The Solution of Second Order Homogeneous Linear Differential Equations

September 4, 2016

Abstract

Second order homogeneous linear differential equations are very common in mathematical physics problems. For example, Bessel equation, Legendre equation, which is the result of variables separation of Helmholtz equation in cylindrical coordinates and spherical coordinates, respectively. In this article, we use two common calculation software, Matlab and Mathemetica, to get the solution in the form of power series, and its general formula of coefficients, based on Fuchs Theorem and Frobenius Method. A brief proof of Fuchs Theorem and an introductory derivation of the two power series of a second order homogeneous linear equation are also included in our work. What's more, we also constructed a GUI module in Matlab to get the result and the graph of it once the coefficient of the equation is input. We hope that our work can be useful in numerical solution of mathematical physics equation, and furthermore, in practical mathematical physics problems.

Keywords: Second order homogeneous linear differential equations, Fuchs Theorem, Frobenius Method, general formula, Matlab, Mathemetica.

Contents

1 Fundamental symbolic derivation and numerical computation by Matlab
   1.1 The preliminary recursion formula numeration ........................................... 2
   1.2 The proof of convergence ........................................................................... 3
   1.3 Numerical computation by Matlab .............................................................. 4
   1.4 Reference .................................................................................................... 7

2 Further discussion and calculation of the solution based on Fuchs Theorem
   2.1 Frobenius Method ...................................................................................... 8
   2.2 Further calculation .................................................................................. 8
   2.3 Further Matlab implementation ............................................................... 9
   2.4 Complementary discussion ..................................................................... 10
   2.5 Reference ................................................................................................. 12

3 Analytical solution of second order ordinary differential equations in special cases by Mathematica
   3.1 Recurrence relations ............................................................................... 13
   3.1.1 The solution for the first power series $y_1(x)$ ...................................... 13
   3.1.2 The solution for the second power series $y_2(x)$ ................................. 14
   3.2 Algorithm and realization using Mathematica ........................................... 14
   3.2.1 Part of source code ............................................................................. 14
   3.2.2 Examples ............................................................................................ 15

1
1 Fundamental symbolic derivation and numerical computation by Matlab

Person in charge: Cong Cong (丛聪)

1.1 The preliminary recursion formula numeration

Firstly, we have a second order homogeneous linear differential equation like (1), and we want to prove and find a convergent solution of generalized power series in the neighbourhood of the regular singular point \( x_0 \).

\[
A(x)y'' + B(x)y' + C(x)y = 0 \tag{1}
\]

Suppose that the differential equation can be adapted in such form.

\[
(x - x_0)^2 P(x)y'' + (x - x_0)Q(x)y' + R(x)y = 0 \tag{2}
\]

In the equation (2), \( P(x) \), \( Q(x) \) and \( R(x) \) can be expanded to series of powers of \( (x - x_0) \) in the \( x_0 's \) neighbourhood, and to meet the requirements of the regular singular point, there are some constraints that \( P(x_0) \neq 0 \) and \( Q(x_0), R(x_0) \) do not equal to 0 at the same time.

Suppose that the differential equation have a solution of generalized powers series which is like this.

\[
y = \sum_{n=0}^{+\infty} C_n(x - x_0)^{n+\rho} \tag{3}
\]

Using the settings above, in \( x_0 's \) neighbourhood \( |x - x_0| < r \), the equation can be rewritted in this way.

\[
(x - x_0)^2 y'' + (x - x_0) \sum_{k=0}^{+\infty} a_k (x - x_0)^k y' + \sum_{k=0}^{+\infty} b_k (x - x_0)^k y = 0 \tag{4}
\]

Substituing the assumed solution in (4), we can get a new equation of these generalized power series

\[
(x - x_0)^\rho \left\{ \sum_{k=0}^{+\infty} (k + \rho)(k + \rho - 1) C_k(x - x_0)^k + \sum_{k=0}^{+\infty} a_k (x - x_0)^k \cdot \sum_{k=0}^{+\infty} (k + \rho)C_k(x - x_0)^k \right. \\
\left. + \sum_{k=0}^{+\infty} b_k (x - x_0)^k \cdot \sum_{k=0}^{+\infty} C_k(x - x_0)^k \right\} = 0 \tag{5}
\]

Eliminating the common factor \( (x - x_0)^\rho \) and using the uniqueness to compare the coefficients, we can get those recursion formulas.

\[
\begin{align*}
C_0 f_0(\rho) &= 0 \\
C_1 f_0(\rho + 1) + C_0 f_1(\rho) &= 0 \\
\cdots &
\end{align*}
\]

\[
C_k f_0(\rho + k) + C_{k-1} f_1(\rho + k - 1) + \cdots + C_0 f_k(\rho) = 0 \\
\cdots &
\tag{6}
\]

The \( f_j(\rho) \) in the formulas is defined in the ways like this.

\[
\begin{align*}
f_0(\rho) &= \rho(\rho - 1) + a_0 \rho + b_0 \\
f_j(\rho) &= a_j \rho + b_j & (j > 0)
\end{align*}
\]

Because of the requirement \( C_0 \neq 0 \), the \( f_0(\rho) \) is equal to 0, and then we can find \( \rho \) easily.

\[
\rho(\rho - 1) + a_0 \rho + b_0 = 0 \tag{7}
\]
Thus, we have two real roots or a pair of conjugate complex roots. We can define $\rho_1$ as the bigger one, or any one among the pair of conjugate complex roots.

\[
\begin{cases}
  f_0(\rho_1) = 0 \\
  f_0(\rho_1 + j) \neq 0
\end{cases}
\]  

(9)

So, for the index $\rho = \rho_1$, we can use the recursion formulas to find the coefficients $C_k$ and a solution of generalized power series for the second order homogeneous linear differential equation.

