IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 August 2025, accepted 21 August 2025, date of publication 25 August 2025, date of current version 29 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3602499

RESEARCH ARTICLE

Convergence of Iterative Descent Algorithms for LEO-PNT

WOUT VAN UYTSEL, ANDRES MARIA MAJORANA, THOMAS JANSSEN, AND MAARTEN WEYN, (Member, IEEE)
IDLab—imec, Faculty of Applied Engineering, University of Antwerp, 2000 Antwerp, Belgium

Corresponding author: Wout Van Uytsel (wout.vanuytsel@uantwerpen.be)

The work of Wout Van Uytsel was supported by the Research Foundation Flanders (FWO) through the Strategic Basic Research Ph.D. Fellowship under Project 1S01325N.

ABSTRACT Positioning Navigation and Timing (PNT) from Low Earth Orbit (LEO) has been gaining momentum in the last couple of years. LEO-PNT can serve as an alternative to current Global Navigation Satellite Systems (GNSS) or as an addition to the current system using a multi-layer approach. The aim of launching satellites into a lower orbit is to mitigate the shortcomings of current GNSS. When moving towards a lower orbit, algorithmic challenges that relate to the initial point estimate for Iterative Descent (ID) algorithms arise. In this paper, we analyze the convergence behavior of ID algorithms, focusing on Steepest Descent, Gauss-Newton, Trust Region, and Levenberg-Marquardt, when performing positioning using only LEO satellites, starting from an initial estimate at the center of the Earth to emulate a cold start scenario. We employ a Monte-Carlo (MC) simulation and multiple proposed LEO-PNT constellations to verify the convergence behavior of the algorithms. Among the evaluated methods, Trust-Region converges most reliably. The commonly used Gauss-Newton method often fails. Moreover, Levenberg–Marquardt is more robust but does not reach a perfect convergence rate. While simple, Steepest Descent requires the most iterations.

INDEX TERMS GNSS, GPS, Gauss-Newton, iterative-descent, LEO-PNT, least-squares, Levenberg-Marquardt, location-based services, positioning velocity and timing, trust region.

I. INTRODUCTION

Global Navigation Satellite Systems (GNSS) like GPS, GLONASS, Galileo, and BeiDou have provided users all over the globe with a means of performing Positioning, Navigation and Timing (PNT). Moreover, GNSS receivers are widespread and can be found in a variety of applications. Additionally, iterative development and refinement of the system have increased the reliability on the user side as well as on the space segment [1]. However, several limitations are inherent to the Medium Earth Orbit (MEO) which is mainly used in GNSS. The large distance between the user and the satellite results in a significant amount of path loss, and therefore GNSS satellite signals do not arrive strongly on Earth (-128.5 dBm). This low Power on Ground (PoG) makes the signals vulnerable to jamming attacks. Furthermore, the lack of authentication can enable spoofing of the signal [2]. Other concerns regard the lack of indoor positioning capabilities due to the low signal strength as well as the relatively slow motion across the sky due to the orbital altitude of GNSS [3].

LEO satellites orbit the Earth with an altitude below 2000 km as can be seen in Figure 1, which is significantly lower than typical GNSS orbits [4]. This lower orbit comes with its own advantages as well as disadvantages. Due to the closer distance to the Earth's surface, the signals experience significantly less path loss. Unlike GNSS satellites, LEO satellites can traverse parts of the ionosphere, which may require more detailed modeling compared to traditional GNSS [5]. Additionally, the lower orbit also increases the complexity of the orbital dynamics [6]. As a result of the higher velocity of the satellites a higher Doppler dynamic can

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenbao Liu.

be observed at the User Equipment (UE), which in turn can be used as an (extra) observable to perform Positioning Velocity and Timing (PVT) estimation. However, the downside of this higher Doppler dynamic is the larger frequency search in the acquisition stage. Another disadvantage of the lower orbital altitude is that it introduces challenges for Iterative Descent (ID) algorithms, particularly regarding the choice of initial position estimates during the navigation solution process. This problem is not present in GNSS due to the distant orbit; however, in LEO-PNT, this problem is present when performing a cold start procedure. In a variety of papers, they solve this issue by using a rough location estimate to initialize the ID algorithm [7], [8], [9]. Nevertheless, obtaining this location estimate relies on the use of assistance data or prior knowledge. In fact, in many real-world scenarios, modern receivers are often equipped with complementary technologies such as cellular, Wi-Fi, or inertial sensors, which can provide a coarse initial estimate of the user's position and thus mitigate the severity of a cold-start condition. However, we argue that the cold-start scenario remains an important and relevant case to study, especially in the context of emerging LEO-PNT systems. There are several practical and methodological considerations that justify its inclusion in this work. Firstly, in GNSS-denied environments—such as in the presence of jamming, spoofing, or policy-based signal restrictions, LEO-PNT may act as an independent positioning layer. In such cases, a receiver may need to acquire its position from scratch without access to any prior information, making a true cold start unavoidable. Secondly, from a system design perspective, cold-start capability is essential for ensuring autonomous operation during initial acquisition or after prolonged power-off periods, particularly in contexts where other subsystems may not yet be initialized or synchronized. Furthermore, in multi-layered PNT architectures that integrate LEO signals with GNSS, NR-5G, or other sources, the ability of a LEO-only solution to resolve an initial position independently enhances system robustness by enabling cross-validation and integrity monitoring. Even when external estimates are available, validating them through independent LEO-PNT cold-start positioning increases the trustworthiness of the fused solution. Additionally, minimalistic receiver designs, such as those used in IoT devices or in remote and infrastructure-sparse environments, may not have access to any complementary signals, thus rendering LEO-PNT the sole means of determining position. In such cases, robust cold-start algorithms are not merely useful but necessary. Finally, from a methodological standpoint, the cold-start condition serves as a worst-case benchmark for assessing algorithmic performance. An algorithm that reliably converges from an initial estimate at the center of the Earth is expected to perform even more efficiently when initialized closer to the true solution, as would be the case in practical deployments. Hence, evaluating convergence under this challenging but realistic constraint is instrumental in stress-testing and validating the robustness of the algorithms under consideration.

In this paper, we analyze the convergence behavior of common ID algorithms and compare their convergence rate for LEO-PNT. Answering the question: ''Which ID algorithm provides the most robust position estimation convergence rate for pseudorange-based LEO-PNT when initializing the algorithm from the center of the Earth?''. Our aim is to determine the ID algorithm that is the most robust for LEO-PNT pseudorange positioning. The remainder of this paper is structured as follows. In Section II, the related work to this paper is described, Section III provides a background to common ID algorithms and pseudorange positioning, and Section IV outlines the methodology used to analyze the convergence behavior. In Section V, we provide the results of the Monte Carlo (MC) simulation and discuss the findings. Section VI provides the conclusion of this work.

