Research Article

Local Polynomial Regression Solution for Differential Equations with Initial and Boundary Values

Liyun Su, 1 Tianshun Yan, 1 Yanyong Zhao, 1 and Fenglan Li 2

1 School of Mathematics and Statistics, Chongqing University of Technology, Chongqing 400054, China
2 Library of Chongqing University of Technology, Chongqing 400054, China

Correspondence should be addressed to Liyun Su; cloudhopping@163.com

1. Introduction

Linear ordinary differential equation boundary value problems are very common theoretical and practical issues. Some solutions are put forward in [1–6]. Most solutions usually include finite difference method and shooting method. The shooting method is the most widely used. Both single and multishooting methods: (1) select the initial value of needing to be solved; (2) use the differential equation initial value algorithm to compute step by step and then get to the end of boundary value; (3) compute the error function between boundary value to compute and real boundary value then readjust the initial value of needing to be solved until the error is up to a certain level of accuracy. In general, this computing process is more tedious. This text makes use of local polynomial estimation to solve the boundary value issues of homogeneous and nonhomogeneous linear differential equations and compare it with the method in [4].

The linear differential equation below has a unique solution:

\[ y''(x) + p(x) y' + q(x) y = f(x), \]
\[ y(a) = y_0, \quad y(b) = y_1. \]  \hspace{1cm} (1)

Among them, \( y_0, y_1 \) are constant, and the function \( p(x), q(x), f(x) \) is continuous in \([a, b]\).

The presented method is useful for obtaining numerical approximations of differential equations, and it is also quite straightforward to write corresponding codes in any programming languages. Also, roundoff errors and necessity for large computer memory are not faced in this method. The computed results obtained by this way have been compared with the exact solution to show the required accuracy of it and the existing methods. Furthermore, the current method is of a general nature and can therefore be used for solving some partial differential equations arising in various areas.

2. Local Polynomial Estimator for Differential Equations

In this paper, we take advantage of the numerical estimation with the use of local polynomial regression to solve the boundary value problem. The basic idea of this method is first proposed in [5]. First, we introduce the mathematical thoughts of local polynomial regression. This idea was mentioned in [7–10]. Since the form of regression function is not specified, so the data points with long distance from \( x_0 \) provide little information to \( y(x_0) \). Therefore, we can only use
the local data points around \( x_0 \). We suppose that \( y(x) \) has \( p+1 \) derivative at \( x_0 \), by the Taylor expansion, for point \( x \), located in the neighborhood of this point \( x_0 \); we can use the \( p \)-order multivariate polynomials to locally approximate \( y(x) \) and the surrounding local point of \( x_0 \); we model \( y(x) \) as

\[
y(x) \approx \sum_{j=0}^{p} \beta_j (x - x_0) ^j,
\]

(2)

where parameter \( \beta_j \) depends on \( x_0 \), so-called local parameter. Obviously, the local parameter \( \beta_j = y^{(j)}(x_0)/j! \) fit the local model with local data, and it can be minimized:

\[
\sum_{i=1}^{n} \left( Y_i - \sum_{j=0}^{p} \beta_j (X_i - x_0) ^j \right) ^2 K_h (X_i - x),
\]

(3)

where \( h \) controls the size of the bandwidth of local area. Using matrix notation to represent the local polynomial regression model with local data, and it can be minimized:

\[
\sum_{i=1}^{n} \left( Y_i - \sum_{j=0}^{p} \beta_j (X_i - x_0) ^j \right) ^2 K_h (X_i - x),
\]

(3)

The weighted least squares problem (4) can be written as

\[
\min (Y - X\beta)^T W (Y - X\beta),
\]

(5)

here,

\[
W = \text{diag}(K_h (X_1 - x_0), \ldots, K_h (X_n - x_0)),
\]

(6)

so the solution vector is

\[
\beta = (X^T W X)^{-1} X^T W Y.
\]

(7)

For the initial value problem (1) and (3), we choose the best quadratic kernel function (Fan and Yao have proved in [10]):

\[
K_h (u) = \begin{cases} 
\frac{3}{4}(1 - u^2) , & \text{if } |u| \leq 1; \\
0 , & \text{otherwise}.
\end{cases}
\]

(8)

### 3. Local Polynomial Estimator

Consider the following:

\[
y'' (x) + p (x) y' (x) + q (x) = f (x) , \quad a \leq x \leq b,
y(a) = y_a , \quad y(b) = y_b.
\]

(9)

Here, we suppose that \( X_1 = a < X_2 < \cdots < X_n = b \), then we obtain the following relation:

\[
\sum_{j=0}^{p} \beta_j (X_i - x_0) ^j = f (x) , \quad a \leq x \leq b.
\]

