Integral Equations

Stan Richardson

We show how *Mathematica* can be used to obtain numerical solutions of integral equations by exploiting a combination of iteration and interpolation. The efficacy of the method is demonstrated by considering three classical integral equations of applied mathematics: Love's equation for the condenser problem, Theodorsen's equation associated with conformal mapping, and Nekrasov's equation arising in the theory of water waves. The success of the approach depends on the use of an appropriate method for the interpolation.

■ Introductory Example

As a first example, we consider the integral equation discussed by Kress [1, 2], namely

$$f(x) = e^{-x} - \frac{1}{2} + \frac{1}{2} e^{-(x+1)} + \frac{1}{2} \int_0^1 (x+1) e^{-xy} f(y)\, dy,$$

which has the exact solution $f(x) = e^{-x}$. Kress quotes errors only at the points $x = 0$, 0.25, 0.5, 0.75, and 1, that at $x = 1$ always being the largest. Among the numerical methods and computations he describes for this example, this largest error is least for Nyström's method using Simpson's rule with 16 intervals, when it is 8.6×10^{-7}.

A classical method of solving such problems is to use iteration. Beginning with some suitable initial function $f_0(x)$, we construct a sequence of functions by

$$f_{n+1}(x) = e^{-x} - \frac{1}{2} + \frac{1}{2} e^{-(x+1)} + \frac{1}{2} \int_0^1 (x+1) e^{-xy} f_n(y)\, dy.$$

For this example, the iterative scheme converges for any continuous $f_0(x)$, and we choose to take $f_0(x) = 1$, setting this as

```
In[1]:= approxsoln[x_] = 1;
```

For comparison purposes, we also have the exact solution

```
In[2]:= exactsoln[x_] = e^-x;
```

and can keep track of the error using

```
In[3]:= error[x_] := exactsoln[x] - approxsoln[x]
```

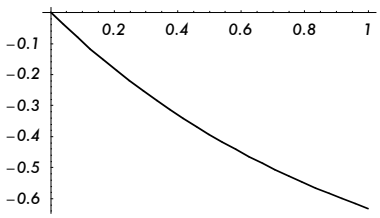
It is useful to have a visual record of this error, so we introduce

```
In[4]:= errorplot := Plot[error[x], {x, 0, 1}, PlotRange → All]
```

At this stage, our approximate solution is not very impressive.

In[5]:=
```
errorplot
```

From In[5]:=



With this example, it is theoretically possible to obtain analytic expressions for the functions $f_n(x)$, but, as this is not feasible for more general examples, we use a numerical approach and adopt the following procedure. With $f_n(x)$ known, we compute $f_{n+1}(x)$ from the iteration formula at a number of points in the interval (0, 1), and make $f_{n+1}(x)$ an `InterpolatingPolynomial` fitted through these points. As a first try, we use a uniform distribution of points on (0, 1), and take just 11 such points. The table of values is given by

In[6]:=
```
values = Table[{x, E^-x - 1/2 + E^(-x-1)/2 + 1/2
        NIntegrate[(x + 1) E^-x y approxsoln[y], {y, 0, 1}]}, {x, 0, 1, 1/10}];
```

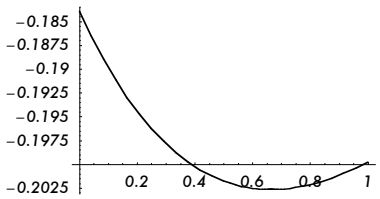
and the improved approximation is then

In[7]:=
```
approxsoln[x_] = InterpolatingPolynomial[values, x];
```

As our plot shows, we now have a better approximation.

In[8]:=
```
errorplot
```

From In[8]:=



We can combine all this into one step.

In[9]:=
```
iterstep := (values =
    Table[{x, E^-x - 1/2 + E^(-x-1)/2 + 1/2 NIntegrate[(x + 1) E^-x y approxsoln[y],
        {y, 0, 1}]}, {x, 0, 1, 1/10}];
  approxsoln[x_] = InterpolatingPolynomial[values, x])
```

Now a little experimentation with `iterstep` and `errorplot` allows us to observe the convergence, but, once done, we know a suitable number of steps to take to obtain a satisfactory answer. For this first example, we demonstrate the convergence graphically in order to illustrate a number of features of our method. To accomplish this, we introduce `errorvalue`, which estimates the maximum error at each iteration from the adaptively-determined `PlotRange` of `errorplot`.

```
In[10]:= errorvalue := Max[Abs[
           Block[{$DisplayFunction = Identity}, Last[PlotRange[errorplot]]]]]
```

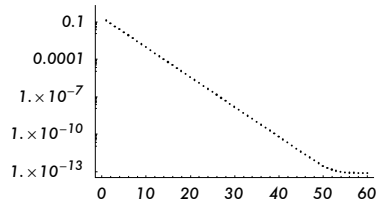
The convergence is nicely illustrated using `LogListPlot`, for which we need to load a package. In fact, other functions from the `Graphics`` package will be required later, so we use

```
In[11]:= << Graphics`
```

We now perform 60 further iterations and plot the result.

```
In[12]:= errorlist = Table[iterstep; errorvalue, {60}]; LogListPlot[errorlist]
```

From In[12]:=

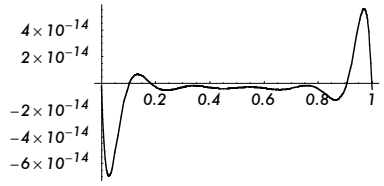


It is clear that we have now converged to an acceptable solution and that further iterations will produce no significant improvement. The remaining error is due solely to the interpolation process we have adopted: we have the best solution to our problem that can be represented in this form. As this makes clear, it is important that an appropriate interpolation routine be used. It is an instructive exercise to try using `Interpolation` with the present example, for this produces results that are vastly inferior, and later examples will require the use of more ingenuity in this respect.

We can show the distribution of the error in the solution we have obtained.

```
In[13]:= errorplot
```

From In[13]:=



When comparing with the results recorded by Kress, remember that he quotes errors only at five specific points.