II. RELATED WORK

PNT services provided by GNSS have become a crucial technology of modern society. The use cases of the technology are very widespread, ranging from basic consumer solutions such as navigation of pedestrians to timing of critical infrastructure such as the electrical grid [10]. This technology is forecasted to reach a cumulative revenue of 4.6 trillion euros by 2031. The economic significance and the need for the services provided by GNSS make it a very critical infrastructure. Having a single system providing these PNT services is a big vulnerability. Therefore, a multi-layer approach can be considered to increase the reliability and diversity of the signals, as outlined as a vision for PNT in 2035 in the white paper by the European Space Agency [11]. The differences, challenges, and advantages of using LEO over MEO satellites for PNT are described in the survey paper by Prol et al. [2]. Moreover, in the paper by Ries et al. the possible methods of performing positioning with LEO-PNT are categorized [12]. Ries et al. consider three different categories of LEO-PNT, the first of which is purpose-built LEO-PNT; these satellites are optimized for PNT purposes and have a dedicated PNT payload onboard. Second, fused LEO-PNT, where the PNT services are embedded within a communication signal. Third, Signals Of OPportunity (SOOP) in this approach, no PNT information is explicitly transmitted, but observables from the communication link, such as Doppler shift, are used to do positioning. A comprehensive evaluation of these LEO-PNT concepts is presented by Eissfeller et al. [13]. Eissfeller et al. compares performance metrics such as positioning accuracy, service availability, and system complexity across different LEO-PNT architectures. In addition, Eissfeller et al. also analyzes multiple signal design characteristics, such as bandwidth and carrier frequency, and the effect of these parameters on the UE.

Moreover, the concept of LEO-PNT is gaining significant interest from the private sector [4], with multiple companies preparing, testing, or having launched such a constellation. Currently, Iridium Satellite Time and Location (STL) is the only fully operational fused LEO-PNT approach. Many companies are currently in the process of launching,

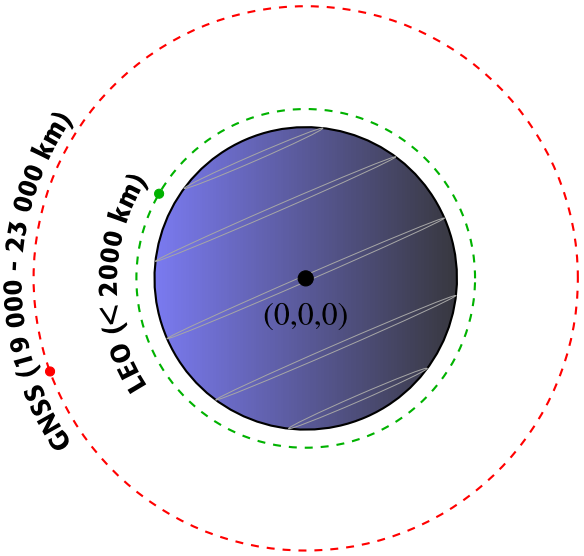


FIGURE 1. Illustration of LEO satellite versus GNSS satellites. LEO satellites operate in lower orbits (<2000 km), while GNSS satellites reside in higher orbits (≈20000 km).

preparing and testing their solution, such as Xona Space, Trustpoint, Centispace, Geely, and China Satellite Network Group. Moreover, governmental initiatives are also being prepared or launched, such as the In Orbit Demonstration (IOD) by ESA or JAXA's initiative to augment GNSS with a LEO layer.

The lower orbit, however, introduces algorithmic challenges that arise at the navigational processor, as described in the paper by Morichi et al. [9]. The paper describes the relationship between the orbital altitude of the satellites and the required accuracy of the initial estimate. A lower orbit requires a more accurate initial estimate for the Least Squares Gauss Newton (GN) algorithm to converge. The GN ID algorithm is commonly employed in GNSS when performing positioning. However, for LEO-PNT, this algorithm faces difficulties in a cold-start scenario. For instance, with an altitude of 1525 km the initial estimate should be within 825 km from the user's true position, for pseudorange positioning and 625 km for pseudorange rate positioning. Moreover, the use of ID algorithms for PNT purposes is described in the paper of Yan et al. [14]. The paper introduces a variety of ID algorithms like Steepest Descent (SD), GN, Levenberg-Marquardt (LM), and Trust Region (TR) for positioning and gives a general description. For a more detailed analysis of ID algorithms and numerical optimatization, we refer to the book by Nocedal and Wright [15]. Furthermore, the use of LM, GN, and SD ID algorithms is explored in the paper for GNSS in challenging visibility situations [16]. To date, no study has validated the use of ID algorithms other than GN for addressing the cold-start position estimation problem in LEO-PNT in scenarios where no a priori information about the receiver is available.

III. BACKGROUND

GNSS receivers determine the position of the receiver by measuring the travel time of the signal. The same principle can be used by LEO-PNT satellites. Each satellite broadcasts information which the UE can use to determine the distance between the satellite and the device [1]. Moreover, the UE should correct for errors such as ionospheric and tropospheric errors, clock errors, etc. [17]. The range obtained by the receiver between the satellite and itself is described in Equation 1.

$$\begin{aligned} p_k &= ||\mathbf{r}_k - \mathbf{r}_r|| + c \cdot b + \varepsilon_{p_k} \\ &= \sqrt{(x_k - x_r)^2 + (y_k - y_r)^2 + (z_k - z_r)^2} + cb + \varepsilon_{p_k} \end{aligned} \tag{1}$$

where,

$$\mathbf{r}_k = \begin{Bmatrix} x_k \\ y_k \\ z_k \end{Bmatrix}_{\text{ECEF}}, \quad \mathbf{r}_r = \begin{Bmatrix} x_r \\ y_r \\ z_r \end{Bmatrix}_{\text{ECEF}} \tag{2}$$

where p_k is the observed pseudorange between the k-th satellite with Earth Centered Earth Fixed (ECEF) coordinates $\mathbf{r}_k$ and the receiver with coordinates $\mathbf{r}_r$, c is the speed of light (299 792 458 m/s), b is the common bias including clock offset, and ε_{p_k} is the measurement error. Moreover, the receiver state that the ID algorithm should solve for is described in Equation 3. To solve for the four unknows the algorithm requires at least four pseudorange observables p_k at one epoch [17].

$$\mathbf{X} = \{x_r, y_r, z_r, cb\}^T = \{\mathbf{r}_r, b\}^T = \begin{Bmatrix} x_r \\ y_r \\ z_r \\ b \end{Bmatrix} = \begin{Bmatrix} \mathbf{r}_r \\ b \end{Bmatrix} \tag{3}$$

