A Note on an inversion algorithm for vertical ionograms

R. Kenyi Takagui Perez[1]

[1] *Pontifical Catholic University of Peru*

A recent article [AIP Advances 14, 065034 (2024)][1] claims to have developed a quasi-parabolic inspired approach to find the plasma frequency profile from a given ionogram. The authors claim their algorithm fits the desired ionogram by producing an artificial plasma frequency profile. We find: (i) poor treatment of the mathematics of their results, (ii) the table's values are inconsistent with their proposed method as many of the consecutive quasi-parabolic layers don't have a continuity point, (iii) their algorithm didn't account for the case of anti-quasi-parabolic layers. Because of the number of errors in the paper and considering the potential of the underlying idea, we present a much clearer exposition of a multi-quasi-parabolic construction of a plasma frequency profile for the E and F layers. A parabolic profile is assumed for the E layer and the F layer is approximated by a series of concatenated quasi-parabolic layers where the continuity of the curve is preserved by assuming a common point between consecutive quasi-parabolic layers where the derivative is the same. The final profile is within the 10 km error. The CPU time required for the inversion is near 180 s. The inversion algorithm input is assumed to be an already scaled ionogram and the critical frequency for the E layer.

I. INTRODUCTION

Most of our knowledge about the ionosphere comes from ionogram records. These $h'(f)$ records give the apparent or virtual heights of reflection $h'(f)$ of a vertically transmitted radio wave, as a function of the wave frequency f. This paper aims to retrieve the electronic density profile for the measured ionogram.

The analysis of ionograms consists basically of converting an observed $h'(f)$ curve, which gives the virtual height of reflection h' as a function of the wave frequency f, into an $N(h)$ curve giving the variation of the electron density N with height h. These two curves are related by

$$h'(f) = \int_0^{h_r} \mu' dh, \tag{1}$$

where the group refractive index μ' is a complicated function of f, N and the strength and direction of the magnetic field. The height of reflection, h_r, of the wave of frequency f depends on f, N, and (for the extraordinary ray only) the strength of the magnetic field.

Previous efforts in solving this ill-posed problem include lamination techniques[4], least-squares polynomial analysis approximation[3,5], and ray tracing[6]. There is no analytic solution of 1, giving $N(h)$ in terms of $h'(f)$. Consequently, the first method of analyzing ionograms was to assume various model $N(h)$ curves and to compare the $h'(f)$ curves calculated from these models with those observed experimentally. Later Budden *et al.*[7] suggested an $N(h)$ model consisting of a large number of linear segments. The integral in 1 can then be replaced by a finite sum and the height increments of the successive segments are determined from the virtual heights of the waves reflected at the ends of the segments. This gives the "lamination" method of analysis which is widely used at present. It gives good accuracy when a large number of segments are used, but the calculations then become rather lengthy, and an electronic computer is generally employed. On the other hand, Reinisch *et al.* used Chebyshev polynomial methods to approximate the F layer to obtain a more efficient process for the analysis of ionograms.

We present an improved treatment of the ideas presented in Niu *et al.*[1] paper using multi-variate-quasi-parabolic layers to create a model inversion algorithm to approximate the F-layer plasma frequency profile. The parabolic model of the ionosphere was introduced by Forsterling and Lassen[8], who showed that radio-ray trajectories in this model could be readily calculated if a few reasonable approximations were made. As implied by the name, the model is defined by the equation of a parabola in electron density versus height. In the notation which is most convenient for our purposes, the parabolic layer is

$$N_e = \begin{cases} N_m \left[1 - \left(\frac{r - r_m}{y_m}\right)^2\right], & r_b < r < r_m + y_m \\ 0, & \text{elsewhere} \end{cases} \tag{2}$$

where:

N_e = electron density, having maximum value N_m
r = radial distance from earth center (height + earth radius)
r_m = value of r where $N_e = N_m$(h_m + earth radius)
r_b = value of r at the layer base $= r_m - y_m$
y_m = layer semithickness

We wish to point out that a very slight modification to the parabolic model permits the derivation of *exact* equations for ray-path parameters. This modified parabolic ionosphere will be termed the "quasi-parabolic" or, more simply, the "QP" model. The exactness of the resulting ray equations probably provides a negligible benefit in many of the more approximate applications of the parabolic layer. However, if one wishes to investigate

small differences between rays that are nearly identical, the use of the QP model will eliminate any suspicion that differences are introduced by the parabolic layer approximations. Also, a very important application for the QP model is the service it can provide as a primary standard of accuracy for testing more versatile ray-tracing methods.

