Advanced Analytical Techniques for the Solution of Single- and Multi-Dimensional Integral Equations

David Keffer
Department of Chemical Engineering
University of Tennessee, Knoxville
August 1999

Table of Contents

Introduction

This hand-out is a primer on integral equations. On the analytical side, it assumes that the reader knows calculus, differential equations, and the basic operations of linear algebra. On the numerical side, it assumes that the reader knows the trapezoidal rule for numerical integration and the Newton-Raphson method of root-finding in nonlinear algebraic equations.

1. Definition of Integral equations

What is an integral equation?

An integral equation (IE) is an equation in which an unknown function appears within an integral, just as a differential equation is an equation in which an unknown function appears within a derivative. Just as the solution to a differential equation is a function, so too is the solution to an integral equation a function.

2. Ordinary versus partial integral equations

When dealing with differential equations, we encounter ordinary differential equations (ODEs) and partial differential equations (PDEs). Are there analogous ordinary integral equations (OIEs) and partial integral equations (PIEs)?

Yes. There are integral equations in which the integration is carried out with respect to a single variable (OIEs) and there are integral equations in which the integration is carried out with respect to multiple variables (PIEs). However, in the mathematics of integral equations, they are not referred to as OIEs and PIEs. The jargon is different. However, for our purposes, we might as well call them OIEs and PIEs, since we understand that terminology.

An example of an ordinary integral equation:

$$\phi(x) = f(x) + \lambda \int_a^x N(x,y)\phi(y)dy \tag{1}$$

An example of a partial integral equation:

$$\phi(x_1,x_2) = f(x_1,x_2) + \lambda \int_{a_2}^{x_2}\int_{a_1}^{x_1} N(x_1,x_2,y_1,y_2)\phi(y_1,y_2)dy_1dy_2 \tag{2}$$

Hopefully you see the analogy between ODEs and PDEs with OIEs and PIEs.

3. Linearity versus nonlinearity of integral equations

Do integral equations come in linear and nonlinear flavors?

Yes. An integral equation is called linear if linear operations are performed in it upon the unknown function, that is, if it has the form

$$A(x)\phi(x) + B(x) + \lambda \int_a^x N(x,y)\phi(y)dy = 0 \tag{3}$$

Here is an example of a nonlinear integral equation

$$\phi(x) = f(x) + \lambda \int_a^x F[x, y, \phi(y)]dy \tag{4}$$

where $F[x, y, \phi(y)]$ is a nonlinear function of $\phi(y)$, for example

$$F[x, y, \phi(y)] = \sin[\phi(y)] \tag{5}$$

4. Single integral equations vs. systems of integral equations

Do integral equations appear in systems as well as individually.

You bet they do. All the time.

5. Categorization of integral equations

We have seen that when dealing with algebraic equations, we used only the linear/nonlinear categorization. When dealing with ODEs, we could expand the categorization to include first order/second order, homogeneous/nonhomogeneous, separable/nonseparable, etc. When dealing with PDEs, we categorized them as elliptic, hyperbolic, and parabolic. How do mathematicians categorize integral equations?

Mathematicians categorize ordinary integral equations in much the same way that ODEs are classified by small sub-groups, for which a particular solution method exists. Thus, the categorization does not apply to all ODEs (the way that the linear/nonlinear categorization does).

Linear OIEs are divided into two basic types:

Volterra integral equations, in which the upper limit of integration is variable,

$$\phi(x) = f(x) + \lambda \int_a^x N(x, y)\phi(y)dy \tag{6}$$

and Fredholm integral equations, in which the limits of integration are fixed,

$$\phi(x) = f(x) + \lambda \int_a^b N(x, y)\phi(y)dy \tag{7}$$

Both Volterra and Fredholm integral equations can be subdivided into two groups: equations of the first and second kind. In equations (6) and (7), we have written Volterra and Fredholm integral equations of the second kind. Volterra and Fredholm integral equations of the first kind have the form, respectively,

$$f(x) = \lambda \int_a^x N(x, y)\phi(y)dy \tag{8}$$

and Fredholm integral equations, in which the limits of integration are fixed,

$$f(x) = \lambda \int_a^b N(x,y)\phi(y)dy \tag{9}$$

These categorizations apply to both linear and nonlinear OIEs.

6. Kernels of integral equations

What is a kernel?

In equations (6) to (9), the function $N(x,y)$ is called the **kernel** of the integral equation. Every integral equation has a kernel. Kernels are important because they are at the heart of the solution to integral equations.

7. Analytical solutions to integral equations

Example 1. We cannot possibly present analytical solutions to al of the types of integral equations. However we can present a few examples. Let us consider the Volterra equation of the second kind.

$$\phi(x) = f(x) + \lambda \int_a^x N(x,y)\phi(y)dy \tag{6}$$

Certain assumptions have to be made about $N(x,y)$ being defined and bounded over the domain, as well as $f(x)$ being a Riemann-integrable function in the domain. Also, $\phi(x)$ is an arbitrary Riemann-integrable function in the domain.

The first step in the solution of this integral equation is a transformation called the Dirichlet transformation. In the Dirichlet transformation, we state that the order of integration doesn't matter.

$$\int_p^q \left[\int_a^b M(x,y)dx \right] dy = \int_a^b \left[\int_p^q M(x,y)dy \right] dx \tag{10}$$

Let us define a function

$$M_x(x,y) = N(x,y)N(y,s)\phi(s) \tag{11}$$

Substituting equation (11) into (10), we have

$$\int_a^x \left[\int_a^x M_x(s,y)ds \right] dy = \int_a^x \left[\int_a^x M_x(s,y)dy \right] ds \tag{12}$$

$$\int_a^x \left[\int_a^y N(x,y)N(y,s)\phi(s)ds \right] dy = \int_a^x \left[\int_s^x N(x,y)N(y,s)dy \right] \phi(s)ds \tag{13}$$

We can now proceed to the solution of the Volterra equation. If there exists an integrable function $\phi(x)$ which satisfies the Volterra equation (6), then this function also satisfies the so-called iterated equation, obtained by substituting the right hand side of equation (6) under the integral sign of equation (6) obtaining.

$$\phi(x) = f(x) + \lambda \int_a^x N(x,y) \left[f(x) + \lambda \int_a^y N(y,s)\phi(s)ds \right] dy \tag{14}$$

For the moment, let us proceed under the assumption that this substitution is valid. When we have completed the transformation, we will verify the assumption.

