IMPLEMENTING THE PICARD ITERATION

G. Edgar Parker
James S. Sochacki

Department of Mathematics
James Madison University
Harrisonburg, VA 22807

Implementing the Picard Iteration
Neural, Parallel & Scientific Computations
March 1996,Volume 4,Number 1, pp. 97-112

ABSTRACT:

In this paper we show that the Picard Iteration can be used to generate the Taylor Series solution to any ordinary differential equation on $\Re^n$ that has a polynomial generator. This is accomplished by embedding evolutions into autonomous problems and translating initial conditions to zero. We present a definition that permits extensive use of a technique from A. Gibbons and R. Moore and prove a theorem which shows the generality of using polynomial generators. The power of the theory in machine implementation is illustrated.

I. INTRODUCTION

The ordinary differential equation with generator $F : \Re^{n+1} \to \Re^n$

$$y'(t) = F(t, y(t)); y(T) = x$$

is equivalent to the integral equation

$$y(t) = x + \int_T^t F \circ (I, y),$$

where I represents the identity function on $\Re$. The Picard iteration [9] (modeled from the integral equation),

$$for\ t \geq T; y_1(t) = x\ and\ y_{n+1}(t) = x + \int_T^t F \circ (I, y_n), n \geq 1,$$

must converge to y on some interval containing T whenever F is locally Lipschitz. For F analytic, the sequence of Taylor Polynomials,

$$y_n(t) = x + y'(T)(t - T) + ... + y^{[n]}(T)\frac{(t - T)^n}{n!}$$

must also converge to y on some interval containing T. The computational difficulties associated with computing the Taylor coefficients (especially for $n > 1$ and/or F nonlinear) are well documented. The Picard iteration is typically dismissed in the literature as impractical. [1]

[1] See, for instance, [2] p.89, 1992, " ... As a rule, it is impossible to compute explicitly more than a few members of the (Picard) sequence, therefore the limit function can only be found in rare cases. ... "

We present five theorems which show that, in fact, a proper modification of the Picard iteration can be used to generate or approximate the Taylor polynomials for the entire class of problems with analytic F. Furthermore, the indicated processes are well suited for computer implementation. Schemes for symbolic and numerical codes are presented. The schemes are applied to some classical problems. Some structural examples are considered in light of this theory.

II. DEFINITIONS AND THEOREMS FOR A MODIFIED PICARD METHOD

In order to realize the full impact of the theory on applications it is important to have a precise definition for a polynomial on $\Re^n$. We offer here a definition that translates directly into a (numerical) computing environment.

Definition 1

Suppose that $n \epsilon N$. Let

$$E = \{f | f : \{k \mid k \epsilon N \text{ and } k \leq n\} \rightarrow N \cup \{0\}\},$$

and suppose that

$$P : \Re^n \rightarrow \Re^n.$$

The statement that

$$\underline{P \text{ is a polynomial function on } \Re^n}$$

means that there are

$$T : \{k \mid k \epsilon N \text{ and } k \leq n\} \rightarrow 2^E,$$

and

$$c : \{(k,q) \mid k \epsilon N \text{ and } k \leq n \text{ and } q \epsilon T(k)\} \rightarrow \Re - \{0\},$$

and there is

$$v \epsilon \Re^n \text{ so that if } k \leq n, \text{ then if } x \epsilon \Re^n,$$

$$\Pi_k P(x) = v_k + \sum_{q \epsilon T(k)} c(k,q) \prod_{i=1}^{n} x_i^{q(i)}.$$

Here, $\Pi_k P(x)$ denotes the projection of $P(x)$ into the k^{th} component and v_k (x_i) denotes the projection of v (x) into the k^{th} (i^{th}) component. Note that T selects the exponents of the factors of the individual terms and that c selects the coefficients for the terms.

The conclusion of Theorem 1 involves a comparison of coefficients for the polynomials which are the Picard iterates. To focus on this comparison we will denote the

coefficient of the i^{th} power of the k^{th} component of the s^{th} iterate, p_s, as $a_{k,s,i}$; that is,

$$\Pi_k p_s(t) = x_k + \sum_{i=1}^{j_s} a_{k,s,i} t^i.$$

Theorem 1

Suppose that $P : \Re^n \to \Re^n$ is a polynomial. Let $x \epsilon \Re^n$.
Define

$$p_1 = \{(t, w) : t \geq 0 \text{ and } w = x\},$$

and if $s \epsilon N$, define

$$p_{s+1} = \{(t, w) : t \geq 0 \text{ and } w = x + \textstyle\int_0^t P \circ p_s\}.$$

Then if $r \epsilon N,\ m > r,\ k \leq n$,

$$(\Pi_k p_r)(t) = x_k + \sum_{i=1}^{j_r} a_{k,r,i} t^i,$$

and

$$(\Pi_k p_m)(t) = x_k + \sum_{i=1}^{j_m} a_{k,m,i} t^i;$$

then if $l < r$,

$$a_{k,r,l} = a_{k,m,l}.$$

Note $j_r \leq j_m$.

