We are IntechOpen, the world’s leading publisher of Open Access books
Built by scientists, for scientists

4,300
Open access books available

117,000
International authors and editors

130M
Downloads

154
Countries delivered to

TOP 1%
Our authors are among the most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™
Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com
Chapter 1

Rational Fitting Techniques for the Modeling of Electric Power Components and Systems Using MATLAB Environment

Eduardo Salvador Bañuelos-Cabral, José Alberto Gutiérrez-Robles and Bjørn Gustavsen

Abstract

This book provides a detailed description of some of the most widely used rational fitting techniques for approximation of frequency domain responses. The techniques are: Bode’s asymptotic approximation, the Levy method, iteratively reweighted least squares, the Sanathanan-Koerner method, the Noda method, Vector Fitting, the Levenberg-Marquardt method, and the Damped Gauss-Newton method. A MATLAB routine for each technique is presented. These techniques are tested by approximating synthetic frequency domain responses. Then, they are applied to the rational approximation of the frequency-dependent parameters corresponding to a single-phase transmission line. The effect of the rational function-based models is evaluated, considering transients in three cases: Open-ended, short-circuited, and perfectly matched lines. The error levels obtained in time domain simulations are consistent with the fitting deviations of the frequency-dependent parameters. The book concludes by showing main advantages and disadvantages for each technique.

Keywords: rational approximation, least squares, weighted least squares, Bode’s asymptotic approximation, vector fitting

1. Introduction

One of the main problems encountered in the modeling of power system components is the inclusion of frequency-dependent effects in a time domain simulation [1]. In practice, these effects are described by discrete frequency domain responses that are obtained via calculations or measurements [1]. For the time domain simulation, implementation of numerical convolutions
is possible but is computationally inefficient. The frequency-dependent effects are usually performed in the frequency domain via complex-curve fitting processes, leading to rational function-based models which can be expressed in pole-zero form, pole-residue form, and polynomial form [2]. Once obtained, these models must be converted into a form suitable for representation in an electromagnetic Transient Program (EMTP-type) circuit simulator [3]. Commonly, such representations can be in the form of network synthesis or in ordinary differential equations (ODEs). Some methods have been adapted to particular problems due to the difficulty present in the development of a general methodology. These rational fitting techniques have been developed since the 1950s for model synthesis based on frequency response data. Some antecedents are presented below.


In addition, a technique based on pole-zero form that quickly became popular for the modeling of overhead lines was presented by Marti [7] in 1982. This method is a numerical implementation of the well-known Bode diagrams technique. Recently, in 2017, Marti and Tavighi [8] presented an investigation based in the same rational approximation technique for the modeling of transmission lines. In this work, this fitting technique will be referred to as Asymptotic Approximation or Bode.

In 1993, Soysal and Semlyen [9] used the Gauss-Seidel optimization method to improve the results given by Levy and, 6 years later Gustavsen and Semlyen [1] developed the Vector Fitting method which has become one of the most widely used techniques.

In 2006, Gustavsen [10] proposed a modification of his algorithm in order to improve the ability of VF to relocate poles to better positions. This is achieved by replacing the high-frequency asymptotic requirement of the VF scaling function with a more relaxed condition. He called this method Relaxed VF (RVF).

The VF method is closely related to the Universal Line Model (ULM) [11]. The ULM is formulated in terms of rational approximation of the line parameters through VF. A recent publication, in 2016, [12] Bañuelos et al. propose the use of only real poles and zeroes in the ULM to improve its numerical efficiency.

In 2005, Noda [13] presented an iterative algorithm that partitions the entire frequency range in order to avoid ill-conditioning of the system when Levy method is used. Recently, in 2015, Bañuelos-Cabral et al. [2] propose the implementation of Damped Gauss-Newton (DGN) to increase the accuracy of this technique.

The aim of this book is to provide a MATLAB algorithm and a detailed description of the rational approximation techniques that are most commonly used to approximate functions in
frequency domain. These techniques are: Bode’s Asymptotic Approximation (Bode), the Levy (Levy or LS) method, Iteratively Reweighted Least Squares (IRLS), the Sanathanan-Koerner (SK) method, the Noda (Noda) method, Vector Fitting (VF), the Levenberg-Marquardt (LM) method and the Damped Gauss-Newton (DGN) method.

In this chapter, the methodology for determining the state-space (SS) representation in concordance with the model used in the approximation is presented first. Next, the abovementioned fitting techniques are described in detail. Then, the techniques are tested by approximating a synthetic frequency domain response. The techniques are implemented in MATLAB environment. Finally, advantages of the methods are demonstrated in the rational approximation of the frequency-dependent parameters corresponding to a single-phase transmission line.

2. State-space model from a transfer function

The SS representation of a given system can be determined from its frequency response on rational form. Basically, there are three different rational forms (models) that can represent a measured or calculated frequency response: pole-zero form, polynomial form, and pole-residue form. In this section, a methodology is presented for determining the SS representation according to the given model type.

2.1. Pole-zero form

Usage of pole-zero form or series realization leads to a rational function-based model of \( n \)th-order given by (1), which is a ratio between products of first-order transfer functions,

\[
F(s) \cong k \frac{(s - z_1)(s - z_2)\cdots(s - z_n)}{(s - p_1)(s - p_2)\cdots(s - p_m)}. \tag{1}
\]

The generalized graphic representation of pole-zero form with \( n = m \) is shown in Figure 1 [14].

From Figure 1, it is possible to obtain the state equations and the output equation as

![Figure 1. Pole-zero form realization for nth-order system.](http://dx.doi.org/10.5772/intechopen.71358)
\[
\begin{align*}
\dot{x}_1 &= -p_1 x_1 + u \\
\dot{x}_2 &= -p_2 x_2 + z_1 x_1 + \hat{x}_1 \\
\dot{x}_3 &= -p_3 x_3 + z_2 x_2 + \hat{x}_2 \\
&\vdots \\
\dot{x}_{n-1} &= -p_{n-1} x_{n-1} + z_{n-2} x_{n-2} + \hat{x}_{n-2} \\
\dot{x}_n &= -p_n x_n + z_{n-1} x_{n-1} + \hat{x}_{n-1} \\
y &= k(z_n x_n + \hat{x}_n). 
\end{align*}
\] 

Using algebraic manipulation in (2) and (3) to remove \(\hat{x}_n\) from the right side, the state equations and output equation in matrix form are

\[
\begin{bmatrix}
\dddot{x}_1 \\
\ddot{x}_2 \\
\vdots \\
\dddot{x}_{n-1} \\
\dddot{x}_n
\end{bmatrix} = 
\begin{bmatrix}
-p_1 & 0 & \cdots & 0 & 0 \\
0 & -p_2 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & \cdots & -p_{n-1} & 0 \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_{n-1} \\
\end{bmatrix} + 
\begin{bmatrix}
1 \\
1 \\
\vdots \\
1 \\
\end{bmatrix} u
\]

(4)

\[
y = \begin{bmatrix}
\hat{c}_1 \\
\hat{c}_2 \\
\vdots \\
\hat{c}_{n-1} \\
\hat{c}_n
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n \\
\end{bmatrix} + ku
\]

(5)

where \(\hat{c}_n = k(z_n - p_n)\). There is a direct relation between the SS coefficients in (4) and (5) with the transfer function coefficients in (1) for pole-zero form realization. Thus, the SS description can be obtained directly from the transfer function by inspection.

2.2. Polynomial form

In this case, polynomial form or direct form realization leads to a rational function-based model of \(n\)th-order given by (6), which is a ratio of two polynomials,

\[
F(s) = \frac{a_0 + a_1 s + a_2 s^2 + \cdots + a_n s^n}{b_0 + b_1 s + b_2 s^2 + \cdots + b_m s^m}.
\]

(6)

A graphical representation of the polynomial form with \(b_m = 1\) and \(n = m\) is shown in Figure 2 [14]. From Figure 2, it is possible to obtain the state equations and output equation as
To remove $\dot{x}_n$ from (8) the last equation of (7) can be used. These equations can be written more conveniently as

$$\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\vdots \\
\dot{x}_{n-1} \\
\dot{x}_n
\end{bmatrix} =
\begin{bmatrix}
0 & 1 & 0 & \cdots & 0 & 0 \\
0 & 0 & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 0 & 1 \\
-b_0 & -b_1 & -b_2 & \cdots & -b_{n-2} & -b_{n-1}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_{n-1} \\
x_n
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
1
\end{bmatrix}
+ 
\begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
u
\end{bmatrix}
$$

(9)
where $\hat{\alpha}_i = a_i - a_n b_i$. Also, there is a direct relation between the SS coefficients in (9) and (10) with the transfer function coefficients in (6). The matrices of the SS description can be obtained directly from the transfer function through inspection.

