An Efficient Adaptive Levin-type Method for Highly Oscillatory Integrals

A. Ihsan Hascelik, Okkes Boy
Department of Mathematics, Gaziantep University
27300 Gaziantep, Turkey

A. Suat Gercelk
Department of Computer Technology and Programming
Gaziantep University, Gaziantep, Turkey

Abstract

In this work, we present an adaptive Levin-type method for high-precision computation of highly oscillatory integrals with integrands of the form $f(x) \exp(i \omega g(x))$. If $g'$ has no real zero in the integration interval and the integrand is sufficiently smooth, the method can attain arbitrarily high asymptotic orders without computation of derivatives. Although the proposed method requires about $2d$-digit working precision for a $d$-digit accurate approximation to the integral, it is more efficient than the numerical steepest decent method for some integrals. Moreover it produces sufficiently accurate approximations even if the integrand is only several times differentiable. For the implementation of the method we develop a Mathematica program, which can also be used for the well-known Levin-type method where the required derivatives are computed by Mathematica’s automatic differentiation package. The effectiveness of the method is discussed in the light of a set of test integrals, one of which is computed to 1000 digit accuracy.

Mathematics Subject Classification: 65D30, 65D32

Keywords: highly oscillatory integrals, adaptive Levin-type method
1 Introduction

In this paper we develop a derivative-free adaptive Levin-type method and associated Mathematica program for the computation of oscillatory integrals of the form

\[ I[f] = \int_a^b f(x) e^{i\omega g(x)} \, dx, \quad i = \sqrt{-1} \quad (1) \]

where \( \omega \), frequency, is a nonzero real number, \( f \) and \( g \), which are called amplitude and phase functions respectively, are nonoscillatory suitably smooth functions over the interval of integration. Any zero of \( g' \) in \([a, b]\) is called a stationary point of \( g \) or the integrand. This type of integrals arises in a wide range of practical problems, e.g., in quantum chemistry, image analysis, fluid mechanics, electrodynamics.

When \( \alpha = \max \{ |\omega g'(x)| : x \in [a, b] \} \gg 1 \), traditional quadrature rules, such as Gauss-Legendre, Clenshaw-Curtis rules, are not efficient for the numerical computation of this type of integrals. For example, Mathematica’s NIntegrate with Method \( \rightarrow \{ \text{"ClenshawCurtisRule"} \}, \text{"Points"} \rightarrow 1000 \) and 32-digit working precision uses more than 400 000 integrand evaluations to produce a 16-digit accurate approximation to \( I_0(10, 20, 1, 4, 0) \), where

\[ I_0(a, b, \omega, p, q) = \int_a^b \frac{10^6 e^{i\omega x p}}{(x + q)^2} \, dx. \quad (2) \]

This integral can be computed more efficiently, with less than 50 integrand evaluations, by anyone of the methods for oscillatory integrals.

When \( g \) is free of stationary points, high asymptotic order is achieved in Filon-type methods [13, 21, 5, 6], the asymptotic method [13], and Levin-type methods [21, 18, 19], but for high precision results these methods require evaluation of higher order derivatives of \( f \) and \( g \) at the endpoints of integration interval. Filon-type methods, in addition, require the evaluation of moments \( \mu_k = I[x^k] \) analytically, which is not possible in general. Numerical steepest descent method [12] achieves high asymptotic order and does not require derivatives and has good stability properties. Unfortunately, it requires that the integrand be analytic in a simply connected and sufficiently large region \( \Omega \subset \mathbb{C} \) containing \([a, b]\). Moreover it requires the calculation of inverse of \( g \). For some of the methods for highly oscillatory integrals with infinite integration interval or with singular integrand, see, e.g., [4, 7, 8, 9, 10, 11, 15].

If \( g \) is strictly monotone in the interval \([a, b]\) and

\[ \delta := \delta(a, b) = \min \{ |\omega g'(a)|, |\omega g'(b)| \} \quad (3) \]

is sufficiently large, the method we present in this paper gives high-precision approximations to integral (1). Although the numerical steepest descent method requires less integrand evaluations when \( \delta > 10 \), it is not efficient for small \( \delta \).
An efficient adaptive Levin-type method

For example, for the integral \( I_0\left(1/10, 4/10, 1000, 4, q\right) \) with \( q = 0 \) or \( q = 10 \) the numerical steepest descent method requires more than 850 points to get an approximation with relative error less than \( 5 \times 10^{-13} \). Splitting the interval of integration does not decrease this number, unless another method is used. The adaptive Levin-type method we will present in Section 3, on the other hand, requires less than 50 integrand evaluations for the same accuracy. For both methods 16-digit working precision was used in Mathematica for calculations.

The main advantage of our method when compared to the Levin-type method, the asymptotic method or the moment-free Filon-type method [21] is that it does not require the computation of derivatives except the first derivative of \( g \). Moreover, it can be applied to many integrals of the form (1) even if \( g \) has stationary points. However, in the latter case the method (without modification) may require the use of more collocation points.

