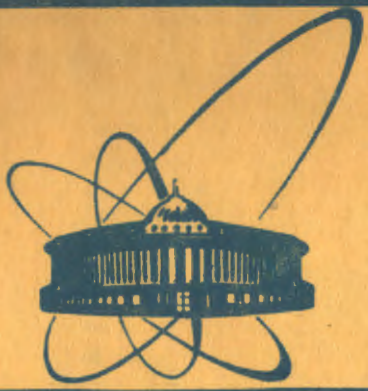

сообщения
объединенного
института
ядерных
исследований
дубна

11-83-551

Р.В.Малышев

ИНТЕГРИРОВАНИЕ ОСЦИЛЛИРУЮЩИХ ФУНКЦИЙ НА БЕСКОНЕЧНОМ ИНТЕРВАЛЕ

1983

ВВЕДЕНИЕ

Многие задачи математической физики приводят к вычислению интегралов вида

$$I=\int_a^{\infty} f(x)J(wx)dx, \qquad (I)$$

где $J(wx)$ – осциллирующая функция, например, тригонометрические функции $Sin(wx), Cos(wx)$ – преобразование Фурье или бесселевы функции $J_0(wx), J_1(wx)$ – преобразование Ханкеля.

В некоторых случаях w может быть большой величиной. Вычисление таких интегралов традиционными методами встречает значительные трудности из-за взаимного погашения положительных и отрицательных вкладов подынтегрального выражения. Эти и другие трудности заставляют искать специальные методы для вычисления подобных интегралов.

Особую группу методов составляют специализированные квадратурные формулы. Классическим примером здесь стала формула Филона (1928 год)[1].

В дальнейшем количество таких формул было значительно умножено. До настоящего времени продолжаются попытки их распространения на бесконечный интервал.

Основные недостатки таких формул – применимость их на конечном интервале и ограничение специальным видом осциллирующего множителя, обычно тригонометрическими функциями.

Вычисление интегралов на бесконечном интервале

Для интегрирования осциллирующих функций на бесконечном интервале получили распространение методы, основанные на представлении интеграла бесконечным рядом с последующим его суммированием различными методами.

Особо следует отметить появившийся в недавнее время в современной геофизике метод интегрирования, использующий представление интеграла в виде свертки с применением в дальнейшем методов теории цифровых фильтров[12].

Представим интеграл (I) бесконечным рядом. Пусть $\{z_i\}$ ($i=1,2,3,\ldots$) – множество неубывающих действительных чисел:

$a \le z_1 \le z_2 \le z_3 \le \ldots$. Разобьем интервал интегрирования на подынтервалы, и тогда:

$$I = \sum_{i=1}^{\infty} I_i \quad , \quad \text{где} \qquad (2)$$

$$I_i = \int_{z_i}^{z_{i+1}} f(x) J(wx) dx. \qquad (3)$$

После вычисления интегралов (3) интеграл (I) представлен числовым рядом (2).

В качестве множества $\{ z_i \}$ обычно берут множество нулей функции $J(wx)$ на интервале (a, ∞). В таком случае ряд (2) получится знакопеременным. Это облегчает суммирование ряда и дает простую оценку погрешности его частных сумм.

Для вычисления нулей обычно не требуется большой точности. Это множество вовсе не обязано содержать все нули функции, вполне достаточно, чтобы разбиение давало знакопеременный ряд (2). В некоторых случаях и это требование бывает лишним.

Создан ряд методов, основанных на аналитическом суммировании ряда (2) без предварительного интегрирования (3). Более подробный обзор подобных методов дан в статье[13].

Не будем останавливаться на таких методах, т.к. они применимы обычно в отдельных частных случаях.

В дальнейшем подробно рассмотрим методы, в которых выполняется интегрирование (3), а затем суммируется числовой ряд (2).

Многообразие алгоритмов подобного типа можно получить, варьируя методы вычисления интегралов (3).

Задача интегрирования на коротких интервалах не представляет сложности, следует искать лишь наиболее экономичные методы. Этот вопрос в дальнейшем обсуждаться не будет. Примеры таких экономичных методов можно найти в[7] и[14], хотя они не позволяют вычислять интегралы с автоматическим выбором шага по заданной точности.