The error distribution here is typical of that obtained when using a uniform distribution of interpolation points: the maximum errors occur near the ends of the interval. This feature can be eliminated and the maximum error reduced by using a nonuniform distribution of interpolation points that places more of them near the ends. Here, we can achieve this via a nonlinear scaling of the x-axis.

```
In[14]:= new[x_] := Sin[π x / 2]^2
```

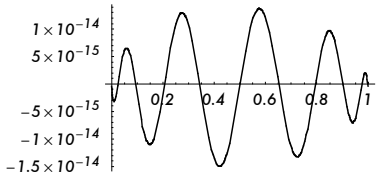
Modifying `iterstep` to incorporate this, we now use

```
In[15]:= iterstep :=
          (values = Table[{new[x], e^-new[x] - 1/2 + 1/2 e^-new[x]-1 + 1/2 NIntegrate[
                (new[x] + 1) e^-new[x] y approxsoln[y], {y, 0, 1}]}, {x, 0, 1, 1/10}];
           approxsoln[x_] = InterpolatingPolynomial[values, x])
```

and find that a few more iterations have indeed improved matters.

```
In[16]:= Do[iterstep, {5}]; errorplot
```

```
From In[16]:=
```



■ The Condenser Problem

An example discussed by Fox and Goodwin [3] in an early paper concerned with the numerical solution of integral equations is the equation derived by Love [4] to describe the electrostatic field produced by a condenser consisting of two parallel circular plates. This is

$$f(x) = 1 + \frac{1}{\pi} \int_{-1}^{1} \frac{\kappa}{\kappa^2 + (x - y)^2} f(y)\, dy$$

when dimensionless variables are taken so that the plates have unit radius. Here κ is the distance between the plates. This is the relevant equation when the potentials of the plates are equal in magnitude but opposite in sign; if these are equal in both magnitude and sign, the first + sign on the right-hand side becomes a − sign.

Of particular interest in this problem is the *capacitance* of the condenser, which is the total charge on one of the plates, given in this nondimensional formulation by

$$\text{capacitance} = \frac{1}{\pi}\int_{-1}^{1} f(x)\,dx.$$

Beginning again with

```
In[17]:= approxsoln[x_] = 1;
```

we now use

```
In[18]:= new[x_] := Sin[π x / 2]
```

and

```
In[19]:= iterstep :=
          (values = Table[{new[x], 1 + 1/π NIntegrate[κ approxsoln[y] / (κ^2 + (new[x] - y)^2),
                {y, -1, new[x], 1}, MaxRecursion → 20]}, {x, 0, 1, 1/6}];
           temp[x_] = InterpolatingPolynomial[Union[values,
              (values /. {x_?NumericQ, y_?NumericQ} → {-x, y})], x];
           approxsoln[x_] = 1/2 (temp[x] + temp[-x]))
```

Here we have taken advantage of the fact that the solution $f(x)$ is an even function of x, so we need not compute entries in `values` for $x < 0$, and used a construction that ensures that our approximation is also an even function. There has also been a modest increase in the degree of the `InterpolatingPolynomial` employed.

We will judge convergence by the estimate we obtain for the capacitance. In fact, most numerical work actually quotes values for the capacitance multiplied by the factor $\pi/2$. This is convenient because a circular disc of unit radius on its own has a capacitance of $2/\pi$, so the difference between the value of this quantity and 1 is a measure of the interference effect between the two discs. This scaled capacitance tends to 1 as $\kappa \to \infty$. We therefore introduce

```
In[20]:= scaledcap := 1/2 NIntegrate[approxsoln[x], {x, -1, 1}]
```

Fox and Goodwin [3] carried out their calculations for $\kappa = 1$, and we will begin with this value, too.

```
In[21]:= κ = 1;
```

A little experimentation with both the interpolation used and the number of iterations performed now tells us that we have convergence to 6 significant figures with

```
In[22]:= Do[iterstep, {20}]; scaledcap
Out[22]= 1.82078
```

Fox and Goodwin [3] do not evaluate the capacitance, but this was calculated for $\kappa = 1$, both from the figures given by Fox and Goodwin and independently, by Cooke [5], who gives the value 1.8208 for the scaled capacitance.

With this solution in hand, we can plot the equipotentials round the condenser. This was done manually by Love [6], but *Mathematica* makes the task somewhat easier. With cylindrical polar coordinates (r, θ, z) chosen so that a plate at potential $+1$ lies in the plane $z = +\kappa/2$ and one at potential -1 is in the plane $z = -\kappa/2$, both in the region $r < 1$, the potential V is independent of θ and is given by

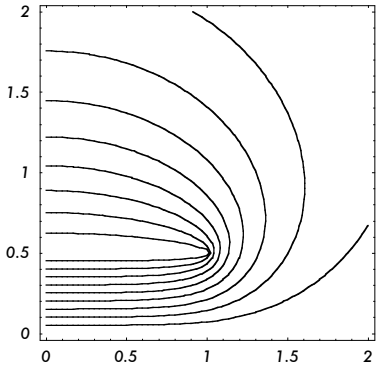
```
In[23]:= V[r_, z_] :=
  Which[z == 0, 0, z == κ/2 && Abs[r] ≤ 1, 1, z == -κ/2 && Abs[r] ≤ 1, -1, True,
   Chop[1/π NIntegrate[(1/Sqrt[r^2 + (z - κ/2 + i x)^2] - 1/Sqrt[r^2 + (z + κ/2 + i x)^2])
     approxsoln[x], {x, -1, 1}, MaxRecursion → 20]]]
```

In fact, with `approxsoln` as a polynomial, *Mathematica* can perform the integrations here analytically, but a numerical approach seems preferable. We have also incorporated the known values of the potential on the plates and the plane $z = 0$ into this definition.

We can economize on the effort required to produce a reasonable sketch by exploiting the symmetry and plotting initially only in the first quadrant, where $r > 0$ and $z > 0$. We should also choose `PlotPoints` to ensure that `ContourPlot` evaluates the potential at the point on the edge of the disk.

```
In[24]:= contplot = ContourPlot[Evaluate[V[r, z]],
           {r, 0, 2}, {z, 0, 2}, ContourShading → False,
           Contours → Range[0.1, 0.9, 0.1], PlotPoints → 61]
```

From In[24]:=



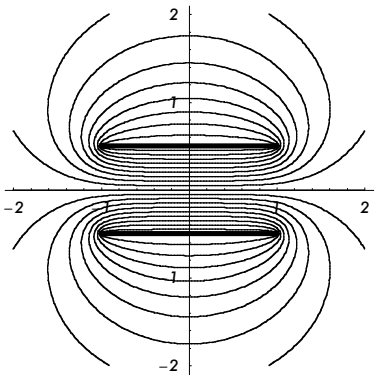
Now extract the information we need from this, and form the data required for the curves in the other quadrants.

```
In[25]:= lines1 = Cases[Graphics[contplot], _Line, ∞];
         lines2 = lines1 /. {x_?NumericQ, y_?NumericQ} → {-x, y};
         lines3 = lines1 /. {x_?NumericQ, y_?NumericQ} → {-x, -y};
         lines4 = lines1 /. {x_?NumericQ, y_?NumericQ} → {x, -y};
```

