НАЦИОНАЛЬНЫЙ АВИАЦИОННЫЙ УНИВЕРСИТЕТ.
ИНСТИТУТ АЭРОНАВИГАЦИИ
КАФЕДРА СИСТЕМ УПРАВЛЕНИЯ ЛЕТАТЕЛЬНЫМИ АППАРАТАМИ

Воронов С.И.

Аппроксимация таблично заданных функций дискретным преобразованием Фурье

(Методы и алгоритмы).

УЧЕБНОЕ ПОСОБИЕ
(Редакция от 2017г.)

ОГЛАВЛЕНИЕ

АННОТАЦИЯ

Материал данного пособия ориентирован на студентов второго курса. Основной задачей пособия является показать логику аппроксимации функций рядами Фурье, рассмотреть простейшие методы для работы с дискретным преобразованием Фурье для табличных функций с постоянным шагом по аргументу, а также показать простейшие алгоритмы реализации таких методов. Кроме того, данный материал призван связать учебный материал дисциплин «Высшая математика» и «Программирование», которые читаются студентам в первого и второго курсов. Для более полного усвоения материала, пособие дополнено компьютерной программой, которая позволяет выполнять множество численных экспериментов для различных табличных функций.

Последняя редакция 03.09.2017

Воронов С.И.

1. Введение

Представление различных функций действительной переменной с помощью рядов, достаточно распространенный математический прием, который позволяет некоторую функцию *f(x)* заменить (смоделировать) эквивалентным рядом. В качестве составляющих такого ряда, как правило, выбирают набор хорошо исследованных функций $v_n(x)$. Например:

$$f(x) \cong \sum_{n=0}^{\infty} c_n v_n(x) \qquad (1.1)$$

Основными задачами, которые необходимо решить для выполнения такой замены, являются задачи:

- Выбора набора функций $v_n(x)$
- Вычисления коэффициентов c_n, при которых достигается максимальная точность моделирования.

Задача выбора набора функций, является задачей, которую достаточно сложно сформулировать формальными методами. Как правило, выбор осуществляется исходя из множества особенностей и критериев.

В тоже время можно предложить некоторую логику, которая позволяет прояснить особую роль тригонометрических функций при выборе искомого набора $v_n(x)$.

Суть этой логики базируется на конечной скорости распространения взаимодействий между любыми объектами принадлежащими объективной реальности. Поскольку практически все взаимодействия осуществляются через физические поля, то с учетом конечной скорости распространения (пределом которой считается скорость света), их можно рассматривать как процессы переноса взаимодействия в пространстве и времени или формализовать как волны различного вида. В свою очередь, простейшим описанием некоторой волны, является периодический процесс, записанный через тригонометрические функции, например:

$$v_n(x) = c_n \sin(nx + \varphi_n) \quad (1.2)$$

К сожалению, такая запись кроме коэффициента c_n, дополнительно включает фазу процесса φ_n, которая также подлежит вычислению. Такая форма явным образом нарушает структуру ряда (1.1). В этом смысле, более удобным для составления набора функций $v_n(x)$, является представление волны путем формального разделения фазовых - φ_n и частотных - nx свойств этой волны:

$$v_n(x) = a_n \sin(nx) + b_n \cos(nx) \quad (1.3)$$

Если коэффициенты a_n, b_n удается тем или иным способом вычислить, то достаточно просто восстановить амплитудную и фазовую характеристики волны с помощью известной тригонометрической зависимости:

$$c_n = \sqrt{a_n^{\,2} + b_n^{\,2}}, \quad \varphi_n = \arccos(a_n / \sqrt{a_n^{\,2} + b_n^{\,2}}) \quad \| \quad \varphi_n = \arcsin(b_n / \sqrt{a_n^{\,2} + b_n^{\,2}}) \text{ . (1.4)}$$

Такой подход, позволяет сохранить достаточно простую структуру ряда, а сформулированный таким образом ряд получил название ряда Фурье:

$$f(x) \cong \sum_{n=0}^{\infty} c_n v_n(x) = \sum_{n=0}^{\infty} c_n \sin(nx + \varphi_n) = \sum_{n=0}^{\infty} [a_n \sin(nx) + b_n \cos(nx)] \quad (1.5)$$

Особое место, при разложении функций в ряды, занимают вопросы сходимости ряда. Ответы на эти вопросы позволяют определить:

- Классы функций, для которых возможно выполнить разложение в некоторый ряд.
- Длину ряда, необходимую для достаточной точности моделирования функции таким рядом.

Поскольку исследование вопросов сходимости рядов является обширным разделом математики, мы ограничимся только основными выводами этих исследований в отношении рядов Фурье:

- Ряд Фурье сходится к функции *f(x)*, если функция отвечает условиям Дирихле:
  - Функция *f(x)* всюду однозначна, конечна и кусочно – непрерывна.
  - Функция *f(x)* имеет ограниченное число максимумов и минимумов.
- В точках разрыва такой функции *f(x)*, ряд Фурье сходится значению *f(d)*, которое является средним для значений, взятых слева и справа от точки разрыва *(x = d)*.

$$f(d) = \frac{f(d-0) + f(d+0)}{2}$$

В некоторых источниках [Г. Корн 4.11-2a] условие сходимости ряда Фурье записывают в интегральной форме, то есть, если на интервале [x_b , x_e] существует интеграл функции *f(x)* вида:

$$\int_{Xb}^{Xe} | f(\tau) | \, d\tau$$

то ряд Фурье сходится во всех точках внутри этого интервала.

2. Коэффициенты ряда Фурье.

Для нахождения коэффициентов a_n, b_n ряда Фурье функции *f(x)*, удовлетворяющей условиям Дирихле, воспользуемся его определением в форме следующей суммы:

$$f(x) = \sum_{n=0}^{\infty} [a_n \sin(n\omega x) + b_n \cos(n\omega x)], \quad \text{где:} \quad \omega = \frac{2\pi}{T}, \quad n \in (0, 1, 2, \ldots). \qquad (2.1)$$

Поскольку функции sin и cos являются периодическими функциями с периодом 2π, то моделируемая на их базе функция *f(x)*, также будет иметь периодический характер. На первый взгляд, указанное свойство существенно снижает область применения рядов Фурье.
Однако, в инженерной практике функции *f(x)* наиболее часто рассматриваются только на некотором участке, который задан начальным - x_b и конечным – x_e значениями независимой переменной *x*. Если период функций sin и cos привязать к этому участку, то в области рассмотрения функция *f(x)* и ее ряд Фурье становятся взаимозаменяемыми. Такую привязку к периоду 2π достаточно просто выполнить, если определить:

$$\omega x = \frac{2\pi}{(x_e - x_b)} x = \frac{2\pi}{T} x, \quad \text{где:} \quad T = (x_e - x_b) \text{ и } x \in [x_b, x_e]. \qquad (2.2)$$

Как следствие, изменение значения независимой переменной x в пределах участка *T* приводит к тому, что соответствующие значения ωx изменяются в пределах 2π.

Метод, которым вычисляются коэффициенты, можно представить следующими шагами:

- Шаг 1. Умножаем обе части равенства (2.1) на множитель $\sin(k\omega x)$ или $\cos(k\omega x)$ где $k \in (0,1,2,\ldots)$,
- Шаг 2. Интегрируем обе части умноженного равенства (2.1) в границах $x_b - x_e$,
- Шаг 3. Преобразуем результат к удобной форме.

В соответствии с предложенным методом, нам потребуется вычисление целого ряда вспомогательных интегралов.
Представим эти интегралы в виде сводной таблицы.