Новое многообразие алгоритмов получим, если будем рассматривать различные методы суммирования ряда (2). Сходимость этого ряда может оказаться очень медленной, и тогда целесообразно применение различных методов ускорения сходимости рядов. Простое суммирование ряда (2) может оказаться невозможным и при быстрой его сходимости. Допустим, сумма ряда I на несколько порядков меньше некоторых его членов $|I| \ll |I_i|$, тогда простое суммирование ряда приведет к большой потере точности. Такие интегралы нельзя вычислять надежно ни одним из известных методов. Больше всего в этом случае подходит, по-видимому, простой метод Лонгмана[4], в котором преобразование Эйлера вычисляется через разности.

Перечень алгоритмов в зависимости от методов суммирования

I) Преобразование Эйлера впервые было предложено для суммирования ряда (2) Лонгманом[3];

2) поведение функции $f(x)$ в начале интервала интегрирования иногда вынуждает представлять ряд (2) в виде двух частей:

$$I = \sum_{i=1}^{N} I_i + \sum_{i=N+1}^{\infty} I_i \quad ,$$

где первые N членов просто суммируются, а к остатку можно применить преобразование Эйлера, этот метод был использован для интегрирования в[5];

3) нелинейное преобразование Шэнкса[2] использовано в статье[6];

4) в работе[7] применен частный случай преобразования Шэнкса - преобразование Эйткена;

5) в работе[8] используется так называемый ε-алгоритм - удобная форма вычисления преобразования Шэнкса;

6) интерполяционные методы улучшения сходимости рядов, разрабатываемые В.И.Крыловым[9], до настоящего времени никем не были испытаны при вычислении интегралов.

Преобразование Шэнкса

Сильным механизмом ускорения сходимости последовательности частных сумм ряда (2) служит оператор Шэнкса e_k[2].

Последовательность $\{ \sigma_{k,n} \}$ $(n=k,k+1,\ldots)$ получается как

$$\{ \sigma_{k,n} \} = e_k \{ s_n \} . \qquad (4)$$

Члены последовательности находятся как отношение двух определителей порядка $(k+1)$, которые выражаются в форме:

$$D_{k,n} = \begin{vmatrix} 1 & 1 & \ldots & 1 \\ \Delta s_{n-k} & \Delta s_{n-k+1} & \ldots & \Delta s_n \\ \Delta s_{n-k+1} & \Delta s_{n-k+2} & \ldots & \Delta s_{n+1} \\ \ldots & \ldots & \ldots & \ldots \\ \Delta s_n & \ldots & \ldots & \Delta s_{n+k-1} \end{vmatrix} , \qquad (5)$$

$$D_{k,n}(S_n) = \begin{vmatrix} S_{n-k} & S_{n-k+1} & \cdots & S_n \\ \Delta S_{n-k} & \Delta S_{n-k+1} & \cdots & \Delta S_n \\ \Delta S_{n-k+1} & \Delta S_{n-k+2} & \cdots & \Delta S_{n+1} \\ \cdots & \cdots & \cdots & \cdots \\ \Delta S_{n-1} & \Delta S_n & \cdots & \Delta S_{n+k-1} \end{vmatrix}, \qquad (6)$$

где

$$\Delta S_n = S_{n+1} - S_n ,$$

$$\sigma_{k,n} = e_k(S_n) = D_{k,n}(S_n)/D_{k,n}. \qquad (7)$$

Формула (7) называется преобразованием Шэнкса порядка К.

Случай $K=1$ эквивалентен хорошо известному преобразованию Эйткена, или Δ^2-преобразованию:

$$\sigma_{1,n} = \frac{\begin{vmatrix} S_{n-1} & S_n \\ \Delta S_{n-1} & \Delta S_n \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ \Delta S_{n-1} & \Delta S_n \end{vmatrix}} = \frac{S_{n-1}S_{n+1} - S_n^2}{S_{n-1}+S_{n+1}-2S_n} . \qquad (8)$$

Условием выполнимости преобразования является условие

$$D_{k,n} \neq 0 \qquad (9)$$

при всех достаточно больших n. Числитель формулы (8) выражает, как легко видеть, отклонение общего члена последовательности от среднегеометрического, а знаменатель соответствует отклонению общего члена от среднеарифметического.

