

TECHNICAL
REPORTS: METHODS

10.1029/2019RS006875

Key Points:

- This study proposes a new integration method for NeQuick-G that speeds calculations a 21% and 49% with respect the two integration algorithms officially recommended
- The main vector to diminish computational costs is the reduction of the number of function calls involved in the integration without increasing the error
- The authors strongly recommend revisiting the convergence control of the integration routines in the official document following the discussions in this study

Correspondence to:

A. Aragon-Angel,
maria-angeles.aragon@ec.europa.eu

Citation:

Aragon-Angel, A., Zürn, M., & Rovira-Garcia, A. (2019). Galileo Ionospheric Correction Algorithm: An Optimization Study of NeQuick-G. *Radio Science*, 54, 1156–1169. <https://doi.org/10.1029/2019RS006875>

Received 10 MAY 2019

Accepted 15 OCT 2019

Accepted article online 26 OCT 2019

Published online 24 NOV 2019

Galileo Ionospheric Correction Algorithm: An Optimization Study of NeQuick-G

A. Aragon-Angel¹ , M. Zürn¹, and A. Rovira-Garcia² ¹European Commission, Joint Research Centre, Directorate for Space, Security and Migration, Ispra, Italy, ²Research Group of Astronomy and Geomatics, Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract At present most low-cost GNSS receivers operate one frequency in the L band. For them one of the largest error contributions is the delay of radio signals in the Ionosphere. NeQuick-G is the official ionospheric correction algorithm (ICA), which has been adopted for Galileo, the European GNSS Programme. The NeQuick-G implementation is complex when compared with other ICAs. It is also demanding in terms of computational resources. The Joint Research Centre completed a reference implementation of NeQuick-G based on the official document “Ionospheric Correction Algorithm for Galileo Single Frequency Users” provided by the European Global Navigation Satellite Systems Agency. The rationale behind the JRC implementation of NeQuick-G was the intent to write an independent source code from scratch, without using the pseudo-codes from the reference document and solely relying on the physics descriptions. Using such implementation as baseline, this paper describes an optimization attempt of the official pseudo-code from an algorithmic perspective. The objective was to reduce the computational load while not sacrificing the performance. The new proposed integration method is able to speed up calculations to 21% and 49% with respect the two official integration algorithms. The overall computational burden depends on the number of operations, which is eventually closely correlated to the number of calls of the ionospheric model. This underlines the quest to find an integration method reducing this number of calls. Moreover, based on the findings of this study, the authors strongly recommend revisiting the convergence control of the integration routines introduced in (European Commission, 2016, https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo_Ionospheric_Model.pdf).

1. Introduction

NeQuick-G is the official Ionospheric Correction Algorithm (ICA) that has been developed for Galileo, the European GNSS Programme. The Joint Research Centre (JRC) of the European Commission has implemented an independent NeQuick-G version based solely on the Reference Document (European Commission, 2016), hereafter referred as RD. The implementation is available to the public at <https://nequick-g.jrc.ec.europa.eu/> for the calculation of reference NeQuick-G Slant Total Electron Content (STEC) and its source code is currently undergoing a licensing procedure to get a European Union Public License (European Commission, 2017) to be openly distributed.

This paper provides a brief overview of the JRC implementation and presents an optimization that goes beyond the official ICA description in RD. Figure 1 depicts an overview of the functions involved in the implementation of NeQuick-G available at RD. One can see that the practical totality of the functions lies within the DoTECintegration function, since the integration procedure is the most demanding feature of this code.

The JRC optimization exploits the fact that NeQuick-G fundamental mathematical task is indeed a numerical integration. Actually the ionospheric electron density is integrated over the Line of Sight (LoS) between a GNSS receiver and a GNSS satellite. The result of this integration is the Total Electron Content (TEC). TEC is a direct measure for the ionospheric contribution to the range error, that is, the distance between the receiver and satellite antennas.

Our optimization approach consists of reducing the number of function calls related to these integrals. Each call consists of the creation of ionospheric parameters for a point on the LoS.

©2019. The Authors.

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

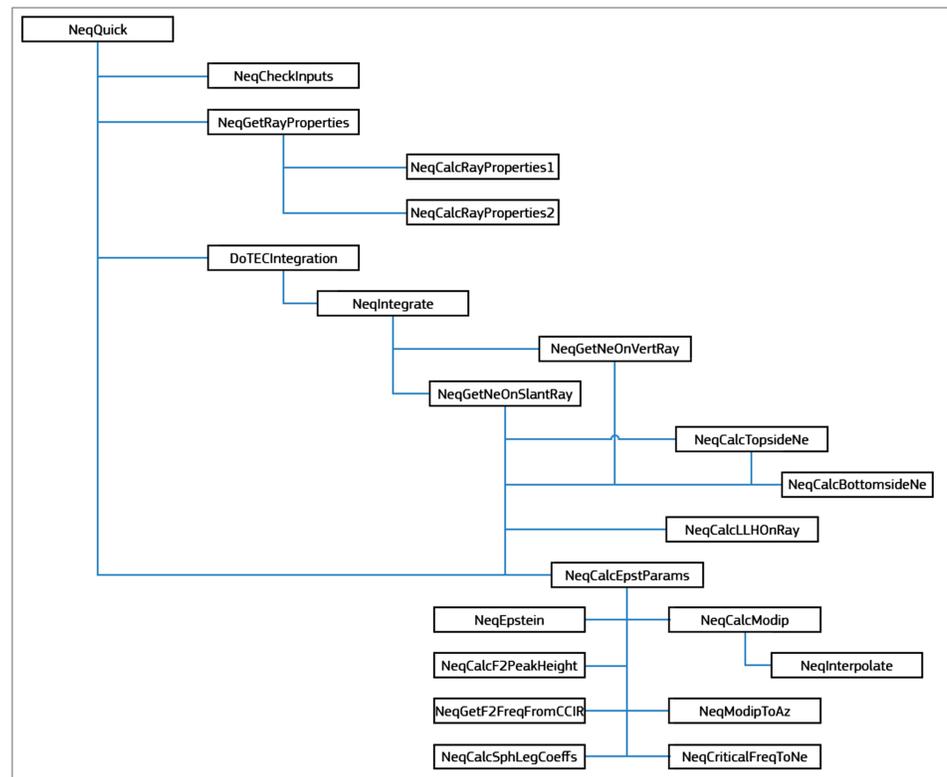


Figure 1. Overview of the NeQuick-G function hierarchy according to Figure 10 in RD.

Two strategies are necessary, one is to achieve a good precision while using a minimum number of points in the LoS between the receiver and the satellite and the other one is to avoid a re-evaluation of a point already used before. The presented optimization is purely mathematical and the rationale to make this decision is explained in the following subsection.

1.1. Profiling NeQuick-G

The GNU profiler “gprof” (Fenlason, 2016) and “valgrind” (Nethercote & Seward, 2003) were used to analyze the performance of the JRC NeQuick-G implementation once programmed. The use of these profilers allowed us identifying the most demanding functions and most frequently used subroutines (see Table 1), which correspond to

1. NeqClipExp: It is basically an exponential function that always returns a valid output (see RD for a detailed description). Operationally, it cannot be allowed to obtain a non-valid value; therefore countermeasures need to be taken within the code.
2. NeqEpstein: Evaluates the Epstein layer function. Inside the integration loop to calculate STEC (see Figure 1: it lies inside the DoTECIntegration branch).
3. NeqJoin: Joins two functions guaranteeing continuous first derivatives at origin.
4. NeqInterpolate: Performs third order interpolation when calculating modified dip latitude (MODIP). Inside the integration loop to calculate STEC (see Figure 1: it lies inside the DoTECIntegration branch).