2.1. Сводная таблица вспомогательных интегралов

Подробно вычисление вспомогательных интегралов рассмотрено в приложении А. В данном случае, когда $n>=0$ и $k>=0$, такая таблица будет иметь вид:

$$\int_{Xb}^{Xe}\cos(n\omega x)dx=\begin{cases}x_e-x_b=T & (n=0)\\ 0 & (n>0)\end{cases} \qquad (2.1.1)$$

$$\int_{Xb}^{Xe}\sin(n\omega x)dx=0 \quad (n=0)\,\|\,(n>0) \qquad (2.1.2)$$

$$\int_{Xb}^{Xe}\sin(k\omega x)\sin(n\omega x)dx=\begin{cases}0 & (k=0,n>0)\,\|\,(n=0,k>0)\,\|\,(k=n=0)\\ 0 & (k\neq n,\quad k>0,\quad n>0)\\ T/2 & (k=n,\quad k>0,\quad n>0)\end{cases} \qquad (2.1.3)$$

$$\int_{Xb}^{Xe}\sin(k\omega x)\cos(n\omega x)dx=0 \qquad (2.1.4)$$

$$\int_{Xb}^{Xe}\cos(k\omega x)\cos(n\omega x)dx=\begin{cases}0 & (n=0,k>0)\,\|\,(n>0,k=0)\\ T & (n=k=0)\\ T/2 & (k=n,\quad k>0,n>0)\\ 0 & (k\neq n,\quad k>0,n>0)\end{cases} \qquad (2.1.5)$$

где: $k\in(0,1,2,\ldots)$, $n\in(0,1,2,\ldots)$, $\omega=\dfrac{2\pi}{x_e-x_b}$, символ $\|$ следует читать как операцию «ИЛИ». Все эти интегралы вычисляются для участка заданного начальным - x_b и конечным – x_e значениями независимой переменной x.

2.2. Вычисление коэффициентов b_k

В соответствии с предложенным методом, выполним над равенством (2.1) первый шаг:

$$f(x)\cos(k\omega x)=\sum_{n=0}^{\infty}[a_n\sin(n\omega x)\cos(k\omega x)+b_n\cos(n\omega x)\cos(k\omega x)]$$

После выполнения второго шага получим:

$$\int_{Xb}^{Xe}f(x)\cos(k\omega x)dx=\int_{Xb}^{Xe}\sum_{n=0}^{\infty}[a_n\sin(n\omega x)\cos(k\omega x)+b_n\cos(n\omega x)\cos(k\omega x)]dx$$

Поменяв порядок интегрального и обычного суммирования, получим:

$$\int_{Xb}^{Xe}f(x)\cos(k\omega x)dx=\sum_{n=0}^{\infty}[a_n\underbrace{\int_{Xb}^{Xe}\sin(n\omega x)\cos(k\omega x)dx}_{2.2.1-A}+b_n\underbrace{\int_{Xb}^{Xe}\cos(n\omega x)\cos(k\omega x)dx}_{2.2.1-B}] \qquad (2.2.1)$$

В соответствии с выражением (2.1.4) интеграл (2.2.1-А) обращается в ноль, а выражение (2.2.1) приобретает вид:

$$\int_{Xb}^{Xe} f(x)\cos(k\omega x)dx = \sum_{n=0}^{\infty} b_n \int_{Xb}^{Xe} \cos(n\omega x)\cos(k\omega x)dx \qquad (2.2.2)$$

Данное выражение удобно рассматривать в форме нескольких случаев.

Случай А. Пусть *k=0*. Для удобства преобразований частично раскроем сумму в выражении (2.2.2) выделив из нее элемент с индексом *n=0*:

$$\int_{Xb}^{Xe} f(x)\cos(0)dx = b_0 \underbrace{\int_{Xb}^{Xe} \cos(0)\cos(0)dx}_{2.2.3-A} + \sum_{n=1}^{\infty} b_n \underbrace{\int_{Xb}^{Xe} \cos(n\omega x)\cos(0)dx}_{2.2.3-B} \qquad (2.2.3)$$

Поскольку косинусы нулевых углов равны единице, выражение (2.2.3) приобретает вид:

$$\int_{Xb}^{Xe} f(x)dx = b_0 \underbrace{\int_{Xb}^{Xe} 1\cdot dx}_{2.2.4-A} + \sum_{n=1}^{\infty} b_n \underbrace{\int_{Xb}^{Xe} \cos(n\omega x)dx}_{2.2.4-B} \qquad (2.2.4)$$

Применяя к (2.2.4) выражение (2.1.1) легко заметить, что интеграл (2.2.4-А) будет равен T, а интегралы (2.2.4-В) примут нулевое значение. В итоге мы получим:

$$\int_{Xb}^{Xe} f(x)dx = b_0 T \quad \text{или} \quad b_0 = \frac{1}{T}\int_{Xb}^{Xe} f(x)dx \qquad (2.2.5)$$

Случай Б. Пусть *k>0*. Для удобства преобразований частично раскроем сумму в выражении (2.2.2) выделив из нее элемент с индексом *n=k*:

$$\int_{Xb}^{Xe} f(x)\cos(k\omega x)dx = \sum_{n=0}^{n=k-1} b_n \underbrace{\int_{Xb}^{Xe} \cos(n\omega x)\cos(k\omega x)dx}_{2.2.6-A}$$

$$+ b_k \underbrace{\int_{Xb}^{Xe} \cos(k\omega x)\cos(k\omega x)dx}_{2.2.6-B} \qquad (2.2.6)$$

$$+ \sum_{n=k+1}^{\infty} b_n \underbrace{\int_{Xb}^{Xe} \cos(n\omega x)\cos(k\omega x)dx}_{2.2.6-C}$$

Полученное выражение (2.2.6) достаточно легко упростить с помощью выражения (2.1.5). Действительно, все интегралы вида (2.2.6-A) и (2.2.6-C) принимают нулевые значения, а интеграл (2.2.6-B) оказывается равным *T/2*.

Как следствие, для значений индекса k>0 получаем:

$$\int_{Xb}^{Xe} f(x)\cos(k\omega x)dx = b_k \frac{T}{2} \qquad \text{или} \qquad b_k = \frac{2}{T}\int_{Xb}^{Xe} f(x)\cos(k\omega x)dx \qquad (2.2.7)$$

2.3. Вычисление коэффициентов a_k

По аналогии с выводом коэффициентов b_k выполним первый и второй шаги метода, использовав в качестве множителя функцию $\sin(k\omega x)$. Как результат получим:

$$\int_{Xb}^{Xe} f(x)\sin(k\omega x)dx = \sum_{n=0}^{\infty}[a_n \underbrace{\int_{Xb}^{Xe}\sin(n\omega x)\sin(k\omega x)dx}_{2.3.1-A} + b_n \underbrace{\int_{Xb}^{Xe}\cos(n\omega x)\sin(k\omega x)dx}_{2.3.1-B}] \qquad (2.3.1)$$

На основании выражения (2.1.4) можно утверждать, что все интегралы вида (2.3.1-B) примут нулевое значение, то есть выражение (2.3.1) примет вид:

$$\int_{Xb}^{Xe} f(x)\sin(k\omega x)dx = \sum_{n=0}^{\infty} a_n \int_{Xb}^{Xe}\sin(n\omega x)\sin(k\omega x)dx \qquad (2.3.2)$$

Данное выражение также удобно рассматривать в форме нескольких случаев.

Случай А. Пусть *k=0*. Для удобства преобразований частично раскроем сумму в выражении (2.2.2) выделив из нее элемент с индексом *n=0*:

$$\int_{Xb}^{Xe} f(x)\sin(0)dx = a_0 \underbrace{\int_{Xb}^{Xe}\sin(0)\sin(0)dx}_{2.3.3-A} + \sum_{n=1}^{\infty} b_n \underbrace{\int_{Xb}^{Xe}\sin(n\omega x)\sin(0)dx}_{2.3.3-B} \qquad (2.3.3)$$

Легко заметить, что в этом случае и правая и левая часть равенства обращаются в ноль, поскольку синус нулевого угла равен нулю. Другими словами коэффициент a_0 всегда равен нулевому значению.

Случай Б. Пусть *k>0*. Для удобства преобразований также частично раскроем сумму в выражении (2.3.2) выделив из нее элемент с индексом *n=k*:

$$\int_{Xb}^{Xe} f(x)\sin(k\omega x)dx = \sum_{n=0}^{n=k-1} a_n \underbrace{\int_{Xb}^{Xe} \sin(n\omega x)\sin(k\omega x)dx}_{2.3.4-A}$$

$$+ a_k \underbrace{\int_{Xb}^{Xe} \sin(k\omega x)\sin(k\omega x)dx}_{2.3.4-B} \qquad (2.3.4)$$

$$+ \sum_{n=k+1}^{\infty} a_n \underbrace{\int_{Xb}^{Xe} \sin(n\omega x)\sin(k\omega x)dx}_{2.3.4-C}$$

Полученное выражение (2.3.4) достаточно легко упростить с помощью выражения (2.1.3). Действительно, все интегралы вида (2. 3.4-A) и (2. 3.4-C) принимают нулевые значения, а интеграл (2. 3.4-B) оказывается равным *T/2*.