\[
y = \sum_{n=0}^{+\infty} C_n(x - x_0)^{n+\rho_1}
\]  

(10)

1.2 The proof of convergence

Defining that $\rho_1 - \rho_2 = m$, we are easily to learn that $Rm(m) \geq 0$, but because the series $\sum_{k=0}^{+\infty} a_k(x - x_0)^k$ and $\sum_{k=0}^{+\infty} b_k(x - x_0)^k$ are convergent in the internal $|x - x_0| < r$. For appointed $0 < r_1 < r$, we can certainly find $M > 0$(for convenience $M > 1$), to make that.

\[
|a_k| \leq \frac{M}{r_1^s}, \quad |b_k| \leq \frac{M}{r_1^s}, \quad |\rho_1 a_k + b_k| \leq \frac{M}{r_1^s}
\]  

(11)

Because $f_0(\rho_1) = 0$, $\rho_1 + \rho_2 = 1 - a_0$ and $\rho_1 - \rho_2 = m$, we have these statements.

\[
f_0(\rho_1 + k) = k(k + m)
\]  

(12)

And using the inequations above, we can get this.

\[
|C_1| = \frac{|\rho_1 a_1 + b_1|}{|f_0(\rho_1 + 1)|} \leq \frac{M}{r_1|\rho_1 + 1|} \leq \frac{M}{r_1}
\]  

(13)

Using the second type of mathematical induction, we suppose that.

\[
|C_k| \leq \left( \frac{M}{r_1} \right)^k \quad (k = 1, 2, \ldots).
\]  

(14)

It works when $k = 1$, we can suppose that it works when $k = 1, 2, \ldots, s - 1$, and set $C_0 = 1$. We want to prove that it also works when $k = s$.

\[
|C_s| = \left| \sum_{j=1}^{s-1} C_j f_s-j(\rho_1 + j) + C_0 f_s(\rho_1) \right| \leq \sum_{j=1}^{s-1} |C_j| \cdot |(\rho_1 + j)a_{s-j} + b_{s-j}| + |\rho_1 a_s + b_s|
\]

\[
\leq \sum_{j=1}^{s-1} |C_j| |\rho_1 a_{s-j} + b_{s-j}| + |ja_{s-j}| \leq \frac{1}{2} s(s + 1) \left( \frac{M}{r_1} \right)^s < \left( \frac{M}{r_1} \right)^s
\]  

(15)

So we can get that.

\[
|C_k| \leq \left( \frac{M}{r_1} \right)^k \quad (k = 1, 2, \ldots).
\]  

(16)

Thus, for arbitrary constant $r_2$, which satisfy $0 < r_2 < r_1$, the generalized power series are convergent in $0 < |x - x_0| \leq \frac{r_2}{2}$. On the other hand, when the index are complex-valued, we can get a complex solution of the generalized power series. But the coefficient or the initial equation are real, so we can divide its real part and complex part, acquiring two real solution of series theoretically.
1.3 Numerical computation by Matlab

Here is the main source code of the Matlab program, in which \( A(x), B(x), C(x) \) are coefficient polynomials, 'size' is the number of terms in the output, and \( x_0 \) is the canonical acnode.

\[
\text{syms } x \\
\text{syms rou} \\
\text{syms rous} \\
A = \text{get(handles.edit1, 'String')}; \\
B = \text{get(handles.edit2, 'String')}; \\
C = \text{get(handles.edit3, 'String')}; \\
cm = \text{str2double(get(handles.edit7, 'String'))}; \\
x0 = \text{str2double(get(handles.edit4, 'String'))}; \\
size = \text{str2double(get(handles.edit5, 'String'))}; \\
step = \text{str2double(get(handles.edit8, 'String'))}; \\
up = \text{str2double(get(handles.edit9, 'String'))}; \\
down = \text{str2double(get(handles.edit10, 'String'))}; \\
\%
\text{Input} \\
\text{for } k = 0:1: \text{size}−1 \%	ext{Using taylor expansion to calculate } a_k \text{ and } b_k \\
\text{if } k==0 \\
f1 = 0; \\
\text{else if } k>0 \\
f1 = \text{taylor}(Q/P, x, x0, 'Order', k); \\
\text{end} \\
f2 = \text{taylor}(Q/P, x, x0, 'Order', k+1); \\
a(k+1) = \text{subs}((f2−f1)/x^k, x, 1); \\
\text{if } k==0 \\
f3 = 0; \\
\text{else if } k>0 \\
f3 = \text{taylor}(R/P, x, x0, 'Order', k); \\
\text{end} \\
f4 = \text{taylor}(R/P, x, x0, 'Order', k+1); \\
b(k+1) = \text{subs}((f4−f3)/x^k, x, 1); \\
\text{end} \\
\text{for } k = 2:1: \text{size} \% \text{Taking the } u_2 \text{ as an example.} \\
qiuhe = 0; \\
\text{for } n = 1:1:k−1 \\
qiuhe = qiuhe + c1(n) * \text{subs}(fx(k−n+1), rou, rous+n−1); \\
\text{end} \\
\text{temp1 = qiuhe; \\
}\text{temp2 =subs(fx(1), rou, rous+k−1); \%	ext{If there is a '0/0' } \\
\text{while (subs(temp1, rou, rou2)==0 && subs(temp2, rou, rou2)==0) \%	ext{If} \\
\text{temp1 = diff(temp1, rou, 1); \\
}\text{temp2 = diff(temp2, rou, 1); \\
\text{end} \\
\text{if (subs(temp2, rou, rou2)==0) 
}
key = 0;
break;
else
c1(k) = subs(temp1/temp2, rous, rou2); % To calculate c_k of u_2
  u2 = u2 + x ^ (rou2) * c1(k) * x ^ (k - 1);
end
end

set(handles.text10, 'String', char(u1)); % Output
set(handles.text11, 'String', char(u2));

Thus, using the GUI module in Matlab, we have these kinds of output in the graphical interface (like figure 1 to 3).
Moreover, in the second picture, the 'u2' is created at first, and A(x), B(x), C(x) are setted to meets the needs of 'u2'.The result shows that the code works well.