We now apply the Dirichlet Transformation to equation (14) obtaining the iterated equation

$$\phi(x) = f(x) + \lambda \int_a^x N(x,y) f(y) dy + \lambda^2 \int_a^x N_1(x,y)\phi(y) dy \tag{15}$$

where $N_1(x,y)$ is called the iterated kernel and is written as

$$N_1(x,y) = \int_y^x N(x,s)N(s,y)ds \tag{16}$$

Repeating the above transformation, we obtain a two-fold iterated equation

$$\phi(x) = f(x) + \lambda \int_a^x [N(x,y) + \lambda N_1(x,y)] f(y) dy + \lambda^3 \int_a^x N_2(x,y)\phi(y) dy \tag{17}$$

where $N_2(x,y)$ is called the two-fold iterated kernel and is written as

$$N_2(x,y) = \int_y^x N(x,s)N_1(s,y)ds \tag{18}$$

After the nth repetition of the transformation we have:

$$\phi(x) = f(x) + \lambda \int_a^x \left[N(x,y) + \sum_{i=1}^{n-1} \lambda^i N_i(x,y) \right] f(y) dy + \lambda^{n+1} \int_a^x N_n(x,y)\phi(y) dy \tag{19}$$

where $N_n(x,y)$ is called the n-fold iterated kernel and is written as

$$N_n(x,y) = \int_y^x N(x,s)N_{n-1}(s,y)ds \tag{20}$$

Intricate proofs have been performed which show that if $N(x,y)$ was defined and bounded over the domain all of the i-fold iterated kernels are also defined and bounded over the domain of integration [Pogorzelski].

The standard procedure is to rewrite equation (19) as

$$\phi(x) = f(x) + \lambda \int_a^x \aleph(x,y,\lambda) f(y) dy \tag{21}$$

where $\aleph(x, y, \lambda)$ is called the resolvent kernel and is written as

$$\aleph(x, y, \lambda) = N(x, y) + \sum_{i=1}^{\infty} \lambda^i N_i(x, y) \tag{20}$$

Now this looks like we cheated in the substitution because, with this resolvent kernel, equation (21) ought to be

$$\phi(x) = f(x) + \lambda \int_a^x \aleph(x, y, \lambda) f(y) dy + \lambda^{n+1} \int_a^x N_\infty(x, y)[\phi(y) - f(y)] dy \tag{22}$$

but we have omitted the last term entirely. Therefore, we must show that it is important, which, fortunately, mathematicians have already done [Pogorzelski, p. 11].

To check that our solution in equation (21) satisfies the Volterra equation (6), substitute (21) into (6). We need to do this because we made an assumption moving from equation (13) to equation (14) that has not yet been proved.

$$\phi(x) = f(x) + \lambda \int_a^x N(x, y) \left[f(y) + \lambda \int_a^y \aleph(y, s, \lambda) f(s) ds \right] dy \tag{23}$$

$$\phi(x) = f(x) + \lambda \int_a^x N(x, y) f(y) dy + \lambda^2 \int_a^x \left[\int_a^y N(x, y) \aleph(y, s, \lambda) f(s) ds \right] dy \tag{24}$$

We change the order of integration for y and s

$$\phi(x) = f(x) + \lambda \int_a^x N(x, y) f(y) dy + \lambda^2 \int_a^x \left[\int_s^x N(x, y) \aleph(y, s, \lambda) dy \right] f(s) ds \tag{25}$$

We transpose the dummy variables y and s

$$\phi(x) = f(x) + \lambda \int_a^x N(x, y) f(y) dy + \lambda^2 \int_a^x \left[\int_y^x N(x, s) \aleph(s, y, \lambda) ds \right] f(y) dy \tag{26}$$

$$\phi(x) = f(x) + \lambda \int_a^x \left[N(x, y) + \lambda \int_y^x N(x, s) \aleph(s, y, \lambda) ds \right] f(y) dy \tag{27}$$

The term in brackets in equation (27) can be rewritten, using the definition of the resolvent kernel in equation (20)

$$N(x, y) + \lambda \int_y^x N(x, s) \aleph(s, y, \lambda) ds = \aleph(x, y, \lambda) \tag{28}$$

So that we can rewrite equation (27) as

$$\phi(x) = f(x) + \lambda \int_a^x \aleph(x, y, \lambda) f(y) dy \tag{29}$$

which is what we have in equation (21), which we substituted into the integral equation for checking. So, our check is complete. A Volterra equation of the second kind has one and only one bounded solution, given by the formula in equation (29). This solution is convergent for all values of λ.

This solution technique is also valid for partial Volterra equations of the second kind,

$$\phi(x_1, x_2) = f(x_1, x_2) + \lambda \int_{a_2}^{x_2} \int_{a_1}^{x_1} N(x_1, x_2, y_1, y_2) \phi(y_1, y_2) dy_1 dy_2 \tag{2}$$

Example 2. Let us consider the Volterra equation of the first kind

$$f(x) = \lambda \int_a^x N(x, y) \phi(y) dy \tag{8}$$

Using Leibnitz's rule for the differentiation of integrals:

$$\frac{d}{d\alpha} \int_{\phi_1(\alpha)}^{\phi_2(\alpha)} F(x, \alpha) dx = \int_{\phi_1(\alpha)}^{\phi_2(\alpha)} \frac{dF(x, \alpha)}{d\alpha} dx + F(\phi_2, \alpha) \frac{d\phi_2}{d\alpha} - F(\phi_1, \alpha) \frac{d\phi_1}{d\alpha} \tag{30}$$

we can differentiate both sides of equation (8), effectively transforming the Volterra equation of the first kind into a Volterra equation of the second kind:

$$\phi(x) = \frac{f'(x)}{N(x, x)} + \frac{\lambda}{N(x, x)} \int_a^x N'(x, y) \phi(y) dy \tag{31}$$

which is a Volterra equation of the second kind. If $N(x, x) = 0$, we can differentiate again to obtain a transformation with $N'(x, x)$ in the denominator.

