2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

DOI: 10.26089/NumMet.v22r209 УДК: 519.677

Итерационные алгоритмы БПФ с высоким частотным разрешением

О. В. Осипов

Белгородский государственный технологический университет им. В. Г. Шухова, кафедра программного обеспечения вычислительной техники и автоматизированных систем, Белгород, Российская Федерация

ORCID: http://orcid.org/0000-0002-5812-5338, e-mail: osipov.fft@yandex.ru

Аннотация: В работе представлены три итерационных алгоритма быстрого преобразования Фурье с прореживанием по времени, имеющие алгоритмическую сложность $O(NR\log_2 N)$, где R — частотное разрешение спектральной характеристики (отношение длины набора частот к длине N набора отсчетов исходного сигнала). Алгоритмы отличаются способами организации вычислений: некоторые используют обратную перестановку битов, другие — дополнительные массивы. Приведены подробные вычислительные графы, а также блок-схемы разработанных алгоритмов. Полученные результаты можно использовать для улучшения отечественной электроники и программного обеспечения, а также включать в учебный процесс при подготовке инженеров в области цифровой обработки сигналов.

Ключевые слова: быстрое преобразование Фурье (БПФ), вычислительный граф, высокое разрешение, сдвиг частоты, частотно-временное разрешение, цифровая обработка сигналов (ЦОС), численный итерационный алгоритм БПФ, прямое БПФ, амплитудно-частотная характеристика (АЧХ), прореживание по времени.

Для цитирования: Осипов О.В. Итерационные алгоритмы БПФ с высоким частотным разрешением // Вычислительные методы и программирование. 2021. **22**, № 2. 123–137. doi: 10.26089/NumMet.v22r209.

Iterative FFT-algorithms with high frequency resolution

O. V. Osipov

Shukhov Belgorod State Technological University, Belgorod, Russia

ORCID: http://orcid.org/0000-0002-5812-5338, e-mail: osipov.fft@yandex.ru

Abstract: This paper presents three iterative algorithms for fast Fourier transform with decimation in time; these algorithms have the algorithmic complexity $O(N\cdot R\cdot\log_2 N)$, where R is the frequency resolution of the spectral characteristic (the ratio of the length of the frequency set to the length of the N set of samples of the source signal). The algorithms differ in the way they organize calculations: some use reverse bit permutation, while the others use additional arrays. Detailed computational graphs and flowcharts of the developed algorithms are provided. The results obtained can be used to improve domestic electronics and software as well as may be included in the training process for engineers in the field of digital signal processing.

Keywords: fast Fourier transform (FFT), computational graph, high resolution, frequency shift, time-frequency resolution, digital signal processing (DSP), numerical iterative FFT algorithm, forward FFT, amplitude-frequency response, decimation in time.

For citation: O. V. Osipov, "Iterative FFT-algorithms with high frequency resolution," Numerical Methods and Programming. **22** (2), 123–137 (2021). doi: 10.26089/NumMet.v22r209.

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

1. Введение. В зависимости от технических условий решаемых задач инженерам нередко приходится искать различные способы оптимизации алгоритмов БПФ [1]. При этом требуется учитывать архитектурные особенности современных вычислительных систем, основные из которых — способность к векторизации и распараллеливанию вычислений, а также структура кэш-памяти. Учет этих особенностей при построении алгоритмов позволяет существенно ускорить вычисления.

Ранние реализации алгоритмов БПФ по основанию "два" на Бейсике и Фортране были ориентированы на однопоточное исполнение и содержали в себе три цикла [2, с. 128–131; 3, с. 405]. Это упрощало индексацию элементов в массиве комплексных чисел и не требовало лишних целочисленных операций. Такие алгоритмы, реализованные на C++ [4], вполне успешно решают свои задачи и по сей день, например при обработке звука. Но с появлением многопроцессорных систем, особенно на базе графических ускорителей (GPU), появилась возможность ускорить вычисление БПФ. Можно сократить количество циклов до двух и сделать итерации внутреннего цикла независимыми друг от друга, что позволит легко распараллелить и, следовательно, ускорить алгоритм БПФ.

Некоторые способы повышения частотного разрешения перечислены в работе [5]. Предложен аналитический метод спектрального анализа сигналов с высоким разрешением в заданном диапазоне частот [6]. Известны программно-аппаратные способы анализа сигналов с высоким разрешением в узком диапазоне частот, например [7]. В работе [8] авторы увеличивают частотное разрешение спектральной характеристики на несколько порядков с использованием параболической и гауссовой интерполяции с различными окнами.

В статье рассмотрены, а затем улучшены три уже известных способа программной реализации итерационных алгоритмов БПФ с прореживанием по времени. Улучшения заключаются в увеличении частотного разрешения БПФ (т.е. уменьшения шага по частоте в результирующей выборке комплексных амплитуд) и приведении его к виду, удобному для распараллеливания. Эти алгоритмы могут быть полезны при решении различных задач спектрального анализа, требующих большой точности обрабатываемых данных, например [9, 10]. Для построенных алгоритмов кратко рассмотрены способы возможного распараллеливания и даны рекомендации для дальнейшей оптимизации.

2. Постановка задачи. Для дискретного комплексного сигнала $S_i (i = 0, \ldots, N-1)$ требуется вычислить прямое преобразование Фурье:

$$A(S, N, k) = \sum_{i=0}^{N-1} S_i e^{-2\pi j \frac{ik}{NR}}, \qquad k = 0, \ldots, NR-1, \quad N = 2^p, \quad p, R \in \mathbb{N}, \tag{1}$$

где R — отношение длины набора частот к длине N исходного сигнала (частотное разрешение), j — мнимая единица. Необходимо построить алгоритм вычисления (1) с прореживанием по времени для набора частот длиной NR, имеющий линейно-логарифмическую алгоритмическую сложность.

Искомый алгоритм должен вычислять набор комплексных амплитуд с шагом по частоте в R раз меньшим, чем это способен сделать стандартный алгоритм БПФ. При этом вычисления должны занять приемлемое время, а значения набора комплексных амплитуд должны точно совпадать со значениями, которые можно получить с использованием ДПФ на тех же значениях частот.

3. Решение задачи. Исходный сигнал S имеет длину N, а длина набора частот кратна длине сигнала и равна NR. При увеличении R шаг по частоте будет уменьшаться, а АЧХ сглаживаться. Построим алгоритм вычисления $A(S, N, k)$ с набором частот $k = 0, \ldots, NR-1$, кратно превышающим по длине исходный сигнал S.

Выполним прореживание по времени [11, с. 204–215; 12] исходного сигнала S и запишем дискретное преобразование Фурье для первой половины частот:

$$\begin{aligned} A(S, N, k) &= \sum_{i=0}^{N-1} S_i e^{-2\pi j \frac{ik}{NR}} = \\ &= \sum_{i=0}^{N/2-1} S_{2i} e^{-2\pi j \frac{2ik}{NR}} + e^{-2\pi j \frac{k}{NR}} \sum_{i=0}^{N/2-1} S_{2i+1} e^{-2\pi j \frac{2ik}{NR}}, \qquad k = 0, \ldots, \frac{NR}{2} - 1. \end{aligned} \tag{2}$$

Для записи выражения для второй половины частот $k = NR/2, \ldots, NR-1$ предварительно отметим,

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

что

$$e^{-2\pi j\frac{2i(k+NR/2)}{NR}} = e^{-2\pi j\left(\frac{2ik}{NR}+i\right)} = e^{-2\pi j\frac{2ik}{NR}},$$

$$e^{-2\pi j\frac{k+NR/2}{NR}} = e^{-2\pi j\left(\frac{k}{NR}+\frac{1}{2}\right)} = e^{-2\pi j\frac{k}{NR}-\pi j} = -e^{-2\pi j\frac{k}{NR}}.$$

Тогда

$$A\left(S,N,k+\frac{NR}{2}\right) = \sum_{i=0}^{N/2-1} S_{2i}e^{-2\pi j\frac{2ik}{NR}} - e^{-2\pi j\frac{k}{NR}}\sum_{i=0}^{N/2-1} S_{2i+1}e^{-2\pi j\frac{2ik}{NR}}, \quad k=0,\ldots,\frac{NR}{2}-1.$$