Proof

Observe that if $s \epsilon N$, each component function of p_s is a polynomial on $\Re$ because each component function of the generator P multiplies numbers times powers of components of the previous iterate. The induction is guaranteed by the fact that each component of p_1 is a constant function and thus a polynomial on $\Re$.

At each step of the iteration, in any component each term from the integral in the iteration, because of the power rule, has exponent at least 1. Thus the constant term for any pair of iterates in corresponding components is the appropriate component of the initial condition, and the induction is begun.

P is given to be a polynomial on $\Re^n$, so let T, c and v define P as described in Definition 1, that is, for $k \leq n$ and $y \epsilon \Re^n$

$$\Pi_k P(y) = v_k + \sum_{q \epsilon T(k)} c(k, q) \prod_{i=1}^{n} y_i^{q(i)}.$$

For the purpose of the inductive argument, let $r \epsilon N$ so that

$$if\ s > r,\ k \leq n, (\Pi_k p_r)(t) = x_k + \sum_{i=1}^{j_r} a_{k,r,i} t^i,\ and\ (\Pi_k p_s)(t) = x_k + \sum_{i=1}^{j_s} a_{k,s,i} t^i,$$

$$then\ if\ m < r,\ a_{k,r,m} = a_{k,s,m}.$$

(This has been established for $r = 1$.)

We will now establish the result for $r+1$. Let $s > r+1$ and consider p_{r+1} and p_s. For $k \leq n$, if $t \geq 0$,

$$\Pi_k p_{r+1}(t) = x_k + \int_0^t v_k + \sum_{q \epsilon T(k)} c(k,q) \prod_{i=1}^{n} (\Pi_i p_r)^{q(i)},$$

and

$$\Pi_k p_s(t) = x_k + \int_0^t v_k + \sum_{q \epsilon T(k)} c(k,q) \prod_{i=1}^{n} (\Pi_i p_{s-1})^{q(i)}.$$

Again letting I represent the identity function on $\Re$

$$\Pi_k p_{r+1}(t) = x_k + \int_0^t v_k + \sum_{q \epsilon T(k)} c(k,q) \prod_{i=1}^{n} ((x_i + \sum_{d=1}^{r-1} a_{i,r,d} I^d) + \sum_{d=r}^{j_r} a_{i,r,d} I^d)^{q(i)}$$

and

$$\Pi_k p_s(t) = x_k + \int_0^t v_k + \sum_{q \epsilon T(k)} c(k,q) \prod_{i=1}^{n} ((x_i + \sum_{d=1}^{r-1} a_{i,s-1,d} I^d) + \sum_{d=r}^{j_{s-1}} a_{i,s-1,d} I^d)^{q(i)}.$$

By the induction hypothesis if $k \leq n, x_k + \sum_{d=1}^{r-1} a_{k,r,d} I^d = x_k + \sum_{d=1}^{r-1} a_{k,s-1,d} I^d$. Denote each such function by M_k.

From the Binomial Theorem ($C_{\alpha,\beta}$ represents the appropriate binomial coefficient)

$$\Pi_k p_{r+1}(t) = x_k + \int_0^t v_k + \sum_{q \epsilon T(k)} c(k,q) \prod_{i=1}^{n} (M_i^{q(i)} + \sum_{b=1}^{q(i)} C_{q(i),b} M_i^{q(i)-b} (\sum_{d=r}^{j_r} a_{i,r,d} I^d)^b)$$

and

$$\Pi_k p_s(t) = x_k + \int_0^t v_k + \sum_{q \epsilon T(k)} c(k,q) \prod_{i=1}^{n} (M_i^{q(i)} + \sum_{b=1}^{q(i)} C_{q(i),b} M_i^{q(i)-b} (\sum_{d=r}^{j_{s-1}} a_{i,s-1,d} I^d)^b).$$

Every term of $\sum_{b=1}^{q(i)} C_{q(i),b} M_i^{q(i)-b} (\sum_{d=r}^{j_r} a_{i,r,d} I^d)^b$ and $\sum_{b=1}^{q(i)} C_{q(i),b} M_i^{q(i)-b} (\sum_{d=r}^{j_{s-1}} a_{i,s-1,d} I^d)^b$ has degree at least r. Therefore, the terms of both $\Pi_k p_{r+1}(t)$ and $\Pi_k p_s(t)$ of degree r must come from integrating $v_k + \sum_{q \epsilon T(k)} c(k,q) \prod_{i=1}^{n} M_i^{q(i)}$ which is the same in the integrand of both $p_{r+1}(t)$ and $p_s(t)$. Thus the induction is continued. $\square$

It should be noted that for efficiency in applications the proof shows that to compute the term of power r, only those terms of power less than r need be carried forward in the iteration.