Example 3. Fredholm's integral equation of the second kind.

Fredholm's integral equation of the second kind has the form

$$\phi(x) = f(x) + \lambda \int_a^b N(x, y) \phi(y) dy \tag{7}$$

The solution to this equation is

$$\phi(x) = f(x) + \lambda \int_a^b \aleph(x, y, \lambda) f(y) dy \tag{32}$$

where $\aleph(x,y,\lambda)$ is called the resolvent kernel and is written as

$$\aleph(x,y,\lambda) = \sum_{i=1}^{\infty} \lambda^{i-1} N_i(x,y) \tag{33}$$

where $N_1(x,y) = N(x,y)$.

Example 4. Let's work a Volterra equation of the second kind with numbers.

$$\phi(x) = e^x - \int_0^x e^{x-y}\phi(y)dy \tag{34}$$

so $f(x) = e^x$, $\lambda = -1$, $N(x,y) = e^{x-y}$, and $a = 0$.

$$N_1(x,y) = \int_y^x N(x,s)N(s,y)ds = \int_y^x e^{x-s}e^{s-y}ds = \int_y^x e^{x-y}ds = e^{x-y}(x-y)$$

$$N_2(x,y) = \int_y^x N(x,s)N_1(s,y)ds = \int_y^x e^{x-s}e^{s-y}(s-y)ds = \int_y^x e^{x-y}(s-y)ds$$

$$N_2(x,y) = e^{x-y}\left(\frac{x^2}{2} - yx - \frac{y^2}{2} + y^2\right) = e^{x-y}\left(\frac{x^2}{2} - yx + \frac{y^2}{2}\right)$$

$$N_3(x,y) = \int_y^x N(x,s)N_2(s,y)ds = \int_y^x e^{x-s}e^{s-y}\left(\frac{s^2}{2} - ys + \frac{y^2}{2}\right)ds = \int_y^x e^{x-y}\left(\frac{s^2}{2} - ys + \frac{y^2}{2}\right)ds$$

$$N_3(x,y) = e^{x-y}\left[\left(\frac{x^3}{6} - \frac{yx^2}{2} + \frac{y^2x}{2}\right) - \left(\frac{y^3}{6} - \frac{yy^2}{2} + \frac{y^2y}{2}\right)\right] = e^{x-y}\left[\frac{x^3}{6} - \frac{yx^2}{2} + \frac{y^2x}{2} - \frac{y^3}{6}\right]$$

Let's truncate the resolvent kernel at the third iterated kernel because things are getting ugly fast.

$$\aleph(x,y,\lambda) = N(x,y) + \sum_{i=1}^{\infty} \lambda^i N_i(x,y) \tag{20}$$

$$\aleph(x,y,\lambda) = e^{x-y} - e^{x-y}(x-y) + e^{x-y}\left(\frac{x^2}{2} - yx + \frac{y^2}{2}\right) - e^{x-y}\left[\frac{x^3}{6} - \frac{yx^2}{2} + \frac{y^2x}{2} - \frac{y^3}{6}\right]$$

$$\aleph(x,y,\lambda) = e^{x-y}\left[1 - x + y + \frac{x^2}{2} - yx + \frac{y^2}{2} - \frac{x^3}{6} + \frac{yx^2}{2} - \frac{y^2x}{2} + \frac{y^3}{6}\right]$$

Now, let's create the solution using both the first, second, and third approximations to the resolvent kernel

$$\phi_1(x) = e^x - \int_0^x e^{x-y}[1-x+y]e^y dy = e^x - e^x \int_0^x [1-x+y]dy = e^x - e^x \left[y - xy + \frac{y^2}{2} \right]_0^x$$

$$\phi_1(x) = e^x \left\{ 1 - \left[x - xx + \frac{x^2}{2} \right]_0^x + \left[0 - x0 + \frac{0^2}{2} \right] \right\} = e^x \left\{ 1 - x + \frac{x^2}{2} \right\}$$

$$\phi_2(x) = e^x - \int_0^x e^{x-y} \left[1 - x + y + \frac{x^2}{2} - yx + \frac{y^2}{2} \right] e^y dy = e^x \left\{ 1 - \int_0^x \left[1 - x + y + \frac{x^2}{2} - yx + \frac{y^2}{2} \right] dy \right\}$$