Как следствие, для значений индекса k>0 получаем:

$$\int_{Xb}^{Xe} f(x)\sin(k\omega x)dx = a_k \frac{T}{2} \qquad \text{или} \qquad a_k = \frac{2}{T}\int_{Xb}^{Xe} f(x)\sin(k\omega x)dx \qquad (2.3.5)$$

2.4. Окончательный вид коэффициентов

Подводя итог проведенным вычислениям, обобщим полученные результаты.

Поскольку мы отдельно вычислили коэффициент b_0 (см. 2.2.5), а также показали, что коэффициент a_0 равен нулю (см. 2.3.3), запишем ряд Фурье в следующей форме:

$$f(x) = b_0 + \sum_{n=1}^{\infty}[a_n \sin(n\omega x) + b_n \cos(n\omega x)] \qquad (2.4.1)$$

где: ($\omega = \frac{2\pi}{T}$, $n \in (1,2,3,\ldots)$, $T = x_e - x_b$).

При этом коэффициенты ряда Фурье должны вычисляться с помощью следующих выражений:

$$b_0 = \frac{1}{T}\int_{Xb}^{Xe} f(x)\,dx$$

$$b_n = \frac{2}{T}\int_{Xb}^{Xe} f(x)\cos(n\omega x)\,dx \qquad (2.4.2)$$

$$a_n = \frac{2}{T}\int_{Xb}^{Xe} f(x)\sin(n\omega x)\,dx$$

Как уже отмечалось (1.5), ряд Фурье может быть представлен как с помощью коэффициентов a_n и b_n, так и коэффициентов c_n и φ_n, которые имеют смысл амплитуд c_n и фаз φ_n волнового процесса:

$$f(x) = \sum_{n=0}^{\infty}[a_n \sin(nx) + b_n \cos(nx)] = \sum_{n=0}^{\infty} c_n \sin(nx + \varphi_n)$$

В выражении (1.4) мы уже ссылались на известное тригонометрическое выражение:

$$a \cdot \sin(x) + b \cdot \cos(x) = \sqrt{a^2 + b^2} \cdot \sin(x + \varphi)$$

$$\text{ãäå:} \quad \varphi = \arcsin(b / \sqrt{a^2 + b^2}) \quad \text{èëè} \quad \varphi = \arccos(a / \sqrt{a^2 + b^2})$$

Это тригонометрическое выражение позволяет относительно просто вычислить при известных коэффициентах a_n и b_n соответствующие коэффициенты c_n и φ_n .

$$c_n = \sqrt{a_n^{\,2} + b_n^{\,2}}, \quad \varphi_n = \arccos(a_n / \sqrt{a_n^{\,2} + b_n^{\,2}}) \quad \| \quad \varphi_n = \arcsin(b_n / \sqrt{a_n^{\,2} + b_n^{\,2}}) .$$

Кроме того, из формулы вычисления коэффициента c_n легко заметить, что его можно рассматривать как гипотенузу прямоугольного треугольника с катетами a_n и b_n, то есть, он может также рассматриваться как вектор с проекциями a_n и b_n. Поскольку положения этого вектора определяется знакопеременными величинами a_n и b_n, а функции arcsin(x) или arccos(x) могут вернуть значения углов только в первом или четвертом квадрантах, то конечное значение φ_n необходимо корректировать в соответствии со следующей таблицей:

Первый квадрант: $a_n \geq 0, \quad b_n \geq 0$ $\varphi_n^{(1)} = \arcsin(\frac{\lvert b_n \rvert}{C_n})$	Второй квадрант: $a_n < 0, \quad b_n \geq 0$ $\varphi_n^{(2)} = \pi - \arcsin(\frac{\lvert b_n \rvert}{C_n})$
Третий квадрант: $a_n < 0, \quad b_n < 0$ $\varphi_n^{(3)} = \pi + \arcsin(\frac{\lvert b_n \rvert}{C_n})$	Четвертый квадрант: $a_n \geq 0, \quad b_n < 0$ $\varphi_n^{(4)} = 2\pi - \arcsin(\frac{\lvert b_n \rvert}{C_n})$

3. Численные методы вычисления ряда Фурье

3.1. Численное вычисление определенных интегралов

В соответствии с определением интеграла определенный интеграл представляет собой площадь под кривой, которая задана функцией Y=F(X) и ограничена аргументами начала и конца интегрирования (см. рис 3.1.1).

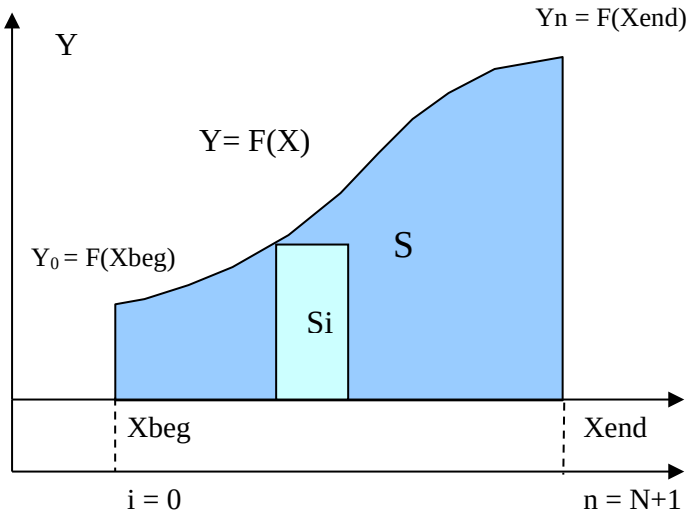

Рис. 3.1.1. Представление определенного интеграла через площадь S.

Пусть исходная функция Y=F(X) является табличной функцией, причем заданной с постоянным шагом по аргументу. Такое ограничение, как правило, обусловлено характером экспериментальных данных, которые снимаются с равномерным шагом. В этом случае с каждой точкой табличной функции связывается некоторый индекс, площадь S плотно покрывается элементарными площадками Si (квадратурами) , а значение определенного интеграла принимает вид:

$$\int_{Xbeg}^{Xend} F(X)\cdot dx = S = \lim_{\Delta X\to 0}\sum_{i=0}^{N} S_i \approx \sum_{i=0}^{N} F_i\cdot\Delta X$$

$$\text{ãäå}\quad N = \frac{Xend - Xbeg}{\Delta X}\quad (\div\text{èñëî êâàäðàòóð}) \qquad (3.1.1)$$

$$\Delta X = x_{i+1} - x_i = const$$

Использование квадратур в виде прямоугольников, является наименее точным способом вычисления определенного интеграла. По этой причине, как правило, применяют элементарные площадки (квадратуры) более точно вписывающиеся в кривую Y=F(X). Рассмотрим наиболее часто применяющиеся квадратуры.

3.1.1. Использование квадратур трапеция

Если на каждом квадратурном участке интерполировать функцию Y=F(X) полиномом P(X) = A0 + A1* X, то геометрическая форма квадратур будет соответствовать трапециям, основаниями которых являются значения Yi и Yi $_{+1}$, а высотой отрезок и Xi $_{+1}$ –Xi. В этом случае последовательность вычисления определенного интеграла примет вид:

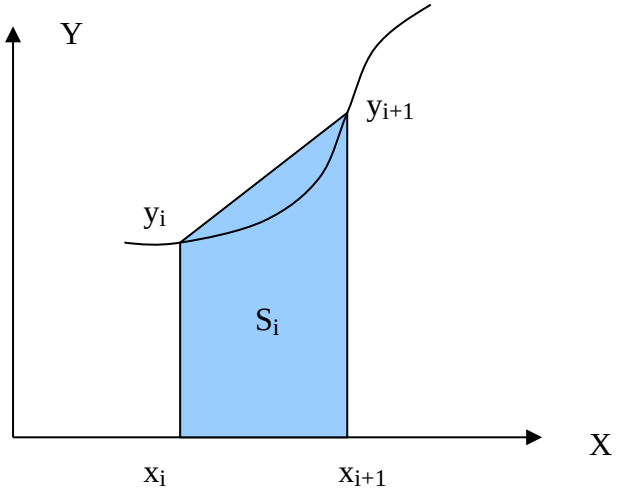

Рис.3.1.1.1. Квадратура в виде трапеции с основаниями Yi и Y_{i+1}

Для неравномерного шага по оси аргумента:

$$S_i = \frac{y_i + y_{i+1}}{2} \cdot (x_{i+1} - x_i)$$

$$S = \sum_{i=0}^{i=n-1} S_i = \frac{1}{2} \cdot \sum_{i=0}^{i=n-1} (y_i + y_{i+1}) \cdot (x_{i+1} - x_i) \qquad (3.1.1.1)$$