Обозначим поворачивающие множители W, четные $S^{(0)}$ и нечетные $S^{(1)}$ элементы последовательности S:

$$W(k,M) = e^{-2\pi j\frac{k}{M}}; \qquad S^{(0)} = \{S_{2i}\}_{i=0,\ldots,N/2-1}, \qquad S^{(1)} = \{S_{2i+1}\}_{i=0,\ldots,N/2-1}.$$

С использованием введенных обозначений перепишем сокращенно вышеприведенные преобразования:

$$\begin{aligned} A(S,N,k) &= A\left(S^{(0)},\frac{N}{2},k\right) + W(k,NR)\,A\left(S^{(1)},\frac{N}{2},k\right),\\ A\left(S,N,k+\frac{NR}{2}\right) &= A\left(S^{(0)},\frac{N}{2},k\right) - W(k,NR)\,A\left(S^{(1)},\frac{N}{2},k\right), \qquad k=0,\ldots,\frac{NR}{2}-1. \end{aligned} \tag{3}$$

Операцию (3) для одной отдельно взятой пары частот k и $k+NR/2$ принято называть “бабочкой” [13]. Продолжим рекуррентные преобразования от второго до последнего слоя вычислений:

$$\begin{aligned} A\left(S,\frac{N}{2},k\right) &= A\left(S^{(0)},\frac{N}{4},k\right) + W\left(k,\frac{NR}{2}\right)\,A\left(S^{(1)},\frac{N}{4},k\right),\\ A\left(S,\frac{N}{2},k+\frac{NR}{4}\right) &= A\left(S^{(0)},\frac{N}{4},k\right) - W\left(k,\frac{NR}{2}\right)\,A\left(S^{(1)},\frac{N}{4},k\right), \qquad k=0,\ldots,\frac{NR}{4}-1;\\ &\ldots\\ A(S,2,k) &= A(S_0,1,k) + W(k,2R)\,A(S_1,1,k),\\ A(S,2,k+R) &= A(S_0,1,k) - W(k,2R)\,A(S_1,1,k), \qquad k=0,\ldots,R-1;\\ A(S,1,k) &= S_0. \end{aligned} \tag{4}$$

Можно заметить, что преобразования (2)–(4) сохранят свою силу, если в них всюду заменить частоту k на $k+\Delta k$, т.е. сдвинуть набор частот на произвольное значение Δk. Частотный сдвиг не нарушит рекуррентное разложение, т.е. позволит применить операцию “бабочка” к элементам последовательности. В этом случае будет найдено БПФ для набора частот $k+\Delta k$, где $k=0,\ldots,NR-1$. Эту возможность также можно использовать при построении алгоритмов, о чем более подробно написано в [5].

Рассмотрим несколько алгоритмов вычисления БПФ с высоким частотным разрешением, отличающихся способом перестановок комплексных чисел в массивах, объемом используемой памяти и наличием (или отсутствием) обратной перестановки битов при инициализации массивов. Идея распараллеливания всех алгоритмов основана на том, что по номеру итерации i внутреннего цикла можно взаимно однозначно индексировать пару комплексных чисел в массиве и выполнить над ними операцию “бабочка”.

Алгоритм 1. Быстрое преобразование Фурье с использованием дополнительного массива. Этот алгоритм имеет сложную схему перестановок между слоями вычислений, но не требует обратную перестановку битов в индексах при инициализации массивов. Это обеспечивается использованием дополнительного массива, что требует в 1.5–2 раза больше памяти. На рис. 1 рассмотрим направленный (слева направо) граф вычислений при $N=16$, $R=1$. На рисунке видно, что для одной отдельно взятой операции “бабочка” (например, вычисление пары (B_0,B_8), (A_0,A_8) или (A_1,A_9)) характерно следующее:

1. На первой итерации (первый слой вычислений) в операции участвуют элементы, индексы которых (в правой части каждого выражения) имеют разность $NR/2=8$; на второй — 4; на третьей — 2; на последней — 1.
2. Индексы элементов в левой части двух выражений всегда имеют разность $NR/2=8$. На рис. 1 синие линии слева направо соединяются с выражениями со знаком “−” в нижней половине графа, а черные — с выражениями со знаком “+” в верхней половине графа.

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

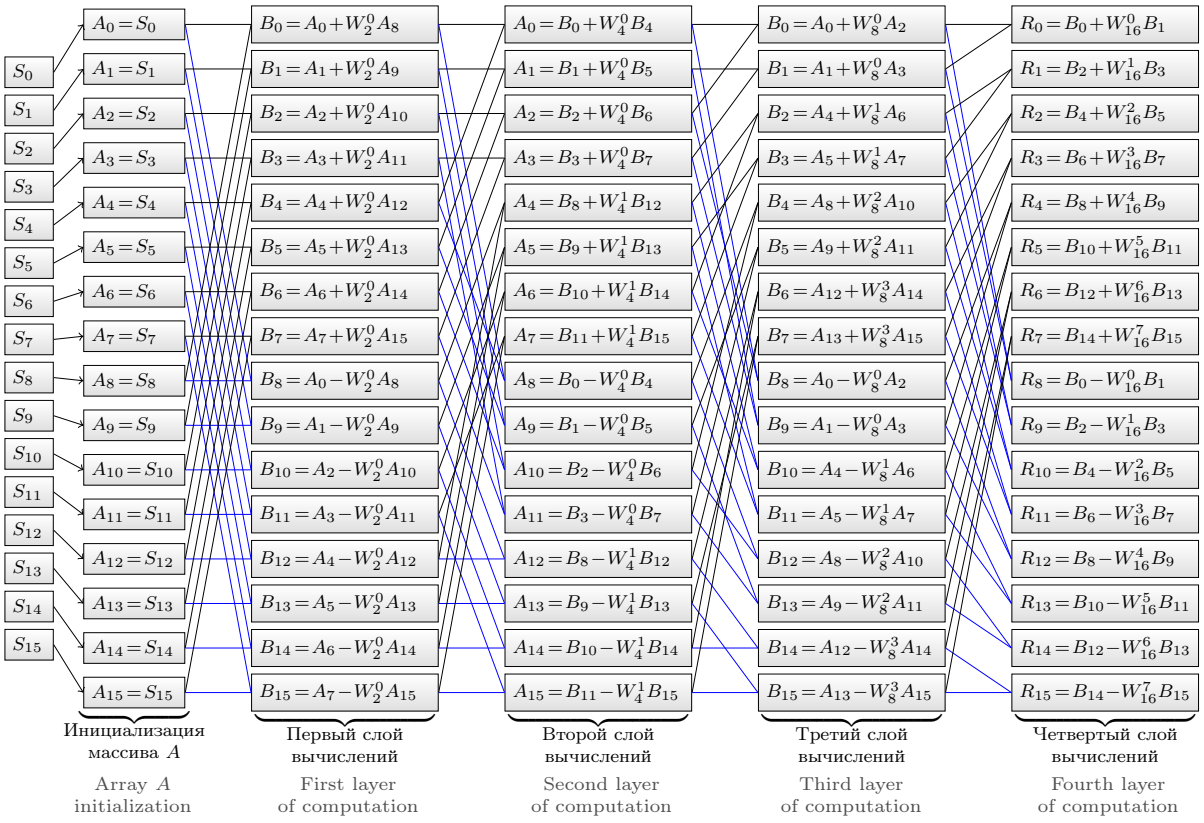

Рис. 1. Прямое БПФ с прореживанием по времени и дополнительным массивом при $N = 16$, $R = 1$

Fig. 1. Direct FFT with time decimation and the additional array for $N = 16$, $R = 1$