These notes are meant to be pedagogical; most computations are performed in detail as we found out many of the steps and writing of the paper[1] with the original idea were flawed. In addition, many software programs available to perform the inversion of ionograms are closed, i.e. the source code is hidden in binaries or simply inaccessible elsewhere. Our overall purpose is to help bring community attention and to soften the introduction for newcomers to what we think is an interesting problem in radio science.

II. MODEL

We make a QP layer the basic unit that will be used to model split sections of the F layer and the entirety of the E layer. We define a QP layer as

$$f_{ni}^2 = f_{ci}^2\left[1 \pm \left(\frac{r - r_{mi}}{y_{mi}}\right)^2\left(\frac{r_{bi}}{r}\right)^2\right] \tag{3}$$

It can also be written as

$$y_i = f_{ni}^2 = a_i \pm b_i\left(1 - \frac{r_{m_i}}{r}\right)^2 \tag{4}$$

where $y_i = f_{ni}^2$, $a_i = f_{ci}^2$, and $b_i = a_i\left(\frac{r_{bi}}{y_{mi}}\right)^2$. There will be two kinds of QP layers: QP_i^+ and QP_i^-. Each QP layer is parameterized by three numbers: a_i, b_i, and r_{mi}.

A. What information do we have?

We assume we count with the autoscaled ionogram virtual heights for frequencies $f_i \in [0, f_F]$ with arbitrary strides, and also the frequency position of the E layer critical frequency f_E.

B. E Layer

It is well known that the E-layer virtual heights per frequency f can be modeled as

$$h'(f) = r_b + \frac{1}{2}y_{mE}\frac{f}{f_E}\ln\left(\frac{f_E + f}{f_E - f}\right) \tag{5}$$

Where $r_b = r_{mE} - y_{mE}$ is the height where the ionogram starts. We impose r_b to be the starting height of the plasma frequency profile too.

We use a brute-force approach to find the best parameters for the E-layer. Since we already know the value of f_E, we need to find r_b and y_m through a simple two-for loop. By each probe pair (r_{mE}, y_{mE}) we compare the produced virtual heights $h'(f)$ with the equation 5 versus the original ionogram. The probe pair with the smallest error difference is the one we take. The metric used for measuring the error we use is the root-mean-squared error. See Fig 1 to note the agreement between the measured and the artificially produced ionogram.

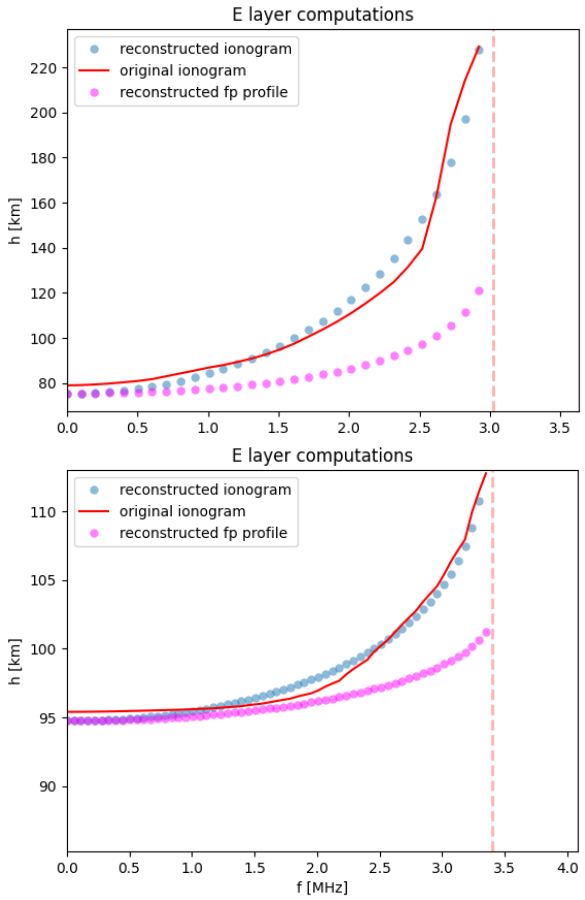


FIG. 1. (top) in blue dots the reconstructed ionogram for the data set 1 and 2, refer to Appendix 1. The E-layer critical frequency is set to $f_E = 3.0$ and $f_E = 3.4$ for the data set 1 and 2, correspondingly.