Table 1

Statistics of the Four Functions More Frequently Invoked in the NeQuick-G Algorithm

Number of calls	Function name
6981	NeqClipExp
3600	NeqEpstein
2700	NeqJoin
1130	NeqInterpolate

Note. Figures correspond to one run of NeQuick-G using one example of the vector tests in RD. Naming convention of the functions corresponds to the original nomenclature given in RD. Results from this profiling can be found in Aragon-Angel and Fortuny (2016).

Looking at the scheme reproduced in Figure 1, which gives an overview of the NeQuick Function Hierarchy, one can realize that the functions

appearing in Table 1 correspond to the STEC integration process. When applying the gprof profiler multiple times using different integration algorithms the final outcome of the profiling is that the occurrences of these various routines are basically integer multiples of the number of function calls in the integration. This insight justifies concentrating the optimization efforts on the integration itself with the following goal: Minimize function calls on the points of LoS receiver-satellite with respect integration.

This profiling exercise is justified as being good standard practice. However, in our present case it is only partially helpful. The main issue corresponds to counting how many points are evaluated using various integration runs comparing all integrating methods available. The summary of this is shown in Table 4 (section 6.4). It will be shown that the various integration methods differ considerably. In addition, since the dominating factor of the computational cost of the integration is the number of function calls, it is also independent from the hardware platform. Platform dependent optimization based on advanced compiler options have not been attempted in this study since NeQuick-G is not expected to play a major role in Desktop environments.

Following this lead, four methods have been explored to perform the one-dimensional numerical integration (or numerical quadrature) of TEC along the ray path from satellite to receiver, using the same tolerance settings as in RD:

1. **Gauss Quadrature (GAU):** The Gauss integration (Hildebrand, 1956) is the algorithm suggested in the theoretical part of RD (section 2.5.8.2.8). The basic idea of this algorithm is to calculate the function at optimized abscissas. This method is known to be precise for functions without singularities. For this work, GAU has been coded following instructions in section 2.5.8.2.8 of RD.
2. **Recursive Gauss-Kronrod Quadrature (G15):** The integration method Gauss-Kronrod (Laurie, 1997) is an alternative to the GAU method also provided in the RD (specifically in section F.2.6, where a detailed processing model of NeQuick-G is given in form of pseudo-code). The integral (in our particular case, STEC) value is firstly estimated using the Gauss rule using 7 points (hence G_7). Secondly, the Gauss-Kronrod rule provides a higher precision estimate using 15 Kronrod nodes (hence G_7-K_{15}). The difference between the two estimates provides an error estimate for the desired integral. For this work, G15 has been coded following instructions in section F.2.6 of RD. The main advantage of using a Gauss and Gauss-Kronrod pair is that the second rule of 15 points actually includes the 7 points evaluated in the previous Gauss rule. This means that the function values from the lesser accurate GAU computation can be reused (Knezevich & Radicella, 2004). Therefore, the Kronrod G_7-K_{15} adaptive quadrature method (hereafter named G15 in short) is computationally more efficient than GAU.
3. **Adaptive Gauss-Kronrod Quadrature (QAG):** The QAG quadrature algorithm is part of the GNU Scientific Library (GSL) and it is described in detail in Piessens et al. (1983) and Gonnet (2010). The integration region is divided into subintervals and, in every iteration, that subinterval with the largest estimated error is bisected. This procedure reduces the overall error rapidly: this is the reason why this method has been chosen to generate a reliable benchmark as it will be introduced in section 4.2. Note that this integrating method is not meant to be an eligible candidate to replace the existing algorithm in NeQuick-G since it requires too many sampling points that are not re-used. Therefore, compromising its velocity (it is known to be slow). For those interested on how QAG method has been configured for this particular study, the key of the GSL library used is `key = 6`; that is, `GSA INTEG GAUSS61` and official tolerances in the RD multiplied by 0.1 have been considered.
4. **Nonadaptive Gauss-Kronrod Quadrature (QNG):** QNG algorithm is a textbook implementation of the Gauss-Kronrod quadrature (M. Galassi et al., 2009). This algorithm uses the Gauss-Kronrod 10-point, 21-point, 43-point, and 87-point integration rules in succession. It is not recursive. If the desired tolerance is not reached in a first run, the number of nodes is simply increased. However, the function values already calculated are always re-used in the subsequent iteration.

The suggested optimization of this work is based on the QNG method as results in the following sections will confirm that is lossless in terms of precision.

2. NeQuick-G Basics

The NeQuick-G ICA has proven to be a useful tool to reduce the ionospheric error in satellite navigation. When compared to the Klobuchar model (Klobuchar, 1987) used successfully in GPS, NeQuick-G

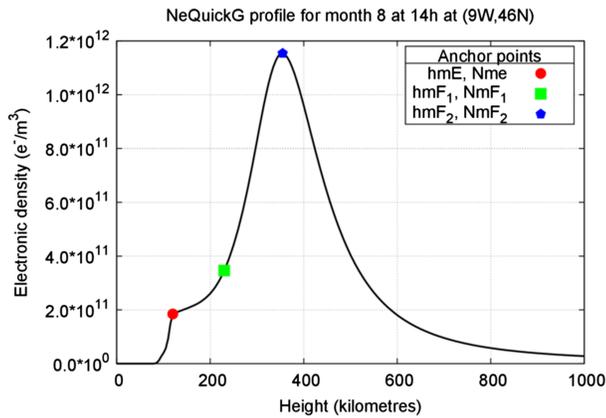


Figure 2. Electron density profile calculated using NeQuick-G for August, at receiver location 9° longitude, 46° latitude. Its anchor points are highlighted: maximum layer density of layers E, F₁, and F₂. From them, using interpolation methods, the whole profile is built.

outperforms Klobuchar by a 5% during maximum solar conditions (Rovira-Garcia et al., 2016). However, in this case, the gain in performance means a higher computational cost. This is not surprising since the Klobuchar model is based on roughly 10 formulas, while NeQuick-G needs to be described by more than 200 formulas, some of which use external matrices to compute and load different model coefficients.

The present contribution describes a first attempt at the JRC investigating whether the NeQuick-G ICA leaves space for some computational optimization. Unlike the competing ICA used in GPS, NeQuick-G uses a three-dimensional model of the Ionosphere based on the NeQuick model, the original profiler developed by Di Giovanni and Radicella (1990). The approach is straightforward: the electronic profile is calculated along the LoS from satellite to navigation receiver. To do so, the corresponding ionospheric parameters such as critical frequencies, maximum layer densities and their locations are calculated. These anchor points (Radicella & Leitinger, 2001) are joined together via various interpolation methods. In this manner, a mathematical function for the electronic density over height is formulated (see example of electron profile in Figure 2 built from

its corresponding anchor points). In a subsequent step, the electronic density is integrated along the LoS. The result of this integration yields the STEC that is proportional to the range error without correction. The main focus of this study is the selection of the integration method because it has a direct impact in the commitment of the computational resources and accuracy of the integral value.

2.1. MODIP: Modified Magnetic Dip

The ionosphere—even undisturbed—experiences diurnal, seasonal, solar cycle, and geographical variations. It is challenging to capture all these variations in a theoretical model. In NeQuick-G the ionospheric state is set to depend on the hour, the month, the solar flux (as equivalent of a sunspot number), and the magnetic field. These are the input set of parameters required to run NeQuick-G. Other input values are the positions of the GNSS receiver and the satellite. After initializing the input values, the second step is to compute the modified dip latitude (MODIP, μ) of the receiver (Rawer, 1963). MODIP takes into account that the geomagnetic field is not just determined by the geometry of the geosphere but also by the magnetic properties of the Earth. According to the RD, five regions have been defined based on their MODIP values to account for the ionospheric effects (see Figure 3).

