УДК 519.6

ЦЕЛОЧИСЛЕННЫЙ АЛГОРИТМ ГЕНЕРАЦИИ СИНУСОИДАЛЬНОГО СИГНАЛА

Благодаров А.В., *к.т.н., доцент кафедры «Вычислительная и прикладная математика» Рязанского государственного радиотехнического университета, e-mail: rnf@rambler.ru.*

INTEGER SINE WAVE GENERATION ALGORITHM

Blagodarov A.V.

A sine wave generation algorithm without any real operations is offered. The program source code is attached.

Key words: algorithm, wave, sine signal, generator, DDA, Integer Wave Algorithm.

Ключевые слова: алгоритм, синусоидальный сигнал, целочисленный волновой алгоритм, генератор, цифровой дифференциальный анализатор.

Предложен алгоритм генерации синусоидального сигнала, не использующий вещественные вычисления с плавающей точкой. Приведен текст программы, реализующей предлагаемый алгоритм.

Введение

При разработке различных электронных устройств, а также при моделировании физических процессов может потребоваться алгоритм генерации сигнала синусоидальной формы. В настоящее время для решения этой задачи обычно применяют табличный метод либо алгоритмы CORDIC (COrdinate Rotation DIgital Computer – цифровой вычислитель поворота системы координат) [1, 2].

Табличный метод предполагает хранение в памяти ЭВМ таблицы заранее вычисленных с некоторым шагом значений синусоидального сигнала. При этом можно хранить как вещественные, так и целые значения. Достоинством метода является очень высокая производительность. Однако в некоторых случаях объем памяти, требуемый для хранения таблицы, может оказаться слишком большим.

Алгоритмы CORDIC экономичны с точки зрения используемого объема памяти, но требуют вещественных вычислений.

Кроме того, существуют различные алгоритмы построения окружности, базирующиеся на целочисленных вычислениях, которые широко применяют в машинной графике. К таковым, в частности, относятся: алгоритм Брезенхема (J. Bresenham) [4–5], алгоритм Мичнера (J. Michener) [8, 11], алгоритм средней точки (Midpoint Circle Algorithm) [6–7, 10].

Перечисленные алгоритмы основаны на принципе цифрового дифференциального анализатора (ЦДА, англ. DDA). На первый взгляд может показаться, что на их основе можно реализовать целочисленное вычисление значений синусоидального сигнала. Однако при ближайшем рассмотрении оказывается, что это не так (причина будет показана ниже).

Отдельно следует назвать очень простой и быстрый алгоритм Минского (M. Minsky) [3, 9], который может применяться для генерации синусоидального сигнала с использованием как вещественных, так и целых чисел. Однако при использовании целочисленных вычислений алгоритм Минского дает слишком грубые результаты.

Автором предлагается алгоритм генерации синусоидального сигнала, использующий только целочисленные вычисления и не требующий большого объема памяти.

Теоретическая часть

Пусть требуется получить последовательность целых чисел $s(t)$, определяющихся по формуле

$$s = a \cdot \sin\left(\frac{1}{a} \cdot t\right) \tag{1}$$

где a – некоторое целое число, значительно большее 1; параметр t – целое число, $t = 0, 1, 2, \ldots$.

Попробуем вычислить значения s без использования операций с плавающей точкой, применяя лишь целочисленные операции.

Возьмем простейшую формулу из тригонометрии:

$$\sin^2\left(\frac{1}{a} \cdot t\right) + \cos^2\left(\frac{1}{a} \cdot t\right) = 1 \tag{2}$$

и умножим обе её части на a^2, получив:

$$a^2 \cdot \sin^2\left(\frac{1}{a} \cdot t\right) + a^2 \cdot \cos^2\left(\frac{1}{a} \cdot t\right) = a^2 . \tag{3}$$