When performing positioning with LEO-PNT satellites, we need to acquire four observables. Afterwards, the objective of the ID algorithms is to minimize the sum of squared residuals between the measured pseudoranges p_k and the predicted pseudoranges $\hat{p}_k(\mathbf{X})$ [14]. ID algorithms can solve such a problem by starting from an initial estimate $\mathbf{X}_0$ and refining it by successive approximations. In a cold start situation, where no a priori information is available about the user's state, the algorithm will initialize in the center of the Earth and with no clock bias, i.e. $\mathbf{X}_0 = \{0, 0, 0, 0\}^T$, and start the approximation process from there onward [1]. Each iteration will update the estimate, and once the stop criteria are met, the algorithm will be terminated and the latest estimate will be provided to the user. Equation 4 describes the predicted pseudorange [14], with $\hat{\mathbf{r}}_r$ the estimated receiver location, and $\hat{b}$ the estimated clock bias.

$$\hat{p}_k(\mathbf{X}) = ||\mathbf{r}_k - \hat{\mathbf{r}}_r|| + c\hat{b} \tag{4}$$

The residual for the k-th satellite can therefore be described as in Equation 5. Where y_k indicates the residual between the

measured observables of the k-th satellite and the predicted observables.

$$y_k(\mathbf{X}_k) = p_k - \hat{p}_k(\mathbf{X}_k) \tag{5}$$

The goal of the algorithm is to minimize the objective function $f(\mathbf{X})$ over all the measurements, as described in Equation 6. With m, the total number of satellites in view at a given epoch of observation.

$$f(\mathbf{X}) = \min_{\mathbf{X}} \sum_{k=1}^{m} (y_k(\mathbf{X}_k))^2 \tag{6}$$

Moreover, we define the Jacobian (J) as in Equation 7. For the full derivation of the Jacobian for pseudorange positioning, refer to [1].

$$J(\mathbf{X}_n) = \begin{bmatrix} \left[\frac{\hat{\mathbf{r}}_r - \mathbf{r}_1}{\|\hat{\mathbf{r}}_r - \mathbf{r}_1\|}\right]^T & , 1 \\ \vdots & \\ \left[\frac{\hat{\mathbf{r}}_r - \mathbf{r}_k}{\|\mathbf{r}_k - \hat{\mathbf{r}}_r\|}\right]^T & , 1 \end{bmatrix} \tag{7}$$

A. GRADIENT DESCENT

Gradient Descent (GD) or SD is a simple ID algorithm. This first-order method updates $\mathbf{X}_{n+1}$ by using the gradient [15]. This is described in Equation 8.

$$\mathbf{X}_{n+1} = \mathbf{X}_n + \alpha \boldsymbol{g}(\boldsymbol{X_n}) \tag{8}$$

The derivative of $f(\mathbf{x})$ is approximated as such $\nabla f(\mathbf{X}_n) \approx J(\mathbf{X}_n)^T y(\mathbf{X}_n) = g(\mathbf{X}_n)$ and α is the step size [18]. With J being the Jacobian and n indicating the iteration number. The GD algorithm is relatively simple: only the first-order derivative of the cost function needs to be computed in order to populate the Jacobian, and no matrix inversion needs to be performed. This leads to an overall low-complexity ID algorithm. Let us set m the number of obtained residuals and n the number of unknows to determine: in terms of big O notation, the evaluation of the residual vector $y(\mathbf{X}_n)$ has a complexity of $O(m)$, whereas the evaluation of the gradient $g(\mathbf{X}_n)$ has a complexity of $O(m\mathrm{n})$. Thus, the GD algorithm as an overall $O(m\mathrm{n})$ complexity. Nevertheless, the algorithm is prone to zig-zagging along the contours of the cost function, and this results in the need for many iterations to converge [14]. Another challenge lies in deciding the magnitude of the step size. One can spend significant computing power on deciding on the best α, however, in our paper, we consider a relatively simple backtracking approach using the Armijo condition check.

B. GAUSS-NEWTON

The GN method can be seen as a modified version of Newton's optimization method with line search [15]. The core idea behind the latter optimization method is that the root-searching operation of a non-linear (objective) function can be approximated by iteratively searching the roots of its tangent evaluated at the respective intermediate step-point $\mathbf{X}_n$ [14]. Newton's optimization method with line search implies the computation of the objective function's Hessian matrix. Thus, the objective function's second-order derivatives need to be obtained: this can be computationally expensive and complicated, especially when dealing with GNSS measurements models. Therefore, the GN method approximates the Hessian matrix by employing the following simplification $\nabla^2 f(\mathbf{X}_n) \approx J(\mathbf{X}_n)^T J(\mathbf{X}_n) = h(\mathbf{X}_n)$, which avoids the need of second-order derivative computation. The update step can be calculated as described in Equation 9 [1], [15].

$$\mathbf{X}_{n+1} = \mathbf{X}_n + h(\mathbf{X}_n)^{-1} g(\mathbf{X}_n) \tag{9}$$

The GN convergence rate depends on the following factors [14], [15]. If the GN algorithm is initialized close enough to the true solution - hence residuals are very small - and the Jacobian matrix is full rank, the term $h(\mathbf{X}_n)$ is an excellent local hessian approximation. Therefore, the GN even shows a quadratic rate of convergence, since it behaves as equal as the sole Newton method. However, in real case GNSS/LEO-PNT scenarios, the residuals do not easily tend to zero, since the measurement readings possess inherent noise. Thus, the hessian approximation isn't perfect, and if the Jacobian is full rank, the algorithm has a superlinear convergence rate. Finally, if the Jacobian is not full-rank, the algorithm slows even further, showing a linear convergence rate.

Overall, the GN algorithm is highly effective for addressing general non-linear optimization problems in a quick manner, such as the one presented in Equation 6. This is the main reason why the GN algorithm is so commonly used in GNSS positioning. In fact, the GN algorithm is much faster than general-purpose optimization or second-order methods, and easily fits real-time or embedded processors. The main drawback is represented by the high computational load that might arise in evaluating the inverse of $h(\mathbf{X}_n)$ [14]. Let us further expand in regards to the algorithm's complexity in big O notation terms. Firstly, we can define the GN step $\delta = h(\mathbf{X}_n)^{-1} g(\mathbf{X}_n)$ as the result of the solution of the following system of equations, expressed in matrix form as in Equation 10 [15].

$$\begin{aligned} h(\mathbf{X}_n)\delta &= g(\mathbf{X}_n) \\ (J(\mathbf{X}_n)^T J(\mathbf{X}_n))\delta &= J(\mathbf{X}_n)^T y(\mathbf{X}_n) \end{aligned} \tag{10}$$

As per the GD algorithm, for the GN one the evaluation of the residuals vector has a $O(m)$ complexity, the Jacobian has a $O(m\mathrm{n})$ complexity, the gradient approximation has a $O(m\mathrm{n})$ complexity, the hessian approximation has a $O(m\mathrm{n}^2)$ complexity, and the evaluation of the GN step defined in Equation 10 has a $O(\mathrm{n}^3)$ complexity. Therefore, the total complexity of the GN algorithm per iteration is $O(m\mathrm{n}^2 + \mathrm{n}^3)$. In a GNSS and LEO-PNT environment, the number of unknowns are set to $\mathrm{n} = \dim(\mathbf{X}_n) = 4$, and thus the $O(\mathrm{n}^3)$ term is quite cheap. Hence, the complexity of the algorithm is dominated by the $O(m\mathrm{n}^2)$ term for a large number of residuals, that is to say when there is a large number of satellites in view at a given epoch.

