Chapter 1
Connections between Power Series Methods and Automatic Differentiation

David C. Carothers, Stephen K. Lucas, G. Edgar Parker,
Joseph D. Rudmin, James S. Sochacki, Roger J. Thelwell,
Anthony Tongen, Paul G. Warne

Abstract There is a large overlap in the work of the Automatic Differentiation community and those whose use Power Series Methods. Automatic Differentiation is predominately applied to problems involving differentiation, and Power series began as a tool in the ODE setting. Three examples are presented to highlight this overlap, and several interesting results are presented.

Introduction

In 1964, Erwin Fehlberg (best known for the Runge-Kutta-Fehlberg method) wrote:

> Like interpolation methods and unlike Runge-Kutta methods, the power series method permits computation of the truncation error along with the actual integration. This is fundamental to an automatic step size control [and leads to a method that is] far more accurate than the Runge-Kutta-Nystrom method.
>
> ⋮
>
> [Though] differential equations of the [appropriate form] . . . are generally not encountered in practice . . . a given system can in many cases be transformed into a system of [appropriate form] through the introduction of suitable auxiliary functions, thus allowing solution by power series expansions. [3]

Fehlberg, it appears, did not continue work on the approach that he believed to be superior to the methods of the day. In this manuscript (prepared as a NASA technical report) Fehlberg was able to efficiently and accurately compute approximations for two important problems: the restricted three-body problem, and the motion of an electron in a field of a magnetic dipole. Introducing auxiliary functions, he recast these problems as a system of first order equations expressed solely as polynomials in existing variables – what we now call *polynomial form*. Although this work was noticed by some in the NASA community [10], Fehlberg's observations remain largely unexploited.

The computation of derivatives lies at the heart of many Automatic Differentiation (AD) routines. AD techniques allow one to generate information about the intrinsic of interest solely in context of other intrinsics. When applied to functions, AD permits efficient evaluation of the derivatives of a given function up to arbitrarily high order, making it ideally suited for higher order Taylor based methods of solution to ODEs. The recursive computation that Fehlberg used is a natural outcome of recursively differentiating polynomial expressions of power series. The trick, then, is to reduce a given problem to one of polynomial form. How does one do so? One answer comes from the so called translator programs from AD.

A higher order Taylor method code is problem specific, requiring the problem to be reduced to one of known recursive relationships. First accomplished by hand, this difficulty was overcome with the advent of automatic translator programs. These programs can parse a given system of ODEs into a form that allows libraries of general recursions to be applied. A nice example of an AD flavored ODE tool is `ATOMFT`, written by Chang and Corliss in 1994. It automatically parses the original ODE expression using functional dependencies, and then efficiently computes a numeric solution via a recursive recovery of Taylor coefficients to arbitrarily high order. The method has also been applied to differential-algebraic equations (DAEs), with great success. The machinery of generating the polynomial form is distinct from the recursive coefficient recovery, and `ATOMF` is a wonderful blend of the techniques of AD applied to the ring of power series.

In 1992 Parker and Sochacki discovered that Picard Iteration, when applied to non-autonomous polynomial ODEs with initial value at $t = 0$, will generate the Maclaurin polynomial plus a polynomial all of whose terms have degree greater than the Maclaurin polynomial. They then looked at how one could use Picard Iteration to classify IV ODEs and what was special about the polynomials generated by Picard Iteration. This led Parker and Sochacki to determine which functions can be posed as a solution (to one component of a system) of non-autonomous polynomial ODEs with initial value at $t = 0$, and called this class of functions *projectively polynomial* [8]. Although the computation of successive terms of the Maclaurin polynomial was expensive, the framework allowed theoretic machinery to be applied to ordinary, partial and integral differential equations [5, 9, 6]. In [2] Carothers *et al* realized that the Picard iteration in the projectively polynomial system was equivalent to a power series method, allowing an efficient recursive computation of the coefficients of the series. For the remainder of this paper, we will refer to this method as the Power Series Method (PSM). PSM includes power series, Picard Iteration and polynomial projection.

