The Mathematica® Journal

Exact and Approximate Solutions of the Abel–Volterra Equations

Javad Abdalkhani

Picard's iteration is used to find the analytical solutions of some Abel–Volterra equations. For many equations, the integrals involved in Picard's iteration cannot be evaluated. The author approximates the solutions of those equations employing a semi-implicit product midpoint rule.The Aitken Δ^2 extrapolation is used to accelerate the convergence of both methods. A blow-up phenomena model, a wave propagation, and a superfluity equation are solved to show the practicality of the methods. Programs offered solve the general equations. A user needs only to enter the particular forcing and kernel functions, constants, and a step size for a particular problem.

Introduction

Abel–Volterra equations are normally represented by

$$y(t) = g(t) + \int_0^t \frac{K(t, s, y(s))}{(t-s)^\alpha}\, ds, \text{ where } 0 \le \alpha < 1, t \in [0, T_0], T_0 \in (0, \infty). \tag{1}$$

Equation (1) is called *regular* if $\alpha = 0$ and *weakly singular* (or of *Abel type*) if $\alpha \neq 0$. Equation (1) is *linear* if $K(t, s, y(s)) = K(t, s)\, y(s)$; otherwise, it is *nonlinear*. In most practical applications, α is either 0 or 1/2. See [1–4] for conditions on existence, uniqueness, and continuity of a solution for equation (1). To solve equation (1) analytically, one normally employs the Picard method, a method of successive iterations, given by

$$y_0(t) = g(t), \tag{2}$$

$$y_n(t) = g(t) + \int_0^t \frac{K(t, s, y_{n-1}(s))}{(t-s)^\alpha}\, ds, \text{ where } n = 1, 2, \ldots. \tag{3}$$

From a theoretical point of view, the successive iterations given by equations (2) and (3) always converge for linear equations; see theorem 10.15, page 152 of [5] and pages 92–95 of [4]; see also solutions of some integral equations by the Picard method in [2]. In practice the convergence of successive iterations depends on the computability of the corresponding integrals in equation (3). One might be able to evaluate the integrals in some cases. Successive iterations may also be effective for some nonlinear equations. In what follows, we first introduce a simple program that implements the successive iterations and solve two examples using this program. For many integral equations whose exact solutions cannot be found by the Picard method, we approximate their analytical solution using a semi-implicit product midpoint rule. Two practical examples are solved to test the validity of this numerical approach.

■ *Picard's Iteration*

The following program implements equations (2) and (3).

```
PicardIteration[Ker_, g_Function, y_, α_?NumericQ,
  n_Integer] := ( y[0] = g;
  y[m_] := Block[{t, temp},
    temp = g[t] + Integrate[Ker[t, s, y[m - 1][s]] / (t - s) ^ α,
        {s, 0, t}, Assumptions → (t > 0 && Re[α] < 1)];
    y[m] = Function @@ List[t, temp]
   ]
  ;
  y[n])
```

One needs only to introduce the kernel `Ker`, the forcing function `g`, and the real value α to solve the corresponding equation.

■ *Statement and Solution of Example 1*

Example 1

To solve the equation

$$y(t) = 1 + \lambda \int_0^t \frac{y(s)}{(t-s)^{\alpha}}\, ds, \tag{4}$$

note that

$$\int_0^t \frac{s^{\gamma}}{(t-s)^{\alpha}}\, ds = t^{\gamma+1-\alpha}\, B(\gamma+1, 1-\alpha), \tag{5}$$

where B is the well-known beta function defined by

$$B(z, w) = \frac{\Gamma(z)\,\Gamma(w)}{\Gamma(z+w)}, \tag{6}$$

and Γ is the gamma function.

Then it is easy to verify that the outputs $y_n(t)$ produced by the Picard method for this example can be written as

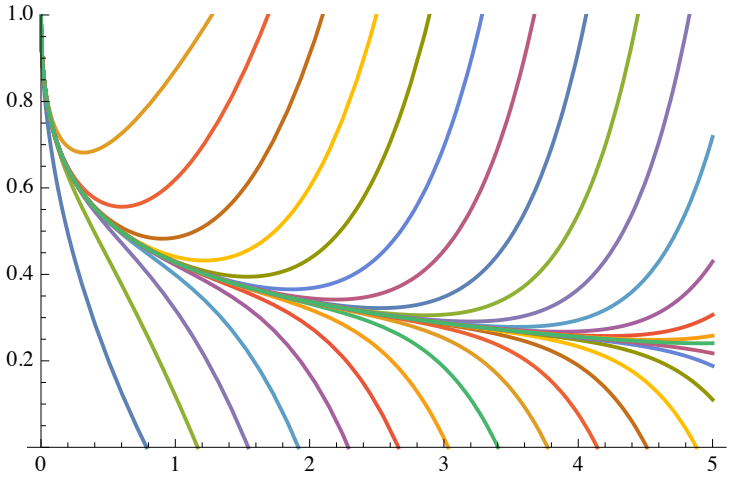
$$y_n(t) = \sum_{i=0}^{n} \lambda^i a_i t^{i-i\alpha}, \tag{7}$$

where $a_0 = 1$, $a_i = a_{i-1} B(i-(i-1)\alpha, 1-\alpha)$, $\alpha = \frac{1}{2}$, $\lambda = -\frac{1}{\sqrt{\pi}}$, $g(t) = 1$, and $K(t, s, y) = -\frac{y}{\sqrt{\pi}}$.