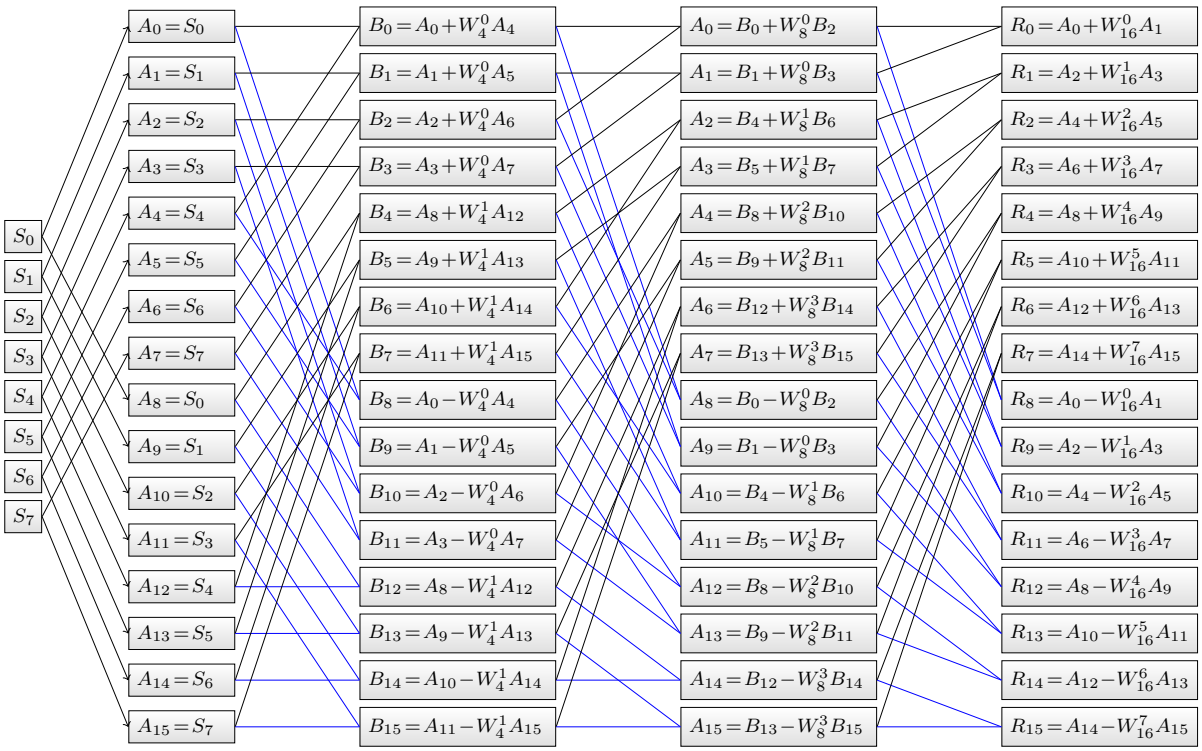

Рис. 2. Прямое БПФ с прореживанием по времени и дополнительным массивом при $N = 8$, $R = 2$

Fig. 2. Direct FFT with time decimation and the additional array for $N = 8$, $R = 2$

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

Обозначим на всех графах поворачивающие множители:

$$W_M^k = e^{-2\pi j \cdot k/M} = \cos(2\pi k/M) - j \cdot \sin(2\pi k/M).$$

Согласно рекуррентным преобразованиям (2)–(4) построим граф вычислений (рис. 2) при $N = 8$, $R = 2$.

Здесь исходная последовательность A получена путем копирования массива S и его повторения R раз (второй столбец). Рассмотренные две схемы очень похожи: вторая схема (рис. 2) получается удалением первого слоя из первой схемы (рис. 1). Соответственно, при каждом увеличении R в два раза при постоянном NR количество слоев уменьшается на единицу. Отметим, что R не обязательно должно быть степенью двойки.

Объясним работу алгоритма **БПФ_МАСС** (рис. 3). Первый цикл требует выполнения $\log_2 N$ итераций, второй цикл — NR итераций. Итерации внутреннего цикла независимы друг от друга, что очень удобно для возможного распараллеливания.

Смысл обозначений S, N, A, R, введенных ранее, в списке аргументов уже известен. Параметр d определяет знак угла поворота, но условно можно считать его направлением БПФ: $d = 1$ — прямое, $d = -1$ — обратное. При $d = -1$ алгоритм целесообразно использовать при $R = 1$, но при этом нужно значения в массиве A разделить на N после завершения алгоритма. Параметр Δk — частотный сдвиг (по умолчанию $\Delta k = 0$). Используя этот параметр, можно построить алгоритм иначе [5]. К примеру, если вычислить БПФ для одного и того же сигнала S с использованием функции **БПФ_МАСС** два раза с параметрами $\Delta k = 0$ и $\Delta k = 1/2$, то полученные в результате прямого БПФ последовательности $A^{(0)}$ и $A^{(1/2)}$ можно объединить в один массив и получить АЧХ с бóльшим в два раза частотным разрешением.

На всех блок-схемах имеется переменная e, которая не участвует в вычислениях, но добавлена для дальнейшей оптимизации. Сопроцессор вычисляет значения тригонометрических функций (sin, cos) сравнительно медленно, поэтому лучше вычислить поворачивающие множители $W(k, N)$ предварительно и сохранить в отдельном массиве. Значение переменной $e = -\alpha/\pi \cdot NR$ при $\Delta k = 0$ может быть использовано для индексации элементов такого массива. Для этого в блок-схеме (рис. 3) достаточно заменить вычисление W на обращение к элементам массива W_e, которые нужно вычислить заранее:

$$W_i = \cos(\alpha) + j \cdot d \cdot \sin(\alpha),$$

где $\alpha = -\pi i/(NR)$, $i = 0, \ldots, NR - 1$. Такая замена позволяет ускорить выполнение БПФ примерно в 3–4 раза.

В алгоритме (рис. 3) вещественными являются только вычисление α, W и операция “бабочка”. Большая часть операций и переменных — целочисленные. Значение m — период функции $W(k, M)$, а $m/2$ — количество поворачивающих множителей в слое. Переменная m увеличивается в 2 раза на каждой итерации

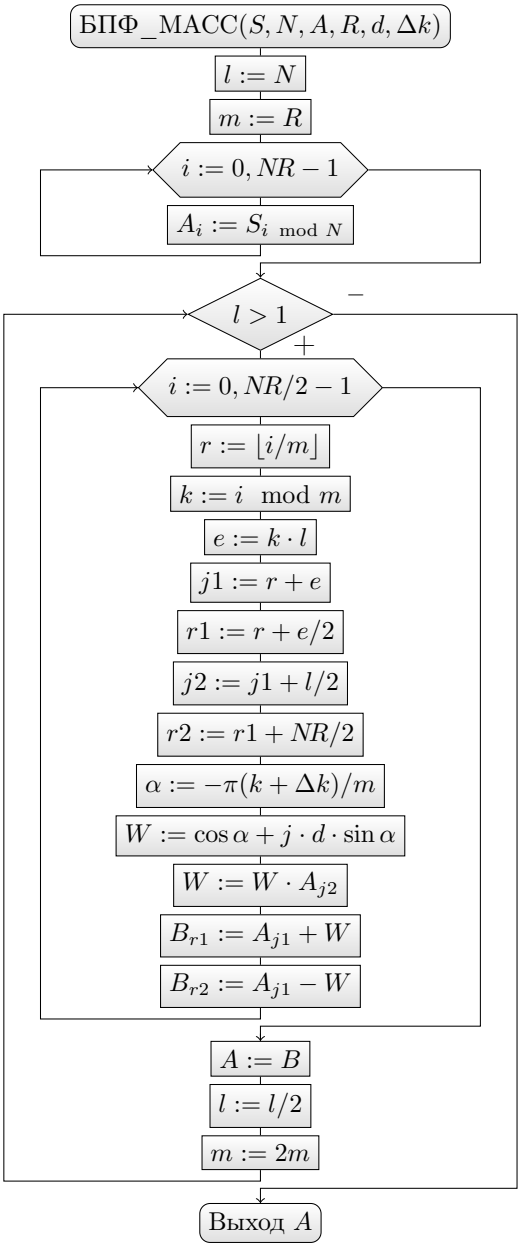

Рис. 3. Алгоритм БПФ с прореживанием по времени и дополнительным массивом

Fig. 3. The FFT algorithm with time decimation and the additional array

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

внешнего цикла, а l, наоборот, уменьшается в 2 раза. Переменные m, l выражают номер слоя и всегда связаны соотношением $ml = NR$. Значение k — частота в поворачивающем множителе $W(k, M)$ на текущей итерации. Пара (k, r) однозначно выражается через номер итерации i. Переменные $j1$, $j2$ — индексы элементов массива, участвующих в операции "бабочка" в правой части выражения, а $r1$, $r2$ — в левой части выражения. Вычисление этих индексов организовано таким образом, чтобы обойтись без обратной перестановки битов и упорядочить элементы результирующего массива A по возрастанию частот.

