Neural, Parallel, and Scientific Computations 18 (2010) 195-206

# WEIGHTED QUADRATURE BY CHANGE OF VARIABLE

LAWRENCE F. SHAMPINE

Mathematics Department, Southern Methodist University
Dallas, TX 75275, USA

**ABSTRACT.** The standard way of dealing with singular integrals is first to write the integral in the form $\int_a^b f(x)w(x)dx$ with a weight function $w(x) > 0$ and a relatively smooth function $f(x)$. Polynomials orthogonal on $[a, b]$ with weight function $w(x)$ are then used to derive accurate formulas for approximating the integral. The approach developed in this paper is to use a change of variable to obtain an integral over a finite interval that has a relatively smooth integrand and no weight function. Popular formulas can be applied to this standard problem to obtain easily alternatives to all the common schemes for weighted quadrature. Moreover, the approach provides a way to apply schemes for estimating the error in the unweighted case to integrals involving weight functions. It can be used with popular codes to approximate integrals with some kinds of strong end point singularities. Implementation of the approach is quite convenient in MATLAB.

**Key Words** quadrature, singular, Gauss-Legendre, Gauss-Hermite, Gauss-Jacobi, MATLAB

## 1. INTRODUCTION

Typical methods for the numerical approximation of definite integrals, *quadrature*, are based on approximating the integrand by a polynomial. Singular integrals have integrands that do not behave like polynomials. One way to deal with this difficulty is to factor the integrand so that the non-polynomial behavior is contained in a weight function $w(x) > 0$,

$$I(f) = \int_a^b w(x)f(x)\,dx \tag{1.1}$$

The integral is approximated by a quadrature formula of the form

$$Q(f) = \sum_{j=1}^{M} A_j\, f(x_j) \tag{1.2}$$

with nodes $x_j$ that are real, distinct, and satisfy

$$a \le x_1 < x_2 < \cdots < x_{M-1} < x_M \le b$$

The non-polynomial behavior of $w(x)$ on $[a, b]$ is accounted for analytically in the coefficients $A_j$. The formula is said to be of degree of precision $d$ if $Q(p) = I(p)$ for all polynomials $p(x)$ of degree less than or equal to $d$. For standard intervals and

Received March 15, 2010 

weight functions the theory of orthogonal polynomials can used to obtain formulas with the highest possible degree of precision. The names of the formulas correspond to the names of the orthogonal polynomials. For example, the Legendre polynomials are orthogonal on $[-1,+1]$ with $w(x) \equiv 1$ and the corresponding quadrature formulas are called Gauss-Legendre formulas.

In the usual approach to quadrature with weight function, it is necessary to derive a formula (1.2) for each $w(x)$, $[a,b]$, and $M$. Our approach accounts for $w(x)$ and $[a,b]$ by a change of variable that results in an integral on a finite interval with no weight function. The approach is illustrated by a special case found in Davis and Rabinowitz [2, p. 89],

$$\int_0^\infty e^{-x} f(x)\,dx = \int_0^1 f(\log(1/t))\,dt \tag{1.3}$$

The Gauss-Laguerre formulas are based on the polynomials of Laguerre that are orthogonal with respect to the weight function $e^{-x}$ on the interval $[0,\infty)$. They provide the highest degree of precision in approximating a well-behaved function $f(x)$ by a polynomial. The change of variable has some advantages, one being that any useful formula/program can be applied to the relatively well-behaved function $F(t) = f(\log(1/t))$ on a finite interval. For instance, Gauss-Legendre formulas could be applied to this integral to obtain an alternative to the Gauss-Laguerre formulas. A great deal of work has been done on the integration of relatively smooth integrands over finite intervals that we can bring to bear after a change of variable. In particular, most of the attention given to estimating the error of a quadrature formula is for an integral over a finite interval with no weight function. This work can be applied immediately to the integral resulting from a change of variable that accounts for the weight function.