Now substitute the corresponding kernel, forcing function, and the value of α. The evaluation takes a minute or two.

```
g1 = Function[t, 1];
Ker1[t_, s_, y_] := -y / Sqrt[Pi];
PicardIteration[Ker1, g1, y1, 1 / 2, 30];

Plot[Evaluate[Table[y1[n][t], {n, 30}]], {t, 0, 5},
 PlotRange → {0, 1}]
```



▲ This graph shows convergence of $y_n(t)$, $n = 1, 2, \ldots, 30$, for $0 \le t \le 5$.

■ Nonlinear Accelerators to Speed Up the Convergence

Aitken's Δ^2 method accelerates the convergence [6], but accelerators like this one are highly unstable numerically. One has to pay careful attention (in particular for division by zero) when working with such accelerators; it is important to use high precision. The following is a program to accelerate Picard's iteration using the already developed `PicardIteration` program.

```
AcceleratedPicardIteration[Ker_, g_Function, {ap_, y_},
  α_?NumericQ, {i_Integer, n_Integer}] := (
  If[Length[DownValues[ap]] == 0,
   ap[-1, -1] = 0;
   ap[0, 0] = y[0];
   ap[1, m_Integer] := Block[{temp, t},
     PicardIteration[Ker, g, y, α, m + 1];
     temp = (y[m + 1][t] y[m - 1][t] - y[m][t]^2) /
        (y[m + 1][t] - 2 y[m][t] + y[m - 1][t]);
     ap[1, m] = Function @@ List[t, temp]
    ];
   ap[j_Integer, m_Integer] := Block[{temp, t},
     temp =
      (ap[j - 1, m + 1][t] ap[j - 1, m - 1][t] -
         ap[j - 1, m][t]^2) /
       (ap[j - 1, m + 1][t] - 2 ap[j - 1, m][t] +
         ap[j - 1, m - 1][t]);
     ap[j, m] = Function @@ List[t, temp]
    ];
  ];
  ap[i, n]
 )
```

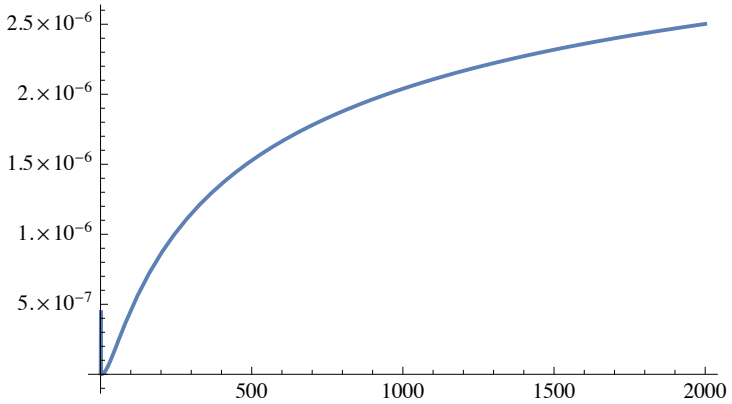
The analytic solution of $y(t) = 1 - \frac{1}{\sqrt{\pi}} \int_0^t \frac{y(s)}{\sqrt{t-s}}\, ds$ is $y(t) = e^t \operatorname{erfc}\left(\sqrt{t}\right)$ (see [3]). Use that to compare with the result of the Aitken accelerator.

```
x1[t_] := x1[t] = Exp[t] Erfc[Sqrt[t]]
```

Define `res1` and plot the error.

```
res1 = AcceleratedPicardIteration[Ker1, g1, {ap1, y1},
   1 / 2, {6, 6}];

Plot[Evaluate[Abs[x1[t] - res1[t]]], {t, 0, 2000}]
```



▲ This graph shows the error in applying the accelerated Picard's method on equation (4). We note that `AcceleratedPicardIteration[Ker1,g1,{ap1,y1},1/2,{6,6}]` uses only the values of $y_1(t)$ to $y_9(t)$ and provides an excellent approximation for $y(t) = e^t \operatorname{erfc}\left(\sqrt{t}\right)$, the exact solution of equation (4).

Equation (7) can be used to find $y_n(t)$, for very large values of n. This equation was obtained as a result of analyzing the corresponding Picard iteration. The same type of analysis will be used for our next nonlinear example.

■ *Method Analysis for Example 2*

Example 2

We solve a nonlinear *blow-up phenomena* model,

$$y(t) = \frac{\sqrt{t}}{2} + \frac{1}{4}\int_0^t \frac{y^2(s) + 2\,y(s)}{\sqrt{t-s}}\,ds. \tag{8}$$

Equation (8) is mentioned as a a blow-up phenomena model on p. 417 of [7]. The analytical solution $y(t)$ exhibits blow-up at finite time. A blow-up means there is a finite time $r > 0$ such that

$$\lim_{t\to r} y(t) = \infty. \tag{9}$$

Our goal is to find the value of r as accurately as possible.

