available at `https://github.com/JuliaSmoothOptimizers/MultiPrecisionR2/tree/mpr2_paper`, tested over the set of unconstrained problems implemented in OptimizationProblems.jl [22]. The results can be generated from `https://github.com/JuliaSmoothOptimizers/MultiPrecisionR2/blob/mpr2_paper/docs/src/mpr2_numerical_test.md`.

A Convergence analysis: Proof

Lemma 5. For all $k > 0$,

$$|\widehat{f}(\widehat{x}_k) - \widehat{f}(\widehat{c}_k) - \widehat{\Delta T}_k| \leq \begin{array}{l} 2\eta_0\widehat{\Delta T}_k + \frac{1}{2}L\|\widetilde{s}_k^a\|^2 + \lambda_k\|\nabla f(\widehat{x}_k)\|\,\|\widehat{s}_k\| \\ +|\widetilde{g}_k^T\widehat{s}_k(1+\vartheta_{n+1}) - \nabla f(\widehat{x}_k)^T\widehat{s}_k|. \end{array} \tag{76}$$

Proof. With the triangular inequality, one has,

$$|\widehat{f}(\widehat{x}_k) - \widehat{f}(\widehat{c}_k) - \widehat{\Delta T}_k| \quad \leq |\widehat{f}(\widehat{x}_k) - f(\widehat{x}_k)| + |\widehat{f}(\widehat{x}_k) - f(\widehat{x}_k)| + |f(\widehat{x}_k) - f(\widehat{c}_k) - \widehat{\Delta T}_k|. \tag{77}$$

The termination condition at Step 3 and (2) enables to bound, we derive

$$|\widehat{f}(\widehat{x}_k) - \widehat{f}(\widehat{c}_k) - \widehat{\Delta T}_k| \quad \leq 2\eta_0\widehat{\Delta T}_k + |f(\widehat{x}_k) - f(\widehat{c}_k) - \widehat{\Delta T}_k|. \tag{78}$$

By Lemma 2, the second term of the right-hand side of (78) can be bounded as,

$$\begin{array}{ll} & |f(\widehat{x}_k) - f(\widehat{c}_k) - \widehat{\Delta T}_k| \\ = & |f(\widehat{x}_k) - f(\widehat{c}_k) + \widetilde{g}_k^T\widehat{s}_k(1+\vartheta_{n+1})| \\ \leq & |f(\widehat{x}_k) - f(\widehat{c}_k) + \nabla f(\widehat{x}_k)^T\widehat{s}_k| + |\widetilde{g}_k^T\widehat{s}_k(1+\vartheta_{n+1}) - \nabla f(\widehat{x}_k)^T\widehat{s}_k|. \end{array} \tag{79}$$

By triangular inequality and with (27), the first term of the right-hand side of (79) can be bounded as,

$$|f(\widehat{x}_k) - f(\widehat{c}_k) + \nabla f(\widehat{x}_k)^T \widehat{s}_k| \leq \begin{array}{l} |f(\widehat{x}_k) - f(\widehat{c}_k) + \nabla f(\widehat{x}_k)^T \widetilde{s}_k^a| \\ +|\nabla f(\widehat{x}_k)^T \widehat{s}_k - \nabla f(\widehat{x}_k)^T \widetilde{s}_k^a| \end{array} \tag{80}$$

With (4) one further has

$$\begin{aligned} |f(\widehat{x}_k) - f(\widehat{c}_k) + \nabla f(\widehat{x}_k)^T \widetilde{s}_k^a| &= |T(\widetilde{s}_k^a) - f(\widehat{c}_k)| \\ &= |T(\widetilde{s}_k^a) - f(\widehat{x}_k + \widetilde{s}_k^a)| \\ &\leq \tfrac{1}{2} L \|\widetilde{s}_k^a\|^2. \end{aligned} \tag{81}$$