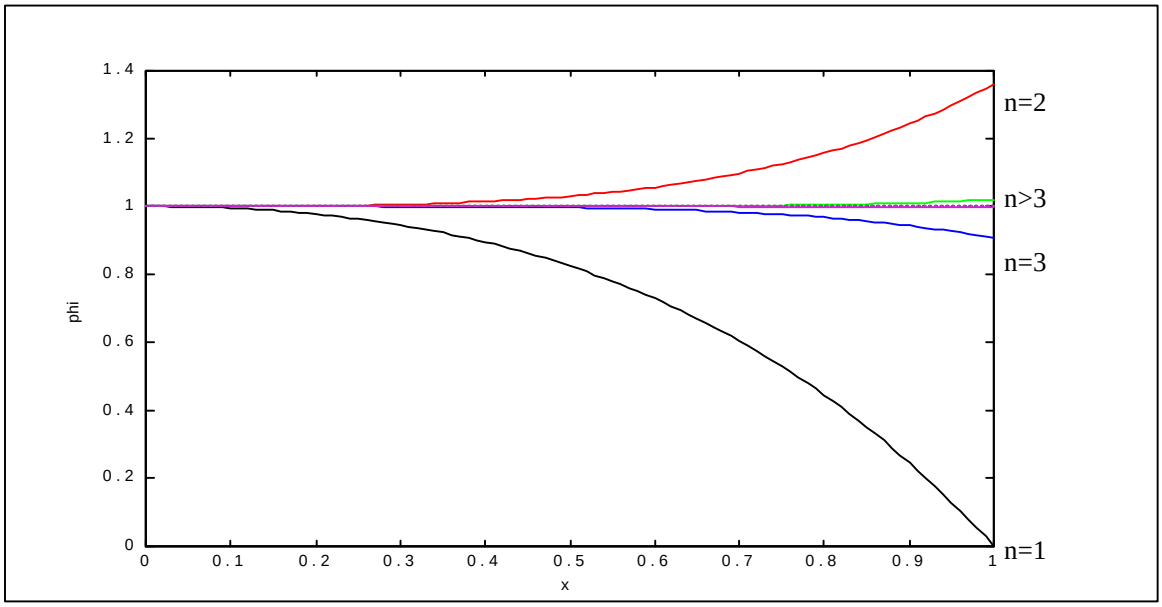
$$\phi_2(x) = e^x \left\{ 1 - \left[y - xy + \frac{y^2}{2} + \frac{x^2}{2} y - \frac{y^2 x}{2} + \frac{y^3}{6} \right]_0^x \right\} = e^x \left\{ 1 - x + \frac{x^2}{2} - \frac{x^3}{6} \right\}$$

$$\phi_3(x) = e^x - \int_0^x e^{x-y} \left[1 - x + y + \frac{x^2}{2} - yx + \frac{y^2}{2} - \frac{x^3}{6} + \frac{yx^2}{2} - \frac{y^2 x}{2} + \frac{y^3}{6} \right] e^y dy$$

$$\phi_3(x) = e^x \left\{ 1 - x + \frac{x^2}{2} - \frac{x^3}{6} + \frac{x^4}{24} \right\}$$

We can pretty much guess what the infinite solution looks like:

$$\phi_\infty(x) = e^x \sum_{i=0}^{\infty} \frac{(-x)^i}{i!}$$



We can plot the analytical solution, varying the number of iterated kernels that we include in the resolvent kernel. The analytical solution is phi(x) = 1.0. On the following page is the MATLAB code used to generate this plot.

```
function volterra2_analytic
%
%  Solution to a Volterra Equation of the Second Kind
%
%  The Volterra Equations have four parameters
%
%  the lower limit of integration, a
%  the constant prefactor outside the integral, lam
%  the function outside the integral, f, given in a function at
the bottom of this file
%  the kernel, kernelorig, given in a function at the bottom of
this file
%
%  This code gives teh analytic solution to a=0, lam=-1, f=e=^x
and N=e^(x-y)
%
%  Author:  David Keffer, University of Tennessee, Department
of Chemical Engineering
%
clf
time_01 = cputime;
a = 0;
lam = -1;
%
%  set up a grid of x values
%
xstart = 0.0;
xf = 1.0;
dx = 0.01;
nx =(xf-xstart)/dx + 1;
x = zeros(nx,1);
for i = 1:1:nx
   x(i) = xstart + (i-1)*dx;
end
%
%  let's run different runs where we truncate the resolvent
%  kernel at different points just to see the effect
%
nruns = 10;
ncutoff=[1;2;3;4;5;6;7;8;9;10]';
phi = zeros(nx,nruns);
%
for j = 1:1:nruns
%
%  calculate and plot purely analytic phi
%
    for i = 1:1:nx
      xi = x(i);
      for k = 0:1:ncutoff(j)
         phi(i,j) = phi(i,j) + (-xi)^k/factorial(k);
      end
      phi(i,j) = phi(i,j)*f(xi);
   end
   %
   % plot
   %
    if (j==1)
     plot (x,phi(:,j),'k-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (j==2)
     plot (x,phi(:,j),'r-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (j==3)
     plot (x,phi(:,j),'b-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (j==4)
     plot (x,phi(:,j),'g-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (j==5)
     plot (x,phi(:,j),'m-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (j==6)
     plot (x,phi(:,j),'k:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (j==7)
     plot (x,phi(:,j),'r:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (j==8)
     plot (x,phi(:,j),'b:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (j==9)
     plot (x,phi(:,j),'g:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (j==10)
     plot (x,phi(:,j),'m:'), xlabel( 'x' ), ylabel ( 'phi' )
    else
     plot (x,phi(:,j),'k-'), xlabel( 'x' ), ylabel ( 'phi' )
    end
    hold on
    fprintf (1,'FINISHED CALCULATING THE %2i RUN where ncutoff =
%4i \n', j, ncutoff(j));
end
hold off
phi

function y = f(x)
y = exp(x);

function y = factorial(x)
    fact = 1;
    for i = 1:1:x
    fact = fact*i;
   end
    y = fact;
```

8. Numerical solutions to linear integral equations

One can imagine different types of numerical solutions to a linear integral equation like the Volterra equation. Frequently we are going to encounter integral series, the which we have no intention of analytically evaluating either because we don't know how to evaluate the integrals analytically or it's not worth our time. Faced with this problem, we could code the analytical solution above, using numerical methods to numerically evaluate all of the integrals required for the kernel, the iterated kernels, and finally for phi. For example, we could use the trapezoidal rule to evaluate all of the iterated kernels in equation (20) (up to however many terms we want to use) and use the trapezoidal rule again to evaluate the solution for phi as given in equation (21). No two ways about it, THIS IS COMPUTATIONALLY INTENSIVE. It results in the evaluation of *each* iterated integral n^2 times where n is the number of points along our axis. So if we want to evaluate phi at 100 x-values, using 5 iterated kernels, we will be forced to evaluate 50,000 integrals! This sucks. If the equation is nonlinear, we may be forced to do this.