Для равномерного шага по оси аргумента:

$$S_i = \frac{y_i + y_{i+1}}{2} \cdot (x_{i+1} - x_i) \Bigg|_{x_{i+1} - x_i = const = h} = \frac{1}{2}(y_i + y_{i+1}) \cdot h$$

$$S = \sum_{i=0}^{i=n-1} S_i = \frac{h}{2} \cdot \sum_{i=0}^{i=n-1} (y_i + y_{i+1}) \qquad (3.1.1.2)$$

или, после преобразований граничных индексов:

$$S = \frac{h}{2} \cdot \sum_{i=0}^{i=n-1} (y_i + y_{i+1}) = \frac{h}{2} \cdot (y_0 + \sum_{i=1}^{i=n-1} y_i + y_n + \sum_{i=1}^{i=n-1} y_i)$$

$$S = h \cdot (\frac{y_0 + y_n}{2} + \sum_{i=1}^{i=n-1} y_i) \qquad (3.1.1.3)$$

3.1.2. Использование квадратур Спипсона

Еще более точное вычисление можно получить, если на каждом квадратурном участке интерполировать функцию Y=F(X) полиномом P(X) = A0 + A1* X + A1* X^2. В этом случае квадратуры называют квадратурами Симпсона, а для их вычисления требуется три точки табличной функции.

> **Внимание!** Для того чтобы квадратуры Симпсона полностью покрыли определенный интеграл необходимо четное значение индекса последней точки (n) при начале индексации с нуля.

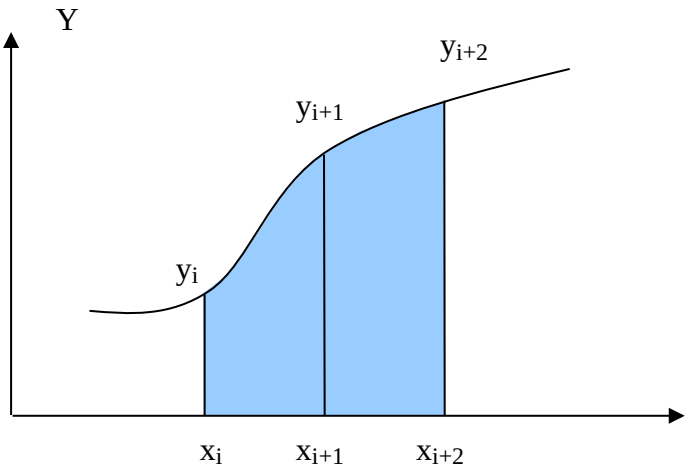

Рис.3.1.2.1. Классическая квадратура Симпсона

$$x_{i+1} - x_i = const = h$$

$$S_i = \frac{1}{3}(y_i + 4 \cdot y_{i+1} + y_{i+2}) \cdot h$$

$$S = \sum_{i=0}^{i=n\ div\ 2} S_i = \frac{h}{3} \cdot \sum_{i=0}^{i=n\ div\ 2} (y_{2i} + 4 \cdot y_{2i+1} + y_{2i+2}),$$

$$\tilde{a}\ddot{a}\mathring{a}: div - \ddot{a}\mathring{a}\ddot{e}\mathring{a}\acute{\imath}\grave{e}\mathring{a}\ \acute{a}\mathring{a}\c{c}\ \hat{\imath}\tilde{n}\grave{o}\grave{a}\grave{o}\hat{e}\grave{a},$$

К сожалению, требование четности индекса последней точки (n) при начале индексации с нуля, не всегда выполняется при снятии экспериментальных данных. В этом случае, когда общее число участков (N) нечетно, один из участков табличной функции приходится вычислять либо квадратурою трапеций, либо использовать модифицированные квадратуры Симпсона, которые вычисляют не полную квадратуру, а только левый или правый ее участок.

Далее рассмотрим применение левых и правых квадратур Симпсона.

3.1.3. Использование левых или правых квадратур Спипсона

Как уже отмечалось выше, левые или правые квадратуры Симпсона применяются в том случае, когда необходимо вычислить не полную квадратуру Симпсона, а только один из ее двух участков.

$$x_{i+1} - x_i = const = h$$

Квадратура для левого участка:

$$S_i^{(L)} = \int_0^h P(T) \cdot dT = \frac{h}{12} \cdot (5 \cdot Y_i + 8 \cdot Y_{i+1} - Y_{i+2})$$

Квадратура для правого участка:

$$S_i^{(R)} = \int_h^{2h} P(T) \cdot dT = \frac{h}{12} \cdot (-Y_i + 8 \cdot Y_{i+1} + 5 \cdot Y_{i+2})$$

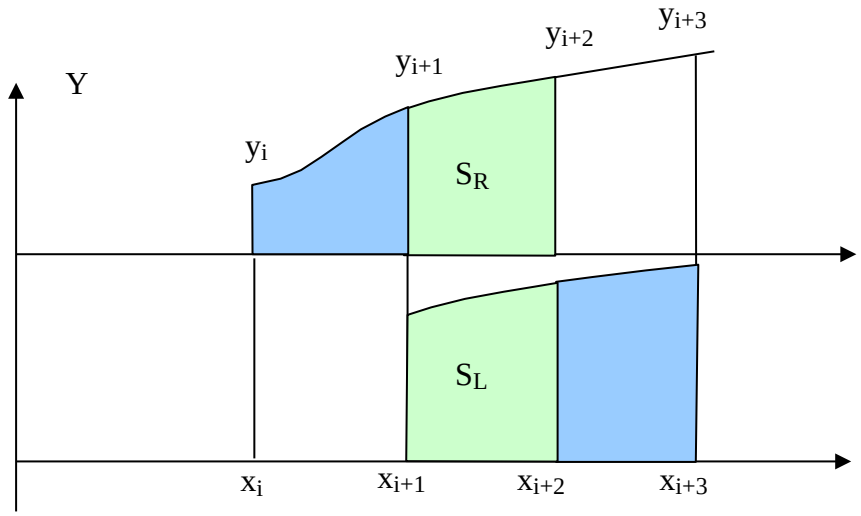

Рис.3.1.3.1. Сдвинутые относительно друг друга левая S_L и правая S_R квадратуры Симпсона

Левые и правые квадратуры Симпсона для функций Y=F(X), с значительной кривизной (большими значениями производных высоких порядков), могут приводить даже к большим ошибкам, чем соответствующие квадратуры трапеций. Однако левая и правая квадратуры обладают особенностью знакопеременной ошибки, если их вычислять со сдвигом на один участок, так как это показано на рис. 3.1.3.1. Данная особенность предполагает, что для дальнейшего повышения точности можно использовать усреднение таких квадратур

$$S^{(M)} = \frac{S_i^{(R)} + S_{i+1}^{(L)}}{2}$$

3.1.4. Использование усредненных квадратур Спипсона

На рис. 3.1.3.1. показан фрагмент представления определенного интеграла с нечетным числом участков. В данном фрагменте четыре участка вычисляются классическими квадратурами Симпсона, а оставшийся пятый участок вычисляется с помощью усреднения левой и правой квадратурой.