In addition, as for Fuchs Theorem, a appropriate second order homogeneous linear differential equations would have two linearly independent solutions.

\[ y_1(x) = (x - x_0)^\alpha \sum_{k=1}^{\infty} C_k (x - x_0)^k \]  \hspace{1cm} (17)

\[ y_2(x) = \alpha y_1(x) \ln(x - x_0) + (x - x_0)^\alpha \sum_{k=1}^{\infty} C_k (x - x_0)^k \]  \hspace{1cm} (18)
1.3 Numerical computation by Matlab

**Figure 2:** Unknown function with singular point

**Figure 3:** Trigonometric function
We can find that the second solution do not have the same form as the Matlab puts forward, and this difference would be discussed in the third part of this Mini Thesis.

1.4 Reference

2 Further discussion and calculation of the solution based on Fuchs Theorem

2.1 Frobenius Method

For second-order homogeneous linear ODE, it can be written in the form below:

\[ y'' + p(z)y' + q(z)y = 0 \]  \hspace{1cm} (19)

if \( z_0 \) is an acnode of \( p(z) \) of at most first order and an acnode of \( q(z) \) of at most second order, we call \( z_0 \) an canonical acnode of equation(19). The following theorem is valid near the canonical acnode.

Fuchs Theorem

Assume that \( z_0 \) is a canonical acnode of equation(19), namely \( (z - z_0)p(z), (z - z_0)^2q(z) \) are holomorphic in \( |z - z_0| < R \), so there are two linear independent solution in the deleted neighborhood \( 0 < |z - z_0| < R \)

\[ w_1(z) = (z - z_0)^{\rho_1} \sum_{n=0}^{+\infty} c_n(z - z_0)^{n+\rho_1} \]  \hspace{1cm} (20)

\[ w_2(z) = \alpha w_1(z)ln(z - z_0) + \sum_{n=0}^{+\infty} d_n(z - z_0)^{n+\rho_2} \]  \hspace{1cm} (21)

in which the coefficient \( a_0b_0 \neq 0 \), constant \( \alpha \) can be 0, \( \rho_1 \), \( \rho_2 \) are indexes of the solutions.

The subject of this part is the method to derive recurrence formula of \( c_n \) and \( d_n \), as well as writing a Matlab function to get numeric values of \( c_n \) and \( d_n \) with a given number of terms, and output the two solutions in the form of power series.

We might assume that \( z = 0 \) is a canonical acnode of the original equation. And we set \( p(z)(z - z_0) = \sum_{n=0}^{+\infty} a_n(z - z_0)^n, q(z)(z - z_0)^2 = \sum_{n=0}^{+\infty} b_n(z - z_0)^n \).

It has been solved in the first part that the indexes of the solutions satisfies \( \rho(\rho - 1) + a_0\rho + b_0 = 0 \), as a result,

\[ c_n = -\frac{\sum_{k=0}^{n-1} c_k(a_{n-k}(k + \rho) + b_{n-k})}{\rho^2 + (2n - 1)\rho + n(n - 1) + a_0(n + \rho) + b_0} \]  \hspace{1cm} (22)

Because the original equation is homogeneous, we might set \( c_0 = 1 \). Then we can get all \( c_n \).

In the following part, we focus on the calculation of \( d_n \).

Case 1: \( (\rho_1 - \rho_2) \) is not an integer.

In this case, \( d_n \) is similar to \( c_n \),

\[ d_n = -\frac{\sum_{k=0}^{n-1} d_k(a_{n-k}(k + \rho_2) + b_{n-k})}{\rho_2^2 + (2n - 1)\rho_2 + n(n - 1) + a_0(n + \rho_2) + b_0} \]  \hspace{1cm} (23)

Set \( d_0 = 1 \), thus we can calculate every \( d_n \).

Case 2: \( \rho_1 = \rho_2 = 0 \)

Define a differential operation \( L[y] = x^2(y'' + p(x)y' + q(x)) \). In this case that \( \rho_1 = \rho_2 \), \( L[y] = c_0x^\rho(\rho - 1)^2 \), differentiate with respect to \( \rho \), we have

\[ L\frac{\partial y}{\partial \rho} = c_0lnx \cdot x^\rho(\rho - 1)^2 + c_0x^\rho \cdot 2(\rho - 1) \]  \hspace{1cm} (24)

\[ (x - x_0)^2y'' + (x - x_0) \sum_{k=0}^{+\infty} a_k(x - x_0)^k y' + \sum_{k=0}^{+\infty} b_k(x - x_0)^ky = 0 \]  \hspace{1cm} (25)
2 FURTHER DISCUSSION AND CALCULATION OF THE SOLUTION BASED ON FUCHS

2.2 Further calculation

\[ L[y] \text{ and } L[\frac{\partial y}{\partial \rho}] \text{ both vanish when } \rho = \rho_1. \] Therefore, if \( y = \sum_{n=0}^{+\infty} c_n x^{n+\rho_1} \), in which \( c_n \) are determined as functions of \( \rho \) using the recurrence function (22), we can get the two linear independent solutions of equation (19)

\[ y_1(x) = (y)_{\rho = \rho_1} = x^{\rho_1} \sum_{n=0}^{+\infty} (c_n)_{\rho = \rho_1} x^n \]

\[ y_2(x) = (\frac{\partial y}{\partial \rho})_{\rho = \rho_1} = y_1(x) \ln x + x^{\rho_1} \sum_{n=1}^{+\infty} (\frac{\partial c_n}{\partial \rho})_{\rho = \rho_1} x^n \]

Great attention should be paid here because in \( y_2(x) \), the series begin at the first order of \( x \), instead of zero-order.