Таким образом, преобразование неприменимо (не выполнено условие (9)) к последовательностям, общий член которых по поведению близок к арифметической прогрессии.

Можно показать[2], что преобразование Шэнкса порядка к переводит последовательность

$$S_n = q_0 + \sum_{i=1}^{k} c_i q_i^n \qquad (10)$$

в константу q_0. На этом основано использование преобразования как ускорителя сходимости последовательностей, близких к линейной комбинации К-геометрических прогрессий.

Прямое вычисление определителей (5),(6) требует значительных затрат труда и может привести к вычислительной неустойчивости, в особенности для преобразований высокого порядка.

Винном в статье[10] предложен эквивалент преобразования Шэнкса в виде рекуррентных формул, получивший название "ε-алгоритм", который определяется следующими равенствами:

$$\varepsilon_n^{(-1)} = 0, \quad \varepsilon_n^{(0)} = S_n , \qquad (11)$$

$$\varepsilon_{n+1}^{(p)} = \varepsilon_{n+1}^{(p-2)} + \left[\varepsilon_{n+1}^{(p-1)} - \varepsilon_n^{(p-1)} \right]^{-1} .$$

Показано[10], что

$$\varepsilon_0^{(2k)} = e_k \left\{ S_0, S_1, S_2, \ldots S_{2k} \right\} = \sigma_{k,k} . \qquad (12)$$

Известно, что этот алгоритм устойчив и легко реализуется на ЭВМ.

Необходимо добавить, что величина в квадратных скобках формулы (11) есть не что иное, как определитель $D_{k,n}$, неравенство нулю которого служит условием выполнимости преобразования Шэнкса.

Программы SUMIR и CSUMIR

Программа SUMIR (суммирование интегралов и рядов) предназначена для вычисления интегралов от осциллирующих функций вида (1) методом (2), (3) с последующим применением ε-алгоритма (11) для ускорения сходимости ряда (2).

SUMIR можно использовать для суммирования рядов (2), если вместо вычисления интегралов (3) соответствующая подпрограмма будет вычислять последовательные члены ряда (2).

При этом следует заметить, что сходимость как ряда (2), так и интеграла (1) существенного значения не имеет.

Программа выдает результат во всех случаях, а истолкование его предоставляется пользователю. Для рядов и интегралов, сходящихся в обычном смысле, программа вычисляет их истинную сумму.

За подробным разъяснением вышесказанного удобнее обратиться к соответствующей литературе по ускорению сходимости рядов и индуцированию сходимости расходящихся рядов[2],[9].

С несколько меньшими удобствами программу можно применять для вычисления пределов последовательностей (представляя последовательности в виде рядов).

Начав перечислять различные нестандартные применения программы, нужно добавить, что подынтегральная функция не обязана быть осциллирующей, т.к. множество $\{z_i\}$ не обязательно есть множество нулей функции.

Для вычисления очередной частной суммы ряда (2) $S_{n+1}=S_n+I_n(z_n,z_{n+1})$ программа SUMIR обращается к подпрограмме EXTERN за вычислением $I_n(z_n,z_{n+1})$. Эту информацию и формат обращения к EXTERN можно использовать для нестандартных применений SUMIR.

Учитывая сказанное выше, можно вычислять интегралы и на конечном интервале (a, b):

$$\int_a^b f(x)dx = \int_a^\infty f(x)dx - \int_b^\infty f(x)dx.$$

Аналогичным приемом была получена и известная формула Лонгмана для интегрирования осциллирующих функций на конечном интервале /4/.

Имеется вариант программы, в котором функция f(x) предполагается комплексной функцией действительного аргумента. Программа имеет наименование CSUMIR.

Все нестандартные применения допустимы и в обоих вариантах программы.