Carothers *et al* realized that projectively polynomial functions had many special symbolic and numerical computational properties that were amenable to obtaining qualitative and quantitative results for the polynomial IVODES and the solution space. Gofen [4] and others also discovered some of these phenomena by looking at Cauchy products and polynomial properties instead of Picard Iteration. Many researchers were able to show large classes of functions were projectively polynomial, that one could uncouple polynomial ODEs and one could do interval analysis with these methods. Carothers *et al* generated an *a-priori* error bound for non-autonomous polynomial ODEs with initial value at $t = 0$. The PSM collaboration at JMU has also shown; the equivalence between power series and non-autonomous polynomial ODEs with initial value at $t = 0$ and Picard Iteration, many of the topological properties of the solutions to these problems, and the structure of the space of polynomial functions. These are summarized in Sochacki [11].

The AD and PSM methods produce equivalent results in the numerical solution of IVODEs. Polynomial form is essential for the simple recursive recovery of series coefficients used by the two groups. Their different history colors the types of problems explored, however. Differentiation forms the backbone of AD, and so problems which involve repeated differentiation are obvious candidates for AD research, with ODE, sensitivity, and root-finding as obvious examples. ODEs lie at the core of the PSM, and so the focus is to re-interpret problems as IVODEs. We present three examples below highlighting this concept. We believe that most problems can be converted to polynomial form, as demonstrated in the following examples.

Applying PSM

Example 1: PSM and AD applied to an IVODE

Consider the IVODE

$$y' = Ky^{\alpha}, \quad y(x_0 = 0) = y_0, \tag{1.1}$$

for some complex valued α. This problem highlights the properties of PSM and AD because it has the closed form solution

$$y(x) = \left(\left(Kx - K\alpha x + {y_0}^{1-\alpha} \right)^{(\alpha-1)^{-1}} \right)^{-1}, \tag{1.2}$$

and because it can be posed in several equivalent polynomial forms. Although not a problem that most would consider immediately amenable to standard power series methods, a simple recursive relationship generates the Taylor coefficients for y^{α} given by

$$a_n = \frac{1}{ny_0} \sum_{j=0}^{n-1} \left(n\alpha - j\left(\alpha + 1\right) \right) y_{n-j} a_j, \tag{1.3}$$

where $a(x) = \sum_{j=0}^{\infty} a_j(x-x_0)^j$ where a_n and y_n represent the nth degree Taylor coefficient of y^α and y, and a series solution of (1.1) is computed.

Consider the following change of variables: $x_1 = y$, $x_2 = y^\alpha$, and $x_3 = y^{-1}$. Fehlberg called these auxiliary variables. Differentiation of these auxiliary variables provides the system,

$$\begin{aligned} x_1' &= -x_2 & x_1(0) &= y_0, \\ x_2' &= -\alpha x_2^2\, x_3 & x_2(0) &= y_0^\alpha, \\ x_3' &= x_2\, x_3^2 & x_3(0) &= y_0^{-1}. \end{aligned} \tag{1.4}$$

The solution x_1 to this system is equal to y, the solution to the original system. Note that this augmented system (1.4) is polynomial, and as such can be easily solved by the PSM with its guaranteed error bound. Computation of the PSM solution requires only addition and multiplication.

However, a better change of variables is obtained by letting $w = y^{\alpha-1}$. Then (1.1) can be written as the following system of differential equations

$$\begin{aligned} y' &= Kyw, & y(0) &= y_0 \\ w' &= (\alpha-1)Kw^2, & w(0) &= y_0^{\alpha-1}, \end{aligned} \tag{1.5}$$

because the right hand side is quadratic in the variables as opposed to cubic in (1.4), and subsequently requires fewer series multiplications.