We develop simple transformations that are efficiently realized in the problem-solving environment MATLAB [8] which provide alternatives for all the common weighted Gaussian quadrature formulas. In particular, we treat Hermite, generalized Laguerre, Jacobi, and logarithmic weights. Our approach is not much affected by the interval, so we can treat the Hermite weight $e^{-x^2}$ not just on the standard interval $[-\infty,\infty]$, but any interval and in particular, on the interval $[0,\infty)$ considered in [6]. Our approach is especially attractive when there is a parameter, as with generalized Laguerre weight, or two parameters, as with Jacobi weight. In the next section we discuss a transformation built into the MATLAB quadrature program `quadgk` [10] and the Fortran quadrature program Q1DA [5] that in conjunction with an adaptive selection of the mesh allows them to deal efficiently with some kinds of moderate end point singularities. With the additional transformations developed here, the programs can deal with some kinds of strong end point singularities. This improvement is illustrated in §8.

## 2. TRANSFORMATION IN `quadgk` AND Q1DA

In their discussion of an example of the transformation (1.3), Kahaner, Moler, and Nash [5, p. 165] write "Transformations must be applied with great care or the result will not be an easier integral, only a finite one." Certainly the transformations of this paper are no panacea. Indeed, Kahaner et alia suggest following (1.3) with another transformation to moderate behavior at infinity that is difficult to approximate after it is mapped to the origin. In the next section we return to this matter and provide some numerical examples computed with the programs Q1DA and `quadgk`. The `quadgk` program uses algebraic changes of variable to transform integrals over infinite intervals to integrals over finite intervals. When the interval is finite, both Q1DA and `quadgk` use a transformation to weaken end point singularities. As stated in [5, p. 158], this transformation is

$$\int_a^b F(x)\,dx = \int_\alpha^\beta F(p(t))\,p'(t)\,dt$$

where

$$x = p(t) = b - (b-a)u^2(2u+3), \qquad u = \frac{t-b}{b-a}$$

and $p(\alpha) = a$, $p(\beta) = b$. It is shown in [5, 10] that after this transformation, an integrand with logarithmic end point singularities $\log(x-a)$ and/or $\log(b-x)$ is no longer singular. The same is true for algebraic singularities $(x-a)^\alpha$ and $(b-x)^\beta$ provided that $\alpha$ and $\beta$ are at least as big as $-1/2$.

## 3. THE BASIC IDEA

If we change the variable in (1.1) to $x = \psi(t)$, we obtain

$$I(f) = \int_a^b w(x)\,f(x)\,dx = \int_{\phi(a)}^{\phi(b)} w(\psi(t))\,f(\psi(t))\,\psi'(t)\,dt$$

In this expression we use the inverse transformation $t = \phi(x)$. The basic idea is to find a change of variable such that

$$w(\psi(t))\,\psi'(t) = c \tag{3.1}$$

for a constant $c$ chosen to result in standard special functions. With this we approximate numerically

$$\int_{\phi(a)}^{\phi(b)} cf(\psi(t))\,dt \tag{3.2}$$

involving only the relatively well-behaved function $cf$. The assumption that the weight function is positive is important in the theory of orthogonal polynomials. It is important in our approach, too, because with this assumption, the fundamental equation (3.1) guarantees that $\psi'(t)$ is of one sign, hence $\psi(t)$ is monotone. The

definition (3.1) leads immediately to an expression for the inverse of this change of variable,

$$t = \int^x \frac{w(\psi)}{c}\, d\psi = \phi(x) \tag{3.3}$$

If we wish to develop a formula, we can apply any standard formula for unweighted quadrature on the interval $[\phi(a), \phi(b)]$ of the form

$$Q(f) = \sum_{j=1}^{n} A_j\, F(x_j)$$

after the change of variable. This results in

$$I(f) = \int_a^b w(x)\, f(x)\, dx \approx Q(f) = \sum_{j=1}^{n} c\, A_j\, f(\psi(x_j))$$

That is, the coefficients of the new formula are $\{c\, A_j\}$ and the nodes are $\{\psi(x_j)\}$. Alternatively, if we can evaluate the change of variables $\psi(t)$ efficiently, we can apply our favorite program for unweighted quadrature to the new problem.

As we shall see in turn, our approach to the standard weight functions results in common special functions that are all available in MATLAB [8]. In §1 we commented that our approach to Gauss-Laguerre quadrature resulted in the change of variable cited and we show this now. For $w(x) = e^{-x}$ on $[0, \infty)$ it proves convenient to take $c = -1$. This choice leads to

$$t = \int^x \frac{e^{-\psi}}{-1}\, d\psi = e^{-x} = \phi(x)$$

The end points of (3.2) are $\phi(0) = 1$ and $\phi(\infty) = 0$. The change of variable is given by the inverse of $\phi(x)$, namely $x = \psi(t) = \log(1/t)$.