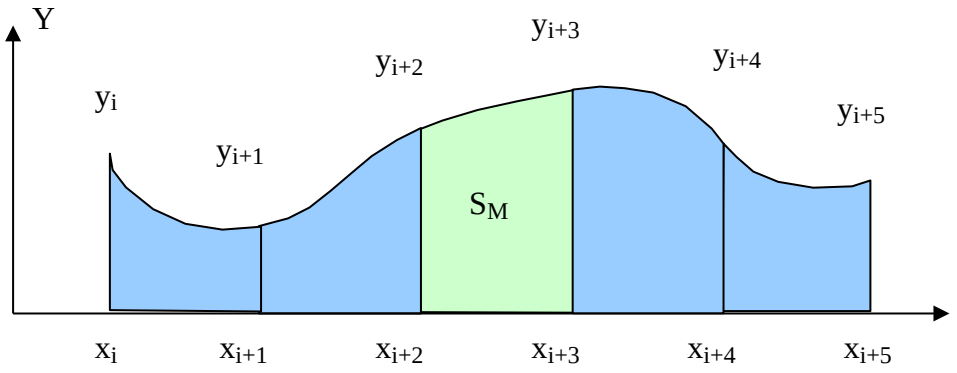

Рис.3.1.4.1. Схема вычисления нечетного участка усредненной квадратурой Симпсона

$$x_{i+1} - x_i = const = h$$

$$S^{(M)} = \frac{S_i^{(R)} + S_{i+1}^{(L)}}{2} = \frac{h}{24} \cdot (13 \cdot (Y_{i+1} + Y_{i+2}) - (Y_i + Y_{i+3})) \qquad (3.1.4.1)$$

3.2. Вычисление коэффициентов дискретного ряда Фурье

В разделе 2.4. мы получили окончательный символьный вид выражений для вычисления коэффициентов ряда Фурье:

$$b_0 = \frac{1}{T} \int_{Xb}^{Xe} f(x)\,dx$$

$$b_n = \frac{2}{T} \int_{Xb}^{Xe} f(x) \cos(k \cdot \omega \cdot x)\,dx \qquad (3.2)$$

$$a_n = \frac{2}{T} \int_{Xb}^{Xe} f(x) \sin(k \cdot \omega \cdot x)\,dx$$

Преобразуем эти выражения для случая, когда функция f(x) представлена таблично в виде Fi(Xi) с равномерным шагом по аргументу. Для этого

воспользуемся квадратурными формами вычисления определенных интегралов.
Для начала введем следующие условия и обозначения:

$k - $ èíäåêñ ãàðìîíèêè $\quad k = 1..K, \quad$ ãäå $K -$ ìàêñèìàëüí ûé èíäåêñ ãàðìîíèêè ;

$T = x_{END} - x_{BEG} \quad -$ ïåðèîä àïïðîêñèìà öèè ;

$$\omega = \frac{2 \cdot \pi}{T} - \text{êðóãîâàÿ ÷àñòîòà} \ ;$$

$$h = x_{i+1} - x_i = const = \frac{x_{END} - x_{BEG}}{N}, \quad \text{ãäå } N - \text{÷èñëî ýëåìåíòàðí ûõ ó÷àñòêîâ} \ ;$$

$$x_i = x_{BEG} + h \cdot i, \quad \text{ãäå} i - \text{èíäåêñ òî÷êè òàáëè÷íîé ôóíêöèè} \ ;$$

$$F_i = f(x_i) \quad \text{çíà÷åíèå òàáëè÷íîé ôóíêöèè} \ ;$$

При заданных условиях и обозначениях для квадратур трапеций

$$S = h \cdot (\frac{y_0 + y_n}{2} + \sum_{i=1}^{i=N-1} y_i)$$

выражения 3.2 примут следующий вид:

$$b_0 = \int_{X_{BEG}}^{X_{END}} f(x)\,dx = \frac{1}{x_{END} - x_{BEG}} (\frac{F_0 + F_N}{2} + \sum_{i=1}^{N-1} F_i) \cdot \frac{x_{END} - x_{BEG}}{N} = \frac{1}{N} (\frac{F_0 + F_N}{2} + \sum_{i=1}^{N-1} F_i)$$

$$b_k = \int_{X_{BEG}}^{X_{END}} f(x)\cos(k \cdot \omega \cdot x)dx = \frac{1}{N} (\frac{F_0 \cdot \cos(x_0) + F_N \cdot \cos(x_N)}{2} + \sum_{i=1}^{N-1} F_i \cdot \cos(x_i))$$

$$a_k = \int_{X_{BEG}}^{X_{END}} f(x)\sin(k \cdot \omega \cdot x)dx = \frac{1}{N} (\frac{F_0 \cdot \sin(x_0) + F_N \cdot \sin(x_N)}{2} + \sum_{i=1}^{N-1} F_i \cdot \sin(x_i))$$

Аналогичным образом можно вывести формулы, в которых коэффициенты ряда Фурье вычисляются с помощью квадратур Симпсона.