After giving some preliminary results for Levin and Levin-type methods in Section 2 we present our method and some of its properties in Section 3. In Section 4 we give a computer program in MATHEMATICA [25] to be used for the implementation of the method. This program can also be used for the implementation of the Levin-type method in which the derivatives are computed symbolically. In Section 5 we present some numerical results showing the performance of the proposed method.

2 Levin and Levin-type methods

If we have a function \( F(x) \) such that

\[
\frac{d}{dx}[F(x)e^{i\omega g(x)}] = f(x)e^{i\omega g(x)},
\]

we can compute \( I[f] \) trivially as \( I[f] = F(b)e^{i\omega g(b)} - F(a)e^{i\omega g(a)} \). Hence, approximating \( F \) by some function \( v \) we can find an approximation to \( I[f] \) by \( QL[f] = [v(x)e^{i\omega g(x)}]_a^b \). Expanding out the derivative on the left-hand side of (4) and canceling the exponential function appearing on both sides we obtain

\[
L[F](x) = f(x)
\]

where \( L \) is the differential operator defined by \( L[F] = F' + i\omega g'F \). In order to determine an approximation \( v \) to \( F \) we use collocation with the operator \( L \). Let \( v(x) = \sum_{k=0}^{n} c_k \psi_k(x) \) where \( \{\psi_k(x)\}_{k=0}^{n} \) is a suitable basis. Then for any set of collocation points \( a \leq x_0 < x_2 < ... < x_\nu \leq b \) with \( \nu \leq n \) we can determine \( c_k \)'s from the linear system of equations

\[
\frac{d^r L[v]}{dx^r}(x_j) = f^{(r)}(x_j), \quad r = 0, 1, ..., n_j - 1, \quad j = 0, 1, ..., \nu,
\]
where $n_j$'s are positive integers such that $n_0 + n_1 + \ldots + n_\nu = n + 1$. The method with $n = \nu$ is called a Levin collocation method or simply Levin method [18, 19, 3, 20, 22], which was introduced by David Levin [18] in 1982. If at least one of the multiplicities $n_j$ is greater than one, then the method is called a Levin-type method. A Levin-type method with $x_0 = a$, $x_\nu = b$ and $s = \min \{n_0, n_\nu\}$ is usually denoted by $Q^L_s[f]$ and has the asymptotic order $s$, i.e.,

$$I[f] - Q^L_s[f] = \mathcal{O}(\omega^{-s-1}), \quad |\omega| \to \infty$$

provided that $g'$ has no zero in $[a, b]$ and that the linear system (6) has a unique solution [21]. According to this definition, for a fixed $\omega$, the asymptotic order of the method depends only on the order of derivatives used at the end points of integration interval. Note that if there is a stationary point in the interval of integration the asymptotic order is less than $s$.

The most commonly used bases for Levin-type methods are the power basis \( \{x^k\}_{k=0}^{n} \) and the Chebyshev basis \( \{T_k(x)\}_{k=0}^{n} \), where $T_k(x) = \cos(k \arccos x)$ are Chebyshev polynomials of the first kind. Since with this definition Chebyshev polynomials are defined only in the interval $[-1, 1]$, the Chebyshev basis can be used after the change of variable

$$x = \frac{b - a}{2} t + \frac{a + b}{2}$$

in (1). For stability reasons, the Clenshaw-Curtis points

$$t_k = \cos \frac{k\pi}{\nu}, \quad k = 0, 1, \ldots, \nu$$

are usually preferred as the collocation points for the method.

3 An adaptive Levin-type method

3.1 Definition of the method

Although the Levin-type method $Q^L_s[f]$ is very efficient to obtain a high-precision approximation to the integral and can be applied to more general integrals than (1), there are two disadvantages of the method. The first one is that the linear algebraic system for the method is not well-conditioned in general. Its second disadvantage is that the method requires the evaluation of high-order derivatives of $f$ and $g$ to get a high precision approximation to (1). This problem, however, can be solved by using a suitable set of collocation points in Levin collocation method, which is the main subject of this paper.