Case 3: \( \rho_1 - \rho_2 = m(\text{m is a positive integer}) \)

In this case, set

\[ c_0 = c_0'(\rho - \rho_2) \]

in which \( c_0' \) is a random constant.

Then we have

\[ L[y] = c_0' x^\rho (\rho - \rho_1)(\rho - \rho_2)^2 \]

Its derivative with respect to \( \rho \) is

\[ L[\frac{\partial y}{\partial \rho}] = c_0' \ln x \cdot x^\rho (\rho - \rho_1)(\rho - \rho_2)^2 + c_0' x^\rho \cdot (\rho - \rho_2)^2 + c_0' x^\rho \cdot 2(\rho - \rho_1)(\rho - \rho_2) \]

Therefore, \( (y)_{\rho = \rho_1}, (y)_{\rho = \rho_2} \) and \( (\frac{\partial y}{\partial \rho})_{\rho = \rho_1} \) are both solutions of \( L[y] = 0 \), namely solutions of equation (19). \( (y)_{\rho = \rho_1} \) has been discussed before. Because we have \( c_0 = c_0' (\rho - \rho_2) \) now, all the \( c_1, c_2, \ldots, c_{m-1} \) determined by equation (22) contain the factor \( (\rho - \rho_2) \), so

\[ (c_1)_{\rho = \rho_2} = (c_2)_{\rho = \rho_2} = \cdots = (c_{m-1})_{\rho = \rho_2} = 0 \]

\( (c_m)_{\rho = \rho_2} \) can be any constant. This is because in equation (22) which determines \( c_m \), when \( n = m \), the dominator

\[ \rho^2 + (2n - 1)\rho + n(n - 1) + a_0(n + \rho) + b_0 = (\rho - \rho_2)(\rho + m - \rho_2) \]

also have the factor \( (\rho - \rho_2) \). \( (c_m)_{\rho = \rho_2} \) can be expressed with \( (c_m)_{\rho = \rho_2} \), and recurrence equation (22) is satisfied. So \( (y)_{\rho = \rho_1} \) and \( (y)_{\rho = \rho_2} \) are not the two linear independent solutions that we want.

The second solution which is linear independent on \( (y)_{\rho = \rho_1} \) is

\[ y_2(x) = (\frac{\partial y}{\partial \rho})_{\rho = \rho_2} = \ln x \cdot x^{\rho_2} \sum_{n=m}^{+\infty} (c_n)_{\rho = \rho_2} x^n + x^{\rho_2} \sum_{n=0}^{+\infty} (\frac{\partial c_n}{\partial \rho})_{\rho = \rho_2} x^n \]

The first term differs from \( y_1(x) \ln x \) at most for a constant coefficient.

2.2 Further calculation

This part mainly focuses on the methods of calculations when \(\rho_1 - \rho_2 = m(\text{m is an integer}). We can know from the 'Case 3' in the last section that \( d_n = (\frac{\partial c_n}{\partial \rho})_{\rho = \rho_2}. \)

When \( \rho_1 - \rho_2 = \text{positive integral}, c_0 = c_0'(\rho - \rho_2), \) according to recurrence equation (22), for \( n < m, \)

\[ c_n = \frac{-\sum_{k=0}^{n-1} c_k(a_{n-k}(k + \rho) + b_{n-k})}{\rho^2 + (2n - 1)\rho + n(n - 1) + a_0(n + \rho) + b_0} \]

\[ = \frac{-\sum_{k=0}^{n-1} c_k(a_{n-k}(k + \rho) + b_{n-k})}{\rho^2 + (2n - 1)\rho + n(n - 1) + a_0(n + \rho) + b_0} (\rho - \rho_2) \]
Dominator \( \rho^2 + (2n-1)\rho + n(n-1) + a_0(n + \rho) + b_0 = (\rho - \rho_2)(\rho + \rho_2 + 2m - 1 + a_0) \) vanishes at \( n = 0 \) and \( n = m \), but not at other \( n \).

Differentiate both sides of equation (34) with respect to \( \rho \), and substrate with \( \rho = \rho_2 \), we have

\[
\frac{\partial c_n}{\partial \rho} \bigg|_{\rho = \rho_2} = -\frac{\sum_{k=0}^{n-1} c_k'(a_{n-k}(k + \rho) + b_{n-k})}{\rho^2 + (2n-1)\rho + n(n-1) + a_0(n + \rho) + b_0} = c_n' \quad (n < m)
\]  

(35)

\[
c_m = -\frac{\sum_{k=0}^{m-1} c_k(a_{m-k}(k + \rho) + b_{m-k})}{\rho^2 + (2m-1)\rho + m(m-1) + a_0(m + \rho) + b_0} = -\frac{\sum_{k=1}^{m-1} c_k'(a_{m-k}(k + \rho) + b_{m-k})}{\rho + \rho_2 + 2m - 1 + a_0} = \frac{\sum_{k=0}^{m-1} c_k'(a_{m-k}(k + \rho) + b_{m-k})}{\rho^2 + (2m-1)\rho + m(m-1) + a_0(m + \rho) + b_0}
\]  

(36)

\[
\frac{\partial c_m}{\partial \rho} = -\frac{\sum_{k=0}^{m-1} c_k'a_{m-k} + (a_{m-k}(k + \rho) + b_{m-k})\frac{\partial a_k'}{\partial \rho} + \sum_{k=0}^{m-1} c_k'(a_{m-k}(k + \rho) + b_{m-k})}{\rho^2 + (2m-1)\rho + m(m-1) + a_0(m + \rho) + b_0}
\]  