Формат обращения к SUMIR

Программа предназначена для вычисления интегралов I вида:

$$I=SUM\pm ERROR, \quad I=\int_A^\infty F(x)J(FRQ*x+PHASE)dx,$$

где J(FRQ*x+PHASE) - осциллирующая функция аргумента X с частотой FRQ и начальной фазой PHASE;

F(x) - произвольная действительная функция, нули которой не принимаются во внимание.

Формат обращения:

```
CALL SUMIR(EXTERN,A,FRQ,PHASE,EF,ZF,NZ,V,EPS,
           SUM,ERROR,N,K)
```

где

A - нижний предел интегрирования (верхний предел всегда предполагается бесконечным);

FRQ,PHASE - названные параметры функции J;

ZF - упорядоченная по возрастанию последовательность нулей функции J(x).

Программа сама пересчитывает нули функции J(x) в нули функции J(FRQ*x+PHASE);

NZ - размерность множества нулей;

V(NZ,2) - двумерный массив указанной размерности;

EPS - допустимая величина относительной погрешности вычисленного значения интеграла;

SUM - вычисленное значение интеграла;

ERROR - величина абсолютной погрешности интеграла.

Параметры N и K - целые числа, задавать которые не требуется, их вычислит программа.

N - количество использованных членов ряда (2);

K - порядок использованного преобразования Шэнкса;

EXTERN - наименование подпрограммы, используемой для вычисления интегралов на подынтервалах (z_i, z_{i+1}). Формат обращения к ней должен быть следующим:

```
CALL EXTERN(A,B,EF,AEPS,SUM,ERR,AS)
```

где

A,B - пределы интегрирования;

AEPS - величина допустимой относительной погрешности;

SUM - вычисленное значение интеграла;

ERR - величина абсолютной погрешности интеграла;

AS - абсолютное значение интеграла |SUM|;

EF - наименование программы-функции, вычисляющей подынтегральное выражение (параметр EF в EXTERN и SUMIR один и тот же).

Примечание:

В качестве EXTERN можно использовать программы KGAUSS или ERSIMP, имеющие необходимый формат обращения.

Формат обращения к CSUMIR

Программа CSUMIR предназначена для вычисления интегралов I вида:

$$I=SUM\pm ERROR, \quad I=\int_A^\infty F(x)J(FRQ*X+PHASE)dx,$$

где

J(FRQ*X+PHASE) осциллирующая функция с частотой FRQ и начальной фазой PHASE;

F(x)- произвольная комплексная функция действительного аргумента X. Нули этой функции не используются в вычислениях.

Формат обращения:

```
CALL CSOMIR(EXTERN,A,FRQ,PHASE,EF,ZF,NZ,V,EPS,SUM,ERROR,N,K).
```

Формат обращения к подпрограмме EXTERN :

```
CALL EXTERN(A,B,EF,AEPS,SUM,ERR,AS).
```

Значения параметров с одинаковыми наименованиями те же, что и при описании программы SUMIR в действительном варианте.

Разница состоит лишь в том,что некоторые параметры описываются как величины комплексные: SUM,ERROR,ERR,AS= |Re(I)| +i |Ym(I)| .

Программа-функция EF выполняется как

```
COMPLEX FUNCTION EF(x) .
```

В качестве EXTERN можно использовать на этот раз программы CKGAUS и CSIMPS , созданные специально для этой цели.

Примечания и примеры

Когда программа SUMIR применяется для суммирования ряда или вычисления предела последовательности, подпрограммы-функции EF не требуется. Вместо нее нужно написать фиктивное наименование. Для вычисления общего члена ряда можно использовать SUBROUTINE EXTERN.

Роль множества $\{z_i\}$ (параметр ZF) успешно может выполнять последовательность действительных чисел $\{0.,1.,2.,3.,\dots\}$.

В этом случае при очередном входе в EXTERN A=n , B=n+1 , которые можно использовать для вычисления общего члена ряда.

При выходе из EXTERN необходимо положить ERROR=0 или равной абсолютной погрешности вычисления SUM , а AS= |SUM|

В качестве примера суммирования ряда возьмем медленно сходящийся ряд Лейбница:

$$\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots) = 3,1415926536.$$

Число π с одиннадцатью знаками получено суммированием только N=16 членов ряда с применением преобразования Шэнкса.