Рассмотрим реализацию данного алгоритма на языке C++ с использованием класса стандартной библиотеки `std::complex<double>` (листинг 1).

Листинг 1. Реализация алгоритма БПФ с дополнительным массивом на C++
Listing 1. Implementation of the FFT algorithm with an additional array in C++

```
void БПФ С МАССИВОМ(complex<double>* S, int N, complex<double>* A, int R, int d,
    double dk=0)
{
  int l = N, i, j1, j2, r1, r2, m = R, NR = N*R, e, k, r;
  double angle;
  for (i=0; i < NR; i++)
    A[i] = S[i % N];
  complex<double>* B = new complex<double>[NR], W;
  while(l > 1)
  {
    for (i=0; i < NR/2; i++)
    {
      r = i / m; k = i % m;
      e = k * l;
      j1 = r + e;
      r1 = r + e / 2;
      j2 = j1 + l / 2;
      r2 = r1 + NR / 2;
      // Бабочка
      angle = -M_PI * (k + dk) / m;
      W = complex<double>(cos(angle), d * sin(angle));
      W = W * A[j2];
      B[r1] = A[j1] + W;
      B[r2] = A[j1] - W;
    }
    memcpy(A, B, sizeof(complex<double>) * NR);
    l = l >> 1;
    m = m << 1;
  }
  delete []B;
}
```

Алгоритм 2. Быстрое преобразование Фурье с использованием обратной перестановки битов. Является самым известным алгоритмом БПФ [12–16]. Давно замечено, что все вычисления можно выполнить, используя один массив, предварительно переставив элементы в нем с позиций i на позиции $\mathrm{inverse}(i, N)$, где $\mathrm{inverse}(i, N)$ — обратная перестановка битов [17, с. 229] числа i в границах битовой маски N. Алгоритм использует мало памяти и имеет понятную схему перестановок, но не лишен недостатков. Процессоры (x86, ARM), к сожалению, не имеют отдельной инструкции для выполнения обратной перестановки битов, поэтому для этого необходим отдельный алгоритм, который отнимает время и замедляет БПФ.

Изменим данный алгоритм БПФ и добавим в него возможность увеличивать частотное разрешение. Построим граф вычислений при $N = 8$, $R = 2$ (рис. 4).

На рис. 4 (второй столбец) видно, что значения из массива S копируются в массив A рядом друг с другом парами (так как $R = 2$). При $R = 3$ они будут скопированы тройками, при $R = 4$ — четверками и т.д. При этом номер каждой пары (тройки, четверки и т.д.) значений S_i, скопированной в массиве A, получен обратной перестановкой бит числа i в границах битовой маски $N - 1$. Для этого в начало алгоритма (рис. 5) добавлен отдельный цикл для инициализации массива A. На блок-схеме (рис. 5) смысл переменных такой же, как и в алгоритме 1. Изменена только схема перестановок, общая структура и сложность алгоритма остались прежними. Так как не используется второй массив, а результат операций

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

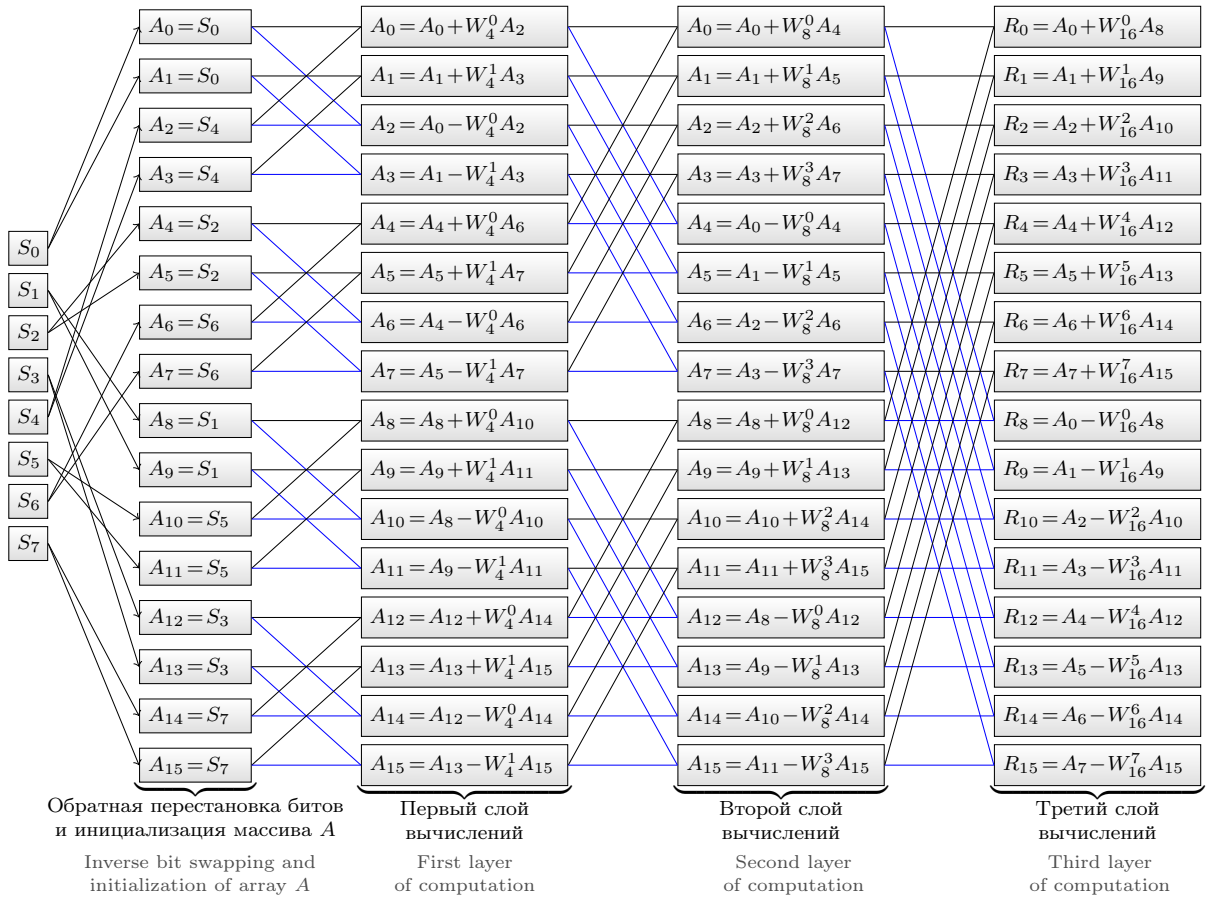

Рис. 4. Прямое БПФ с прореживанием по времени и обратной перестановкой битов при $N = 8$, $R = 2$

Fig. 4. Direct FFT with time decimation and inverse bit swapping for $N = 8$, $R = 2$

“бабочка” записывается на те же позиции массива A, в которых находились исходные комплексные числа, достаточно только двух переменных-индексов $j1$, $j2$ вместо четырех.

Реализация алгоритма БПФ с прореживанием по времени и обратной перестановкой битов на языке C++ представлена в листинге 2.