3. Класс DELPHI для вычисления коэффициентов ряда Фурье

Приведем полный исходный текст класса TDTF03, предназначенного для расчета коэффициентов дискретного ряда Фурье.

```
unit DTF03;
// =============================================================================
(*
   Класс вычисления коеффициентов дискретного преобразования Фурье для
   таблично заданной функции с равномерным шагом по аргументу.
   Пример вызова:
     // ---------------------------------------------------
     DTF := TDTF03.Create;
     // ---------------------------------------------------
     // Вывод фазы в радианах/градусах (для property Pn)
     DTF.RqDegree := False;           // Радианы
     // Метод вычисления интегралов
     DTF.CodMet := 0;                 // 0 - Квадратуры трапеций, 1 - Симпсона
     // Диапазон табличной функции
     DTF.XB := XB;                    // Начало по X
     DTF.XE := XE;                    // Конец  по X
     // Выполнить DTF для табличной функции ArrY
     DTF.RunDTF03(ArrY);              // var ArrY : array of extended
     // ---------------------------------------------------
     // Выборка результатов выполняется посредством property:
     // ------------------------------
     // Классическая запись ряда Фурье
     DTF.An[Indx] - Коэфф. при Sin ряда Фурье
     DTF.Bn[Indx] - Коэфф. при Cos ряда Фурье
     // ------------------------------
     // Запись ряда Фурье в виде SUM( Cn * Sin(n * Omega * X + Pn))
     // Pn - в радианах
     // Omega - Круговая частота 2 * Pi / (XE - XB)
     DTF.Cn[Indx] - Амплитуды гармоник при Sin ряда Фурье
     DTF.Pn[Indx] - Фазы (радианах/градусах) гармоник для Sin ряда Фурье
     // ---------------------------------------------------
     Вычисление значений функции в точке X по коэфф. ряда Фурье:
     Y := DTF.CalcFuncTDF(X);
     // ---------------------------------------------------
     DTF.Free;
*)
// =============================================================================
interface

uses Dialogs, DateUtils, Math, SysUtils, StdCtrls;

// =============================================================================
// Описание основных структур
// =============================================================================
type TSnCs  = record
   Sn  : extended;  // Значение SIN
   Cs  : extended;  // Значение COS
end;

type TDTF03 = class(TObject)
  private
     fMNumPnt  : integer;           // Число точек в исходных данных
     fMNumGrm  : integer;           // Максимальное число гармоник
     fMIndGrm  : integer;           // Максимальный индекс гармоники
     fDegree   : boolean;           // Запрос на вывод фазы в градусах
     // Диапазон
     fXB       : extended;          // Начальное значение аргумента
     fXE       : extended;          // Конечное  значение аргумента
     fStepX    : extended;
     fOmega    : extended;
     // ------------------
     fCodMet   : byte;              // Код метода вычисления квадратур (0,1)
     // ------------------
     // РАБОЧИЕ ПЕРЕМЕННЫЕ
     fBSnCs    : TSnCs;
     fESnCs    : TSnCs;
```

```
    // Таблица для квадратурых сумм по каждой гармрнике
    fQuadTAB  : array of record
                  // Для сверток с SIN
                  SS2  : extended; // Сумма светрок с индексом: 0 и 2
                  SS1  : extended; // Сумма светрок с индексом: 1
                  SSM  : extended; // Квадратура Симпсон-М на точках 1..4
                  SSUM : extended; // Итоговая квадратурная сумма
                  // Для сверток с COS
                  CS2  : extended; // Сумма светрок с индексом: 0 и 2
                  CS1  : extended; // Сумма светрок с индексом: 1
                  CSM  : extended; // Квадратура Симпсон-М на точках 1..4
                  CSUM : extended; // Итоговая квадратурная сумма
                  // Окончательные коэффициенты ряда Фурье в двух форматах
                  An   : extended;
                  Bn   : extended;
                  Cn   : extended;
                  Pn   : extended;
                end;
    // Очистка таблицы квадратурных сумм
    procedure ClearQuadTAB();
    // Подготовка массива гармоник fQuadTAB = таблицы для квадратурных сумм
    function ReSetQuadTAB(RqArr : array of extended) : boolean;
    // Вычисление fOmega и fStepX
    procedure CalcQmegaAndStepX(RqArr : array of extended);
    // Вычисление Sin(X) и Cos(X) для заданной гармоники и точки табличной функции
    function GetSnCs(RqIndG, RqIndX : integer) : TSnCs;
    // Вычисление интегралов сверток квадратурными суммами Трапеций
    function TrapeziumQuadrature(RqArr : array of extended) : boolean;
    // Вычисление интегралов сверток квадратурными суммами Симпсона
    function SimpsonQuadrature(RqArr : array of extended) : boolean;
    // Вычисление коэффициентов An, Bn одной гармоники по интегралам сверток
    procedure CalcAnBn(RqArr : array of extended; IndG : integer);
    // Преобразование коэффициентов An, Bn одной гармоники в коэффициенты Cn, Pn
    procedure AnBnToCnPn(IndG : integer);
    // Вычисление квадратурных сумм для всех гармоник
    procedure CalcQuadTAB(RqArr : array of extended);
    // Поддержка property
    function GetAn(Ind : word) : extended;
    function GetBn(Ind : word) : extended;
    function GetCn(Ind : word) : extended;
    function GetPn(Ind : word) : extended;
    // Установка начального значения аргумента
    procedure SetXB(RqXB : extended);
    // Установка конечного значения аргумента
    procedure SetXE(RqXE : extended);
  public
    procedure Free();
    // Вычислить коэффициенты ряда Фурье
    procedure RunDTF03(RqArr : array of extended);
    // Вычислить значение ряда фурье в точке, заданной индексом
    // function CalcFuncTDF (RqInd : word) : extended; overload;
    // Вычислить значение ряда фурье в точке, заданной аргументом
    function CalcFuncTDF(RqX : extended) : extended; // overload;
    // ОТЛАДОЧНЫЕ МЕТОДЫ КЛАССА
    // Для отладки вычисления квадратурных сумм
    procedure LookDTF03(RqRep : TMemo);
    // ОСНОВНЫЕ СВОЙСТВА КЛАССА
    property XB : extended read fXB write SetXB;
    property XE : extended read fXE write SetXE;
    property StepX : extended read fStepX;
    // Для выборки коэффициентов ряда Фурье
    property An[Ind : word] : extended read GetAn;
    property Bn[Ind : word] : extended read GetBn;
    property Cn[Ind : word] : extended read GetCn;
    property Pn[Ind : word] : extended read GetPn;
    // Установка / чтение режима выборки фазы в градусах / радианах
    property RqDegree  : boolean read fDegree write fDegree;
    // Индекс максимальной гармоники
    property MaxIndGgm : integer read fMIndGrm;
    // Код метода вычисления квадратур 0-Трапеции, 1 - Симпсон
    property CodMet    : byte read fCodMet write fCodMet;
end;
```

```
implementation
// ------------------------------------------------------------------------
const MinNumPnt = 6;     // Минимальное число точек в исходных данных
// ------------------------------------------------------------------------
// 28.02.2014
procedure TDTF03.Free();
begin
   SetLength (fQuadTAB, 0);
   inherited Free();
end;
// ------------------------------------------------------------------------
// 23.03.2014
// ФУНКЦИОНАЛЬНАЯ ПОДДЕРЖКА PROPERTY
// ------------------------------------------------------------------
function TDTF03.GetAn(Ind : word) : extended;
begin
    if Ind <= fMIndGrm then Result := fQuadTAB[Ind].An
    else Result := 0;
end;
// ------------------------------------------------------------------
function TDTF03.GetBn(Ind : word) : extended;
begin
    if Ind <= fMIndGrm then Result := fQuadTAB[Ind].Bn
    else Result := 0;
end;
// ------------------------------------------------------------------
function TDTF03.GetCn(Ind : word) : extended;
begin
    if Ind <= fMIndGrm then Result := fQuadTAB[Ind].Cn
    else Result := 0;
end;
// ------------------------------------------------------------------
function TDTF03.GetPn(Ind : word) : extended;
begin
  if Ind <= fMIndGrm
  then begin
    if not fDegree
    then Result := fQuadTAB[Ind].Pn
    else Result := 180 * fQuadTAB[Ind].Pn /Pi;
  end
  else Result := 0;
end;
// ------------------------------------------------------------------
// Установка начального значения аргумента
procedure TDTF03.SetXB(RqXB : extended);
begin
   if (RqXB = fXB) then Exit;
   if (RqXB >= fXE)