C. LEVENBERG-MARQUARDT

LM varies between the GN and GD update by introducing a dampening coefficient λ [19]. Depending on the size of the dampening coefficient λ, the algorithm varies between a GN update (small values of λ) or a GD update (large values of λ).

$$\mathbf{X}_{n+1} = \mathbf{X}_n + (h(\mathbf{X}_n) + \lambda \mathbf{I})^{-1} g(\mathbf{X}_n) \tag{11}$$

Here $\mathbf{I}$ represents the identity matrix. Initially, λ is chosen to be a large value, which will result in the first step being a small GD step. Afterwards, the algorithm evaluates the dampening factor on each iteration. In the case that the next update step $\mathbf{X}_{n+1}$ decreases the objective function, λ is decreased by dividing by a factor γ and the update step is accepted meaning that $\mathbf{X}_{n+1}$ is accepted as new current estimate. If this is not the case λ is increased by multiplying with γ and $\mathbf{X}_{n+1}$ is not accepted as new best estimate, the algorithm will repeat with the increased λ and the current estimate of $\mathbf{X}$ is used in the next iteration ($\mathbf{X}_{n+1} \leftarrow \mathbf{X}_n$). The scaling of λ is described in Equation 12.

$$\lambda \leftarrow \begin{cases} \lambda/\gamma, & \text{if } f(\mathbf{X}_{n+1}) < f(\mathbf{X}_n) \quad \text{(successful step)} \\ \lambda \cdot \gamma, & \text{otherwise} \quad \text{(rejected step)} \end{cases} \tag{12}$$

Levenberg–Marquardt shares GN's fast quadratic convergence when the residual is nearing zero and the Jacobian is full-rank. The convergence rate drops to super-linearly when a small residual remains, and only linearly if the problem is ill-conditioned [15]. Levenberg-Marquardt's computational complexity is similar to that of GN, with the addition of the dampening term. The computational cost remains $O(mn^2 + n^3)$ per iteration, with the diagonal update for the dampening term costing $O(n^2)$, not altering the overall Big-O order.

D. TRUST-REGION

In TR, the algorithm aims to find the best step to take within a constrained region or 'Trust Region' around the current point $\mathbf{X}_n$ [20]. It first builds a quadratic model of the objective function at the current iteration. Second, a trust region (Δ) is defined around the current point where the model is expected to be accurate. Third, the algorithm computes a step direction that minimizes the model in this region. Fourth, the step will be evaluated, and, if accepted, the current point will be updated. These four steps will be repeated until the stopping criteria are met. While this generally holds true for TR algorithms, many variations exist. In the paper by Xinyao Li, multiple of these variations are described [20].

1) TRUST-REGION CAUCHY

One basic and simple approach to solving the subproblem is to use the *Cauchy point* [15]. Instead of solving the full quadratic minimization problem, the Cauchy point follows the direction of steepest descent, scaled to either reach the boundary of the trust region or stop earlier if the quadratic model suggests overstepping would not improve the cost.

The Cauchy step is defined in Equation 13 [15].

$$\mathbf{s}_C = -\tau \Delta \frac{g(\mathbf{X}_n)}{\|g(\mathbf{X}_n)\|} \tag{13}$$

where the step length τ is chosen as:

$$\tau = \begin{cases} 1 & \text{if } \mathbf{g}^\top \mathbf{h}\mathbf{g} \leq 0 \\ \min\left(\dfrac{\|g(\mathbf{X}_n)\|^3}{\Delta g(\mathbf{X}_n)^\top h(\mathbf{X}_n) g(\mathbf{X}_n)}, 1 \right) & \text{otherwise} \end{cases} \tag{14}$$

This guarantees that the step is within the trust region and follows a descent direction. To not confuse the reader, we will introduce $\mathbf{s}$ as the step. In the case of the TR with the Cauchy point $\mathbf{s} = \mathbf{s}_C$. The step is then evaluated using the ratio ρ between the actual and predicted reduction in the objective:

$$\rho = \frac{\frac{1}{2}\|\mathbf{y}(\mathbf{X}_n)\|^2 - \frac{1}{2}\|y(\mathbf{X}_n + \mathbf{s})\|^2}{-g(\mathbf{X}_n)^T \mathbf{s} - \frac{1}{2}\mathbf{s}^T h(\mathbf{X}_n)\mathbf{s}} \tag{15}$$

The trust region radius Δ is updated based on ρ:

$$\Delta_{n+1} = \begin{cases} \frac{1}{4}\Delta_n & \text{if } \rho < \frac{1}{4} \\ \min(2\Delta_n, \Delta_{\max}) & \text{if } \rho > \frac{3}{4} \text{ and } \|\mathbf{s}\| \approx \Delta_n \\ \Delta_n & \text{otherwise} \end{cases} \tag{16}$$

The step $\mathbf{s}$ is accepted (i.e., $\mathbf{X}_{n+1} = \mathbf{X}_n + \mathbf{s}$) if $\rho > \varepsilon$, where ε is a predefined acceptance threshold. Otherwise, the step is rejected and the iteration is repeated with the updated Trust Region radius.

The complexity of the algorithm is the summation of the computational cost when evaluating the residuals, the Jacobian and the gradient and the Hessian, the Cauchy step, the step evaluation and update of the estimation process. The computational cost of the residuals, of the Jacobian, of the gradient and of the hessian have already been disclosed when analyzing the previous algorithms. Equation 14 shows a conditional clause which shows a complexity of $O(n^2)$, meanwhile the norm operation has a $O(n)$ one, whereas the scalar division and minimum operation complexities are negligible; hence, the overall complexity of $O(n^2)$. Moreover, Equation 13 presents a vector norm operation of $O(n)$ complexity, and a scalar division and a dot product with the same complexity; hence, the overall complexity of just $O(n)$. The complexity of Equation 15 is negligible, since it is a simple scalar division; however, the evaluation of the actual reduction has a computational cost $O(m)$, whereas the evaluation of the predicted reduction, which presents a dominant term $h(\mathbf{X}_n)\mathbf{s}$, has a computational cost $O(n^2)$, which therefore coincides with the computational complexity of Equation 15. Nevertheless, the most computationally dominant term in big O notation remains the evaluation of the hessian approximation, hence the complexity of the TR Cauchy Step algorithm is of $O(mn^2)$.