Введем обозначение

$$c = a \cdot \cos\left(\frac{1}{a} \cdot t\right). \tag{4}$$

Тогда формулу (4) можно записать в виде

$$s^2 + c^2 = a^2 . \tag{5}$$

Найдем производную от s по t:

$$\frac{ds}{dt} = \cos\left(\frac{1}{a} \cdot t\right). \tag{6}$$

Используя формулы (5), (6) и (7), получаем дифференциальное уравнение:

$$s^2 + a^2 \cdot \left(\frac{ds}{dt}\right)^2 = a^2 . \tag{7}$$

Преобразуем его в следующий вид:

$$(ds)^2 = (dt)^2 \cdot \frac{a^2 - s^2}{a^2}. \quad (8)$$

Теперь запишем уравнение (9) в приближенном виде:

$$(\Delta s)^2 \approx (\Delta t)^2 \cdot \frac{a^2 - s^2}{a^2}. \quad (9)$$

Поскольку по условию задачи параметр t изменяется с шагом 1, имеем $\Delta t = 1$, т.е.

$$(\Delta s)^2 \approx \frac{a^2 - s^2}{a^2}. \quad (10)$$

Используя формулу (6), преобразуем формулу (11):

$$(\Delta s)^2 \approx \frac{c^2}{a^2}. \quad (11)$$

Можно также записать (12) в виде

$$a^2 \cdot (\Delta s)^2 \approx c^2. \quad (12)$$

Из формулы (12) совершенно очевидно, что при $\Delta t = 1$ соблюдается неравенство $|\Delta s| \le 1$ (значение модуля косинуса никак не может превысить 1). Поскольку необходимо получить целые значения s, возможно всего два варианта: $|\Delta s| = 1$ или $|\Delta s| = 0$.

Рассмотрим участок кривой, заданной формулой (2), для $t \in [0, a \cdot \pi / 2]$. При этом имеем $\Delta s \ge 0$ (функция синусоидального сигнала возрастает), а $c \ge 0$.

Извлечем квадратный корень из обеих частей приближенного равенства (13):

$$a \cdot \Delta s \approx c. \quad (13)$$

Тогда из (14) можно получить

$$\begin{cases} 0 \approx c, & \text{если} \quad \Delta s = 0, \\ a \approx c, & \text{если} \quad \Delta s = 1. \end{cases} \quad (14)$$

Оценим погрешность вычисления выражения $a \cdot \Delta s$. Это выражение удобно тем, что оно, в отличие от Δs, позволит оперировать с целыми числами. Погрешность вычисления $a \cdot \Delta s$ может быть определена по формулам:

$$\begin{cases} e_{S0} = e_S - c, & \text{если} \quad \Delta s = 0, \\ e_{S1} = e_S + a - c, & \text{если} \quad \Delta s = 1. \end{cases} \quad (15)$$

Здесь e_S – сумма погрешностей вычисления выражения $a \cdot \Delta s$ для предыдущих значений s. По сути, e_S – это погрешность вычисления s, умноженная на a.

Теперь выберем из двух вариантов погрешностей e_{S0} и e_{S1} минимальную по модулю и определим оптимальную для нас Δs:

$$\Delta s = \begin{cases} 0, & \text{если} \quad |e_{S0}| < |e_{S1}|, \\ 1, & \text{если} \quad |e_{S0}| \ge |e_{S1}|. \end{cases} \quad (16)$$

Запомним значение накопленной погрешности e_S:

$$e_S = \begin{cases} e_{S0}, & \text{если} \quad |e_{S0}| < |e_{S1}|, \\ e_{S1}, & \text{если} \quad |e_{S0}| \ge |e_{S1}|. \end{cases} \quad (17)$$

Остается добавить к предыдущему значению s вычисленное значение Δs:

$$s = s + \Delta s. \quad (18)$$

В начальный момент (при $t = 0$) имеем $s = 0$.

Однако осталось неясным, как определить значение c, необходимое для вычисления s?

Значение c можно найти, используя тот же принцип, что и при определении s.