In Levin collocation method $Q^L_1[f]$, instead of using equally spaced nodes or Clenshaw-Curtis points as collocation points we can use a set of points in the
vicinity of the end points of integration interval as in the adaptive Filon-type method proposed by A. Iserles and S. P. Nørsett in [14]. One possible choice is given below. First, using the change of variables (8) we can transform the given integral to an equivalent integral with integration interval $[-1, 1]$. Then we choose the collocation points according to the following rule:

If there is no stationary point in $[a, b]$ and $\delta \geq M$ is sufficiently large, take

$$t_k = \begin{cases} 
-1 + k/\delta, & k = 0, 1, ..., N_0 - 1 \\
0, & k = N_0 \\
1 - (n - k)/\delta, & k = N_0 + 1, N_0 + 2, ..., n,
\end{cases} \quad (10)
$$

where $\delta = \delta(a, b) = \min \{|\omega g'(a)|, |\omega g'(b)|\}$ as before and $N_0 = \lceil n/2 \rceil$; otherwise divide the integration interval into subintervals and use either the Clenshaw-Curtis points (9) or the points (10) for $Q_m^L[f]$ in each subinterval $[t_i, t_{i+1}]$ according to the value of $\delta(t_i, t_{i+1})$. Taking $M = 10$ is usually sufficient to get a 16-digit accurate result. The details will be explained in the following paragraphs.

Notice that the nodes given by (10) except $t_{N_0}$ approach the end points of integration interval as $\delta$ increases. The Levin collocation method with nodes (10) will be called a basic adaptive Levin-type method and denoted by $Q_m^{baL}$, where $m = \lfloor n/2 \rfloor$. In our numerical examples we have used either the power basis or the Chebyshev basis for this method and observed that the associated linear system with the Chebyshev basis has relatively small condition number. Here relatively means only relatively, the condition number is actually very large in both cases and increases exponentially as $n$ increases. For example, for the integral $\mathcal{I}_0(2, 4, 10^5, 4, 0)$, if we take $n = 11$ for the method, the condition number (w.r.t. Euclidean norm) of the associated linear system with the Chebyshev basis is $10^{25}$, while with the power basis it is $8 \times 10^{26}$. In each case, using Mathematica’s LinearSolve with WorkingPrecision \rightarrow 50, we obtained an approximation to the integral with relative error less than $10^{-31}$, which is smaller than the expected relative error ($\approx 10^{-25}$).

If the length of integration interval is large or $\delta$ is not sufficiently large, then a subdivision of the integration interval may be needed. For example, without subdivision, if we use Mathematica’s LinearSolve or LeastSquares with 32-digit working precision to solve the linear system, the method $Q_m^{baL}$ for any $m < 60$ yields at most 14 digit accurate approximation to the integral $\mathcal{I}_2(1/10, 4, 10^3)$, where

$$\mathcal{I}_2(a, b, \omega) = \int_a^b \frac{e^{4x}}{x^2 + 1} e^{i\omega(x^3 + x^4)} \, dx. \quad (11)$$

The reason for this is that the condition number of the associated linear system is very large. Thus, if we need, say, 16-digit accurate approximation, we have to increase the working precision and the number of collocation points. If we do not want to increase the working precision, we have to partition the
integration interval. Splitting the integral into two parts at \( x = a + (b - a)/20 \), the method (with 32-digit working precision) gives 16-digit accuracy with the use of 33 points for the first part and 21 points for the second part. The condition numbers in this case are \( \approx 10^{15} \) and \( \approx 10^{21} \), respectively.

**Remark 3.1** If we take \( \delta = \omega \) in the node set (10) we obtain the adaptive Levin-type method proposed by S. Olver in his PhD thesis [23]. Olver’s method, however, usually requires more function evaluations than \( Q_{baL} \) and is not suitable for high precision computation of some integrals (see, Table 1 and Table 2 in Section 5).

Now let \( \alpha := \alpha(a, b) = \max \{|\omega g'(a)|, |\omega g'(b)|\} \) and \( M \geq 10 \) be a preassigned number, which may depend on the accuracy required. Then, assuming that \( g \) is strictly monotone in \((a, b)\), we have three cases. The Levin collocation method defined according to the following rules will be called an **adaptive Levin-type method**, \( Q^{aL}[f] \).

**Case 1.** If \( \delta > M \), use \( Q_{baL}^m \) as explained above.

**Case 2.** If \( \alpha < M \), use \( Q_1^L \), based on an adaptive subdivision of \([a, b]\), or use Mathematica’s LevinRule.

**Case 3.** \( \delta \leq M \leq \alpha \): If \( g \) is increasing in \([a, b]\), take \( c = a + (b - a)/d, d \in [2, 50] \), and apply \( Q_1^L \) on the subinterval \([a, c]\) and \( Q_{baL}^m \) on \([c, b]\). If \( \delta(c, b) < M \), continue this process on the interval \([c, b]\). If \( g \) is decreasing, change the order of the methods taking \( c = b - (b - a)/d \). The default value of \( d \) will be taken as 10.

As an example, consider again the integral (11) with \( a = 0 \), \( b = 1 \), \( \omega = 10^4 \). Even though the left end of the integration interval is a stationary point we can still apply our algorithm. Take \( d = 10 \) in Case 3. Then, with 32-digit working precision, the method \( Q_1^L \) with 27 Clenshaw-Curtis points gives a 16-digit accurate result for the integral over \([0, 1/10]\). Since \( \delta = 340 > 30 \) on \([1/10, 1]\) we can apply \( Q_{baL}^m \) to the integral over this subinterval. With \( m = 24 \) the method gives a 17-digit accurate result for the integral over this subinterval.