Implementing a few steps of Picard's iteration to equation (8) shows that the solution $y(t)$ is of the form

$$y(t) = \sum_{i=0}^{\infty} b_i\, t^{i/2}, \tag{10}$$

which implies that

$$y^2(t) = \sum_{k=0}^{\infty} t^{k/2} \sum_{j=0}^{k} b_j\, b_{k-j}. \tag{11}$$

Make a change of variable from s to $t\,s$ in equation (8) and replace $y(t)$ and $y^2(t)$ using equations (10) and (11), respectively, on both sides of equation (8). We obtain

$$\sum_{k=0}^{\infty} b_k\, t^{k/2} = \frac{\sqrt{t}}{2} + \frac{1}{4} \sum_{k=0}^{\infty} t^{(k+1)/2} \sum_{j=0}^{k} b_j\, b_{k-j}\, s_k + \frac{1}{2} \sum_{k=0}^{\infty} t^{(k+1)/2}\, b_k\, s_k, \tag{12}$$

where

$$s_k = \int_0^1 \frac{s^{k/2}}{\sqrt{1-s}}\, ds = \sqrt{\pi}\ \Gamma\left(\frac{k+2}{2}\right) \Big/ \Gamma\left(\frac{k+3}{2}\right). \tag{13}$$

Equating coefficients corresponding to equal powers of t from both sides, we get

$$b_0 = 0,\ b_1 = \frac{1}{2},\ b_n = \frac{s_{n-1}}{2}\, b_{n-1} + \frac{s_{n-1}}{4} \sum_{k=0}^{n-1} b_k\, b_{n-k-1},\ n = 2, 3, 4, \ \ldots\,. \tag{14}$$

Note that

$$\underset{n\to\infty}{\text{limit}}\ \frac{b_n}{b_{n+1}} \simeq 0.94745798, \tag{15}$$

and that $y(t^2)$ is a power series in t. Therefore, the radius of convergence for $y(t)$ given by equation (10) is $0.94745798^2 \approx 0.897677$, which is the blow-up number r in equation (9).

All terms in equation (10) are positive, and the series is wildly divergent beyond 0.897677. For a series with all positive terms, the nonlinear accelerators are not useful in evaluating $y(t)$ beyond the radius of convergence. For an alternating series, the situation is different. Nonlinear extrapolations are normally quite effective in evaluating the sums of alternating divergent series for variable values far beyond the radius of convergence.

We evaluate

$$y_n(t) = \sum_{i=0}^{n} b_i\, t^{i/2} \tag{16}$$

for large values of n, where $b_i\,s$ are given by equation (14).

■ *Solution of Example 2*

```
s2[k_] := N[Sqrt[π] Gamma[(k + 2)/2] / Gamma[1 + (k + 1)/2], 100]

bcoeff[s_, b_, n_Integer] :=
 Block[{$MaxExtraPrecision = 100},
  b[0] = 0 / 1;
  b[1] = 1 / 2;
  b[m_Integer] :=
   b[m] = Block[{k, gam},
     gam = (s[m - 1] / 2) b[m - 1] +
        (s[m - 1] / 4) Sum[b[k] b[m - k - 1], {k, 0, m - 1}];
  b[n]]

blowupphenomena[y_, s_, f_Function, g_Function, b_,
  n_Integer] := Block[{$MaxExtraPrecision = 100},
  y[-1] = 0;
  y[0] = f;
  y[1] = g;
  y[m_Integer] := y[m] = Block[{t, temp}, bcoeff[s, b, m];
     temp = y[m - 1][t] + bcoeff[s, b, m] t^(m/2);
     y[m] = Function @@ List[t, temp]];
  y[n]]

g2 = Function[t, Sqrt[t]/2];

f2 = Function[t, 0];
```

For the following tables, the execution time is provided for cases $n = 5000$ and $n = 10\,000$.

```
Table[s2[n], {n, 0, 5000}]; // AbsoluteTiming
```

```
{0.580887, Null}
```

```
Table[s2[n], {n, 0, 10 000}]; // AbsoluteTiming
```

```
{2.43222, Null}
```

```
Table[bcoeff[s2, b2, n], {n, 0, 5000}]; // AbsoluteTiming
```

```
{30.8509, Null}
```

```
Table[bcoeff[s2, b2, n], {n, 0, 10 000}]; //
 AbsoluteTiming
```

```
{104.835, Null}
```

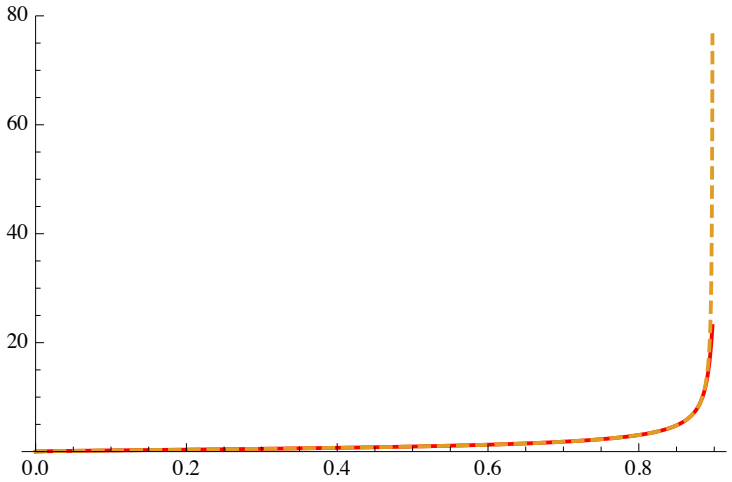
```
Table[blowupphenomena[y2, s2, f2, g2, b2, n],
   {n, 0, 5000}]; // AbsoluteTiming
```

```
{7.93702, Null}
```