Theorem 1 guarantees that, for an ODE with an autonomous polynomial generator and initial conditions given at 0, the 0 through $s-1$ degree terms in the s^{th} Picard iterate (denoted by p_s in the proof) are the corresponding terms in the s^{th} degree Maclaurin polynomial for the solution to the ODE.

We now present theorems that show how to broaden the applicability of Theorem 1 to evolution equations, ODE's with initial conditions not given at 0, and to ODE's with analytic generators that are not polynomials.

Initially, we consider evolution equations and establish connections among the technique suggested by Theorem 1, Picard Iteration, and Taylor Series. The technique utilizes the classic embedding of an evolution in an autonomous problem. (See for instance Goldstein [5].)

Theorem 2

Let P be a polynomial on $\Re^{n+1}$ and y be a function into $\Re^n$ so that if $t \geq T$ and $t \epsilon D(y')$, then

$$y'(t) = P(t, y(t)),\ y(T) = x.$$

Let u be a function into $\Re^{n+1}$ so that if $t \geq 0$ and $t \epsilon D(u')$, then

$$u'(t) = (1, P(u(t)),\ u(0) = (T, x).$$

Define

$$p_1 = \{(t, w) : t \geq T \ \text{ and } \ w = x\},$$

and if $s \epsilon N$,

$$p_{s+1} = \{(t, w) : t \geq T \ \text{ and } \ w = x + \textstyle\int_T^t P \circ (I, p_s)\}.$$

Define

$$q_1 = \{(t, w) : t \geq 0 \ \text{ and } \ w = (T, x)\},$$

and if $s \epsilon N$,

$$q_{s+1} = \{(t, w) : t \geq 0 \ \text{ and } \ w = (T, x) + \textstyle\int_0^t (1, P \circ (q_s))\}.$$

Then if $s \epsilon N, \Pi_2 q_s(t - T) = p_s(t)$.

Proof

From the embedding, $y = \{(\Pi_1 u(t), \Pi_2 u(t)), t \geq 0\}$. Therefore if $t \geq T$, then $y(t) = \Pi_2 u(t - T)$. We observe that if $t \geq T, \Pi_2 q_1(t - T) = p_1(t) = x$. For the purpose of induction, suppose that for $t \geq T, \Pi_2 q_s(t - T) = p_s(t)$.

$$q_{s+1}(t) = (T, x) + \int_0^t (1, P \circ q_s)$$

Every iterate produces $\Pi_1 q_s = I + T$, thus

$$\Pi_2 q_{s+1}(t) = x + \int_0^t P \circ (I + T, \Pi_2 q_s).$$

Therefore,

$$\Pi_2 q_{s+1}(t-T) = x + \int_0^{t-T} P \circ (I+T, \Pi_2 q_s).$$

Shifting the interval of integration

$$\Pi_2 q_{s+1}(t-T) = x + \int_T^t P \circ ((I+T)\circ(I-T), \Pi_2 q_s \circ (I-T)) = x + \int_T^t P \circ (I, \Pi_2 q_s \circ (I-T))$$

From the induction hypothesis, $\Pi_2 q_s \circ (I-T) = p_s$. Thus,

$$\Pi_2 q_{s+1}(t-T) = x + \int_T^t P \circ (I, p_s) = p_{s+1}(t),$$

and the induction is continued.□

Since Theorem 1 guarantees that the Maclaurin Series is generated by Picard iteration, the embedding in the proof of Theorem 2 now gives the Taylor Series by the substitution $t-T$ in the Maclaurin series for the second component. Furthermore, the fact that $\Pi_2 q_s(t-T) = p_s(t)$ means that the Picard iterate must telescope to produce the first s terms of the Taylor Series. Thus to get these s terms the ENTIRE Picard iterate must be computed. On the other hand, from Theorem 1 using the embedding, only the first s terms must be computed. Furthermore, from the proof of Theorem 1, the computation can be further simplified since only those terms known to be terms of the Maclaurin Series for the solution need be used to continue the iteration.