For our purposes it would be nearly as convenient to use a polynomial in $t$ of low degree as the constant $c$ in (3.1). Of course, this is useful only if it leads to $\psi(t)$ and $\phi(x)$ that can be evaluated in terms of familiar special functions. An example used by Krylov [7, p. 131] to illustrate Gauss-Laguerre quadrature,

$$I = \int_0^\infty e^{-x} \frac{x}{1 - e^{-2x}}\, dx = \frac{\pi^2}{8} \tag{3.4}$$

shows why this might be desirable. For large $x$ the function $f(x)$ is very nearly the polynomial $x$, so we can expect Gauss-Laguerre quadrature to be very effective because it approximates this function with a polynomial. This is not true of the function that results from the transformation (1.3). The original function $f(x)$ tends to infinity as $x \to \infty$, hence after transformation the new integrand tends to infinity as $t \to 0$. Example 5.8 of [5],

$$I = \int_0^\infty e^{-x} \cos^2(x^2)\, dx \approx 0.70260$$

shows a different aspect of the transformation. In this case $f(x)$ oscillates infinitely often as $x \to \infty$ and correspondingly, after the transformation (1.3), the new function $\cos^2(\log^2(t))$ oscillates infinitely often as $t \to 0$. Section 2 describes a transformation built into `quadgk` and Q1DA to weaken end point singularities. As a result of this transformation and adaptive mesh selection, `quadgk` approximates both the new integrals without difficulty. Kahaner, Moler, and Nash [6] report that the same is true for Q1DA, but they suggest an alternative transformation for the original problem which can be derived with our approach by requiring that

$$w(\psi(t))\,\psi'(t) = -2t$$

This leads to $\psi(t) = \log(1/t^2)$ and $\phi(x) = \sqrt{e^{-x}}$. With this transformation,

$$\int_0^\infty e^{-x}\,f(x)\,dx = \int_0^1 2t\,f(\log(1/t^2))\,dt$$

The additional factor $2t$ does not complicate the application of familiar methods and it moderates the behavior at the origin. It would seem better to use this scheme if the object is to derive formulas analogous to the Gauss-Laguerre formulas.

## 4. $\int_a^b e^{-x^2}\,f(x)\,dx$

Our approach to weights is distinctive in that the interval of integration plays a minor role. With a modern quadrature program, the weight function $e^{-x^2}$ presents no difficulty except perhaps when the interval is infinite. The classic interval for the Hermite weight function is $(-\infty, +\infty)$, but $[0, \infty)$ is also interesting, especially as regards deriving a formula and error estimate. Indeed Kahaner et al. [6] describe many sources of integrals in statistical computation with this weight function on $[0, \infty)$ and derive Gaussian quadrature formulas to approximate them. With this weight the expression (3.3) for the inverse transformation is

$$t = \int^x \frac{e^{-\psi^2}}{c}\,d\psi$$

If we choose $c = \sqrt{\pi}/2$ and take the lower limit to be 0, we find that $t = erf(x)$. The basic transformation is $x = erf^{-1}(t)$. In the classic Hermite case, the interval maps into $(-1, +1)$, but it was convenient for our experiments in MATLAB to allow general $[a, b]$,

$$\int_a^b e^{-x^2}\,f(x)\,dx = \frac{\sqrt{\pi}}{2}\int_{erf^{-1}(a)}^{erf^{-1}(b)} f(erf^{-1}(t))\,dt \tag{4.1}$$

The constant $c$ is evaluated in MATLAB as `0.5*sqrt(pi)`. Vectorization is so important to efficiency in this computing environment that the quadrature programs require that the procedure for evaluating the integrand accept a vector `t` of arguments and return a corresponding array of function values. There is a fast vectorized function built into MATLAB for evaluating $erf^{-1}(t)$, namely `erfinv(t)`. With it, the

change of variable (4.1) is an easy and efficient alternative to Gauss-Hermite quadrature on $(-\infty, +\infty)$ and offers other possibilities as well. A family of integrals with the Hermite weight and a finite interval is given as 7.4.12 in [1]. It includes

$$\int_0^1 \frac{e^{-x^2}}{1+x^2}\,dx = \frac{\pi}{4}\,e^1\,[1 - erf^2(1)]$$

`quadgk` is perfectly capable of approximating this integral directly, but it is easy to change variables to account for the weight:

```
Q = quadgk(@(t) 0.5*sqrt(pi)./(1+erfinv(t).^2),erf(0),erf(1))
```