2.3. Pole-residue form

Finally, pole-residue form or parallel realization leads to rational function-based model of $n$th-order given by (11), which is a sum of partial fractions

$$F(s) = \frac{c_1}{s-p_1} + \frac{c_2}{s-p_2} + \cdots + \frac{c_n}{s-p_n} + d.$$  

(11)

The generalized graphic representation of the pole-residue form is shown in Figure 3 [14].

From this Figure 3, it is possible to obtain the state equations and output equation as

Figure 3. Pole-residue form realization for $n$th-order system.
\begin{align*}
\dot{x}_1 &= -p_1 x_1 + u \\
\dot{x}_2 &= -p_2 x_2 + u \\
&\vdots \\
\dot{x}_n &= -p_n x_n + u \\
y &= c_1 x_1 + c_2 x_2 + \cdots + c_n x_n + du. 
\end{align*}

In matrix form it yields
\begin{equation}
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\vdots \\
\dot{x}_n
\end{bmatrix} = 
\begin{bmatrix}
-p_1 & 0 & \cdots & 0 \\
0 & -p_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & -p_n
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix} + 
\begin{bmatrix}
1 \\
1 \\
\vdots \\
1
\end{bmatrix} u
\end{equation}

\begin{equation}
y = 
\begin{bmatrix}
c_1 \\
c_2 \\
\vdots \\
c_n
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix} + du.
\end{equation}

Similarly to the pole-zero form and the polynomial form, the SS representation can be obtained through inspection. It should be mentioned that the pole-residue form produces an uncoupled SS system.

3. Fitting methods

This section provides a description of the most widely used techniques for rational approximation of frequency domain responses within the power systems area.

3.1. Asymptotic approximation (Bode) technique

The Asymptotic Approximation technique consists of a numerical implementation of the graphical technique known as Bode diagrams [15]. The Asymptotic Approximation was first implemented by Marti [7] to include the frequency dependence in transmission line modeling. This technique considers the pole-zero form with real poles and zeros only,
\begin{equation}
F(s) = k \frac{(s - z_1)(s - z_2)\cdots(s - z_n)}{(s - p_1)(s - p_2)\cdots(s - p_m)}.
\end{equation}

Considering \( s = j\omega \), we get
\begin{equation}
F(s) = k \frac{(j\omega - z_1)(j\omega - z_2)\cdots(j\omega - z_n)}{(j\omega - p_1)(j\omega - p_2)\cdots(j\omega - p_m)}.
\end{equation}

Eq. (17) can be expressed in a standard form as
and in polar form

\[ F(j\omega) = K_0 \frac{|\omega z_1 - 1|}{|\omega p_1 - 1|} \frac{|\omega z_2 - 1|}{|\omega p_2 - 1|} \ldots \frac{|\omega z_n - 1|}{|\omega p_m - 1|} \]

(19)

where \( K_0 = k z_1 z_2 \ldots z_n / p_1 p_2 \ldots p_m \). The Asymptotic Approximation technique approximates only the magnitude of the frequency response, which can be expressed as

\[ |F(j\omega)| = K_0 \frac{|\omega z_1 - 1|}{|\omega p_1 - 1|} \frac{|\omega z_2 - 1|}{|\omega p_2 - 1|} \ldots \frac{|\omega z_n - 1|}{|\omega p_m - 1|} \]

(20)

Finally, by using logarithmic properties in Eq. (20), it is obtained:

\[ \log_{10}|F(j\omega)| = \log_{10}|K_0| + \log_{10}|j\omega / z_1 - 1| + \ldots + \log_{10}|j\omega / z_n - 1| - \log_{10}|j\omega / p_1 - 1| - \ldots - \log_{10}|j\omega / p_m - 1|. \]

(21)

In Bode diagrams, each term of (21) is plotted individually in accordance with its already known asymptotic behavior and combined in order to obtain the desired diagram.

In the numerical implementation, the function to be fitted is compared with the sum of the line segments given by the addition of a pole or a zero in the model (21); the precision of the fitting depends on the sensitivity to locate these poles and zeros into the model.

3.2. Levy (Levy) method

This method was introduced by Levy [4] for complex-curve fitting. The Levy method makes the identification of the polynomial coefficients (22) in a LS sense. It is also known simply as Least Squares (LS).

\[ F(s) \approx \frac{N(s)}{D(s)} = \frac{a_0 + a_1 s + a_2 s^2 + \ldots + a_n s^n}{1 + b_1 s + b_2 s^2 + \ldots + b_m s^m}. \]

(22)

The numerical difference between the frequency response to be fitted and the model represents the error in the approximation, that is,

\[ \epsilon(s) = F(s) - \frac{N(s)}{D(s)}. \]

(23)

Multiplying both sides of Eq. (23) by \( D(s) \), we obtain
$$\epsilon'(s) = \epsilon(s)D(s) = F(s)D(s) - N(s).$$  \hspace{1cm} (24)

Considering that $\epsilon'$ tends to zero, (24) can be expressed as

$$F(s)\left(1 + b_1s + b_2s^2 + \cdots + b_ms^m\right) - \left(a_0 + a_1s + a_2s^2 + \cdots + a_ns^n\right) = 0.$$  \hspace{1cm} (25)

The unknown coefficients in (22) can now be solved by formulating (25) as a LS problem ($Ax = b$). Frequently, the sample size of the frequency response is greater than the number of polynomial coefficients to be calculated, so an overdetermined system is obtained, as follows:

$$A_k = \begin{bmatrix} 1 & s_k^0 & \cdots & s_k^m & -s_kF(s_k) & \cdots & -s_k^mF(s_k) \end{bmatrix}$$  \hspace{1cm} (26)

$$x = \begin{bmatrix} a_0 & a_1 & \cdots & a_n & b_1 & b_2 & \cdots & b_m \end{bmatrix}^T$$  \hspace{1cm} (27)

$$b = \begin{bmatrix} F(s_1) & \cdots & F(s_k) \end{bmatrix}^T,$$  \hspace{1cm} (28)

where $k$ denotes the $k$-th data sample. The objective function to be minimized is

$$\min_x \|Ax - b\|_2.$$  \hspace{1cm} (29)

This is called a linear least squares problem. The solution $x$ satisfies the normal equations [16]:

$$A^T Ax = A^T b.$$  \hspace{1cm} (30)

Finally, the LS solution is obtained by

$$x = \left(A^T A\right)^{-1} A^T b.$$  \hspace{1cm} (31)

### 3.3. Weighted least squares (WLS)

The LS solution of a given system equation assumes that each equation (row in $A$) is equally important. However, the system equation itself may be biased, e. g. in the Levy method which is biased due to the multiplication of $F(s)$ with $D(s)$ in (24). In the rational approximation of frequency domain responses, Weighted Least Squares (WLS) is used to mitigate the biasing by giving more weight to specific equations in order to overcome the own deficiency of the used technique. Iteratively Reweighted Least Squares (IRLS), the Sanathanan-Koerner (SK) method and the Noda (Noda) method are techniques that implement the concept of WLS. These techniques are described below.

#### 3.3.1. Iteratively reweighted least squares (IRLS) iteration

A robust regression procedure is an alternative to LS solution when data are contaminated with outliers or influential observations (measurement and/or computation errors) [16, 17]. The idea of robust regression is to weight (less) these observations differently based on the proposal of weighting functions. IRLS can be considered as a robust regression procedure.
It is proposed to use IRLS for the rational approximation of frequency domain responses by using the weighting functions in inverse form, namely, data are not considered to be contaminated with outliers or influential observations; the error in the fitting in each iteration is used to weight (more) the frequency response data that have not been approximated correctly.

In WLS, the objective function to be minimized is

$$\min_x \| W(b - Ax) \|_2$$  \hspace{1cm} (32)

where $W$ is a diagonal weighting matrix. Eq. (32) is called a linear weighted least squares problem and $x$ the linear weighted least squares solution of the system. This solution satisfies the normal equations [16]:

$$A^T W A x = A^T W b.$$  \hspace{1cm} (33)

Then, the WLS solution is given by

$$x = (A^T W A)^{-1} A^T W b.$$  \hspace{1cm} (34)

It is possible to solve (34) through an iterative process (35), where $i$ is the iteration number.

$$x^{(i+1)} = (A^T W^{(i)} A)^{-1} A^T W^{(i)} b.$$  \hspace{1cm} (35)

Eq. (35) is the IRLS method. According to (35), $W$ must be updated iteratively; the error in the approximation is used for this purpose, which can be expressed as

$$\epsilon(s, x) = F(s) - \bar{F}(s, x),$$ \hspace{1cm} (36)

where $\bar{F}(s, x)$ is the model to be fitted. Each element of $W$ is updated according to

$$w^{(i)} = w^{(i-1)} \psi(\epsilon^{(i-1)}),$$ \hspace{1cm} (37)

with $\psi(\epsilon)$ being the weighting function. In the beginning of the process, $W$ is an identity matrix.