Many important applications involve ODE's with polynomial generators (For example, Lorenz, van der Pol, Volterra-Lotka, the Logistic equation). Many other analytic problems, however, do not (For example, Duffing, the N-body problem, the torus equation). With the following definition and theorems we formalize a technique from Gibbons [4] and Moore [8] to extend the applicability of Theorem 1 to a broad class of problems with analytic generators.

Definition 2

Suppose that $y : [0, T) \to \Re$ is analytic. The statement that

y is projectively polynomial

means that there is $n \epsilon N$, a polynomial P on $\Re^n$, and w so that $w' = P \circ w$ for which there is $k \leq n$ so that $y = \Pi_k w$.

To illustrate Definition 2 consider sine. $sin' = I^{\frac{1}{2}} \circ (1 - I^2) \circ sin; sin(0) = 0$. Note that $I^{\frac{1}{2}} \circ (1 - I^2)$ is analytic around 0, but not polynomial. However, sine **is** projectively polynomial since $sin' = cos$ and $cos' = -sin$ and thus there exists the ODE with polynomial generator $P(u, v) = (-v, u)$ for which sine is the projection into the first component of w defined by $w' = P \circ w; w(0) = (0, 1)$.

Theorem 3

Suppose that each of f and g is projectively polynomial. Then $f+g, fg,$ and $f \circ g$ are projectively polynomial.

Proof

$$(f+g)' = f' + g';$$
$$(fg)' = f'g + fg';$$
$$(f \circ g)' = f' \circ gg'.$$

To complete the proof, the component of the generator into which f or g is projected replaces f' or g' as appropriate and the polynomial equations into whose solutions f and g are projected are appended. $\square$

Theorem 4

The projectively polynomial elements of $C_{[0,1]}$ are dense in $C_{[0,1]}$.

Proof

Consider $f = a_0 + \sum_{k=1}^{n} a_k I^k$.

$$(a_0 + \sum_{k=1}^{n} a_k I^k)' = a_1 + \sum_{k=2}^{n} k a_k I^{k-1}$$

Let $f_k = I^k$.

$$f' = a_1 + \sum_{k=2}^{n} k a_k f_{k-1}.$$

$$f_1' = 1.$$

For $1 < m < n$,

$$f_m' = m f_{m-1}.$$

The fact that the polynomials are dense in $C_{[0,1]}$ completes the argument. $\square$

Theorem 5

Suppose that A is a convergent power series on $\Re^n$, that B is a polynomial which is a partial sum for A, that $x \epsilon \Re^n, 0 < T < \frac{1}{|B|_{Lip}}$, and y and w are functions so that if $t \epsilon [0,T]$,

$$y'(t) = A \circ y\ (t) \text{ and } y(0) = x;$$
$$w'(t) = B \circ w\ (t) \text{ and } w(0) = x.$$

Then if $t \epsilon [0,T]$,

$$||y(t) - w(t)|| \leq \frac{|(A-B) \circ y|T}{1 - (T|B|_{Lip})}.$$

Proof

Let A, B, x, y, w and T be as in the premise.
Let $s \epsilon [0,T]$ so that $|y-w|_{sup} = ||y(s)-w(s)||$. Then if $t \epsilon [0,T]$,

$$||y(t)-w(t)|| \leq ||y(s)-w(s)||.$$

Since

$$\int_0^s (y'-w') = \int_0^s A \circ y - B \circ w,$$

$$||y(s)-w(s)|| \leq \int_0^s |B \circ y - B \circ w|_{sup} + \int_0^s |(A-B) \circ y|_{sup}.$$

Thus

$$|y-w|_{sup} \leq \int_0^s |B|_{Lip} |y-w|_{sup} + \int_0^s |(A-B) \circ y|_{sup},$$

and

$$|y-w|_{sup} \leq |B|_{Lip} |y-w|_{sup} T + |(A-B) \circ y|_{sup} T.$$

Subtracting, dividing by $1 - T|B|_{Lip}$, and applying the first inequality, then

$$||y(t)-w(t)|| \leq \frac{|(A-B) \circ y|_{sup} T}{1 - T|B|_{Lip}}. \square$$

In the event we were to have an analytic generator which was not projectively polynomial, [2] Theorem 5 allows us to use an approximating problem with a polynomial generator to approximate solutions. The proof of Theorem 4 allows us to replace this approximating equation with an equivalent system of lower degree.

III. SOME APPLICATIONS

We present examples that use our modified Picard method and demonstrate the power and flexibility of polynomial projections. All the presented Taylor polynomials are generated using a single code that implements our modified Picard method.