Even with the default tolerances used, this `Q` has an error of $3.3 \times 10^{-16}$. The more interesting family of integrals 7.4.6 in [1] includes

$$\int_0^\infty e^{-x^2}\,\cos(x)\,dx = \frac{\sqrt{\pi}}{2}\,e^{-1/4}$$

Noting that $erf^{-1}(0) = 0$, $erf^{-1}(+\infty) = +1$, and using `quadgk` with default tolerances,

```
Q = 0.5*sqrt(pi)*quadgk(@(t) cos(erfinv(t)),0,1)
```

provides an approximation with error $-1.5 \times 10^{-8}$.

The popular high order quadrature formulas for $[-1, +1]$ have nodes that are close to the end points. If an adaptive program is applied to an integrand that presents difficulty near an end point, it will also evaluate the integrand at points very near the end points. There can be precision difficulties in these circumstances. The error function is very flat for large arguments and this makes inverting the function ill-conditioned. The inverse error function of MATLAB returns a result that is accurate for the argument it is given. The difficulty is that the argument we supply may not have many significant digits. That is because for an argument close to $\pm 1$, it is the difference between the argument and the limit value that determines the value of the inverse function. Multiple precision could be used to deal with this when deriving a formula. Unfortunately, when using the change of variables and a program for quadrature on finite intervals, this loss of significance can limit the accuracy possible when the new integrand has end point singularities.

## 5. $\int_0^\infty x^\alpha e^{-x} f(x)\,dx, \quad \alpha > -1$

Earlier we examined the case of Gauss-Laguerre quadrature ($\alpha = 0$) and now we examine *generalized* Gauss-Laguerre quadrature. To mention one source of such integrals recall that in §4 we considered the weight $e^{-x^2}$ on $[0, \infty)$. The paper [6] notes that one way to deal with such integrals is to use a change of variables that

results in a generalized Gauss-Laguerre quadrature,

$$\int_0^\infty e^{-x^2} f(x)\,dx = \int_0^\infty t^{-1/2} e^{-t} \frac{f(\sqrt{t})}{2}\,dt$$

As defined in MATLAB, the incomplete gamma function `gammainc(x,a)` is

$$P(x,a) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t}\,dt$$

If we note that

$$\Gamma(1+\alpha)\frac{dP(\psi,1+\alpha)}{d\psi} = \psi^\alpha e^{-\psi}$$

and we take $c = \Gamma(1+\alpha)$ in (3.1), we find that $t = \phi(x) = P(x,1+\alpha)$. The end points map into $P(0,1+\alpha) = 0$ and $P(\infty,1+\alpha) = 1$. The basic transformation is then $x = \psi(t) = P^{-1}(t,1+\alpha)$.

For $\alpha > -1$ we have found the change of variable

$$\int_0^\infty x^\alpha e^{-x} f(x)\,dx = \int_0^1 \Gamma(1+\alpha)\, f(P^{-1}(t,1+\alpha))\,dt \tag{5.1}$$

This can be implemented both easily and efficiently in MATLAB: The constant $\Gamma(1+\alpha)$ is evaluated as `gamma(1+alpha)`. There is a fast built-in function that evaluates $P^{-1}(t,1+\alpha)$ for vector `t`, namely `gammaincinv(t,1+alpha)`.