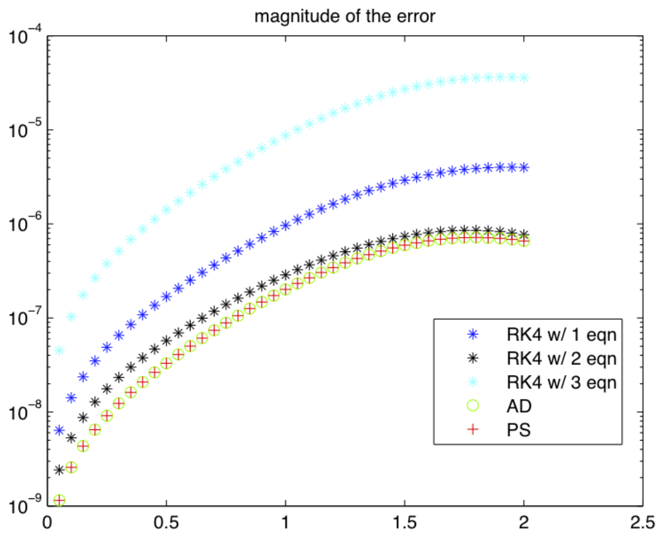

Fig. 1.1 Solving differential equations (1.1), (1.4), and (1.5) using a fixed step Runge-Kutta on [0,2] with $h = .05$ and $y_0 = 1, K = 1, \alpha = e/2 + i/\pi$.

Fig. 1.1 contrasts, on a $\log_{10}$ scale, the absolute error when approximate solutions to

$$y' = y^{\frac{e}{2}+\frac{i}{\pi}} \quad y(0) = 1$$

are computed by the standard Runge-Kutta order 4 method (RK4) for (1.1), (1.4), and (1.5), and automatic differentiation (using 1.3) and PSM (using 1.5) to 4th order. In this example, note that the PSM system (1.5) recovers the AD recursion (1.3).

Fig. 1.1 demonstrates that the fixed step solution with automatic differentiation and the power series solution of (1.5) give the same solution. Of course, we have fixed these methods at 4^{th} order in order to fairly compare with RK4; however, it is straightforward to keep more terms and solve this problem to machine accuracy, as Fehlberg points out. It also demonstrates that by rewriting the equations in polynomial form and solving with a fixed step RK4, the solution to the system of two equations (1.5) is more accurate than the straightforward implementation (1.1). Interestingly, not all systems are equal - the system of two equations (1.5) is more accurate than the system of three equations (1.4), because the right hand side of (1.5) is quadratic in the variables on the right hand side.

Example 2: Root-finding

Newton's Method is a prime example of the efficacy of AD. Consider

$$f(x) = e^{-\sqrt{x}} \sin(x \ln(1 + x^2)), \tag{1.6}$$

and computing the iteration $x_{i+1} = x_i - f(x_i)/f'(x_i)$, as in the example presented by Neidinger [7] to show the power of AD. The machinery of AD makes the calculation of $f(x_i)$ and $f'(x_i)$ simple, and Neidinger used object oriented programming and overloaded function calls to evaluate both the function and their derivative at a given value.

We take a different approach. We pose the determination of roots as a non-autonomous polynomial initial value ODE at 0. If one wants to determine the roots of a sufficiently nice function $f : \mathbb{R}^n \to \mathbb{R}^n$ one can define $g : \mathbb{R}^n \to \mathbb{R}$ by $g(x) = \frac{1}{2}\langle f(x), f(x)\rangle$ where $\langle \cdot, \cdot \rangle$ is the standard inner product. Since $g(x) = 0$ if and only if $f(x) = 0$ and is non-negative, we will determine the conditions that make $\frac{d}{dt}g(x) < 0$. This condition is necessary if one wants to determine $x(t)$ so that $x \to z$, a zero of f (or g). We have

$$\begin{aligned} \frac{d}{dt}g(x) &= \langle \tfrac{d}{dt}f(x), f(x)\rangle & (1.7)\\ &= \langle Df(x)x'(t), f(x)\rangle & (1.8)\\ &= \langle x'(t), Df(x)^T f(x)\rangle, & (1.9) \end{aligned}$$

where $Df(x)$ is the Jacobian of f and $Df(x)^T$ is the transpose of $Df(x)$.