Example 1: (Milne [7])

Milne offered the following two examples to illustrate the impracticality of the Picard iteration. (To see that this attitude has not changed, recall Boyce & DiPrima's more recent comment given in the Introduction.)

Milne notes that if y_k is the k^{th} Picard iterate for the equation

$$y'(t) = t^2 - y(t)^2; \; y(-1) = 0$$

then 'It is apparent that the labor of obtaining successive approximations increases rapidly. Evidently ... y_5 (is) of degree 63 ... '.

[2] At present the authors have no example of an analytic problem to which the method does not apply.

Imbedding the evolution so that Theorem 1 applies:

$$w'(t) = 1;\ w(0) = -1$$
$$v'(t) = w(t)^2 - v(t)^2;\ v(0) = 0.$$

It follows that

$$y = \{(w(t), v(t)) | t\ \epsilon\ D(v')\}.$$

Using only those terms of the Maclaurin Polynomial approximating the solution to continue the iteration, we get

$$w_1(t) = -1, w_k(t) = -1 + t, k > 1,$$
$$v_1(t) = 0,$$
$$v_2(t) = \int_0^t w_1^2 - v_1^2 = t,$$
$$v_3(t) = \int_0^t w_2^2 - v_2^2 = t - t^2,$$
$$v_4(t) = \int_0^t w_3^2 - v_3^2 = t - t^2 + \frac{t^4}{2} - \frac{t^5}{5},$$
$$v_5(t) = \int_0^t (w_4^2 - (I - \frac{I^2}{2} + 0I^3)^2) = t - t^2 + \frac{t^4}{2} - \frac{t^5}{5},^3$$
$$v_6(t) = \int_0^t (w_5^2 - (I - \frac{I^2}{2} + 0I^3 + \frac{I^4}{2})^2) =$$
$$t - t^2 + \frac{t^4}{2} - \frac{t^5}{5} - \frac{t^6}{6} + \frac{t^7}{5} - \frac{t^8}{20} - \frac{t^9}{36} + \frac{t^{10}}{50} - \frac{t^{11}}{275}.$$

Thus the 5^{th} degree Taylor polynomial for y is

$$(t+1) - (t+1)^2 + \frac{(t+1)^4}{2} - \frac{(t+1)^5}{5}.$$

($\frac{-(t+1)^6}{6}$ is also a term due to the special properties of the problem, but is not guaranteed by the proof to be a term of the Maclaurin Series.)

In way of contrast, from Theorem 2 this polynomial can be gotten by telescoping (not a trivial computational task itself!) the 6^{th} Picard iterate for the original equation, a 63^{rd} degree polynomial, and then taking the first 5 powers. However, note the savings in calculation; to continue the Picard iteration a 63^{rd} degree polynomial must be squared, whereas to continue our process a 5^{th} degree polynomial is squared. Therefore, the savings in calculation is exponential.

$$y'(t) = sin(t) + cos(y(t));\ y(0) = 0$$

Milne points out that the integral that yields the third Picard iterate cannot be obtained by elementary methods. However, $sin + cos \circ y$ is projectively polynomial, and we can make the projection $w = sin, v = cos, z = cos \circ y, x = sin \circ y$.

[3] We have included the third degree term of the Maclaurin Series to conform to the proof of Theorem 1. In practice this is not necessary since the next iterate simply repeats the previous step.

$$w'(t) = v(t);\ w(0) = 0$$
$$v'(t) = -w(t);\ v(0) = 1$$
$$y'(t) = w(t) + z(t);\ y(0) = 0$$
$$z'(t) = -x(t)(w(t) + z(t));\ z(0) = 1$$
$$x'(t) = z(t)(w(t) + z(t));\ x(0) = 0$$

has a second degree generator and thus the Maclaurin series is easily generated. Note that determining the Maclaurin Series by the classic textbook method of differentiating the generator is comparatively more computationally expensive for even moderate powers.

Example 2: A Stiff System

$$y_1'(t) = -11y_1(t) + 10y_2(t) + 5cos(t) - 0.25sin(t);\ y_1(0) = 2.25$$
$$y_2'(t) = 10y_1(t) - 11y_2(t) - 9cos(t) + 0.25sin(t);\ y_2(0) = 0.75$$

The eigenvalues of this system are -1 and -21. The eigenvalue -21 effects the stability and the eigenvalue -1 has an effect over time.

