

Designing an Algorithm for the Series Solution of n^{th} Order Linear ODE's with Polynomial Coefficients

Nikolaos Glinos

Department of Mathematics, Section of Applied Mathematics and Informatics, University of Ioannina, Ioannina GR-451 10, Greece

Abstract

In this paper we design an algorithm using operational calculus methods and *Mathematica* functional programming operators. The purpose of the algorithm is the computation of power series solutions of n^{th} order linear differential equations in the case where the coefficients and the right hand side are polynomials or generally functions having Taylor series expansions at the origin. The algorithm performs very well, particularly when the high order derivatives $(n-1)$, $(n-2)$, ..., in the differential equation are missing and the polynomial coefficients do not have low degree terms.

1 Introduction

The symbolic solution of differential equations is one of the most demanding problems in Computer Algebra [1]. Operational Calculus [2, 3] provides tools for solving some classes of differential equations [4]. If a closed form solution is not possible then a series solution [5, 6, 7] is a good alternative. In this work we use operational calculus methods and functional operators of *Mathematica* [8] to design and implement a series solution algorithm for differential equations of the form

$$y^{(n)} + p_1(x)y^{(n-1)} + \dots + p_{n-1}(x)y' + p_n(x)y = f(x), \quad n \geq 1 \quad (1)$$
$$y(0) = y_0, \quad y'(0) = y'_0, \dots, \quad y^{(n-1)}(0) = y_0^{(n-1)}$$

where the coefficients and right hand side are polynomials or generally functions with Taylor series expansions at the origin.

162 Innovation In Mathematics

Let C be the set of all continuous functions $f(x)$ on $[0, \infty)$ and K the set of all functions $k(x, t)$, called kernels, continuous in t, x for $0 \leq t \leq x$. The application (denoted by juxtaposition) of a kernel $k(x, t)$ to a function $f(x)$, is defined by $k(x, t)f(x) = \int_0^x k(x, t)f(t)dt$. The operator $[1+k(x, t)]: C \rightarrow C$, is defined by $[1+k(x, t)]f(x) = f(x) + k(x, t)f(x)$. We define the generalized convolution of two kernels $k_1, k_2 \in K$ as $k_1(x, t)*k_2(x, t) = \int_t^x k_1(x, \xi)k_2(\xi, t)d\xi$. It can be shown that $(k_1(x, t)*k_2(x, t))f(x) = k_1(x, t)(k_2(x, t)f(x))$ and we can use the notation $k^{*n} = k^n$, where $k^n f(x)$ denotes the function resulting from applying the kernel k iteratively n times. The following lemma supports our main theorem [2, 4], which is the mathematical basis for our work.

Lemma. For any $X \geq 0$ there exists a number Θ such that,

$$|k^{*n}(x, t)| \leq \frac{1}{(n-1)!} \Theta^n (x-t)^{n-1} \leq \frac{1}{(n-1)!} \Theta^n X^{n-1}, \quad 0 \leq t \leq x \leq X$$

Proof. Let X be any fixed value. We proceed by induction on n . For $n = 1$ the inequality becomes $|k(x, t)| \leq \Theta$, which is true for some arbitrary but fixed number Θ , since $k(x, t)$ is continuous and hence bounded on the compact set $0 \leq t \leq x \leq X$. Assuming the inequality is true for n , we will prove the inequality for $n+1$.

$$\begin{aligned} |k^{*n+1}(x, t)| &= |k(x, t)*k^{*n}(x, t)| = \left| \int_t^x k^{*n}(x, \xi)k(\xi, t)d\xi \right| \leq \\ &\leq \int_t^x |k^{*n}(x, \xi)k(\xi, t)|d\xi \leq \int_t^x \frac{1}{(n-1)!} \Theta^n (x-\xi)^{n-1} \Theta d\xi = \\ &= \frac{1}{(n-1)!} \Theta^{n+1} \int_t^x (x-\xi)^{n-1} d\xi = \frac{1}{(n-1)!} \Theta^{n+1} \frac{(x-t)^n}{n} = \frac{1}{n!} \Theta^{n+1} (x-t)^n \end{aligned}$$

The last inequality of the lemma is clearly true since $x-t \leq X$.