(10)

Then, the corresponding minimization function is

\[
\left[ y_a - \sum_{j=0}^{p} \beta_j (X_1 - x_0) ^j \right] ^2 + \sum_{j=2}^{p-1} \left( f (X_i) - \sum_{j=0}^{p} j (j-1) \beta_j (X_i - x_0) ^{j-2} \right) \]

\[
- \sum_{j=1}^{p} \left( p (x) j \beta_j (X_i - x_0) ^{j-1} \right) ^2
\]

\[
+ \sum_{j=0}^{p} q (x) \beta_j (X_i - x_0) ^j
\]

(11)

\[
\left[ y_b - \sum_{j=0}^{p} \beta_j (X_n - x_0) ^j \right] ^2 K_h (X_i - x).
\]

We can write the solution vector:

\[
\tilde{\beta} = (\tilde{X}^T W \tilde{X})^{-1} \tilde{X}^T W \tilde{Y},
\]

(12)

where

\[
\tilde{Y} = \begin{pmatrix} 
y_a \\
f (X_2) \\
f (X_3) \\
\vdots \\
f (X_{n-1}) \\
y_b
\end{pmatrix}.
\]

(13)

In order to find the expression of \( \tilde{X} \), we need to find the elements of \( \tilde{X} \) at first.

Consider the following:

\[
i = 1, \quad a_{ij} = (X_i - x_0) ^j , \quad j = 0, \ldots, p
\]

\[
i = 2, \ldots, n - 1, \quad b_{ij} = j (j-1) (X_i - x_0) ^{j-2}, \quad j = 2, \ldots, p;
\]

\[
i = 0, \ldots, p;
\]

\[
i = n, \quad \alpha_{nj} = (X_n - x_0) ^j , \quad j = 1, \ldots, p.
\]

(14)
Table 1: Comparison of LPR and OPA.

<table>
<thead>
<tr>
<th>Parameter $\left(n, p, h\right)$</th>
<th>Residuals $e^2_n$</th>
<th>Pe of LPR $n$</th>
<th>Pe of OPA $n$</th>
<th>$h'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 21, p = 5, h = \pi/30$</td>
<td>$1.0435 \times 10^{-6}$</td>
<td>0.032</td>
<td>0.342</td>
<td>$\pi/2/100$</td>
</tr>
<tr>
<td>$n = 45, p = 6, h = \pi/60$</td>
<td>$4.0214 \times 10^{-8}$</td>
<td>0.0004</td>
<td>0.035</td>
<td>$\pi/2/1000$</td>
</tr>
<tr>
<td>$n = 60, p = 8, h = \pi/80$</td>
<td>$3.6503 \times 10^{-9}$</td>
<td>0.00001</td>
<td>0.003</td>
<td>$\pi/2/10000$</td>
</tr>
</tbody>
</table>

Consequently, we can get the $\bar{X}$

\[
\bar{X} = \begin{bmatrix}
    a_{10} & a_{11} & \cdots \\
    -c_{20} - t_{20} & b_{22} - c_{22} - t_{22} & \cdots \\
    -c_{30} - t_{30} & b_{32} - c_{32} - t_{32} & \cdots \\
    \vdots & \vdots & \ddots \\
    -c_{n-1,0} - t_{n-1,0} & b_{n-1,2} - c_{n-1,2} - t_{n-1,2} & \cdots \\
    a_{n0} & a_{n1} & \cdots & a_{np}
\end{bmatrix}_{n,p+1}.
\]

Substituting $\bar{X}$ and $\bar{Y}$ into

\[
\bar{\beta} = \left(\bar{X}^T W \bar{X}\right)^{-1} \bar{X}^T W \bar{Y}.
\]

Now, we can obtain the value of $\bar{\beta}$.

4. Example Simulation

In this part, we verified this method by solving three examples, solved the solution of specific differential problems and integration with local polynomial estimator, calculated the residual sum of squares, and compared with the existing algorithms. Calculations were performed by using MATLAB 7.0 programming solution.

4.1. Initial Value Problem of Homogeneous Linear Differential Equation

Example 1. Boundary value problem of linear differential equation with constant coefficients:

\[
y''(x) + y(x) = 0, \quad 0 \leq x \leq \frac{\pi}{2},
\]

\[
y(0) = 0, \quad y\left(\frac{\pi}{2}\right) = 1.
\]

(17)

The eigenvalue method can be used to obtain the real solution of initial value problem. Different parameters $n$, $h$, and $p$ correspond to different residual sum of square, given in Figure 1 and Table 1. LPR represents local polynomial regression. The exact solution is $y = \sin(x)$. In Figure 1, LPR is carried out for $n = 21$ and $p = 5$ giving $h = \pi/30$, and we get the $e^2_n$ and percentage error (Figure 1: Solution of local polynomial estimator and exact solution of Example 1).