In NeQuick-G the MODIP values are actually calculated from a 39×39 matrix describing the entire Earth as a longitude/latitude grid with 5° latitude steps and 10° longitude steps. The value used for a NeQuick-G

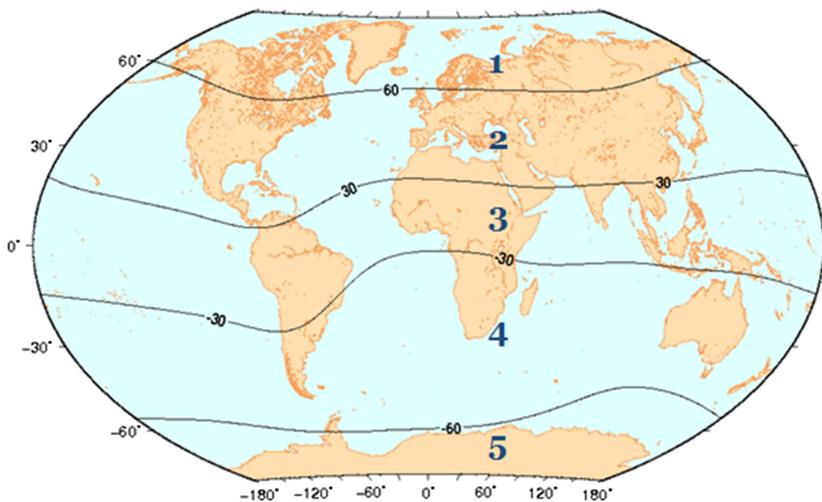


Figure 3. MODIP regions according to RD.

Table 2
Comparison of NeQuick-G Derived Parameters Versus Digisonde Ionogram Data for Rome on 25 June 2017 at 12 UTC Time

Parameter	NeQuick-G	Ionogram
Effective ionization level Az	66.73	SFI 80
Critical frequency of the F_2 layer in MHz (f_oF_2)	5.60	5.65
Critical frequency of the F_1 layer in MHz (f_oF_1)	4.43	4.32
Critical frequency of the E layer in MHz (f_oE)	3.17	3.26
M3000/ F_2 factor (3000-km MUF over f_oF_2)	3.10	3.39
Height of maximum F_2 layer density in km	243.17	255.8
Height of maximum F_1 layer density in km	181.59	152.1
Height of maximum E layer density in km	120.0000	96.4

Note. Digisonde Ionogram data available online from the Istituto Nazionale di Geofisica e Vulcanologia. Please note that the NeQuick-G version used for this intercomparison is the one fully compliant with RD.

calculus is an interpolation of neighboring points in the grid. This matrix is stored in a text file (modipNeQG_wrapped.txt). It is linked to the magnetic model of the Earth which undergoes updating in irregular intervals of few years. To our understanding, a production software would have a protocol for updating this matrix accordingly. Test versions of NeQuick-G are running with MODIP matrix available at its first release dated on year 2001 and calculated at a height of 300 km (Private communication, Galileo Service Centre, 2018).

2.2. Ionospheric Structure in NeQuick-G

NeQuick-G reproduces the ionospheric structure along the ray path between satellite and receiver. In terms of frequencies, two important ionospheric parameters are the maximum usable frequency (MUF) for a given distance and the maximum (called critical) frequencies being reflected at the ionosphere at vertical incidence. NeQuick-G uses as anchor points (i) the 3,000-km MUF (actually $M3000 = MUF3000/f_oF_2$);

(ii) the critical frequencies at layers: E , F_1 , and F_2 (f_oE , f_oF_1 , f_oF_2); (iii) the layer heights and thicknesses; and (iv) the electron densities.

NeQuick-G ignores completely the D layer because of its negligible share contribution, while the height of E layer is set constantly to 120 km as reasonable simplification following the original climatological NeQuick. The electron density in the remaining E , F_1 , and F_2 layers is described using the analytical Epstein function, presented as equation 3 in the RD. The different layer parameters get combined with suitable functions and as an overall result the electronic density of the ionosphere can be described for any point on Earth by NeQuick-G. The calculus for the E and F_1 properties (i.e., frequencies, electronic densities, heights, and thickness) is pretty straight forward using geographic position, ionization level, and zenithal angle of the Sun as a function of time. Instead, the F_2 and M3000 calculus is lengthy and it requires the use of a large set of coefficients of which there is one available for every month, rather there are two sets per file, one for f_oF_2 and one for M3000 value mentioned above. Both can be calculated with the same algorithm, which is based on the gamma function of the International Reference Ionosphere (IRI) (Bilitza, 1990).

2.3. Ionospheric State

The main NeQuick-G parameters to describe the state of the ionosphere are directly linked to the sunspot number and the Solar Flux Index (SFI). The latter is subject to a simple measurement of the energy transmitted by the Sun on microwaves at the wavelength 10.7 cm; the former is subject to astronomic observation. However, the best correlation with the ionosphere is not achieved by simply counting the spots; rather, it is also important how the spots are distributed, if they appear in groups their influence on the ionosphere is stronger. This is taken into account in the R12 sunspot number, often called Smoothed Sunspot Number (SSN), where smoothing means averaging a monthly value and the six months before and after the month of observation. The correlation between SSN and SFI is so strong that one can be converted to the other by a simple formula (ITU-R, 1999), also used in NeQuick-G. Earlier versions of NeQuick have used SSN and SFI parameters directly; this is meant by the so called “climatological NeQuick” as opposed to “NeQuick-G” for Galileo, where SFI is overridden by the “Effective Ionization Level” Az. While sunspot numbers or solar flux are Sun related (i.e., the same worldwide), Az is also linked to the geographic position via three broadcast coefficients and the MODIP value. Likewise, NeQuick-G uses a receiver position dependent sunspot number Azr called “Effective Sunspot Number.” Both values, Effective Ionization Level and Effective Sunspot Number, are internally calculated but basically “Effective Ionization” is used. This matter is dealt with exactly as suggested by RD. Therefore, NeQuick-G needs the above mentioned three coefficients that are transmitted in the navigation message of the Galileo constellation. Such coefficients have been used for 25 June 2017 to calculate an ionospheric profile over the location of the Rome digisonde at 12 UTC time to be compared against the ionogram data. Digisondes are well known for their continuous ionosonde measurements of high quality that make possible to derive ionospheric parameters of interest. See Table 2 that allows us to intercompare between ionogram data (actual real data) versus NeQuick-G predictions (modelled data) to have a flavor of how accurate NeQuick-G can be against reality.

In this particular comparison for Rome, the solar activity derived from the Galileo coefficients ingested in NeQuick-G is slightly underestimated with respect the real measurements: the real critical frequencies are higher and so are their corresponding layer heights. Note that the E layer in NeQuick-G is always defined to be at a height of 120 km (following RD), while its real height was 96 km in this particular case.

2.4. From Electronic Profile to STEC

The electronic profile is calculated based on the parameters outlined in the previous section. The ultimate goal of the NeQuick-G model is to provide a suitable estimate of how much time the radio signals get delayed while crossing the ionosphere. This goal is achieved by integrating the electronic density profile over the ray path between receiver and satellite. This integral yields the STEC, and it has a direct linear relationship to the range error of this path, that is, the overestimation of the distance. In the RD, G15 and GAU integration methods are recommended to perform such integration. The authors of this study have followed the directions in RD to implement them.