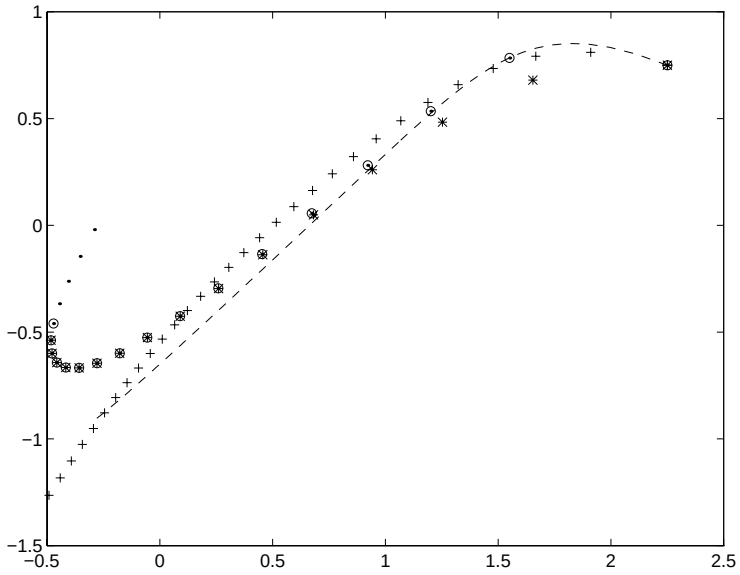
An equivalent autonomous problem with polynomial generator for the above system is

$$y_1' = -11y_1 + 10y_2 + 5y_3 - (0.25)y_4;\ y_1(0) = 2.25$$
$$y_2' = 10y_1 - 11y_2 - 9y_3 + (0.25)y_4;\ y_2(0) = 0.75$$
$$y_3' = -y_4;\ y_3(0) = 1.0$$
$$y_4' = y_3;\ y_4(0) = 0.0.$$

We applied Runge-Kutta of order 4 to each of these systems. We applied our modified Picard scheme to produce fourth and eighth degree Taylor polynomials for the autonomous problem. The same code was used to generate both of these Taylor Polynomials; to get an eighth order scheme for Runge-Kutta a new algorithm must be used and thus a new code must be written.

The largest increment for which the Runge-Kutta scheme of order 4 on the original system gave meaningful results was .0125. Smaller increments did not improve the results. The impact of the polynomial projection is noticeable by observing that the Runge-Kutta scheme of order 4 on the autonomous problem gives meaningful results for an increment of 0.05. The Taylor Polynomials are computed for an increment of 0.1.

The following graph displays the results in comparison to a graph of the closed-form solution for $0 \leq t \leq 1.5$.



ooooo Graph of Closed-Form Solution
..... Graph of 8^{th} Degree Taylor (h = 0.1) [4]
***** Graph of 4^{th} Degree Taylor (h = 0.1)
+++++ Graph of 4^{th} Order Runge-Kutta for the projected system (h = 0.05)
- - - - Graph of 4^{th} Order Runge-Kutta for the original system (h = 0.0125)

Example 3:

This example illustrates the effects of choosing different projections for the generator and of choosing higher degree relative to smaller increment, and shows how our methods provide better accuracy than NDSolve, the built-in numerical ODE solver of *Mathematica*.

Consider

$$y'(t) = \frac{1}{y^3(t)} + t,\ y(0) = \frac{1}{4}.$$

By projecting the generator using y_2 as the identity on $\Re$ and y_3 as y^{-3}, we get the autonomous system

$$y' = y_3 + y_2;\ y(0) = \frac{1}{4}$$
$$y_2' = 1; y_2(0) = 0$$
$$y_3' = -3y_3^2 y_1^2 (y_3 + y_2); y_3(0) = 64.$$

[4]The graph has been extended to distinguish it from the closed-form solution.

By projecting the generator using y_2 as the identity on $\Re$, y_3 as y^{-3}, and y_4 as $\frac{1}{y}$ we get the autonomous system

$$y' = y_3 + y_2;\ y(0) = \frac{1}{4}$$
$$y_2' = 1; y_2(0) = 0$$
$$y_3' = -3y_3y_4(y_3 + y_2);\ y_3(0) = 64$$
$$y_4' = -y_4^2(y_3 + y_2);\ y_4(0) = 4.$$

We sought a numerical solution on $[0, 1]$ for the original equation by applying our modified Picard scheme to produce fourth degree Taylor polynomials and continuing the solution by using final values for one increment as initial values for the next increment. The first projection overflows between 0.007 and 0.008. The second projection, however, computes values over the entire interval, although the data does not have even two-digit accuracy. Thus, for this ODE under these conditions, larger dimension (4) with smaller degree (3) is more robust than smaller dimension (3) with larger degree (5). Thus not only is it true that using a polynomial generator can improve computational ability and/or accuracy (see Example 2), but which projection is chosen may also affect these factors.

