ELSEVIER Applied Mathematics and Computation 139 (2003) 165–178

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

An algorithmic approach to exact power series solutions of second order linear homogeneous differential equations with polynomial coefficients

Onur Kıymaz, Şeref Mirasyedioğlu *

Department of Mathematics Education, Gazi University, Egitim Fakultesi, K-Blok, 06500 Teknikokullar-Ankara, Turkey

Abstract

In 1992, Koepf [J. Symb. Comp. 13 (1992) 581] introduced an algorithm for computing a Formal Power Series of a given function using generalized hypergeometric series and a recurrence equation of hypergeometric type. The main aim of this paper is to develop a new algorithm for computing exact power series solutions of second order linear differential equations with polynomial coefficients, near a point $x = x_0$, if its recurrence equation is hypergeometric type. The algorithm, which has been implemented in MAPLE, is based on symbolic computation.

Keywords: Power series solutions; Generalized hypergeometric series; Symbolic computation

1. Introduction

In 1812, Gauss presented to the Royal Society of Sciences at Göttingen his famous paper [4] in which he considered the infinite series

$$1 + \frac{ab}{1c}z + \frac{a(a+1)b(b+1)}{1 \times 2c(c+1)}z^2 + \cdots$$

* Corresponding author.
E-mail addresses: okiymaz@gazi.edu.tr (O. Kıymaz), seref@gef.gazi.edu.tr (Ş. Mirasyedioğlu).

0096-3003/02/$ - see front matter
PII: S0096-3003(02)00208-4

as a function of a, b, c, z where it is assumed that $c \neq 0, -1, -2, \ldots$, so that no zero factors appear in the denominator of the terms of the series. He showed that the series converges absolutely for $|z| < 1$ and for $|z| = 1$ when $\mathrm{Re}(c - a - b) > 0$, gave its recurrence equation, and derived his famous formula for the sum of the series when $|z| = 1$ and $\mathrm{Re}(c - a - b) > 0$.

Gauss series is also called the (ordinary) hypergeometric series. A generalization of Gauss series is the (generalized) hypergeometric series with p numerator parameters and q denominator parameters.

Except Gauss, Euler (1748), Heine (1846, 1847, 1878), Vendermonde (1972), Takacs (1973) and Askey (1975) had derived several results for hypergeometric series.

In 1992, Koepf [6–9] used generalized hypergeometric series and the definition of a recurrence equation of hypergeometric type for calculating the power series of a given function. Koepf showed that each Formal Laurent Series of hypergeometric type satisfies a linear differential equations with polynomial coefficients [6]. In this paper, we developed a new algorithm for computing the exact power series solutions of the second order linear differential equation with polynomial coefficients by using some transformations and the coefficient formula in [6]. Without loss of generality we assume that, all solutions will be computed at the point $x_0 = 0$. For further background, see [1,2,10].

2. Basic definitions

In this section, we introduce relevant material, which will be used later.

Definition 2.1. A (generalized) hypergeometric function has a series representation $\sum_{n=0}^{\infty} c_n$ with c_{n+1}/c_n a rational function of n. The ratio c_{n+1}/c_n can be factored and it is usually written as

$$\frac{c_{n+1}}{c_n} = \frac{(n + a_1)(n + a_2)\ldots(n + a_p)x}{(n + b_1)(n + b_2)\ldots(n + b_q)(n + 1)}. \tag{2.1}$$

Then if $c_0 = 1$, the Eq. (2.1) can be solved for c_n as

$$c_n = \frac{(a_1)_n (a_2)_n \ldots (a_p)_n x^n}{(b_1)_n (b_2)_n \ldots (b_p)_n n!}$$

where $(a)_n$ denotes the Pochammer symbol (or shifted factorial)

$$(a)_n = \begin{cases} 1 & n = 0 \\ a(a+1)\ldots(a+n-1) & n \in IN \end{cases}$$

and

$$pFq\left(\begin{matrix} a_1 & a_2 & \cdots & a_p \\ b_1 & b_2 & \cdots & b_q \end{matrix}, x\right) = \sum_{n=0}^{\infty} \frac{(a_1)_n (a_2)_n \ldots (a_p)_n}{(b_1)_n (b_2)_n \ldots (b_q)_n n!} x^n \tag{2.2}$$

is the usual notation [3].