2) TRUST-REGION DOG STEP

A popular adaptation of the regular TR algorithm is the Dog Leg Method. In this algorithm, firstly, the SD step is evaluated with exact line search [15].

$$\mathbf{s}_{sd} = \frac{g(\mathbf{X}_n)^T g(\mathbf{X}_n)}{g(\mathbf{X}_n)^T h(\mathbf{X}_n) g(\mathbf{X}_n)} g(\mathbf{X}_n) \tag{17}$$

Since the Dog Step TR cannot be described within one general update step, unlike the other algorithms. Therefore $\mathbf{s}_{sd}$ is the SD step and $\mathbf{s}_{GN}$ is the Gauss-Newton step. If the norm of the SD step ($\mathbf{s}_{sd}$) falls outside the trust region, it is normalized and scaled to the trust radius as described in equation 18.

$$\|\mathbf{s}_{sd}\| \geqslant \Delta \longmapsto \mathbf{s} = \Delta \frac{\mathbf{s}_{sd}}{\|\mathbf{s}_{sd}\|} \tag{18}$$

If the SD step ($\mathbf{s}_{sd}$) falls within the trust region ($\|\mathbf{s}_{sd}\| < \Delta$), the GN step is calculated, which is described in Equation 19.

$$\mathbf{s}_{GN} = (h(\mathbf{X}_n))^{-1} g(\mathbf{X}_n) \tag{19}$$

If $\|s_{GN}\| \leqslant \Delta$ we take the GN step $s = s_{GN}$ otherwise the dogstep is taken. To calculate the dogstep, the following equation needs to be solved: $a\beta^2 + b\beta + c = 0$ for β, where

$$\begin{aligned} \mathbf{d} &= \mathbf{s}_{GN} - \mathbf{s}_{sd} \\ \mathbf{a} &= \mathbf{d}^\top \mathbf{d}, \quad b = 2\,\mathbf{s}_{sd}^\top \mathbf{d}, \\ \mathbf{c} &= \mathbf{s}_{sd}^\top \mathbf{s}_{sd} - \Delta^2 \\ \beta &= \left(-\mathbf{b} + \sqrt{\mathbf{b}^2 - 4\mathbf{ac}}\right)/(2\mathbf{a}) \end{aligned} \tag{20}$$

The Dog Leg Step is then equal to:

$$\mathbf{s} = \mathbf{s}_{sd} + \beta(\mathbf{s}_{GN} - \mathbf{s}_{sd}) \tag{21}$$

Afterwards, the step is evaluated and the trust radius is adjusted. This is done in a similar manner as in Equation 15 and Equation 16.

With regards to the TR Dog Leg algorithm's complexity, we invite to take into account what has been previously discussed in sections III-D with regards to this topic. Therefore, in the worst-case scenario, the TR Dog Leg algorithm presents a $O(mn^2 + n^3)$ complexity. However, if the GN step is not needed, the complexity drops to $O(mn + n^2)$.

IV. METHODOLOGY

To support the testing of the convergence of ID algorithms for LEO-PNT, we use MATLAB in combination with the MATLAB Satellite Toolbox [21]. For the LEO space segment, we leverage the designs of Çelikbilek et al. [22] and Marchionne et al. [23]. The UE has a minimum elevation angle of 10 °. Moreover, the constellation parameters are described in Table 1.

Çelikbilek et al. [22] propose five LEO-PNT constellation designs. Among these, Çelikbilek 2 is optimized to minimize the overall Geometric Dilution of Precision (GDOP). To do so the constellation uses the higher region of LEO to increase satellite visibility. Whereas Çelikbilek 3 aims to maximize the Carrier-to-Noise Density Ratio (C/N_0) and utilizes a significantly lower orbit than Çelikbilek 2. In addition to these two constellations, Marchionne 2 is included for comparison. The constellation proposed by Marchionne et al. employs a fixed orbital altitude of 1250 km and uses a multi-objective optimization approach based on NSGA-II [23]. Unlike the designs of Çelikbilek et al., which target performance under challenging conditions such as indoor or urban scenarios and explicitly include metrics like C/N_0 and atmospheric drag, the Marchionne design focuses on achieving global coverage while minimizing GDOP at both average and worst user locations, as well as reducing the number of satellites and orbital planes to control deployment cost. Furthermore, the algorithms need to be limited to a maximum number of iterations since they cannot run indefinitely. Therefore, we limit first-order algorithms like SD and TR Cauchy to 2500 iterations, second-order algorithms to 100 iterations [15]. This cap should give each algorithm sufficient iterations to converge.

TABLE 1. Overview of the LEO constellations and their specifications used in this study. The shells follow either a Walker Delta (WD) or Walker Star (WS) configuration.

Constellation	Shell	Alt. [km]	Planes	Incl. [°]	Satellites	Ref.
Çelikbilek 2	WD-1	1835	19	34	133	[22]
	WS-2	1550	3	86	99	
	WD-3	1477	30	80	150	
Çelikbilek 3	WD-1	681	28	44	140	[22]
	WD-2	1134	11	86	132	
	WD-3	914	12	67	132	
Marchionne 2	WD-1	1250	4	12	80	[23]
	WD-2	1250	6	50	120	
	WD-3	1250	6	82	126	

A. MONTE CARLO SIMULATION

To test the performance of the different ID algorithms, we constructed a MC simulation. The MC simulation begins with the propagation of the satellite states over a 24 h period with 1 second sampling intervals, which are then saved. Afterwards, a random time within this 24 h time frame is drawn from a uniform distribution. Furthermore, a random location is also drawn from a uniform distribution with the location being somewhere near the Earth's surface with altitudes between 0 m and 15 km. If at this location fewer than four satellites are in view, a new location is drawn from the uniform distribution until four satellites are in view. This results in a worldwide test of the different algorithms with different satellite geometries at different times. In total, we test one million different locations and timeslots. The ID algorithms are then run, and convergence or divergence of the algorithm, and the number of steps till convergence are saved for each iteration.

B. SIMULATION OF OBSERVABLES

To ensure realistic testing of the algorithms, we need to simulate the pseudorange observables. To compute these observables, we start by computing the true ranges between