By triangular inequality and with (33) and (35),

$$\begin{aligned} |\nabla f(\widehat{x}_k)^T \widehat{s}_k - \nabla f(\widehat{x}_k)^T (\widehat{x}_k \delta'_k + \widehat{s}_k(1+\delta'_k))| &= |\delta'_k \nabla f(\widehat{x}_k)^T (\widehat{x}_k + \widehat{s}_k)| \\ &\leq u'_k |\nabla f(\widehat{x}_k)^T (\widehat{x}_k + \widehat{s}_k)| \\ &\leq u'_k \|\nabla f(\widehat{x}_k)\| \, \|\widehat{x}_k + \widehat{s}_k\| \\ &\leq u'_k \|\nabla f(\widehat{x}_k)\| \, (\|\widehat{x}_k\| + \|\widehat{s}_k\|) \\ &\leq u'_k \|\nabla f(\widehat{x}_k)\| \, (\phi_k \|\widehat{s}_k\| + \|\widehat{s}_k\|) \\ &\leq \lambda_k \|\nabla f(\widehat{x}_k)\| \, \|\widehat{s}_k\|. \end{aligned} \tag{82}$$

Injecting Inequalities (81) and (82) into (80) yields,

$$|f(\widehat{x}_k) - f(\widehat{c}_k) - \nabla f(\widehat{x}_k)^T \widehat{s}_k| \leq \tfrac{1}{2} L \|\widetilde{s}_k^a\|^2 + \lambda_k \|\nabla f(\widehat{x}_k)\| \, \|\widehat{s}_k\| \tag{83}$$

Putting together Inequalities (78), (79) and (83) yields (76). □

Lemma 6. For all $k > 0$,

$$\left| \frac{\widetilde{g}_k^T \widehat{s}_k (1+\vartheta_{n+1}) - \nabla f(\widehat{x}_k)^T \widehat{s}_k}{\widehat{\Delta T}_k} \right| \leq \frac{u_k^g + \xi_{n+1}(u_k^g)\alpha_{n+1}(u_k^g) + \alpha_{n+1}(u_k^g)\omega_g(\widehat{x}_k)}{1 - u_k^g}. \tag{84}$$

Proof. With Lemma 2 and triangular inequality,

$$\begin{aligned} \left| \frac{\widetilde{g}_k^T \widehat{s}_k (1+\vartheta_{n+1}) - \nabla f(\widehat{x}_k)^T \widehat{s}_k}{\widehat{\Delta T}_k} \right| &= \left| \frac{\widetilde{g}_k^T \widehat{s}_k (1+\vartheta_{n+1}) - \nabla f(\widehat{x}_k)^T \widehat{s}_k}{\|\widetilde{g}_k\|^2 / \widehat{\sigma}_k (1+\vartheta_{n+1})} \right| \\ &\leq \frac{\|\widetilde{g}_k(1+\vartheta_{n+1}) - \nabla f(\widehat{x}_k)\| \, \|\widehat{s}_k\|}{\|\widetilde{g}_k\|^2 / \widehat{\sigma}_k (1+\vartheta_{n+1})} \end{aligned} \tag{85}$$

Note that in (85), we implicitly use the equality

$$\widetilde{g}_k^T \widehat{s}_k (1+\vartheta_{n+1}) = (\widetilde{g}_k \odot (1+\vartheta_{n+1}, \ldots, \vartheta_{n+1}))^T \, \widehat{s}_k \tag{86}$$

and with the abusive notation $\widetilde{g}_k \odot (1+\vartheta_{n+1}, \ldots, \vartheta_{n+1})^T$ rewrites $\widetilde{g}_k(1+\vartheta_{n+1})$.

Recalling that $\|\widehat{s}_k\| = \|\widetilde{g}_k\| / \widehat{\sigma}_k$, we further derive from (85),

$$\begin{aligned} \left| \frac{\widetilde{g}_k^T \widehat{s}_k (1+\vartheta_{n+1}) - \nabla f(\widehat{x}_k)^T \widehat{s}_k}{\widehat{\Delta T}_k} \right| &\leq \frac{\|\widetilde{g}_k(1+\vartheta_{n+1}) - \nabla f(\widehat{x}_k)\|}{\|\widetilde{g}_k\|(1+\vartheta_{n+1})} \\ &\leq \frac{\|\widetilde{g}_k(1+\vartheta_{n+1}) - \widehat{g}_k\|}{\|\widetilde{g}_k\|(1+\vartheta_{n+1})} + \frac{\|\widehat{g}_k - \nabla f(\widehat{x}_k)\|}{\|\widetilde{g}_k\|(1+\vartheta_{n+1})} \end{aligned} \tag{87}$$