(37)

Because \( c_0' \) is random (For equation (19) is linear, we can set \( c_0' = 1 \), so \( \frac{\partial c_0'}{\partial \rho} = 0 \), for \( 0 < n < m \),

\[
\frac{\partial c_n'}{\partial \rho} = -\frac{\sum_{k=0}^{n-1} c_k'a_{n-k} + (a_{n-k}(k + \rho) + b_{n-k})\frac{\partial a_k'}{\partial \rho}}{\rho^2 + (2n-1)\rho + n(n-1) + a_0(n + \rho) + b_0} + \frac{(2n + 2m - 1 + a_0)\sum_{k=0}^{n-1} c_k(a_{n-k}(k + \rho) + b_{n-k})}{(\rho^2 + (2n-1)\rho + n(n-1) + a_0(n + \rho) + b_0)}
\]  

(38)

So, based on (35) and (38) we can get all the values of \( c_k' \) and \( \frac{\partial a_k'}{\partial \rho} \) \( k = 0, 1, \cdots, (m - 1) \) by recursion. And based on (37), we can calculate \( \frac{\partial c_m}{\partial \rho} \), and substrate with \( \rho = \rho_2 \), then it is just the \( d_m \).

for \( n > m \),

\[
c_n = -\frac{\sum_{k=0}^{n-1} c_k(a_{n-k}(k + \rho) + b_{n-k})}{\rho^2 + (2n-1)\rho + n(n-1) + a_0(n + \rho) + b_0} = -\frac{\sum_{k=1}^{n-1} c_k'(a_{n-k}(k + \rho) + b_{n-k})}{\rho^2 + (2n-1)\rho + n(n-1) + a_0(n + \rho) + b_0}(\rho - \rho_2)
\]  

(39)

\[
\frac{\partial c_n}{\partial \rho} \bigg|_{\rho = \rho_2} = -\frac{\sum_{k=0}^{m-1} c_k'(a_{n-k}(k + \rho) + b_{n-k})}{\rho^2 + (2n-1)\rho_2 + n(n-1) + a_0(n + \rho_2) + b_0} + \frac{(2n + 2m - 1 + a_0)\sum_{k=0}^{m-1} c_k(a_{n-k}(k + \rho_2) + b_{n-k})}{(\rho^2 + (2n-1)\rho_2 + n(n-1) + a_0(n + \rho_2) + b_0)}
\]  

(40)

So far, we have derived all the recurrence relationships of \( d_n \).

2.3 Further Matlab implementation

Introduction: Program in this part is developed based on the program in the first part of this essay. It shares the same essential thinking with the one in the first part. Main improvement is on the calculation of \( d_n \). Three new arrays: d1, d2 and d2 are built in this program. d is used for storage of \( d_n \). The first \( m \) terms of d1 is used to store \( c_k'(0 \leq k \leq m-1) \), and the terms after the \( (m+1) \) th of d1 are used to store \( c_k(k \geq m) \). The length of d2 is \( m \), it stores \( \frac{\partial c_k}{\partial \rho}(0 \leq k \leq m-1) \). We perform the recurrence calculation respectively in the 3 cases discussed in 3.2.

Source code: Here is a brief skeleton of source code of the Matlab program, in which \( A(x), B(x), C(x) \) are coefficient functions, size is the number of terms in the output, and \( x0 \) is the canonical acnode. The full code is in the supplementary materials.
2.3 Further Matlab implementation

```matlab
clear;
syms x
A = x;
B = 1;
C = x;
x0 = 0;
size = 15; % Input%
a(size) = 0;
b(size) = 0;
c(size) = 0;
d(size) = 0; % Initialization, c and d are the results we want.%
% Taylor expansion of the coefficient functions of the equation, omitted here.%
rho1 = (1 - a(1) + sqrt((a(1) - 1).^2 - 4 * b(1))) / 2;
rho2 = (1 - a(1) - sqrt((a(1) - 1).^2 - 4 * b(1))) / 2;
c(1) = 1;
d(1) = 1;
m = rho1 - rho2; % Result of the parameter 'rho' and the difference between two 'rho's
if rem(m, 1) == 0
    d1(size) = 0;
    if m == 0 % Case 3: m is a non-zero integer%
        % Calculation of d(n), omitted here%
    end
    if m == 0 % Case 2: m equals to zero%
        % Calculation of d(n), omitted here%
    end
else % Case 1: m is not an integer%
    % Calculation of d(n), omitted here%
end
u1 = u1 - x;
u2 = u2 - x;
for i = 1: size
    u1 = u1 + c(i) * x^(i - 1 + rho1);
    u2 = u2 + d(i) * x^(i - 1 + rho2);
end
rho1
u1
rho2
u2
m % Output%

Examples: 0-order Bessel equation xy'' + y' + xy = 0 (set in the source code), the output is

rho1 =
0
u1 =
- (2974531672253493*x^14)/1237940039285380274899124224 +
(4554751623138161*x^12)/9671406556917033397649408 -
```

Examples: 0-order Bessel equation $xy'' + y' + xy = 0$ (set in the source code), the output is

$\rho_1 = 0$

$u_1 = - \frac{(2974531672253493x^{14})}{1237940039285380274899124224} + \frac{(4554751623138161x^{12})}{9671406556917033397649408}$
(5124095576030431*x^10)/75557863725914323419136 + 
(2001599834386887*x^8)/295147905179352825856 - x^6/2304 + x^4/64 - 
x^2/4 + 1 

\( \rho_2 = 0 \) 

\( u_2 = \frac{(7712535693057269*x^14)/123794003928538027489912424}{(2789785369172123*x^12)/2417851639229258349412352} + 
\frac{(585009115968075*x^10)/37778931862957161709568}{(416999654972681*x^8)/295147905179352825856} + \frac{(11*x^6)/13824}{(3*x^4)/128} + x^2/4 \) 

\( m = 0 \) 

Compare this output with 0-order Bessel function \( J_0(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(k!)^2} \left( \frac{x}{2} \right)^{2k} \) and 0-order Neumann function \( N_0(x) = 2J_0(x)(\ln \frac{x}{2} + \gamma) - \frac{2}{\pi} \sum_{k=0}^{\infty} \frac{(-1)^k}{(k!)^2} \left( \sum_{l=0}^{k-1} \frac{1}{l+1} \right) \left( \frac{x}{2} \right)^{2k} \), we can find the output is right.