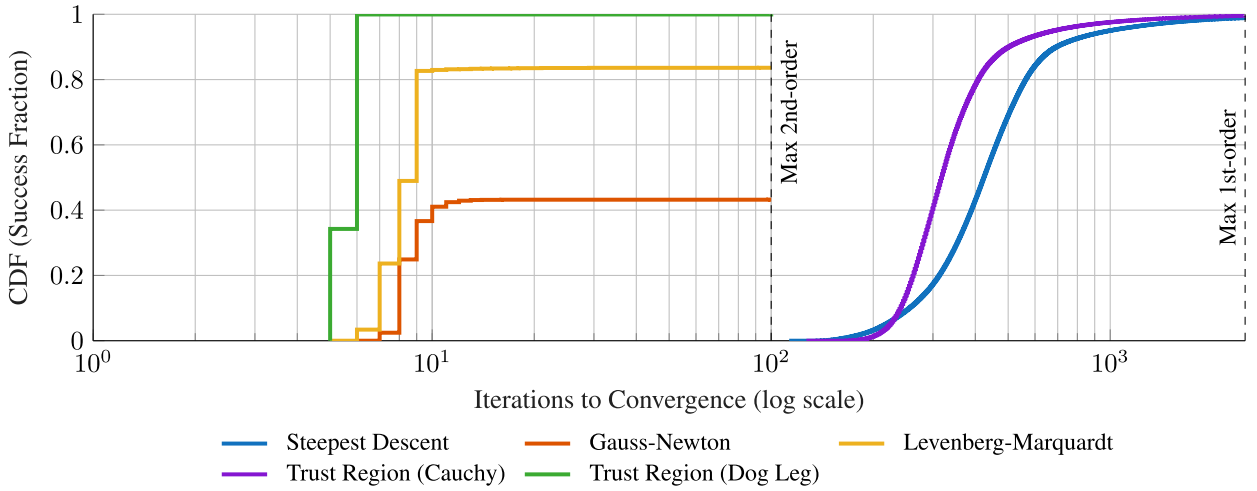


FIGURE 2. Cumulative Distribution Function (CDF) of the number of iterations required for convergence in the Monte Carlo simulation for constellation Çelikbilek 2. The y-axis indicates the proportion of successful runs that converged within the given number of steps for each Iterative Descent algorithm.

the satellite ($\mathbf{r}_k$) and the chosen receiver location ($\mathbf{r}_r$). These can be calculated as $p_k = \|\mathbf{r}_k - \mathbf{r}_r\|$. However, these ranges are perfect and therefore, we add normally distributed noise to each range individually to mimic a more realistic situation. This is done in the following manner: $p_{k,noise} = p_k + \sigma_p \cdot N_i$, with $p_{k,noise}$ the noisy pseudorange observable for the k-th satellite, σ_p the standard deviation which is set to 6 m to mimic a realistic pseudorange accuracy, and N_i the normal distribution variable [24]. Moreover, the maximum noise added is capped at 50 m since large outliers are removed beforehand for ID algorithms in GNSS, and we ensure this by limiting the maximum noise to a set maximum [17].

V. RESULTS AND DISCUSSION

Before moving to the MC results, we will first analyze one case to illustrate the difference in behavior of the algorithms using the constellation Çelikbilek 2 and given the skyplot in Figure 3. Moreover, it provides a clear insight into how the mathematical models try to improve the cost function step by step. Clearly visible in Table 2 is the difference between

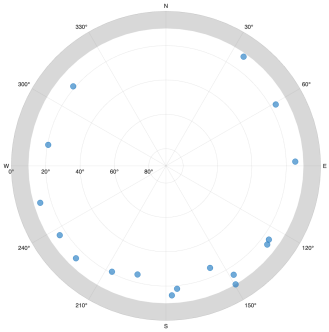


FIGURE 3. Skyplot of visible LEO satellites used in the one case example from Table 2. This one case example uses the Çelikbilek 2 constellation. The grey area indicates the 10 ° mask angle.

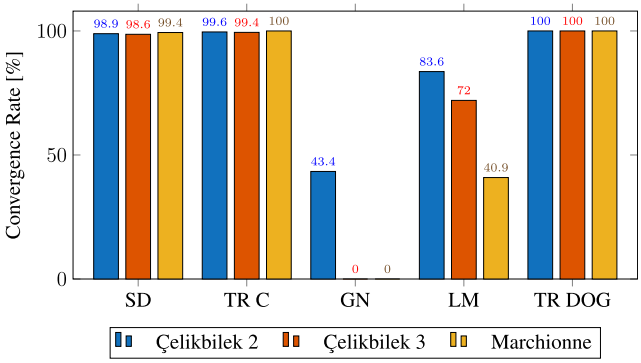
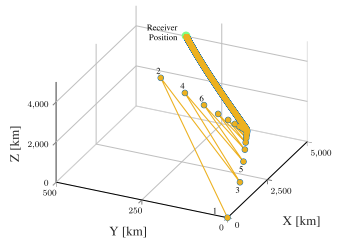
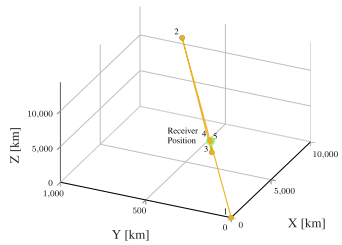
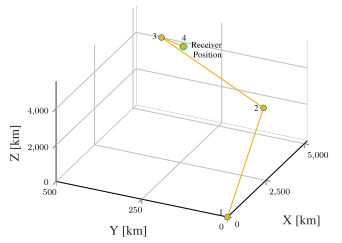
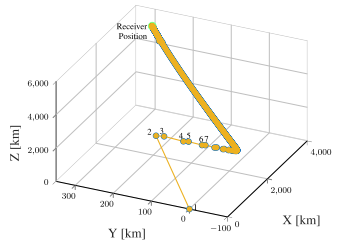
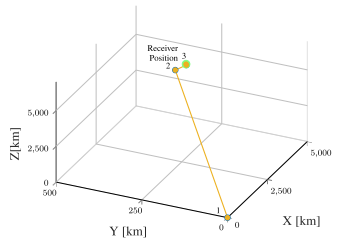


FIGURE 4. Comparison of convergence rates for each algorithm across constellations (rounded to one decimal place).

first-order methods and second-order methods. While the first-order methods might be relatively simple to compute, they require a very high number of iterations to converge. The SD method, in particular, demonstrates slow convergence due to its reliance on gradient direction alone, often resulting in inefficient zigzagging behavior, especially in ill-conditioned scenarios where the cost function has narrow, elongated contours. In contrast, second-order methods often show a greater improvement in significantly fewer steps. However, when no dampening is introduced, like for instance in GN, the algorithm can significantly overshoot the real location. The large residuals in this scenario lead to disproportionately large update steps, which can cause the algorithm to diverge or oscillate rather than converge. The results of the MC simulation for the constellation Çelikbilek 2 are shown in Figure 2, which depicts the number of steps required by various ID algorithms to reach convergence when using only LEO satellites and initializing from the Earth's center. When analyzing the results of the MC simulation with the constellation Çelikbilek 2, it is evident that first-order

TABLE 2. Convergence visualization and discussion for each iterative descent algorithm. The iteration numbers are shown above the nodes only if the steps are not too close together.