Finally, we can `Show` all this, together with the lines omitted from `contplot`.

```
In[29]:= Show[Graphics[{lines1, lines2, lines3,
           lines4, Line[{{-2, 0}, {2, 0}}], AbsoluteThickness[2],
           Line[{{-1, κ/2}, {1, κ/2}}], Line[{{-1, -κ/2}, {1, -κ/2}}]}],
         AspectRatio → Automatic, Axes → True]
```

From In[29]:=



Whatever method one uses to solve the integral equation of this example, the computations become more critical as κ decreases. Cooke [5] takes his computations down to $\kappa = 0.4$, but records a value for $\kappa = 0.1$ obtained by Fox and Blake using the methods of Fox and Goodwin [3]; for this value of κ they find a scaled capacitance of 9.233. Using our approach, we need more iterations, but otherwise there are no problems.

```
In[30]:= κ = 0.1; Do[iterstep, {150}]; scaledcap
Out[30]= 9.23285
```

Since this value is within 0.01 percent of that given by the asymptotic formula, valid for $\kappa \to 0$, obtained by Chew and Kong [7], namely

```
In[31]:= approxscaledcap[κ_] := π/(4 κ) + 1/4 (Log[16 π/κ] - 1) + κ/(16 π) (Log[16 π/κ]^2 - 2)
In[32]:= approxscaledcap[0.1]
Out[32]= 9.23194
```

there is little practical need to carry out the computations with the integral equation for smaller values of κ. (See Chew and Kong's paper and the references therein for an account of the history of this formula; it is the product of over a century of effort by a large number of mathematicians.) However, *Mathematica* will do this if required. It is obviously sensible to reorganize the computations

presented in this rather discursive fashion into a more efficient program, but the main changes required are in the interpolation. For nonsmall κ, the graph of $f(x)$ is close to a parabola and `InterpolatingPolynomial` is appropriate, but as κ decreases the gradients become large near the ends of the interval $(-1, 1)$ and interpolation by neither polynomials nor splines is appropriate; `Fit` with suitable basis functions is more effective. Since this modification will be employed later in a different example, we will not illustrate it here.

It should be noted that the familiar "rule of thumb" criteria for stopping a sequence of iterations (e.g., continue iterating until the difference between two consecutive iterates is less than some predetermined small value) become more and more inadequate as κ decreases, because the operator involved approaches the identity as $\kappa \to 0$. This feature is independent of the means we use to represent our solution: the effective application of our method in dealing with nontrivial problems depends on some judicious experimentation with the method of interpolation employed and the number of iterations used.

Fabrikant [8] tabulates many values for the scaled capacitance, down to $\kappa = 0.001$, using a different method based on integral equations that produces upper and lower bounds. However, an asymptotic approach yields results of much greater accuracy for such values of κ. Fabrikant also quotes an error of approximately 0.01 percent for $\kappa = 0.1$, but his error estimates inevitably increase as κ decreases, while the asymptotic formula becomes *more* accurate for smaller values of κ.

■ Conformal Mapping

Many different methods exploiting integral equations have been developed to determine numerically the function giving the conformal map of the unit disk onto some given simply-connected domain; for an account of many of these, see Henrici [9]. As an example of an integral equation that is both nonlinear and singular, we consider the one introduced by Theodorsen for this purpose.

Let the unit disk reside in the plane of $z = r\, e^{i\theta}$ so its boundary is given by $r = 1$, and let the domain onto which we wish to map this disk reside in the plane of $\zeta = \rho\, e^{i\phi}$. We assume that the origin lies within this domain and that its boundary is given in polar form by some known function $\rho = \rho(\phi)$. The map we seek takes points on the boundary of the disk in the z-plane (identified by the angle θ) onto points on the boundary of the domain in the ζ-plane (identified by the angle ϕ). The problem reduces to finding the dependence between these two angles; once we have determined the function $\phi = \phi(\theta)$, the mapping itself can easily be constructed.

If we normalize the map by requiring that the origins in the two planes correspond, and that the map induces no rotation at the origin, the function $\phi(\theta)$ satisfies Theodorsen's integral equation

$$\phi(\theta) = \theta + \frac{1}{2\pi} \int_0^{2\pi} \log(\rho(\phi(\tau))) \cot\left(\frac{\theta - \tau}{2}\right) d\tau.$$

The integral here is to be interpreted as a Cauchy principal value, there being a singularity at $\tau = \theta$. (It is possible to transform this equation to a form that is technically nonsingular (see Kythe [10], for example), but the removable singularity that results involves the computation of values as the ratio of small quantities, even if one manages to avoid $0/0$. It is generally preferable to keep the singularity and deal with it correctly, rather than disguise it and hope for the best.) We solve this equation using the iteration

$$\phi_{n+1}(\theta) = \theta + \frac{1}{2\pi}\int_0^{2\pi} \log(\rho(\phi_n(\tau)))\cot\left(\frac{\theta-\tau}{2}\right)d\tau.$$

Since an increase of 2π in θ must yield an increase of 2π in ϕ, a sensible approximate solution with which to begin the iteration is

```
In[33]:= approxsoln[θ_] = θ;
```

This is also the exact solution if, and only if, the domain in the ζ-plane is a circular disk centered on the origin.

As a first example, we will consider a domain bounded by a lemniscate of Booth (called an inverted ellipse by some authors because it is the inverse of an ellipse with respect to a circle), for which we know that the exact mapping function $\zeta = f(z)$ is given by

```
In[34]:= f[z_] := 2 p z / (1 + p + (1 - p) z^2)
```

Here p is a parameter satisfying $0 < p \le 1$. For $p = 1$ the domain is a circular disk of unit radius, while as $p \to 0$ the domain degenerates to two circular discs of radius $1/2$ touching tangentially. We will consider the particular case

```
In[35]:= p = 0.6;
```

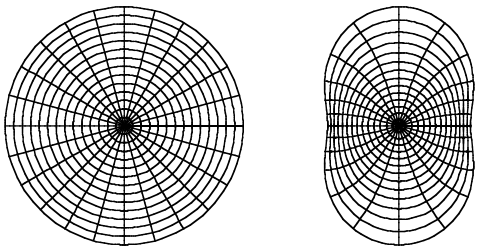
We can see how the mapping distorts the unit disk using `PolarMap` from the `Graphics`ComplexMap`` package that is already available to us.

```
In[36]:= originaldisk = PolarMap[Identity, {0, 1}, {0, 2 π}, Lines → {15, 25},
           PlotRange → All, Axes → False, DisplayFunction → Identity];

In[37]:= distortion = PolarMap[f, {0, 1}, {0, 2 π}, Lines → {15, 25},
           PlotRange → All, Axes → False, DisplayFunction → Identity];

In[38]:= DisplayTogetherArray[{originaldisk, distortion}]
```