With the triangular inequality, Lemma 1 and the bound (26), the first term of the right-hand side of (87) can be bounded as,

$$\begin{aligned} \frac{\|\widetilde{g}_k(1+\vartheta_{n+1}) - \widehat{g}_k\|}{\|\widetilde{g}_k\|(1+\vartheta_{n+1})} &= \frac{\|\widetilde{g}_k(1+\vartheta_{n+1}) - \widehat{g}_k - \widehat{g}_k(1+\vartheta_{n+1}) + \widehat{g}_k(1+\vartheta_{n+1})\|}{\|\widetilde{g}_k\|(1+\vartheta_{n+1})} \\ &\leq \frac{\|\widetilde{g}_k(1+\vartheta_{n+1}) - \widehat{g}_k(1+\vartheta_{n+1})\|}{\|\widetilde{g}_k(1+\vartheta_{n+1})\|} + \frac{\|\widehat{g}_k(1+\vartheta_{n+1}) - \widehat{g}_k\|}{\|\widetilde{g}_k\|(1+\vartheta_{n+1})} \\ &= \frac{\|\widetilde{g}_k - \widehat{g}_k\|}{\|\widetilde{g}_k\|} + \frac{\|\widehat{g}_k\| \, |\vartheta_{n+1}|}{\|\widetilde{g}_k\|(1+\vartheta_{n+1})} \\ &\leq \frac{\|\widetilde{g}_k - \widehat{g}_k\|}{\|\widetilde{g}_k\|} + \xi_{n+1}(u_k^g)\alpha_{n+1}(u_k^g) \frac{\|\widehat{g}_k\|}{\|\widetilde{g}_k\|} \end{aligned} \tag{88}$$

By Lemma 1, we further derive from (88),

$$\begin{aligned}\frac{\|\widetilde{g}_k(1+\vartheta_{n+1})-\widehat{g}_k\|}{\|\widetilde{g}_k\|(1+\vartheta_{n+1})} &\leq u_k^g\frac{\|\widehat{g}_k\|}{\|\widetilde{g}_k\|}+\frac{\alpha_{n+1}(u_k^g)\xi_{n+1}(u_k^g)}{1-u_k^g}\\ &\leq \frac{u_k^g+\alpha_{n+1}(u_k^g)\xi_{n+1}(u_k^g)}{1-u_k^g}.\end{aligned} \tag{89}$$

Using (26) to bound $1/(1+\vartheta_{n+1})$, and with Lemma 1, the second term in the right-hand side of (87) can be bounded as,

$$\begin{aligned}\frac{\|\widehat{g}_k-\nabla f(\widehat{x}_k)\|}{\|\widetilde{g}_k\|(1+\vartheta_{n+1})} &\leq \alpha_{n+1}(u_k^g)\frac{\omega_g(\widehat{x}_k)\|\widehat{g}_k\|}{\|\widetilde{g}_k\|}\\ &\leq \alpha_{n+1}(u_k^g)\frac{\omega_g(\widehat{x}_k)}{1-u_k^g}.\end{aligned} \tag{90}$$

Putting Inequalities (89) and (90) in Inequality (87), one obtains (84). □

Lemma 7. For all $k>0$,

$$\frac{1}{\widehat{\sigma}_k}\leq\left[1-\eta_2-\eta_0-\frac{\kappa_\mu}{2}\right]\frac{1}{\alpha_{n+1}(u_k^g)L(1+\lambda_k)^2}\implies \rho_k\geq\eta_2. \tag{91}$$