C. QP_i Layers

Next, we approximate the F layer by concatenating several QP layers while alternating between QP^- to QP^+ and QP^+ to QP^-. We start the F layer with a QP^+ layer. We ensure the continuity of the plasma frequency profile between QP_i and QP_{i-1} by making its

derivatives equal at r_i.

$$\left.\frac{dy_i^{\pm}}{dr}\right|_{r=r_i} = \left.\frac{dy_{i-1}^{\mp}}{dr}\right|_{r=r_i} \quad \text{or} \quad \left.\frac{dy_i^{\mp}}{dr}\right|_{r=r_i} = \left.\frac{dy_{i-1}^{\pm}}{dr}\right|_{r=r_i}, \tag{6}$$

where r_i is the last considered data point in the QP_{i-1} layer corresponding to a plasma frequency f_{p_i}. From these relations we can get expressions for b_i and r_{mi}, dependent on a_i and the parameters of the QP_{i-1} layer.

Let us compute the derivatives for the different types of QP layers. For y^-

$$y^-_{i(r)} = a_i - b_i\Big(1 - \frac{r_{mi}}{r}\Big)^2 \tag{7}$$

its corresponding derivative is

$$y^-_{i(r)}{}' = -\frac{2b_i r_{mi}}{r^2}\Big(1 - \frac{r_{mi}}{r}\Big) \tag{8}$$

On the other hand, for y^+

$$y^+_{i(r)} = a_i + b_i\Big(1 - \frac{r_{mi}}{r}\Big)^2 \tag{9}$$

its derivative is

$$y^+_{i(r)}{}' = \frac{2b_i r_{mi}}{r^2}\Big(1 - \frac{r_{mi}}{r}\Big) \tag{10}$$

Then, as we mentioned, there will be two cases: QP^- to QP^+ or QP^+ to QP^-. First, for the case of QP^-_{i-1} to QP^+_i, to find the dependence of parameters b_i and r_{mi} with the ones of QP_{i-1} and ensure the continuity of the curve, we equate $y^-_{i-1,(r_i)}{}' = y^+_{i,(r_i)}{}'$. r_i is the point where both curves meet, computationally speaking r_i is the last point we took from the curve QP_{i-1}. We find

$$r_{mi} = \frac{r_i^2 y^-_{i-1,(r_i)}{}'}{2(y^-_{i-1(r_i)} - a_i) + r_i y^-_{i-1,(r_i)}{}'} \tag{11}$$

$$b_i = \frac{\Big[2(y^-_{i-1(r_i)} - a_i) + r_i y^-_{i-1,(r_i)}{}'\Big]^2}{4(y^-_{i-1(r_i)} - a_i)} \tag{12}$$

Finally, for the case of QP^+_{i-1} to QP^-_i, we equate $y^+_{i-1,(r_i)}{}' = y^-_{i,(r_i)}{}'$.

$$r_{mi} = \frac{r_i^2 y^+_{i-1,(r_i)}{}'}{2(y^+_{i-1(r_i)} - a_i) + r_i y^+_{i-1,(r_i)}{}'} \tag{13}$$

$$b_i = -\frac{\Big[2(y^+_{i-1(r_i)} - a_i) + r_i y^+_{i-1,(r_i)}{}'\Big]^2}{4(y^+_{i-1(r_i)} - a_i)} \tag{14}$$