From In[38]:=



We see that we have nonnegligible distortion, for the image domain is not convex. For this domain the function $\rho(\phi)$ is

```
In[39]:= ρ[ϕ_] := Sqrt[1 - (1 - p^2) Cos[ϕ]^2]
```

and we can calculate that the exact solution $\phi(\theta)$ is

```
In[40]:= exactsoln[θ_] := ArcTan[p Tan[θ]]
```

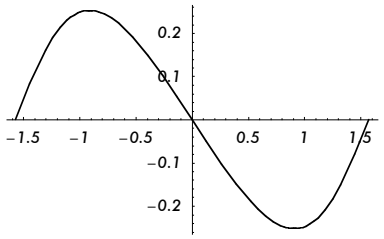
for $-\pi/2 < \theta < \pi/2$; the restriction on the range here is necessary because it is not the principal value of the inverse tangent function that is relevant outside this range. (Using the two-variable form `ArcTan[Cos[θ], p Sin[θ]]` instead gives a formula that is correct for $-\pi < \theta < \pi$, but the symmetry makes such an extension irrelevant here.) As before, we can introduce a plot to enable us to keep track of the error.

```
In[41]:= errorplot :=
          Plot[exactsoln[θ] - approxsoln[θ], {θ, -π/2, π/2}, PlotRange → All]
```

Our initial approximation is not very accurate.

```
In[42]:= errorplot
```

From In[42]:=



We need another package to deal with the Cauchy principal value.

```
In[43]:= << NumericalMath`CauchyPrincipalValue`
```

Because the domain onto which we are mapping is symmetric about both axes in the ζ-plane, the function $\phi(\theta)$ must be the sum of θ and a function that is an odd function of both θ and $\theta - \pi/2$. It is natural to approximate this function using a linear sum of functions of the form $\sin(n\theta)$, where n must be an even integer. For this we use `Fit` with just 10 such functions, and therefore introduce

```
In[44]:= basis = Table[Sin[2 i θ], {i, 10}]

Out[44]= {Sin[2θ], Sin[4θ], Sin[6θ], Sin[8θ], Sin[10θ],
          Sin[12θ], Sin[14θ], Sin[16θ], Sin[18θ], Sin[20θ]}
```

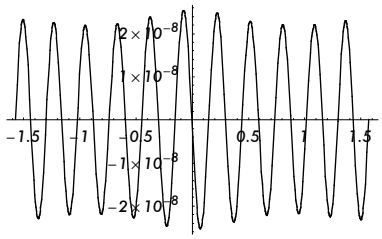
With this basis enforcing the correct symmetry, we need to compute values for the interpolation of `approxsoln[θ]` only for $0 < \theta < \pi/2$, and should not compute for $\theta = 0$ or $\theta = \pi/2$ because the symmetry already implies that we will have the correct values there. Such thinking leads us to define

```
In[45]:= iterstep :=
          (values = Table[{θ, 1/(2 π) CauchyPrincipalValue[Log[ρ[approxsoln[τ]]]
                Cot[(θ - τ)/2], {τ, 0, {θ}, 2 π}]}, {θ, π/22, π/2 - π/22, π/22}];
           approxsoln[θ_] = θ + Fit[values, basis, θ])
```

Some preliminary trials now establish that we obtain satisfactory results with

```
In[46]:= Do[iterstep, {30}]; errorplot
```

From In[46]:=



At points in $|z| < 1$, the mapping function $f(z)$ is expressed in terms of $\phi(\theta)$ via the Cauchy integral formula, while on $|z| = 1$ the relationship between $f(z)$ and $\phi(\theta)$ is more immediate. As an approximation to $f(z)$ for $|z| \le 1$, we have

```
In[47]:= approxf[z_] := Chop[If[Abs[z] == 1,
           ρ[approxsoln[Arg[z]]] Exp[i approxsoln[Arg[z]]], z Exp[1 / (2 π)
             NIntegrate[(Log[ρ[approxsoln[θ]]] + i (approxsoln[θ] - θ)) /
               (1 - z Exp[-i θ]), {θ, -π, Arg[z], π}]]]]
```

`Arg[z]` has been inserted into the range of integration here to cope with instances when $|z|$ is near 1 and the pole of the Cauchy integral is close to the contour of integration; *Mathematica* appreciates being told where problems are likely to occur!

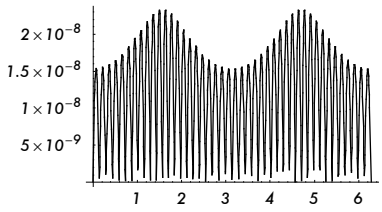
We can test the error at various points using

```
In[48]:= error[z_] := f[z] - approxf[z]
```

We can now plot the absolute value of the error on the boundary to assess the accuracy of our approximation there.

```
In[49]:= Plot[Abs[error[e^(i θ)]], {θ, 0, 2 π}]
```

From In[49]:=



We can also obtain an estimate of the overall accuracy in the interior by finding the maximum absolute value of the error at a random distribution of interior points. Because of the maximum modulus principle we, of course, expect the maximum error to occur on the boundary.

```
In[50]:= Table[error[Random[] e^(i Random[Real,{0,2 π}])], {100}] // Abs // Max
```

$$\text{Out[50]}= 8.69614 \times 10^{-9}$$

We give a second example that requires more ingenuity; the difficulties will mean that *Mathematica* takes a little longer, but it can still cope. Consider the square in the plane of $\zeta = \xi + i\,\eta$ defined by $|\xi| < 1$ and $|\eta| < 1$. The exact mapping function is given by a Schwarz-Christoffel transformation and can be expressed in terms of elliptic integrals (see Kober [11], for example), but *Mathematica* handles the computations more efficiently if we use numerical integration and define the mapping function as follows.