Theorem. The operators $[1+k(x, t)]$ form a group under the operation (denoted by juxtaposition) $[1+k_1][1+k_2] = [1+k_1+k_2+k_1*k_2]$, $k_1, k_2 \in K$. The inverse with respect to this operation is given by $[1+k(x, t)]^{-1} = [1 + \sum_{n=1}^{\infty} (-1)^n k^{*n}(x, t)]$.

Proof. It is easy to verify that the group axioms hold. The element $[1+k_1][1+k_2]$ belongs in the group since $k_1+k_2+k_1*k_2$ belongs in K . The identity operator,

$1 = [1+0(x,t)]$, is the group identity. The group operation is associative. This can be shown by using Dirichlet's formula. For every operator $[1+k(x,t)]$, the inverse $[1+k(x,t)]^{-1}$ exists, is unique and well defined. To show this we use the Weierstrass M test for uniform convergence. In every interval $0 \leq t \leq x \leq X$, by the above lemma, we have that,

$$|(-1)^n k^{*n}(x,t)| = |(-1)^n \|k^{*n}(x,t)\|| = \|k^{*n}(x,t)\| \leq \frac{1}{(n-1)!} \Theta^n X^{n-1}$$

The series $\sum_{n=1}^{\infty} \frac{1}{(n-1)!} \Theta^n X^{n-1} = \Theta \sum_{n=0}^{\infty} \frac{M^n}{n!}$, where $M = \Theta X$, has constant terms

and converges to Θe^M . Therefore the series $\sum_{n=1}^{\infty} (-1)^n k^{*n}(x,t)$ converges

uniformly to an element in K , and so the operator $[1+k(x,t)]^{-1}$ is an element of the group. Furthermore the uniqueness of the inverse operator is easily proven because if $[1+k(x,t)]^{-1}$ had two inverses $[1+k_1(x,t)]$ and $[1+k_2(x,t)]$ then $k_1(x,t) = k_2(x,t)$, since they are both uniform limits. It is easy to see that $[1+k(x,t)][1+k(x,t)]^{-1} = [1+0(x,t)] = 1$.

2 Series solution of linear ode's with polynomial coefficients

If L is the kernel $k(x,t) = 1$ then its function application is $Lf(x) = \int_0^x f(t)dt$ and

we can show that $L^{*n} = L^n = \frac{1}{(n-1)!} (x-t)^{(n-1)}$. If in equation (1) we substitute

$y, y', y'', \dots, y^{(n-1)}$ with their Taylor series expansions,

$y(x) = \sum_{i=0}^{n-1} \frac{y_0^{(i)}}{i!} x^i + \int_0^x \frac{(x-t)^{n-1}}{(n-1)!} y^{(n)}(t)dt$ and similarly for the derivatives, then

after rearranging terms the differential equation becomes,

$$y^{(n)}(x) + \int_0^x k(x,t) y^{(n)}(t)dt = g(x) \Rightarrow [1+k(x,t)] y^{(n)}(x) = g(x) \quad (2)$$

where

$$k(x,t) = p_1(x)L + p_2(x)L^2 + \dots + p_n(x)L^n \quad (3)$$

and

$$g(x) = f(x) - (p_1(x) + xp_2(x) + \frac{x^2}{2} p_3(x) + \dots + \frac{x^{n-1}}{(n-1)!} p_n(x)) y_0^{(n-1)} \quad (4)$$

$$- (p_2(x) + xp_3(x) + \dots + \frac{x^{n-2}}{(n-2)!} p_n(x)) y_0^{(n-2)} - \dots - p_n(x) y_0$$

164 Innovation In Mathematics

From equation (2), using the above theorem, we obtain $y^{(n)}(x)$ as the following series.

$$\begin{aligned} y^{(n)}(x) &= [1 + k(x, t)]^{-1} g(x) = g(x) + \sum_{n=1}^{\infty} (-1)^n k^{*n}(x, t) g(x) \\ &= g(x) - k(x, t) g(x) + k(x, t) (k(x, t) g(x)) \\ &\quad - k(x, t) (k(x, t) (k(x, t) g(x))) + \dots \end{aligned} \quad (5)$$