**Table 1** shows a list of weighting functions proposed to use in the implementation of IRLS. Weighting functions 1, 2, and 3 are the inverse of the functions used in robust regression [16, 17], and functions 4, 5, and 6 are proposed in this work.

Ultimately, in **Figure 4** the behavior of the weighting functions with $\epsilon = [-2, 2]$ is shown.

**3.3.2. Sanathanan-Koerner (SK) iteration**

This method is based on the polynomial form:

$$F(s) \cong \frac{N(s)}{D(s)} = \frac{a_0 + a_1 s + a_2 s^2 + \cdots + a_n s^n}{1 + b_1 s + b_2 s^2 + \cdots + b_m s^m}.$$  \hspace{1cm} (38)
The objective is to identify the coefficients for the polynomials $N(s)$ and $D(s)$ in (38) that minimize the error function

$$\varepsilon(s) = F(s)D(s) / N(s).$$  \hspace{1cm} (39)

Sanathanan and Koerner [5] proposed an iterative procedure where (39) is divided by the denominator from the previous iteration; thus, (39) can be expressed as

<table>
<thead>
<tr>
<th>Number</th>
<th>$\psi(\varepsilon)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>2</td>
<td>$\sqrt{1 + \varepsilon^2} - 1$</td>
</tr>
<tr>
<td>3</td>
<td>$\frac{1}{</td>
</tr>
<tr>
<td>4</td>
<td>$\frac{1}{</td>
</tr>
<tr>
<td>5</td>
<td>$\frac{1}{\sqrt{1 + \varepsilon^2}}$</td>
</tr>
<tr>
<td>6</td>
<td>$\frac{</td>
</tr>
</tbody>
</table>

Table 1. Weighting functions.

Figure 4. Behavior of the weighting functions.
The subscript \( l \) denotes the iteration number, and \( D(s) \) is considered 1 in the first iteration.

3.3.3. Noda (Noda) iteration

This algorithm proposed by Noda [13] for the rational fitting identification is based on the function:

\[
F(s) \approx \frac{N(s)}{D(s)} = \frac{a_0 + a_1 s + a_2 s^2 + \cdots + a_n s^n}{1 + b_1 s + b_2 s^2 + \cdots + b_m s^m}.
\]  

(41)

This technique corresponds to IRLS using the weighting function number 1 in Table 1. Additionally, Noda also proposed a procedure to prevent the ill-conditioning of the system by partitioning the given frequency response data into sections along the frequency axis. The technique is then applied to each section of the frequency response in order to identify the poles, and finally, the corresponding residues are obtained by means of a standard LS procedure, using the entire frequency response.

3.4. Vector Fitting (VF) method

Vector Fitting performs the fitting by replacing a set of heuristically calculated initial poles with a set of relocated ones through an iterative procedure [1]. VF works in two stages: First, it improves the position of the initial poles iteratively. Second, it calculates the residues in one step. This method is based on the pole-zero form

\[
F(s) \approx \sum_{n=1}^{N} \frac{c_n}{s - p_n} + d + sh, \quad (42)
\]

where \( c_n \) are the residues, \( p_n \) are the poles, \( d \) is the constant term and \( h \) the proportional part.

3.4.1. Pole identification

Considering the initial poles as \( a_n \), and multiplying \( F(s) \) by an auxiliary function \( \sigma(s) \) gives

\[
\sigma(s)F(s) \approx \sigma F_p(s) = \sum_{n=1}^{N} \frac{c_n}{s - a_n} + d + sh, \quad (43)
\]

where \( \sigma(s) \) is defined as

\[
\sigma(s) \approx \sigma_p(s) = \sum_{n=1}^{N} \frac{\tilde{c}_n}{s - \tilde{a}_n} + 1, \quad (44)
\]
with $\sigma_F(s)$ being the fitting of $\sigma(s)F(s)$ and $\sigma_{fit}(s)$ being the fitting of $\sigma(s)$. From (43) one can obtain,

$$F(s) \cong \frac{\sigma_{fit}(s)}{\sigma(s)}.$$  \hspace{1cm} (45)

Substituting (43) and (44) into (45) we obtain,

$$F(s) \cong h \frac{\prod_{n=1}^{N+1} s - z_n}{\prod_{n=1}^{N} s - a_n} \frac{\prod_{n=1}^{N} s - z_n}{\prod_{n=1}^{N+1} s - a_n} = h \frac{\prod_{n=1}^{N+1} s - z_n}{\prod_{n=1}^{N} s - a_n}.  \hspace{1cm} (46)$$

Eq. (46) indicates that the zeros of $\sigma(s)$ are an approximation of the poles of $F(s)$. To obtain the zeros one multiplies (44) by $F(s)$ which results in

$$\sigma(s)F(s) \cong \left( \sum_{n=1}^{N} \frac{\tilde{c}_n}{s - a_n} + 1 \right) F(s).  \hspace{1cm} (47)$$

Equating Eqs. (43) and (47) yields

$$\sum_{n=1}^{N} \frac{c_n}{s - a_n} + d + sh = \left( \sum_{n=1}^{N} \frac{\tilde{c}_n}{s - a_n} + 1 \right) F(s).  \hspace{1cm} (48)$$

Algebraic manipulation in (48) gives

$$\sum_{n=1}^{N} \frac{c_n}{s - a_n} + d + sh = \sum_{n=1}^{N} \frac{\tilde{c}_n F(s)}{s - a_n} = F(s).  \hspace{1cm} (49)$$

Eq. (49) can now be formulated as a LS problem ($Ax = b$). Usually, the sample size of the frequency response is greater than the number of coefficients to be calculated, so an overdetermined system is obtained:

$$A_{k} = \begin{bmatrix} 1 & \cdots & 1 & 1 & s_k & \cdots & s_k & F(s_k) & \cdots & \frac{F(s_k)}{s_k - a_N} \end{bmatrix}  \hspace{1cm} (50)$$

$$x = \begin{bmatrix} c_1 & \cdots & c_N & d & h & \tilde{c}_1 & \cdots & \tilde{c}_N \end{bmatrix}^T  \hspace{1cm} (51)$$

$$b = \begin{bmatrix} F(s_1) & \cdots & F(s_k) \end{bmatrix}^T  \hspace{1cm} (52)$$

The LS solution of the system ($Ax = b$) delivers the residues of $\sigma_{fit}(s)$; the zeros of this function are the poles $a_n$ for the next iteration, which correspond to the eigenvalues of [1].
\[ \mathbf{H} = \mathbf{G} - \mathbf{k} \mathbf{c}^T, \]  

(53)

where \( \mathbf{G} \) is a diagonal matrix containing the poles and \( \mathbf{k} \) is a unit column vector.

### 3.4.2. Residue Identification

Once the poles \( p_n \) for the rational approximation in (42) have been obtained, the residues can be found by solving (42) as a LS problem. The new overdetermined system is solved similarly as (50), (51) and (52).

### 3.4.3. Relaxation (RVF)

The scaling function \( \sigma(s) \) (44) is observed to approach unity at high frequencies. This asymmetry with respect to the frequency gives a tendency to relocate poles in the direction of lower frequencies. In addition to this biasing, it also reduces the convergence speed and often leads to a reduced accuracy for the final model. This problem is effectively alleviated by the introduction of Relaxed Vector Fitting (RVF) [10], where the scaling function (44) is replaced by

\[
\sigma(s) \cong \sigma_{\text{fit}}(s) = \sum_{n=1}^{N} \frac{c_n}{s - a_n} + \delta
\]

(54)

with \( \delta \) as a real variable. To obtain a non-trivial solution, a single line is added to the LS problem, where the sum of the real part of \( \sigma(s) \) over the frequency samples is fixed. This relaxed criterion allows \( \sigma(s) \) to freely vary in shape.

### 3.5. Levenberg-Marquardt (LM) method

The Levenberg–Marquardt (LM) method is a technique used to improve iteratively parameter values in order to reduce the sum of the squares of the errors between the model and the calculated or measured data [18, 19]. This method is actually a combination of two minimization methods: The gradient descent (GD) method and the Gauss-Newton (GN) method.