### 2.4 Complementary discussion

A special condition is discussed here:

\[ \sum_{k=0}^{m-1} c'_k(a_{m-k}(k + \rho_2) + b_{m-k}) = 0 \]  \( \text{(41)} \)

In this condition, we can use L'Hospital's law to define

\[ c'_m = -\frac{\partial(\sum_{k=0}^{m-1} c'_k(a_{m-k}(k + \rho) + b_{m-k}))/\partial \rho |_{\rho = \rho_2}}{\partial(\rho^2 + (2m - 1)\rho + n(n - 1) + a_0(n + \rho) + b_0)/\partial \rho |_{\rho = \rho_2}} \]

\[ = -\frac{\sum_{k=0}^{m-1} c'_k(a_{m-k}(k + \rho) + b_{m-k}) \frac{\partial c'_k}{\partial \rho |_{\rho = \rho_2}}}{\rho + \rho_2 + 2m - 1 + a_0} \]

\[ = -\frac{\sum_{k=0}^{m-1} c'_k(a_{m-k}(k + \rho_2) + b_{m-k}) \frac{\partial c'_k}{\partial \rho |_{\rho = \rho_2}}}{2 \rho_2 + 2m - 1 + a_0} \]  \( \text{(42)} \)

It is obvious that \( c'_m = \left( \frac{\partial c}{\partial \rho} \right)_{\rho = \rho_2} \). So, in this special condition, \( c_n \) can be recurred directly into the scope of \( n \geq m \).

\[ c'_n = -\frac{\sum_{k=0}^{n-1} c'_k(a_{n-k}(k + \rho) + b_{n-k})}{\rho^2 + (2n - 1)\rho + n(n - 1) + a_0(n + \rho) + b_0} \]  \( \text{(43)} \)

\[ c_n = (\rho - \rho_2)c'_n \]  \( \text{(44)} \)

So \( c'_n = \left( \frac{\partial c}{\partial \rho} \right)_{\rho = \rho_2} \). That is to say we can derive \( d_n \) just by recuring \( c'_n \) directly, and \( c'_n = d_n \).

### 2.5 Reference

[1] 季孝达，薛兴恒，陆英，宋立功. 2009. 数学物理方程. 科学出版社
3 Analytical solution of second order ordinary differential equations in special cases by Mathematica

Person in charge: Tianzhe Zheng (郑天哲)

3.1 Recurrence relations

In the first section, Cong Cong has proved that for any second order homogeneous linear differential equation, there is a convergent solution of generalized power series in the neighborhood of the regular singular point $x_0$. In this section, let us focus on some special cases: the coefficients of the ODE are polynomials:

$$A(x - x_0)(x - x_0)^2 y'' + B(x - x_0)(x - x_0)y' + C(x - x_0)y = 0$$  \hspace{1cm} (45)

Of course, $A, B, C$ should meet the regular singular requirement. That means $A(x - x_0), B(x - x_0), C(x - x_0)$ are all polynomials. $x_0$ is regular singular point of the ODE. Define $n$ is the maximum exponent of $A(x - x_0), B(x - x_0), C(x - x_0)$.

The reason we choose not to solve analytical solutions are as follows:

Firstly, the software Mathematica cannot solve recurrence equations following:

$$f(c_1, c_2, ..., c_n, c_{n+1}) = 0, \text{ for all } n$$  \hspace{1cm} (46)

However, equations (45) shows that we have to use $c_1, c_2, ..., c_n$ to solve $c_{n+1}$, thus we cannot use Mathematica to solve this complicate problem.

Secondly, many differential equations we meet have polynomial coefficients, and in scientific research we would like to do some approximation when we meet an equation which is too hard to solve. In many cases we simplify the problem and finally only need to solve the equations which people are familiar with and often have polynomial coefficients.

The convergent solutions of generalized power series are given by Fuchs Theorem:

$$y_1(x - x_0) = (x - x_0)^{\nu} \sum_{k=1}^{\infty} Y_k (x - x_0)^k$$  \hspace{1cm} (47)

$$y_2(x - x_0) = \alpha y_1(x - x_0) \ln(x - x_0) + (x - x_0)^{\nu} \sum_{k=1}^{\infty} Y_k (x - x_0)^k$$  \hspace{1cm} (48)

$a_0 b_0 \neq 0, \alpha$ could be zero.

