Connections between Power Series and Automatic Differentiation

D.C. Carothers, S. Lucas, **G. E. Parker**, J.D. Rudmin, **J.S. Sochacki**, R.J. Thelwell, A. Tongen & P. G. Warne

James Madison University, VA

6th International Conference on Automatic Differentiation

Outline

History

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.

⋮

History

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.

Fehlberg, in 1964 [1]

History

- 1989 : Parker and Sochacki and Picard iteration

History

- 1964: Fehlberg
- 1989 : Parker and Sochacki and Picard iteration

History

- 1830s Cauchy & Weierstrass
- 1964: Fehlberg

- 1989 : Parker and Sochacki and Picard iteration

History

- 1830s Cauchy & Weierstrass
- 1964: Fehlberg
- 1982: Chang and Corliss
- 1989 : Parker and Sochacki and Picard iteration

History

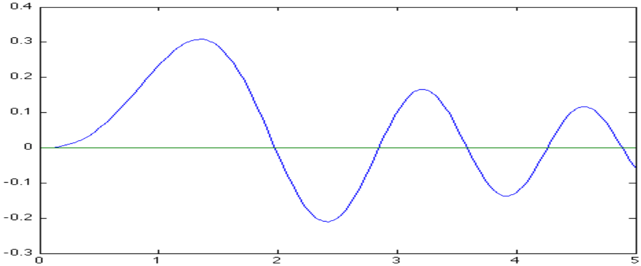
- 1830s Cauchy & Weierstrass
- 1964: Fehlberg
- 1980: Rall
- 1982: Chang and Corliss
- 1989: Lohner
- 1989 : Parker and Sochacki and Picard iteration

EXAMPLE: ROOTFINDING

The Problem

Consider

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



Neidinger's 2010 SIAM article. [2]

EXAMPLE: ROOTFINDING

Neidinger's Newton

Neidinger: Define valder, a MATLAB OOP class and overload functions to handle the class. For example:

```
function h = sin(u)
h = valder(sin(u.val), cos(u.val)*u.der);
end
```

and then evaluate as needed...

```
function vec = fdf(a)
x = valder(a,1);
y = exp(-sqrt(x))*sin(x*log(1+x^2));
vec = double(y);
```

EXAMPLE: ROOTFINDING

TAPENADE's SCT for Newton

From

```
Y = EXP(-SQRT(x))*SIN(x*LOG(1+x**2))
```

to the preprocessed

```
result1 = SQRT(x)
arg1 = 1 + x**2
arg2 = x*LOG(arg1)
y = EXP(-result1)*SIN(arg2)
```

EXAMPLE: ROOTFINDING

TAPENADE's SCT for Newton

And then (tangent) mode

```
      result1d = xd/(2.0*SQRT(x))
      result1 = SQRT(x)
      arg1d = 2*x*xd
      arg1 = 1 + x**2
      arg2d = xd*LOG(arg1) + x*arg1d/arg1
      arg2 = x*LOG(arg1)
      yd = EXP(-result1)*arg2d*COS(arg2) -
+                 result1d*EXP(-result1)*SIN(arg2)
      y = EXP(-result1)*SIN(arg2)
```