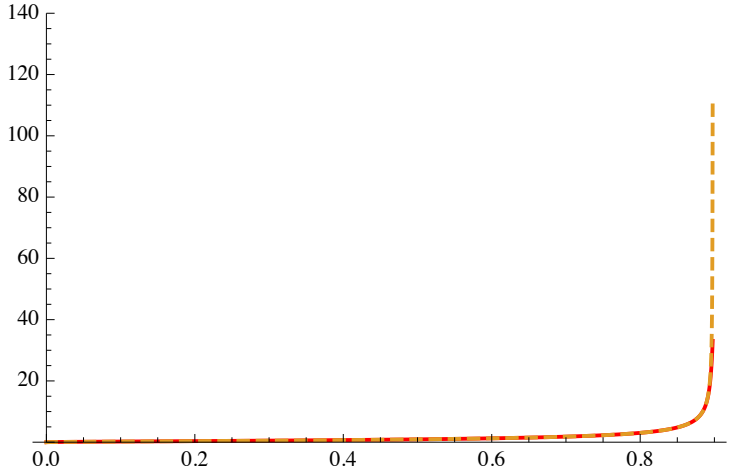
```
Table[blowupphenomena[y2, s2, f2, g2, b2, n],
   {n, 0, 10 000}]; // AbsoluteTiming
```

```
{25.9239, Null}
```

```
Plot[{y2[500][t], y2[5000][t]}, {t, 0, .89767},
 PlotRange → {0, 80}, PlotStyle → {Red, Dashed}]
```




```
Plot[{y2[1000][t], y2[10 000][t]}, {t, 0, 0.89767},
 PlotRange → {0, 140}, PlotStyle → {Red, Dashed}]
```



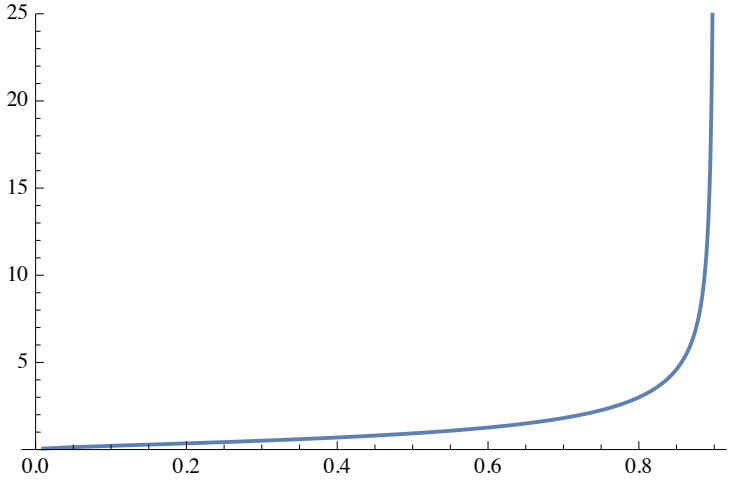
▲ These graphs clearly demonstrate that $\lim_{n\to\infty} y_n(t) \to \infty$, as $t \to 0.897677$.

The following program is for the accelerated Picard method for example 2. All terms in equation (16) are positive, and the acceleration is not as helpful as in example 1, where the series was alternating.

```
AcceleratedPicard2[s_, b_, f_Function, g_Function,
  {ap_, y_}, {i_Integer, n_Integer}] :=
 Block[{$MaxExtraPrecision = 100},
  ap[-1, -1] = 0;
  ap[0, 0] = f;
  ap[1, m_Integer] := ap[1, m] = Block[{temp, t},
     blowupphenomena[y, s, f, g, b, m + 1];
     temp = (y[m + 1][t] y[m - 1][t] - y[m][t]^2) /
       (y[m + 1][t] - 2 y[m][t] + y[m - 1][t]);
     ap[1, m] = Function @@ List[t, temp]];
  ap[j_Integer, m_Integer] := ap[j, m] = Block[{temp, t},
     temp =
      (ap[j - 1, m + 1][t] ap[j - 1, m - 1][t] -
         ap[j - 1, m][t]^2) /
       (ap[j - 1, m + 1][t] - 2 ap[j - 1, m][t] +
         ap[j - 1, m - 1][t]);
     ap[j, m] = Function @@ List[t, temp]];
  ap[i, n]]

res2 = AcceleratedPicard2[s2, b2, f2, g2, {ap2, y2}, {3, 3}];

Plot[Evaluate @@ res2[t], {t, 0.01, 0.8976},
 PlotRange → {0, 25}]
```



▲ `AcceleratedPicard2[s, b, f, g, {ap, y}, {3, 3}]` uses the values of $y_1(t)$ to $y_6(t)$ and yet demonstrates the blow-up phenomena as $t \to 0.89767$.