D. Algorithm

The algorithm starts with the computation of the E-layer, also known as QP^-_0 layer, with its parameters. Next, we can iteratively compute the following QP_i parameters since we will know the QP_{i-1} layer. We have established that the parameters of the QP_i layer can be calculated with the ones of QP_{i-1}, so at the end, r_{mi} and b_i will be left as a function of $a_i = f_{ci}^2$. To find f_{ci}, we use an exhaustive search algorithm, which means trying several values of f_{ci} by brute force. Assuming the last frequency data point appended to our final plasma frequency profile is f_L, if we are computing a QP^+ layer we will search for f_{ci} in the range of $[f_L + \epsilon, f_L + 2.0]$, and if, on the other hand, we are computing a QP^- the range where we search for f_{ci} will be $[f_L - 2.0, f_L - \epsilon]$. Then, per selected f_{ci} value we will take another `numt` frequency data points. For example, say f_L has an index `j` in the frequency data points from the original ionogram: `data_frq[ j ] = f_L`, then we take `data_frq[j : j + numt + 1]` and calculate their corresponding real heights and attach them into our temporal plasma frequency profile. Finally, we compute its corresponding ionogram with the forward model, which will be explained later, and compare it with the original. We chose the f_{ci} that produces the smallest error. The python algorithm implementing this can be found below.

```
def get_qp(qp_number):
    numt = len(QP['plasma_frequency'])
    if numt >= len(data_f): return
    tam = 1000

    # Set a lowerbound and upperbound depending on if it is a QP or anti-QP layer
    fc_lowerbound, fc_upperbound, qp_type = get_bounds_and_type(qp_number)
    possible_fc = np.linspace(fc_lowerbound, fc_upperbound, num=tam)

    # Variable to store the best set of parameters based on error L2
    store = {'error': 1e12, 'a_1':-1, 'b_0':-1, 'r_m1':-1, 'data_f':np.array([]), 'data_r': np.
        array([])}

    for fi in possible_fc:
        # compute QP parameters
        fi = fi * 1e6 # Hz
        ai = fi ** 2  # Hz^2
        prev_real = QP['real_height'][-1] * 1e3 # m
        rmi = find_rm(ri=prev_real, a=ai, ai1=QP['a_'+str(qp_number-1)] * 1e12, bi1=QP['b_'+str(
            qp_number-1)] * 1e12, rmi1=QP['r_m'+str(qp_number-1)] * 1e3, tipo=qp_type) # m
        bi  =  find_b(ri=prev_real, a=ai, ai1=QP['a_'+str(qp_number-1)] * 1e12, bi1=QP['b_'+str(
            qp_number-1)] * 1e12, rmi1=QP['r_m'+str(qp_number-1)] * 1e3, tipo=qp_type) # Hz^2

        # create a local plasma frequency profile
        tmp_f, tmp_r = QP['plasma_frequency'] * 1e6, QP['real_height'] * 1e3

        # if error is the smallest, add this data points
        i_data_r, i_data_f = np.array([]), np.array([])

        for idx_numt in range(numt_for):
            if numt+idx_numt>=len(data_f): break

            frq = data_f[numt + idx_numt] * 1e6 # Hz
            rh_frq = get_real_height(frq, ai, bi, rmi, prev_real, qp_type = qp_type)
            prev_real = rh_frq

            tmp_f, tmp_r = np.append(tmp_f,frq), np.append(tmp_r,rh_frq)
            i_data_f, i_data_r = np.append(i_data_f, frq), np.append(i_data_r, rh_frq)

        # produce artificial ionogram with new added points
        fp_ans , hv_ans = get_ionogram(h = tmp_r/1e3,fp = tmp_f/1e6, range_f = data_f[:len(tmp_f)])

        # compute initial error L2 | compare with original virtual heights from the ionogram's data
        errors = np.array([abs(hv_ans - data_r[:len(fp_ans)])])
        L2_error = L2_ERROR(errors)

        # If the error with the current f_c1 is lesser than the previous one, then we update
        if L2_error < store['error']:
            store['error'] = L2_error
            update_store_values()
            store['data_r'], store['data_f'] = i_data_r / 1e3,  i_data_f / 1e6

    # Pass stored information to the final solution reconstruction
    store_parameters_in_QP_map(qp_number)
    QP['plasma_frequency']   = np.append(QP['plasma_frequency'],store['data_f'])
    QP['real_height']        = np.append(QP['real_height'], store['data_r'])
```

Listing 1. Algorithm to get parameters for a QP layer

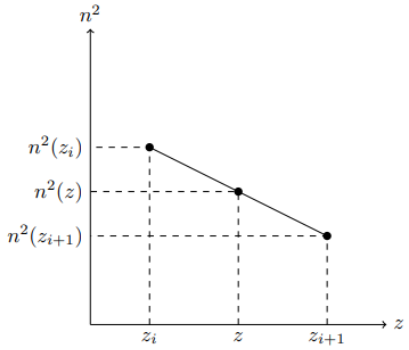
E. Forward model

Each time we add new frequency and real height data points to our plasma frequency curve, we need to test it with the original ionogram and calculate its error. To achieve this we need a forward model that can calculate an ionogram given a plasma frequency profile.