Definition 2.2. Let $F = \sum_{n=n_0}^{\infty} a_n x^n \ (a_{n_0} \neq 0)$ be a formal Laurent series for some $n_0 \in \boldsymbol{Z}$.

Then F is called to be hypergeometric type if it has a positive radius of convergence, and if its coefficients satisfy a recurrence equation of form

$$\begin{aligned} a_{n+m} &= R(n) a_n \quad n \geqslant n_0 \\ a_n &= A_n \quad n = n_0, n_0 + 1, \ldots, n_0 + m - 1 \end{aligned} \tag{2.3}$$

for some $m \in \boldsymbol{IN}$, $A_n \in \boldsymbol{C}$ $(n = n_0 + 1, \ldots, n_0 + m - 1)$, $A_{n_0} \in \boldsymbol{C} \setminus \{0\}$, and some rational function R. The number m is then called the symmetry number of F. A recurrence equation of type (2.3) is also called to be of hypergeometric type [6].

Note that a function of the form $f(x^m)$ and so as the formal power series of the form $F(x^m)$ is called m-fold symmetric. It is clear that, each formal Laurent series with symmetry number m can be represented as the sum of m shifted m-fold symmetric functions [6].

3. Theoretical background

We consider the differential equation of the form

$$P(x) \frac{\mathrm{d}^2 y}{\mathrm{d}x^2} + Q(x) \frac{\mathrm{d}y}{\mathrm{d}x} + R(x) y = 0 \tag{3.1}$$

where $P(x)$, $Q(x)$ and $R(x)$ are polynomials which has degrees m, p and q respectively. The power series solutions of differential Eq. (3.1) is investigated and explained extremely clear in [5,11].

Let the point $x_0 = 0$ be a regular singular point of differential Eq. (3.1). Let r_1 and r_2 (where $\mathrm{Re}(r_1) \geqslant \mathrm{Re}(r_2)$) be the roots of the indicial equation associated with x_0 and let $y(r,x)$ be the general solution of differential Eq. (3.1). If the recurrence equation of differential Eq. (3.1) is hypergeometric type then our algorithm computes the second linearly independent solution of the differential equation by using the following conditions.

(a) If $r_1 - r_2 \neq 0$ or N then $y_1(x) = y(r,x)|_{r=r_1}$ and $y_2(x) = y(r,x)|_{r=r_2}$
(b) If $r_1 = r_2$ then $y_1(x) = y(r,x)|_{r=r_1}$ and $y_2(x) = \frac{\partial}{\partial r} y(r,x)\big|_{r=r_1}$
(c) If $r_1 - r_2 = N$ then $y_1(x) = y(r,x)|_{r=r_1}$ and $y_2(x) = \frac{\partial}{\partial r}\left[y(r,x)(r - r_2)\right]\big|_{r=r_2}$

Lemma 3.1. *Let $x_0 = 0$ be an ordinary point of an nth order linear differential equation with polynomial coefficients. The recurrence equation of the differential equation can be computed applying the transformation*

$$x^m y^{(k)} \to (n+1-m)_k a_{n+k-m} \tag{3.2}$$

to the all terms of the differential equation [6].

Lemma 3.2. *Let $x_0 = 0$ be a regular singular point of an nth order linear differential equation with polynomial coefficients. The recurrence equation of the differential equation, can be computed applying the transformation*

$$x^m y^{(k)} \to (n+r+1-m)_k a_{n+k-m} \tag{3.3}$$

to the all terms of the differential equation.

Proof. Since the point $x_0 = 0$ is a regular singular point of the differential equation, we seek a solution of form $y = \sum_{n=0}^{\infty} c_n x^{n+r}$ where $c_0 \neq 0$. Differentiating it k times, multiplying with x^m and rewriting the power of x as $n+r$, we obtain

$$x^m y^{(k)} = \sum_{n=m-k}^{\infty} (n+r+k-m)\dots(n+r+1-m) a_{n+k-m} x^{n+r}$$

Then the inner term in the sum is $(n+r+1-m)_k a_{n+k-m}$. □

Note that the lowest power of x is assumed to be $n+r$ in the recurrence equation obtained in Lemma 3.2. In the method of Frobenius, the recurrence equation is rearranged such that power of x could be the lowest.