Proof. With Lemma 5,

$$\begin{aligned}|\rho_k-1| &= \left|\frac{\widehat{f}(\widehat{x}_k)-\widehat{f}(\widehat{c}_k)-\widehat{\Delta T}_k}{\widehat{\Delta T}_k}\right|\\ &\leq 2\eta_0+\frac{\frac{1}{2}L\|\widetilde{s}_k^a\|^2+\lambda_k\|\nabla f(\widehat{x}_k)\|\,\|\widehat{s}_k\|+|\widetilde{g}_k^T\widehat{s}_k(1+\vartheta_{n+1})-\nabla f(\widehat{x}_k)^T\widehat{s}_k|}{\widehat{\Delta T}_k},\end{aligned} \tag{92}$$

and with Lemma Lemma 6 we further derive

$$\begin{aligned}&|\rho_k-1|\\ \leq\;& 2\eta_0+\frac{\frac{1}{2}L\|\widetilde{s}_k^a\|^2}{\widehat{\Delta T}_k}+\frac{\lambda_k\|\nabla f(\widehat{x}_k)\|\,\|\widehat{s}_k\|}{\widehat{\Delta T}_k}+\frac{u_k^g+\xi_{n+1}(u_k^g)\alpha_{n+1}(u_k^g)+\alpha_{n+1}(u_k^g)\omega_g(\widehat{x}_k)}{1-u_k^g}.\end{aligned} \tag{93}$$

With Lemma 2, Lemma 3 and (26), and recalling that $\|\widehat{s}_k\|=\|\widetilde{g}_k\|/\widehat{\sigma}_k$, the second term of the right-hand side of (93) can be bounded as,

$$\begin{aligned}\frac{\frac{1}{2}L\|\widetilde{s}_k^a\|^2}{\widehat{\Delta T}_k} &= \frac{\frac{1}{2}L\|\widetilde{s}_k^a\|^2}{\|\widetilde{g}_k\|^2/\widehat{\sigma}_k(1+\vartheta_{n+1})}\\ &\leq \alpha_{n+1}(u_k^g)\frac{\frac{1}{2}L\|\widehat{s}_k\|^2(1+\lambda_k)^2}{\|\widetilde{g}_k\|^2/\widehat{\sigma}_k}\\ &= \alpha_{n+1}(u_k^g)\frac{\frac{1}{2}L(1+\lambda_k)^2}{\widehat{\sigma}_k}.\end{aligned} \tag{94}$$

with λ_k as in (35).

Similarly, with Lemma 2 and (26) the third term of the right-hand side of (93) can be bounded as

$$\frac{\lambda_k\|\nabla f(\widehat{x}_k)\|\,\|\widehat{s}_k\|}{\widehat{\Delta T}_k}\leq\alpha_{n+1}\frac{\lambda_k\|\nabla f(\widehat{x}_k)\|\,\|\widehat{s}_k\|}{\|\widetilde{g}_k\|^2/\widehat{\sigma}_k}, \tag{95}$$

and by Lemma 1 and triangular inequality,

$$\begin{aligned}\frac{\lambda_k\|\nabla f(\widehat{x}_k)\|\,\|\widehat{s}_k\|}{\widehat{\Delta T}_k} &\leq \alpha_{n+1}(u_k^g)\lambda_k\left(\frac{\|\nabla f(\widehat{x}_k)-\widehat{g}_k\|}{\|\widetilde{g}_k\|}+\frac{\|\widehat{g}_k\|}{\|\widetilde{g}_k\|}\right)\\ &\leq \alpha_{n+1}(u_k^g)\lambda_k\left(\frac{\omega_g(\widehat{x}_k)}{1-u_k^g}+\frac{1}{1-u_k^g}\right).\end{aligned} \tag{96}$$

Putting Inequalities (93), (94) and (96) together,

$$|\rho_k - 1| \leq 2\eta_0 \quad +\alpha_{n+1}(u_k^g)\frac{\frac{1}{2}L(1+\lambda_k)^2}{\widehat{\sigma}_k} + \frac{\alpha_{n+1}(u_k^g)\omega_g(\widehat{x}_k)(1+\lambda_k) + \alpha_{n+1}(u_k^g)\lambda_k + u_k^g + \xi_{n+1}(u_k^g)\alpha_{n+1}(u_k^g)}{1-u_k^g}. \tag{97}$$