Another way to express the ionospheric virtual height reflection is

$$h(f) = \int_0^{z_r} \frac{dz}{n(z)} \tag{15}$$

and by looking at the expression of $n(z) = \sqrt{1 - \frac{f_p(z)^2}{f^2}}$, plus the condition that $f_p(z_r) = f$, we can foresee that the curve of $n(z)^2$ will look like an inverse function beginning in the coordinate $(0, 1)$ and then converging towards



$n(z)^2 = 0$ at $z = z_r$. If we zoom in on a curve segment, the ending points would be like the ones in the figure below. Using triangle relations we get

$$\frac{z_{i+1} - z_i}{n^2(z_i) - n^2(z_{i+1})} = \frac{z - z_i}{n^2(z_i) - n^2(z)} \tag{16}$$

where $n(z)$ can be extracted from this equation

$$n^2(z) = n^2(z_i) - (z - z_i)\frac{[n^2(z_i) - n^2(z_{i+1})]}{z_{i+1} - z_i} \tag{17}$$

and replace it in equation 15. Because the integrated is an inverse function the biggest contributions will come from the segment with z_i value closer to z_r. If we treat the integral from equation 15 by parts (imagine our line of integration is formed by N points), then there will be two different contributions, one coming from the last segment (z_1, z_r) and another from (z_0, z_1).

$$h(f) = \int_{z_1}^{z_r} \frac{dz}{n(z)} + \int_0^{z_1} \frac{dz}{n(z)} \tag{18}$$

First, we deal with the latter.

$$\begin{aligned} &= \sum_{i=0}^{N-2} \int_{z_i}^{z_{i+1}} \frac{dz}{n(z)} \\ &= \sum_{i=0}^{N-2} \int_{z_i}^{z_{i+1}} \frac{1}{\sqrt{n^2(z_i) - (z - z_i)\frac{[n^2(z_i) - n^2(z_{i+1})]}{z_{i+1} - z_i}}} dz \end{aligned} \tag{19}$$

Doing a change of variables $z' = z - z_i$ we are left with

$$\begin{aligned} &= \int_0^{z_1} \frac{dz}{n(z)} \\ &= \sum_{i=0}^{N-2} \int_0^{z_{i+1} - z_i} \frac{1}{\sqrt{n^2(z_i) - z'\frac{[n^2(z_i) - n^2(z_{i+1})]}{(z_{i+1} - z_i)}}} dz', \end{aligned} \tag{20}$$

Then using the result $\int \frac{1}{\sqrt{a-bx}} dx = \frac{-2\sqrt{a-bx}}{b} + C$ to calculate the integral analytically

$$\begin{aligned} \int_0^{z_1} \frac{dz}{n(z)} &= \sum_{i=0}^{N-2} \frac{2(z_{i+1} - z_i)}{\sqrt{n^2(z_i)} + \sqrt{n^2(z_{i+1})}} \\ &= \sum_{i=0}^{N-2} \frac{2\Delta z}{n(z_i) + n(z_{i+1})} \end{aligned} \tag{21}$$

Secondly, we deal with the former integral in eq. 18. As the upper limit for the integral is z_r, we need to remember that $n(z = z_r) = 0$, then because of $z_r = n^2(z_i)\frac{z_{i+1} - z_i}{n^2(z_i) - n^2(z_{i+1})} + z_i$ the integral will be

$$n^2(z_i) \int_0^{\frac{z_{i+1} - z_i}{n^2(z_i) - n^2(z_{i+1})}} \frac{dz}{n(z')} \tag{22}$$

already applied the change of variables. The result is

$$\int_{z_i}^{z_r} \frac{dz}{n(z)} = 2n(z_i)\frac{(z_{i+1} - z_i)}{n^2(z_i) - n^2(z_{i+1})} \tag{23}$$

with $z_i = z_{N-1}$ and $z_{i+1} = z_N$.