Найдем производную от c по t:

$$\frac{dc}{dt} = -\sin\left(\frac{1}{a} \cdot t\right). \quad (19)$$

Используя формулы (5), (6) и (20) получаем дифференциальное уравнение:

$$a^2 \cdot \left(-\frac{dc}{dt}\right)^2 + c^2 = a^2. \quad (20)$$

Преобразуем его в следующий вид:

$$(-dc)^2 = (dt)^2 \cdot \frac{a^2 - c^2}{a^2}. \quad (21)$$

Запишем уравнение (22) в приближенном виде:

$$(-\Delta c)^2 \approx (\Delta t)^2 \cdot \frac{a^2 - c^2}{a^2}. \quad (22)$$

Поскольку $\Delta t = 1$, получаем

$$(-\Delta c)^2 \approx \frac{a^2 - c^2}{a^2}. \quad (23)$$

Используя формулу (6), преобразуем формулу (24):

$$(-\Delta c)^2 \approx \frac{s^2}{a^2}. \quad (24)$$

Можно также записать (25) в виде

$$a^2 \cdot (-\Delta c)^2 \approx s^2. \quad (25)$$

Из формулы (25) видно, что при $\Delta t = 1$ соблюдается неравенство $|\Delta c| \le 1$ (значение модуля синуса не может превысить 1). Поскольку необходимо получить целые значения c, может быть всего два варианта: $|\Delta c| = 1$ или $|\Delta c| = 0$.

Для рассматриваемого участка кривой $t \in [0, a \cdot \pi / 2]$. При этом $\Delta c \le 0$ (функция косинусоидального сигнала убывает), а $s \ge 0$.

Извлечем квадратный корень из обеих частей приближенного равенства (26):

$$-a \cdot \Delta c \approx s. \quad (26)$$

Тогда из (27) можно получить

$$\begin{cases} 0 \approx s, & \text{если} \quad \Delta c = 0, \\ a \approx s, & \text{если} \quad \Delta c = -1. \end{cases} \quad (27)$$

Оценим погрешность вычисления выражения $-a \cdot \Delta c$:

$$\begin{cases} e_{C0} = e_C - s, & \text{если} \quad \Delta c = 0, \\ e_{C1} = e_C + a - s, & \text{если} \quad \Delta c = -1. \end{cases} \quad (28)$$

Здесь e_C – сумма погрешностей вычисления выражения $-a \cdot \Delta c$ для предыдущих значений c. По сути, e_C – это погрешность вычисления c, умноженная на a.

Выберем из двух вариантов погрешностей e_{C0} и e_{C1} минимальную по модулю и определим оптимальную для нас Δc:

$$\Delta c = \begin{cases} 0, & \text{если} \quad |e_{C0}| < |e_{C1}|, \\ -1, & \text{если} \quad |e_{C0}| \geq |e_{C1}|. \end{cases} \tag{29}$$

Запомним значение накопленной погрешности e_C:

$$e_C = \begin{cases} e_{C0}, & \text{если} \quad |e_{C0}| < |e_{C1}|, \\ e_{C1}, & \text{если} \quad |e_{C0}| \geq |e_{C1}|. \end{cases} \tag{30}$$

Остается добавить к предыдущему значению c вычисленное значение Δc, которое может быть нулевым или отрицательным:

$$c = c + \Delta c\ . \tag{31}$$

В начальный момент (при $t = 0$) имеем $c = a$.

Таким образом, мы одновременно находим как значение синусоидального сигнала s, так и значение косинусоидального сигнала c. При этом для вычисления s используется имеющееся значение c, а для вычисления c – имеющееся значение s.