2.5. Limitations of the Model

The aforementioned description of the ionosphere is a reasonable approximation of its averaged state. However, the ionosphere is subject to highly disturbing events such as X-ray flares from the Sun or solar or magnetic storms. X-ray flares lead to an unusual strengthening of the D layer. Even the E layer may behave in an unpredictable manner, especially in the months after the spring equinox of the Northern hemisphere: detached clusters of ionization can become much stronger than the E layer itself and these clusters appear sporadically in time and space, the phenomenon being called E_s (sporadic E) and it is beyond the scope of NeQuick-G. Optical and X-ray flares are followed by Coronal Mass Ejections (CMEs) with an intense particle flow. When such particles reach the Earth, the magnetic field gets drastically modified and defeat any calculus or model. Another situation difficult to accurately model corresponds to sunset, where the decline of the ionization may lead to ionized spots or plasma bubbles, giving rise to equatorial scintillation. Other detrimental factors may be high latitude phase scintillation, direct disturbances caused by aurora phenomena or the presence of scattering related to meteorites. Under many adverse circumstances the NeQuick-G algorithm falls back to reasonable defaults that still account for a considerable part of the ionospheric error. The reduction in reliability is indicated by warning flags transmitted by the Galileo constellation. However, in the official NeQuick-G implementation, these flags are not exploited. For instance, it was suggested in (Aragon-Angel & Fortuny, 2016) to reduce the computational effort whenever these flags were not set by lowering the tolerance thresholds in the STEC calculations, that is, in the integrations.

3. Overview of the JRC Implementation of NeQuick-G

The need to apply the NeQuick-G model in real-time imposes that the application should run as fast as possible. An obvious choice has been the programming language C. This suggests certain implications in the algorithm like the extensive use of functions as main strategy to ensure modularity. The JRC NeQuick-G source code has been developed from scratch, without using the pseudo-codes from the RD and solely relying on the physics descriptions found in the first part of the official RD. This had implications in the naming conventions and the programming style but it helped to improve the RD itself by the detection of errors therein. Since this work was performed independently from other research groups, an added value was provided by delivering inputs for an errata sheet and eventually a new version of the RD was published via the official channels of the European Commission. An extensive set of test versions was implemented by having several versions of the main program while using the same library that act as a collection of modules and functions. These test functions are available to the public via a dedicated JRC web site: <https://nequick-g.jrc.ec.europa.eu/>. One of its apps shows all ionospheric parameters for arbitrary input. This is meant as aid for developers of new versions for new platforms.

4. Benchmark

In this study, a benchmark is considered to be a collection of test case, where in each case, the satellite-receiver pair time, location of both satellite and receiver, and STEC value along the LoS are known.

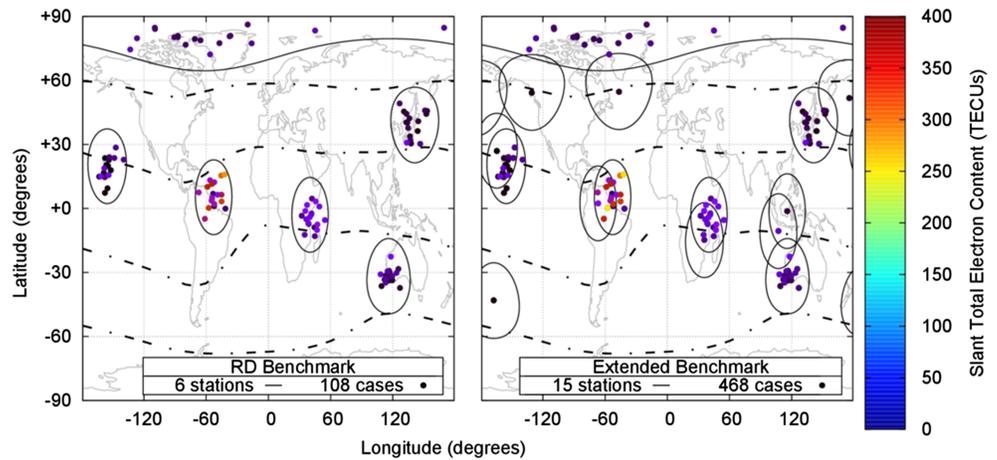


Figure 4. Left and right benchmarks used in the RD and the extended version used by JRC, respectively. The color dots represent the STEC at the Ionospheric Pierce Points at an altitude of 500 km. The solid lines enclose the projection area at a 500 km of altitude, for elevation angles higher than 5°. The dashed lines depict MODIP latitudes, from top to bottom: +60, +36, -36, and -60.

4.1. Benchmark in the Reference Document

The NeQuick-G algorithm can be considered to be a black box with input ports for universal time, month, locations of the ray path, and three coefficients describing the state of the ionosphere (a_0 , a_1 , and a_2 ionospheric coefficients that can be obtained from the Galileo navigation message) and it has a single output port that outputs the STEC as a measure of the range error along the ray path. This input-output mapping is found Annex E: Input/Output Verification Data of RD. There are a total number of 108 examples divided into high, medium, and low solar activity following this strict order (36 examples per case), where the given values of a_0 , a_1 , and a_2 are the ones that drive the examples to belong to each solar activity case.

When testing different quadrature algorithms to try and verify the data from Annex E of RD, it was noticed the following: While the geographic locations of the receiver stations in Annex E are well distributed (see Figure 4), all calculations correspond to the month of April. Moreover, not a single example is provided for a satellite located directly over a receiver. Such a vertical case forks to a different (simplified) algorithm, which should be tested too. Therefore, the vertical case uses a different (less complicated) algorithm. However, this vertical case is not considered in the benchmark from Annex E. For this reason, it was decided to complement the official benchmark to be sure all potential cases of pair satellite-receiver were accounted for.

4.2. JRC Extended Benchmark

In order to get an impartial, more general and extended test bench, new examples were defined and calculated. The geographic locations were maintained from Annex E in the RD but examples were added for three other months, one close to the opposite equinox and two more in months far from equinox. The reason it is three months is because of the season-dependent parameter ($seas$) introduced in the implementation of the NeQuick-G according to RD. This parameter “ $seas$ ” is a function of the month of the year as follows:

```

if month=1,2,11,12 then seas=-1
if month=3,4,9,10 then seas=0
if month=5,6,7,8 then seas=1

```

Therefore, the 12 months of a year are collapsed into three potential values for $seas$. In this way, the authors make sure that all potential values of $seas$ are tested. In contrast, notice that in the original benchmark provided in the RD, only one of the potential values of $seas$ is tested, the one corresponding to month of year 4.

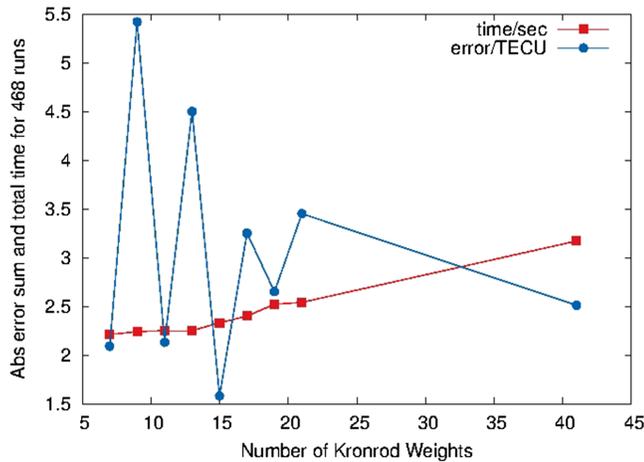


Figure 5. Error (TECU) and user run time (s) as a function of the number of Gauss-Kronrod nodes.

algorithms and their variants with respect to the reference STECs obtained with QAG in the JRC extended benchmark. The *Abs* value that corresponds to the sum of the absolute values of all deviations in TECU from the JRC benchmark and, the *Squ* value that stands for the sum of the squares of these 468 deviations:

$$Abs = \sum_{n=1}^{468} |STEC_n^{INT} - STEC_n^{QAG}|$$

$$Squ = \sum_{n=1}^{468} (STEC_n^{INT} - STEC_n^{QAG})^2$$

where *INT* indicates the specific integration method used. *Abs* has been chosen in this particular way (by taking the absolute value) in order to avoid compensating with the potential negative results from the inter-comparisons. Indeed, using *Abs* one loses sight of whether the integration routines over or under model the STEC with respect the reference but we make sure that the worst possible difference between models is accounted for.