   then begin
      MessageDlg('Начальное значение аргумента не может быть' +  #13#10
               + 'больше или равно конечнонму значению...',
          mtError, [mbOk], 0);
      Exit;
   end;
   fXB := RqXB;
end;
// ------------------------------------------------------------------
// Установка конечного значения аргумента
procedure TDTF03.SetXE(RqXE : extended);
begin
   if (RqXE = fXE) then Exit;
   if (RqXE <= fXB)
   then begin
      MessageDlg('Конечное значение аргумента не может быть' +  #13#10
               + 'Меньше или равно начальному значению...',
          mtError, [mbOk], 0);
      Exit;
   end;
   fXE := RqXE;
end;
// ------------------------------------------------------------------------
// 01.03.2014
// Очистка таблицы квадратурных сумм
procedure TDTF03.ClearQuadTAB();
var Ind : integer;
begin
   if Length(fQuadTAB) > 0
```

```
   then begin
     for Ind := Low(fQuadTAB) to High(fQuadTAB)
     do FillChar(fQuadTAB[Ind], SizeOF(fQuadTAB[Ind]), #0);
   end;
end;
// ---------------------------------------------------------------------------
// 01.03.2014
// Подготовка массива гармоник fQuadTAB = таблицы для квадратурных сумм
function TDTF03.ReSetQuadTAB(RqArr : array of extended) : boolean;
// Length(RqArr) - Число точек в массиве RqArr;
// High(RqArr)   - Индекс последней точки = числу участков в мвссиве RqArr;
// High(RqArr) div 2 - Число квадратур Симпсона;
// High(RqArr) mod 2 - Число участков сверх покрытия Симпсоном = 0 или 1;
// Минимальное число участков для расчета высшей гармоники = 4;
begin
   try
      // Оценка макс. индекса гармоник по числу участков массиве исходных точек
      if (fMIndGrm <> High(RqArr) div 4)
      then begin
         // Новый максимальный индекс в массиве гармоник = High(fQuadTAB)
         fMIndGrm := High(RqArr) div 4;
         // Новое число точек в массиве гармоник = Length(fQuadTAB)
         fMNumGrm := 1 + fMIndGrm;
         // Установка нового размера массива гармоник
         SetLength (fQuadTAB, fMNumGrm);
      end;
      // Очистка массива гармоник (таблицы квадратурных сумм)
      ClearQuadTAB();
      Result := True;
   except
      Result := False;
   end;
end;
// ---------------------------------------------------------------------------
// 30.08.2017
// Вычисление fOmega и fStepX
procedure TDTF03.CalcQmegaAndStepX(RqArr : array of extended);
const TwoPi = 2 * Pi;
begin
   fStepX  := (fXE - fXB) / High(RqArr);
   fOmega  :=  TwoPi / (fXE - fXB);
end;
// ---------------------------------------------------------------------------
// 30.08.2017
// Вычисление Sin(X) и Cos(X) для заданной гармоники и точки табличной функции
function TDTF03.GetSnCs(RqIndG, RqIndX : integer) : TSnCs;
begin
   SinCos(RqIndG * fOmega * (fXB + fStepX * RqIndX),
          Result.Sn, Result.Cs);
end;
// ---------------------------------------------------------------------------
// 30.08.2017
// Вычисление интегралов сверток квадратурными суммами Трапеций
function TDTF03.TrapeziumQuadrature(RqArr : array of extended) : boolean;
var   IndX               : integer;  // Рабочий индекс точки исходных данных
      IndG               : integer;  // Рабочий индекс гармоники
      Sum0, Sum1         : extended; // Рабочие накопители
begin
  // Очистка таблицы квадратурных сумм
  ClearQuadTAB();
  // Вычисление fOmega и fStepX
  CalcQmegaAndStepX(RqArr);
  // Построение по отдельным гармоникам суммы квадратур Трапеций
  for IndG := 0 to High(fQuadTAB) do
  begin
     // Sin и Cos для начальной точки в RqArr
     fBSnCs := GetSnCs(IndG, Low(RqArr));
     // Свертки исходных данных и накопление квадратурных сумм
     for IndX := Low(RqArr) to High(RqArr) - 1
     do begin
          // Sin и Cos для очередной точки в RqArr
          fESnCs := GetSnCs(IndG, IndX + 1);
          // Свертки исходных данных с синусом
          Sum0 := RqArr[IndX]     * fBSnCs.Sn;
          Sum1 := RqArr[IndX + 1] * fESnCs.Sn;
          fQuadTAB[IndG].SSUM := fQuadTAB[IndG].SSUM + Sum0 + Sum1;
          // Свертки исходных данных с косинусом
```

```
          Sum0 := RqArr[IndX]     * fBSnCs.Cs;
          Sum1 := RqArr[IndX + 1] * fESnCs.Cs;
          fQuadTAB[IndG].CSUM := fQuadTAB[IndG].CSUM + Sum0 + Sum1;
          // Sin и Cos для очередной начальной точки в RqArr
          fBSnCs := fESnCs;
       end;
      // Вычислим окончательные суммы квадратур Трапеций
      fQuadTAB[IndG].SSUM := fQuadTAB[IndG].SSUM / 2;
      fQuadTAB[IndG].CSUM := fQuadTAB[IndG].CSUM / 2;
  end;
  Result := True;
end;
// ---------------------------------------------------------------------------
// 30.08.2017
// Вычисление интегралов сверток квадратурными суммами Симпсона
function TDTF03.SimpsonQuadrature(RqArr : array of extended) : boolean;
var   StartIndX  : integer;  // Стартовый индекс начальной точки квадратуры
      IndX       : integer;  // Рабочий индекс начальной точки квадратуры
      IndG       : integer;  // Рабочий индекс гармоники
      //----------
      // Рабочие для Sin и Cos различных точках квадратуры квадратуры
      w1SnCs     : TSnCs;
      w2SnCs     : TSnCs;
      w3SnCs     : TSnCs;
      w4SnCs     : TSnCs;
      //----------
      Sum0, Sum1, Sum2 : extended; // Рабочие накопители классических квадратур
      Sum3, Sum4       : extended; // Рабочие накопители M - квадратуры
begin
  Result := False;
  // Очистка таблицы квадратурных сумм
  ClearQuadTAB();
  // Контроль минималного числа исходных точек
  if High(RqArr) < 5 then Exit;
  // Вычисление fOmega и fStepX
  CalcQmegaAndStepX(RqArr);
  // ---------------------------------------------------
  // Вычисление квадратурных сумм Симпсона и M - Симпсона
  if (Length(RqArr) mod 2) = 0       // Четное число точек
  then begin
      // Четное число точек
      for IndG := 0 to High(fQuadTAB)
      do begin
          // ---------------------------------------
          // Первая классическая квадратура Симпсона
          // на точках с индексами 0, 1, 2
          IndX := Low(RqArr);
          // Sin и Cos для начальной точки квадратуры
          fBSnCs := GetSnCs(IndG, IndX);
          // Sin и Cos для второй точки квадратуры
          w1SnCs := GetSnCs(IndG, IndX + 1);
          // Sin и Cos для конечной (третьей) точки квадратуры
          fESnCs := GetSnCs(IndG, IndX + 2);
          // Свертки исходных данных с синусом
          Sum0 := RqArr[IndX]     * fBSnCs.Sn;
          Sum1 := RqArr[IndX + 1] * w1SnCs.Sn;
          Sum2 := RqArr[IndX + 2] * fESnCs.Sn;
          fQuadTAB[IndG].SS2 := fQuadTAB[IndG].SS2 + Sum0 + Sum2;
          fQuadTAB[IndG].SS1 := fQuadTAB[IndG].SS1 + Sum1;
          // Свертки исходных данных с косинусом
          Sum0 := RqArr[IndX]     * fBSnCs.Cs;
          Sum1 := RqArr[IndX + 1] * w1SnCs.Cs;
          Sum2 := RqArr[IndX + 2] * fESnCs.Cs;
          fQuadTAB[IndG].CS2 := fQuadTAB[IndG].CS2 + Sum0 + Sum2;
          fQuadTAB[IndG].CS1 := fQuadTAB[IndG].CS1 + Sum1;
          // ---------------------------------------
          // Вычисление полной M-квадратуры Симпсона
          // на точках с индексами 1, 2, 3, 4.
          // Sin и Cos для первой точки M-квадратуры Симпсона
          // w1SnCs
          // Sin и Cos для второй точки M-квадратуры Симпсона
          w2SnCs :=  fESnCs;
          // Sin и Cos для третьей точки M-квадратуры Симпсона
          w3SnCs := GetSnCs(IndG, IndX + 3);
          // Sin и Cos для четвертой точки M-квадратуры Симпсона
          w4SnCs := GetSnCs(IndG, IndX + 4);
          // Свертки исходных данных с синусом
```

```
          Sum1 := RqArr[IndX + 1] * w1SnCs.Sn;
          Sum2 := RqArr[IndX + 2] * w2SnCs.Sn;
          Sum3 := RqArr[IndX + 3] * w3SnCs.Sn;
          Sum4 := RqArr[IndX + 4] * w4SnCs.Sn;
          fQuadTAB[IndG].SSM := ((Sum2 + Sum3) * 13 - Sum1 - Sum4) / 24;
          // Свертки исходных данных с косинусом
          Sum1 := RqArr[IndX + 1] * w1SnCs.Cs;
          Sum2 := RqArr[IndX + 2] * w2SnCs.Cs;
          Sum3 := RqArr[IndX + 3] * w3SnCs.Cs;
          Sum4 := RqArr[IndX + 4] * w4SnCs.Cs;
          fQuadTAB[IndG].CSM := ((Sum2 + Sum3) * 13 - Sum1 - Sum4) / 24;
          // --------------------------------------
      end;
      // Стартовый индекс частичных групп сум для квадратур Симпсона
      StartIndX := 3;
  end
  else begin
      // Стартовый индекс частичных групп сум для квадратур Симпсона
      StartIndX := 0;