Consider the model \( \mathbf{F}(s, \mathbf{x}) \), which is a function of \( s \) and \( n \) unknown parameters contained in vector \( \mathbf{x} \). The model is to be fitted to a set of frequency response data \( \mathbf{F}(s) \) as follows:

\[
\mathbf{F}(s) \cong \mathbf{F}(s, \mathbf{x}),
\]

(55)

where \( \mathbf{x} = [x_1, x_2, \ldots, x_n] \) is a vector that contains the coefficients to be fitted. The LS error between the fitting of the model and the frequency response data is

\[
\gamma(\mathbf{x}) = \frac{1}{2} \sum_{k=1}^{N} \left( \mathbf{F}(s_k, \mathbf{x}) - \mathbf{F}(s_k) \right)^2
\]

(56)

\[
\gamma(\mathbf{x}) = \frac{1}{2} \left( \mathbf{F}(s, \mathbf{x}) - \mathbf{F}(s) \right)^T \left( \mathbf{F}(s, \mathbf{x}) - \mathbf{F}(s) \right)
\]

(57)
The aim of LM method is to find iteratively a perturbation $h$ to modify the parameters $x$ that reduces the objective function $\gamma(x)$ [18].

3.5.1. Gradient descent (GD) method

The gradient of a function indicates the direction of the maximum rate of change in a specific point of the function. GD updates the parameter values in the opposite direction to the gradient of the objective function. Thus, the gradient of $\gamma(x)$ is:

$$\frac{\partial \gamma(x)}{\partial x} = \left( \tilde{F}(s, x) - F(s) \right)^T \frac{\partial F(s)}{\partial x}$$

(59)

$$\frac{\partial \gamma(x)}{\partial x} = \left( \tilde{F}(s, x) - F(s) \right)^T J,$$

(60)

where $J$ is the Jacobian matrix which represents the local sensitivity of the model $\tilde{F}(s, x)$ to variation in the parameters $x$. Finally, the perturbation $h$ that moves the parameters in the direction of steepest descent is given by

$$h_{GD} = -J^T \left( \tilde{F}(s, x) - F(s) \right).$$

(61)

3.5.2. Gauss-Newton (GN) method

The Gauss-Newton (GN) method supposes that the objective function $\gamma(x)$ is approximately quadratic in the parameters near the optimal solution. Approximating the model with a first-order Taylor series,

$$\tilde{F}(s, x + h) \approx \tilde{F}(s, x) + \frac{\partial \tilde{F}(s, x)}{\partial x} h = \tilde{F}(s, x) + Jh.$$

(62)

Substituting (62) into (57) gives

$$\gamma(x + h) \approx \frac{1}{2} F^T F - \tilde{F}^T F + \frac{1}{2} F^T F + \left( \tilde{F} - F \right)^T Jh + \frac{1}{2} h^T J^T Jh.$$  

(63)

The perturbation $h$ that minimizes $\gamma(x)$ is reached when

$$\frac{\partial \gamma(x + h)}{\partial h} = 0.$$  

(64)

Taking the derivative for (63) gives

$$\frac{\partial \gamma(x + h)}{\partial h} = \left( \tilde{F}(s, x) - F(s) \right)^T J + h^T J^T J$$

(65)

and applying condition (64), we obtain the perturbation $h$ calculated by the GN method.
\[ h_{GN} = -\left(J^T J\right)^{-1}J^T \left(\tilde{F}(s, x) - F(s)\right). \] (66)

3.5.3. Levenberg-Marquardt method (LM)

LM works more like the GD method when the parameters are far from their optimal value, and more like the GN method when the parameters are close to their optimal value. Levenberg [18] and Marquardt [19] proposed this method:

\[ (J^T J + \lambda I) h_{LM} = -J^T \left(\tilde{F}(s, x) - F(s)\right), \] (67)

where \( \lambda \) is known as the Levenberg–Marquardt parameter. Clearly, for large values of \( \lambda \), the LM method results in a GD update; large distance from the function. The GD method is utilized to provide steady and convergent progress toward the solution. As the solution approaches the minimum, \( \lambda \) is adaptively decreased; then, the LM method approaches the GN method, and the solution typically converges rapidly to the local minimum.

The following modification has also been suggested:

\[ (J^T J + \lambda \text{diag}(J^T J)) h_{LM} = -J^T \left(\tilde{F}(s, x) - F(s)\right) \] (68)

There are different methods for update the Levenberg-Marquardt parameter \( \lambda \). A complete review is presented in Ref. [17]. Nevertheless, a simple method consists of reducing this parameter as the solution approaches the optimal value. The numerical implementation consists iteratively of:

1. Calculate the objective function \( \gamma(x) \), (57).
2. Calculate the Jacobian \( J \), taking into account the implemented model (1), (6) or (11).
3. Calculate the perturbation \( h \), (67).
4. If \( \gamma(x + h) \leq \gamma(x) \), then \( x \) is replaced by \( x + h \) and \( \lambda \) is reduced.
5. If \( \gamma(x + h) > \gamma(x) \), then \( x \) is not replaced and \( \lambda \) is increased.
6. Convergence is achieved when the parameters reach a tolerance value or when the iteration count exceeds a pre-specified limit.

3.5.4. Damped gauss-Newton (DGN)

The DGN method can be applied to any of the alternative function-based models: pole-zero form (1), polynomial form (6), and pole-residue form (11). In the DGN method, a damping factor \( \alpha \) is introduced

\[ h_{DGN} = -\alpha\left(J^T J\right)^{-1}J^T \left(\tilde{F}(s, x) - F(s)\right). \] (69)

The DGN method (69) always takes the descent direction that satisfies a linear search method. It has slow convergence when the parameters to be fitted are far from the solution,
but the method maximizes its advantages when it is close to the solution [16]. For that reason, DGN should be used when the parameters are close to their optimal values.

Clearly, $\alpha$ defines a fraction of the step taken from GN to update the $x$ parameters. In this work, we select $\alpha$ by a backtracking strategy, whereby $\alpha$ is reduced from unity until an acceptable value for $\gamma(x)$ is found.

4. MATLAB algorithms and applications

In this section, algorithms for the rational approximation of frequency domain responses in MATLAB environment are provided. Rational fitting of synthetics functions through Bode, DGN, LS or Levy, IRLS, LM and VF are presented. Finally, the techniques are applied to the rational approximation of the frequency-dependent parameters corresponding to a single-phase transmission line.

4.1. Bode and DGN algorithms

Bode and DGN routines are presented in this section. The concept is simple, first Bode is implemented for the curve fitting process of a synthetic function, then DGN is used to improve the accuracy of the rational approximation given by Bode. A synthetic function is selected in the main program “Fitting_Bode_DGN.m” presented in Table 2. After application of the Bode routine “Bode_process.m,” shown in Table 3, a set of poles, residues and a constant term are obtained. These results are the initial values for the DGN routine “Damped_Gauss_Newton.m” presented in Table 4.

In Figure 5 the asymptotic behavior of the Bode routine is shown, and in Figure 6 the final rational approximation given by Bode is presented. Following, DGN routine improves the curve fitting process, this is presented in Figure 7. Moreover, the RMS-error on each iteration is shown in Figure 8.

4.2. LS or Levy algorithm

In this section the LS method or Levy method routine is introduced. In the main program “Fitting_OLS.m,” Table 5, a synthetic function can be selected by the user. Then the LS routine “Least_Squares_Method.m,” Table 6, is implemented for the rational approximation of this function.

Figure 9 shows the synthetic frequency response data and the approximation given by the LS routine, additionally, the deviation in terms of absolute error is presented.

4.3. IRLS algorithm

Because in practice the Noda method and the SK method correspond to a specific weighting function in the IRLS technique, only the latter is presented. In the beginning of the implementation a synthetic function is selected in the main program “Fitting_IRLS.m,” presented in Table 7. Afterwards, the algorithm of the method “IRLS.m,” Table 8, is implemented for the rational approximation of the selected function.
Main program of BODE-DGN