If, guided by (1.8), we let

$$x'(t) = -(Df(x))^{-1} f(x), \tag{1.10}$$

then certainly $\frac{d}{dt}g(x) < 0$. If we now approximate the solution to this ODE using forward Euler with $h = 1$ we have

$$x_{k+1} = x_k - (Df(x_k))^{-1} f(x_k), \tag{1.11}$$

which is Newton's method. In (1.10), we let $x_2 = (Df(x))^{-1}$, and obtain

$$x'(t) = - x_2 f(x) \tag{1.12}$$

$$x_2'(t) = x_2^3 f(x) f''(x). \tag{1.13}$$

Adding initial conditions $x(0)$ and $x_2(0)$ gives us a non-autonomous initial value ODE. If f is a polynomial, we can apply PSM directly. If f is not polynomial, we make further variable substitutions to make the ODE polynomial.

Now consider (1.9). If we choose

$$x'(t) = -(Df(x))^T f(x) \tag{1.14}$$

then certainly $\frac{d}{dt}g(x) < 0$. Once again, approximating $x'(t)$ with forward Euler we have

$$x_{k+1} = x_k - h(Df(x_k))^T f(x_k), \tag{1.15}$$

which is the method of Steepest Descent.

We note that both (1.10) and (1.14) can be approximated using PSM or AD methods to arbitrary order (h^k), beating standard implementations of these methods. These ODEs could also be initially regularized in order for PSM or AD to converge faster to the root of f. In the case of the Newton form, we would then solve

$$x'(t) = -\alpha(t)(f'(x))^{-1} f(x),$$

where $\alpha(t)$ could be adaptive. Of course, this approach applies easily to higher dimensions and the method of Steepest Descent in a straightforward manner. We now have many options for developing numerical methods to approximate the zeroes of a function f.

Neidinger chose the initial condition 5.0 and produced the approximation 4.8871. We used the IVODE (1.10) and did a polynomial projection on f to obtain a non-autonomous polynomial IVODE. Using 5.0 as our initial condition at $t = 0$, a time step of 0.0625 and 32^{nd} degree Maclaurin polynomials in this IVODE, we obtained the first 30 digits of the root. This approximation is $z = 4.88705596745554192341704846671$ with $f(z) = -9.26192756612973974458163983522238 * 10^{-30}$. In comparison $f(4.8871)$ is -0.00002478163.

Example 3: The Maclaurin polynomial for an inverse function

In 2000, Apostol [1] developed a method for obtaining the power series of the inverse of a polynomial by exploiting the Inverse Function Theorem. To turn this problem into a non-autonomous polynomial ODE with initial value at $t = 0$ we differentiate $f(f^{-1}(t)) = t$ to obtain $f'(z)z' = 1$, where we let $z = f^{-1}(t)$. We now let $y = [f'(z)]^{-1}$ and $x = y^2$ to obtain

$$z' = \frac{1}{f'(z)} = [f'(z)]^{-1} = y \tag{1.16}$$

$$y' = -y^2 f''(z)z' = -xf''(z)z'. \tag{1.17}$$

$$x' = 2yy' \tag{1.18}$$

Suppose f is a polynomial. We now outline how to get the power series for its inverse. Let $f(t) = \sum_{i=0}^{n+2} a_i t^i = a_0 + a_1 t + ... + a_{n+2} t^{n+2}$. Using the above polynomial ODE we now have

$$y' = -y^2 f''(z)z' = -xf''(z)z' = -xp_n z' = -xyp_n \tag{1.19}$$

$$x' = 2yy' \tag{1.20}$$

$$p_n' = f'''(z)z' = p_{n-1}y \tag{1.21}$$

$$p_{n-1}' = f^{(iv)}(z)z' = p_{n-2}y \tag{1.22}$$

$$\vdots$$

$$p_1' = f^{(n+2)}(z)z' = (n+2)!a_{n+2}y, \tag{1.23}$$

where $p_n = f''(z)$. We have ignored the z' equation since $z' = y$. Now we use Cauchy products and find

$$y = \sum_{i=0}^{K} y_i t^i \ ; \ y' = \sum_{i=0}^{K-1} y_{i+1} t^i \tag{1.24}$$

$$x = \sum_{i=0}^{K} x_i t^i \tag{1.25}$$

$$p_{n-k} = \sum_{i=0}^{K} p_{(n-k),i} t^i \ ; \ k = 0, ..., n-1. \tag{1.26}$$

Substituting these in gives us a simple algorithm for generating the power series for the derivative of the inverse of a polynomial. One integration gives the power series for the inverse.