In the *Mathematica* functions described below we use parameters, p for the polynomial coefficients, f for the right hand side, y for the unknown function, x for the dependent variable, ics for the initial conditions and n for the equation order. If $maxorder$ is the highest power of the series solution then we need only to compute up to the $m = maxorder - n$ power, since n integration operations will be performed at the end. The following function is used for truncating polynomial terms beyond some power m .

```
In[1]:= PolyTrunc[poly_, x_, m_] := Module[{t = Collect[Expand[poly], x],  
If[m == 0, t /. x -> 0, t /. x ^ (e_ /; e > m) -> 0]}
```

The expression $g(x)$ in equation (4) can be thought as an initial approximation to the series solution and basically it involves a series of inner products of diminishing length vectors with one last inner product at the end involving the initial conditions. This computation is done by the following function.

```
In[2]:= InitApprox[p_, f_, ics_, n_, x_, m_] := Module[{r, vx, vp, i},  
vx = Table[x^(i-1)/(i-1)!, {i, n}]; vp = p;  
PolyTrunc[f -  
Table[Drop[vx, -i+1] . Drop[vp, i-1], {i, n}] . Reverse[ics], x, m]]
```

We can clearly observe in equation (5) the iterative form of the calculation of each new term of the series. It appears that the computation of the series is time consuming, since the operators $L, L^2, \dots, L^n$ of $k(x, t)$ in equation (3) involve integrations. However, in the case of polynomial coefficients the integration steps are all easy to perform and explicit integration can be avoided. The initial approximation $g(x)$ in equation (4), and generally the expression to which the kernel $k(x, t)$ is applied every time, is a polynomial of the form $g(x) = g_m x^m + g_{m-1} x^{m-1} + \dots + g_1 x + g_0$. Using the definitions given, we can see that the application of the kernel $k(x, t)$ to $g(x)$ has the following form.

$$k(x, t) g(x) = \sum_{j=1}^n p_j(x) \int_0^x \frac{(x-t)^{j-1}}{(j-1)!} g(t) dt \quad (6)$$

Using the binomial expansion of $(x-t)^{j-1}$, performing the integrations and with some manipulation of terms, we can rewrite the summation in equation (6) as follows.

$$\sum_{j=1}^n p_j(x) \left[\sum_{i=0}^m \left(\sum_{k=0}^{j-1} \frac{(-1)^k}{k!(j-k-1)!(i+k+1)} \right) g_i x^{i+j} \right] = \sum_{j=1}^n p_j(x) q_j(x) \quad (7)$$

We would like now to find a closed form expression for the inner most summation which represents part of the coefficients of the polynomial $q_j(x)$ in equation (7). Using *Mathematica*'s summation package we find:

In[3]:= <<Algebra`SymbolicSum`

In[4]:= Sum[(-1)^k / (k! (j-k-1)! (i+k+1)), {k,0,j-1}] /. Gamma[j] -> (j-1)!

Out[4]= -----
(i+j)!

and so we have
$$\sum_{k=0}^{j-1} \frac{(-1)^k}{k!(j-k-1)!(i+k+1)} = \frac{i!}{(i+j)!} = \frac{1}{(i+1)(i+2) \cdots (i+j)} \quad (8)$$

The kernel application $k(x,t)g(x)$ in equation (6) therefore becomes,

$$k(x,t)g(x) = \sum_{j=1}^n p_j(x) \left[\sum_{i=0}^m \frac{i!}{(i+j)!} g_i x^{i+j} \right] = \sum_{j=1}^n p_j(x) q_j(x) \quad (9)$$

where the polynomials q_j have the following form:

$$q_1 = g_0 \frac{1}{(0+1)} x^{(0+1)} + g_1 \frac{1}{(1+1)} x^{(1+1)} + \cdots + g_m \frac{1}{(m+1)} x^{(m+1)}$$

...

$$q_n = g_0 \frac{1}{(0+1) \cdots (0+n)} x^{(0+n)} + g_1 \frac{1}{(1+1) \cdots (1+n)} x^{(1+n)} + \cdots + g_m \frac{1}{(m+1) \cdots (m+n)} x^{(m+n)}$$

The structure of these polynomials is such that one can be computed from the previous. The functional operators are very useful here. We can generate a list with the q_j polynomials using *NestList*, perform an inner product with the list of the p_j polynomials and thus compute the kernel application $k(x,t)g(x)$. This is done by the *ApplyKernel* function. The pattern function *MonPat* constructs the monomials of polynomial q_k from the monomials of q_{k-1} starting from the expression $g(x)$ to which the kernel is applied.