A second, much more elegant method is to take advantage of the linearity of the equation. We can use a numerical method for the evaluation of the integrals but plug this method directly into the integral equation. Consider Volterra's equation of the second kind

$$\phi(x) = f(x) + \lambda \int_a^x N(x,y)\phi(y)dy \tag{6}$$

and consider the trapezoidal rule for n intervals:

$$\int_a^b g(y)dy = \frac{b-a}{n}\left[\frac{f(a)}{2} + \sum_{j=2}^{n} f(y_j) + \frac{f(b)}{2}\right] \tag{35}$$

If we substitute equation (35) in for the integral in equation (36) we have

$$\phi(x_i) = f(x_i) + \lambda h\left[\frac{N(x_i,a)\phi(a)}{2} + \sum_{j=2}^{i-1} N(x_i,y_j)\phi(y_j) + \frac{N(x_i,x_i)\phi(x_i)}{2}\right] \tag{37}$$

$$-\lambda h\frac{N(x_i,a)}{2}\phi(a) - \lambda h\sum_{j=2}^{i-1} N(x_i,y_j)\phi(y_j) + \left[1-\lambda h\frac{N(x_i,x_i)}{2}\right]\phi(x_i) = f(x_i) \tag{38}$$

for i = 1, equation (38) becomes, where $x_1 = a$

$$\phi(a) = f(a) \tag{39}$$

and for i = 2, we have

$$-\lambda h\frac{N(x_2,x_1)}{2}\phi(x_1) + \left[1-\lambda h\frac{N(x_2,x_2)}{2}\right]\phi(x_2) = f(x_2) \tag{40}$$

and for i = 3, we have

$$-\lambda h\frac{N(x_3,x_1)}{2}\phi(x_1) - \lambda h N(x_3,x_2)\phi(x_2) + \left[1-\lambda h\frac{N(x_3,x_3)}{2}\right]\phi(x_3) = f(x_3) \tag{41}$$

Now we have n equations of type equation (37), where $x_1 = a$ and $x_{n+1} = b$. If you look at these equations, you will see that they form a set of linear algebraic equations! Whoa boy! We know how to solve that. The unknowns n unknowns are the function evaluated at each of our n points.

Our problem is going to look like this:

$$\underline{\underline{A}}\,\underline{\phi} = \underline{b} \tag{42}$$

where

$$\underline{\phi} = \left[\phi(x_1 = a)\phi(x_2), \phi(x_3)\ldots\phi(x_i)\cdots\phi(x_n), \phi(x_{n+1})\right]^T \tag{43}$$

$$A_{off,i,j} = -\lambda h N(x_i, y_j) \tag{44}$$

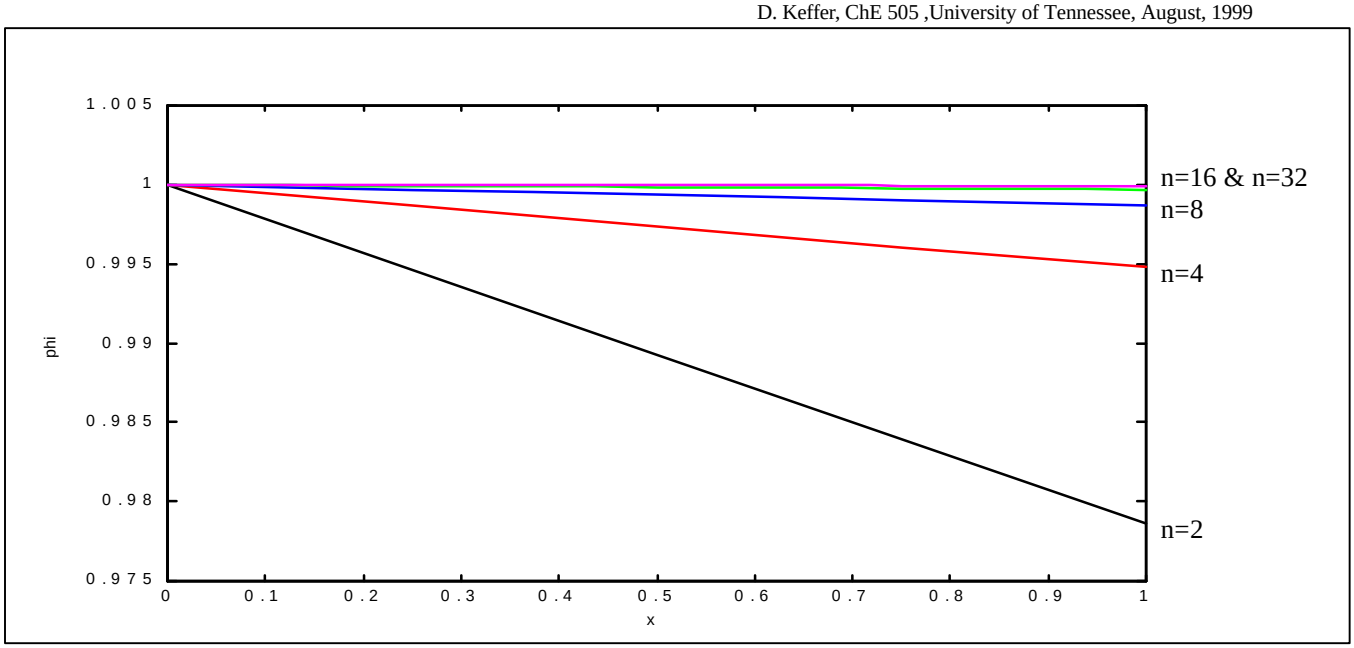
$$A_{diag,i} = \lambda h\left(1 - \frac{N(x_i, x_i)}{2}\right) \tag{45}$$

$$\underline{\underline{A}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0.5A_{off,2,1} & A_{diag,2} & 0 & 0 & 0 & 0 \\ 0.5A_{off,3,1} & A_{off,3,2} & A_{diag,3} & 0 & 0 & 0 \\ 0.5A_{off,i,1} & A_{off,i,2} & A_{off,i,3} & A_{diag,i} & 0 & 0 \\ 0.5A_{off,n,1} & A_{off,n,2} & A_{off,n,3} & A_{off,n,i} & A_{diag,n} & 0 \\ 0.5A_{off,n+1,1} & A_{off,n+1,2} & A_{off,n+1,3} & A_{off,n+1,i} & A_{off,n+1,n} & A_{diag,n+1} \end{bmatrix} \tag{46}$$

and where

$$\underline{b} = \begin{bmatrix} f(x_1 = a) \\ f(x_2) \\ f(x_3) \\ f(x_i) \\ f(x_n) \\ f(x_{n+1}) \end{bmatrix} \tag{47}$$

Now, the accuracy of the solution depends on the step-size. As an illustration, we solve the same problem that we solved analytically in the last section. This time we solve it with this method for various values of the step-size.