Lemma 3.3. *Let $x_0 = 0$ be a regular singular point of differential Eq.* (3.1). *The recurrence equation, with the lowest power of x, can be computed applying the transformation* (3.3) *and $n \to n+m-2$ respectively.*

Proof. Since $x_0 = 0$ is a regular singular point of the differential equation, the following proposition is valid.

$$\{(m > q) \wedge [(m \leqslant p+1) \wedge (m \leqslant q+2)]\} \vee \{(m > p) \wedge [(m \leqslant p+1) \wedge (m \leqslant q+2)]\}$$

Let $m > p$. Then $m = p+1$. So the powers of x are $n+r-2+(p+1)$, $n+r-1+p$ and $n+r+q$ respectively. Then the lowest power of x is $n+r-1+p = n+r+m-2$.

Let $m > p$. Then $m = q+1$ or $m = q+2$. First, let $m = q+1$. Then $q \leqslant p$. So the powers of x are $n+r-1+q$, $n+r-1+p$ and $n+r+q$ respectively. Then the lowest power of x is $n+r-1+q = n+r+m-2$.

Let $m = q + 2$. Then $(q + 1) \leqslant p$. So the powers of x are $n + r + q$, $n + r - 1 + p$ and $n + r + q$ respectively. Then the lowest power of x is $n + r + q = n + r + m - 2$. □

Theorem 3.1. *If the corresponding recurrence equation of a second order linear differential equation with polynomial coefficient is hypergeometric type, then the exact power series solutions of differential equation can be computed.*

Proof. If the recurrence equation is hypergeometric type then it is of the form (2.3). So the symmetry number is equal to m. If we apply the transformation $n \to mn$ to recurrence Eq. (2.3) we get

$$a_{m(n+1)} = R(mn)a_{mn}. \tag{3.4}$$

Applying a second transformation $a_{mn} \to c_n$ to Eq. (3.4), we get

$$c_{n+1} = R(mn)c_n. \tag{3.5}$$

Then we get Eq. (2.1) by multiplying Eq. (3.5) the variable x. Making similar calculations with the transformations $n \to mn + 1, \ldots, mn + (m - 1)$, we get m equations of form (2.1). For each equation, we can find the coefficients of corresponding power series. So, we can represent the solutions as the sum of m shifted m-fold symmetric functions. □

4. The algorithm and the maple implementation

Here we present two sub-algorithms that compute the exact power series solutions of a given second order linear differential equation with polynomial coefficients at the point $x_0 = 0$ if its recurrence equation is hypergeometric type. The first algorithm work for ordinary point and the second algorithm works for regular singular point.

Algorithm 4.1

(1) Apply transformation (3.2) to the all terms at the left hand side (lhs) of the input. Write the results in the equation. Then construct the recurrence equation.
(2) In the right hand side (rhs) of the recurrence equation, apply a substitution which makes the index of coefficient c, n. Determine the symmetry number m. Represent the solutions as the sum of m shifted m-fold symmetric functions.
(3) From these, choose the series, which are of form (2.1). Find the explicit formula using some calculations and hypergeometric coefficient formula (2.2). Then construct the results.

Theorem 4.1. *Let $x = x_0$ be the ordinary point of the second order linear homogeneous differential equation with polynomial coefficients. Then Algorithm* 4.1 *computes the exact power series solutions correctly if the recurrence equation of the differential equation is hypergeometric type.*

Proof. The proof of step (1) is given by Koepf in [6]. In step (2), we transformed the recurrence equation to form (2.3) by a substitution that makes the index of coefficient c, n. Then applying the transformations described in proof of Theorem 3.1 we get m shifted m-fold symmetric functions. After choosing the series which are of form (2.1) in step (3), the proof of the calculations for finding the explicit formula is also given in [6]. □