The Python implementation of the forward model can be found below.

```python
def get_ionogram(h,fp,range_f):

    ne = fp2_to_ne(fp**2) # from plasma frequency squared to electron density

    # Some modifications for numerical stability
    ne[0]=0.00
    epsilon = 1e-7
    range_f = range_f - epsilon
    # end of modifications

    if h[0]!=0:
        h  = np.append(0,h)
        ne = np.append(0,ne)

    n_elements = len(h)
    dif_h = h[1:len(h)] - h[0:len(h)-1]

    f  = np.array([])
    hv = np.array([])

    fp2 = ne_to_fp2(ne)
    max_possible_fp = max(fp2)

    for f_i in range_f:
        f_probe = f_i**2
        if f_probe<max_possible_fp:
            n2l = 1 - fp2[0]/f_probe

            integral = 0
            if n2l>0:
                sqrt_n2l = np.sqrt(n2l)
                for i in range(n_elements-1):
                    n2r = 1 - fp2[i+1]/f_probe
                    if n2r>0:
                        sqrt_n2r = np.sqrt(n2r)
                        integral += 2*dif_h[i]/(sqrt_n2l+sqrt_n2r)
                        n2l = n2r
                        sqrt_n2l=sqrt_n2r
                    else:
                        integral += 2*dif_h[i]*sqrt_n2r/(n2l-n2r)
                        break
            f  = np.append(f,f_i)       # Hz
            hv = np.append(hv,integral) # m
    return f, hv # (Hz,m)
```

Listing 2. Algorithm for the forward model.

III. RESULTS

ACKNOWLEDGMENTS

R. K. T. P. acknowledges the financial support by Pontificia Catolica Universidad del Peru and Prof. Marco Milla for useful comments and discussions.

[1] L. Niu, L. Wen, C. Zhou, and M. Deng, "A profile inversion method for vertical ionograms," *AIP Advances*, vol. 14, no. 6, p. 065034, Jun. 2024. doi: 10.1063/5.0208687.

[2] J.E. Titheridge, "A new method for the analysis of ionospheric h'(f) records," *Journal of Atmospheric and Terrestrial Physics*, vol. 21, no. 1, pp. 1-12, 1961. doi: 10.1016/0021-9169(61)90185-4.

[3] B. W. Reinisch and X. Huang, "Automatic calculation of electron density profiles from digital ionograms: 3. Processing of bottomside ionograms," *Radio Science*, vol. 18, no. 3, pp. 477-492, 1983. doi: 10.1029/RS018i003p00477.

[4] M. H. Reilly and J. D. Kolesar, "A method for real height analysis of oblique ionograms," *Radio Science*, vol. 24, no. 04, pp. 575-583, 1989, doi: 10.1029/RS024i004p00575.

[5] J. E. Titheridge, "Direct Manual Calculations of Ionospheric Parameters Using a Single-Polynomial Analysis," *Radio Science*, vol. 2, no. 10, pp. 1237-1253, 1967, doi: https://doi.org/10.1002/rds19672101237.

[6] A. Manjrekar and S. Tulasiram, "Iterative Gradient Correction (IGC) Method for True Height Analysis of Ionograms," *Radio Science*, vol. 58, Nov. 2023, doi: 10.1029/2023RS007808.

[7] K. G. Budden, "The Numerical Solution of the Differential Equations Governing the Reflexion of Long Radio Waves from the Ionosphere. II," *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 248, no. 939, pp. 45-72, 1955. [Online]. Available: http://www.jstor.org/stable/91622.

[8] V. K. Forsterling and H. Lassen, "Die Ionisation der Atmosphiire und die Ausbreitung der kurzen elektrischen Wellen (IQ-100 m) iiber die Erde. III," *Zeitschrift fur technische Physik*, vol. 12, pp. 502-527, 1931.

IV. APPENDIX A: DATA SET

The two data sets used in the experiments can be found in the hyperlinks: dataset 1 and dataset 2.