Numerical Approximation

For most integral equations, Picard iteration is not practical, and we approximate the solution of equation (1) $y(t)$ using a semi-implicit product midpoint rule. To understand the method, we start by subdividing the interval of integration $[0, t]$ into n equal subintervals using a step size h. Equation (1) becomes

$$y(n\,h) = g(n\,h) + \int_0^{n\,h} \frac{K(n\,h, s, y(s))}{(n\,h - s)^{\alpha}}\, ds; \tag{17}$$

note that $y(0) = g(0)$. For $n = 1$, we only have the subinterval $[0, h]$. Let s (the middle variable of the kernel function K) be the midpoint of the interval $[0,\ h]$; that is, $s = h/2$. Let y (the third variable of K) be the midpoint of $y(0)$ and $y(h)$; that is, $y = (y(0) + y(h))/2$, and recall that $y(0) = g(0)$. The denominator of the integral that contains the singularity stays as is, so

$$\begin{aligned} y(h) &\cong g(h) + \int_0^{h} \frac{K(h, h/2, (y(0) + y(h))/2)}{(h - s)^{\alpha}}\, ds = \\ & g(h) + K(h, h/2,\ (y(0) + y(h))/2)) \int_0^{h} \frac{1}{(h - s)^{\alpha}}\, ds = \\ & g(h) + \frac{h^{1-\alpha}}{1 - \alpha} K\left(h, \left(\frac{h}{2}\right), \frac{g(0) + y(h)}{2}\right). \end{aligned} \tag{18}$$

We solve equation (17) for $y(h)$ using Mathematica's built-in function `FindRoot` with an initial guess of $g(0)$. The integral that contains the singularity is solved exactly and therefore does not introduce any inaccuracy in the method, which is why the word "product rule" is added to the name of this technique; see [8] for more details on product integration. Also, at each step only the very last variable (in this case $y(h)$) needs to be found, which is why the name "semi-implicit" is used. We also use the midpoints of the intervals; hence the term "a semi-implicit product midpoint rule." Now let $n = 2$ to get

$$\begin{aligned} y(2\,h) \cong g(2\,h) &+ \int_0^{h} \frac{K(2\,h, h/2, (y(0) + y(h))/2)}{(2\,h - s)^{\alpha}}\, ds + \\ & \int_h^{2\,h} \frac{K(2\,h, 3\,h/2,\ (y(h) + y(2\,h))/2)}{(2\,h - s)^{\alpha}}\, ds. \end{aligned} \tag{19}$$

Or

$$\begin{aligned} y(2\,h) \cong g(2\,h) &+ \frac{h^{1-\alpha}}{1 - \alpha}\left(2^{1-\alpha} - 1^{1-\alpha}\right) K\left(2\,h, \left(\frac{h}{2}\right), \frac{g(0) + y(h)}{2}\right) + \\ & \frac{h^{1-\alpha}}{1 - \alpha} K(2\,h, 3\,h/2, (y(h) + y(2\,h))/2. \end{aligned} \tag{20}$$

Then equation (19) is solved to find an approximation for $y(2\,h)$ using `FindRoot` with an initial guess of $y(h)$ from the previous step. Continuing the same procedure, we arrive at the following program.

- ***A Program for the Semi-implicit Product Midpoint Rule and Its Corresponding Extrapolation to Approximate the Solution of Equation (1)***

```
Midpoint[Ker_, α_, g_, 0, h_] := g[0]

Midpoint[Ker_, α_, g_, n_, h_] :=
 Midpoint[Ker, α, g, n, h] =
  Evaluate[
   x /.
    FindRoot[
     x == g[n h] + Sum[(h^(1-α)/(1-α)) ((n + 1 - i)^(1-α) - (n - i)^(1-α))
          Ker[n h, ((2 i - 1)/2) h,
            1/2 (Midpoint[Ker, α, g, i - 1, h] +
               Midpoint[Ker, α, g, i, h])], {i, 1, n - 1}] +
       h^(1-α)/(1-α) Ker[n h, ((2 n - 1)/2) h,
         (Midpoint[Ker, α, g, n - 1, h] + x)/2],
     {x, Midpoint[Ker, α, g, n - 1, h]}]]

ExtrapolatedMidpoint[Ker_, α_, g_, n_, h_] :=
  (Midpoint[Ker, α, g, 4 n, h/4] Midpoint[Ker, α, g, n, h] -
     (Midpoint[Ker, α, g, 2 n, h/2])^2)/
   (Midpoint[Ker, α, g, 4 n, h/4] - 2 Midpoint[Ker, α, g, 2 n, h/2] +
     Midpoint[Ker, α, g, n, h]);
```

■ *Solution of Example 3*

Example 3

The nonlinear integral equation

$$y(t) = -\frac{1}{\sqrt{\pi}} \int_0^t \frac{(y(s) - \sin(s))^3}{\sqrt{t-s}}\, ds \tag{21}$$

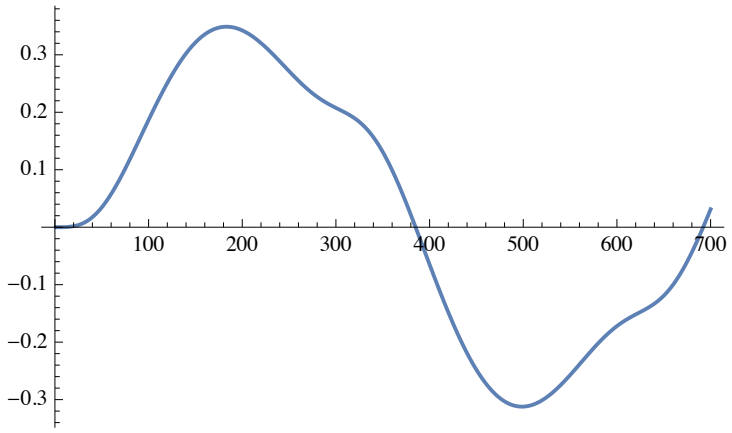
arises in the theory of superfluity [9].