These 3 examples are meant to show the similarities and differences of PSM and AD and how PSM can be applied to many problems of applied and computational mathematics by posing them as non-autonomous polynomial IVODEs. These examples have also raised questions of interest. For example; (1) Is it more efficient to pose the problem as a non-autonomous polynomial IVODE or solve it in the existing form using AD? (2) Does the structure and topology of non-autonomous polynomial IVODEs lead to answers in applied and computational mathematics? (3) What are the symbolic and numerical differences and similarities between PSM and AD? (4) How can the PSM, AD and polynomial communities come together to answer these questions?

PSM Theory and AD

Picard Iteration and polynomial projection for IVODEs has led to an interesting space of functions and some interesting results for polynomial ODEs. We present the basic definitions and important theorems arising from Picard Iteration and polynomial projections. The proofs can be found in the papers. Gofen and others have obtained some of these results through the properties of polynomials and power series.

We begin with the question of which ODEs may be transformed into an autonomous polynomial system as in Example 1; that is, a system of the form:

$$\mathbf{x}'(t) = \mathbf{h}(\mathbf{x}(t)), \quad \mathbf{x}(a) = \mathbf{b}, \tag{1.27}$$

noting that a non-autonomous system $\mathbf{y}'(t) = \mathbf{h}(\mathbf{y}(t), t))$ may be recast by augmenting the system with an additional variable whose derivative is 1.

To this end the class of *projectively polynomial* functions consists of all real analytic functions which may be expressed as a component of the solution to (1.27) with h a polynomial. The following properties of this class of functions are summarized in [2] and elsewhere. It may be shown that any polynomial system, through the introduction of additional variables, may be recast as a polynomial system of degree at most two.

The projectively polynomial functions include the so-called elementary functions. The class of projectively polynomial functions is closed under addition, multiplication, and function composition. A local inverse of a projectively polynomial function f is also projectively polynomial (when $f'(a) \neq 0$), as is $\frac{1}{f}$. The following theorem illustrates the wide range of ODEs that may be recast as polynomial systems.

Theorem 1. *Suppose that f is projectively polynomial. If y is a solution to*

$$y'(t) = f(y(t)); \quad y(a) = b \tag{1.28}$$

then y is also the component of a solution to a polynomial system at point a. That is, y is also projectively polynomial.

As an interesting consequence, it is possible for a very wide range of systems of ODEs to provide an algorithm by which the system may be "decoupled" by applying standard Gröbner basis techniques.

Theorem 2. *A function is the solution to a polynomial system of differential equations if and only if for some n there is a polynomial Q in $n+1$ variables so that $Q(u, u', \cdots, u^{(n)}) = 0$.*

That is, for any component x_i of the polynomial system $\mathbf{x}' = \mathbf{h}(\mathbf{x})$ the component x_i may be isolated in a single equation involving x_i and its derivatives. This implies, for example, that the motion of one of the two masses in a double pendulum may be described completely **without** reference to the second mass.

Of very special practical and theoretical interest is the existence of explicit *a-priori* error bounds for PSM solutions to ODEs of this type which depends only on immediate observable quantities of the polynomial system.