Step 2 of Algorithm 2 ensures that the last term of the right-hand side of (97) is bounded as

$$\mu_k = \frac{\alpha_{n+1}(u_k^g)\omega_g(\widehat{x}_k)(1+\lambda_k) + \alpha_{n+1}(u_k^g)\lambda_k + u_k^g + \xi_{n+1}(u_k^g)\alpha_{n+1}(u_k^g)}{1-u_k^g} \leq \kappa_\mu, \tag{98}$$

and it follows that,

$$|\rho_k - 1| \leq 2\eta_0 + \kappa_\mu + \alpha_{n+1}(u_k^g)\frac{\frac{1}{2}L(1+\lambda_k)^2}{\widehat{\sigma}_k}. \tag{99}$$

Therefore, with $1/\widehat{\sigma}_k \leq [1-\eta_2-2\eta_0-\kappa_\mu]\dfrac{1}{\alpha_{n+1}(u_k^g)L(1+\lambda_k)^2}$ enforces by (91), the inequality rewrites

$$|\rho - 1| \quad \leq \tfrac{1}{2}(1-\eta_2) + \eta_0 + \frac{\kappa_\mu}{2} \tag{100}$$

and since the parameters are chosen such that $\eta_0 + \dfrac{\kappa_\mu}{2} \leq \frac{1}{2}(1-\eta_2)$, it follows that $|\rho - 1| \leq 1-\eta_2$. □

Lemma 8. For all $k > 0$,

$$\widehat{\sigma}_k \leq \sigma_{\max} = \frac{\gamma_3 L(1+\lambda_{\max})^2\alpha_{n+1}(u_{max})}{1-\eta_2-\eta_0-\dfrac{\kappa_\mu}{2}}, \tag{101}$$

with

$$\lambda_{\max} = \frac{\kappa_\mu}{\alpha_{n+1}(u_{min})}(1-u_{min}). \tag{102}$$

Proof. The termination condition at Step 2 ensures that $\mu_k \leq \kappa_\mu$, which implies that,

$$\frac{\alpha_{n+1}(u_k^g)\lambda_k}{1-u_k^g} \leq \kappa_\mu. \tag{103}$$

It follows that

$$\lambda_k \leq \frac{\kappa_\mu}{\alpha_{n+1}(u_k^g)}(1-u_k^g) \leq \frac{\kappa_\mu}{\alpha_{n+1}(u_{min})}(1-u_{min}) = \lambda_{\max}. \tag{104}$$

With Lemma 7,

$$\widehat{\sigma}_k \geq \frac{\alpha_{n+1}(u_k^g)L(1+\lambda_{\max})^2}{1-\eta_2-\eta_0-\frac{\kappa_\mu}{2}} \geq \frac{\alpha_{n+1}(u_k^g)L(1+\lambda_k)^2}{1-\eta_2-\eta_0-\frac{\kappa_\mu}{2}} \implies \rho_k \geq \eta_2. \tag{105}$$

With (48), and since $\gamma_1 < 1$, we further have,

$$\widehat{\sigma}_k \geq \frac{\alpha_{n+1}(u_k^g)L(1+\lambda_{\max})^2}{1-\eta_2-\eta_0-\frac{\kappa_\mu}{2}} \implies \widehat{\sigma}_j \leq \widehat{\sigma}_k. \tag{106}$$

As a consequence, the largest value that can be reached by $\widehat{\sigma}_k$ is obtained for the case where $\rho_{k-1} < \eta_1$ and $\sigma_{k-1} = \dfrac{\alpha_{n+1}(u_k^g)L(1+\lambda_{\max})^2}{1-\eta_2-\eta_0-\frac{\kappa_\mu}{2}} - \epsilon'$ with $\epsilon' > 0$ infinitesimally small. In this case,

$$\widehat{\sigma}_k = \sigma_{k-1}\gamma_3 \leq \frac{\alpha_{n+1}(u_k^g)L(1+\lambda_{\max})^2}{1-\eta_2-\eta_0-\frac{\kappa_\mu}{2}}\gamma_3 \leq \frac{\alpha_{n+1}(u_{max})L(1+\lambda_{\max})^2}{1-\eta_2-\eta_0-\frac{\kappa_\mu}{2}}\gamma_3 = \sigma_{\max} \tag{107}$$

□

Lemma 9. Let $S_k = \{0 \le j \le k \,|\, \rho_j \ge \eta_1\}$ be the set of successful iterations up to iteration k. For all $k > 0$,

$$k \le |S_k| \left(1 + \frac{|\log(\gamma_1)|}{\log(\gamma_2)}\right) + \frac{1}{\log(\gamma_2)} \log\left(\frac{\sigma_{\max}}{\widehat{\sigma}_0}\right). \tag{108}$$