We use the semi-implicit product midpoint rule program to approximate the solution of equation (1).

```
g3[t_] := g3[t] = 0
```

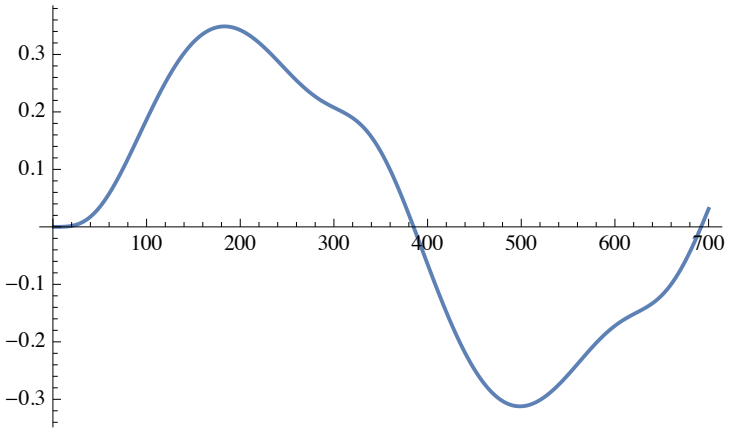
```
Ker3[t_, s_, y_] := -1/Sqrt[π] (y - Sin[s])^3
```

```
ListLinePlot[Table[Midpoint[Ker3, 1 / 2, g3, n, 1 / 100],
  {n, 1, 700}]]
```



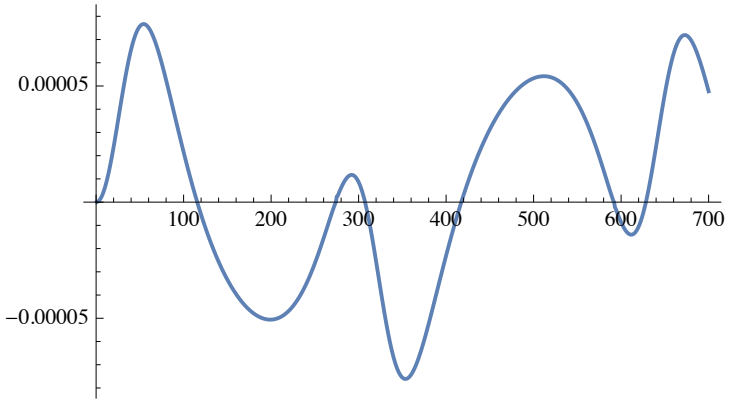
▲ Graph of superfluity equation with no acceleration and a step size of 0.01 and $\alpha = 1/2$.

```
ListLinePlot[
 Table[ExtrapolatedMidpoint[Ker3, 1 / 2, g3, n, 1 / 100],
  {n, 1, 700}]]
```



▲ Graph of superfluity equation with acceleration and a step size of 0.01.

```
ListLinePlot[
 Table[ExtrapolatedMidpoint[Ker3, 1 / 2, g3, n, 1 / 100] -
   Midpoint[Ker3, 1 / 2, g3, n, 1 / 100], {n, 1, 700}]]
```



▲ This graph shows the difference between the accelerated and non-accelerated methods for the superfluity equation. The difference is of order 10^{-5}, which shows that iterating the extrapolation once more is not sensible. The execution time goes up exponentially, and not much is gained in accuracy.

For our next example, we were able to find the corresponding analytical solution. Therefore, we can demonstrate the efficiency of our method by comparing the numerical solution with the exact solution.

Before solving example 4, we prove the following theorem for a class of inhomogeneous equations to find its analytic solution.

Theorem 1

The equation

$$y(t) = g(t) + \lambda \int_0^t \frac{y(s)}{\sqrt{t-s}}\, ds \tag{22}$$

is equivalent to the differential equation

$$y'(t) = \pi \lambda^2 + g'(t) + \lambda\left(\frac{g(0)}{\sqrt{t}} + \int_0^t \frac{g'(s)}{\sqrt{t-s}}\, ds\right), \text{ with } y(0) = g(0). \tag{23}$$

Proof

From the general theory of the Abel equations ([2], p. 224), if

$$f(t) = \int_0^t \frac{\varphi(s)}{\sqrt{t-s}}\, ds, \tag{24}$$

then

$$\varphi(t) = \frac{1}{\pi}\frac{d}{dt}\int_0^t \frac{f(s)}{\sqrt{t-s}}\, ds. \tag{25}$$

Equation (22) can be written as

$$\frac{y(t) - g(t)}{\lambda} = \int_0^t \frac{y(s)}{\sqrt{t-s}}\, ds. \tag{26}$$

Using equations (24), (25), and (26), we get

$$y(t) = \frac{1}{\lambda\pi}\frac{d}{dt}\left(\int_0^t \frac{y(s)}{\sqrt{t-s}}\, ds - \int_0^t \frac{g(s)}{\sqrt{t-s}}\, ds\right). \tag{27}$$

Now on the right side of equation (27), replace the integral $\int_0^t \frac{y(s)}{\sqrt{t-s}}\, ds$ by $\frac{y(t)-g(t)}{\lambda}$ from equation (26) to get the desired result, equation (23).

□

We now use Theorem 1 to find the exact solution of wave propagation for example 4 and test the efficiency of our midpoint rule and its extrapolated version.