MATLAB has programming tools that facilitate the use of our approach to weighted quadrature. Suppose that we have written a vectorized function `f` to evaluate $f(x)$ and have assigned a value to the parameter `alpha`. We can then define a new function

```
F = @(x) gamma(1+alpha)*f(gammaincinv(t,1+alpha))
```

and supply it to a quadrature program. For instance, if we use `quadgk` with default tolerances, we can approximate the integral with

```
Q = quadgk(F,0,1)
```

We have written a MATLAB program to experiment with the approximation of integrals with weight functions. The user indicates the nature of the weight and any parameters involved and supplies a vectorized function for evaluating $f(x)$. The program uses this function to define a function in a new variable in the manner just illustrated. It determines the end points for the transformed integral and then uses `quadgk` to approximate the integral. Of course, the program allows error tolerances to be specified in the usual way for the underlying `quadgk` and simply passes them on for the actual computation.

$$6.\ \int_0^1 x^\alpha (1-x)^\beta f(x)\,dx, \quad \alpha > -1, \beta > -1$$

The Jacobi weights correspond to $\int_{-1}^1 (1-s)^\alpha (1+s)^\beta f(s)\,ds$. These weights are more complicated than the ones previously considered because there are two parameters. Still, if we work instead on the interval $[0,1]$, there is an elegant change of variable that is applicable for all permissible values of the parameters. The expression (3.3) for the inverse transformation is now

$$t = \int^x \frac{\psi^\alpha (1-\psi)^\beta}{c}\,d\psi$$

As defined in MATLAB, the beta function `beta(z,w)` is

$$B(z,w) = \int_0^1 t^{z-1}(1-t)^{w-1}dt$$

and the incomplete beta function `betainc(x,z,w)` is

$$I_x(z,w) = \frac{1}{B(z,w)} \int_0^x t^{z-1}(1-t)^{w-1}dt$$

Evidently if we take $c = B(1+\alpha, 1+\beta)$, we have $t = I_x(1+\alpha, 1+\beta)$. The end points map into $I_0(1+\alpha, 1+\beta) = 0$ and $I_1(1+\alpha, 1+\beta) = 1$.

For $\alpha > -1$ and $\beta > -1$ we have found the change of variable

$$\int_0^1 x^\alpha (1-x)^\beta f(x)\,dx = B(1+\alpha, 1+\beta) \int_0^1 f(I_t^{-1}(1+\alpha, 1+\beta))\,dt \tag{6.1}$$

This change of variable can also be implemented both easily and efficiently in MATLAB: The constant $B(1+\alpha, 1+\beta)$ is evaluated as `beta(1+alpha,1+beta)`. There is a fast built-in function that evaluates $I_t^{-1}(1+\alpha, 1+\beta)$ for vector `t`, namely `betaincinv(t,1+alpha,1+beta)`.

This change of variable is elementary when $\beta = 0$ (or $\alpha = 0$). With $c = 1/(1+\alpha)$, the expression (3.3) is

$$t = \int^x \frac{\psi^\alpha}{c}\,d\psi = x^{1+\alpha} = \phi(x)$$

The transformation maps $[0,1]$ into $[0,1]$ and $x = \psi(t) = t^{1/(1+\alpha)}$, so

$$\int_0^1 x^\alpha f(x)\,dx = \frac{1}{1+\alpha} \int_0^1 f(t^{1/(1+\alpha)})\,dt, \qquad \alpha > -1$$

This change of variable is discussed in [2, 4]. A number of authors have developed formulas for this weight with $\alpha = 1/2$ or $\alpha = -1/2$. For instance, Gautschi [3, pp. 160–163] develops analytical expressions for the coefficients of the two point Gauss-Jacobi formula for $\alpha = -1/2$ and $\beta = 0$,

$$Q = 1.3043\cdots \times f(0.1559\ldots) + 0.6957\ldots \times f(0.74156\ldots)$$

He applies the formula to $\int_0^1 x^{-1/2}\cos(\pi x/2)\,dx = 1.5597865\ldots$ to get the approximation $Q = 1.55759\ldots$. In our approach to weights, we change the variable to get

$$\int_0^1 x^{-1/2} f(x)\,dx = \int_0^1 2f(t^2)\,dt$$

and can then apply any formula we wish. After transformation from the standard interval $[-1,+1]$ to $[0,1]$, the two point Gauss-Legendre formula has nodes $x_1 = 1/2 - \sqrt{3}/6$ and $x_2 = 1/2 + \sqrt{3}/6$ and weights $A_i = 1/2$. This leads to a new formula

$$Q = f(x_1^2) + f(x_2^2) = f(0.04466\ldots) + f(0.6220\ldots)$$

For Gautschi's example, this formula provides $Q = 1.55701\ldots$. There is no reason to expect that one approach will approximate this integral better than the other and neither shows any great advantage for this example. An interesting aspect of this particular weight function is that Krylov [7, p. 120] shows how to derive the Gauss-Jacobi formula of $n$ nodes for $w(x) = x^{-1/2}$ in terms of the Gauss-Legendre formula with $2n$ nodes. Our approach derives a formula of $n$ nodes for any admissible $\alpha$ and $\beta$ by a change of variable and subsequent application of the Gauss-Legendre formula with $n$ nodes.