By decreasing the increment from 0.001 to 0.00001 we obtain agreement through 11 decimal places on $[0, 1]$ for both fourth and eighth degree Taylor polynomials for the second projection, a strong indication of the accuracy of the solution. Another verification of the accuracy of our results is comparing these results with the closed form solution to

$$w' = \frac{1}{w^3};\ w(0) = 0.25.$$

Analysis shows that $|y(0.0001) - w(0.0001)| < 510^{-8}$. Our data displays this property. For example, the value of $w(0.0001)$ to 12 decimal places is 0.256167960022 and, our approximation of $y(0.0001)$ is 0.256167964906. Running the standard NDSolve, *Mathematica* gives the approximation 0.256166029504 for $y(0.0001)$ which is not within the above estimate. Increasing the accuracy in NDSolve to 25 digits rather than the default of 16 digits gives our result. Note that we bettered *Mathematica*'s default accuracy with only a fourth degree Taylor Polynomial on an increment of 0.00001.

Example 4: The N-Body Problem (Birkhoff [1])

$$x_i''(t) = \sum_{j \neq i} \frac{m_j(x_j - x_i)}{\xi_{i,j}^{\frac{3}{2}}}$$
$$y_i''(t) = \sum_{j \neq i} \frac{m_j(y_j - y_i)}{\xi_{i,j}^{\frac{3}{2}}}$$
$$z_i''(t) = \sum_{j \neq i} \frac{m_j(z_j - z_i)}{\xi_{i,j}^{\frac{3}{2}}}$$

where $\xi_{i,j} = [(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2]$.

The generator is projectively polynomial. A projection for this problem with a 5^{th} degree generator obtained by letting $s_{i,j} = \xi_{i,j}^{-\frac{1}{2}}$, $u_i = x_i'$, $v_i = y_i'$, and $w_i = z_i'$ is

$$x_i' = u_i$$
$$y_i' = v_i$$
$$z_i' = w_i$$
$$u_i' = \sum_{j \neq i} m_j (x_j - x_i) s_{i,j}^3$$
$$v_i' = \sum_{j \neq i} m_j (y_j - y_i) s_{i,j}^3$$
$$w_i' = \sum_{j \neq i} m_j (z_j - z_i) s_{i,j}^3$$

$$s_{i,j}' = -\frac{1}{2} s_{i,j}^3 [2(x_i - x_j)(u_i - u_j) + 2(y_i - y_j)(v_i - v_j) + 2(z_i - z_j)(w_i - w_j)].$$

This system has dimension $6N + \frac{N(N-1)}{2}$, since $s_{i,j} = s_{j,i}$. Fortunately, the iteration can be done in parallel. So even though memory requirements may be substantial the time factor can be managed.

Joseph Rudmin of the Physics Department at James Madison has used the modified Picard scheme on planetary models.

Example 5:

$$y' = a_0 + \sum_{k=1}^{n} a_k y^k$$

Let $w_k = y^k$. An equivalent system (following the proofs of Theorems 3 and 4) is

$$y' = a_0 + a_1 y + \sum_{k=2}^{n} a_k w_k$$

For $k \geq 2$,

$$w_k' = k w_{k-1} (a_0 + a_1 y + \sum_{k=2}^{n} a_k w_k)$$

Note

- (i) The system is linear in the first component.
- (ii) The generator in the first component is a factor of the generator of every other component and need be computed only once.
- (iii) Every other component is quadratic using this as the second factor.

Thus in a parallel environment, the solution can be computed quite rapidly. Therefore, if your generator is a Maclaurin polynomial (such as when Theorem 5 is applied) the above substitution is effective.

Example 6: Resolvent of a Semigroup

Let T represent the semigroup on $C_{[0,1]}$ with infinitesimal generator A, defined by

$$A(f) = ff''.$$

For the resolvent equation,

$$(I - \lambda A)^{-1}(g) = f$$

implies

$$f - \lambda ff'' = g.$$

If g is projectively polynomial into a system with solution $\overline{g}$ and generator P, the system

$$f' = h$$
$$h' = w\frac{(f-g)}{\lambda}$$
$$w' = -w^2 h$$
$$\overline{g}' = P \circ \overline{g}$$

solves the resolvent problem and gives a computational basis from which to iterate the resolvent such as in the Crandall-Liggett theory [3].