166 Innovation In Mathematics

```
In[5]:= ApplyKernel[g_, p_, n_, x_, m_] :=  

      PolyTrunc[p . Rest[NestList[MonPat[# , x] &, g, n]], x, m]
```

```
In[6]:= MonPat[poly_, x_] :=  

      Collect[Expand[poly * x], x] /. c_. x^e_. -> 1/e c x^e
```

The computation of $y^{(n)}(x)$ in equation (5) involves repeated application of the kernel $k(x, t)$. Eventually the application of the kernel will return zero, since powers above some degree are not computed, and will not add any new terms to the series solution of the differential equation. This of course suggest the use of the *FixedPoint* function. *MapIndexed* is used to deal with the alternating signs of the kernel applications in equation (5) and *Flatten* prepares the resulting list of terms for summation. Having found $y^{(n)}(x)$, we integrate n times (done by the *IntegrateMon* function in the following code) and considering also the initial conditions we find the series solution for $y(x)$.

```
In[7]:= DESeries[p_, f_, ics_, n_, x_, m_] := Module[{g, apkerlist, sersol},  

      g = InitApprox[p, f, ics, n, x, m];  

      apkerlist = Flatten[MapIndexed[Times[#1, (-1)^(#2-1)] &,  

                          Drop[FixedPointList[ApplyKernel[# , p, n, x, m] &, g], -2]]];  

      sersol = Collect[Expand[Plus @@ apkerlist], x];  

      IntegrateMon[#1, x, n] & /@ sersol + Sum[ics[[i+1]] * x^i/i!, {i, 0, n-1}]]
```

```
In[8]:= IntegrateMon[mon_, x_, n_] := Module[{e, c},  

      e = Exponent[mon, x]; c = Coefficient[mon, x, e];  

      1/Product[e+i, {i, 1, n}] c x^(e+n) ]
```

3 An Example

In the following session we give a small example. Intermediate results were printed to show the initial approximation and the results of the kernel applications. *SeriesSolution* is a driver function which basically reads the differential equation, the initial conditions, the degree of the series solution, converts the coefficients and right hand side to normal series (if they are not polynomials) and calls the *DESeries* function.

```
In[10]:= DE1 = Dt[y[x], {x, 5}] - (x^5 + 6x^4) * y'[x] + (11x^7 - 8x^5) * y''[x]  

      == x^3
```

```
Out[10] = (-8 x^5 + 11 x^7) y''[x] - (6 x^4 + x^5) y'[x] + y'''[x] == x^3
```

```
In[11]:= SeriesSolution[DE1, y, x, {1, 0, 1, 0, 1}, 25]
```

Initial Approximation g:



$$x^3 + 6x^4 + x^5 + 11x^6 + \frac{x^7}{2} - \frac{29x^8}{3} - \frac{11x^{10}}{6}$$

$$\text{Kernel Application : } \frac{x^{10}}{20} + \frac{151x^{11}}{840} + \frac{47x^{12}}{840} + \frac{13x^{13}}{80} +$$

$$\frac{47x^{14}}{3024} - \frac{8839x^{15}}{110880} - \frac{13x^{16}}{1008} - \frac{239531x^{17}}{6486480} - \frac{11x^{18}}{6240} + \frac{2459x^{19}}{294840}$$

$$\text{Kernel Application : } \frac{x^{17}}{5720} + \frac{1759x^{18}}{3363360} + \frac{457x^{19}}{2059200} + \frac{39029x^{20}}{110073600}$$