Algorithm 4.2

(1) Apply transformation (3.3) to the all terms at the (lhs) of the input. Write the results in the equation. Make the transformation described in Lemma 3.3. Then construct the recurrence equation.
(2) Apply substitution $n = 0$ to the term which has the coefficient of maximum index in the recurrence equation. Equate it to zero. Find the roots r_1, r_2 of the indicial equation. Let $r_1 > r_2$.
(3) Write the root r_1 into the recurrence equation. In the rhs of the recurrence equation, apply a substitution which makes the index of coefficient c, n. Determine the symmetry number m. Apply the transformations $n \to mk$ and $a_{mk} \to c_k$. Find the explicit formula for the coefficients using some calculations and hypergeometric coefficient formula (2.2). Then construct the result.
(4) Find $t = r_1 - r_2$.
   a. If $t \neq 0$ or positive integer then apply steps (3) for the root r_2.
   b. If $t = 0$ then differentiate the recurrence equation with respect to r. Substitute $r = r_1$. Then construct the result.
   c. If $t = N$ (positive integer) then multiply $y(r,x)$ with $(r - r_2)$. Differentiate $y(r,x)(r - r_2)$ with respect to r. Substitute $r = r_2$. Then construct the result.

Theorem 4.2. *Let $x = x_0$ be the regular singular point of the second order linear homogeneous differential equation with polynomial coefficients. Then Algorithm* 4.2 *computes the exact power series solutions correctly if the recurrence equation of the differential equation is hypergeometric type.*

Proof. The proof of step (1) is given in Lemma 3.2 and Lemma 3.3. Step (2) computes the roots of the indicial equation. The proof of step (3) is similar with the proof of step (2) in Algorithm 4.1. Step (4) computes the second linear independent solution using the method of Frobenius. □

Example 4.1. Consider the equation $y'' + xy' + y = 0$. Applying transformation (3.2) to all terms of the given differential equation we get the recurrence equation as

$$(n+2)a_{n+2} + a_n = 0. \tag{4.1}$$

The recurrence equation is hypergeometric type and the symmetry number is 2. Applying the transformations $n \to 2k$ and $a_{2k} \to c_k$ to Eq. (4.1), we get

$$c_{k+1} = -c_k/2(k+1). \tag{4.2}$$

This is of form (2.1) and it is the first solution of the differential equation. So with setting the coefficient $c_0 = 1$ in Eq. (4.2), we obtain the result as $y_1(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{2^n n!}$. Similarly, applying the transformations $n \to 2k+1$ and $a_{2k+1} \to c_k$ to (4.1), we get

$$c_{k+1} = c_k/2(k+3/2). \tag{4.3}$$

This is of form (2.1). So with setting coefficient $c_1 = 1$ in Eq. (4.3), we get second linearly independent solution of the differential equation as $y_2(x) = \sum_{n=0}^{\infty} \frac{(-1)^n 2^n n! x^{2n+1}}{(2n+1)!}$.

Example 4.2. Consider the equation $4xy'' + 2y' + y = 0$. Applying transformation (3.3) to all terms of the given differential equation and the transformation described in Lemma 3.3 we get the recurrence equation

$$[4(n+r)^2 - 2(n+r)]a_n(r) + a_{n-1} = 0. \tag{4.4}$$

Substituting $n = 0$ on the lhs of the coefficient which has the maximum index in Eq. (4.4) and solving it for r, we get the roots of the indicial equation as 1/2 and 0. Rewriting the root 1/2 in the recurrence equation and applying step (3) we get the solution as $y_1(x) = x^{1/2} \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{(2n+1)!}$. Because of the difference of the roots is not a positive integer or zero we apply the step (5a). So applying the steps (3) and (4) for the smaller root $r_2 = 0$, we obtain second linearly independent solution of the differential equation as $y_2(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{(2n)!}$.

Example 4.3. Consider the equation $y'' + 2y' + y = 0$. Applying transformation (3.2) we get the recurrence equation as $(n+1)(n+2)a_{n+2} + 2(n+1)a_{n+1} + a_n = 0$. The recurrence is not hypergeometric type because it is not of form (2.3). So the algorithm cannot compute the solutions for this case.