3.1.1 The solution for the first power series $y_1(x)$

Substitute (49) with (46), we can get:

$$\sum_{k=0}^{\infty} a_k (x - x_0)^k \sum_{\rho=0}^{\infty} (k + \rho)(k + \rho - 1) Y_{k + \rho} (x - x_0)^k + \sum_{k=0}^{n} b_k (x - x_0)^k \sum_{\rho=0}^{n} (k + \rho) Y_{k + \rho} (x - x_0)^k + \sum_{k=0}^{n} c_k \sum_{\rho=0}^{n} Y_{k + \rho} (x - x_0)^k = 0$$  \hspace{1cm} (49)
The coefficient of every term of the series have to be zero, so there is a series of equations below:

\[
\begin{cases}
  a_0\rho \cdot (\rho - 1) + b_0\rho + c_0 = 0 \\
a_0Y_1f(1) + a_1Y_0f(0) + b_0g(0)Y_1 + b_1g(0)Y_0 + c_0Y_1 + c_1Y_0 = 0 \\
  \ldots \\
  \sum_{l=0}^{k} a_lY_{k-l}f(k - l) + \sum_{l=0}^{k} b_lY_{k-l}g(k - l) + \sum_{l=0}^{k} c_lY_{k-l} = 0, k \leq n \quad (50) \\
  \sum_{l=0}^{n} a_lY_{k-l}f(k - l) + \sum_{l=0}^{n} b_lY_{k-l}g(k - l) + \sum_{l=0}^{n} c_lY_{k-l} = 0, k > n \\
  \ldots 
\end{cases}
\]

In the equations above, we define

\[
\begin{align*}
f(i) &= (\rho + i)(\rho + i - 1) \\
g(i) &= \rho + i
\end{align*}
\]

Without loss of generality, set \(Y_0 = 1\), then \(Y_n\) could be solved by the equations (50). When \(k > n\), \(Y_k\) is only related to \(Y_{k-1}, Y_{k-2}, \ldots, Y_{k-n}\), so the function of Mathematica RSolve could be used. With \(Y_1, Y_2, Y_3, \ldots, Y_n\) added as initial conditions, RSolve could give us fully solution of \(y_1(x)\).

### 3.1.2 The solution for the second power series \(y_2(x)\)

The other solution is discussed by Shuai Shao in detail. Using Frobenius method, it will provide the solution of \(y_2(x)\). The main difference between \(y_1(x)\) and \(y_2(x)\) is the term \(\frac{\partial^n}{\partial \rho^n}\). To reach the goal, I have to keep \(c_n\) as the functions of \(n\) and \(\rho\), so that I could obtain the derivative of \(c_n\) with respect to the variable \(\rho\). Finally, use the formula above to get the other solution which is linearly independent to the first one.

### 3.2 Algorithm and realization using Mathematica

Based on the equations above, we put forward an algorithm which is used to solve the problem:

1. Use tables to store the coefficients of \(A(x), B(x), C(x)\). The polynomials \(A(x), B(x), C(x)\) is given in advance.
2. Define the function \(f(x)\) and \(g(x)\)
3. Set \(Y_0 = 1\), use (50) to obtain all \(Y_k\) where \(k < n\).
4. Use RSolve and initial conditions which has been obtained before to get the analytical solutions.
5. Solve \(Y_k\) as a function of \(k\) and \(\rho\), find the partial derivative of \(Y_k\) with respect to \(\rho\)
6. Add the other terms to get the final solution.

### 3.2.1 Part of source code

The main part of source code is below:

\[
\begin{align*}
(P1(x_) = (x - 1)^2; & \quad (Q1(x_) = x - 1; \quad (R1(x_) = (x - 1)^2 - 1; \quad (*\text{set initial polynomials*}) \\
p = \text{CoefficientList}[(P(x + x0), x)]; \\
q = \text{CoefficientList}[(Q(x + x0), x)]; \\
r = \text{CoefficientList}[(R(x + x0), x)]; \quad (*\text{save coefficients as tables*}) \\
\text{AppendTo}[S, \text{Solve}][f(\rho)a(1) + \rho b(1) + c(1) = 0, \rho]] \\
\rho1 = \rho. S[[1]][1]; \quad (\rho2 = \rho. S[[1]][2]; \quad (*\text{solve } \rho \text{ in (47) or (48)*})
\end{align*}
\]
For \( k = 1, k < n + 1, k = k + 1, \)

```plaintext
AppendTo[S1, 
Solve[Sum[f[i + \rho 1]Y1[[i + 1]]a[[-i + k + 1]] + 
Sum[g[i + \rho 1]Y1[[i + 1]]b[[-i + k + 1]] + 
Sum[Y1[[i + 1]]c[[-i + k + 1]] + 
\rho f(k)a[1] + \rho g(k)b[1] + \rho c[1] = 0, \rho], 
If[S1[[k]] = {}, Y1[[k + 1]] = 0, If[S1[[k]] = {{}}, Y1[[k + 1]] = 0, Y1[[k + 1]] = \rho \cdot S1[[k]][[1]]]]; 
(Solve first n coefficients in Y_k)
sol1 = RSolve[Sum[re1[(-i + 1)]f[(-i + \rho 1 + t)] + 
Sum[re1[t - i]b[[-i + 1]]g[(-i + \rho 1 + t)] + 
Sum[re1[t - i]c[[-i + 1]]] = 0, g1, re1[t], t]; 
(*Use initial conditions to obtain analytical results *)
h2(t_: \!\text{\textstyle \frac{\partial h(t, \rho)}{\partial \rho}}, \rho \rightarrow \rho 2 
(*Get \!\text{\textstyle \frac{\partial Y_k}{\partial \rho}*)
```

### 3.2.2 Examples

To test whether the program meet our demands, I use some differential equations which are familiar to us to test it. (Some of the second solution is too complicate to print it in this issue, so I skip it to show the main idea)

**Bessel equations** Enter \( P1(x) = x^2, Q1(x) = x, R1(x) = x^2, \) which means we want to solve Bessel equations when \( \nu = 0 \) to get \( J_0(x) \):

\[
x^2 y'' + xy' + \left( x^2 - \nu^2 \right) y = 0
\]

The results are as follows:

\[
i^2 2^{-t-1} (-1)^t + 1) x^t \frac{1}{(t + 1)^2} \left( \frac{i^2 2^{-t-1} (-1)^t + 1}{(Pochhammer(2, t-1)]^2} - i^2 2^{-t-1} (-1)^t + 1) \left( (Pochhammer(2, t-1)]^2 \right) x^{-1+t}
\]