Остается лишь адаптировать формулы (16–19, 29–32) для работы с произвольными значениями t. Результирующее значение синусоидального сигнала с учетом квадранта, к которому принадлежит t, может быть найдено следующим образом:

$$s(t) = \begin{cases} s, & \text{если} \quad t \in [0 + 2\pi n, a \cdot \pi / 2 + 2\pi n) \\ \text{c}, & \text{если} \quad t \in [a \cdot \pi / 2 + 2\pi n, a \cdot \pi + 2\pi n) \\ -\text{s}, & \text{если} \quad t \in [a \cdot \pi + 2\pi n, a \cdot 3\pi / 2 + 2\pi n) \\ -c, & \text{если} \quad t \in [a \cdot 3\pi / 2 + 2\pi n, a \cdot 2\pi + 2\pi n), \end{cases} \tag{32}$$

где $n = 0,1,2...$ – номер периода сигнала.

Схема предлагаемого алгоритма приведена на рис. 1. Назовем этот алгоритм **целочисленным волновым алгоритмом** (Integer Wave Algorithm).

В алгоритме применяются только операции сложения и вычитания, ни одна мультипликативная операция не требуется.

Экспериментальные исследования

Целочисленный волновой алгоритм был реализован на языках C и C++. Ниже приведен текст консольного приложения на языке C, написанного в среде программирования Qt Creator 4.0.0. Автор позволил себе использовать однострочные комментарии в стиле C++.

```
#include <stdint.h>
#include "stdio.h"

#define A 50  // Амплитуда сигнала
#define N 2   // Количество периодов сигнала

int main(void)
{
  int32_t a;   // Амплитуда
  int32_t s;   // Значение sin в пределах
               // 1-го квадранта
  int32_t e_s; // Ошибка вычисления sin
  int32_t sin; // Значение sin
  int32_t c;   // Значение cos в пределах
               // 1-го квадранта
  int32_t e_c; // Ошибка вычисления cos
  uint32_t Q;  // Квадрант
  uint32_t t;  // Номер отсчета
  uint32_t n;  // Номер периода
  // Вспомогательные переменные
  int32_t e0, e1, ae0, ae1;

  a = A; Q = 1;
  s = 0; e_s = 0;
  c = A; e_c = 0;
  t = 0; sin = s;

  printf ("%8d\t%8d\n", t, sin);
  for (t = 1, n = 1; n <= N; t++)
  {
    // Оценка ошибки при вычислении c
    e0 = e_c - s; // Если c = c + 0
    e1 = e0 + a;  // Если c = c - 1
    ae0 = ((e0 < 0) ? -e0 : e0); // |e0|
    ae1 = ((e1 < 0) ? -e1 : e1); // |e1|

    // Выбираем вариант, при котором
    // модуль ошибки вычисления c меньше
    if (ae1 < ae0) // |e1| < |e0|
    {
      e_c = e1;
      c--;
    }
    else
      e_c = e0;

    // Оценка ошибки при вычислении s
    e0 = e_s - c;  // Если s = s + 0
    e1 = e0 + a;   // Если s = s + 1
    ae0 = ((e0 < 0) ? -e0 : e0); // |e0|
    ae1 = ((e1 < 0) ? -e1 : e1); // |e1|

    // Выбираем вариант, при котором
    // модуль ошибки вычисления s меньше
    if (ae1 < ae0) // |e1| < |e0|
    {
      e_s = e1;
      s++;
    }
    else
      e_s = e0;

    // Пересчитать результат
    // в зависимости от квадранта
    switch (Q)
    {
      case 1: // Квадрант 1
        sin = s;
        break;
      case 2: // Квадрант 2
        sin = c;
        break;
      case 3: // Квадрант 3
        sin = -s;
        break;
      case 4: // Квадрант 4
        sin = -c;
        break;
    }

    if (c == 0) // Квадрант закончен
    {
      s = 0; e_s = 0;
      c = a; e_c = 0;
      if (++Q > 4) // Завершили полный период
      {
        Q = 1;     // Вернулись к квадранту 1
        n++;
      }
    }
    printf ("%8d\t%8d\n", t, sin);
  }
  return 0;
}
```