Листинг 2. Реализация алгоритма БПФ с прореживанием по времени и обратной перестановкой битов

Listing 2. Implementation of the FFT algorithm with time decimation and inverse bit swapping

```
void БПФ С ПЕРЕСТАНОВКОЙ БИТОВ(complex<double>* S, int N, complex<double>* A, int
    R, int d, double dk=0)
{
  int m = R, l, k, i, j1, j2, e, s = 0, t, NR = N*R;
  double angle;
  complex<double> E1, E2, W;
  // Обратная перестановка битов
  for (i=0; i < NR; i++)
  {
    k = inverse(i / R, N-1);
    A[i] = S[k];
  }
  l = N;
  while(l > 1)
  {
    for (i=0;i < NR/2;i++)
    {
      t = l / 2;
```

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

```
18          k = i / t;
19          e = k * l;
20          j1 = k + (2*m) * (i % t); // Индекс первой ячейки
21          j2 = j1 + m;              // Индекс второй ячейки
22          // Бабочка
23          angle = -M_PI * (k + dk) / m; // Угол поворота
24          W = complex<double>(cos(angle), sin(angle) * d);
25          W = W * A[j2];
26          A[j2] = A[j1] - W;
27          A[j1] = A[j1] + W;
28        }
29        l = l >> 1;
30        m = m << 1;
31      }
32    }
```

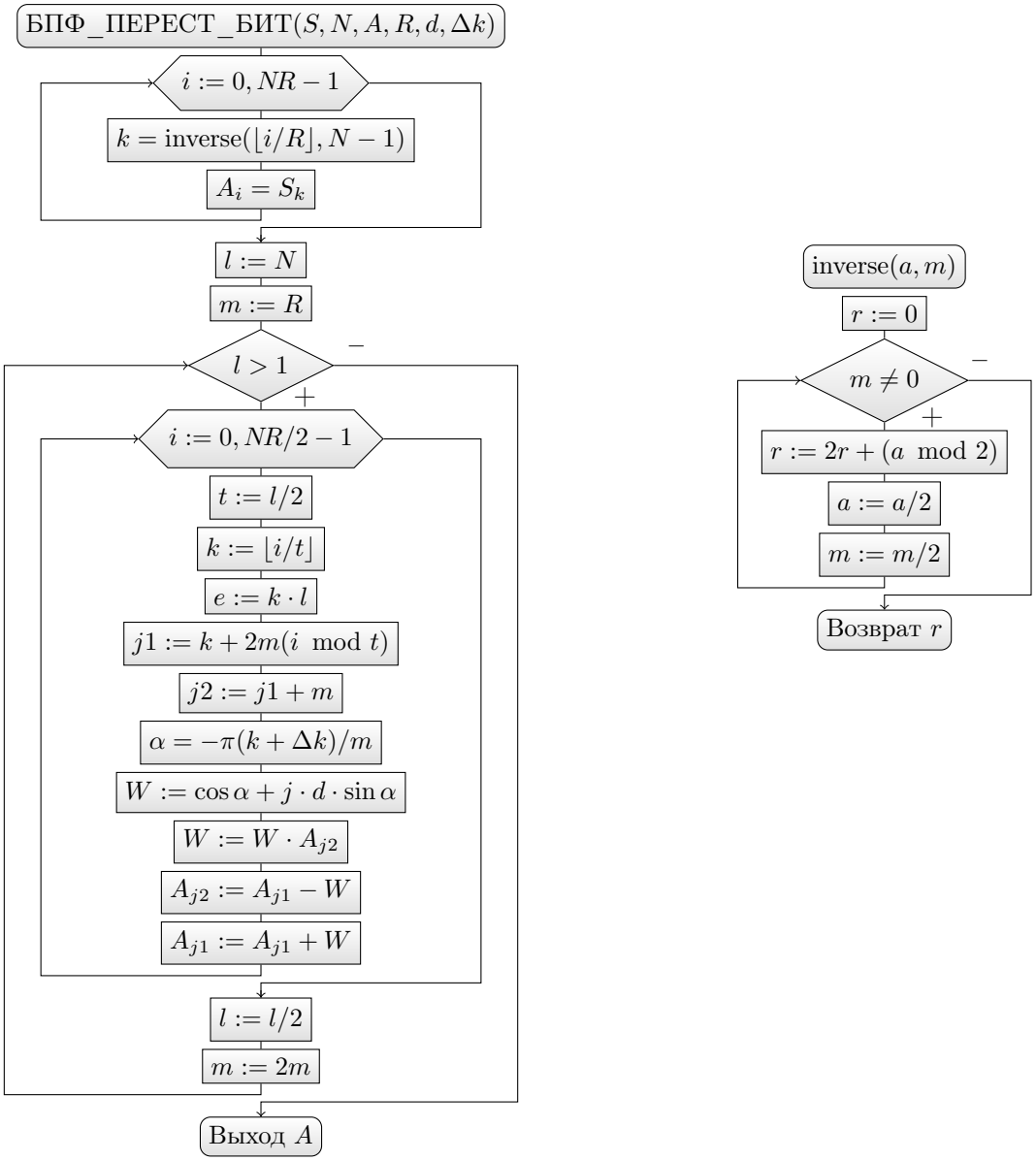

Рис. 5. Алгоритм БПФ с прореживанием по времени и обратной перестановкой битов

Fig. 5. FFT algorithm with time decimation and inverse bit swapping

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

Алгоритм 3. Быстрое преобразование Фурье с использованием обратной перестановки битов и дополнительного массива. Самым интересным с точки зрения выполнения перестановок между слоями является алгоритм, описанный в [18, 19]. В нем одна операция "бабочка" выполняет действия над двумя соседними элементами с индексами $2i$, $2i+1$ и записывает результат на позиции i, $i+NR/2$. При этом схема перестановок фиксированная и не меняется при переходе от текущего слоя к следующему (рис. 6). Данный алгоритм также можно изменить и добавить в него возможность задавать частотное разрешение. Для этого нужно инициализировать массив A (второй столбец) следующим образом: каждый элемент S_i скопировать R раз с шагом N на позиции $A_{\text{inverse}(i,N-1)+rN}$, $r = 0, \ldots, R-1$.

Так же как и в предыдущих алгоритмах, увеличение R в 2 раза при неизменном NR приводит к удалению одного слоя, как будто нужно пропустить одну итерацию внешнего цикла. Можно заметить, что в данном алгоритме числа считываются и записываются в массивы последовательно, что благоприятно влияет на работу с кэш-памятью. Фиксированная схема перестановок может очень упростить возможную аппаратную реализацию такого алгоритма. Приведем блок-схему (рис. 7) и программную реализацию данного алгоритма.

В рассмотренных алгоритмах операции умножения и деления на степени двойки можно заменить более быстрыми побитовыми сдвигами, а нахождение остатков от деления на степени двойки — побитовым умножением.

Реализация алгоритма БПФ с использованием обратной перестановки битов и дополнительного массива на языке C++ представлена в листинге 3.