Proof. Let $U_k = \{0 \le j \le k \,|\, \rho_j < \eta_1\}$ be the set of unsuccessful iterations. From the update formula of $\widehat{\sigma}_k$ (48), one has for each $k > 0$,

$$\forall j \in S_k,\ \gamma_1\widehat{\sigma}_j \le \max\left[\gamma_1\widehat{\sigma}_j, \sigma_{\min}\right] \le \widehat{\sigma}_{j+1} \text{ and } \forall i \in U_k,\ \gamma_2\widehat{\sigma}_i \le \widehat{\sigma}_{i+1}. \tag{109}$$

It follows that,

$$\widehat{\sigma}_0\gamma_1^{|S_k|}\gamma_2^{|U_k|} \le \widehat{\sigma}_k. \tag{110}$$

With Lemma 8 one obtains,

$$|S_k| \log\gamma_1 + |U_k| \log\gamma_2 \le \log\left(\frac{\sigma_{\max}}{\widehat{\sigma}_0}\right). \tag{111}$$

Since $\gamma_2 > 1$, it follows that,

$$|U_k| \le -|S_k|\frac{\log(\gamma_1)}{\log(\gamma_2)} + \frac{1}{\log(\gamma_2)} \log\left(\frac{\sigma_{\max}}{\widehat{\sigma}_0}\right). \tag{112}$$

Since $k = |S_k| + |U_k|$, the statement of Lemma 9 holds true. □

Proof of Theorem 1.

For all $j \in S_k$,

$$\begin{aligned} \rho_j = \frac{\widehat{f}(\widehat{x}_j) - \widehat{f}(\widehat{c}_j)}{\widehat{\Delta T}_k} &= \frac{\widehat{f}(\widehat{x}_j) - \widehat{f}(\widehat{x}_{j+1})}{\widehat{\Delta T}_j} \ge \eta_1 \\ \implies \quad \widehat{f}(\widehat{x}_j) - \widehat{f}(\widehat{x}_{j+1}) &\ge \eta_1\widehat{\Delta T}_j. \end{aligned} \tag{113}$$

The condition at Step 3 ensures that $\omega_f(\widehat{x}_j) \le \eta_0\widehat{\Delta T}_j$ and $\omega_f(\widehat{x}_{j+1}) \le \eta_0\widehat{\Delta T}_j$. As a consequence,

$$f(\widehat{x}_j) - f(\widehat{x}_{j+1}) \ge \widehat{f}(\widehat{x}_j) - \widehat{f}(\widehat{x}_{j+1}) - 2\eta_0\widehat{\Delta T}_j. \tag{114}$$

Putting (113) and (114) together,

$$f(\widehat{x}_j) - f(\widehat{x}_{j+1}) \ge (\eta_1 - 2\eta_0)\widehat{\Delta T}_j. \tag{115}$$

As a consequence, with Lemma 2,

$$\begin{aligned} f(x_0) - f_{low} &\ge (\eta_1 - 2\eta_0) \sum_{j\in S_k} \widehat{\Delta T}_j \\ &\ge (\eta_1 - 2\eta_0)(1 - \xi_{n+1}(u_k^g)) \sum_{j\in S_k} \widetilde{T}(x_j, 0) - \widetilde{T}(x_j, \widehat{s}_j) \\ &\ge (\eta_1 - 2\eta_0)(1 - \xi_{n+1}(u_k^g)) \sum_{j\in S_k} \|\widetilde{g}_j\|^2/\widehat{\sigma}_j \end{aligned} \tag{116}$$

From the termination condition at Step 1, before termination $\|\widehat{g}_j\|$ can be bounded below as,

$$\|\widehat{\widehat{g}_j}\| > \frac{1}{1 + \beta_{n+2}(u_k^g)} \frac{\epsilon}{1 + \omega_g(\widehat{x}_j)} \ge \frac{1}{1 + \beta_{n+2}(u_k^g)} \frac{\epsilon}{1 + \kappa_\mu}. \tag{117}$$