Figure	Algorithm and Discussion
	**Steepest Descent**: In the first step, α is set to 1, and when the step does not meet Armijo condition, α gets halved. Initially, the step taken would be very large and doesn't meet Armijo condition, the algorithm therefore significantly reduces α and that results in a relatively small first step. Every iteration α is set to 1 again, and the algorithm performs a similar Armijo condition check and adjusts the step size accordingly. This ensures that the largest possible step is taken when possible.
	**Gauss-Newton**: The initial step significantly overestimates the size of the step it has to take, therefore, resulting in a significant overshoot of the real receiver location. The approximation of the Hessian assumes that the initial point is relatively close to the real receiver location, but due to the large residuals, this approximation breaks down. Without any dampening mechanism in place, the algorithm trusts the local approximation, leading to a very large initial update step.
	**Levenberg-Marquardt**: Hybrid approach that interpolates between GN and SD. Initial values are set to: $\tau = 0.01, \gamma = 10$. In the first iteration, the GN step is very large due to the big residual, however, the cost function is still lower than the initial cost function. Therefore λ is decreased by a factor of γ. In the case that the current step is worse than the previous one, the new step is not accepted and τ is increased.
	**Trust Region (Cauchy)**: Implements a TR approach the initial values are set to $\varepsilon = 0.15$ and $\Delta = 6 * 10^6$ or roughly the radius of the Earth. In the first iteration, τ or the step length is 0.5715, which is calculated according to Equation 14 and therefore it doesn't scale all the way till the trust region bound.
	**Trust Region (Dog Leg)**: Uses the Dog Leg method to solve the trust-region subproblem. Initial values are set the same as in the Trust Region (Cauchy Point). Depending on the step proposed by SD it decides on using SD, GN or a combination (Dog Leg step). In this specific case, it uses the Dog Leg step; afterwards, it continues by using GN. Since it takes a step with the size of Δ, this step gets very close to the real receiver location.

methods require significantly more iterations to converge compared to second-order methods, which was also visible in Table 2. Out of 1 million MC trials, SD successfully converged in 988951 cases and has a Wilson confidence interval with 99 % (3)-σ) of [0.988678, 0.989217]. The TR Cauchy algorithm reaches convergence in 995960 of the cases and has a Wilson confidence interval with 99 % (3)-σ) of [0.995793, 0.996120]. Moreover, the TR Cauchy approach requires fewer iterations in most cases to converge compared to SD.

In contrast to first-order methods, which follow the gradient direction, second-order approaches also account for the local curvature, often requiring significantly fewer steps to converge. However, due to the large overshoot of GN, it is the only second-order method that does not reach a high convergence rate. Although the global minimum of the

residuals is situated near the Earth's surface, local minima can cause the algorithm to converge to suboptimal solutions and GN fails to improve from thereon. In this case GN, overshoots the real receiver location and afterwards oftentimes gets trapped at a local minimum on the backsides of the satellites. The GN algorithm only reaches convergence in 433579 of the cases and has a Wilson confidence interval with 99 % (3)-σ) of [0.432303, 0.434856]. In contrast to GN and LM, TR Dog Leg reaches a significantly better convergence rate - this is due to the dampening that is introduced by the algorithms, preventing this large overshoot. With TR Dog Leg reaching 999998 out of the 1 million trials. Moreover, TR Dog Leg only requires 4 to 6 iterations to converge. LM has a higher convergence rate than GN but still does not reach 100 %. LM reached convergence in 836185 of the cases and has a Wilson confidence interval with 99 % (3)-σ) of [0.835229, 0.837136].

Across the three LEO constellations (Çelikbilek 2, Çelikbilek 3, Marchionne), the convergence behavior differs. GN's success rate collapses from 43.3579 % to 0.0079 % for Çelikbilek 3 and 0.0002 % for the Marchionne constellation. This sharp drop suggests that the closer the satellites are to the user (i.e., the lower the orbital altitude), the more pronounced GN's overshoot becomes, increasing the risk of divergence as was also described in the paper by Morichi et al. [9]. Moreover, a difference in convergence rate can also be seen when employing LM with the different constellations. Its convergence rate drops from 83.6 % when employing the Çelikbilek 2 constellation to 72 % for Çelikbilek 3 and 40.9 % for the Marchionne constellation. In contrast, first-order SD and TR Cauchy maintain close to 99 % convergence but still require an order of magnitude more iterations. The Trust-Region Dog-Leg algorithm remains virtually unaffected, reaching >99.998 % convergence in just 4–6 iterations across all scenarios, making it the most robust choice.

These results clearly show that there is a big contrast in convergence behavior between the different ID algorithms for LEO-PNT pseudoranging. First-order methods have been shown to require a very high number of iterations when initialized from the center of the Earth. Their progress in decreasing the cost function is slow (linear) and even when reaching near the real receiver location they need many more iterations to find the global minimum as can be seen in Table 2. Second-order methods like GN have been shown to take larger steps. However, there is the risk of overstepping and getting stuck in a local minimum which oftentimes is the case for GN. LM has been shown to increase the robustness, but still does not reach convergence all of the time. TR Dog Leg has been shown to reach almost perfect convergence rates. Moreover, it requires the least number of iterations to converge but also is the most complex [14].

VI. CONCLUSION

In this work, we evaluated common ID algorithms for LEO-PNT pseudorange positioning. We found that TR Dog Leg performed the best out of all tested algorithms, reaching a near-perfect convergence rate of 99.9998 % to 100 % in very few iterations. The other second-order methods like LM and GN also exhibited fast convergence, but with lower success rates for LM (83.6185 % - 40.9 %) and even less for GN. First-order methods required an excessive number of iterations, making them impractical for real-time applications despite their relatively high success rate. Future work could focus on verifying these results with real LEO-PNT satellite measurements or tuning the configurable parameters of the ID algorithms for LEO-PNT. In conclusion, this work addressed the cold-start initialization problem for PVT estimation in LEO-PNT by analyzing and validating the convergence behavior of commonly used ID algorithms with a MC simulation and multiple constellation designs. With LEO-PNT gaining momentum, this paper presents a robust solution for the cold-start positioning problem with solely LEO satellites by applying the Dog Leg method.

REFERENCES

[1] F. Van Diggelen, "A-GPS: Assisted GPS, GNSS, and SBAS," in *House GNSS Technology and Applications Library*. Boston, MA, USA: Artech House, 2009.

[2] F. S. Prol, R. M. Ferre, Z. Saleem, P. Välisuo, C. Pinell, E. S. Lohan, M. Elsanhoury, M. Elmusrati, S. Islam, K. Çelikbilek, K. Selvan, J. Yliaho, K. Rutledge, A. Ojala, L. Ferranti, J. Praks, M. Z. H. Bhuiyan, S. Kaasalainen, and H. Kuusniemi, "Position, navigation, and timing (PNT) through low Earth orbit (LEO) satellites: A survey on current status, challenges, and opportunities," *IEEE Access*, vol. 10, pp. 83971–84002, 2022.

[3] M. B. Kjærgaard, H. Blunck, T. Godsk, T. Toftkjær, D. L. Christensen, and K. Grønbæk, "Indoor positioning using GPS revisited," in *Pervasive Computing*. Berlin, Germany: Springer, 2010, pp. 38–56.