At hardware level, times and deviations have been calculated in the JRC extended benchmark with two different high-end standard PCs running different Linux distributions: Stable Debian on a 10-year-old machine with a CPU Intel(R) Xeon(R) CPU 5110 at 1.60-GHz model 15 and Stable Ubuntu on a recent machine equipped with a CPU Intel(R) Xeon(R) CPU E5-2620 v3 at 2.40-GHz model 63.

The entire set of data was run thirty times on each computer and the mean time was selected as most representative for the elapsed time.

5. Optimization Attempt: Different Sampling in Gauss-Kronrod Recursion

The first optimization attempt tackled G15, checking whether a different number of Gauss-Kronrod nodes would show different results when compared to the recommended 15 Kronrod and 7 Gauss nodes in the RD. As a rule, the number of Kronrod nodes always corresponds to the number of Gauss nodes (*N*) multiplied by two plus one ($2N + 1$). In a first step of this study, alternative numbers of nodes to the pair 15-7 have been calculated by modifying accordingly the corresponding algorithm in the NeQuick-G implementation.

Interestingly enough, Gauss-Kronrod has shown to be pretty weak when having an even number of Gauss nodes but it is consistently better when the number of selected Gauss nodes is odd. The calculations show that there is minimum error when choosing 15 Kronrod weights and 7 Gauss weights. Notice that the calculation time goes moderately up when increasing the node number (see Figure 5). Therefore the design of the quadrature algorithm in NeQuick-G is optimal from the point of view of nodes selection, which is exactly the

Moreover, a dozen examples were added to account for the vertical case. The final benchmark corresponds to a total of 468 test cases as opposed to 108 in Annex E in RD.

At this point, it was needed to get reliable STEC reference values for this JRC expanded benchmark. To do so, a well-known and tested quadrature algorithm was used. The reference STEC values corresponding to the 468 examples were calculated using the algorithm QAG from the GSL strengthening the tolerance level by one order of magnitude with respect indications in RD.

4.3. Assessment Criteria

Two metrics have been used to assess the different quadrature methods analyzed for NeQuick-G in this study: on the one hand, the system speed, and on the other hand, the deviations from the JRC extended benchmark.

The system speed is calculated through the UNIX utility “time,” which measures the execution time when running the entire JRC benchmark with its 468 examples. In contrast, two metrics are used to quantify the deviations of the STEC produced by various candidate integration

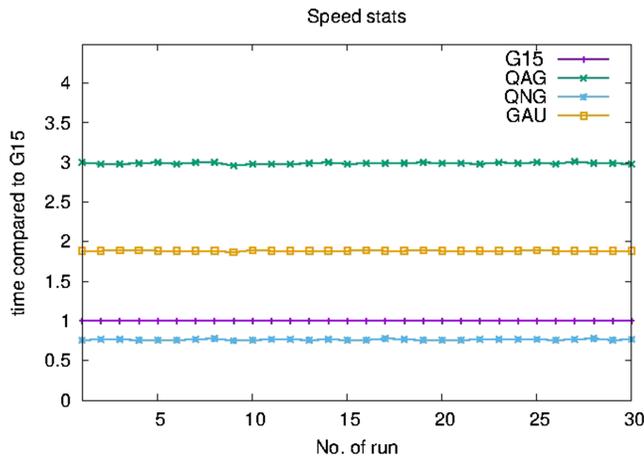


Figure 6. Elapsed times for thirty runs compared against G15 (official NeQuick-G) used as reference.

Figure 6 presents the elapsed time for each consecutive run using G15 as reference. See further results in Table 3. The times observed in the calculations depend mainly on the processor speed. Nevertheless, the results in terms of absolute differences and square errors with respect to the JRC benchmark are the same on both machines. This suggests that the overhead created by the use of GSL in QAQ and QNG is negligible at run time.

Table 3 shows that the time of NeQuick-G algorithm to execute the JRC extended benchmark is reduced by using the integration routine QNG instead of the two recommended integration methods in the RD, that is, GAU and G15. Numerically, QNG reduces 20.8% (14.6%) and 49.2% (45.9%) the time of G15 and GAU, respectively, when using the 2.4 (1.6)-GHz computer. Regarding the deviations from the reference STECs generated with QAG, QNG reduces the *Abs* 65.2% and 22.2% of GAU and QNG, whereas QNG reduces the *Squ* of GAU and G15 by 90%.

In the fastest machine, the run process was analyzed using the profiler valgrind with the “massif-visualizer” flag. The summary of this analysis showed a constant Random Access Memory (RAM) load of roughly 16 MB over the entire lifetime of the process. However, this NeQuick-G implementation is not a challenge for RAM size; any current handheld device can handle a 16-MB RAM load. The overall computational log file does depend mainly on the number of operations, which is eventually closely correlated to the number calls of the ionospheric model. This underlines the quest to find a quadrature method reducing this number of calls.

6.2. Comparison Details Against the JRC Extended Benchmark

As previously stressed out, in this optimization study, the priority was always to try not to deviate from the official results obtained with the G15 and given as benchmark in the RD. Therefore, tolerances in all quadrature methods were kept tight to avoid diverging from the RD. Figure 7 shows the deviations (in TECU) of all tested methods with respect to G15 in the JRC extended benchmark. All tested methods present an agreement within ± 0.2 TECU from G15 with a standard deviation of 0.013 TECU. However, observing the maximum deviation peaks of QAG, QNG, and GAU, it can be inferred that these three quadrature methods encounter similar difficulties to integrate some particular cases, whereas G15 does not.

choice of the authors of the reference algorithm in RD. Note that in this case, the absolute error sum is not zero since the reference STEC used are not the ones from running NeQuick-G as in RD but the ones from the JRC extended benchmark. Thus, there is no gain in varying the number of nodes since the implemented choice is confirmed to be the preferable choice in terms of precision and speed.

6. Analysis of the Performance for the Different Quadrature Methods

6.1. Speed and Deviation Results

The time required to process the entire JRC extended benchmark has been measured using the four quadrature methods: QAQ, G15, QNG, and GAU. In order to rule out that time may be affected by other running machine processes not related to NeQuick-G, we processed the JRC Benchmark 30 times for every quadrature process. Regarding system speed, there were actually no significant differences between different runs.