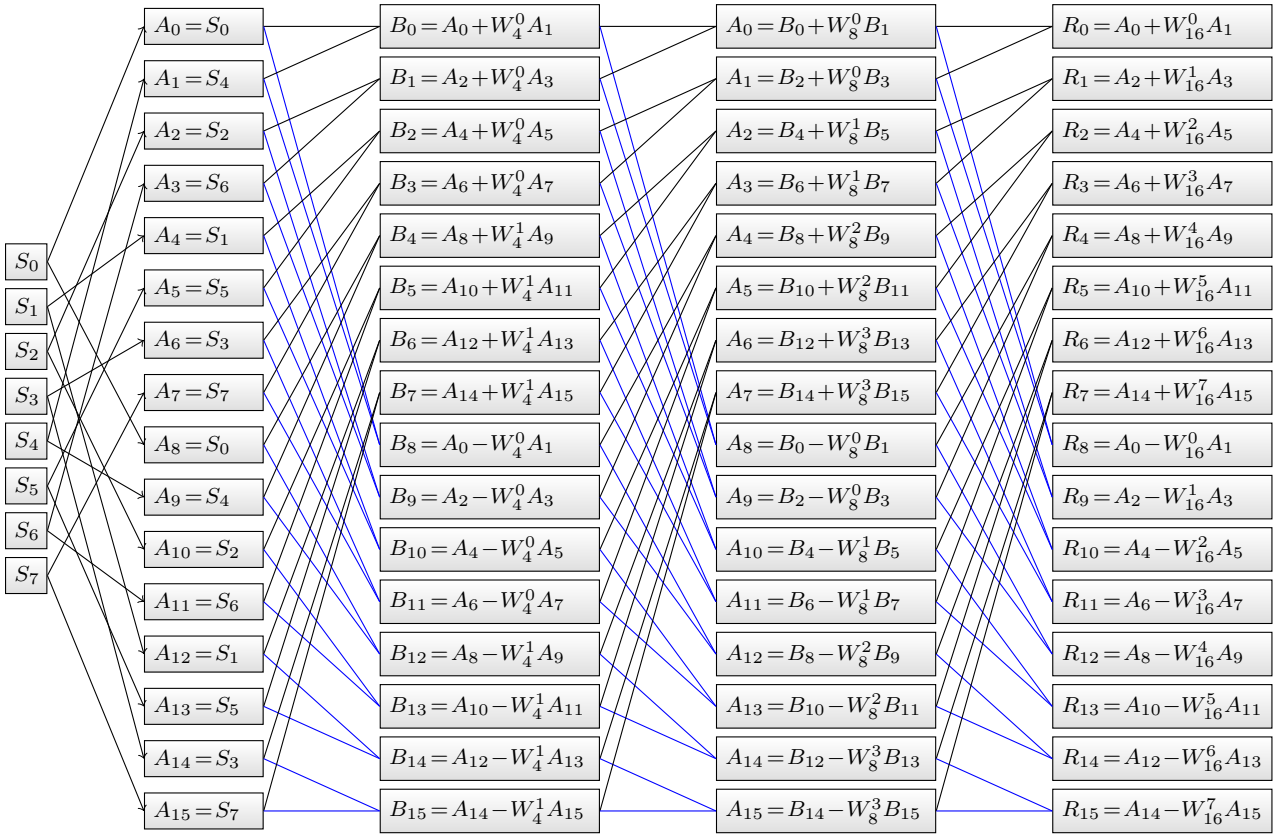

Рис. 6. Прямое БПФ с прореживанием по времени, обратной перестановкой битов и дополнительным массивом при $N = 8$, $R = 2$

Fig. 6. Direct FFT with time decimation, inverse bit swapping and the additional array for $N = 8$, $R = 2$

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

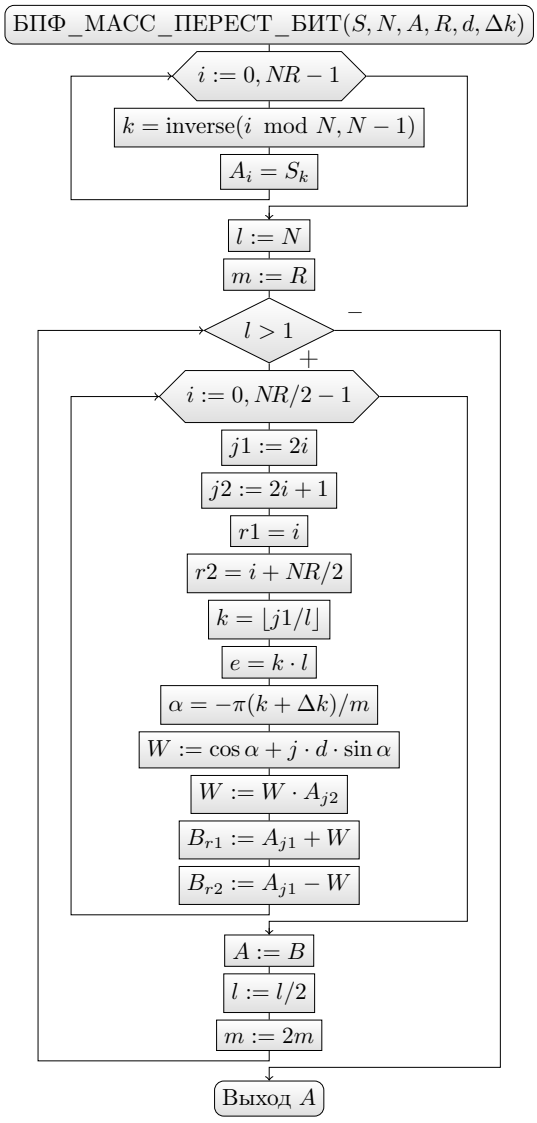

Рис. 7. Алгоритм БПФ с прореживанием по времени, обратной перестановкой битов и дополнительным массивом

Fig. 7. The FFT algorithm with time decimation, inverse bit swapping and the additional array

Листинг 3. Реализация алгоритма БПФ с прореживанием по времени и обратной перестановкой битов

Listing 3. Implementation of the FFT algorithm with time decimation and inverse bit swapping

```
1  void БПФ С ПЕРЕСТ БИТОВ И ВТОРЫМ
       МАССИВОМ(complex<double>* S, int N,
       complex<double>* A, int R, int d,
       double dk=0)
2  {
3    int l, i, j1, j2, r1, r2, m, NR = N*R,
       k, e;
4    double angle;
5    for (i=0; i < NR; i++)
6    {
7      k = inverse(i % N, N-1);
8      A[i] = S[k];
9    }
10   complex<double>* B = new
       complex<double>[NR], W;
11   l = N; m = R;
12   while(l > 1)
13   {
14     for (i=0; i < NR/2; i++)
15     {
16       j1 = 2*i; j2 = 2*i + 1;
17       r1 = i;   r2 = i + NR/2;
18       k = j1 / l;
19       e = k * l;
20       angle = -M_PI * (k + dk) / m;
21       W = polar<double>(1, d * angle);
22       W = W * A[j2];
23       B[r1] = A[j1] + W;
24       B[r2] = A[j1] - W;
25     }
26     memcpy(A, B, sizeof(complex<double>)
       * NR);
27     l = l >> 1;
28     m = m << 1;
29   }
30   delete []B;
31 }
```

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

4. Результаты численных экспериментов.

Эксперимент 1. Исследуем способность алгоритмов обнаруживать в сигнале две близкие по частоте гармоники. Дискретизируем сигнал

$$S(t) = 1000 \cdot \sin(2\pi \cdot 500t) + 900 \cdot \sin(2\pi \cdot 520t)$$

набором из $N = 2048$ отсчетов с частотой 44100 Гц и построим несколько АЧХ при различных значениях R.

Кривая с $R = 1$ на рис. 8 соответствует стандартному БПФ методом Кули–Тьюки с набором амплитуд длиной N. При увеличении R гармоники становятся все более различимыми, и при $R = 4$ уже видно, что их две. Таким образом, алгоритмы обладают большей различающей способностью по отношению к обычному БПФ. Здесь разность по частоте близка к $44100/N$ Гц. Эксперименты показывают, что при дальнейшем уменьшении этой разницы гармоники начинают постепенно объединяться на спектрограмме в одну.

Эксперимент 2. Построим несколько АЧХ реального сигнала при различных значениях R (рис. 9). Пусть дан фрагмент произвольного акустического сигнала с частотой дискретизации 44100 Гц, взятого из 16-битной звуковой дорожки с расширением `*.wav` и представленного набором из $N = 2048$ отсчетов. На графиках представлена только низкочастотная часть спектра: до 1000 Гц из 22050 возможных. Видно, что с ростом числа R функция АЧХ сглаживается, а шаг по частоте становится меньше. Уже при $R = 8$ можно вычислить частоты несущих гармоник с точностью 2.69 Гц. Больше всего теряется информации о низких частотах при $R = 1$. Используя любой из трех предложенных алгоритмов, можно получать необходимое частотное разрешение с линейными затратами процессорного времени. К примеру, для получения 4-кратного частотного разрешения нужно выполнить в 4 раза больше вычислений, чем по стандартному алгоритму БПФ, т.е. при $R = 1$.