The implementation of the algorithm via MapleV is given below. The procedure *ssolve* computes the exact power series solutions of the second order linear homogeneous differential equation with polynomial coefficients at the point $x_0 = 0$ if its recurrence equation is hypergeometric type.

```
###
# This symbolic Maple program was developed to find the exact power series
# solutions of a given differential equation (DE).
# Conditions:
# 1. DE must be second order linear homogeneous with polynomial coeffi-
  cients.
# 2. The corresponding recurrence equation (RE) must be hypergeometric
  type.
# 3. The power series solutions will be computed at the point x0 = 0.
# Input: A differential equation.
# Output: Exact power series solution of DE at the point x0 = 0 if possible.
###

  ### MAIN PROGRAM ###
  ssolve := proc(de)
    typecheck(lhs(de));
    if DEtype = 1 then findRE(lhs(de)); sop(RE, sn); RETURN (y(x) =
    pss_op)
    elif DEtype = 2 then findRE(lhs(de)); ssp(RE, sn); RETURN (y(x) =
    pss_sp) fi
  end

  ### CHECK THE TYPE OF THE GIVEN DE ###
  typecheck := proc(f) local a, i, term, index, px, qx; global DEtype;
    with(PDEtools, difforder);
    if difforder(f) <> 2 then ERROR('Not A second degree ODE!')
    else for i to nops(f) do
     term[i] := op(i, f); index := difforder(term[i]);
     if index <> 0 then a[index] := lcoeff(term[i], diff(y(x), x $ index))
     else a[index] := lcoeff(term[i], y(x)) fi od;
      px := simplify(a[1]/a[2]); qx := simplify(a[0]/a[2]);
      if traperror(subs(x = 0, px)) <> 'division by zero'
        and traperror(subs(x = 0, qx)) <> 'division by zero' then DEtype := 1
      elif traperror(subs(x = 0, x*px)) <> 'division by zero'
        and traperror(subs(x = 0, x^2*qx)) <> 'division by zero' then
    DEtype := 2
      else ERROR('x0 is an irregular singular point!') fi fi
  end

  ### FIND THE CORRESPONDING RE OF THE GIVEN DE ###
  findRE := proc(f) local i, g, m, m1, k, term, ind1, ind2; global RE, sn, rts;
    RE := 0; g := expand(f);
```

```
  for i to nops(g) do
    term[i] := op(i, g); k := difforder(term[i]);
    if type(term[i], '*') then
       if k <> 0 then m := degree(subs(diff(y(x), x $ k) = 1, term[i]), x)
       else m := degree(subs(y(x) = 1, term[i]), x) fi else m := 0 fi;
    if i = 1 then m1 := m fi;
    term[i] := lcoeff(term[i])*Poch(n + r + 1 - m, k)* a[n + k - m];
    RE := RE + term[i] od;
  for i to nops(f) do RE := collect(RE, a[n + i - 1]) od;
  if DEtype = 1 then RE := subs(r = 0, RE) else RE := subs(n = n +
  m1 - 2, RE) fi;
  rts := solve(subs(n = 0, op(1, op(1, RE))), r);
  if nops(RE) <> 2 then ERROR('RE is not hypergeometric type!') fi;
  if type(op(1, RE), '*') then ind1 := op(1, op(nops(op(1, RE)), op(1, RE)))
  else ind1 := op(1, op(1, RE)) fi;
  if type(op(2, RE), '*') then ind2 := op(1, op(nops(op(2, RE)), op(2, RE)))
  else ind2 := op(1, op(2, RE)) fi;
  if ind2 < ind1 then sn := ind1 - ind2 else sn := ind2 - ind1 fi;
  if ind2 < ind1 then
    RE := solve(RE = 0, a[ind1]);
    RE := subs({n = n + solve(ind2 = 0, n), a[ind2] = 1},RE);
    RE := factor(simplify(numer(RE)))/factor(simplify(denom(RE)))
  else
    RE := solve(RE = 0, a[ind2]);
    RE := subs({n = n + solve(ind1 = 0, n), a[ind1] = 1},RE);
    RE := factor(simplify(numer(RE)))/factor(simplify(denom(RE))) fi
end

### COMPUTE THE POCHAMMER OF A GIVEN EXPRESSION
###
Poch := proc(exp, k) local i, num, den; global P;
  num := numer(exp); den := denom(exp); P := 1;
  if type(k, integer) then
    if k <> 0 then
       if den = 2 then P := (num + 2 * k - 1)! * (1/2*num-1/2)!*
       ((num - 1)! * (1/2 * num - 1/2 + k)! * 2^(2 * k)
       else for i from 0 to k - 1 do P := P * (exp +i) od fi else P := 1 fi
  elif has(num, r) then P := Product(num + (j - 1)*den, j = 1..k)/den^k
  else
    if den = 1 then P := (exp +k - 1)!/(exp -1)!
    elif den = 2 and 0 < num then
       P := (num + 2 * k - 1)! * (1/2 * num - 1/2)!/((num - 1)!*
```

```
        (1/2 * num -1/2 + k)! * 2^(2 * k))
    else P := ([num, num + den, num + 2 * den, '..',num + den * (k - 1)])/