To try to better understand these results, QAG was used as reference instead of G15. Figure 8 shows the deviations of the quadrature methods with respect to QAG. In this case, what one can observe is that G15 is the method presenting the highest deviation. Furthermore, QNG is highly compatible (biggest discrepancies of the order of 0.03 TECU) and GAU presents some deviations but lower than those in Figure 7 when G15

Table 3
Speed and Deviations of Different Quadrature Methods Analyzed for NeQuick-G After Running 30 Times the JRC Extended Benchmark of 468 Test Cases

Method	Mean execution time (s)		Deviations (TECU)	
	1.6 GHz	2.4 GHz	<i>Abs</i>	<i>Squ</i>
GAU	3.68	2.40	3.53	0.1437
G15	2.33	1.54	1.58	0.1064
QAG	7.34	3.89	0.01	0.0001
QNG	1.99	1.22	1.23	0.0129

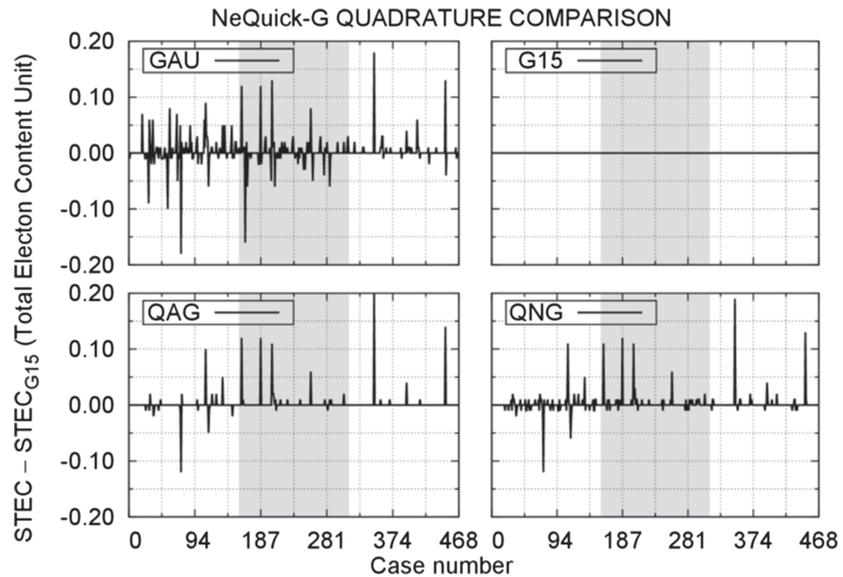


Figure 7. Deviations (in TECU) in the JRC extended benchmark of NeQuick-G using the different proposed quadrature methods. Case number goes from high, mid (grey background) to low solar activity.

was used as baseline. Thus, it seems that G15 (implemented as described in RD) is the outlier method with respect to the others. It was required to understand this result. For this reason, the pseudo-code included in Annex F of the RD was thoroughly followed to find a potential reason for the different behavior of G15 from the rest of methods.

Figure 9 depicts the pseudo-code corresponding to the NeQuick-G integrating module. The criteria to accept a quadrature result are controlled by the Kronrod tolerance (pdKronrodTol, type double, units N/A). In the first conditional sentence, there is an OR condition: either the absolute value of the relative error is smaller or equal than the tolerance OR the absolute value of the absolute error is smaller or equal than the tolerance.

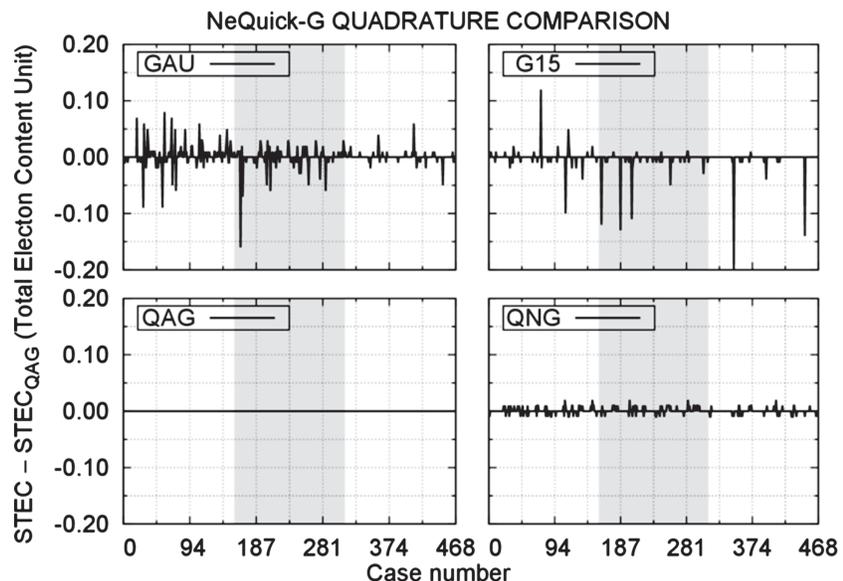


Figure 8. Deviations (in TECU) with respect QAG using the different proposed quadrature methods. Case number goes from high, mid (grey background) to low solar activity.

```

Check if the result is within tolerance
if1(fabs(intk - intg)/intk) <= tolerance || fabs(intk - intg) <= tolerance
    Result is within tolerance so set return value = intk
else if1 current level = MaxRecurse (max recursion level reached)
    Can do not further integration
    Set return value = intk as best guess
else1
    Result is not within tolerance
    Split portion into two equal halves (from dh1 to dh1+h2 and from dh1+h2 to dh2 with h2=(dh2-dh1)/2) and call
        NeqIntegrate on each new portion
    Sum the return values from the two NeqIntegrate calls and set as return value
end if1
    
```

Figure 9. Snapshot of the pseudo-code for the NEQINTEGRATE.c module in section F.2.6.1. Annex F of the reference document (European Commission, 2016).

Our first concern was regarding this tolerance: what are the units of this parameter? According to the first part of the conditional sentence it should be unitless. According to the second part of the conditional sentence, it should have the same units as *intk* and *intg*, in this particular case TECU. In the former, it corresponds to a relative error; in the latter, to an absolute error. Looking at the references (Piessens & De Doncker, 1977a) and (Piessens & De Doncker, 1977b) where the Gauss-Kronrod method is thoroughly explained among others, it was possible to check that the authors defined two different tolerances: an absolute tolerance (*epsabs*) and a relative tolerance (*epsrel*) and the convergence was controlled by taking the maximum of the two values:

$$\text{abs}(i - \text{reslt}).\text{le}.\text{max}(\text{epsabs}, \text{epsrel} * \text{abs}(i))$$

where *i* stands for the integral of a given function *f* over the interval (a,b) and *reslt* is the resulting value from the numerical method used. By rewriting this condition one can obtain:

$$\text{abs}(i - \text{reslt}).\text{le}.\text{max}(|i - \text{reslt}|, |i - \text{reslt}|/i * \text{abs}(i))$$

which is totally compatible with our findings. Therefore, to our understanding there should be some order of magnitude of difference among these two tolerances while in the implementation following RD both have collapsed to only one. To test this hypothesis, a modification of G15 was run considering the relative error being 10 times smaller than its absolute tolerance. Figure 10 depicts the new results, where the label G15_A corresponds to this proposed adaptation of the integration routine present in the ICA pseudo-code. Now QAG and G15_A are totally compatible with minor discrepancies. At the light of these results, the Gauss method introduced in section 2.5.8.2.7 of RD was also revisited and equation 201 was modified to account for the absolute error instead of the relative error. Results are found under the GAU_A label. In this case, the consistency of the deviations improves with respect to QAG. The tolerance of the QNG method was also tightened under the QNG_A label; however, the results did not vary, indicating its robustness.