Clearly as the number of intervals, n, increases, the accuracy of our solution to the integral equation also increases.

In a result that should not surprise us, Linz has shown that should one use better methods for numerical integration than the trapezoidal rule, one will obtain more accurate results.

```
function volterra2_analytic
%
%  Solution to a Volterra Equation of the Second Kind
%
%  The Volterra Equations have four parameters
%  the lower limit of integration, a
%  the constant prefactor outside the integral, lam
%  the function outside the integral, f, given in a function at
the bottom of this file
%  the kernel, kernelorig, given in a function at the bottom of
this file
%
%  This code gives the numeric solution to a=0, lam=-1, f=e=^x
and N=e^(x-y)
%  Author:  David Keffer, University of Tennessee,
%  Department of Chemical Engineering
clf
time_01 = cputime;
a = 0;
lam = -1;
%
%  set up a grid of x values
%
xstart = 0.0;
xf = 1.0;
%
%  let's run different runs where we change the step size
%  just to see the effect
%
nruns = 5;
nintervals=[2;4;8;16;32;64;128;256;512;1024]';
npoints = nintervals+1;
for jj = 1:1:nruns
   %
   %  set up the x grid
   %
   ninterval = nintervals(jj);
   npoints = ninterval+1;
   phi = zeros(npoints,1)';
   dx = (xf-xstart)/ninterval;
    x = zeros(npoints,1);
    for i = 1:1:npoints
    x(i) = xstart + (i-1)*dx;
    end
    %
    %  calculate and plot purely analytic phi
   %
   amat = zeros(npoints,npoints);
    bnum = zeros(npoints,1);
    fac = lam*dx;
    for i = 1:1:npoints
    xi = x(i);
    if (i == 1)
        amat(i,i) = 1.0;
    else
        amat(i,i) = 1.0  - fac*0.5*kernelorig(xi,xi) ;
    end
    bnum(i) = f(xi);
    for j = 1:1:npoints
        yj = x(j);
        if (j < i)
            if (j == 1)
                amat(i,j) = - fac*0.5*kernelorig(xi,yj);
            else
                amat(i,j) = -fac*kernelorig(xi,yj);
            end
        end
    end
    end
    amatinv = inv(amat);
    phi = amatinv*bnum
% plot
    if (jj==1)
     plot (x,phi(:),'k-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==2)
     plot (x,phi(:),'r-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==3)
     plot (x,phi(:),'b-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==4)
     plot (x,phi(:),'g-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==5)
     plot (x,phi(:),'m-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==6)
     plot (x,phi(:),'k:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==7)
     plot (x,phi(:),'r:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==8)
     plot (x,phi(:),'b:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==9)
     plot (x,phi(:),'g:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==10)
     plot (x,phi(:),'m:'), xlabel( 'x' ), ylabel ( 'phi' )
    else
     plot (x,phi(:),'k-'), xlabel( 'x' ), ylabel ( 'phi' )
    end
    hold on
    fprintf (1,'FINISHED CALCULATING THE %2i RUN where
nintervals = %10f \n', jj, ninterval);
end
hold off

function y = f(x)
y = exp(x);

function y = kernelorig(x,y)
y = exp(x-y);
```

9. Numerical solutions to nonlinear integral equations

There are many ways to skin a cat. There are at least as many ways to solve nonlinear integral equations. I will speak briefly on two methods: (1) an extension to the linear method of Section 8, and (2) the method of successive approximations.

Consider a general nonlinear integral equation

$$\phi(x) = f(x) + \lambda \int_a^x F[x, y, \phi(y)]dy \tag{4}$$

where $F[x, y, \phi(y)]$ is a nonlinear function of $\phi(y)$, for example

$$F[x, y, \phi(y)] = \sin[\phi(y)] \tag{5}$$

First, in Section 8, we showed how to reduce a linear integral equation to a system of linear algebraic equations, which we then know how to solve. It should come as no surprise that we can reduce a nonlinear integral equation into a system of nonlinear algebraic equations, which we also know how to solve, although it frequently is a pain to do so. Still, if we go through the trouble to code it up once, then we have it for time immemorial, so long as there continue to be FORTRAN compilers. Since FORTRAN will surely outlast us, we are alright.

We can discretize the x-axis into n intervals. Thus we obtain n+1 unknowns $\phi(x_i)$ for i = 1 to n+1. If we are smart, we choose to discretize the y-axis (the dummy axis of integration) in the exact same way that we discretized the x-axis. Then, we can write something like, again using Trapezoidal rule,

$$\phi(x_1) = f(x_1 = a) \tag{48}$$

$$\phi(x_2) = f(x_2) + \lambda \frac{h}{2}\left[F[x_2, x_1, \phi(x_1)] + F[x_2, x_2, \phi(x_2)]\right] \tag{49}$$

$$\phi(x_3) = f(x_3) + \lambda \frac{h}{2}\left[F[x_3, x_1, \phi(x_1)] + 2F[x_3, x_2, \phi(x_2)] + F[x_3, x_3, \phi(x_3)]\right] \tag{50}$$

$$\phi(x_i) = f(x_i) + \lambda \frac{h}{2}\left[F[x_i, x_1, \phi(x_1)] + 2\sum_{j=2}^{j=i-1} F[x_i, x_j, \phi(x_j)] + F[x_i, x_i, \phi(x_i)]\right] \tag{51}$$

This is a system of nonlinear algebraic equations. We could solve this using any technique in our arsenal for solving systems of nonlinear algebraic equations. This case is particularly amenable to the Newton-Raphson method since all of the partial derivatives in the Jacobian are going to have the same functional form. Remember the derivatives are taken with respect to $\phi(x_i)$ and not with respect to x_i.