Эксперимент 3. Оценим время работы разработанных алгоритмов. Для этого проведем ряд численных расчетов при различных значениях N, R. На рис. 10 представлены графики зависимости времени

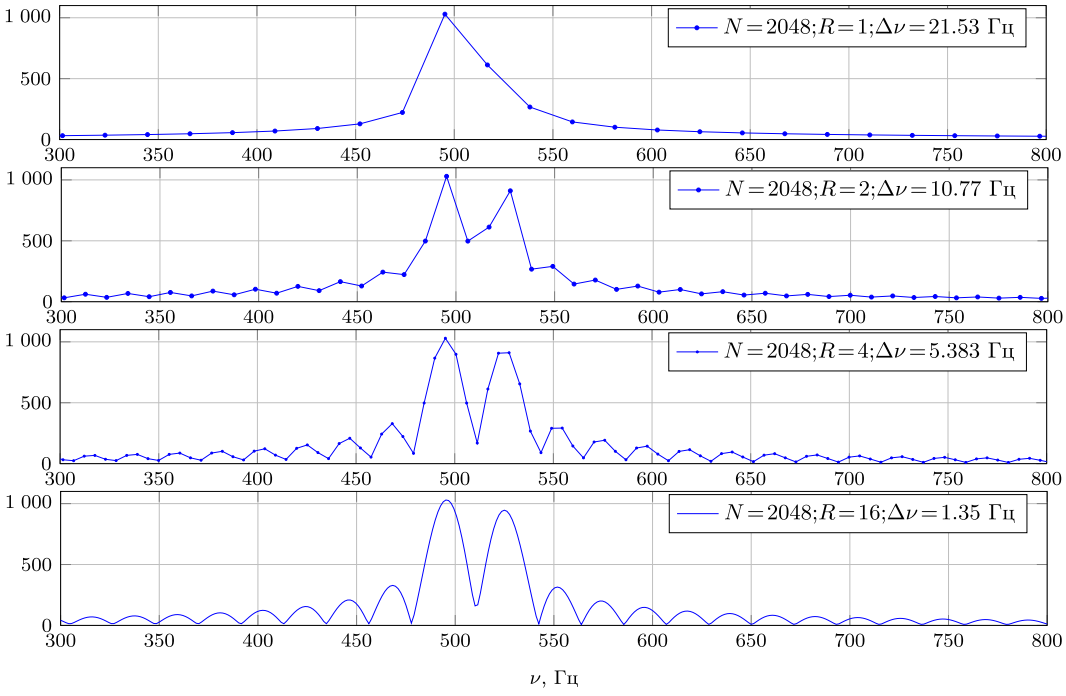

Рис. 8. АЧХ акустического сигнала с двумя гармониками при различных параметрах частотного разрешения R

Fig. 8. Frequency response of an acoustic signal with two harmonics at different parameters of the frequency resolution R

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

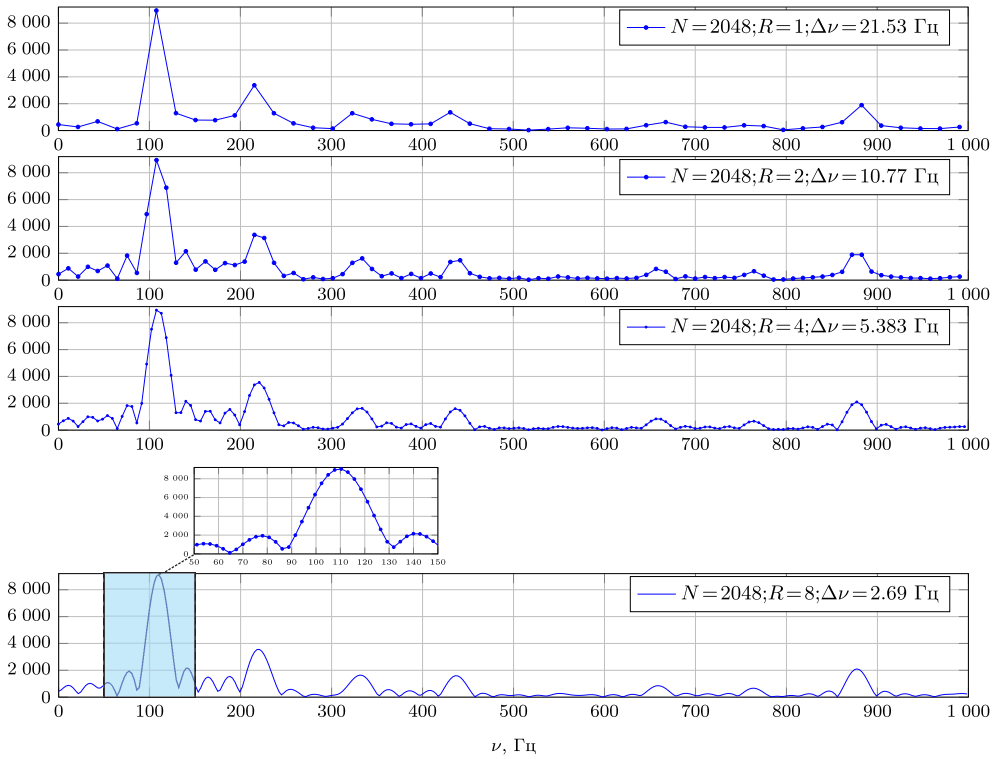

Рис. 9. Набор АЧХ реального акустического сигнала при различных параметрах частотного разрешения R

Fig 9. A set of frequency response of a real acoustic signal at various parameters of the frequency resolution R

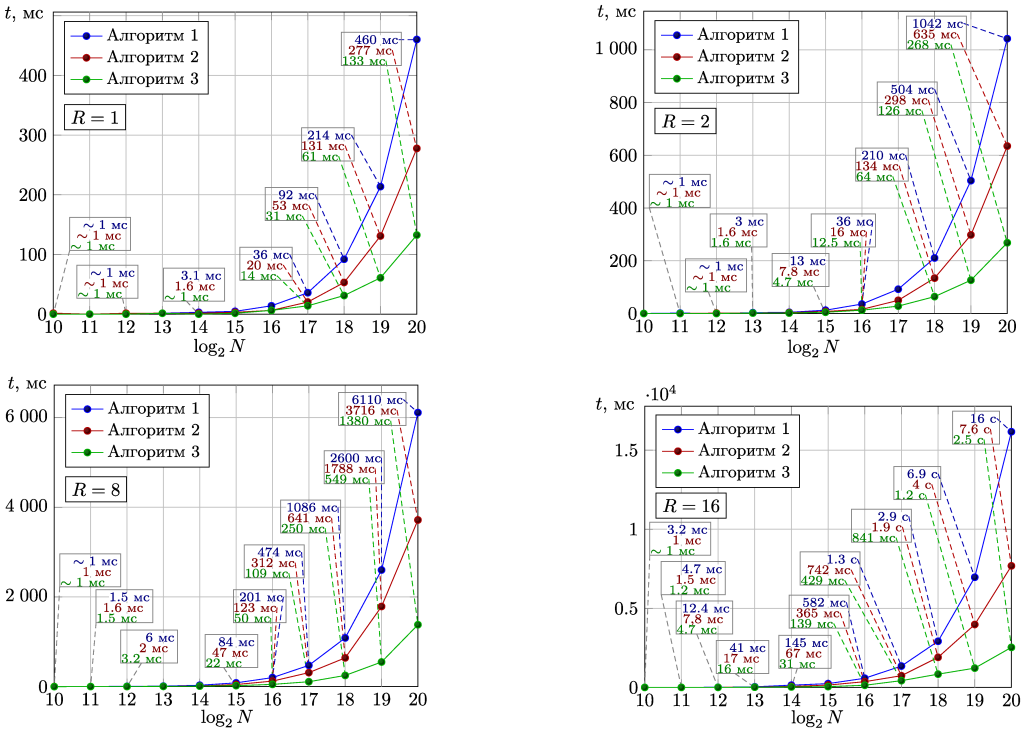

Рис. 10. Время работы алгоритмов БПФ при различных значениях длины N дискретного сигнала и параметрах частотного разрешения R

Fig. 10. The operating time of the FFT algorithms for different values of the length N of the discrete signal and the parameters of the frequency resolution R

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

выполнения БПФ от длины N исходного сигнала S в логарифмическом масштабе для $R = 1, 2, 8, 16$. Чтобы сравнение было объективным, алгоритмы 1–3 немного оптимизированы: поворачивающие множители и бит-реверсные индексы вычислены заранее и сохранены в отдельные массивы, действия над комплексными числами организованы без вложенных вызовов функций (класса std::complex).