If there is a stationary point \( \xi \) in \((a, b)\), then we can divide \([a, b]\) into two parts \([a, \xi] \cup [\xi, b]\) or three parts \([a, \xi - \varepsilon] \cup [\xi - \varepsilon, \xi + \varepsilon] \cup [\xi + \varepsilon, b]\) where \( \varepsilon > 0 \) is chosen to satisfy \( \alpha(\xi - \varepsilon, \xi + \varepsilon) < M \). Then we can apply the method to the integral over each of these subintervals according to the cases given above. If \( \max |\omega g'(x)| < M \) in \([a, b]\), any subdivision of the subinterval containing stationary points may not be required, as in the example given in Section 5.2.
3.2 Solution of the linear system for $Q_{baL}^m$

Since the condition number of the linear system for the method $Q_{baL}^m$ is very large in general we can not use a method based on $PA = LU$ decomposition obtained by Gaussian elimination with partial pivoting to solve the system. A second alternative is to use the truncated singular value decomposition (TSVD) method. However, numerical tests show that using TSVD does not help very much, especially if we need several hundred digit accurate approximations. A third alternative is to use the least squares method.

In the least squares method [24, Chp.4], to solve $Ax = y$ approximately, we can always find a vector $x \in \mathbb{R}^n$ that minimizes $\|Ax - y\|_2 = \|Rx - Qy\|_2$, where $Q$ is a product of Householder transformation matrices to transform $A$ to an upper triangular matrix $R$ such that $QA = R$. Since $Q$ is a product of Householder matrices, it is unitary, i.e., $Q^H Q = I$. It is easy to show that unitary matrices leave the Euclidean norm of a vector invariant, therefore the least squares method is more suitable for the linear system of the method $Q_{baL}^m$. Although it is not based on Householder transformation matrices, we have used in this paper Mathematica’s LeastSquares package to solve the linear system. Using Mathematica’s LeastSquares, however, can not solve the stability problem entirely, therefore to obtain a $d$-digit accurate approximation we need more than $d$-digit working precision(wprec) for calculations. Numerical experiments show that, for most integrals of the form (1), taking $1.5d \leq \text{wprec} \leq 2d$ is sufficient to obtain a $d$-digit accurate result. Note that, when cond$_2(A)$ is not too large, a better way of choosing wprec may be to take wprec = wprec = $\lceil 2\log_{10}(\text{cond}_2(A)) \rceil$. With this choice Mathematica’s LinearSolve can also be used, which is two times faster in general than LeastSquares.

As an illustration consider again the integral $I_2(1,3,10^4)$, given by (11). Figure 1 displays the behavior of the relative errors of $Q_{baL}^m$ for $m = 4$ and $m = 8$ as the working precision is increased from 16 digits to 160 digits. The condition numbers of the associated linear systems are $1.4 \times 10^{16}$ and $8.9 \times 10^{33}$ respectively. It is seen that the relative error in each case decreases as the working precision increases from 16 to wpcond and then remains constant for all wprec $\geq$ wpcond, as expected.

3.3 Asymptotic order of the basic adaptive Levin-type method

The relation (7) shows that the asymptotic order of the Levin-type method $Q^L_s[f]$ is equal to $s$. With exact arithmetic the same asymptotic order is expected for the basic adaptive Levin-type method $Q_{baL}^s[f]$, since for $|\omega| \to \infty$ the method becomes the Levin-type method $Q^L_s[f]$. Hence we give the following
Figure 1: Behavior of the relative error of $Q_{baL}^m$ for $n = np = \{8, 16\}$ as the working precision is increased from 16 to 160 with increment 8 for the integral $I_2(1, 3, 10^4)$.

result without proof.

**Theorem 3.2** Let $f$ and $g$ be analytic (or sufficiently smooth) in a region containing the integration interval $[a, b]$ of the integral $I[f]$ and $g'(x) \neq 0$ in $[a, b]$. Then the $n$-point basic adaptive Levin-type method $Q_{baL}^m[f]$, where $m = \lfloor n/2 \rfloor$, is of asymptotic order $m$, i.e.,

$$I[f] - Q_{baL}^m[f] = O(\omega^{-m-1}), \quad |w| \rightarrow \infty.$$  

(12)

An analytic proof of Theorem 3.2 may be obtained similar to the proof of the adaptive Filon-type method given by Iserles and Norsett in [14], but it may not be as easy as in [14]. Numerical tests confirm Theorem 3.2, as seen in Figure 2 and Figure 3 for the integral $I_2(1, 2, 10^4)$. In Figure 2 we have plotted the error of 7-point basic adaptive Levin-type method as a function of $\omega$ for $\omega = 100(100)10000$ using double MachinePrecision in Mathematica for the method. The graphs in Figure 3 shows the error behavior of the method with 8 collocation points for $\omega = 19(3)1000$ for the same integral. Observe that with double MachinePrecision the error scaled by $7^4 \omega^4$ oscillates regularly, while
with MachinePrecision irregular oscillations occur. For \( \omega = \{50, 400, 1000\} \) the condition numbers \( \approx \{9 \times 10^7, 4 \times 10^{10}, 6 \times 10^{11}\} \), respectively.