The second method for solving a nonlinear integral equation is the method of successive approximations. Again, referring to Pogorzelski [page 192], we see that equation (4) can be solved using the recursive relation

$$\phi_i(x) = f(x) + \lambda \int_a^x F[x, y, \phi_{i-1}(y)]dy \tag{52}$$

You repeat this procedure until $\left|\phi_{i+1}(x)-\phi_i(x)\right|$ is acceptably small. The only additional information needed for this method is the starting point

$$\phi_0(x)=f(x) \tag{53}$$

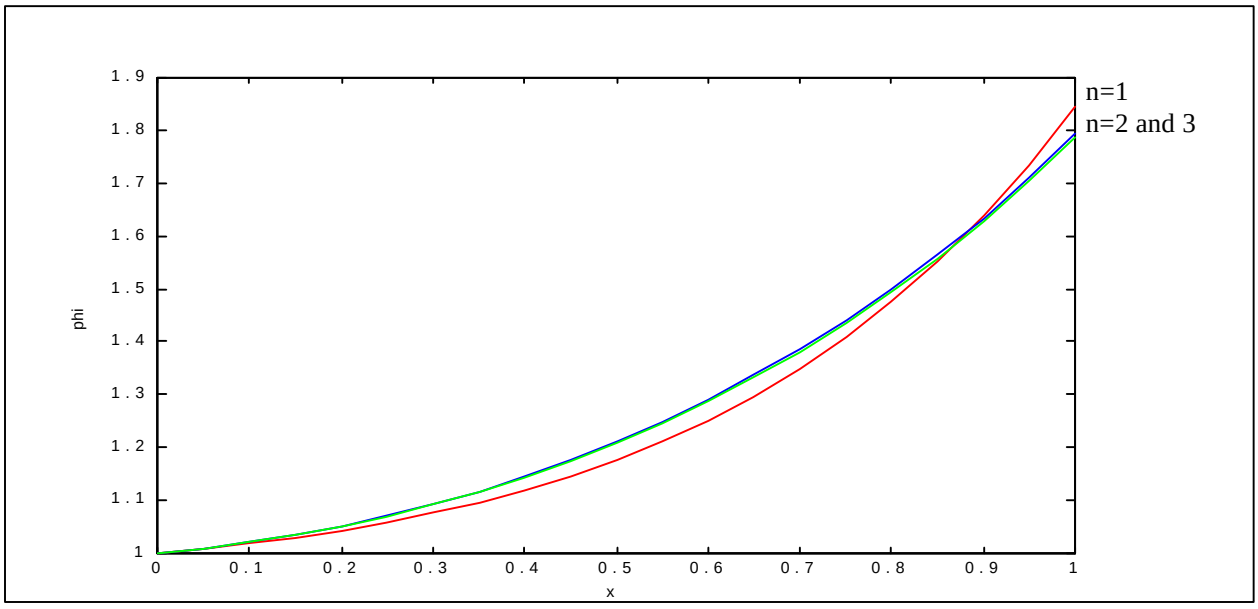
Now, certain qualifications have to be made about when this method works. The solution has to satisfy the Lipschitz condition and various other criteria have to be met. Well, at this point, I wave my hands and leave this to the mathematicians. We are engineers and we are here to use the equations. If we try and use it and it doesn't converge, then we go to the other method of solutions! Mathematicians please forgive us ignorant and sacrilegious engineers.

Now, if the truth be told, $\phi_0(x)$ can be any arbitrary continuous function in our range of interest. We set it to f(x) just for the heck of it. You could just as easily set it to unity.

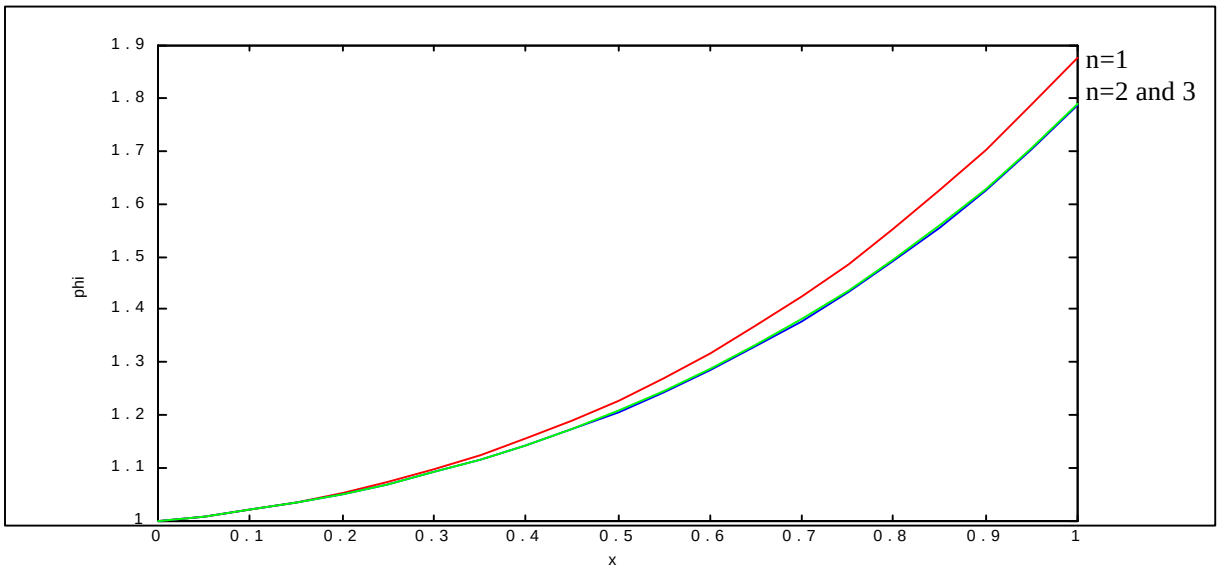
Let's work an example and solve it three ways, using (1) an extension to the linear method of Section 8, and (2) the method of successive approximations with $\phi_0(x)=f(x)$ and (3) the method of successive approximations with $\phi_0(x)=1.0$.