In Table 1, comparison of the two methods, when $h'$ is $\pi/2/100$, we can get the percentage error of optimization algorithm. By comparing the two methods, we can know the difference of percentage error. Certainly, local polynomial regression has more advantages. Then, we can get better results for $n = 45$ and $p = 6$ given $h = \pi/60$, $n = 60$, $p = 8$, and $h = \pi/80$.

In Table 1, LPR represents local polynomial regression, OPA stands for optimization algorithm, Pe stands for Percentage error and $h$ represents step size in [3]. Residuals $e^2_n = \sum_{i=1}^{n} [y(X_i) - \bar{y}(X_i)]^2$.

Example 2. Boundary value problem of homogeneous linear differential equations:

\[
y'' - (2 \sin x) y' - \left(\cos x - \sin x^2\right) y = 0, \quad 0 \leq x \leq 1,
\]

\[
y(0) = 0, \quad y(1) = e^{-\cos 1}.
\]

(18)

![Figure 1: LPR of Example 1 with $n = 21, p = 5$ and $h = \pi/30$.](image-url)
Different parameters \( n, h, \) and \( p \) correspond to different residual sum of square, given in Figure 2 and Table 2. LPR represents local polynomial regression. The exact solution is \( y = xe^{-\cos x} \). In Figure 2, local polynomial estimator is carried out for \( n = 50 \) and \( p = 7 \) given \( h = 1/20 \), and we can get the residual sum of square \( e_n^2 \) which is very small. Here, \( e_n^2 = \sum_{i=1}^{n} [y(X_i) - \hat{y}(X_i)]^2 \). We conclude that local polynomial regression is more effective. Certainly, we can get better results for \( n = 60 \) and \( p = 9 \) given \( h = 1/35 \) (Figure 2: Solution of local polynomial estimator and exact solution of Example 2).

Example 3. Boundary value problem of nonhomogeneous linear differential equations boundary value problem:

\[
xy'' + 2(1-x)y' + (x-2)y = 2e^x, \quad y\left(\frac{1}{2}\right) = \frac{1}{2}e^{1/2}, \quad y(1) = e.
\]

The exact solution is \( y = xe^x \). Different parameters \( n, h, \) and \( p \) correspond to different residual sum of square, given in Figure 3 and Table 3. LPR represents local polynomial regression. In Figure 3, local polynomial estimator is carried out for \( n = 20 \) and \( p = 4 \) giving \( h = 1/11 \), and we can get the residual sum of square \( e_n^2 \) which is very small. Here, \( e_n^2 = \sum_{i=1}^{n} [y(X_i) - \hat{y}(X_i)]^2 \). In Table 3, local polynomial regression, it can be concluded that the effect of local polynomial regression is good and the residual sum of square \( e_n^2 \) is very small. It is sure that we can get better results for \( n = 70 \) and \( p = 10 \) giving \( h = 0.0045 \) (Figure 3: Solution of local polynomial estimator and exact solution of Example 3).

Figures 4, 5, and 6 present the absolute errors of this example under three groups of parameters. We can see that the absolute errors obtain \( \times 10^{-4} \) magnitude given different parameters \( n, h \) with the same \( p = 5 \). Moreover, with the increase in \( n \), the absolute errors decrease gradually. From Examples 1–3, we can conclude that the residual sum of square and absolute error are smaller by using local polynomial method which shows good accuracy.

5. Conclusions

In this paper, a local polynomial-based fitting method has been proposed for numerical solutions of differential equations with boundary values. Comparisons of the computed results with exact solutions and existing methods showed that the method has the capability of solving the differential equations with boundary values. It is also capable of obtaining
highly accurate results with minimal computational effort for both time and space. It was seen that the local polynomial-based technique approximates the exact solution very well. Numerical results demonstrated that the proposed method is more effective, and local polynomial estimator is more accurate compared with other algorithms, for instance, optimization algorithm. Furthermore, by using these methods, we can also study the numerical solutions of boundary value problems for differential equations on whole line or fractional differential equations which have been studied by the authors in [11–13]. In order to illustrate our method further, we will focus on these kinds of fractional differential, integral differential equations.

\section*{Acknowledgments}

This work was supported by the Natural Science Foundation Projects of CQ CSTC of China (CSTC2010BB2310, CSTC2011jjA40033, and CSTC2012jjA00037) and Chongqing CMEC Foundation of China (KJ120829).

\section*{References}