Authors: Eduardo Salvador Bañuelos Cabral
José Alberto Gutiérrez Robles
Bjørn Gustavsen

```matlab
clc
clear all
close all

%% Initial Settings
Ns = 400; % Number of samples
f = logspace(-2,10,Ns); % Frequency (Hz)
w = 2.*pi.*f; % Frequency (rad/sec)
s = 1j*w; % Complex Frequency

%% Synthetic functions
choice = menu('CHOOSE A FUNCTION','Function F1(s)',
               'Function F2(s)','Function F3(s)','Function F4(s)',
               'Function F5(s)','Function F6(s)');
if choice == 1
    Fs = (110.*s./((s+10).*(s+100)));
tol = 2.2; % Tolerance in decibels to set a new pole and/or new zero
ite = 40; % Number of iteration in DGN method
end
if choice == 2
    Fs = ((s+10).*(s+100))./((s+14580).*(s+10355550));
tol = 2.2; % Tolerance in decibels to set a new pole and/or new zero
ite = 40; % Number of iteration in DGN method
end
if choice == 3
    Fs = ((s+10655).*(s+10))./((s+148450).*(s+198545852).*(s+156222220));
tol = 2.2; % Tolerance in decibels to set a new pole and/or new zero
ite = 40; % Number of iteration in DGN method
end
if choice == 4
    Fs = (10124).*s.*(s+35.7).*(s+88.9)./((s+100.5).*(s+220.7).*(s+5900).*(s+1370).*(s+21000));
tol = 2; % Tolerance in decibels to set a new pole and/or new zero
ite = 50; % Number of iteration in DGN method
end
if choice == 5
    Fs = ((s+79).*(s+1045))./((s+1458).*(s+103555).*(s+127710355).*(s+1244103555));
tol = 2; % Tolerance in decibels to set a new pole and/or new zero
ite = 40; % Number of iteration in DGN method
end
if choice == 6
    Fs = ((s+64518).*(s+8451629).*(s+312).*(s+54841216192))./((s+456).*((s+982365).*(s+93256888).*(s+79325684588536)));
tol = 2; % Tolerance in decibels to set a new pole and/or new zero
ite = 40; % Number of iteration in DGN method
end

%% Bode method
[P,Z,k] = Bode_process(Fs,f,Ns,tol); % Bode subroutine
as = k.*poly(1); bs = poly(P); % Polynomials
[k,p,ks] = residue(as,bs); if(TF==1);ks=0;end
```

Rational Fitting Techniques for the Modeling of Electric Power Components and Systems Using MATLAB Environment
%% Damped Gauss-Newton
Xmcp = [r; p; ks]; % Poles, residues and constant term from Bode
Np = length(p); % Number of poles
[KGN, RGN, PGN, Ff] = Damped_Gauss_Newton(Np,Ns,Fs,s,Xmcp,ite);
Fs_fitDGN = zeros(1,Ns);
for k = 1:length(PGN)
    Fs_fitDGN = Fs_fitDGN + (RGN(k)./(s.' - PGN(k))).';
end
Fs_fitDGN = Fs_fitDGN + KGN;
error = abs(Fs - Fs_fitDGN);

%% Plots
figure();
loglog(f,abs(Fs),'k',f,abs(Fs_fitDGN),'r--',f,error,'--b','LineWidth',2);
xlabel('Frequency [Hz]'); ylabel('Magnitude [p.u.]');
legend('Data','DGN','Deviation',2)
figure (4)
semilogy(Ff,'--b','LineWidth',2)
xlabel('Iteration count'); ylabel('RMS-error');

Table 2. "Fitting_Bode_DGN.m".
for r = 1:1:Ns
    error = abs((Fsdb(r))-(Fsfitdb(r))); % Deviation
    if error >= (tol) % Tolerance
        % New zero
        if (Fsfitdb(r)< Fsdb(r))
            Z(h)=f(r); h=h+1;
        end
        % New pole
        if (Fsfitdb(r)> Fsdb(r))
            P(c)=f(r); c=c+1;
        end
    end
end
% Contribution of each zero in decibels
Arg1 = 0;
for kp = 1:length(Z)
    Arg1 = Arg1 + 20.*log10(abs(1+i*f/Z(kp)));
end
% Contribution of each pole in decibels
Arg2 = 0;
for kp = 1:length(P)
    Arg2 = Arg2 - 20.*log10(abs(1+i*f/P(kp)));
end
Fsdb2 = Arg1 + Arg2;

% Poles and zeros
P = -P*2*pi;
Z = -Z*2*pi;

% Construct the constant term
Num=1;for k=1:length(P);Num=Num*P(k);end
Den=1;for k=1:length(Z);Den=Den*Z(k);end
k = abs(Num)*k0/abs(Den);

% Plots
figure(2)
semilogx(f,Fsdb,'k',f,Fsfitdb,'--r','LineWidth',2);
xlabel('Frequency [Hz]'); ylabel('Decibels')
legend('Data','Bode',2)

Table 3. "Bode_process.m".

%================================================
% DAMPED GAUSS NEWTON METHOD
% Authors: Eduardo Salvador Bañuelos Cabral
% José Alberto Gutiérrez Robles
% Bjørn Gustavsen
%================================================
% Inputs
% --- Np, number of poles
% --- Ns, Number of samples
% --- Fs, function to be fitted
% --- s, complex frequency [rad/s]
% --- X, initial poles, zeros and constant term
% --- ite,iterations
% Outputs
% --- Ks, constant term
% --- Rs, residues
% --- Ps, poles
% --- FF, objective function (deviation)
function [Ks,Rs,Ps,Ff] = Damped_Gauss_Newton(Np,Ns,Fs,s,X,ite)
a = ones(Ns,1); % Vector column
J1 = zeros(Ns,Np); % Matrix size (J1)
J2 = zeros(Ns,Np); % Matrix size (J2)
Jn = zeros(2*Ns,2*Np+1); % Matrix size (Jn)
At = zeros(2*Np+1,2*Np+1); % Matrix size (At)
en = zeros(2*Ns,1); % Vector size (en)
epn = zeros(2*Ns,1); % Vector size (epn)
Ff = zeros(ite,1); % Vector size (Ff)
Euclidian = zeros(1,1+2*Np); % Vector size
for ki = 1:ite % Damped Gauss Newton methodology
    R = X(1:Np); % Residues
    P = X(Np+1:2*Np); % Poles
    K = X(2*Np+1); % Constant term
    Fa = zeros(1,Ns); % Set the approximation to the function
    for k = 1:length(P)
        Fa = Fa + (R(k)./(s.' - P(k))).';
    end
    Fa = Fa + K; % Construct the approximation to the function
    error = (Fa.' - Fs.'); % Deviation
    for n = 1:1:Np % Loop to construct the Jacobian
        J1(1:Ns,n) = 1./(s-P(n));
        J2(1:Ns,n) = R(n)./((s-P(n)).^2);
    end
    J = [J1 J2 a]; % Jacobian
    [Xm Ym] = size(J); % Matrix size (J)
    Jr = real(J); % Real part of vector J
    Ji = imag(J); % Imaginary part of vector J
    er = real(error); % Real part of vector error
    ei = imag(error); % Imaginary part of vector error
    km = 1; % Counter
    for k = 2:2:2*Xm % Interleaved
        Jn(k-1,:) = Jr(km,:);
        Jn(k,:) = Ji(km,:);
        en(k-1,1) = er(km);
        en(k,1) = ei(km);
        km = km+1;
    end
    F = ((norm(en,2)^2)); % Objective function
    Ff(ki,1) = F; % Storage the objective function must tend to zero
    [Q,R] = qr(Jn); % Matrix Q and R of Jn
    Jn = R; % It updates matrix Jn
    Gf = (Jn.'*Q.').*en; % Gradient (QR decomposition)
    Hess = Jn.'*Jn; % Hessian (QR decomposition)
    for col = 1:Ym % Euclidian norm
        Euclidian(col) = norm(Hess(:,col),2);
    end
    h = (At)
    h = h./Euclidian.'; % Real solution
    stop = 1; % Variable to stop the next loop
    al = 1; % Variable to weigh the approximation
end

Rational Fitting Techniques for the Modeling of Electric Power Components and Systems Using MATLAB Environment
http://dx.doi.org/10.5772/intechopen.71358
In Figure 10 the synthetic frequency response behavior and the rational approximation given by IRLS is presented, together with its deviation in terms of absolute error. Moreover, the RMS-error on each iteration is shown in Figure 11.

4.4. LM algorithm

In this section the LM routine is introduced. First, a synthetic function can be selected in the main program “Fitting_LM_Polynomials.m,” presented in Table 9. Afterwards, the algorithm of the method “LM_method.m,” Table 10, is implemented for the rational approximation of the selected function.
Figure 5. Asymptotic behavior of the Bode routine.

Figure 6. Final rational approximation of the synthetic function given by Bode.
Figure 7. Fitting deviation of the approximation given by DGN.