EXAMPLE: ROOTFINDING

Roots as IVODE

Roots of f coincide with the roots of

$$g(x) = \frac{1}{2}\langle f(x), f(x)\rangle.$$

Since $g(x)$ is non-negative and $g(x) = 0$ if and only if $f(x) = 0$, we want

$$\frac{d}{dt}g(x) < 0.$$

EXAMPLE: ROOTFINDING

Root conditions

If

$$g(x) = \frac{1}{2}\langle f(x), f(x)\rangle.$$

then

$$\frac{d}{dt}g(x) = \langle \tfrac{d}{dt}f(x), f(x)\rangle \quad (2)$$

$$= \langle Df(x)x'(t), f(x)\rangle \quad (3)$$

$$= \langle x'(t), Df(x)^T f(x)\rangle. \quad (4)$$

EXAMPLE: ROOTFINDING

Option A

From

$$\frac{d}{dt}g(x) = \langle Df(x)x'(t), f(x)\rangle$$

$g'(x) < 0$ if

$$x'(t) = -(Df(x))^{-1}f(x). \tag{5}$$

Approximating x' with forward Euler (and $\Delta t = 1$) yields

$$x_{t+1} = x_t - (Df(x_t))^{-1}f(x_t),$$

EXAMPLE: ROOTFINDING

Option A

From

$$\frac{d}{dt}g(x) = \langle Df(x)x'(t), f(x)\rangle$$

$g'(x) < 0$ if

$$x'(t) = -(Df(x))^{-1}f(x). \quad (5)$$

Approximating x' with forward Euler (and $\Delta t = 1$) yields

$$x_{t+1} = x_t - (Df(x_t))^{-1}f(x_t),$$

Newton's Method!

EXAMPLE: ROOTFINDING

Option B

From

$$\frac{d}{dt}g(x) = \langle x'(t), Df(x)^T f(x) \rangle$$

we see $g'(x) < 0$ if

$$x'(t) = -Df(x)^T f(x). \tag{6}$$

EXAMPLE: ROOTFINDING

Option B

From

$$\frac{d}{dt}g(x) = \langle x'(t), Df(x)^T f(x)\rangle$$

we see $g'(x) < 0$ if

$$x'(t) = -Df(x)^T f(x). \tag{6}$$

Again approximating x' with forward Euler (this time with arbitrary Δt)...

$$x_{t+\Delta t} = x_t - \Delta t(Df(x_t))^T f(x_t),$$

EXAMPLE: ROOTFINDING

Option B

From

$$\frac{d}{dt}g(x) = \langle x'(t), Df(x)^T f(x) \rangle$$

we see $g'(x) < 0$ if

$$x'(t) = -Df(x)^T f(x). \tag{6}$$

Again approximating x' with forward Euler (this time with arbitrary Δt)...

$$x_{t+\Delta t} = x_t - \Delta t (Df(x_t))^T f(x_t),$$

Steepest Descent - and easily applied in higher dimensions!

EXAMPLE: ROOTFINDING

Newton as polynomial ODE

To recast

$$x'(t) = -(Df(x))^{-1} f(x). \tag{7}$$

in polynomial form, first introduce $x_2 = (Df(x))^{-1}$.
Then

$$x'(t) = -\, x_2 f(x) \quad \text{and} \tag{8}$$
$$x_2'(t) = x_2^3 f(x) f''(x), \tag{9}$$

to handle the reciprocal. Of course, f' and f'' might be messy...

EXAMPLE: ROOTFINDING

IVODE approach to Newton

for $f(x) = e^{-\sqrt{x}} \sin(x \ln(1 + x^2))$ we'll need...

$$
\begin{aligned}
x_4 &= \ln(1 + x^2) \\
x_5 &= (1 + x^2)^{-1} \\
x_6 &= x * x_4 \\
x_7 &= \sin(x_6) \\
x_8 &= \cos(x_6) \\
x_9 &= x^{1/2} \\
x_{10} &= x^{-1/2} \\
x_{11} &= e^{-x_9}
\end{aligned}
$$

EXAMPLE: A SIMPLE ODE

The problem

Consider

$$y' = \sin(y) \qquad y(t_0) = y_0 \tag{10}$$

If we let

$$x_1 = y, \quad x_2 = \sin(y), \quad \text{and} \quad x_3 = \cos(y) \tag{11}$$

we get a polynomial system.