[4] FrontierSI, "State of the market report: Low Earth orbit positioning navigation and timing (LEO PNT)," FrontierSI, Melbourne, VIC, Australia, Market Rep. V1.1, 2024.

[5] C. Oezmaden, S. Pelzer, O. G. Crespillo, M. Brachvogel, M. Niestroj, and M. Meurer, "Residual GNSS ionospheric error analysis in future low Earth orbit applications," in *Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS)*, Apr. 2025, pp. 526–538.

[6] F. D. O. Salgueiro, M. C. Limon, and P. Giordano, "A novel navigation message for LEO satellites," in *Proc. I_{ON} GNSS+, Int. Tech. Meeting Satell. Division Inst. Navigat.*, Oct. 2024, pp. 1082–1093.

[7] Z. M. Kassas, S. Kozhaya, H. Kanj, J. Saroufim, S. W. Hayek, M. Neinavaie, N. Khairallah, and J. Khalife, "Navigation with multi-constellation LEO satellite signals of opportunity: Starlink, OneWeb, orbcomm, and iridium," in *Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS)*, Apr. 2023, pp. 338–343.

[8] N. Jardak, R. Adam, and Q. Jault, "Leveraging multi-LEO satellite signals for opportunistic positioning," *IEEE Access*, vol. 12, pp. 127100–127114, 2024.

[9] L. Morichi, A. Minetto, A. Nardin, S. Zocca, and F. Dovis, "Pseudorange and Doppler-based state estimation from MEO to LEO: A comprehensive analysis of maximum likelihood estimators," in *Proc. Int. Tech. Meeting The Inst. Navigat.*, Long Beach, CA, USA, Feb. 2024, pp. 677–691.

[10] *EUSPA EO and GNSS Market Report.2024/Issue 2*, European Union Agency for the Space Programme, Luxembourg, 2024.

[11] Navigation Innovation Support Programme (NAVAC), "PNT vision 2035," Eur. Space Agency (ESA), European, Tech. Rep., 2024.

[12] L. Ries, M. C. Limon, F.-C. Grec, M. Anghileri, R. Prieto-Cerdeira, F. Abel, J. Miguez, J. V. Perello-Gisbert, S. D'Addio, R. Ioannidis, A. Ostillio, M. Rapisarda, R. Sarnadas, and P. Testani, "LEO-PNT for augmenting Europe's space-based PNT capabilities," in *Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS)*, Monterey, CA, USA, Apr. 2023, pp. 329–337.

[13] B. Eissfeller, T. Pany, D. Dötterböck, and R. Förstner, "A comparative study of LEO-PNT systems and concepts," in *Proc. I_{ON} Pacific PNT*, May 2024, pp. 758–782.

[14] J. Yan, C. C. J. M. Tiberius, G. J. M. Janssen, P. J. G. Teunissen, and G. Bellusci, "Review of range-based positioning algorithms," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 28, no. 8, pp. 2–27, Aug. 2013.
[15] J. Nocedal and S. J. Wright, "Numerical optimization," in *Springer Series in Operations Research and Financial Engineering*, 2nd ed., New York, NY, USA: Springer, 2006.
[16] P. S. Kumar, A. Jayalaxmi, V. B. S. S. I. Dutt, P. K. Rao, P. K. Rao, and L. Ganesh, "Advanced algorithms for pseudo-range estimation and positioning accuracy in challenging satellite visibility conditions," *J. Commun.*, vol. 19, no. 10, pp. 1–10, 2024.
[17] I. G. Petrovski, *GPS, GLONASS, Galileo, and BeiDou for Mobile Devices: From Instant to Precise Positioning*, 1st ed., Cambridge, U.K.: Cambridge Univ. Press, May 2014.
[18] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, Jan. 1996.
[19] H. P. Gavin, "The Levenberg–Marquardt algorithm for nonlinear least squares curve-fitting problems," Dept. Civil Environ. Eng., Duke Univ., pp. 1–23, 2024.
[20] X. Li, "Overview of trust-region methods," *Frontiers Comput. Intell. Syst.*, vol. 8, no. 3, pp. 25–27, Jun. 2024.
[21] *Satellite Communications Toolbox*, MATLAB, Natick, MA, USA, 2025.
[22] K. Çelikbilek, E. S. Lohan, and J. Praks, "Optimization of a LEO-PNT constellation: Design considerations and open challenges," *Int. J. Satell. Commun. Netw.*, vol. 43, no. 4, pp. 272–292, Jul./Aug. 2024.
[23] L. Marchionne, L. M. Gessato, F. Toni, and S. L. Barbera, "Striking a balance: Performance and cost optimization of LEO-PNT constellation for hybrid users using a meta-heuristic approach," in *Proc. IEEE 10th Int. Workshop Metrology Aerosp. (MetroAeroSpace)*, Jun. 2023, pp. 609–614.
[24] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance*, 2nd ed., Lincoln, MA, USA: Ganga-Jamuna Press, 2006.

WOUT VAN UYTSEL received the bachelor's and master's degrees in applied engineering: electronics-ICT from the University of Antwerp, Belgium, in 2022 and 2023, respectively, where he is currently pursuing the Ph.D. degree with the IDLab—imec Research Group. His research interests include low Earth orbit (LEO) satellite communication and indoor positioning using satellite signals.

ANDRES MARIA MAJORANA received the bachelor's degree in aerospace engineering from the University of Padua, Italy, in 2022, and the master's degree in space engineering from Aforesaid University, in 2025. He is currently pursuing the Ph.D. degree with the IDLab—imec Research Group, University of Antwerp. He pursued an interest in LEO-PNT under a joint academic program between the University of Padua and the University of Antwerp, in collaboration with the IDLab-imec Research Group. His research interests include navigation and positioning algorithms for multi-layered fixing with space-based assets.

THOMAS JANSSEN received the master's and Ph.D. degrees from the Department of Electronic, Information and Communication Technology, Faculty of Applied Engineering, University of Antwerp. He is currently a Senior Researcher with the IDLab, Research Group of imec, and the Department of Electronic, Information and Communication Technology, Faculty of Applied Engineering, University of Antwerp. In addition, as a Doctoral Assistant, he teaches courses in network security and cybersecurity. His research interests include non-terrestrial networks (NTN), specializing in satellite communication (5G/6G NTN) and localization (LEO-PNT).

MAARTEN WEYN (Member, IEEE) received the Ph.D. degree in computer science on the topic of opportunistic seamless localization from the University of Antwerp, Antwerp, Belgium, in 2011. He is currently a Full Professor and the Vice-Rector of research and impact with the University of Antwerp. He teaches wireless communication systems. His research at imec—IDLab, focuses on ultra-low power sensor communication, embedded systems, sub-1 GHz communication, sensor processing, and localization. He has co-founded spin-offs Aloxy, CrowdScan, IoSa, and AtSharp, contributed to 1OK and Viloc.