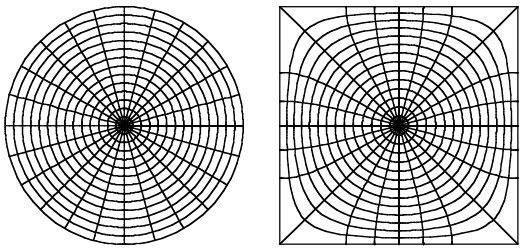
```
In[51]:= Clear[f]; f[z_?NumericQ] := Chop[2 NIntegrate[1/Sqrt[1+t^4], {t, 0, z}] / EllipticK[1/2]]
```

Unaided, `PolarMap` does not deal with this example very effectively; it cuts off the corners of the square. We can improve matters by substituting a correct `Line` for the incorrect one.

```
In[52]:= distortion = ReplacePart[PolarMap[f, {0, 1}, {0, 2 π}, Lines → {15, 25},
           PlotRange → All, Axes → False, DisplayFunction → Identity],
         Line[{{1, -1}, {1, 1}, {-1, 1}, {-1, -1}, {1, -1}}], {1, 15, 1, 1}];
```

```
In[53]:= DisplayTogetherArray[{originaldisk, distortion}]
```

```
From In[53]:=
```



For this example, $\rho(\phi)$ is a function of period $\pi/2$ that is equal to $\sec(\phi)$ for $-\pi/4 \le \phi \le \pi/4$, and can therefore be defined by

```
In[54]:= ρ[ϕ_] := Sec[Mod[ϕ + π/4, π/2] - π/4]
```

As before, Theodorsen's integral equation is solved for the function $\phi(\theta)$, and the exact solution for $-\pi/4 \le \phi \le \pi/4$ is now

```
In[55]:= exactsoln[θ_] := ArcTan[EllipticF[ArcSin[Sqrt[2] Sin[θ]], 1/2] / EllipticK[1/2]]
```

Again we begin with

```
In[56]:= approxsoln[θ_] = θ;
```

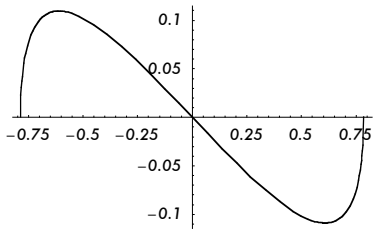
To give a visual assessment of the accuracy we now use

```
In[57]:= errorplot :=
           Plot[exactsoln[θ] - approxsoln[θ], {θ, -π/4, π/4}, PlotRange → All]
```

The initial approximation is not very close.

```
In[58]:= errorplot
```

From In[58]:=



Because of the symmetry, we can concentrate on the interval $-\pi/4 < \theta < \pi/4$; we know that $\phi(\theta)$ is equal to θ plus an odd function of θ that must vanish at both $\theta = -\pi/4$ and $\theta = \pi/4$. Since these points correspond to the corners of the square, it is near them that difficulties arise, but a local analysis shows that $\phi(\theta)$ has square root singularities and an unbounded derivative at these points. Thus, in spite of the periodicity, trigonometric interpolation is not appropriate for this example and we use `Fit` with a more apt set of basis functions for our representation. A suitable set is given by

```
In[59]:= basis =
           Union[Table[(π^2/16 - θ^2) θ^(2 i-1), {i, 4}], Table[θ (π^2/16 - θ^2)^(1/2 (2 i-1)), {i, 4}]]
```

$$\text{Out[59]=}\ \left\{\theta\sqrt{\frac{\pi^2}{16}-\theta^2},\ \theta\left(\frac{\pi^2}{16}-\theta^2\right),\ \theta^3\left(\frac{\pi^2}{16}-\theta^2\right),\ \theta^5\left(\frac{\pi^2}{16}-\theta^2\right),\right.$$
$$\left.\theta^7\left(\frac{\pi^2}{16}-\theta^2\right),\ \theta\left(\frac{\pi^2}{16}-\theta^2\right)^{3/2},\ \theta\left(\frac{\pi^2}{16}-\theta^2\right)^{5/2},\ \theta\left(\frac{\pi^2}{16}-\theta^2\right)^{7/2}\right\}$$

To try to obtain a uniform distribution of errors we should, as before, concentrate the interpolation points near the ends of the interval, and therefore introduce a nonlinear scaling similar to one used before.

```
In[60]:= new[θ_] := 1/4 π Sin[2 θ]
```

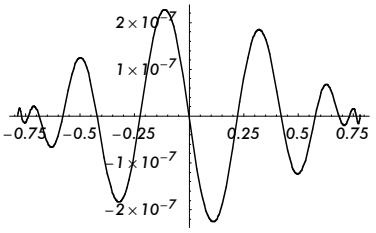
In fact, because of the extra difficulties we now have at the ends, we will push points even closer to them by using `new[new[θ]]`. (If larger errors near the ends are not a problem, you can obtain smaller errors over the major part of the interval by just using `new[θ]`.) We use `iterstep` constructed much as before; it produces from the interpolation on $-\pi/4 < \theta < \pi/4$ an `approxsoln[θ]` that is applicable for all values of θ.

```
In[61]:= iterstep := (values = Table[
           {new[new[θ]], 1/(2 π) CauchyPrincipalValue[Log[ρ[approxsoln[τ]]]
             Cot[1/2 (new[new[θ]] - τ)], {τ, 0, {new[new[θ]]}, π/4,
             3 π/4, 5 π/4, 7 π/4, 2 π}]}, {θ, π/36, π/4 - π/36, π/36}];
         partapproxsoln[θ_] = Fit[values, basis, θ];
         approxsoln[θ_] = θ + partapproxsoln[Mod[θ + π/4, π/2] - π/4])
```

Satisfactory results are now obtained with

```
In[62]:= Do[iterstep, {50}]; errorplot
```

From In[62]:=



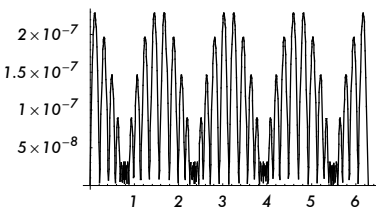
Our approximation for the mapping is given by

```
In[63]:= approxf[z_] :=
           Chop[If[Abs[z] == 1, ρ[approxsoln[Arg[z]]] E^(i approxsoln[Arg[z]]),
             z Exp[1 / (2 π) NIntegrate[(Log[ρ[approxsoln[θ]]] +
                 i (approxsoln[θ] - θ)) / (1 - z E^(-i θ)), Evaluate[Flatten[
                 {θ, Sort[N[{-π, -3 π/4, -π/4, Arg[z], π/4, 3 π/4, π}]]}]]]]]]
```

This is much as before, but *Mathematica* now appreciates being warned to expect trouble with `NIntegrate` at points corresponding to the corners. We can test the error at various points as in the previous example. This gives a plot of the absolute value of the error on the boundary.

```
In[64]:= Plot[Abs[error[E^(i θ)]], {θ, 0, 2 π}]
```