EXAMPLE: A SIMPLE ODE

The polynomial system

Taking $x_1 = y, \quad x_2 = \sin(y) \quad$ and $\quad x_3 = \cos(y)$, then

$$\begin{aligned} x_1' &= 1 \cdot y' = x_2 \\ x_2' &= x_3 \cdot y' = x_2\ x_3 \\ x_3' &= -x_2 \cdot y' = -x_2^2 \end{aligned} \quad \text{and} \quad \begin{aligned} x_1(t_0) &= y_0 \\ x_2(t_0) &= \sin(y_0) \\ x_3(t_0) &= \cos(y_0) \end{aligned} \tag{12}$$

We can solve this system with series recursion.

EXAMPLE: A SIMPLE ODE

The polynomial system

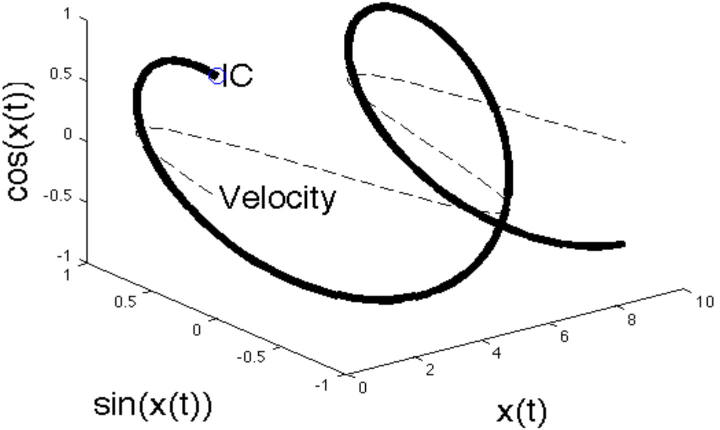
Taking $x_1 = y$, $x_2 = \sin(y)$ and $x_3 = \cos(y)$, then

$$\begin{aligned} x_1' &= 1 \cdot y' = x_2 \\ x_2' &= x_3 \cdot y' = x_2\ x_3 \\ x_3' &= -x_2 \cdot y' = -x_2^2 \end{aligned} \quad \text{and} \quad \begin{aligned} x_1(t_0) &= y_0 \\ x_2(t_0) &= \sin(y_0) \\ x_3(t_0) &= \cos(y_0) \end{aligned} \qquad (12)$$

We can solve this system with series recursion. But, we can also consider the geometry....

EXAMPLE: A SIMPLE ODE

The Geometry



EXAMPLE: A SIMPLE ODE

What about AD?

Calling TAYLOR, (or ATOMFT, or ...)

```
$ taylor -main -o simple_ex.c  simple_ex.in
```

we get (a differential equation) AND the final variable list...

```
        v_008 (state variable)
        v_022 = sin(v_008)                              (1 0)
        v_023 = cos(v_008)                              (2 0)
```

which is exactly our change of variables!

EXAMPLE: ANOTHER ODE
The problem

Consider the IVODE

$$y' = Ky^{\alpha}, \quad y(x_0 = 0) = y_0 \tag{13}$$

- Why?

EXAMPLE: ANOTHER ODE

The problem

Consider the IVODE

$$y' = Ky^{\alpha}, \quad y(x_0 = 0) = y_0 \tag{13}$$

- Why?
- Because we have an analytic solution!

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

EXAMPLE: ODEs

Recurrent power series

First represent $y(x) = \sum_{j=0}^{\infty} y_j (x - x_0)^j$,.
Since

$$y'(x) = \sum_{j=1}^{\infty} j\, y_j (x - x_0)^{j-1},$$

and $y^\alpha = \sum_{j=0}^{\infty} a_j (x - x_0)^j$, where

$$a_n = \frac{1}{n y_0} \sum_{j=1}^{n-1} (n\alpha - j\,(\alpha + 1))\, y_{n-j} a_j, \tag{14}$$

it's a simple recursion to recover coefficients y_j.