With Lemma 1 and 4, (117) yields,

$$\begin{aligned} \|\widetilde{g}_j\| &\ge \|\widehat{g}_j\|(1 - u_j^g) \\ &\ge \|\widehat{\widehat{g}_j}\| \left(1 - \beta_{n+2}(u_k^g)\right) (1 - u_j^g) \\ &> (1 - \beta_{n+2}(u_k^g))(1 - u_k^g) \frac{1}{1 + \beta_{n+2}(u_k^g)} \frac{\epsilon}{1 + \kappa_\mu}. \end{aligned} \tag{118}$$

With Lemma 8, putting (116) and (118) together yields,

$$\begin{aligned}
f(x_0) - f_{low} &\geq (\eta_1 - 2\eta_0)(1 - \xi_{n+1}(u_k^g)) \sum_{j \in S_k} \|\widetilde{g}_j\|^2 / \widehat{\sigma}_j \\
&\geq (\eta_1 - 2\eta_0)(1 - \xi_{n+1}(u_k^g)) \sum_{j \in S_k} \|\widetilde{g}_j\|^2 / \sigma_{\max} \\
&> (\eta_1 - 2\eta_0)(1 - \xi_{n+1}(u_k^g)) \sum_{j \in S_k} \left(\frac{1 - \beta_{n+2}(u_k^g)}{1 + \beta_{n+2}(u_k^g)} \frac{1 - u_k^g}{1 + \kappa_\mu} \right)^2 \frac{\epsilon^2}{\sigma_{\max}} \\
&> (\eta_1 - 2\eta_0)(1 - \xi_{n+1}(u_k^g)) |S_k| \left(\frac{1 - \beta_{n+2}(u_k^g)}{1 + \beta_{n+2}(u_k^g)} \frac{1 - u_k^g}{1 + \kappa_\mu} \right)^2 \frac{\epsilon^2}{\sigma_{\max}} \\
&> (\eta_1 - 2\eta_0)(1 - \xi_{n+1}(u_{max})) |S_k| \left(\frac{1 - \beta_{n+2}(u_{max})}{1 + \beta_{n+2}(u_{max})} \frac{1 - u_{max}}{1 + \kappa_\mu} \right)^2 \frac{\epsilon^2}{\sigma_{\max}} \\
&= \frac{1}{\kappa_s} |S_k| \epsilon^2.
\end{aligned} \tag{119}$$

As a consequence,

$$|S_k| < (f(x_0) - f_{low}) \kappa_s \epsilon^{-2}, \tag{120}$$

which proves (59). With Lemma 9, (59) yields,

$$\begin{aligned}
k &< |S_k| \left(1 + \frac{|\log(\gamma_1)|}{\log(\gamma_2)} \right) + \frac{1}{\log(\gamma_2)} \log \left(\frac{\sigma_{\max}}{\widehat{\sigma}_0} \right) \\
&< \epsilon^{-2} (f(x_0) - f_{low}) \kappa_s \left(1 + \frac{|\log(\gamma_1)|}{\log(\gamma_2)} \right) + \frac{1}{\log(\gamma_2)} \log \left(\frac{\sigma_{\max}}{\widehat{\sigma}_0} \right),
\end{aligned} \tag{121}$$

which proves (61) □

References

[1] JSOSolvers.jl. `https://github.com/JuliaSmoothOptimizers/JSOSolvers.jl`.

[2] IEEE standard for floating-point arithmetic. IEEE Std 754-2008, pages 1–70, 2008.

[3] Aleksandr Y Aravkin, Robert Baraldi, and Dominique Orban. A proximal quasi-newton trust-region method for nonsmooth regularized optimization. SIAM Journal on Optimization, 32(2):900–929, 2022.

[4] Afonso S Bandeira, Katya Scheinberg, and Luis Nunes Vicente. Convergence of trust-region methods based on probabilistic models. SIAM Journal on Optimization, 24(3):1238–1264, 2014.

[5] Stefania Bellavia, Gianmarco Gurioli, Benedetta Morini, and Philippe L Toint. Adaptive regularization for nonconvex optimization using inexact function values and randomly perturbed derivatives. Journal of Complexity, 68:101591, February 2022.