From In[64]:=



And here we obtain the maximum absolute value of the error at a random distribution of interior points.

```
In[65]:= Table[error[Random[] e^(i Random[Real,{0,2 π}])], {100}] // Abs // Max
Out[65]= 8.19077 × 10^-8
```

■ Nekrasov's Equation

An important integral equation in hydrodynamics is that derived by Nekrasov [12] to describe waves moving on the surface of a fluid. Assuming the fluid to be inviscid and the flow to be irrotational, the form of this equation that is relevant when the depth of the fluid is infinite is

$$\phi(s) = \frac{1}{3\pi}\int_{-\pi}^{\pi} \frac{\mu \sin(\phi(t))}{1+\mu\int_0^t \sin(\phi(u))\,du}\log\left(\frac{1}{|2\sin(\frac{s-t}{2})|}\right)dt.$$

Here the dependent variable ϕ is the angle between the tangent to the wave surface and the horizontal, while the independent variable s is the velocity potential of the flow along the wave surface. We assume that we are dealing with a periodic wave train and translating with the wave so that it has a fixed shape in our frame of reference, and $\phi(s)$ then has 2π as a period. With $s = 0$ corresponding to the crest of a wave, so $\phi(0) = 0$, it can be shown that $\phi(s)$ is necessarily an odd function of s if there is just one crest per period, and the waves are therefore symmetric about their crests with $s = -\pi$ and $s = \pi$ corresponding to troughs; thus $\phi(-\pi) = \phi(\pi) = 0$. Granted this, it is usual to deal with the above equation in the slightly modified form

$$\phi(s) = \frac{1}{3\pi}\int_{0}^{\pi} \frac{\mu \sin(\phi(t))}{1+\mu\int_0^t \sin(\phi(u))\,du}\log\left(\left|\frac{\sin(\frac{s+t}{2})}{\sin(\frac{s-t}{2})}\right|\right)dt.$$

The parameter μ is given in terms of physical variables by

$$\mu = \frac{3\,g\,\lambda\,c}{2\,\pi\,Q^3},$$

where g is the acceleration due to gravity, λ is the wavelength, c is the speed with which the wave is progressing (since we are moving with the wave, this is actually the speed of the fluid at infinite depth), and Q is the fluid speed at the crest.

Whatever the value of μ, there is always the solution $\phi(s) = 0$, and this is the only solution for small values μ. As μ increases, there is a bifurcation at $\mu = 3$ with a genuine waveform having just one crest per period appearing. Because of the nature of s, increasing s from $-\pi$ to π corresponds to moving from right to left along the wave surface, with the usual orientation of axes, so this solution has $\phi(s) > 0$ for $0 < s < \pi$ and $\phi(s) < 0$ for $-\pi < s < 0$. For larger μ, there are other possible solutions corresponding to there being several crests between $s = -\pi$ and $s = \pi$, but we can select the one with just one crest there by choosing an initial approximation with the correct variation. We use

```
In[66]:= approxsoln[s_] = 0.1 Sin[s];
```

There is a logarithmic singularity of the integrand in this form of Nekrasov's equation. This can be transformed into a removable singularity by performing an integration by parts but, as with Theodorsen's equation, *Mathematica* seems to prefer the singular form provided it is given a little help.

With this example, we will compromise on accuracy to avoid excessive computation times; we shall see that the problem becomes more difficult as μ increases, and any method of solution then requires lengthy procedures to obtain accurate results. With this in mind, we use `NDSolve` to compute the integral in the denominator of the integrand as a function of t just once during each iteration, rather than using `NIntegrate` every time we evaluate this integrand. The basic philosophy behind the iteration is much as before; we use

```
In[67]:= k[s_, t_] := Log[Abs[Sin[(s+t)/2] / Sin[(s-t)/2]]]
```

```
In[68]:= iterstep := (Clear[denomint]; denomint[t_] =
           denomint[t] /. NDSolve[{denomint'[t] == Sin[approxsoln[t]],
               denomint[0] == 0}, denomint[t], {t, 0, π}][[1]];
         values = Table[{new[s], 1/(3 π) NIntegrate[
               k[new[s], t] μ Sin[approxsoln[t]] / (1 + μ denomint[t]),
               {t, 0, new[s], π}, SingularityDepth → 40,
               MaxRecursion → 40]}, {s, π/n, π - π/n, π/n}];
         approxsoln = Interpolation[Union[values,
            (values /. {x_?NumericQ, y_?NumericQ} → {-x, -y}),
            {{-π, 0}, {0, 0}, {π, 0}}]])
```

We have `Interpolation` here rather than `InterpolatingPolynomial` or `Fit` for reasons to be given later. We also have an integer n governing the number of interpolation points we use, and `new[s]` to allow us to use a scaling to redistribute these points, but we begin with a moderate value of the former and no scaling for the latter.

```
In[69]:= n = 10; new[s_] := s
```

Now for a moderate value of μ.

```
In[70]:= μ = 5;
```

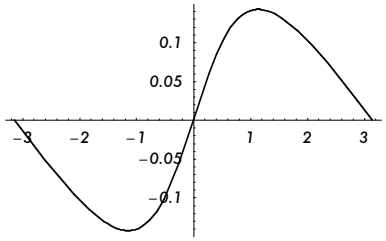
We obtain the relevant approximation and save it for later use.

```
In[71]:= Do[iterstep, {30}]; soln[1, s_] = approxsoln[s];
```

We can now plot the result.

In[72]:=
```
Plot[soln[1, s], {s, -π, π}]
```

From In[72]:=

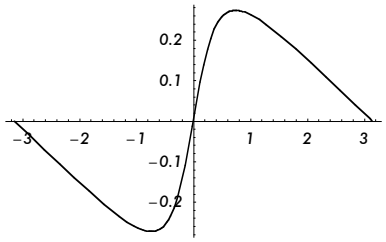


We see that there has been only moderate distortion of our original approximation as part of a sine wave, but there is a steepening near the origin. Anticipating that there will be more such distortion for larger μ, we increase the value of n for the next case.

In[73]:=
```
μ = 10; n = 20; Do[iterstep, {30}];
```

In[74]:=
```
soln[2, s_] = approxsoln[s]; Plot[soln[2, s], {s, -π, π}]
```

From In[74]:=

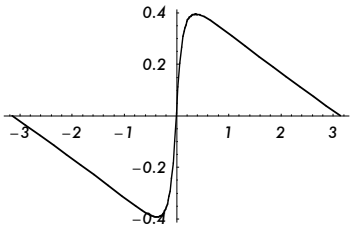


The distortion has indeed increased somewhat and we note that the solution looks roughly like a cubic near $s = 0$, but is virtually linear towards the ends of the interval. Neither a polynomial nor a finite trigonometric sum is suitable for approximating such a function, which is why we have used the cubic splines produced by `Interpolation`. Since it is now clear that the neighborhood of the origin is becoming more important, we will introduce a scaling that pushes more interpolation points into this region as we increase μ further, and also increase n.