Figure 2: The error, scaled by \( 7^4 \omega^4 \) and \( \omega^5 \), for the 7-point basic adaptive Levin-type method \( Q_{3baL} \) for \( L_2(1, 2, \omega) \) and \( 100 \leq \omega \leq 10^4 \), obtained by using double MachinePrecision in Mathematica.

What happens when \( \omega \) is very large? To see this we applied the method to the same integral with \( \omega = 10^{12} \) taking \( m = 2, 3, \ldots, 6 \) in \( Q_{mL}^{ba} \). The log_10 values of the relative errors when MachinePrecision is used in Mathematica are \(-\{14.4, 15.1, 15.4, 15.4, 15.1, 15.4\}\) respectively. With 100-digit working precision we obtained \(-\{14.4, 27.8, 41.1, 53.9, 66.5, 78.9\}\).

4 A unified Mathematica code for the Levin-type and adaptive Levin-type methods

In this section we present a Mathematica program for the adaptive Levin-type method. The main advantage of the program is that it can also be used for the implementation of the Levin and Levin-type methods. The program, however, does not use automatic subdivision; therefore the user should make
Figure 3: Behavior of the error of the 8-point basic adaptive Levin-type method for $I_2(1,2,\omega), \omega = 19(3)1000$, when MachinePrecision and double Machine-Precision is used for calculations in Mathematica.
an appropriate subdivision of the integration interval as explained in Section 3.1, whenever it is necessary.

(*Subroutine for the Levin-type and adaptive Levin-type methods*)

\begin{verbatim}
Module[{r, betam, A, AA, BB, S, F, w = N[omega, wprec]}, M = Length[nu] - 1;

PB[t_, t_] := If[basis == 1, t^k, ChebyshevT[k, t];

ff[t_] := ((b - a)/2)f[(b - a)*t/2 + (a + b)/2];

gg[t_] := g[(b - a)*t/2 + (a + b)/2];

dgg[t_] := Derivative[1][gg][t];

If[test == 0, betam = Min[Abs[dgg[-1]*w], Abs[dgg[1]*w]];

While[prm*M/betam >= 1, betam = 2*betam];

If[test > 0, x[k_] := N[Cos[k*Pi/M], wprec],

Which[k < prm*M, N[-1 + k/betam, wprec],

k == Ceiling[prm*M], 0,

k > prm*M, N[1 - (M - k)/betam, wprec]],

Psi[k_, t_] := Derivative[0, 1][PB][k, t] + I*w*dgg[t]*PB[k, t];

ne[j_] := nu[[j + 1]]; S[-1] = 0; S[j_] := Sum[ne[i], {i, 0, j}];

nn = S[M] - 1; A = ConstantArray[0, {nn + 1, nn + 1}];

F = ConstantArray[0, nn + 1]; r = 0;

While[r < M + 1, Do[Do[AA[j, k] =

Limit[Derivative[0, Mod[j - S[r - 1], ne[r]]][Psi][k, t], t -> x[r]],

{k, 0, S[M] - 1}, {j, S[r - 1], S[r] - 1}];

Do[BB[j] = Limit[Derivative[Mod[j - S[r - 1], ne[r]]][ff][t],

t -> x[r]], {j, S[r - 1], S[r] - 1}];

Do[Do[A[[j, k]] = AA[j - 1, k], {j, 1, S[M]}],

{k, S[r - 1] + 1, S[r]}];

r = r + 1;]; (*sv = SingularValueList[N[A, wprec]]; con = sv[[1]]/sv[[-1]]; Print["cond2(A) = ", N[con, 3]];*)

LS = Block[{MaxExtraPrecision = 0, LeastSquares[NA, wprec], F]};

vvv[t_] := Sum[LS[[k + 1]]*PB[k, t], {k, 0, nn}];

NR = vvv[1]*Exp[I*w*gg[1]] - vvv[-1]*Exp[I*w*gg[-1]];

Print["n = ", np + ii + 2s - 2, ", Result = ", NR];

If[ii == 0, PR = NR];

(* End of subroutine aLevinTQ /A.I. Hascelik, July 2013/ *)
\end{verbatim}

The above subroutine is called by the following main program.