Figure 8. RMS-error on each iteration of the approximation given by DGN.
In Figure 12 the synthetic frequency response behavior and the rational approximation given by IRLS is presented, together with its deviation in terms of absolute error. Moreover, the RMS-error on each iteration is shown in Figure 13.
% Function to implement ORDINARY LEAST SQUARES
% Authors: Eduardo Salvador Bañuelos Cabral
% José Alberto Gutiérrez Robles
% Bjørn Gustavsen

% Inputs
% --- Np, number of poles
% --- Ns, number of samples
% --- Fs, function to be fitted
% --- s, complex frequency (rad/s)
% Outputs
% --- Ks, constant term
% --- Rs, residues
% --- Ps, poles
% --- x, vector of coefficients (polynomials)
% --- A1, matrix to evaluate the fitting
% --- an, coefficients of the numerator
% --- bn, coefficients of the denominator

function [Ks,Rs,Ps,x,A1,an,bn] = Least_Squares_Method(Np,Ns,Fs,s)
    a = ones(Ns,1); % Column vector
    a1 = zeros(Ns,Np); % Matrix size (s^n)
    a2 = zeros(Ns,Np); % Matrix size (Fs*s^n)
    An = zeros(2*Ns,1+2*Np); % Matrix size (real and imaginary part)
    Bn = zeros(2*Ns,1); % Vector size (real and imaginary part)
    Euclidian = zeros(1,1+2*Np); % Vector size

    % Construction of vector b.
    b = Fs.';
    % Construction of matrix A.
    for n = 1:1:Np
        a1(1:Ns,n) = s.^(Np+1-n);
        a2(1:Ns,n) = -b.*a1(1:Ns,n);
    end
    A1 = [a1 a]; % Matrix to evaluate the fitting
    A = [a1 a a2]; % Matrix of the system

    [Xmax Ymax] = size(A); % Size for matrix A
    Ar = real(A); % Real part of matrix A
    Ai = imag(A); % Imaginary part of matrix A
    br = real(b); % Real part of vector B
    bi = imag(b); % Imaginary part of vector B

    km = 1; % Construction of matrix An and vector Bn (interleaved)
    for k = 2:2:2*Xmax
        An[k-1,:] = Ar(km,:);
        An[k,:] = Ai(km,:);
        Bn[k-1,1] = br(km);
        Bn[k,1] = bi(km);
        km = km+1;
    end

    [Q,R] = qr(An,0); % QR decomposition - Matrix Q and R of An
    At = R; % It updates the matrix An
    Bn = Q.'*Bn; % It updates the vector Bn
4.5. VF algorithm

VF has become one of the main methodologies for the rational approximation of frequency domain responses, the MATLAB routine is available online [20], “vectfit3.m.” Here, we present an example of its application to the rational approximation of a synthetic function.

In the beginning of the implementation a synthetic function is selected in the main program “Fitting_VF.m,” presented in Table 11. Afterwards, initial poles for the fitting must be established
Main program of ITERATIVELY REWEIGHTED LEAST SQUARES
Authors: Eduardo Salvador Bañuelos Cabral
José Alberto Gutiérrez Robles
Bjørn Gustavsen

Main program of ITERATIVELY REWEIGHTED LEAST SQUARES
Authors: Eduardo Salvador Bañuelos Cabral
José Alberto Gutiérrez Robles
Bjørn Gustavsen

clc
clear all
close all

%% Initial Settings
Ns = 500; % Number of samples
f = logspace(-2,6,Ns); % Frequency (Hz)
w = 2.*pi.*f; % Frequency (rad/seg)
s = 1i*w; % Complex Frequency

%% Synthetic functions
choice = menu('CHOOSE A FUNCTION','Function F1(s)','Function F2(s)','Function F3(s)','Function F4(s)','Function F5(s)','Function F6(s)');
if choice == 1
    Np = 12;
    Fs = (0.83*s.^12-0.35*s.^11+0.32*s.^10+60000*s.^9+600.36*s.^8+650.23*s.^7+54.5*s.^6+45.6*s.^5 +0.1*s.^4-1240*s.^3+80025*s.^2+5547.*s+5008)./...
        (s.^12-0.89*s.^11-0.23*s.^10-4565.*s.^9+34569.02*s.^8-55450*s.^7+60*s.^6+1005*s.^5+8050*s.^4+7502*s.^3+10*s.^2+21014.*s-50);
end
if choice == 2
    Np = 9;
    Fs = (6000*s.^9+600*s.^8+650*s.^7+54*s.^6+457*s.^5+100*s.^4+1240*s.^3+8000*s.^2+55400.*s+5000)./(450*s.^9+3500*s.^8+500*s.^7-600*s.^6+1000*s.^5+700*s.^4-4... 7500*s.^3+1000*s.^2+500.*s);
end
if choice == 3
    Np = 7;
    Fs = (514*s.^7-364.25*s.^6-0.35*s.^5+635*s.^4-20802*s.^3+5304*s.^2+24520.*s-... 18020.25)./(241*s.^7-7.32-0.52-0.12; -0.04; Rs = [-2.18; -7192.25; 20.58];
    Rs = [5];
    Fs = zeros(1,Ns);
    for k = 1:length(Ps)
        Fs = Fs + (Rs(k)./(s.' - Ps(k))).';
    end
    Fs = Fs + Ks;
end
if choice == 5
    Np = 11;
    Ps = [0.83; -5.42; - 27.28; 0.88; -9665; 235.7; -0.26; -3.87; 46.32; -0.13; -3756];
    Rs = 1.0e+005.*[834; 22593; 893; 2653; 654; 32; 44.14; 6405; 136.79; 0.12; 125];
InitialPoles.m, Table 12, for the VF routine. Finally in “VF.m,” VF is used (via vectfit3.m) to perform the rational approximation of the selected function. In VF.m routine, Table 13, the parameter opts.relax is set to 1. This implies that “relaxation” is being used.

Figure 14 shows the synthetic frequency response behavior and the rational approximation given by VF, together with its deviation in terms of absolute error.
function [Ks, Rs, Ps, x, A1, an, bn, k1, Error] = IRLS(Np, Ns, Fs, s, fw)
a = ones(Ns, 1); % Vector column
a1 = zeros(Ns, Np); % Matrix size (s^n)
a2 = zeros(Ns, Np); % Matrix size (Fs*s^n)
An = zeros(2*Ns, 1+2*Np); % Matrix size considering real and imaginary part
Bn = zeros(2*Ns, 1); % Vector size considering real and imaginary part
Euclidian = zeros(1,1+2*Np); % Vector size
W = eye(length(Ns*2)); % Weighting matrix

b = Fs.'; % Construction of vector b.
for n = 1:1:Np % Construction of matrix A.
a1(1:Ns,n) = s.^(Np+1-n);
a2(1:Ns,n) = -b.*a1(1:Ns,n);
end
A1 = [a1 a]; % Matrix to evaluate the fitting
A = [a1 a a2]; % Matrix of the system

[Q, R] = qr(A1, 0); % Matrix Q and R of An
A1 = R; % It updates matrix An
B = Q.'*Bn; % It updates vector Bn
for col = 1:Ymax  % Applying the Euclidian norm to At
    Euclidian(col) = norm(At(:,col),2);
    At(:,col) = At(:,col)/.Euclidian(col);
end

xn(:,1) = At\B;  % Solution for the system (Ax = B)
xn(:,1) = xn(:,1)/.Euclidian.';  % Real solution

% weighting iterative
Error(1) = 1e2;  % Initial error
Disc = 1e2;  % Discriminating
n=1;  % Counter to storage de error
k0=1;  % Counter of iterations
while Disc < 0 || Error(n) > 1e-3 || k0 < 13
    an = xn(1:Np+1,k0);  % Numerator coefficients
    bn = [xn(Np+2:2*Np+1,k0);1];  % Denominator coefficients
    Fsfit = (A1*an)./(A1*bn);  % Fitting evaluation
    vres = (Fsfit - b);  % Direct error
    n=n+1;  % Counter to storage de error
    Error(n)=sum(abs(vres));  % Error
    Disc = Error(n)-Error(n-1);  % Discriminating
    vres_r = real(vres);  % Real part of the vector vres
    vres_i = imag(vres);  % Imaginary part of the vector vres
    km = 1;  % (interleaved
    for k = 2:2:2*Xmax
        res(k-1,1) = vres_r(km);
        res(k,1) = vres_i(km);
        km = km+1;
    end

    Ds = std(res);  % Standard deviation
    res = res./Ds;  % Vector 'res' with standard deviation

% WEIGHING FUNCTIONS
if fw ==1, w = diag(W).*abs(res) + 1e-130; end
if fw ==2, w = diag(W).*(sqrt(1+((res.^2)))-1) + 1e-130; end
if fw ==3, w = diag(W).*1./(abs(res).^(1.2-2)) + 1e-130; end
if fw ==4, w = diag(W).*((res).^2)./(1+(res).^2) + 1e-130; end
if fw ==5, w = diag(W).*((res).^2)./sqrt(1+(res).^2)+ 1e-130; end
if fw ==6, w = diag(W).*(abs(res).^(1.2))./1.2 + 1e-130; end