Gautschi [3, pp. 169–170] shows how to change the variable so as to deal with an interesting class of integrals on $[0,\infty)$ by reducing them to an integral with Jacobi weights on a finite interval. Suppose that we want to approximate

$$\int_0^\infty F(x)\,dx$$

and the integrand behaves like

$$F(x) \sim \begin{cases} f_0\, x^p & \text{with } p > -1 \text{ as } x \to 0, \\ f_\infty\, x^{-q} & \text{with } q > +1 \text{ as } x \to \infty. \end{cases}$$

By moving appropriate factors of $F(x)$ to a weight function, we can write the integral as

$$\int_0^\infty \frac{x^p}{(1+x)^{p+q}}\, f(x)\,dx, \quad p > -1, q > +1$$

for a function $f(x)$ that is well-behaved at both the origin and infinity. Indeed, $f(0) = f_0$ and $f(\infty) = f_\infty$. We need to modify slightly the transformation used by Gautschi to get an integral on $[0,1]$. To this end we take $x = t/(1-t)$, so that $t = x/(1+x)$ and

$$\int_0^\infty \frac{x^p}{(1+x)^{p+q}}\, f(x)\,dx = \int_0^1 t^p\,(1-t)^{q-2}\, f\left(\frac{t}{1-t}\right) dt$$

We can now apply the transformation developed earlier for the Jacobi weight.

$$7.\ \int_0^1 -\log(x)\, f(x)\, dx$$

The polynomial change of variable used by Q1DA and `quadgk` deals well enough with a logarithmic singularity, but for a number of applications it is useful to develop formulas with $-\log(x)$ as weight. Also, this case illustrates what has to be done if the special function needed in our approach is not available. However, before going into this, we note that a preliminary change of variable can be used to deal not only with this logarithmic weight, but even with powers of the weight: For $\alpha > -1$, changing the variable of integration to $x = e^{-t}$ leads to

$$\int_0^1 (-\log(x))^\alpha\, f(x)\, dx = \int_0^\infty t^\alpha\, e^{-t} f(e^{-t})\, dt$$

This is, of course, an integral with a generalized Gauss-Laguerre weight and a comparatively smooth function $F(t) = f(e^{-t})$ to which we can apply the scheme developed in §5.

Returning now to the direct application of our approach to the particular weight function $-\log(x)$, if we take $c = 1$, the expression (3.3) for the inverse transformation is

$$t = \int_0^x -\log(\psi)\, d\psi = x(1 - \log(x)) = \phi(x)$$

The end points map into $\phi(0) = 0$ and $\phi(1) = 1$. There is no special function in MATLAB for the inverse of $\phi(x)$, so we evaluate it by solving the algebraic equation $0 = \phi(x) - t$ numerically in the function `psilog` displayed below. Notice the coding of the nested function `phi_minus_t`. For given $t$ the root solver `fzero` evaluates $\phi(\xi) - t$ at both ends of the interval $0 \le \xi \le 1$ where it is to locate a root. At $\xi = 0$, MATLAB evaluates $\xi \log(\xi)$ as `NaN`. Accordingly, we have to code the function so as to provide the proper limit value 0. The function is smooth and has opposite signs at the ends of the interval. In the circumstances `fzero` is globally and superlinearly convergent.

```
function x = psilog(t)
  x = zeros(size(t));
  for entry = 1:length(t)
    if t(entry) == 0,  x(entry) = 0;
    elseif t(entry) == 1,  x(entry) = 1;
    else  x(entry) = fzero(@phi_minus_t,[0,1]);
    end
  end
      function v = phi_minus_t(xi)
        if xi == 0,  v = - t(entry);
        else  v = xi*(1 - log(xi)) - t(entry);
        end
      end % nested function phi_minus_t
```

```
end % psilog
```