(*** Main Program ***)

Clear[omega, a, b, f, g, nu, wprec, prm, test];

a = 1; b = 2; omega = 100; prm = 1/2;

f[t_] := Exp[4t]; (*amplitude function*)

g[t_] := t + Exp[4t]*Gamma[t]; (*phase function*)

dg[t_] := Derivative[1][g][t]; (*derivative of g*)
In the main program, $s \geq 1$ denotes the multiplicity of nodes $a$ and $b$. If we take $s > 1$, the program computes the given integral by $Q_s^L$ using nodes \{a, (a + b)/2, b\} with multiplicities \{s, 1, s\}. The absolute error is estimated by taking the difference of two results generated by the method with $np$ and $(np + inc)$ collocation points, with $inc > 1$. One can also test the correctness of the result by using two different working precision.

The program with the above input values, for $\int_1^2 e^{4x} \exp \{100i(x + e^{4x} \Gamma(x))\} \, dx$, where $\Gamma$ is the usual Gamma function, returns the output

\begin{align*}
n = 10, \quad \text{Result} &= 0.00435354129735323908804 + 0.00202865398517716214364i \\
n = 12, \quad \text{Result} &= 0.00435354129735323908804 + 0.00202865398517716214366i \\
\text{Error} &= 1.59 \times 10^{-23}.
\end{align*}

If we take $s = 5$ in the program, it gives the value of Levin-type method with nodes \{1, 3/2, 2\} and multiplicities \{5, 1, 5\}, The output in this case is

\begin{align*}
n = 3, \quad \text{Result} &= 0.00435354129735323908804 + 0.00202865398517716214366i \\
n = 5, \quad \text{Result} &= 0.00435354129735323908804 + 0.00202865398517716214366i \\
\text{Error} &= 1.20 \times 10^{-24}.
\end{align*}

5 Numerical examples and comparison

In this section we give some numerical examples to show the performance of the proposed method $Q_{nt}^L[f]$. Note that all the computations except in subsection 5.4 in this paper were performed by \textsc{Mathematica} (v.8) on a PC with Windows 7 OS.
5.1 High precision computation of some integrals

Example 1. Consider the following integrals, taken from [8],

\[ I_3(a, b, \omega) = -\int_a^b e^{3x} \sin \omega x^x \, dx, \quad I_4(a, b, \omega) = \int_a^b x^x \cos \omega e^{3x} \, dx. \] (13)

Unfortunately, the function \( \cos \) in the integral given by Equation 20 in [8] has been mistyped and should be replaced by \( -\sin \). The result given there is an approximation to \( I_3(20, 30, 100) \).

Taking \( \text{prec}=300 \) in the Mathematica program given in the previous section for the basic adaptive Levin-type method \( Q_{baL}^L[f] \) we obtained

\[
\begin{align*}
0.001975447206079127997166845421747420086285518156156463873147 \\
01526447363262482656391307921546632371840428583100963646441856 \\
7060215773837245005888922607195618630849002900191136705601556 \\
3493739153396951324254543727042916568165761240418658978550520 \\
6151893100373077343703003026147568963184423570898955283013370 \\
01526447363262482656391307921546632371840428583100963646441856 \\
7060215773837245005888922607195618630849002900191136705601556 \\
3493739153396951324254543727042916568165761240418658978550520
\end{align*}
\]

for \( I_3(20, 30, 100) \), which agree with the results given in [8]. Note that for each result the number of collocation points were automatically determined as \( np = 23 \) by the program. Taking \( s = 11 \) in the program we also computed these integrals by the Levin-type method. The results agree to (at least) 300 decimal digits, but \( Q_{11L}^L[f] \) is about 3 and 20 times slower than \( Q_{baL}^L[f] \) for the integrals \( I_4 \) and \( I_3 \), respectively. A time comparison (in Mathematica 8) of the two methods for \( I_4 \) is displayed in Figure 4.

Example 2. As the second example consider the following integral taken from [8], the exact value of which is equal to \( \text{exp}(i \omega(e^x + x^r))/x^a \).

\[ I_5(a, b, r, \omega) = \int_a^b \left( -\frac{1 + i\omega(r x^r + x e^x)}{x^2} \right) e^{i\omega(e^x + x^r)} \, dx, \] (14)

Table 1 shows the performance of our method and some of other well-known methods, which are suitable and efficient for high-precision computation of
Figure 4: Time comparison in seconds of the Levin-type method and the basic adaptive Levin-type method for $\int_{20}^{30} x^5 \cos 100 e^{3x} \, dx$ in obtaining 50(25)300-digit accurate approximations.

oscillatory integrals, for the integral $\mathcal{I}_5(100, 120, 40, \omega)$, $\omega = 10^{10k}, k = -2(1)4$. As seen in the table, the basic adaptive Levin-type method, the adaptive Levin-type method proposed by S. Olver, the Levin-type method, and the asymptotic Filon-type method proposed by A.I. Hascelik in [8], all of which have the same asymptotic order $s$, require about the same number of integrand evaluations to get a 1000-digit accurate result. Since the integrand is analytic in a sufficiently large region containing the integration interval and $\delta$ is sufficiently large, the numerical steepest method, having the asymptotic order $2s$, requires the least integrand evaluations. The results in Table 1 confirm this, as expected.