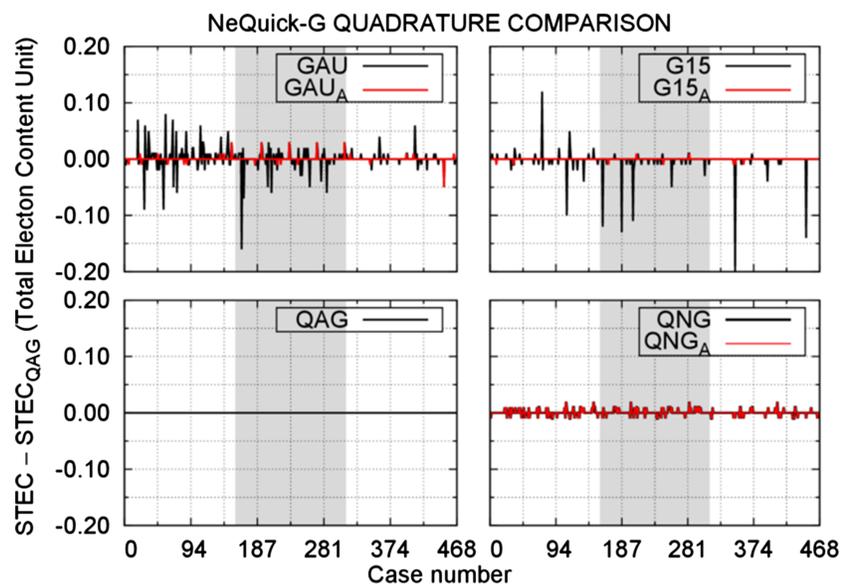


Figure 10. Deviations (in TECU) with respect to QAG using the different proposed quadrature methods with the adaptation of the tolerance control for G15, GAU, and QNG. Case number goes from high, mid (grey background) to low solar activity.

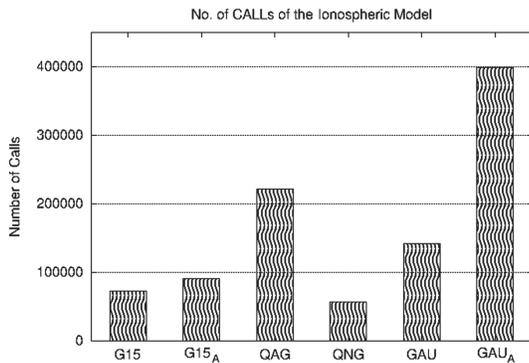


Figure 11. Number of call of the ionospheric model when using different quadrature methods.

At the light of these results, G15 and GAU (as described in RD) become more compatible with the rest of quadrature routines when using the absolute and relative tolerances accordingly in G15_A and introducing the absolute error in GAU to control the convergence of the integrals to be calculated (GAU_A). The authors recommend revisiting the convergence control of the integration routines introduced in the RD following the approach in this study. For completeness, the number of calls of the ionospheric model using each explored quadrature method is provided in Figure 11.

6.3. Some Comments Regarding the Tested Algorithms

At the light of the results presented in the previous subsections, some conclusive remarks are provided for each of the studied quadrature methods:

GAU Gauss Method (according to section 2.5.8.2.7. of the RD): This simple algorithm is slower and yields more deviations from the JRC extended

benchmark (see Table 3). The slowness is caused by the need to calculate too many function values (see Figure 11), which explodes when the double tolerance is used. Therefore, it is not recommended according to the metrics of this study.

G15 Recursive Gauss-Kronrod (according to section F.2.6 of the RD): While being both fast and precise (see Table 3), the Gauss-Kronrod performance is inferior to the QNG routine introduced previously. While the difference in the absolute error is moderate, the square sum is heavily deteriorated because this algorithm seems to suffer from occasional error increases as shown in upper right plot of Figure 8, zoom of which is provided in Figure 12. The major weakness of this algorithm is its behavior when a first integration fails to meet the tolerance. In that case the whole interval is split into two halves, within each half everything starts from scratch and all function values within the new interval have to be recalculated. The values previously calculated in this range are not used any more. This is the reason why the integrand function is called excessively when compared to QNG. This procedure defeats the intent of minimizing the number of function calls.

QAG Adaptative Gauss-Kronrod: The re-run of this algorithm brought no changes compared to the benchmark definition run with tighter tolerances. This means that the convergence of QAG is strong. However the computationally demanding requirements of QAG (see Table 4) flag this algorithm to be prohibitive for any production release of NeQuick-G.

QNG Nonadaptative Gauss-Kronrod: This algorithm from GSL is the most interesting and the most promising of all optimization candidates. It yielded more speed and more precision than any other implemented

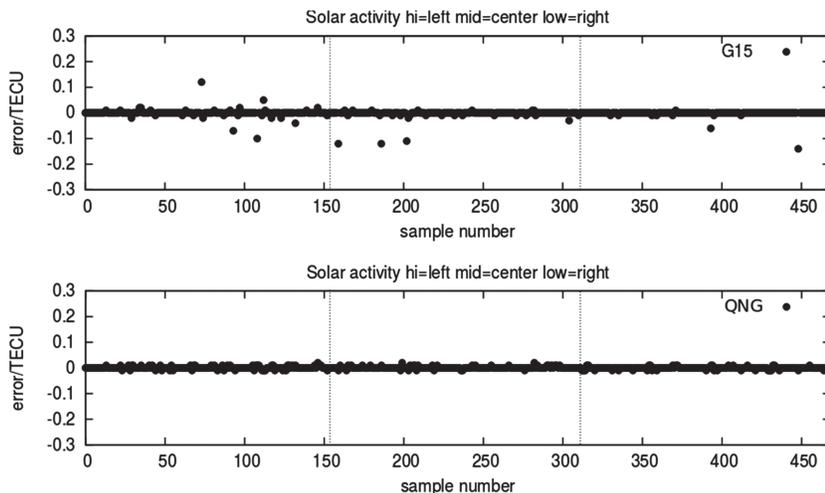


Figure 12. G15 versus QNG. Solar activity has been indicated for each plot: left, high solar activity; centered sample values, middle and right, low.

Table 4
Total Number of Integrand Calls Per Method and Reduction of function Calls (Percentage) With Respect to QNG After Running 30 Times the JRC Extended Benchmark of 468 Test Cases

Method	Total	%
GAU	141,312	-59.6
GAU _A	398,928	-85.6
G15	72,750	-21.5
G15 _A	90,765	-37.1
QAG	220,759	-74.1
QAG _A	313,418	-81.8
QNG	57,114	-

method in the current study. Numerically, the authors observed a reduction of 21.5% of the functions calls with respect G15 translates into 21% increase of computational speed for QNG versus G15. The tolerance parameter plays a minor role. It gives no space to fine tuning because there are only few sets of node numbers to choose from.

Despite the excellent speed and accuracy, the main disadvantage of QNG is the weak error estimate. Specifically, the error tolerances are largely overestimated. In order to force QNG to run in any case, the tolerance failure condition of this routine has been switched off using a GSL configuration parameter. This could be mathematically arguable but appropriate for our assessment. In the JRC clone of the QNG software, the algorithm was modified accordingly to just run and take the result the algorithm

provides. However, the bottom right plot of Figure 8, zoom of which is provided in Figure 12, clearly shows that this simplistic approach outperforms the G15 implementation.

6.4. Lessons Learnt About Quadrature

The different competing integration routines were called from an extended function library called “neqlib.c.” In the calculation of the integrated values, the test software opened a particular file that allowed counting the number of function calls. Each of the 468 examples needed as many function calls as shown in Table 4. Note that these numbers were retrieved over the entire benchmark set after 30 runs, not just over one run in order to have statistical meaning. These figures were not achieved by a profiler but by a manipulation of the source code that triggered an external counter at each run of the integration routine. Table 4 also shows, for each method, the reduction (in %) of integrand calls with respect the proposed optimization method in this study, QNG.

From Table 4 the slowness of each algorithm can be inferred from the total number of function calls needed to calculate the TEC Integral. The G15 algorithm presented in section F.2.6 of the RD defeats the principle of the expensive function because at any recursion a new set of nodes (function calls) has to be calculated. Instead the QNG algorithm just calculates additional nodes while re-using the old ones. The use of the non-adaptive Gauss-Kronrod Code reduces the number of G15 calls.