$$\phi(x)=e^x-\int_0^x \sin[\phi(y)]dy \tag{54}$$

Here we plot the solution to equation (54) using successive approximations.



In this figure, n is the number of iterations performed. In this case, $\phi_0(x)=f(x)=e^x$.



In this figure, n is the number of iterations performed. In this case, $\phi_0(x) = 1.0$. From this we can see that, for this particular equation, the approximation converges pretty dang quickly.

It is left up to the student to verify these results using the other method, namely that of using a method for solving a system of nonlinear algebraic equations.

Below is the MATLAB code used to implement the successive approximations solution method.

```
function volterra2_nonlinear
%
%  Solution to a Nonlinear Volterra Equation of the Second Kind
%
%  The Volterra Equations have four parameters
%
%  the lower limit of integration, a
%  the constant prefactor outside the integral, lam
%  the function outside the integral, f, given in a function at
the bottom of this file
%  the kernel, kernelorig, given in a function at the bottom of
this file
%  This code gives the numeric solution to a=0, lam=-1, f=e=^x
and K=sin(phi)
%
%  Author:  David Keffer, University of Tennessee, Department
of Chemical Engineering
%
clf
a = 0;
lam = -1;
%
%  set up a grid of x values
%
xstart = 0.0;
xf = 1.0;
%
%  USING SUCCESSIVE APPROXIMATIONS
%
ninterval = 20;
npoints = ninterval+1;
dx = (xf-xstart)/ninterval;
x = zeros(npoints,1);
for i = 1:1:npoints
   x(i) = xstart + (i-1)*dx;
end
nruns = 3;
ncutoff=[1;2;3;4;5;6;7;8;9;10]';
phi = zeros(npoints,ncutoff(nruns))';
%
%  initialize phi(:,1)
%
for i = 1:1:npoints
%   phi(i,1) = f(x(i));
   phi(i,1) = 1.0;
end
%
%  start iteration loop
%
jrun = 1;
for jj = 2:1:ncutoff(nruns)+1
   jjm1 = jj-1;
   dxh = 0.5*dx;
   kernphi1 = kern(phi(1,jjm1));
   integral_mid = 0.0;
   phi(1,jj) = f(x(1));
   for i = 2:1:npoints
      integral_mid = integral_mid + kern(phi(i,jjm1));
      integral = dxh*( kernphi1 + 2.0*integral_mid -
kern(phi(i,jjm1)) );
    phi(i,jj) = f(x(i)) + lam*integral;
    end
   %
   % plot
   %
   if ( jjm1 == ncutoff(jrun) )
      jrun = jrun + 1;
    if (jj==1)
     plot (x,phi(:,jj),'k-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==2)
     plot (x,phi(:,jj),'r-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==3)
     plot (x,phi(:,jj),'b-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==4)
     plot (x,phi(:,jj),'g-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==5)
     plot (x,phi(:,jj),'m-'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==6)
     plot (x,phi(:,jj),'k:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==7)
     plot (x,phi(:,jj),'r:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==8)
     plot (x,phi(:,jj),'b:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==9)
     plot (x,phi(:,jj),'g:'), xlabel( 'x' ), ylabel ( 'phi' )
    elseif (jj==10)
     plot (x,phi(:,jj),'m:'), xlabel( 'x' ), ylabel ( 'phi' )
    else
     plot (x,phi(:,jj),'k-'), xlabel( 'x' ), ylabel ( 'phi' )
    end
    hold on
    fprintf (1,'THE %2i RUN where nintervals = %4i and iteration
= %4i, phi(%5.2f) = %5.2f \n', jjm1, ninterval, ncutoff(jrun-
1), x(npoints), phi(npoints,ncutoff(jrun))  );
end
end
hold off

function y = f(x)
y = exp(x);

function y = kern(x,y)
y = sin(x);
```

10. Numerical solutions to systems of integral equations

We know that the formalism for solving systems of equations is often precisely the same as that for solving a single equation. We have seen this again and again. With algebraic equations, we can use the Newton-Raphson method to solve a single non-linear algebraic equation. Similary, we can use the multivariate Newton-Raphson method to solve a system of non-linear algebraic equations. With ODEs, we use Runge-Kutta to solve 1 ODE and multivariate Runge-Kutta to solve a system of ODEs. With PDEs, the same trend holds. With IEs the same trend still holds.

For linear integral equations, we reduced a single IE to a system of n linear algebraic equations, where n was the number of intervals used in our discretization of the x-axis. For a system of linear integral equations, of the form:

$$\phi_i(x) = f_i(x) + \lambda_i \int_a^x \left[\sum_{j=1}^{m} N_{i,j}(x,y)\phi_j(y) \right] dy \tag{55}$$

where i ranges from 1 to m we then have a system of mn equations, where n is again the number of intervals used in our discretization of the x-axis, and m is the number of integral equations.

I stop here because the derivation of the method is totally analogous to what was given above.

For nonlinear equations, we can use successive approximation on a system of nonlinear IEs just as easily as we used it on a single nonlinear IE. We have to simultaneously successively approximate all m $\phi_i(x)$ based on the previous m, but so what? That's a piece of cake. The extension to systems requires no new intellectual effort whatsoever.