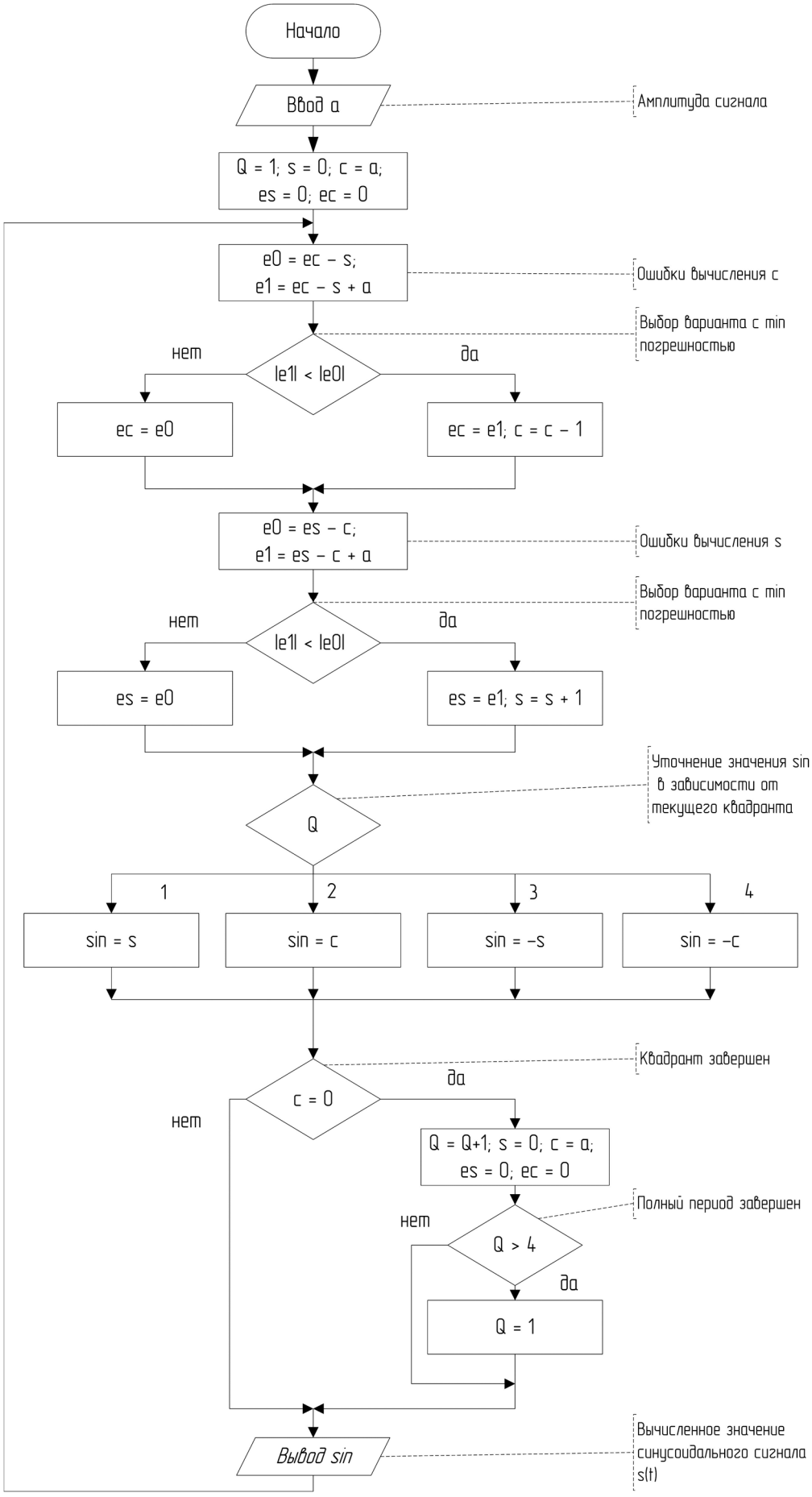

Рис. 1. Целочисленный волновой алгоритм

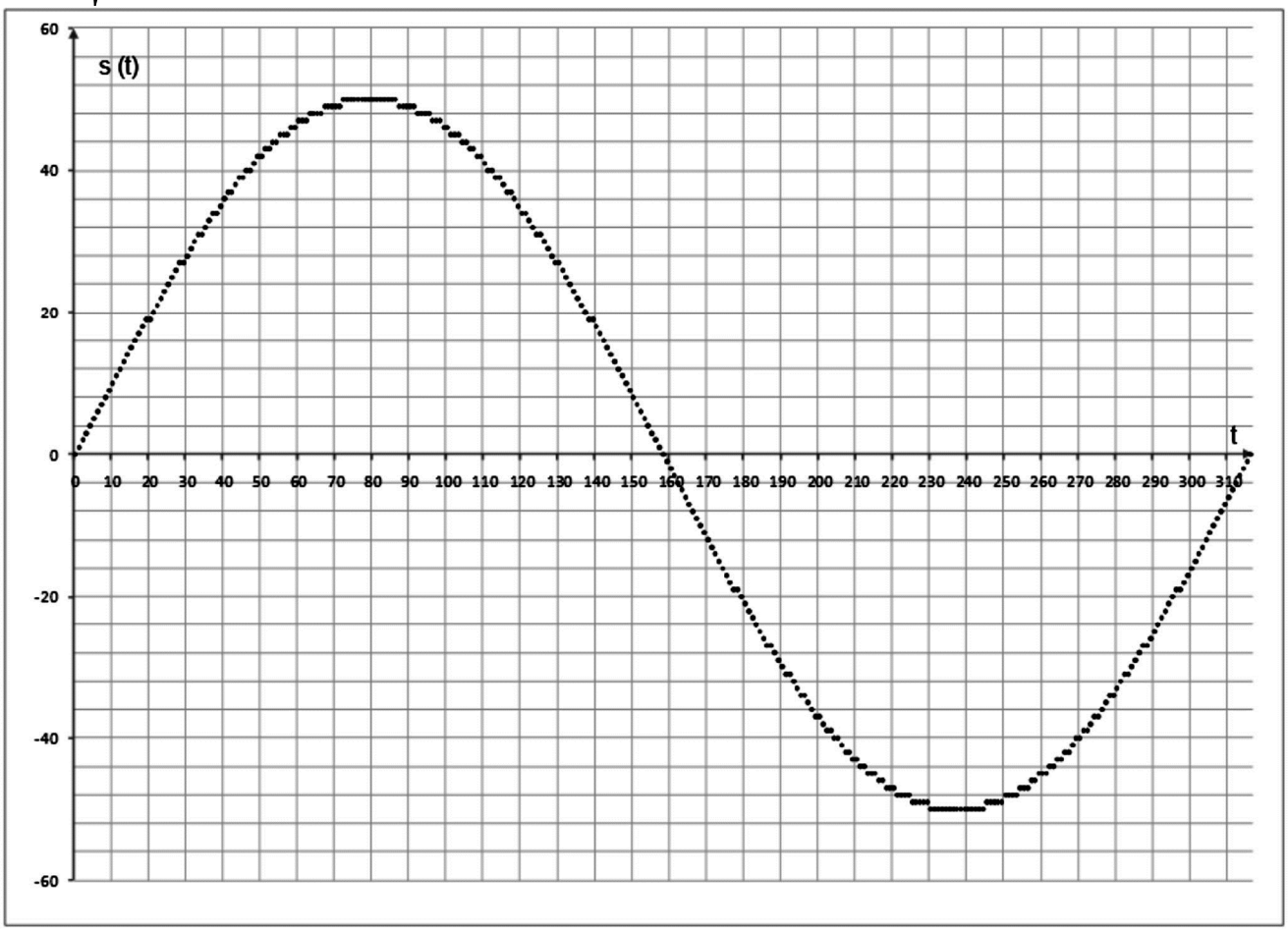

Рис. 2. Результат работы алгоритма

На рис. 2 показан результат работы целочисленного волнового алгоритма для $a = 50$.