7. Some Comments About Found Issues

For low solar activities, like those listed in the validation table, F1 critical frequency can become negative, which is physically impossible. For the current study, some safeguards have been put in place to avoid calculating the square root of a negative number, that otherwise interrupt the execution of the program.

8. Conclusions and Follow-Up

This study presents an optimization attempt from an algorithmic point of view of the Galileo ionospheric correction algorithm, NeQuick-G, performed by the Joint Research Centre (JRC). By profiling the original NeQuick-G as developed following the RD, it was revealed that the algorithm performing the integrations held the key to improve the performance in terms of processing load and time. This has been the guideline followed at the JRC for this optimization study.

During the first stages of this study it has been confirmed that, initially, the original version of NeQuick-G detailed in the RD was already optimized thoroughly, especially the choice of the basic Gauss-Kronrod Algorithm using 7 Gauss weights and 15 Kronrod weights, which is an optimal choice as corroborated by our findings.

However, the use of the standard Nonadaptive Gauss-Kronrod algorithm (QNG) as implemented in the GNU Scientific Library has shown superior results not only in precision but also in calculation speed. The new proposed integration QNG method for NeQuick-G is able to speed calculations up to 21% and 49% with respect the two integration algorithms officially recommended with no losses in absolute error. The main vector of the reduction of computational cost is simply the reduction of the number of function calls, as the QNG integration uses the function values from the previous call. The price to pay is that the margin

for fine tuning is greatly reduced on the QNG algorithm. Besides, the tolerance setting of QNG has no other influence than choosing up to 4-point models.

Further optimization may still be possible but this requires a clear understanding what the current bottlenecks in the integration of NeQuick-G in GNSS receivers are. In this context, establishing a dialog with the community of receiver manufacturers and standardization groups is absolutely needed.

During the review process of this manuscript, the authors discussed with the European Space Agency the findings regarding the tolerances in G15, that is, the relative and absolute tolerances should be different. It was acknowledged that it was needed to be fixed in the RD. The following steps were recommended to be taken, which are already being undertaken:

1. Contact the European Commission to update of the current version of the RD.
2. Contact the European GNSS Agency for evaluating the impact in the Galileo Ground Stations and at user level.
3. Submit the current paper explaining the actions taken by Galileo to correct this issue.

Therefore, the thorough analysis performed in this work has led to some concrete steps toward the revision and update of the RD.

Data Availability Statement

The web version of Nequick-G and its test functions are available to the public via a dedicated JRC website: <https://nequick-g.jrc.ec.europa.eu/>. The source code of NeQuick-G developed by JRC will be distributed through the European GNSS Service Centre web portal. Digisonde Ionogram data are available online from the Istituto Nazionale di Geofisica e Vulcanologia at www.eswua.ingv.it/.

Acknowledgments

The first draft of the manuscript was made during a short visit to Joint Research Centre within the Horizon 2020 Marie Skłodowska-Curie Individual Global Fellowship 797461 NAVSCIN.

References

- Aragon-Angel, A. & Fortuny, J. (2016). Exploiting Galileo ionospheric disturbance flags to boost NeQuick. International Conference on Localization and GNSS (ICL-GNSS), pp. 1–6, June 2016
- Bilitza, D. (1990). International Reference Ionosphere 1990. NSSDC/WDC-A-R&S 90-22, Greenbelt, MD
- Di Giovanni, G., & Radicella, S. (1990). An Analytical Model of the Electron Density Profile in the Ionosphere. *Advances in Space Research*, 10(11), 27–30. [https://doi.org/10.1016/0273-1177\(90\)90301-F](https://doi.org/10.1016/0273-1177(90)90301-F)
- European Commission (2017). European Union Public Licence (EUPL), version 1.2 https://joinup.ec.europa.eu/sites/default/files/custom-page/attachment/eupl_v1.2_en.pdf
- European Commission (2016). European GNSS (Galileo) Open Service-Ionospheric Correction Algorithm for Galileo Single Frequency Users. Issue 1.2 https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo_Ionospheric_Model.pdf
- Fenlason, J. (2016). GNU gprof. Retrieved from: <https://sourceware.org/binutils/docs/gprof/>
- Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., et al. (2009). *GNU Scientific Library Reference Manual*, (3rd ed.). ISBN: 0954612078. Godalming, United Kingdom: Network Theory Limited.
- Gonnet, P. (2010). Increasing the reliability of adaptive quadrature using explicit interpolants. *ACM Transactions on Mathematical Software*, 37(3), 26:1–26:32. <https://doi.org/10.1145/1824801.1824804>
- Hildebrand, F. B. (1956). *Introduction to Numerical Analysis*, (pp. 323–325). New York: McGraw-Hill.
- ITU-R (1999). Choices of indices for long-term ionospheric predictions. Recommendation ITU-R P.371-8
- Klobuchar, J. (1987). Ionospheric Time-Delay Algorithm for Single-Frequency GPS Users. *IEEE Transactions on Aerospace and Electronic Systems*, 23(3), 325–331. <https://doi.org/10.1109/TAES.1987.310829>
- Knezevich, M., & Radicella, S. M. (2004). Development of an ionospheric NeQuick model algorithm for GNSS receivers, Proceedings of ESA workshop on satellite navigation user equipment technologies (NAVITEC) Noordwijk, The Netherlands, 8-10 December 2004
- Laurie, D. (1997). Calculation of Gauss-Kronrod quadrature rules. *Mathematics of Computation of the American Mathematical Society*, 66(219), 1133–1145. <https://doi.org/10.1090/s0025-5718-97-00861-2>
- Nethercote, N., & Seward, J. (2003). Valgrind: A Program Supervision Framework. *Electronic Notes in Theoretical Computer Science*, 89(2), 44–66. [https://doi.org/10.1016/S1571-0661\(04\)81042-9](https://doi.org/10.1016/S1571-0661(04)81042-9)
- Piessens, R., & DeDoncker, E. (1977a). A subroutine package for automatic integration. Part I: general purpose quadrature. TW Reports, Report No. TW37, Leuven, Belgium: Applied Mathematics and Programming Division, K.U.Leuven
- Piessens, R., & DeDoncker, E. (1977b). A subroutine package for automatic integration. Part II: special purpose quadrature. TW Reports, Report No. TW38, Leuven, Belgium: Applied Mathematics and Programming Division, K.U.Leuven
- Piessens, R., Doncker-Kapenga, E., Überhuber, C. W., & Kahan, D. K. (1983). *Quadpack: A Subroutine Package for Automatic Integration*, Springer Series in Computational Mathematics edition, Berlin-Heidelberg, Germany: Springer. <https://doi.org/10.1007/978-3-642-61786-7>
- Private communication, Galileo Service Centre (2018). ID ticket #569
- Radicella, S. M., & Leitinger, L. (2001). The evolution of the DGR approach to model the electron density profiles. *Advances in Space Research*, 27(1), 35–40. [https://doi.org/10.1016/S0273-1177\(00\)00138-1](https://doi.org/10.1016/S0273-1177(00)00138-1)
- Rawer, K. (1963). Propagation of Decameter Waves (HF Band). In B. Landmark (Ed.), *Meteorological and Astronomical Influences on Radiowave Propagation*, (pp. 221–250). New York, USA: Pergamon Press.
- Rovira-García, A., Juan, J. M., Sanz, J., González-Casado, G., & Ibáñez-Segura, D. (2016). Accuracy of ionospheric models used in GNSS and SBAS: methodology and analysis. *Journal of Geodesy*, 90(3), 229–240. <https://doi.org/10.1007/s00190-015-0868-3>