EXAMPLE: ODEs

Recurrent power series

First represent $y(x) = \sum_{j=0}^{\infty} y_j (x - x_0)^j,$.
Since

$$y'(x) = \sum_{j=1}^{\infty} j\, y_j (x - x_0)^{j-1},$$

and $y^\alpha = \sum_{j=0}^{\infty} a_j (x - x_0)^j$, where

$$a_n = \frac{1}{n y_0} \sum_{j=1}^{n-1} \left(n\alpha - j\,(\alpha + 1) \right) y_{n-j} a_j, \tag{14}$$

it's a simple recursion to recover coefficients y_j.
Just like Lara in the 1990s. Or Steffensen in the 1950s. Or Cauchy in 1830s?

EXAMPLE: ODEs

some AD ODE tools

- ATOMFT (Chang & Corliss)

EXAMPLE: ODEs
some AD ODE tools

- ATOMFT (Chang & Corliss)
- TAYLOR (Jorba & Zou)

EXAMPLE: ODEs

some AD ODE tools

- ATOMFT (Chang & Corliss)
- TAYLOR (Jorba & Zou)
- The Taylor Center (Gofen)

EXAMPLE: ODEs

some AD ODE tools

- ATOMFT (Chang & Corliss)
- TAYLOR (Jorba & Zou)
- The Taylor Center (Gofen)
- TIDES (Abad, Barrio, Blesa, Rodriguez)

EXAMPLE: ODEs

Option A

Consider the following change of variables:
$x_1 = y$, $x_2 = y^\alpha$, and $x_3 = y^{-1}$.

Then,

$$\begin{aligned} x_1' &= -x_2 & x_1(0) &= y_0, \\ x_2' &= -\alpha x_2^2\ x_3 & x_2(0) &= y_0^\alpha, \\ x_3' &= x_2\ x_3^2 & x_3(0) &= y_0^{-1}. \end{aligned} \tag{15}$$

EXAMPLE: ODEs

Option B

Or, better yet, let $w = y^{\alpha-1}$.

Then,

$$\begin{aligned} y' &= Kyw, & y(0) &= y_0 \\ w' &= (\alpha-1)Kw^2, & w(0) &= y_0^{\alpha-1}, \end{aligned} \tag{16}$$

EXAMPLE: ODEs

A comparison

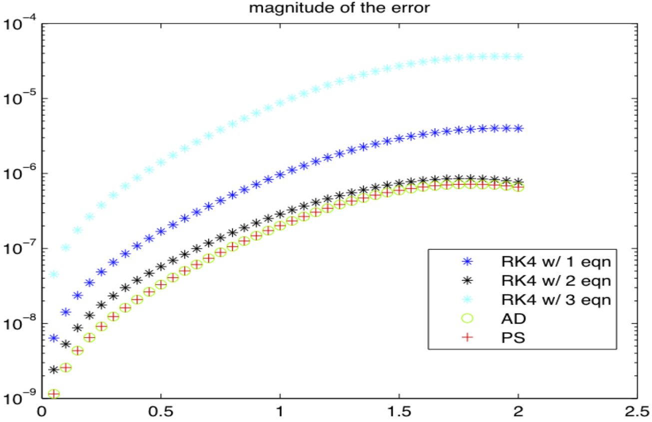


Figure: Error when using a fixed step Runge-Kutta on [0,2] with $h = .05$ and $y_0 = 1, K = 1, \alpha = e/2 + i/\pi$.

EXAMPLE: INVERSE FUNCTIONS

Series representations of inverse functions are easy. From

$$f(f^{-1}(t)) = t,$$

differentiate to obtain $f'(x_1)x_1' = 1$, where $x_1 = f^{-1}(t)$.
To cast in polynomial form, let $x_2 = [f'(x_1)]^{-1}$, and $x_3 = x_2^2$ to obtain

$$x_1' = \frac{1}{f'(x_1)} = [f'(x_1)]^{-1} = x_2 \quad (17)$$

$$x_2' = -x_2^2 f''(x_1)x_1' = -x_3 f''(x_1)x_1'. \quad (18)$$

$$x_3' = 2x_2x_2' \quad (19)$$

THEORY

Projectively Polynomial class

$\mathbf{x_i}$ is Projectively Polynomial if

$$\mathbf{x}'(t) = \mathbf{h}\left(\mathbf{x}(t)\right) \quad \text{where} \quad \mathbf{x}(a) = \mathbf{b},$$

where $\mathbf{h}$ is polynomial.