[6] Stefania Bellavia, Serge Gratton, and Elisa Riccietti. A Levenberg-Marquardt method for large nonlinear least-squares problems with dynamic accuracy in functions and gradients. Numerische Mathematik, 140(3):791–825, 2018.

[7] Stefania Bellavia, Gianmarco Gurioli, Benedetta Morini, and Philippe L Toint. The impact of noise on evaluation complexity: The deterministic trust-region case. Journal of Optimization Theory and Applications, 196(2):700–729, January 2023.

[8] Ernesto G Birgin, John L Gardenghi, José Mario Martínez, Sandra Augusta Santos, and Philippe L Toint. Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models. Mathematical Programming, 163(1):359–368, 2017.

[9] Jose Blanchet, Coralia Cartis, Matt Menickelly, and Katya Scheinberg. Convergence rate analysis of a stochastic trust-region method via supermartingales. INFORMS journal on optimization, 1(2):92–119, 2019.

[10] Liyuan Cao, Albert S Berahas, and Katya Scheinberg. First-and second-order high probability complexity bounds for trust-region methods with noisy oracles. arXiv preprint arXiv:2205.03667, 2022.

[11] Coralia Cartis, Nicholas I M Gould, and Philippe L Toint. An adaptive cubic regularization algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity. IMA Journal of Numerical Analysis, 32(4):1662–1695, 2012.

[12] Coralia Cartis, Nicholas I M Gould, and Philippe L Toint. Worst-case evaluation complexity and optimality of second-order methods for nonconvex smooth optimization. In Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018, pages 3711–3750. World Scientific, 2018.

[13] Anthony M Castaldo. Error analysis of various forms of floating point dot products. Technical report, The University of Texas at San Antonio, 2007.

[14] Andrew R Conn, Nicholas I M Gould, and Philippe L Toint. Trust region methods. SIAM, Philadelphia, 2000.

[15] Sameh Galal and Mark Horowitz. Energy-efficient floating-point unit design. IEEE Transactions on computers, 60(7):913–922, 2010.

[16] Serge Gratton, Clément W Royer, Luís N Vicente, and Zaikun Zhang. Complexity and global rates of trust-region methods based on probabilistic models. IMA Journal of Numerical Analysis, 38(3):1579–1597, 2018.

[17] Serge Gratton, Ehouarn Simon, and Philippe L Toint. An algorithm for the minimization of nonsmooth nonconvex functions using inexact evaluations and its worst-case complexity. Mathematical Programming, 187(1):1–24, 2021.

[18] Serge Gratton and Philippe L Toint. A note on solving nonlinear optimization problems in variable precision. Computational Optimization and Applications, 76(3):917–933, 2020.

[19] Nicholas J Higham. Accuracy and stability of numerical algorithms. SIAM, Philadelphia, 2002.

[20] Claude-Pierre Jeannerod and Siegfried M Rump. Improved error bounds for inner products in floating-point arithmetic. SIAM Journal on Matrix Analysis and Applications, 34(2):338–344, 2013.

[21] Dominique Monnet and and contributors. MultiPrecisionR2.jl. https://github.com/JuliaSmoothOptimizers/MultiPrecisionR2.

[22] Dominique Orban, Abel S Siqueira, and and contributors. OptimizationProblems.jl: A collection of optimization problems in JuMP syntax. https://github.com/JuliaSmoothOptimizers/OptimizationProblems.jl, March 2021.

[23] David P Sanders and and contributors. Juliaintervals guaranteed computations. https://juliaintervals.github.io/, March.

[24] Shigeng Sun and Jorge Nocedal. A trust region method for the optimization of noisy functions. arXiv preprint arXiv:2201.00973, 2022.

[25] Xiao Sun, Naigang Wang, Chia-Yu Chen, Jiamin Ni, Ankur Agrawal, Xiaodong Cui, Swagath Venkataramani, Kaoutar El Maghraoui, Vijayalakshmi Viji Srinivasan, and Kailash Gopalakrishnan. Ultra-low precision 4-bit training of deep neural networks. Advances in Neural Information Processing Systems, 33:1796–1807, 2020.

[26] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. Advances in neural information processing systems, 31, 2018.