We consider again a polynomial system (at $a = 0$) of the form $\mathbf{x}'(t) = \mathbf{h}(\mathbf{x}(t)),\ \ \mathbf{x}(0) = \mathbf{b}$. In the following $K = (m-1)c^{m-1}$, where m is the degree of $\mathbf{h}$ (the largest degree of any single term on the right hand side of the system), M is the larger of unity and the maximum row sum of the absolute values of the constant coefficients of the system, c the larger of unity and the magnitude of $\mathbf{b}$ (the largest of the absolute value of the elements of the initial condition), and $\sum_{k=0}^{n} \mathbf{x}_k t^k$ is the n^{th} degree Taylor approximation of $\mathbf{x}(t)$. As an example we have the following error estimate with $m \geq 2$ ([12]):

Theorem 3.

$$\left\| \mathbf{x}(t) - \sum_{k=0}^{n} \mathbf{x}_k t^k \right\| \leq \frac{\|\mathbf{b}\| \, |Kt|^{n+1}}{1 - |Mt|} \quad \textit{for} \quad m \geq 2 \tag{1.29}$$

for any $n \in \mathbb{N}$, with $|t| < \frac{1}{K}$.

It can be shown that no universally finer error bound exists for all polynomial systems than one that is stated in a tighter but slightly more involved version of this theorem.

Conclusion

Clearly, there is a large overlap in the work of the AD community and the PSM group. However, while AD is predominately applied to problems involving differentiation, PSM began as a tool in the ODE setting. There are numerous benefits to sharing the tool-sets of recursive computation of Taylor coefficients between these two communities. Some of these are: (1) There are methods that easily compute *arbitrarily high order Taylor coefficients*, (2) The tools can *solve highly nonlinear IV ODEs*, and *automatically solve stiff problems*, (3) There are numerical and symbolic computational tools that lead to semi-analytic methods and (4) Evaluation of functions can be *interpolation free* to machine capability (error and calculation).

References

1. T.M. Apostol. Calculating higher derivatives of inverses. *Amer. Math. Monthly*, 107(8):738–741, 2000.
2. David Carothers, G. Edgar Parker, James Sochacki, and Paul G. Warne. Some properties of solutions to polynomial systems of differential equations. *Electronic Journal of Differential Equations*, 2005:1–18, 2005.
3. E. Fehlberg. Numerical integration of differential equations by power series expansions, illustrated by physical examples. Technical Report NASA-TN-D-2356, NASA, 1964.
4. A.M. Gofen. The ordinary differential equations and automatic differentiation unified. *Complex Variables and Elliptic Equations*, 54:825–854, 2009.
5. James Liu, G. Edgar Parker, James Sochacki, and Aren Knutsen. Approximation methods for integrodifferential equations. *Proceedings of the International Conference on Dynamical Systems and Applications,III*, pages 383–390, 2001.
6. James Liu, James Sochacki, and Paul Dostert. Chapter 16: Singular perturbations and approximations for integrodifferential equations. In Sergiu Aizicovici and Nicolae H. Pavel, editors, *Differential Equations and Control Theory*. CRC press, 2001. ISBN: 978-0-8247-0681-4.
7. R.D. Neidinger. Introduction to automatic differentiation and matlab object-oriented programming. *SIAM Review*, 52(3):545–563, 2010.
8. G. Edgar Parker and James Sochacki. Implementing the Picard iteration. *Neural, Parallel Sci. Comput.*, 4(1):97–112, 1996.
9. G. Edgar Parker and James Sochacki. A Picard-Maclaurin theorem for initial value PDE's. *Abstract Analysis and its Applications*, 5:47–63, 2000.
10. I.E. Perlin and C.P. Reed. The application of a numerical integation procedure developed by Erwin Fehlberg to the restricted problem of three bodies. Technical Report NAS8-11129, NASA, 1964.
11. James Sochacki. Polynomial ordinary differential equations - examples, solutions, properties. *Neural Parallel & Scientific Computations*, 18(3-4):441–450, 2010.
12. P. G. Warne, D.A. Polignone Warne, J. S. Sochacki, G. E. Parker, and D. C. Carothers. Explicit a-priori error bounds and adaptive error control for approximation of nonlinear initial value differential systems. *Comput. Math. Appl.*, 52(12):1695–1710, 2006.