Для читателей, пожелавших реализовать и исследовать эту программу под операционной системой Windows 7 и выше, хотелось бы посоветовать следующее. Любое консольное приложение можно запускать из командной строки с параметром clip, например:

```
QTWaveConsole.exe | clip
```

При этом вывод на экран будет перенаправлен в буфер обмена, что позволит легко сохранить и проанализировать результаты выполнения программы.

Изначально автор пытался решить поставленную задачу с помощью алгоритма Брезенхема [4–5], но положительного результата достигнуть не смог. Идея заключалась в том, чтобы в качестве искомого значения s принять значение координаты y очередной точки рисуемой окружности.

К сожалению, алгоритм Брезенхема и иные известные алгоритмы построения окружности не гарантируют, что при заданном радиусе окружности a получится количество точек, которое можно заранее оценить. Это приводит к тому, что предел отношения количества шагов T алгоритма Брезенхема (по сути, количество точек), требуемых для построения окружности, к диаметру окружности $2a$, не стремится к числу π:

$$\lim_{a\to\infty}\frac{T}{2a} \neq \pi . \qquad (33)$$

Эксперименты показали, что этого предела для алгоритма Брезенхема вовсе не существует. Другими словами, задав некоторую амплитуду a синусоидального сигнала, мы не будем знать, какой у него получится период.

Для предложенного алгоритма экспериментальная проверка показала наличие предела

$$\lim_{a\to\infty}\frac{T}{2a} = \pi , \qquad (34)$$

где T – количество шагов, требуемое для генерации полного периода синусоидального сигнала. Тогда, задав достаточно большую амплитуду a, можно говорить, что период T синусоидального сигнала будет определяться по формуле

$$T \cong 2\pi \cdot a . \qquad (35)$$

В табл. 1 приведены результаты экспериментальной проверки предела (35). Жирным шрифтом отмечены разряды, совпадающие с истинным значением числа π. Из таблицы видно, что при увеличении a отношение периода полученного сигнала к его удвоенной амплитуде действительно стремится к числу π.

Предложенный алгоритм можно использовать и для построения окружности, но его эффективность будет ниже, чем у алгоритмов Брезенхема, Мичнера или алгоритма Midpoint. Кроме того, полученная окружность будет менее гладкой. Тем не менее, может оказаться небезынтересным посмотреть результаты такого применения (рис. 3).

Таблица 1 – Отношение полученного периода синусоидального сигнала к его удвоенной амплитуде

a	$\frac{T}{2a}$
10^2-1	**3.1**31313131313
10^2	**3.1**40000000000
10^2+1	**3.1**28712871287
10^3-1	**3.14**1141141141
10^3	**3.14**0000000000
10^3+1	**3.14**0859140859
10^4-1	**3.141**514151415
10^4	**3.141**400000000
10^4+1	**3.141**485851415
10^5-1	**3.1415**91415914
10^5	**3.1415**80000000
10^5+1	**3.1415**88584114
10^6-1	**3.14159**1141591
10^6	**3.14159**2000000
10^6+1	**3.14159**0858409
10^7-1	**3.141592**514159
10^7	**3.141592**600000
10^7+1	**3.141592**485841
10^8-1	**3.1415926**51416
10^8	**3.1415926**40000
10^8+1	**3.1415926**48584
10^9-1	**3.14159265**3142
10^9	**3.14159265**2000
10^9+1	**3.14159265**2858

На рисунке представлены фрагменты окружности радиусом 50 точек, построенные с использованием целочисленного волнового алгоритма (слева) и алгоритма Midpoint [10] (справа). Стрелками показаны дополнительные точки, появляющиеся на окружности, построенной по предлагаемому алгоритму. Эти точки делают окружность менее гладкой и неравномерной по толщине, но зато они позволяют выровнять шаг изменения угла, что очень важно при генерации синусоидального сигнала.