    den^k fi fi
end

### THE TRANSFORMATIONS ###
maketr := proc(req) local i, deg, part, new_part; global sol;
  sol := 1;
  if DEtype = 1 and has(req, n + 1) or DEtype = 2 then
    for i to nops(req) do
      part[i] := op(i, req);
    if numer(part[i]) <> 1 and denom(part[i]) = 1 then deg := degree
    (part[i], n)
    else deg := -degree(denom(part[i]), n) fi;
    if deg = -1 then part[i] := denom(part[i]) fi;
    if deg < -1 then part[i] := op(1, denom(part[i])) fi;
    if 1 < deg then part[i] := op(1, part[i]) fi;
    if is(part[i], real) then new_part[i] := part[i]^n
    elif part[i] = n then new_part[i] := 0
    else new_part[i] := (lcoeff(part[i])^n)^deg*Poch(-solve(part[i] = 0, n),
    n)^deg fi;
    sol := sol*new_part[i] od else sol := 0 fi
end

### COMPUTE THE SOLUTION AT ORDINARY POINT ###
sop := proc(req, snum) local i, new_req; global pss_op;
  pss_op := 0;
  for i from 0 to snum - 1 do
    new_req := simplify(subs(n = snumn + i, req)); maketr(new_req);
    if sol <> 0 then pss_op := pss_op + Sum(sol * x^(snum * n + i),
    n = 0... infinity)
    else pss_op := pss_op fi od
end

### COMPUTE THE SOLUTION AT REGULAR SINGULAR POINT
###
ssp := proc(req, snum)local i, new_req, b_root, s_root, sol1, sol2, prd, spn,
n_s, n_p, t_num, ns; global pss_sp;
  pss_sp := 0; spn := 0;
  if rts[1] - rts[2] < 0 then b_root := rts[2]; s_root := rts[1]
  else b_root := rts[1]; s_root := rts[2] fi;
  if not type(b_root - s_root, integer) then
    for i to 2 do
```

```
    new_req := simplify(subs({n = snum*n,   r = rts[i]},   req));   maketr
    (new_req);
    pss_sp := pss_sp + x^rts[i]*Sum(sol*x^(snum*n), n = 0.. infinity) od
elif b_root = s_root then
 new_req := simplify(subs(n = snum*n, req)); maketr(new_req);
 pss_sp := pss_sp + subs(r = b_root,x^r*(1 + Sum(sol*x^(snum*n),
 n = 1.. infinity)));
 sol1 := diff(x^r*(1 + Sum(sol*x^(snum*n), n = 1.. infinity)),r);
 pss_sp := pss_sp + subs(r = b_root, sol1)
else
 if 1 < lcoeff(denom(req), r) then
    new_req := simplify(subs({n = snum*n,   r = b_root},   req));   maketr
    (new_req);
    pss_sp := pss_sp + subs(r = b_root,x^r*Sum(sol*x^(snum*n),   n = 0..
    infinity));
    new_req := simplify(subs({n = snum*n,   r = s_root},   req));   maketr
    (new_req);
    pss_sp := pss_sp + subs(r = s_root, x^r*Sum(sol*x^(snum*n), n = 0..
    infinity))
 else
    new_req := simplify(subs(n = snum*n, req)); maketr(new_req);
    pss_sp := pss_sp + subs(r = b_root,     x^r*(1 + Sum(sol*x^(snum*n),
    n = 1.. infinity)));
    if not (select(has, denom(sol), Product) = Product()) then
       for i to nops(select(has, denom(sol), Product)) do
       prd := op(i, select(has, denom(sol), Product));
       if has(op(1, prd), Product) then prd := op(1, prd) fi;
       if not has(prd, Product) then prd := select(has, denom(sol), Prod-
    uct) fi;
       if 0 < solve(op(1, prd) -r + s_root, j) then
         spn := solve(op(1, prd) -r + s_root, j); t_num := i fi od;
    if not has(op(1, denom(sol)), Product) thent_num := t_num - 1 fi;
 else
    prd := denom(sol);
    if 0 < solve(op(1, prd) -r + s_root, j) then spn := solve(op(1, prd)
    -r + s_root, j) fi;
    t_num := 1 fi;
 if spn = 1 then
    if t_num = 1 then prd := denom(sol) else prd := op(t_num, de-
    nom(sol)) fi;
    n_p := subsop(2 = subs(1 = 2, op(2, prd)), prd);
    n_s := subs(prd = n_p, sol);
    ns := subs(Product = product,      sum(sol*x^(snum*n)*(r - s_root),
```

```
    n = 1.. 1)) + Sum(n_s*x^(snum*n), n = 2.. infinity); ns := eval(ns) fi;
  if 1 < spn then
    if t_num = 1 then prd := denom(sol) else prd := op(t_num, denom
    (sol)) fi;
    n_s := subs(prd = Product(op(1, prd), j = 1.. spn - 1)* Prod-
    uct(op(1, prd), j = spn + 1.. n),sol);
    ns := subs(Product = product, sum(sol*x^(snum*n)*(r - s_root),
    n = spn.. spn)) + Sum(n_s*x^(snum*n), n = spn + 1.. infinity);
    ns := eval(Sum(sol*x^(snum*n)*(r - s_root), n = 1 .. spn - 1) + ns) fi;
  sol2 := diff(x^r*(r - s_root + ns), r); pss_sp := pss_sp + subs(r = s_root,
  sol2) fi fi
end
```