Example 4

The equation

$$y(t) = \frac{1}{\sqrt{t}\ e^{a/4\,t}} + \frac{i}{\sqrt{\pi}} \int_0^t \frac{y(s)}{\sqrt{t-s}}\, ds \tag{28}$$

represents wave propagation over a flat surface (see p. 229 and p. 235, exercise 3 of [2]). Using Theorem 1, the exact solution of equation (28) can be obtained as

$$y(t) = i\, e^{-\sqrt{-a}\,-t}\, \sqrt{\pi} + \frac{e^{-\frac{a}{4t}}}{\sqrt{t}} - i\, e^{-i\sqrt{a}\,-t}\, \sqrt{\pi}\ \operatorname{erf}\left(\frac{\sqrt{a} - 2\,i\,t}{2\sqrt{t}}\right). \tag{29}$$

■ *Solution of Example 4*

Solving equation (28) with our midpoint rule and an Aitken extrapolation with an step size $h = 1/10$, we obtain an approximation with a maximum absolute error of order 10^{-6}.

Example 4 is the three-dimensional case. Therefore, the program for the general midpoint rule given for equation (1) must be adjusted a little.

```
MidpointWave[Ker_, a_, α_, g_, 0, h_] := 0

MidpointWave[Ker_, a_, α_, g_, n_, h_] :=
 MidpointWave[Ker, a, α, g, n, h] =
  Evaluate[
   x /.
    FindRoot[
     x == g[n h, a] +
       Sum[(h^(1-α)/(1-α)) ((n + 1 - i)^(1-α) - (n - i)^(1-α))
         Ker[n h, ((2 i - 1)/2) h,
           1/2 (MidpointWave[Ker, a, α, g, i - 1, h] +
               MidpointWave[Ker, a, α, g, i, h])],
        {i, 1, n - 1}] +
       h^(1-α)/(1-α) Ker[n h, ((2 n - 1)/2) h,
         (MidpointWave[Ker, a, α, g, n - 1, h] + x)/2],
     {x, MidpointWave[Ker, a, α, g, n - 1, h]}]]
```

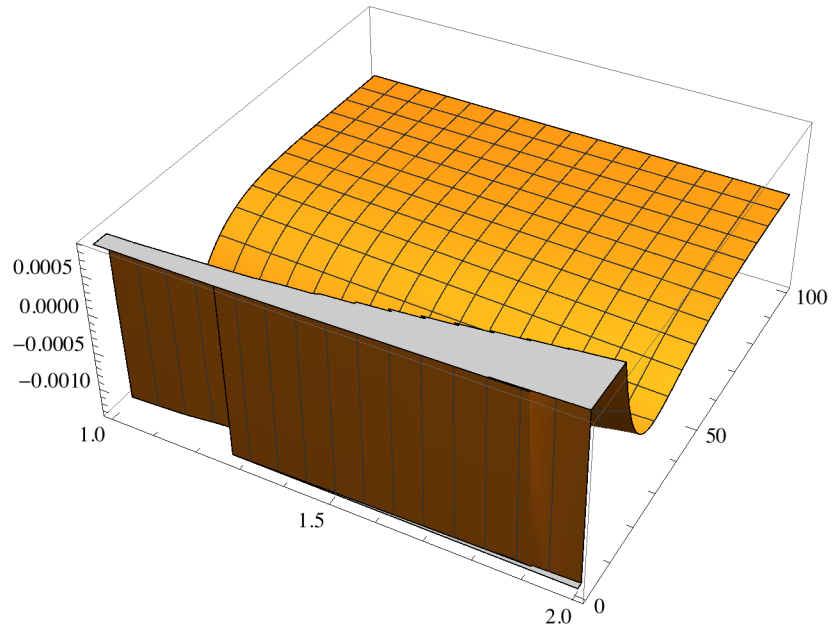
```
ExtrapolatedMidpointWave[Ker_, a_, α_, g_, n_, h_] :=
 (MidpointWave[Ker, a, α, g, 4 n, h/4]
     MidpointWave[Ker, a, α, g, n, h] -
    (MidpointWave[Ker, a, α, g, 2 n, h/2])^2)/
  (MidpointWave[Ker, a, α, g, 4 n, h/4] -
    2 MidpointWave[Ker, a, α, g, 2 n, h/2] +
    MidpointWave[Ker, a, α, g, n, h])
```