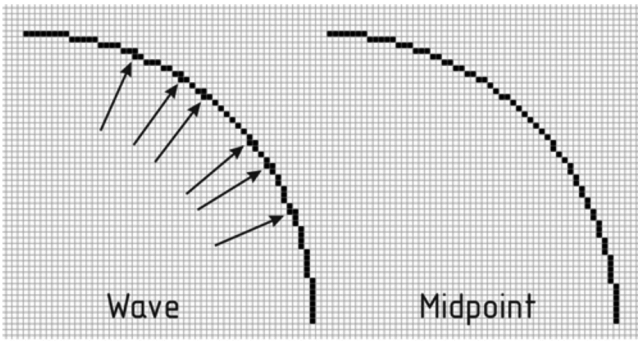

Рис. 3. Фрагмент окружности

Предлагаемый алгоритм также был реализован на языке С для микроконтроллеров (МК) семейства 1986ВЕ9х фирмы Миландр, имеющих архитектуру ARM Cortex M3. Производилось периодическое (по срабатыванию аппаратного таймера-счетчика) вычисление значений синусоидального сигнала. Полученное значение подавалось на вход цифро-аналогового преобразователя (ЦАП). При тактовой частоте процессорного ядра МК, равной 80 МГц, время, требуемое на вычисление одного значения сигнала, не превышало 1,1 мкс. При этом удавалось достичь частоты дискретизации сигнала до 500 кГц.

Заключение

Предложен полностью целочисленный алгоритм генерации синусоидального сигнала, использующий только аддитивные операции. Представляется возможным последующая реализация алгоритма на базе ПЛИС.

Литература

1. Захаров А.В. Алгоритмы CORDIC. Современное состояние и перспективы / А. В. Захаров, В. М. Хачумов // Труды международной конференции «Программные системы: теория и приложения».– Переславль-Залесский, 2004.– Т.1.– С. 353–372.

2. Ashrita T., Chidambara Rao. K. «Sine Wave Generation Using CORDIC Algorithm» IJECT Vol. 3, Issue 4, Oct - Dec 2012 ISSN: 2230-7109 (Online) | ISSN: 2230-9543 (Print).

3. Beeler, M., Gosper, R.W., and Schroeppel, R. HAKMEM. MIT AI Memo 239, Feb. 29, 1972. Item 149 (Minsky).

4. Bresenham J.E. Algorithm for computer control of a digital plotter// IBM Systems Journal, vol. 4, No. 1, pp. 25–30, 1965.

5. Bresenham J., A Linear Algoritm for Incremental Didgital Display of Circular Arcs, CACM, vol. 20, pp. 100-106, 1977.

6. Pitteway M.L.V. Algorithm for drawing ellipses or hyper-bolae with a digital plotter. Comptr. J. 10, 3 (Nov. 1967), p. 282-289.

7. Pitteway M.L.V. Integer circles: some further thoughts. Compt. Graphics and Image Processing 3, 3 (Sept. 1974), 262-265.

8. Salomon D. The Computer Graphics Manual, Texts in Computer Science, DOI 10.1007/978-0-85729-886-7, Springer-Verlag London Limited 2011.

9. «Fast iterative circles (and ellipses, and other figures)» (http://cabezal.com/misc/minsky-circles.html, дата просмотра 15.12.2017).

10. «Midpoint circle algorithm» (https://en.wikipedia.org/wiki/Midpoint_circle_algorithm, дата просмотра 15.12.2017).

11. «Drawcircle: алгоритм Мичнера для 1/8 части окружности» (http://www.stepwood.com/mcucodes/2012/02/13/drawcircle-algoritm-michnera-dlya-18-chasti-okruzhno/, дата просмотра 15.12.2017).