Example 7: Computer Generated Output

Because the implementation of the method involves only the addition and multiplication of polynomials and the power rule for anti-derivatives, our method can be implemented in both numerical and symbolic environments. Consider

$$(x', y') = (1 + y - \alpha x^2, \beta x),\ (x, y)(0) = (c, d),$$

the differential equation which generates the Henon Dynamical System [6].

In a numerical environment we must specify the parameters and initial conditions. Using Henon's values for illustration purposes, let $\alpha = 1.4, \beta = 0.43$, and $(c, d) = (0.63135448, 0.18940634)$. We obtain

(x,y)(t) =

$$(0.631354 + 0.631352t + -0.422308t^2 + 0.108082t^3 + 0.123738t^4,$$

$$0.189403 + 0.271482t + 0.135741t^2 + -0.0605308t^3 + 0.0116188t^4)$$

as the fourth degree Maclaurin polynomial.

By working in a symbolic environment we can generate the general fourth degree Maclaurin polynomial.

(x,y)(t) =

$$(c + [1 + (\alpha)(c^2) + d]t + [(\alpha c) + ((\beta c)/2) + (\alpha^2)(c^3) + (\alpha cd)]t^2 +$$

$$[\alpha/3 + \beta/6 + (4/3)(\alpha^2)(c^2) + (1/2)(\alpha\beta)(c^2) +$$

$$(\alpha^3)(c^4) + (2/3)(\alpha d) + (1/6)(\beta d) + (4/3)(\alpha^2)(c^2)d + (\alpha/3)(d^2)]t^3 +$$

$$[(2/3)(\alpha^2)c + (5/12)(\alpha\beta c) + (1/24)(\beta^2)c + (5/3)(\alpha^3)(c^3) + (7/12)(\alpha^2)(c^3)\beta +$$

$$(\alpha^4)(c^5) + (4/3)(\alpha^2)(cd) + (5/12)(\alpha\beta cd) + (5/3)(\alpha^3)(c^3)d + (2/3)(\alpha^2)(d^2)c]t^4,$$

$$d + [\beta c]t + [\beta/2)(1 + \alpha(c^2) + d]t^2 + [\beta c/6)(2\alpha + \beta + 2(\alpha^2)(c^2) + 2\alpha d]t^3 +$$

$$[\beta/24)(1 + (3\alpha)(c^2) + d)(2\alpha + \beta + 2(\alpha^2)(c^2) + 2\alpha d]t^4).$$

This formula essentially gives us access to the entire dynamical system; the solution from any initial conditions can be continued by using final values as the next initial conditions, new initial conditions can be posed by substituting for c and d, and new parameters imposed by substituting for α and β. Substituting Henon's values into this formula verifies that both outputs give the same polynomial.

IV. SOME QUESTIONS

Section 2 provides a link between the Picard process and Taylor's Theorem for a large class of linear and non-linear ordinary differential equations. However, nothing in the theory guarantees whether or not the estimates for the Picard process, which can be estimated *a priori*, hold for the Taylor polynomials which have been shown to be truncations of the Picard iterates and are known to be truncations of the solution. An estimate for the error in using the Taylor polynomial as an approximation that does not depend on the n^{th} derivative of the solution would clearly be advantageous. Can the link between Picard and Taylor be used to accomplish this?

We have proven that the analytic generators which are projectively polynomial are dense in the analytic functions. Are they the set of analytic functions?

REFERENCES

[1] Birkhoff, G. and Rota, G-C. *Ordinary Differential Equations* Third Edition. John Wiley and Sons, Inc. (1978).

[2] Boyce, W.E. and DiPrima, R.C. *Elementary Differential Equations* Fifth Edition. John Wiley and Sons, Inc. (1992).

[3] Crandall, M.G. and Liggett, T.M., Generators of semigroups of nonlinear transformations on general Banach spaces, *American Journal of Mathematics* 93, (1971) 265 ff.

[4] Gibbons, A. A program for the automatic integration of differential equations using the method of Taylor series, *Computer Journal* 3 (1960) 108-111.

[5] Goldstein, J.A. An example of a nonlinear semigroup, *Nieuw Archief Voor Wiskunde* 3 XXIII (1974) 170-174.

[6] Henon, M. A two-dimensional mapping with a strange attractor *Communications in Mathematical Physics* 50 (1976) 69-77.

[7] Milne, W.E. *Numerical Solution of Differential Equations* Second Revised and Enlarged Edition. Dover Publications, Inc. (1970).

[8] Moore, R.E. *Interval Analysis* Prentice-Hall (1966).

[9] Picard, E. *Traite D'Analyse* Volume 3, Gauthier-Villars (1922-1928).