5. Examples

In this section some examples are presented to demonstrate the applicability of the method described in Section 4 via Maple V.

Example 5.1. Consider the differential equation $y'' + y = 0$. We observe that $x_0 = 0$ is an ordinary point of the differential equation. Then the computer output for the given differential equation is below.

```
> ssolve(diff(y(x), x$2) + y(x) = 0);
```

$$y(x) = \sum_{n=0}^{\infty} \left(\frac{\left(\frac{-1}{2}\right)^n 2^{2n} x^{2n}}{2^n (2n)!} \right) + \sum_{n=0}^{\infty} \left(2 \frac{\left(\frac{-1}{2}\right)^n (n+1)! 2^{2n} x^{2n+1}}{2^n (2n+2)! n!} \right)$$

Example 5.2. Consider the differential equation $x^2 y'' + xy' + (x^2 - 1/4)y = 0$. We observe that $x_0 = 0$ is a regular singular point of the differential equation and the roots of the indicial equation are 1/2 and $-1/2$. Then the computer output for the given differential equation is below.

```
> ssolve(x^2 * diff(y(x), x$2) + x * diff(y(x), x) + (x^2 - 1/4) * y(x) = 0);
```

$$y(x) = \sqrt{x} \sum_{n=0}^{\infty} \left(2 \frac{\left(\frac{-1}{2}\right)^n (n+1)! 2^{2n} x^{2n}}{2^n (2n+2)! n!} \right) + \frac{1}{\sqrt{x}} \sum_{n=0}^{\infty} \left(\frac{\left(\frac{-1}{2}\right)^n 2^{2n} x^{2n}}{2^n (2n)!} \right)$$

Example 5.3. Consider the differential equation $x^2 y'' + xy' + x^2 y = 0$. We observe that $x_0 = 0$ is a regular singular point of the differential equation and the roots of the indicial equation are 0. Then the computer output for the given differential equation is below.

```
> ssolve(x^2 * diff(y(x), x$2) + x * diff(y(x), x) + x^2 * y(x) = 0);
```

$$y(x) = 1 + \left(\sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{\left(\prod_{j=1}^{n} (2j) \right)^2} \right) + \ln(x) \left(1 + \sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{\left(\prod_{j=1}^{n} (2j) \right)^2} \right) + \sum_{n=1}^{\infty} \left(-2 \frac{(-1)^n x^{2n} \left(\sum_{k=1}^{n} \left(\prod_{j=1}^{k-1} 2j \right) \left(\prod_{j=k+1}^{n} 2j \right) \right)}{\left(\prod_{j=1}^{n} (2j) \right)^3} \right)$$

Example 5.4. Consider the differential equation $x^2y'' + xy' + (x^2 - 1)y = 0$. We observe that $x_0 = 0$ is a regular singular point of the differential equation and the roots of the indicial equation are 1 and -1. Then the computer output for the given differential equation is below.