$$g4[t\_, a\_] := \frac{1}{\sqrt{t}}\, \text{Exp}\left[\frac{-a}{4\,t}\right]$$

$$\text{Ker4}[t\_, s\_, y\_] := \frac{i}{\sqrt{\pi}}\, y$$

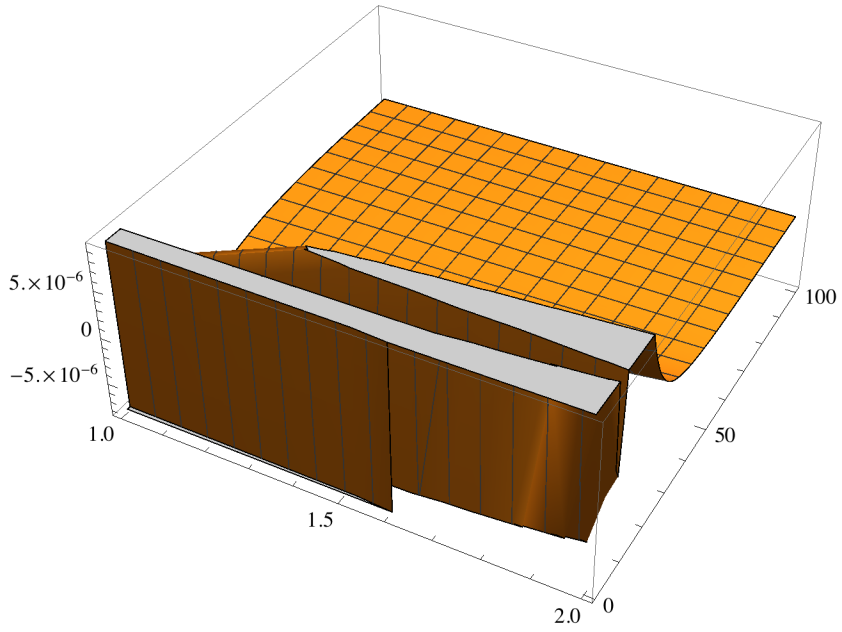
$$f4[t\_, a\_] := i\, e^{-\sqrt{-a}\,t}\sqrt{\pi} + \frac{e^{-\frac{a}{4t}}}{\sqrt{t}} - i\, e^{-i\sqrt{a}\,t}\sqrt{\pi}\, \text{Erf}\left[\frac{\sqrt{a} - 2\,i\,t}{2\sqrt{t}}\right]$$

```
ListPlot3D[
 Table[
  {Re[MidpointWave[Ker4, n / 10, 1 / 2, g4, n, 1 / 10] -
     f4[n / 10, n / 10]],
   Im[MidpointWave[Ker4, n / 10, 1 / 2, g4, n, 1 / 10] -
     f4[n / 10, n / 10]]}, {n, 1, 100}], ImageSize → {450, 350}]
```



▲ This plots the error in applying the midpoint rule with a step size $h = 1/10$; for $0 \le a = n\,h \le 10$ and $0 \le t = n\,h \le 10$, with $n = 1, \ldots, 100$, the maximum absolute error is of order 0.001.

```
ListPlot3D[
 Table[
  {Re[ExtrapolatedMidpointWave[Ker4, n / 10, 1 / 2, g4,
      n, 1 / 10] - f4[n / 10, n / 10]],
   Im[ExtrapolatedMidpointWave[Ker4, n / 10, 1 / 2, g4,
      n, 1 / 10] - f4[n / 10, n / 10]]}, {n, 1, 100}],
 ImageSize → {450, 350}]
```



▲ This plots the error in applying the Aitken extrapolation with a step size $h = 1/10$; for $0 \le a = n\,h \le 10$ and $0 \le t = n\,h \le 10$, $n = 1, \ldots, 100$, the maximum absolute error is of order 10^{-6}.

Conclusion

We studied two methods: Picard's iteration and a semi-implicit product midpoint rule. The Aitken Δ^2 extrapolation was used to accelerate the convergence of these methods. In some cases, the extrapolation demonstrated a significant improvement. A blow-up phenomena model, a wave propagation, and a superfluity equation were solved, and the efficiency and the practicality of the methods were established. The user-friendly programs created here solve the general equations. One only needs to enter a forcing function, a kernel function, the α value, and a step size h for a particular problem. We used Mathematica 10.2 on the Mac OS X operating system with 16 GB RAM and a 2.8 GHz processor. The execution time was always under five minutes, and the vast majority of problems were executed in less than five seconds.

Acknowledgments

I am grateful for constructive suggestions by a reviewer, resulting in more transparent coding.

Dedication

The author dedicates his work to Mahshid, Arman, and Ida; wife, son, and daughter.

References

[1] W. Hackbusch, *Integral Equations: Theory and Numerical Treatment*, Boston: Birkhäuser Verlag, 1995.

[2] R. P. Kanwal, *Linear Integral Equations*, 2nd ed., Boston: Birkhäuser, 1997.

[3] R. K. Miller, *Nonlinear Volterra Integral Equations*, Menlo Park, CA: W. A. Benjamin, Inc., 1971.

[4] A. Pipkin, *A Course on Integral Equations*, New York: Springer-Verlag, 1991.

[5] R. Kress, *Linear Integral Equations*, Berlin: Springer-Verlag, 1989.

[6] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 2nd ed., New York: Springer, 1996.

[7] H. Brunner, *Collocation Methods for Volterra and Related Functional Equations*, Cambridge: Cambridge University Press, 2004.

[8] P. Linz, *Analytical and Numerical Methods for Volterra Equations*, Philadelphia: SIAM, 1985.

[9] N. Levinson, "A Nonlinear Volterra Equation Arising in the Theory of Superfluidity," *Journal of Mathematical Analysis and Applications*, **1**(1), 1960 pp. 1–11. doi:10.1016/0022-247X(60)90028-7.

J. Abdalkhani, "Exact and Approximate Solutions of the Abel–Volterra Equations," *The Mathematica Journal*, 2016. dx.doi.org/doi:10.3888/tmj.18-2.

About the Author

Javad Abdalkhani is an associate professor of mathematics at the Ohio State University, Lima campus and a Distinguished Alumni teacher at the Ohio State University. His area of research is numerical analysis. His hobbies are reading and cycling.

Javad Abdalkhani
Department of Mathematics
The Ohio State University, Lima
4240 Campus Drive
Lima, Ohio 45804
abdalkhani.1@osu.edu