For complex $\omega$ the method also gives accurate results. For example, with 31 points it gives a 200-digit accurate result for $\mathcal{I}_5(3, 6, 9, 10^9 + 10^{-4}i)$. For the same accuracy, Mathematica's NIntegrate's LevinRule (with "Points" $\rightarrow$ 150, "TimeConstraint" $\rightarrow$ 100) uses more than 1500 integrand evaluations [25].

Example 3. Suppose we need a 48-digit accurate approximation to

$$\mathcal{I}_6(a, b) = \int_a^b \frac{10^3 e^{4x} \sin [10^4(x^3 + x^4 e^{4x})]}{1 + x^2} \, dx$$

(15)
Table 1: Minimum number of integrand evaluations required for each method in Column 1 to yield a 1000-digit accurate approximation to $\mathcal{I}_5(100, 120, 40, \omega)$ for $\omega = 10^{10^k}, k = -2(1)4$. Here (*) denotes that time limit was exceeded (no result returned in 15 min.).

<table>
<thead>
<tr>
<th>$\log_{10} \omega$</th>
<th>-20</th>
<th>-10</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic adaptive Levin-type method</td>
<td>35</td>
<td>31</td>
<td>27</td>
<td>25</td>
<td>23</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>Levin-type method</td>
<td>35</td>
<td>31</td>
<td>27</td>
<td>25</td>
<td>23</td>
<td>21</td>
<td>19</td>
</tr>
<tr>
<td>Adaptive asymptotic Filon-type method</td>
<td>34</td>
<td>30</td>
<td>26</td>
<td>24</td>
<td>22</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Numerical steepest descent method</td>
<td>18</td>
<td>16</td>
<td>14</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Asymptotic method</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>18</td>
</tr>
<tr>
<td>Olver’s adaptive Levin-type method</td>
<td>*</td>
<td>*</td>
<td>140</td>
<td>81</td>
<td>63</td>
<td>51</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Required (minimum) number of integrand evaluations for each method in Column 1 to produce a 48-digit accurate approximation to the integral $\mathcal{I}_6(a,b)$ of Example 3.

METHOD 

<table>
<thead>
<tr>
<th>METHOD</th>
<th>$[a, b] = [0.12, 0.14]$</th>
<th>[6, 12]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic adaptive Levin-type method</td>
<td>39</td>
<td>7</td>
</tr>
<tr>
<td>Adaptive Levin-type method (S.Olver)</td>
<td>47</td>
<td>17</td>
</tr>
<tr>
<td>Levin-type method</td>
<td>47</td>
<td>7</td>
</tr>
<tr>
<td>Asymptotic method</td>
<td>*</td>
<td>6</td>
</tr>
<tr>
<td>Numerical steepest descent method</td>
<td>144</td>
<td>4</td>
</tr>
<tr>
<td>Levin collocation method</td>
<td>37</td>
<td>40</td>
</tr>
<tr>
<td>NIntegrate’s LevinRule</td>
<td>92</td>
<td>101</td>
</tr>
</tbody>
</table>

for $[a, b] = [6/50, 7/50]$ and $[a, b] = [6, 12]$. The method $Q_{m}^{baL}[f]$ gives

$\mathcal{I}_6(0.12, 0.14) \approx 1.432615065170845414327040417499177932311628980083,$
$\mathcal{I}_6(6, 12) \approx 1.197424339435837008785981124395898409696563822892 (−7).$

where $m$ is automatically determined by the program. We also obtained these results by using NIntegrate’s LevinRule, Levin-type method $Q_{s}^{L}[f]$, and the numerical steepest descent method. Table 2 shows the required number of integrand evaluations for each method in the table to obtain a 48-digit accurate approximation.
5.2 Application of the method to an integral with a highly oscillatory amplitude function

In this subsection we consider the Bessoid integral \([2, 1]\) defined as

\[
J(x, y) = \int_0^\infty J_0(yu) u \exp \left[ i (u^4 + xu^2) \right] du
\]

(16)

where \(J_0(t)\) denotes the Bessel function of order zero. Although \(f\) is oscillatory in this case our method/algorithm gives sufficiently accurate results when \(|y| \leq 100\). Note that the Bessoid integral \(J(x, y)\) plays an important role in the description of cusped focusing when there is axial symmetry present. It arises in the diffraction theory of aberrations, in the design of optical instruments and of highly directional microwave antennas and in the theory of image formation for high-resolution electron microscopes, see \([16, 17]\) and the references therein. The Bessoid integral can also be expressed by the double series

\[
J(x, y) = \frac{1}{4} \sum_{k=0}^{\infty} \sum_{j=0}^{\infty} \frac{\exp \left[ i \pi (3j - 3k + 1)/4 \right]}{2^{2k} (k!)^2 j!} \Gamma \left( \frac{j + k + 1}{2} \right) x^j y^{2k}
\]