W = diag(w);  % It updates the matrix W whit weighting functions
Bn_p = W*An;  % Weighting An
An_p = W*An;  % Weighting Bn
[Q,R] = qr(An_p,0);  % Matrix Q and R of An_p
At = R;  % It updates the matrixAn_p
B = Q.'*Bn_p;  % It updates the matrixBn_p

for col = 1:Ymax  % Euclidian norm
    Euclidian(col) = norm(At(:,col),2);
    At(:,col) = At(:,col)/.Euclidian(col);
end

k0=k0+1;  % Number of iterations
xn(:,k0) = At\B;  % Solution for the system (Ax = B)
xn(:,k0) = xn(:,k0)/.Euclidian.';  % Real solution
if k0 > 20, break, end  % End the while, if k0 > 20
end
4.6. Single-phase transmission line modeling

Bode, Levy or LS, IRLS, VF, and LM are applied to the rational approximation of the frequency-dependent parameters corresponding to a single-phase transmission line. In Figure 15a, the diagram of the equivalent circuit for the test with \( L = 70 \text{ mH} \) and \( R_i = 1.058 \Omega \) is shown. The current source is considered as an ideal 60 Hz. The line length \( l \) is 100 km, with a diameter of 2.7 cm. The conductor is placed horizontally at a height of 20 m, with a 100 \( \Omega \text{m} \) resistivity. The termination impedance at the end of line \( (R_e) \) is placed as 1e6 \( \Omega \), 1e-6 \( \Omega \), and 462 \( \Omega \) for evaluation of the different line end cases.
Figure 11. RMS-error on each iteration given by the IRLS method.

```matlab
%=======================================================
% Main program of LEVENBERG-MARQUARDT (Polynomials)
% Authors: Eduardo Salvador Bañuelos Cabral
% José Alberto Gutiérrez Robles
% Bjørn Gustavsen
%=======================================================
clc
clear all
close all
%% Initial Settings
Ns = 500; % Number of samples
f = logspace(-2,8,Ns); % Frequency *Hertz*
w = 2*pi.*f; % Frequency *rad/seg*
s = 1i*w; % Complex Frequency
%% Synthetic functions
choice = menu('CHOOSE A FUNCTION','Function F1(s)','Function F2(s)',...
'Function F3(s)','Function F4(s)','Function F5(s)');
if choice == 1
    ite = 150; % Number of iterations
    Np = 7; % Order of the proposed function
    Fs = (-364*s.^6-s.^5+635*s.^4-20802*s.^3+5304*s.^2+4520.*s-18020)./...
        (241*s.^7-32*s.^6+538*s.^5-2588*s.^4-22560*s.^3-604*s.^2+4150.*s-21052);
end
if choice == 2

```
ite = 50; % Number of iterations
Np = 12; % Order of the proposed function
Fs = (s.^12-0.4*s.^11+0.3*s.^10+6003*s.^9+600*s.^8+650*s.^7+54*s.^6+45*s.^5+
0.1*s.^4-1240*s.^3+80025*s.^2+5547*s+5008)./(s.^12-0.9*s.^11-0.2*s.^10-
4565*s.^9+34569*s.^8+55450*s.^7+60*s.^6+1005*s.^5+8050*s.^4+7502*s.^3+
10*s.^2+1014*s-50);
end

if choice == 3
ite = 50; % Number of iterations
Np = 14; % Order of the proposed function
Fs = 2.*pi*(652*s.^14-0.8*s.^13-0.8*s.^12-0.4*s.^11+25487*s.^10+600003*s.^9+
6040*s.^8-60050*s.^7+54*s.^6+45*s.^5+0.1*s.^4-1240*s.^3+8025*s.^2+5547*s+509)./
(2365*s.^14-8362*s.^13-8*s.^12-0.9*s.^11-65547*s.^10-4565*s.^9+34569*s.^8-
55450*s.^7+60*s.^6+1005*s.^5+8050*s.^4+7502*s.^3-10*s.^2-1014*s+75230);
end

if choice == 4
ite = 50; % Number of iterations
Np = 3; % Order of the proposed function
Ps = [-0.5; -0.1; -0.05];
Rs = [-2.2; -7192; 20];
% Process to obtain Fs from poles and residues
Ks = 5;
Fs = zeros(1,Ns);
for k = 1:length(Ps)
    Fs = Fs + (Rs(k)./(s.' - Ps(k))).';
end
Fs = Fs + Ks;
end

if choice == 5
ite = 100; % Number of iterations
Np = 11; % Order of the proposed function
Ps = 1.0e4*[0.9; -5.4; -27.3; 0.9; -9665; 235];-0.1; -3.9;46.3; -0.1; -3756];
Rs = 1.0e5*[834;22593;893;2653;654;32;44;6405;136;0.1;126];
% Process to obtain Fs from poles and residues
Ks = 10;
Fs = zeros(1,Ns);
for k = 1:length(Ps)
    Fs = Fs + (Rs(k)./(s.' - Ps(k))).';
end
Fs = Fs + Ks;
end

%% Levenberg Marquardt
[Ks,Rs,Ps,x,A1,an,bn,Ff] = LM_method(Np,Ns,Fs,s,ite);

% Fitting using partial fractions
Fs_fit = zeros(1,Ns);
for i = 1:Np
    Fs_fit = Fs_fit + (Rs(i)./(s.' - Ps(i))).';
end
Fs_fit = Fs_fit + Ks;

% Deviation in the fitted process
error = abs((Fs.' - Fs_fit.' ));

%% Plots
figure(1)
loglog(f,abs(Ps),'-k',f,abs(Fs_fit),'-.r',f,error,'-.b','LineWidth',2);
legend('Data','LM (Polynomials)','Deviation');
xlabel('Frequency [Hz]'); ylabel('Magnitude [p.u.]');
figure(2)
semilogy(Ff,'--b','LineWidth',2);
xlabel('Iteration count');ylabel('RMS-error');

Table 9. "Fitting_LM_Polynomials.m".

function [Ks,Rs,Ps,x,A1,an,bn,Ff] = LM_method(Np,Ns,Fs,s,ite)

x = ones((2*Np)+1,1); % Initial values
a = ones(Ns,1); % Vector column
a1 = zeros(Ns,Np); % Matrix size (s^n)
J1 = zeros(Ns,Np+1); % Matrix size (J1)
J2 = zeros(Ns,Np); % Matrix size (J1)
A1 = zeros(2*Np+1,2*Np+1); % Matrix size (AT)
en = zeros(2*Ns,1); % Vector size (en)
enp = zeros(2*Ns,1); % Vector size (enp)
Ff = zeros(ite,1); % Vector size (Ff)
Euclidian = zeros(1,1+2*Np); % Euclidian norm

% Construction of the matrix A1
for n = 1:1:Np
    a1(1:Ns,n) = s.^(n+1); % (AT) = [a1 s];
end
A1 = [a1 a];

v = 2; % LM coefficient (m = m*v)
u = 5; % LM coefficient (m = m/u)

% Levenberg-Marquardt method for kl = 1:ite % Main loop
\( x_1 = x(1:Np+1); \) \% Coefficients of the numerator
\( x_2 = [x(Np+2:2*Np+1);1]; \) \% Coefficients of the denominator
\( Y_g = (A1*x_1)/(A1*x_2); \) \% Fitting evaluation
\( \text{error} = (Y_g - F_s.'); \) \% Deviation
\% Process to obtain the Jacobian (\( J \))
for \( k = 1:1:Np \)
\( J1(:,k) = (s.\text{\textasciitilde}^((Np+1)-k))./(A1*x_2); \)
\( J2(:,k) = -(A1*x_1).\text{\textasciitilde}^((Np+1)-k))./(A1*x_2)^2; \)
end
\( J1(:,Np+1) = 1./(A1*x_2); \)
\( J = [J1 J2]; \)