В качестве второго примера возьмем наиболее нестандартное применение программы SUMIR . Здесь монотонно возрастающая функция интегрируется на конечном интервале. Интеграл на конечном интервале представим в виде разности двух несобственных расходящихся в обычном смысле интегралов. Для этих интегралов найдем их обобщенные суммы (s -суммы):

$$\int_0^1 e^x dx = \int_0^\infty e^x dx - \int_1^\infty e^x dx = (-1,0000000000) - (-2,7182818285) = 1,7182818285;$$

ERROR=0,2 10^{-13}.

Для вычисления каждого из этих интегралов потребовалось N=4 подынтервала при K=1. Принималось $z_{i+1} - z_i = 0,1$.

При суммировании расходящихся интегралов следует быть внимательным.

Большие значения N должны настораживать, а если при этом и K велико (K сравнимо с N), то результат неверен. В этом случае и ERROR будет велика.

Пример "невычислимого" интеграла позаимствуем из /13/:

$$I = \int_0^\infty (x^2 + \frac{1}{4})^{-1} \cos(100x)\,dx = \pi e^{-50} = 6,059\ 10^{-22}.$$

Программа SUMIR и другие известные методы дают в результате вычислений случайный результат, образованный из ошибок округления чисел на ЭВМ. Программа SUMIR дает: $I = 2,64 \cdot 10^{-15}$, $ERROR = 4,5 \cdot 10^{-15}$. При этом N=NZ=500, K=249 . Для вычисления таких интегралов необходима арифметика с многократной точностью. Эти и другие многочисленные примеры показывают, что метод имеет более широкое применение, чем предполагалось.

ЛИТЕРАТУРА

1. Filon L.N.G.,Proc. Roy. Soc.,Edinburgh, 1928, v.49,p.38.
2. Shanks D., J.Math. and Phys.,1955,v.34,No1,p.1-42.
3. Longman J.M.,Proc. Cambridge Philos. Soc.,1956,v.52,No4,p.764-768.
4. Longman J.M.,Mathematics of Computation,1960,v.14,No 69,p.53-59.
5. Малышев Р.В. Совместный научный сборник ОИЯИ (Дубна, СССР) и ЦИФИ (Будапешт,Венгрия), 1974, KFKI-74-34, с.43-56.
6. Alaylioglu A.,Evans G.A.,Hyslop J.,J.Computational Physics, 1973,v.13,p.433.
7. Hillion P.,Nurdin G.,J.Computational Physics,1977,v.23,No1,p.74-81.
8. Chisholm R., Genz A.,Rowlands A., J.Computational Physics, 1972,v.10,p.284.
9. Крылов В.И., Бобков В.В., Монастырный П.И. Вычислительные методы высшей математики, изд. "Вышэйшая школа", Минск,1975,с.442-593.
10. Wynn P., Math. Tables and other Aids to Computation,1956, v.54, p. 91-96.
11. Wynn P.,Computer Journal, 1971,v.14,No4,p.437-441.
12. Johanson H.,Sorensen K., Geophysical Prospecting, 1979, v.27, No4,p.876-901.
13. Blakemore M.,Evans A.,Hislop J.,J.Computational Physics,1976,v.22, No 3,p.352-376.

Рукопись поступила в издательский отдел
28 июля 1983 года.

Малышев Р.В.

11-83-551

Интегрирование осциллирующих функций на бесконечном интервале

Дан краткий перечень применяющихся методов интегрирования осциллирующих функций.

Детально рассмотрен метод, использующий нелинейное преобразование Шэнкса и его эквивалент, именуемый ε-алгоритмом.

Этот алгоритм реализован в двух вариантах программы на фортране, предназначенных для интегрирования действительных и комплексных функций.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1983

Malyshev R.V.

11-83-551

Oscillatory Function Integration on Infinite Range

Review of oscillatory function integration methods is presented. A method based on nonlinear Shanks transformations and its equivalent (so called ε-algorithm) are discussed in detail. Two versions of Fortran programs for integration of real and complex functions have been realised.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1983

Перевод автора