(17)

For illustrative purpose let us compute this integral for \(\{x = -8, y = 8\}\) to 32-digit accuracy. With subdivision \([0, 9/4] \cup [9/4, 4] \cup [4, 7] \cup [7, 10^4] \cup [10^4, 10^14]\), the method \(Q_{aL}\) requires about 80+72+40+32+16=240 points, plus 260 points for the error estimation, to produce a 32-digit accurate approximation. We have omitted the interval \([10^{14}, \infty)\), since it can be shown by applying one or two integration by parts to the integral that magnitude of the integral over this subinterval is less than \(10^{-34}\). We increased the working precision in the program, to \(3*\text{prec}\), since the default value of \(wprec\) did not work for this integral due to highly oscillatory amplitude function. Our result to 34 figures is 0.1890251049716899771565691374653238, which agrees with the result given in \([8]\) to 32 significant digits. Note that Mathematica’s LevinRule computes this integral but uses about 8659 integrand evaluations. Note also that the asymptotic Filon-type method \([8]\) requires less integrand evaluations, 21+25=50 (together with the error estimation) to compute the integral over \([7, \infty)\).

5.3 Application of the method to an integral with a non-analytic integrand

Consider

\[
I_1(a, b, c, \omega) = \int_a^b \frac{100(x - 13/4)^c}{x^2 + 100} e^{i \omega(x^{10} + x^c)} \, dx.
\]

(18)
The numerical steepest descent method cannot be applied to $I_1(2, 5, 5, 10^5)$ without splitting the interval of integration or approximating $f$ by a polynomial. The 15-point adaptive Levin-type method with 80-digit working precision computes this integral to 53-digit accuracy. This is the maximum reachable accuracy without subdivision, however. Increasing the working precision or the number of collocation points does not increase the accuracy.

On the other hand, if the subdivision $[2, 13/4] \cup [13/4, 5]$ is used, the basic adaptive Levin-type method with the use of 200-digit working precision and $37 + 31 = 68$ collocation points gives a 150-digit accurate approximation to the integral. The numerical steepest method is more efficient in this case, since it requires 40 integrand evaluations when the same number of nodes is used at both ends of the interval. Note that NIntegrate’s LevinRule with the options \{Points=100, TimeConstraint=80\}, with or without the above subdivision, uses more than 3000 integrand evaluations for the same accuracy; the exact number together with the error estimation is 6090. However, we should note that everything in LevinRule is automatic: Automatic singularity handling, automatic subdivision, automatic error estimation, etc.

### 5.4 MatLab results

We have tested the basic adaptive Levin-type method on MatLab using a small number of collocation points (less than 8 points). Our numerical results show that the method gives 8-14 significant-digit accurate approximations in general. Fig. 5 shows the behavior of the relative error scaled by $10^9$ and the absolute error scaled by $7^2\omega^3$ in the results generated by $Q_{200}$ in MatLab for the integral $I_2(1, 2, \omega)$ for $\omega = 1000(3)4000$.

### 6 Conclusion

We have presented a derivative free adaptive Levin-type method for computing highly oscillatory integrals of the form (1). We have tested the method on many integrals. The test results show that the basic adaptive Levin-type method is comparable with the existing methods and has the same asymptotic order as the corresponding Levin-type method. Although the numerical steepest descent method is superior in general than the presented method when the integrand is analytic, our method is more efficient for some integrals and can give sufficiently accurate approximations for integrals with only several times differentiable amplitude functions, without splitting the integral. Moreover it does not require finding the inverse of $g$ analytically or numerically, while the numerical steepest descent method does require the inverse of $g$ at collocation points. A disadvantage of the method is that it requires about $2d$-digit working precision for a $d$-digit accurate approximation. However, this may
Figure 5: Behavior of the relative error scaled by $10^9$ and the absolute error scaled by $7^2\omega^3$ in the results generated by $Q_{2baL}^L$ in MatLab for $I_2(1,2,\omega)$, $\omega = 1000(3)4000$.

not be a significant problem if the method is implemented in Mathematica, Maple, etc. The Mathematica program developed in this paper for the method, which can also be used for Levin-type methods, is probably the first computer program for automatic implementation of Levin-type methods.

We have also tested the basic adaptive Levin-type method on MatLab using a small number of collocation points (less than 8 points). Our test results show that the method produces 8-14 digit accurate approximations in general. High-precision results may be obtained in MatLab if MatLab’s variable precision arithmetic vpa is used for calculations, or in ARPREC [2, 1] which is an arbitrary precision computation package in FORTRAN/C++.

References

An efficient adaptive Levin-type method


**Received: June 15, 2014**