Experimental Numerical Quadrature of Improper Integrals

Abstract: A powerful scheme for the numerical evaluation of $\int_A^B f(x)\,dx$ approximates it by

$$\sum_{-\infty < n < \infty} f(X(n\cdot\Delta w))\cdot X'(n\cdot\Delta w)\cdot\Delta w$$

as $\Delta w \to 0$ for substitutions $x := X(w)$ like $X(w) := (A+B)/2 + \tanh(\mu + U\cdot\sinh(w))\cdot(B-A)/2$ that approach the integral's endpoints extremely quickly — doubly exponentially for this X. Such substitutions were introduced by Takahashi and Mori about four decades ago, and have been found to tolerate mild singularities of $f(x)$ at $x = A$ and/or $x = B$ by, among others,

D.H. Bailey et al. [2005] "A Comparison of Three High-Precision Quadrature Schemes" pp. 317-329 of *Experimental Math.* **14**:3 .

Convergence as $\Delta w \to 0$ is ultimately astonishingly fast, usually like $\exp(-\text{Const}/\Delta w)$. Questions arise when the scheme is adapted to fixed-precision floating-point arithmetic in an environment like, say, MATLAB's, which is predisposed more to vectorized than to parallel computations:

<> How should $X(w)$ be chosen; in this instance, the constants μ and U ?
<> How should the infinite sum on n be truncated to a finite sum?
<> To what extent can the sum be compensated for that truncation?
<> If $\Delta w = w_{max}\cdot 2^{-k}$ for $0 \le k \le < K$, what are good choices for w_{max} and K ?
<> How should a disgustingly parallel $\sum_n$ be vectorized instead?
<> How reliably can the error $|\sum_n - \int|$ be estimated?
<> How do roundoff and over/underflow complicate these questions?

Only a few if these questions were answered for the [Integrate] key on the HP-34C and HP-15C calculators over three decades ago; see W. Kahan [1980] "Handheld Calculator Evaluates Integrals" pp. 23-32 of *The Hewlett-Packard Journal* Aug. 1980
also posted at www.eecs.berkeley.edu/~wkahan/Math128/INTGTkey.pdf .

Coping with Roundoff in $X(w) := (A+B)/2 + \tanh(\mu + U\cdot\sinh(w))\cdot(B-A)/2$

As (vectorized) w runs from $-\infty$ to $+\infty$ we hope to compute $X(w)$ and its derivative

$$X'(w) = \mathrm{sech}^2(\mu + U\cdot\sinh(w))\cdot\cosh(w)\cdot U\cdot(B-A)/2$$

about as accurately as roundoff allows, and with only two **calls** on the Math. library.

First compute $s := $ **sinh**(w) and $c := \sqrt{(1 + s^2)}$ … $= \cosh(w)$; and then compute

$$\sigma(w) := \mu + U\cdot s \text{ , } \quad \ldots = \text{the argument of } \tanh(\sigma(w)) \text{ .}$$

Where $|\sigma(w)| < \mathrm{arcsech}(1/\sqrt{2}) = \mathrm{arctanh}(1/\sqrt{2}) \approx 0.881373587\ldots$ we compute

$\tau(w) :=$ **tanh**$(\sigma(w))$ and $\xi(w) := c - \tau(w)\cdot c\cdot\tau(w)$ … $= \mathrm{sech}^2\cdot\cosh$.

Then $X(w) := (A+B)/2 + \tau(w)\cdot(B-A)/2$ and $X'(w) := \xi(w)\cdot U\cdot(B-A)/2$.

Where $\sigma(w) \le -\mathrm{arcsech}(1/\sqrt{2})$ we compute

$\varepsilon(w) :=$ **exp**$(\sigma(w))$, $\rho(w) := 2\cdot\varepsilon(w)/(1 + \varepsilon(w)^2)$ and $\xi(w) := \rho(w)^2\cdot c$ … $= \mathrm{sech}^2\cdot\cosh$.

Then $X(w) := A + \varepsilon(w)\cdot\rho(w)\cdot(B-A)/2$ and $X'(w) := \xi(w)\cdot U\cdot(B-A)/2$.

Similarly where $\sigma(w) \ge +\mathrm{arcsech}(1/\sqrt{2})$, $\varepsilon(w) :=$ **exp**$(-\sigma(w))$ and $X(w) := B - \ldots$.