To illustrate our approach to deriving a formula, we first note the conventional formula of Gaussian type for this weight function as stated in Table 25.7 of [1],

$$\int_0^1 -\log(x)\, f(x)\, dx \approx 0.718539\, f(0.112009) + 0.281461\, f(0.602277)$$

If we use the two-point Gauss-Legendre formula on $[0, 1]$ stated in §6, the nodes of a new formula are computed by

```
x = psilog([1/2-sqrt(3)/6,1/2+sqrt(3)/6])
```

and the resulting formula is

$$\int_0^1 -\log(x)\, f(x)\, dx \approx 0.5\, f(0.0539011) + 0.5\, f(0.425020)$$

We have no reason to think that one of these formulas is better than the other. However, the table cited provides only the formulas of 2, 3, and 4 nodes. In our approach it is easy to obtain a formula from transformation and Gauss-Legendre quadrature of whatever number of nodes seems appropriate to the task. In passing we remark that Trefethen [11] provides a short MATLAB program for computing the coefficients of the Gauss-Legendre formula of any desired number of nodes.

Evaluating $\psi(t)$ in this way is perfectly satisfactory for developing formulas. Indeed, this example shows that it is easy enough to develop formulas for any $w(x)$ with a convenient analytical integral. However, for the routine computation of integrals involving $w(x)$, we would want the speed gained by compiling a function to solve the algebraic equations.

## 8. A FINAL EXAMPLE

Mori [9] and others have considered the approximation of

$$\int_{-1}^{+1} \frac{1}{x-2}\,(x+1)^{-3/4}\,(1-x)^{-1/4}\, dx = -1.949\ldots$$

With default tolerances, `quadgk` reports that it goes to minimum mesh spacing near $x = -1$ and does not achieve the desired accuracy. The change of variables built into the program to weaken end point singularities deals with the singularity at $x = +1$, but the singularity at $x = -1$ is too strong. These end point singularities correspond to a Jacobi weight and the change of variables developed in §6 for this weight makes the integral an easy one for `quadgk`. Mori and others have considered changes of variable that in principle deal with end point singularities without asking the user to specify just what the singularity is. These IMT and DE transformations are certainly attractive, but all the authors who have investigated transformations of this kind have remarked on precision difficulties. Indeed, Mori [9, p. 126] provides details for Jacobi

weights as in this example—there are severe difficulties with cancellation and overflow that require careful handling. A similar discussion of the practical difficulties for a different transformation of this kind is found in Yserentant [12]. These difficulties arise in evaluating the integrand near a singular point, difficulties that are avoided in our approach because we ask the user to specify the nature of the singularity, which allows us to deal with it analytically.

## REFERENCES

[1] M. Abramowitz and I.A. Stegun, eds., *Handbook of Mathematical Functions*, Dover Publications, New York, 1972.

[2] P.J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, 2nd ed., Academic, Orlando, 1984.

[3] W. Gautschi, *Numerical Analysis An Introduction*, Birkhäuser, Basel, 1997.

[4] E. Isaacson and H.B. Keller, *Analysis of Numerical Methods*, Wiley, New York, 1966.

[5] D. Kahaner, C. Moler, and S. Nash, *Numerical Methods and Software*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[6] D. Kahaner, G. Tietjen, and R. Beckman, Gaussian-quadrature formulas for $\int_0^\infty e^{-x^2}\, g(x)\, dx$, *J. Statistical Computation and Simulation*, 15:155–160, 1982.

[7] V.I. Krylov, *Approximate Calculation of Integrals*, translated by A.H. Stroud, ACM Monograph Series, MacMillian Company, New York, 1962.

[8] MATLAB, The MathWorks, Inc., 3 Apple Hill Dr., Natick, MA 01760.

[9] M. Mori, Quadrature formulas obtained by variable transformation and the DE-rule, *J. Comp. Appl. Math.*, 12&13:119–130, 1985.

[10] L.F. Shampine, Vectorized adaptive quadrature in MATLAB, *J. Comp. Appl. Math.*, 211:131–140, 2008.

[11] L.N. Trefethen, Is Gauss quadrature better than Clenshaw-Curtis?, *SIAM Review*, 50:67–87, 2008.

[12] H. Yserentant, A remark on the numerical computation of improper integrals, *Computing*, 30:179–183, 1983.