There were 2 Kernel Applications

$$\text{Out}[11] = 1 + \frac{x^2}{2} + \frac{x^4}{24} + \frac{x^8}{6720} + \frac{x^9}{2520} + \frac{x^{10}}{30240} + \frac{x^{11}}{5040} + \frac{x^{12}}{190080} -$$

$$\frac{29x^{13}}{463320} - \frac{107x^{15}}{21621600} + \frac{151x^{16}}{440294400} + \frac{47x^{17}}{623750400} + \frac{13x^{18}}{82252800} +$$

$$\frac{47x^{19}}{4219568640} - \frac{8839x^{20}}{206290022400} - \frac{13x^{21}}{2461415040} - \frac{238397x^{22}}{20497795718400} -$$

$$\frac{139x^{23}}{452694802560} + \frac{1110751x^{24}}{661683230208000} + \frac{39029x^{25}}{701785244160000} + O[x]^{26}$$

4 Conclusions

The purpose of this work, apart from the design of the algorithm, was to show the power and flexibility of some of the functional operators in *Mathematica* and their usefulness in writing rather compact code, avoiding explicit iteration loops. As far as the behavior of the algorithm we observe the following.

If *maxorder* is the highest power of the series solution requested, then computations should only be done up to the $m = \text{maxorder} - n$ power since n integrations will be performed at the end. An implementation of the algorithm should truncate the polynomial coefficients $p_1(x), \dots, p_n(x)$ and the right hand side $f(x)$ of the differential equation after the m^{th} power. An analysis of the

168 Innovation In Mathematics

algorithm shows that in the worst case the time complexity is cubic in m , assuming that $\text{maxorder} \gg n$. However, from equation (9) we can see that the lowest power of polynomial $g(x)$ will be updated by a power which is the $\min\{j + \text{LowestPower}(p_j(x))\}$ for $j=1,2,\dots,n$, $p_j(x) \neq 0$, and the highest power of the polynomial $g(x)$ will be updated by a power which is the $\max\{j + \text{HighestPower}(p_j(x))\}$ for $j=1,2,\dots,n$, $p_j(x) \neq 0$. Therefore the worst case for the algorithm is when the polynomial $p_1(x)$ is present in the differential equation and contains the constant term, since in this case every new kernel application will update the series solution by a power only one higher than the previous kernel application. This means that if the initial approximation $g(x)$ also contains the constant term, the function *ApplyKernel* will be called the maximum number of times, that is m . On the other hand the algorithm performs very well for equations where the $(n-1)$, $(n-2)$, ... , derivatives are missing and the existing polynomials $p_j(x)$ do not have low degree terms. If for example $p_1(x), \dots, p_k(x)$ are zero polynomials, it is clear that the *ApplyKernel* function is applied fewer times since the degree of its argument, starting from the initial approximation $g(x)$, increases by $\min\{j + \text{LowestPower}(p_j(x))\}$ for $j = k+1, \dots, n$, $p_j(x) \neq 0$, and therefore we approach faster the maximum power of the series solution.

5 References

1. Singer, M.F. Formal Solutions of Differential equations, *Journal of Symbolic Computation*, 1990, 10, 59-94.
2. Berg, L. *Introduction to the Operational Calculus*, Interscience, John Wiley, New York, 1967.
3. Krabbe, G. *Operational Calculus*, Springer-Verlag, NewYork-Heidelberg-Berlin, 1970.
4. Glinos, N. Operational Calculus Methods for the Closed Form Solution of Ordinary Differential Equations, Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy N.Y., 1985.
5. Della Dora, J., Di Crescenzo, Cl., & Tournier, E. An algorithm to obtain formal solutions of a linear homogeneous differential equation at an irregular singular point, *Computer Algebra - EUROCAM '82, Lecture Notes in Computer Science*, Springer-Verlag, 144, 273-280, 1982.
6. Williamson, C. Taylor Series Solutions of Explicit ODEs in a Strongly Typed Algebra System, *ACM SIGSAM Bull.*, Vol. 18, No. 1., 1984.
7. Dietrich, V. ELISE: An Algorithm to Compute Asymptotic Representations for Solutions of Linear Differential Equations, Realized with the Computer Algebra System Maple, *Journal of Symbolic Computation*, 14, 85-92. 1992.
8. Wolfram, S. *MATHEMATICA: A System for doing Mathematics by Computer*, Addison Wesley, Second edition, 1991.