In[75]:=
```
μ = 30; n = 30; new[s_] := π (s/π)^1.4;
Do[iterstep, {30}]; soln[3, s_] = approxsoln[s];
```

In[76]:=
```
Plot[soln[3, s], {s, -π, π}, PlotRange → All]
```

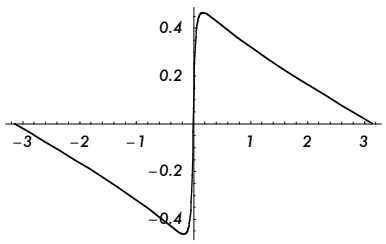
From In[76]:=



We will perform the calculations for just one more value of μ, modifying both n and `new[s]` again.

```
In[77]:= μ = 100; n = 50; new[s_] := π (s/π)^1.5;
         Do[iterstep, {30}]; soln[4, s_] = approxsoln[s];
In[78]:= Plot[soln[4, s], {s, -π, π}, PlotRange → All]
```

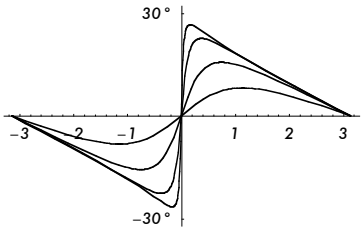
From In[78]:=



We can now bring together the four solutions computed thus far.

```
In[79]:= Plot[Evaluate[Table[soln[i, s], {i, 4}]],
          {s, -π, π}, PlotRange → {-0.56, 0.56},
          Ticks → {Automatic, {{π/6, "30°"}, {-π/6, "-30°"}}}]
```

From In[79]:=



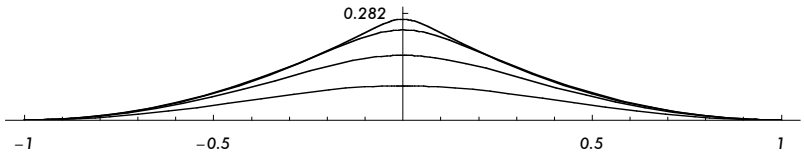
The trend seems to be clear; as μ increases further, we might expect the solution to approach one where this graph has a discontinuity at $s = 0$. This would mean that the surface of the wave is no longer smooth at the crest, but has a corner there. Now Stokes showed that one could have such corners only if the angle (as seen by the fluid) is 120°, which here corresponds to ϕ jumping from $-30°$ to $+30°$ as s increases through 0, and conjectured that the limit of this family of waveforms as $\mu \to \infty$ is, indeed, such a wave with a 120° corner. This is supported by the fact that the flow in such a corner has zero fluid speed actually at the corner, for letting $Q \to 0$ in the definition of μ above we obtain $\mu \to \infty$. This Stokes's conjecture (now proved) is also illustrated by the previous graph.

We can use the solutions obtained to plot the actual shape of the waves. For convenience, we normalize the wavelengths to 2 and choose the horizontal axis to pass through the troughs, so the profiles covering one period start at the point $(-1, 0)$ and end at the point $(1, 0)$.

```
In[80]:= Do[μ = 60 - 280 i/3 + 85 i^2/2 - 25 i^3/6;
 Clear[denomint, dtau, tau, y];
 denomint[s_] =
  denomint[s] /. NDSolve[{denomint'[s] == Sin[soln[i, s]],
      denomint[0] == 0}, denomint[s], {s, -π, π}][[1]];
 dtau[s_] = μ Sin[soln[i, s]] / (3 (1 + μ denomint[s]));
 tau[s_] = tau[s] /.
   NDSolve[{tau'[s] == dtau[s], tau[0] == 0}, tau[s], {s, -π, π}][[1]];
 y[s_] = y[s] /. NDSolve[{y'[s] == e^(i soln[i,s]-tau[s]), y[0] == 0},
     y[s], {s, -π, π}][[1]];
 zed[s_] = (y[s] - i Im[y[-π]])/Re[y[-π]];
 waveplot[i] = ParametricPlot[{Re[zed[s]], Im[zed[s]]},
   {s, -π, π}, DisplayFunction → Identity], {i, 4}]

In[81]:= Show[Evaluate[Table[waveplot[i], {i, 4}]],
 AspectRatio → Automatic, Ticks → {Automatic, {0.282}},
 PlotRange → {0, 0.3}, DisplayFunction → $DisplayFunction]
```

From In[81]:=



Again we see evidence that the waveform is approaching a state with a corner as $\mu \to \infty$. The point marked 0.282 on the vertical axis is where the corner forming the crest lies in this limiting wave of greatest height. (To illustrate the difficulty of calculations in this area, however they are done and whatever theoretical approach is used, while there is agreement on this value to 3 decimal places in the literature, there still seems to be some doubt over what the fourth digit should be.)

The previous two graphs also suggest a further conjecture, due to Krasovskii, to the effect that, for all waves of this family, we have $|\phi(s)| \leq 30^\circ$ for all values of s. In fact, this conjecture is now known to be false. For large values of μ, the graph of $\phi(s)$ acquires small ripples near the maximum and minimum in our figure, and these are responsible for values of $|\phi(s)|$ that are slightly greater than 30° at points close to the crest. To demonstrate these in a numerical approach based on Nekrasov's equation, calculations performed with $\mu = 10^{18}$ showed a maximum value for $|\phi(s)|$ of 30.3787°. Since the effect being investigated looks very like the familiar Gibbs phenomenon associated with approximations, one must be careful to produce a real effect and not one artificially introduced by the method used. With such large values of μ involved, it is evident that asymptotic methods are more appropriate for such investigations. A recent article concerning this problem is Byatt-Smith [13], and earlier references can be traced from this source.