> ssolve(x^2 * diff(y(x), x$2) + x * diff(y(x), x) + (x^2 - 1) * y(x) = 0);

$$y(x) = x \left(1 + \sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{\prod_{j=1}^{n} (2j) \prod_{j=1}^{n} (2 + 2j)} \right) + \frac{\ln(x)}{x} \left(-\frac{1}{2} x^2 + \sum_{n=2}^{\infty} \frac{(-1)^n x^{2n}}{\prod_{j=1}^{n} (2j) \prod_{j=2}^{n} (2j - 2)} \right) + \frac{1}{x} \left(1 + \frac{x^2}{4} + \sum_{n=2}^{\infty} \left(-\frac{(-1)^n x^{2n} \left(\sum_{k=1}^{n} \left(\prod_{j=1}^{k-1} 2j \right) \left(\prod_{j=k+1}^{n} 2j \right) \right)}{\left(\prod_{j=1}^{n} (2j) \right)^2 \prod_{j=2}^{n} (2j - 2)} - \frac{(-1)^n x^{2n} \left(\sum_{k=2}^{n} \left(\prod_{j=2}^{k-1} (2j - 2) \right) \left(\prod_{j=k+1}^{n} (2j - 2) \right) \right)}{\prod_{j=1}^{n} (2j) \left(\prod_{j=2}^{n} (2j - 2) \right)^2} \right) \right)$$

Example 5.5. Consider the differential equation $y'' + y' + y = 0$. Then the computer output for the given differential equation is below.

> ssolve(diff(y(x), x$2) + diff(y(x), x) + y(x) = 0);

Error, (*in findRE*) *RE is not hypergeometric type*!

Example 5.6. Consider the differential equation $y''' + 2xy' + y = 0$. Then the computer output of the given differential equation is below.

> ssolve(diff(y(x), x$3) + 2 * x * diff(y(x), x) + y(x) = 0);

Error, (*in typecheck*) *Not a second degree ODE*!

Example 5.7. Consider the differential equation $x^3y'' + (x - 1)y' + y = 0$. Then the computer output for the given differential equation is below.

$>$ ssolve(x^3 $*$ diff(y(x), x$2) + (x − 1) $*$ diff(y(x), x) + y(x) = 0);

Error, (*in typecheck*) x_0 *is an irregular singular point*!

Acknowledgements

We are especially indebted to Prof. Dr. Aydın Tiryaki at Gazi University, Turkey, for his many helpful comments and suggestions. We are also deeply grateful to Tolga Güyer for his continuous support on programming.

References

[1] E. Compoint, Differential equations and algebraic relations, J. Symb. Comp. 25 (1998) 705–725.
[2] D. Gruntz, Powerseries, A package for Infinite Power Series, Maple Share Library, 1992.
[3] G. Gasper, M. Rahman, Basic Hypergeometric Series, Cambridge University Press, Cambridge, 1990.
[4] C.F. Gauss, Disquisitiones Generales Circa ..., Commun. Soc. Req. Sci. Gött. Rec., vol. II. Reprinted from: Werke, 3 (1876) 123–162.
[5] E.L. Ince, Ordinary Differential Equations, Dover Pub. Inc., New York, 1956.
[6] W. Koepf, Power series in computer algebra, J. Symb. Comp. 13 (1992) 581–603.
[7] W. Koepf, Examples for the algorithmic calculation of formal Puiseux Laurent and power series, SIGSAM Bull. 27 (1993) 20–32.
[8] W. Koepf, Algorithmic development of power series, in: J. Calmet, J.A. Campell (Eds.), Artificial Intelligence and Symbolic Mathematical Computing, Int. Conference AISMC-1, Karlsruhe, Germany, August 1992, Proc., Lecture Notes in Computer Science 737, Springer-Verlag, Berlin, 1993, pp. 195–213.
[9] W. Koepf, Symbolic computation of formal power series with MACSYMA, in: W. Tutschke, A. Mshimba (Eds.), Proceedings of the Second Workshop on Functional-Analytic Methods in Complex Analysis and Applications to Partial Differential Equations, Trieste, Italy, January 1993, World Scientific Publishing Co.
[10] J.J. Kovacic, An algorithm for solving second order linear differential equations, J. Symb. Comp. 2 (1985) 3–43.
[11] S.L. Ross, Differential Equations, Ginn-Blaisdell, London, 1964.