The second line In my program it will output the result polynomials in the first ten terms:

\[
\begin{align*}
\{0\}, \{0\}, \left\{ -\frac{x}{3} \right\}, \{0\}, \left\{ \frac{x^3}{24} \right\}, \{0\}, \left\{ \frac{x^5}{576} \right\}, \{0\}, \left\{ \frac{x^7}{27648} \right\}, \{0\}, \left\{ -\frac{x^9}{2211840} \right\}, \\
0, 0, 0, -\frac{1}{128}, 0, \frac{5}{13824}, 0, -\frac{13}{1769472}, 0, \frac{77}{884736000}
\end{align*}
\]
As initial conditions, the function RSolve could not identify that the result of step 3 in my algorithm, it shows that the second solution matches Neumann function $N_0(x)$. Another ODE in common use is called Laguerre equation:

Theorem 2.1

Finite series

The ODE following have a solution of finite series:

$$y'' - 0.5x y' + y = 0$$

(53)

$y = x^2 - 2$ is one of the solutions.

The input is $P1(x) = 1, Q1(x) = -0.5x, R1(x) = 1$

The results are as follows:

$$\text{re}1(t) \rightarrow \frac{1.772.223.1(1.0.1 .0 .0.5 t) - 1.01 .0.5 t - 1.0.0.5 t}{\Gamma(0.5t + 1)\Gamma(0.5(t + 1))}$$

I skip the second solution because its complexity.

Table first ten terms:

1, $-x(1.38187 * 10^{-16} - 2.76373 * 10^{-16}C[1]), -0.5x^2 - 0.166667x^3(-6.99033 * 10^{-17} + 1.38187 * 10^{-16}C[1], 0) - 0.00833333x^5(-3.45466 * 10^{-17} + 6.99033 * 10^{-17}C[1]), 0 - 0.000198413x^7(-5.182 * 10^{-17} + 1.0364 * 10^{-16}C[1], 0) - 2.75573 * 10^{-6}x^9(-1.2955 * 10^{-16} + 2.591 * 10^{-16}C[1], 0)$

(*Write in forms which are easier to read*)

At the first glance the complicate number values are obviously not the solution we want. But if we check every term we can find that there is a unknown constant $C[1]$. In fact, when I output the temporary result of step 3 in my algorithm, it shows that $Y_0 = 1, Y_1 = 0, Y_2 = -0.5$. But when I use these values as initial conditions, the function RSolve could not identify that $C[1] = \frac{1}{2}$. And if $C[1] = \frac{1}{2}$, it is obvious that all terms including $C[1]$ are zeros. Thus, Mathematica could not provide accurate solutions because $Y_1 = 1.38187 * 10^{-16} - 2.76373 * 10^{-16}C[1]$ could not determined by Mathematica. I guess that the calculation to solve $C[1]$ may lead to considerable errors so Mathematica decide not to keep this constant.

Laguerre equation

Another ODE in common use is called Laguerre equation:

$$xy'' + (1 - x)y' + \lambda y = 0$$

(54)

Set $\lambda = 0.5$, namely $P1(x) = x, Q1(x) = 1 - x, R1(x) = 0.5$, the results are as follows:

$$\text{re}1(t) \rightarrow -0.5\text{Pochhammer}(0.5, t - 1, \frac{(\text{Pochhammer}(2, t - 1))^2}{(\text{Pochhammer}(2, t - 1))^2}}$$

This equation lead to two multiple root $\rho$, so I only list one of them.

Table first ten terms:

1, $-0.5x, -0.0625x^2, -0.0104167x^3, -0.00162765x^4, -0.00027865x^5, -0.0000284831x^6, -3.19708 * 10^{-6}x^7, -3.24703 * 10^{-7}x^8, -3.00651 * 10^{-8}x^9, -2.55554 * 10^{-9}x^{10}$

That is exactly solution of Laguerre equations when $\lambda = 0.5$. **
Legendre equations

The ODEs above could be easily settled because $A(x)=1$ in (45). In Legendre equations it is $1-x^2$:

$$((1-x^2)y')' + \lambda y = 0$$

(55)

Let $\lambda = 6$, the results are as follows:

$$\text{re}1(t) \rightarrow -\frac{(t + 1) \left(c_1(-1)^t - c_1 + (-1)^{t+1} \right) \left(-1\right)^{\frac{t}{2}}}{\Gamma \left(\frac{t}{2} + 1\right)}$$

$$x^{1+t} \left(-\frac{3 \ln(t + 2) \left((-1)^t + 1\right)}{4(t - 1)(t + 1)} + \frac{3(t + 2) \left((-1)^t + 1\right)}{4(t - 1)^2(t + 1)}\right) +$$

$$x^{1+t} \left(\frac{3(t + 2) \left((-1)^t + 1\right)}{4(t - 1)(t + 1)^2} - \frac{(t + 2) \left((-1)^t + 1\right)}{4(t - 1)(t + 1)} + \frac{3 \left((-1)^t + 1\right)}{4(t - 1)(t + 1)}\right)$$

The second solution is complex. When I try to table the first ten terms, it meet the expression $\frac{1}{x}$, which means the result have singular points, so it cannot be used in this situation.

It seems that the first result provide another unknown constant $C[1]$, but in fact the analytical solution is equal to the polynomial $1 - 3x^2$.

If we set $\lambda = 2$, the RSolve function could not provide any solution because the result should be $x$. To solve this problem, if RSolve gives us an empty solution, the program will show the coefficients determined in step 3.:

$$x, 0, 0$$

Therefore the program could give us polynomial solutions in this special case.