  end;
  // ---------------------------------------------------
  // Вычисление остальных квадратурных сумм Симпсона
  for IndG := 0 to High(fQuadTAB)
  do begin
     IndX := StartIndX;
     // Sin и Cos для начальной точки квадратуры
     fBSnCs := GetSnCs(IndG, IndX);
     repeat
          // Sin и Cos для второй точки квадратуры
          w2SnCs := GetSnCs(IndG, IndX + 1);
          // Sin и Cos для конечной точки квадратуры
          fESnCs := GetSnCs(IndG, IndX + 2);
          // Свертки исходных данных с синусом
          Sum0 := RqArr[IndX]     * fBSnCs.Sn;
          Sum1 := RqArr[IndX + 1] * w2SnCs.Sn;
          Sum2 := RqArr[IndX + 2] * fESnCs.Sn;
          fQuadTAB[IndG].SS2 := fQuadTAB[IndG].SS2 + Sum0 + Sum2;
          fQuadTAB[IndG].SS1 := fQuadTAB[IndG].SS1 + Sum1;
          // Свертки исходных данных с косинусом
          Sum0 := RqArr[IndX]     * fBSnCs.Cs;
          Sum1 := RqArr[IndX + 1] * w2SnCs.Cs;
          Sum2 := RqArr[IndX + 2] * fESnCs.Cs;
          fQuadTAB[IndG].CS2 := fQuadTAB[IndG].CS2 + Sum0 + Sum2;
          fQuadTAB[IndG].CS1 := fQuadTAB[IndG].CS1 + Sum1;
          // Sin и Cos для очередной начальной точки квадратуры
          fBSnCs := fESnCs;
          // Начальный индекс следующей квадратуры
          IndX := IndX + 2;

     until (IndX >= High(RqArr));
     // Вычислим окончательные квадратуры Симпсона для всех гармоник
     fQuadTAB[IndG].SSUM := (4 * fQuadTAB[IndG].SS1 + fQuadTAB[IndG].SS2)/3;
     fQuadTAB[IndG].CSUM := (4 * fQuadTAB[IndG].CS1 + fQuadTAB[IndG].CS2)/3;
     // Присоеденим М-квадратуры Симпсона
     fQuadTAB[IndG].SSUM := fQuadTAB[IndG].SSUM + fQuadTAB[IndG].SSM;
     fQuadTAB[IndG].CSUM := fQuadTAB[IndG].CSUM + fQuadTAB[IndG].CSM;
  end;
  Result := True;
end;
// ---------------------------------------------------------------------------
// 14.03.2014
// Вычисление коэффициентов An, Bn одной гармоники по интегралам сверток
procedure TDTF03.CalcAnBn(RqArr : array of extended; IndG : integer);
var h : extended;
begin
  h := 1/High(RqArr);
  if IndG = 0
  then begin
     fQuadTAB[IndG].An := 0;
     fQuadTAB[IndG].Bn := fQuadTAB[IndG].CSUM * h;
  end
  else begin
     fQuadTAB[IndG].An := 2 * fQuadTAB[IndG].SSUM * h;
     fQuadTAB[IndG].Bn := 2 * fQuadTAB[IndG].CSUM * h;
  end;
end;
// ---------------------------------------------------------------------------
```

```
// 14.03.2014
// Преобразование коэффициентов An, Bn одной гармоники в коэффициенты Cn, Pn
procedure TDTF03.AnBnToCnPn(IndG : integer);
const Noise = 1E-12;              // Уровень шумов вычисления
begin
   // Вычислим коэффициенты Cn ряда Фурье (амплитуды гармоник)
   fQuadTAB[IndG].Cn := Sqrt(fQuadTAB[IndG].An * fQuadTAB[IndG].An
                           + fQuadTAB[IndG].Bn * fQuadTAB[IndG].Bn);
   // Вычислим коэффициенты Pn ряда Фурье (фазы гармоник)
   if fQuadTAB[IndG].Cn > Noise
   then begin
      // I квадрант (для вычисления ARCSIN требуется модуль Math)
      fQuadTAB[IndG].Pn := Abs(ARCSIN(fQuadTAB[IndG].Bn / fQuadTAB[IndG].Cn));
      // Остальные квадранты
      if fQuadTAB[IndG].Bn >= 0
      then begin
         // I и II   квадранты ( Bn >= 0 )
         if fQuadTAB[IndG].An < 0
         then fQuadTAB[IndG].Pn := Pi - fQuadTAB[IndG].Pn;     // II квадрант
      end
      else begin
         // III и IV квадранты ( Bn < 0)
         if fQuadTAB[IndG].An < 0
         then fQuadTAB[IndG].Pn := fQuadTAB[IndG].Pn + Pi      // III квадрант
         else fQuadTAB[IndG].Pn := 2 * Pi - fQuadTAB[IndG].Pn; // IV  квадрант
      end;
   end
   else begin
     // Отсечение шумов вычисления
     fQuadTAB[IndG].Cn := 0;
     fQuadTAB[IndG].Pn := 0;
   end;
end;
// ---------------------------------------------------------------------------
// 29.05.2016
// Вычисление квадратурных сумм для всех гармоник
procedure TDTF03.CalcQuadTAB(RqArr : array of extended);
var   IndG : integer;             // Рабочий индекс гармоники
begin

  // Ловкшки граничных условий алгоритма
  if (Length(RqArr) < 6) then Exit;
  if not(Low(RqArr) = 0) then Exit;

  // Вычисление квадратурных сумм по методу Симпсона
  case fCodMet of
     0 : TrapeziumQuadrature(RqArr);
     1 : SimpsonQuadrature(RqArr);
  end;
  // Вычисление окончательных значений интегралов Фурье
  for IndG := Low(fQuadTAB) to High(fQuadTAB)
  do begin
     // Вычислим коэффициенты ряда Фурье в формате An и Bn
     CalcAnBn(RqArr, IndG);
     // Вычислим коэффициенты ряда Фурье в формате Cn и Pn
     AnBnToCnPn(IndG);
  end; // of for IndG
end;  // of procedure CalcQuadTAB
// ---------------------------------------------------------------------
// 30.08.2017
// Вычислить значение ряда фурье в точке, заданной аргументом
function TDTF03.CalcFuncTDF(RqX : extended) : extended;
const cEps = 1e-10;
var wIndG   : integer;
    wXToPhs : extended;
    wPhase  : extended;
    wY      : extended;
begin
   Result := 0;
   //Входной контроль
   if Length(fQuadTAB) <= 0 then Exit;
   if (fXE < fXB) then Exit;
   if Abs(fXE - fXB) < cEps then fXE := fXB + cEps;
   // Круговая частота
   wXToPhs := 2 * Pi /(fXE - fXB);
   // Вычислить значение табличной функции
   wY := 0;
```

```
   for wIndG := Low(fQuadTAB) to High(fQuadTAB)
   do begin
      wPhase  := wXToPhs * wIndG * RqX + fQuadTAB[wIndG].Pn;
      if wIndG = Low(fQuadTAB)
      then wY := fQuadTAB[wIndG].Cn
      else wY := wY + fQuadTAB[wIndG].Cn * Sin(wPhase);
   end;
   Result := wY;
end;
// ---------------------------------------------------------------------------
// 02.03.2014
// Для отладки
procedure TDTF03.LookDTF03(RqRep : TMemo);
var Ind : integer;
    WStr : string;
begin
    RqRep.Clear;
    for Ind := 0 to High(fQuadTAB)
    do begin
      WStr := 'SSn' + IntToStr(Ind) + '=' + FloatToStr(fQuadTAB[Ind].SSUM);
      RqRep.Lines.Add(WStr);
      WStr := 'MSn' + IntToStr(Ind) + '=' + FloatToStr(fQuadTAB[Ind].SSM);
      RqRep.Lines.Add(WStr);

      WStr := 'SCs' + IntToStr(Ind) + '=' + FloatToStr(fQuadTAB[Ind].CSUM);
      RqRep.Lines.Add(WStr);
      WStr := 'MCs' + IntToStr(Ind) + '=' + FloatToStr(fQuadTAB[Ind].CSM);
      RqRep.Lines.Add(WStr);

      RqRep.Lines.Add('');
    end;
end;
// ---------------------------------------------------------------------------
// 30.08.2017
// ВЫЧИСЛИТЬ КОЭФФИЦИЕНТЫ ПРЕОБРАЗОВАНИЯ ФУРЬЕ
procedure TDTF03.RunDTF03(RqArr : array of extended);
begin
   // ----------------------------------------
   // Ловкшки граничных условий алгоритма
   if (Length(RqArr) < MinNumPnt) then Exit;
   if not(Low(RqArr) = 0) then Exit;
   // Фиксация числа точек в исходных данных
   fMNumPnt := Length(RqArr);
   // Проверка диапазона аргумента
   if (fXE <= fXB)
   then begin
      MessageDlg('Конечное значение аргумента не может быть' +  #13#10
               + 'Меньше или равно начальному значению...',
          mtError, [mbOk], 0);
      Exit;
   end;
   // ----------------------------------------
   // ВЫПОЛНЕНИЕ
   // Подготовка таблицы квадратурных сумм для очередного DFT
   if not ReSetQuadTAB(RqArr) then Exit;
   // Вычисление квадратурных сумм для всех гармоник
   CalcQuadTAB(RqArr);
end;
// ===========================================================================
// КОНЕЦ
// ===========================================================================
end.
```

Литература

1. Корн Г., Корн Т. Справочник по математике (для научных работников и инженеров). М.: Наука, -1978, - 832с.: ил.

«Справочник» содержит сведения по следующим разделам: высшая алгебра, аналитическая и дифференциальная геометрия, математический анализ (включая интегралы Лебега и Стилтьеса); векторный и тензорный анализ; криволинейные координаты; функции комплексного переменного; операционное исчисление; дифференциальные уравнения обыкновенные и с частными производными; вариационное исчисление; абстрактная алгебра; матрицы; линейные векторные пространства; операторы и теория представлений; интегральные уравнения; краевые задачи; теория вероятностей и математическая статистика; численные методы анализа; специальные функции.

2. Андре Анго. Математика для электро - и радиоинженеров. М.: Наука, - 1965, - 779с.: ил.

В книге рассматриваются много интересных приложений из области электро - и радиотехники будет интересна широкому кругу инженерно-технических и научных работников, имеющих дело с математикой и ее приложениями, а также студентам и аспирантам.

КИЕВ 2017г.