\([Xmax Ymax] = \text{size}(J); \) \% Matrix size (\( J \))
\( Jc = \text{real}(J); \) \% Real part of vector \( J \)
\( Jl = \text{imag}(J); \) \% Imaginary part of vector \( J \)
\( \text{er} = \text{real(error); \} \% Real part of vector \( e \)
\( \text{ei} = \text{imag(error); \} \% Imaginary part of vector \( e \)
\( km = 1; \) \% Interleaved
for \( k = 2:2:2*Xmax \)
\( Jn(k-1,:) = Jc(km,:); \)
\( Jn(k,:) = Jl(km,:); \)
\( \text{en}(k-1,1) = \text{er}(km); \)
\( \text{en}(k,1) = \text{ei}(km); \)
\( km = km+1; \)
end
\( F = ((\text{norm}((\text{en},2)^2))); \) \% Objective function tends to zero to converge
\( F_f(ki,1) = F; \) \% Storage the RMS-error (objective function)
\([Q,R] = \text{qr}(Jn); \) \% Matrix \( Q \) and \( R \) of \( Jn \)
\( Jn = R; \) \% It updates matrix \( Jn \)
\( Gf = (Jn.'*Q.'); \) \% Gradient (QR-decomposition)
\( \text{Hess} = Jn.*Jn; \) \% Hessian (QR-decomposition)
if \( ki == 1 \)
\( m = \text{max}((\text{diag(Hess)})); \)
end
\( \text{paro} = 0; \) \% Variable to enter into the while loop
while (\( \text{paro}==0 \))
\( I = \text{diag}((\text{diag(Hess)})); \) \% Construction of matrix \( I \), diagonal of Hessian
\( \text{Hess}_\text{mod} = (\text{Hess} + m*I); \) \% Modified Hessian
for \( \text{col} = 1:Ymax \)
\( \text{Euclidian(col)} = \text{norm}(\text{Hess}_\text{mod}(:,\text{col}),2); \) \% Euclidian norm
\( \text{At}(:,\text{col}) = \text{Hess}_\text{mod}(:,\text{col})/\text{Euclidian(col); \}
end
\( h = (\text{At})\backslash Gf; \) \% Solution for the system (A\( x = b \))
\( h = h./\text{Euclidian}.; \) \% Real solution
\( xnew = x + h; \) \% New coefficients (without updating \( x \))
\% == Updated coefficients are evaluated, if meets the assessment then \( x \) is updated ==
\( x1new = xnew(1:Np+1); \) \% Coefficients of the numerator
\( x2new = xnew(Np+2:2*Np+1); \) \% Coefficients of the denominator
\( Y_gnew = (A1*x1new)/(A1*x2new); \) \% Fitting evaluation
\( \text{error} = (Y_gnew - F_s.'); \) \% Deviation
\( \text{ep} = \text{real(error); \} \% Real part of vector \( ep \)
\( \text{eip} = \text{imag(error); \} \% Imaginary part of vector \( ep \)
km = 1; % Interleaved
for k = 2:2:2*Xmax
    epn(k-1,1) = epr(km);
    epn(k,1) = epi(km);
    km = km+1;
end
Fp = ((norm(epn,2)^2)); % Objective function

% Updating of m
if (F < Fp)
    m = m*v;
end
if (F >= Fp) % Condition to go out the while
    x = x+h;
    m = m/u;
    break;
end
end

an = x(1:Np+1); % Coefficients of the numerator
bn = [x(Np+2:2*Np+1);1]; % Coefficients of the denominator

% Poles, residues and constant term
[Rs,Ps,Ks] = residue(an,bn);
TF = isempty(Ks);
if (TF==1)
    Ks = 0;
end

Table 10. “LM_method.m”.

Figure 12. Synthetic frequency response behavior and the rational approximation given by LM.
Figure 13. RMS-error on each iteration given by LM.
The modeling consists of calculating rational approximations for the line series impedance $Z$ and line shunt admittance $Y$ by applying the aforementioned rational fitting techniques. 

Figure 15b shows the behavior of $Z$ and $Y$ as a function of frequency, calculated with 16,384 linearly spaced samples and Figure 16 shows a flowchart for the implementation of this test case.
Figure 17a shows the results for the fitting of $Y$ with $N_p = 3$, in terms of model deviations (relative error) by Bode, Levy, IRLS, VF, and LM. It can be observed that the best approximations are obtained with Levy, IRLS, LM and VF which converge to practically the same deviation.

Figure 17b shows the results for the fitting of $Z$ with $N_p = 14$, in terms of the model deviations (relative error) by Bode, Levy, IRLS, VF, and LM. It can be seen that the best approximation is obtained with VF, IRLS, and LM.

Then, the effect of the modeling errors in $Y$ and $Z$ on a time domain response is evaluated. The line terminal nodal admittance matrix $Y_n$ is established with respect to the two line ends 2 and 3 of Figure 15a as

$$
\begin{bmatrix}
I_2 \\
I_3
\end{bmatrix} = Y_n
\begin{bmatrix}
V_2 \\
V_3
\end{bmatrix}
$$

where $Y_n$ is given by

\begin{verbatim}
% Function to implement INITIAL POLES for Vector Fitting
% Authors: Eduardo Salvador Bañuelos Cabral
% Jose Alberto Gutiérrez Robles
% Bjørn Gustavsen
%
% Inputs
% --- f, frequency
% --- Npol, number of poles
% Outputs
% --- Ps, poles

function [Ps] = InitialPoles(f,Npol)
% Set the initial poles (even, odd)
even = fix(Npol/2);
p_odd = Npol/2 - even;
disc = p_odd == 0;

% Set a real pole in case of disc == 1
if disc == 0
    pols = [ ];
else
    % Odd initial poles
    pols = [(max(f)-min(f))/2];
end

% Initial complex poles
bet = linspace(min(f),max(f),even);
for n=1:length(bet)
    alf=-bet(n)*1e-2;
    pols=[pols (alf-1j*bet(n)) (alf+1j*bet(n)) ];
end
Ps = pols.'; % Column vector of initial poles

Table 12. “InitialPoles.m”.
\end{verbatim}
and \( Y_c, A \) and \( B \) by

\[
Y_c(a) = Z(a)^{-1} \sqrt{Z(a)} Y(a)
\]
Figure 14. Synthetic frequency response data and the rational approximation given by VF.

Figure 15. (a) Diagram of the equivalent circuit for the test, (b) behavior of $Z(\omega)$ and $Y(\omega)$.

By using the Numerical inversion of Laplace Transform (NLT) [2, 12], the voltage responses on time domain ($v_1, v_2$ and $v_3$) are calculated for the cases of open-ended, short-circuited, and perfectly matched lines end. These results are considered as a reference solution.
The rational function-based models obtained with Bode, Levy, IRLS, VF, and LM are used to calculate \(Z\) and \(Y\). Using a similar procedure as mentioned for each technique, the voltage responses \((v_1, v_2, \text{and } v_3)\) are calculated through the NLT.

The objective is to calculate the error in the time domain introduced by the rational approximations of \(Z\) and \(Y\) in frequency domain, taking into account the reference solution.

The comparative results are shown in Figures 18-20 for the cases of short-circuited, perfectly matched, and open-ended lines end, respectively. The absolute errors are consistent with the deviations for the rational approximation of \(Z\).
Figure 18. (a) Time domain simulation of the voltage response at the short-circuited line end, (b) absolute error with respect to the reference solution ($v_3$).

Figure 19. (a) Time domain simulation of the voltage response at the matched line end, (b) absolute error with respect to the reference solution ($v_3$).

Figure 20. (a) Time domain simulation of the voltage response at the open line end, (b) absolute error with respect to the reference solution ($v_3$).
5. Conclusions

In this book, Bode, Levy, Noda, SK, IRLS, VF, LM, and DGN methods have been described in detail for complex-curve fitting, and a routine implemented in MATLAB environment presented for each technique. Rational approximation of synthetic frequency responses have used to show the operation of the programs. Moreover, the techniques are used to approximate the series line impedance ($Z$) and the shunt line admittance ($Y$) corresponding to a single-phase transmission line. The main conclusions are:

1. Asymptotic Approximation or Bode has proved to be a reliable technique for the modeling of overhead transmission lines. Also, this method can realize the fitting on the magnitude of the function and uses only real poles and zeros. However, the error level in the approximation can be substantial and it depends on the sensitivity criterion used when inserting new poles and zeros in the fitting.

2. The Levy method is pioneered for complex-curve fitting. Many techniques have been developed based on this methodology. Nevertheless, the method is biased, because it weights the fitting on high frequencies too much; this fact is demonstrated in the test cases.

3. The IRLS method is an accurate technique for the rational fitting of frequency responses. This method permits the implementation of different weighting functions in order to improve the level error in the approximation. Its disadvantage lies in the numerical ill-conditioning encountered in approximations with a wide frequency range.

4. The VF method is a robust and accurate technique. This fact has positioned this methodology as one of the most important in this field.

5. The LM method is a technique that shows good results in terms of level error. An advantage is that it can be implemented in pole-zero form, pole-residue form, and polynomial form.

6. The main disadvantage for the optimization techniques like LM and DGN is that they can get stuck in a local minimum.

7. The rational approximation for $Z$ and $Y$ in the single-phase transmission line modeling shows that the same technique does not always deliver the same result. Levy, LM and IRLS delivers more accurate result for the fitting of $Y$ while VF reach the best result for the fitting of $Z$. The error levels obtained in time domain simulations are consistent with the fitting of $Z$, because in the modeling of overhead transmission lines, these parameters are more relevant, therefore VF reach the best level error.

Author details

Eduardo Salvador Bañuelos-Cabral¹*, José Alberto Gutiérrez-Robles¹ and Bjørn Gustavsen²

*Address all correspondence to: ebanuelos@gdl.cinvestav.mx

1 University of Guadalajara, Guadalajara, Mexico
2 SINTEF Energy Research, Trondheim, Norway
References