Projectively polynomial family contains the elementary functions:

1. polynomials
2. `exp` and `ln`
3. Trig funcs: `sin, cos, tan`

THEORY

Projectively Polynomial class

$\mathbf{x_i}$ is Projectively Polynomial if

$$\mathbf{x}'(t) = \mathbf{h}\left(\mathbf{x}(t)\right) \quad \text{where} \quad \mathbf{x}(a) = \mathbf{b},$$

where $\mathbf{h}$ is polynomial.
The class is closed under:

1. +, -, *, /
2. Functional composition and inverse

THEORY

Decoupling

Carothers et. al. 2005 [3]

Theorem

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

THEORY

Decoupling

Carothers et. al. 2005 [3]

Theorem

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

This implies that the motion of one of the two masses in a double pendulum may be described completely **without** reference to the second mass.

Warne et. al. 2006 [1]

If we have (at $a = 0$) a system $\mathbf{x}'(t) = \mathbf{h}(\mathbf{x}(t))$, $\mathbf{x}(0) = \mathbf{b}$. then

$$\left\| \mathbf{x}(t) - \sum_{k=0}^{n} \mathbf{x}_k t^k \right\|_\infty \leq \frac{\|\mathbf{b}\|_\infty |Kt|^{n+1}}{1 - |Mt|} \quad \text{for} \quad m \geq 2 \tag{20}$$

Where the parameters K and M depend on immediately observable quantities of the original system;

M is the largest row sum of coefficients, and $K = (m-1)c^{m-1}$, where $c = \max\{1, ||\mathbf{b}||_\infty\}$ and $m = deg(\mathbf{h})$.

QUESTIONS

- Efficiency

QUESTIONS

- Efficiency
- Links in Structure and Parsing

QUESTIONS

- Efficiency
- Links in Structure and Parsing
- Intuition

QUESTIONS

- Efficiency
- Links in Structure and Parsing
- Intuition
- Other connections between PSM and AD

CONCLUSION

AD is predominately applied to problems involving differentiation, while 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 are:

- Easily compute *arbitrarily high order Taylor coefficients*

CONCLUSION

AD is predominately applied to problems involving differentiation, while 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 are:

- Easily compute *arbitrarily high order Taylor coefficients*
- The tools can *solve highly nonlinear and stiff problems*

CONCLUSION

AD is predominately applied to problems involving differentiation, while 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 are:

- Easily compute *arbitrarily high order Taylor coefficients*
- The tools can *solve highly nonlinear and stiff problems*
- Semi-analytic methods and

CONCLUSION

AD is predominately applied to problems involving differentiation, while 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 are:

- Easily compute *arbitrarily high order Taylor coefficients*
- The tools can *solve highly nonlinear and stiff problems*
- Semi-analytic methods and
- *interpolation free* to machine capability (error and calculation)

References

E. Fehlberg
Numerical integration of differential equations by power series expansions, illustrated by physical examples.
Technical Report NASA-TN-D-2356, NASA, 1964.

R.D. Neidinger
Introduction to automatic differentiation and Matlab object-oriented programming.
SIAM Review, 52(3):545–563, 2010.

David Carothers et. al.
Some properties of solutions to polynomial systems of differential equations.
Electronic Journal of Differential Equations, 2005:1–18, 2005.

References

P. G. Warne et. al.
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.

SUMMARY of PSM

James Sochacki
Polynomial ordinary differential equations - examples, solutions, properties.
Neural Parallel & Scientific Computations, 18(3-4):441–450, 2010.