■ Concluding Remarks

In our examples, we have been able to use the natural iteration scheme, but it is useful to know that a slight modification can yield convergence when this fails. With a linear integral equation, whose general form we write as

$$f(x) = g(x) + \lambda \int_a^b K(x, y)\, f(y)\, dy,$$

the standard iteration leading to the Neumann series is guaranteed to converge for any initial guess only if $|\lambda|$ is less than the modulus of the smallest eigenvalue. But Bückner [14] has shown that if we use instead the iteration

$$f_{n+1}(x) = (1-\theta)\, g(x) + \theta\, f_n(x) + (1-\theta)\, \lambda \int_a^b K(x, y)\, f_n(y)\, dy,$$

with θ some suitable constant, we can achieve convergence for λ over a greater range of values; this reduces to the standard iteration if we set $\theta = 0$. As an example of this, consider the equation

$$f(x) = e^x + \lambda \int_0^1 e^{x-y}\, f(y)\, dy,$$

which has the single eigenvalue $\lambda = 1$, and the solution for all $\lambda \neq 1$ is

```
In[82]:= exactsoln[x_] := e^x/(1 - λ)
```

We begin with

```
In[83]:= Clear[approxsoln]; approxsoln[x_] = 1;
```

and use the same means of testing accuracy as previously with

```
In[84]:= errorplot :=
          Plot[exactsoln[x] - approxsoln[x], {x, 0, 1}, PlotRange → All]
```

The procedure is much as used earlier, with

```
In[85]:= new[x_] := Sin[π x/2]^2
```

```
In[86]:= iterstep :=
          (values = Table[{new[x], (1 - θ) e^new[x] + θ approxsoln[new[x]] + (1 - θ) λ
                NIntegrate[e^(new[x] - y) approxsoln[y], {y, 0, 1}]}, {x, 0, 1, 1/10}];
           approxsoln[x_] = InterpolatingPolynomial[values, x])
```

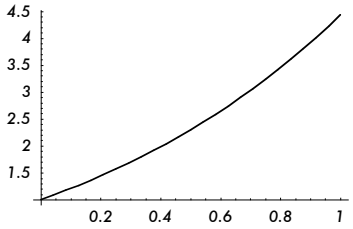
For the value $\lambda = 0.5$, the usual iteration with $\theta = 0$ converges.

```
In[87]:= λ = 0.5; θ = 0;
```

The initial approximation is not very good.

```
In[88]:= errorplot
```

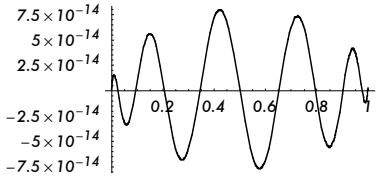
From In[88]:=



But we easily improve on it.

```
In[89]:= Do[iterstep, {50}]; errorplot
```

From In[89]:=



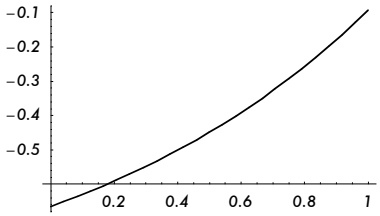
Now try the same routine with $\lambda = -2$.

```
In[90]:= approxsoln[x_] = 1; λ = -2; θ = 0;
```

First a look at the initial error.

```
In[91]:= errorplot
```

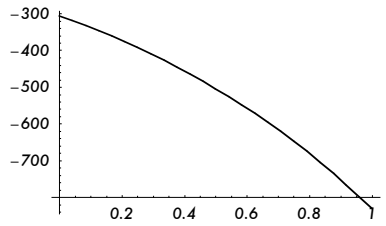
From In[91]:=



Now matters are *not* improved by this iteration!

```
In[92]:= Do[iterstep, {10}]; errorplot
```

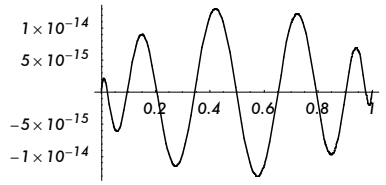
From In[92]:=



But if we use $\theta = 0.6$ in the scheme instead of $\theta = 0$, convergence is restored.

```
In[93]:= approxsoln[x_] = 1; λ = -2; θ = 0.6; Do[iterstep, {70}]; errorplot
```

From In[93]:=



■ References

[1] R. Kress, *Linear Integral Equations*, New York: Springer-Verlag, 1989.

[2] R. Kress, *Numerical Analysis*, New York: Springer-Verlag, 1998.

[3] L. Fox and E. T. Goodwin, "The Numerical Solution of Non-Singular Linear Integral Equations," *Philosophical Transactions of the Royal Society of London*, Series A, **245**, 1953 pp. 501–534.

[4] E. R. Love, "The Electrostatic Field of Two Equal Circular Conducting Disks," *Quarterly Journal of Mechanics and Applied Mathematics*, **2**, 1949 pp. 428–451.

[5] J. C. Cooke, "The Coaxial Circular Disc Problem," *Zeitschrift für Angewandte Mathematik und Mechanik*, **38**, 1958 pp. 349–356.

[6] E. R. Love, "The Potential Due to a Circular Parallel Plate Condenser," *Mathematika*, **37**, 1990 pp. 217–231.

[7] W. C. Chew and J. A. Kong, "Microstrip Capacitance for a Circular Disk through Matched Asymptotic Expansions," *SIAM Journal on Applied Mathematics*, **42**, 1982 pp. 302–317.

[8] V. I. Fabrikant, "Electrostatic Problem of Several Arbitrarily Charged Unequal Coaxial Disks," *Journal of Computational and Applied Mathematics*, **18**, 1987 pp. 129–147.

[9] P. Henrici, *Applied and Computational Complex Analysis*, Vol. 3, New York: John Wiley & Sons, 1986.

[10] P. K. Kythe, *Computational Conformal Mapping*, Boston: Birkhäuser, 1998.

[11] H. Kober, *Dictionary of Conformal Representations*, New York: Dover, 1957.

[12] A. I. Nekrasov, "On Waves of Permanent Type," *Izv. Ivanovo-Voznesensk. Politekhn. Inst.*, **3**, 1921 pp. 52–65; **6**, 1922 pp. 155–171.

[13] J. G. Byatt-Smith, "Numerical Solution of Nekrasov's Equation in the Boundary Layer Near the Crest for Waves Near the Maximum Height," *Studies in Applied Mathematics*, **106**, 2001 pp. 393–405.

[14] H. Bückner, "A Special Method of Successive Approximations for Fredholm Integral Equations," *Duke Mathematical Journal*, **15**, 1948 pp. 197–206.

About the Author

Stan Richardson received his Ph.D. in applied mathematics from the University of Cambridge in 1968 and has been at Edinburgh since 1971. His principal research interest is in free boundary problems, particularly those arising in fluid mechanics, the approach being essentially analytic using methods based on conformal mapping and complex variable theory.

Stan Richardson
School of Mathematics
University of Edinburgh
James Clerk Maxwell Building
The King's Buildings
Mayfield Road
Edinburgh EH9 3JZ
Scotland
S.Richardson@ed.ac.uk