Расчеты выполнены с двойной точностью (тип double) в однопоточном режиме на процессоре x86 с тактовой частотой 2,6 ГГц, размерами кэш-памяти L1 = 64 КБ, L2 = 512 КБ, L3 = 3 МБ. Использовался набор 32-разрядных инструкций x86 (без использования расширений AVX и SIMD) с максимальной оптимизацией компилятора MSVC v142. Самым эффективным оказался алгоритм 3 с дополнительным вторым массивом и предварительной перестановкой элементов согласно бит-реверсной адресации. Этот алгоритм выполняется в 3–4 раза быстрее остальных, потому что обрабатывает числа в памяти последовательно. Практика программирования в данном случае показывает, что не всегда быстрее работают те программы, которые выполняют меньше операций. Иногда более быстрыми оказываются те, которые эффективнее используют оперативную память. Относительная скорость алгоритма 3 не уменьшается с ростом NR, потому что количество кэш-обменов у него при любых значениях NR будет меньше, чем у остальных. Какого бы размера ни был массив, последовательно он обрабатывается все равно быстрее. При этом даже не обязательно, чтобы массив целиком попадал в какой-либо кэш-уровень.

На легендах (рис. 10) также показано время работы алгоритмов (в миллисекундах). Алгоритм 2 занимает для хранения массивов комплексных чисел, участвующих в вычислениях, $2^4 NR$ байт оперативной памяти, алгоритмы 1 и 3 — $2^5 NR$ байт.

5. Заключение.. Разработан алгоритм БПФ, по быстродействию сопоставимый с БПФ методом Кули–Тьюки, а по качеству с ДПФ. Конечно, невозможно удовлетворить этим двум свойствам одновременно, но можно найти нужный баланс между качеством и быстродействием, варьируя параметр R.

В результате работы были подробно рассмотрены и улучшены самые распространенные алгоритмы БПФ по основанию “два” с прореживанием по времени. Во-первых, алгоритмы приведены к единообразной структуре, которая удобна для дальнейшего распараллеливания; во-вторых, в них добавлена возможность задавать произвольное частотное разрешение. Полученные результаты полезны при решении инженерных задач в области спектрального анализа сигналов различной физической природы.

Статья подготовлена в рамках программы развития опорного университета на базе БГТУ им. В. Г. Шухова.

Список литературы

1. *Альтман Е.А.* Оптимизация вычислительной схемы быстрого преобразования Фурье // Омский научный вестник. 2008. № 1 (64). 149–151.
2. *Гольденберг Л.М., Матюшкин Б.Д., Поляк М.Н.* Цифровая обработка сигналов: Учеб. пособие для вузов. М.: Радио и связь, 1990.
3. *Рабинер Л., Гоулд Б.* Теория и применение цифровой обработки сигналов. М.: Мир, 1978.
4. Быстрое преобразование Фурье за $O(N \log N)$. https://e-maxx.ru/algo/fft_multiply.
5. *Осипов О.В.* Спектральный анализ дискретных сигналов с высоким частотным разрешением // Вычислительные методы и программирование. 2019. **20**. 270–282. doi 10.26089/NumMet.v20r324
6. *Пономарева Н.В.* Быстрое параметрическое преобразование Фурье для спектрального анализа сигналов с высоким разрешением в заданном частотном диапазоне // DSPA: Вопросы применения цифровой обработки сигналов. 2019. **9**, № 1. 28–32.
7. *Чан В.Н., Дам В.Н.* Реализация спектрального анализатора с высоким разрешением на FGPA // DSPA: Вопросы применения цифровой обработки сигналов. 2016. **6**, № 4. 725–729.
8. *Gasior M., Gonzalez J.L.* Improving FFT frequency measurement resolution by parabolic and Gaussian spectrum interpolation // AIP Conference Proceedings. 2004. **732**, N 1. 276–285.
9. *Сидоренко И.А., Кускова П.А.* О повышении точности спектрального анализа фонем при использовании звуковых редакторов // Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика. 2015. № 7. 188–193.
10. *Полешенков Д.Д., Басов О.О.* Способ выделения траектории частоты основного тона речи на основе частотной демодуляции // Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика. 2019. **46**, № 2. 359–366.
11. *Голд Б., Рэйдер Ч.* Цифровая обработка сигналов. М.: Сов. радио, 1973.
12. *Белов В.И., Панимаскин Е.И.* Быстрое преобразование Фурье (БПФ) с прореживанием по времени. https://docplayer.ru/25793573-Belov-v-i-panimaskin-e-i-bystroe-preobrazovanie-fure-bpf-s-prorezhivaniem-po-vremeni.html.

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

13. *Галанина Н.А.* Синтез функциональных модулей БПФ в СОК // Вестник Чувашского университета. 2005. № 2. 124–127.
14. *Cooley J.W., Tukey J.W.* An algorithm for the machine calculation of complex Fourier series // Mathematics of Computation. 1965. **19**, № 90. 297–301.
15. *Выдрин Д.Ф., Абзалилова Ю.Р., Вдовин А.К.* Быстрое преобразование Фурье в цифровой обработке сигналов // Теория и практика современной науки. 2017. № 2. 161–163.
16. *Тимошенко Л.И.* Дискретное преобразование Фурье и его быстрые алгоритмы // Современные наукоемкие технологии. 2014. № 12-2. 188–193.
17. *Солонина А.И., Улахович Д.А., Яковлев Л.А.* Алгоритмы и процессоры цифровой обработки сигналов. СПб.: БХВ-Петербург, 2001.
18. *Lerner B.* Writing efficient floating-point FFTs for ADSP-TS201 TigerSHARC® processors, 2004. https://www.analog.com/media/en/technical-documentation/application-notes/EE-218.pdf.
19. *Логинов В.Е., Ишин П.А.* Оптимизация для архитектуры "Эльбрус" быстрого преобразования Фурье применительно к 32-разрядным числам с плавающей точкой // Вопросы радиоэлектроники. 2012. **4**, № 3. 108–118.

Поступила в редакцию
11 января 2021 г.

Принята к публикации
4 марта 2021 г.

Информация об авторе

Олег Васильевич Осипов — к.ф.-м.н., доцент, Белгородский государственный технологический университет им. В. Г. Шухова, кафедра программного обеспечения вычислительной техники и автоматизированных систем, ул. Костюкова, 46, 308012, Белгород, Российская Федерация.

References

1. E. A. Al'tman, "Optimization of Computational Scheme of Fast Fourier Transformation," Omsk. Nauch. Vestn., No. 1 (64), 149–151 (2008).
2. L. M. Gol'denberg, B. D. Matyushkin, and M. N. Polyak, *Digital Processing of Signals* (Radio Svyaz, Moscow, 1990) [in Russian].
3. L. R. Rabiner, B. Gold, and C. K. Yuen, *Theory and Application of Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, 1975; Mir, Moscow, 1978).
4. Fast Fourier Transform for $O(N \log N)$. https://e-maxx.ru/algo/fft_multiply. Cited May 15, 2021.
5. O. V. Osipov, "Spectral Analysis of Discrete Signals with High Frequency Resolution," Vychisl. Metody Programm. **20**, 270–282 (2019).
6. N. V. Ponomareva, "Fast Parametric Fourier Transform for Spectral Analysis of Signals with High Resolution in a Given Frequency Range," DSPA: Voprosy primeneniya cifrovoj obrabotki signalov **9** (1), 28–32 (2019).
7. V. N. Chan and V. N. Dam, "Implementation of the Spectrum Analyzer with High-Resolution on FGPA," DSPA: Voprosy primeneniya cifrovoj obrabotki signalov **6** (4), 725–729 (2016).
8. M. Gasior and J. L. Gonzalez, "Improving FFT Frequency Measurement Resolution by Parabolic and Gaussian Spectrum Interpolation," AIP Conf. Proc. **732** (1), 276–285 (2004).
9. I. A. Sidorenko and P. A. Kuskova, "On Improving the Accuracy of the Spectral Analysis of Phonemes when Using Sound Editors," Belgorod State Univ. Sci. Bull. Ser.: Econ. Inform., No. 7, 188–193 (2015).
10. D. D. Poleshenkov and O. O. Basov, "Pitch Frequency Separation Method Based on a Frequency Demodulation," Belgorod State Univ. Sci. Bull. Ser.: Econ. Inform. **46** (2), 359–366 (2019).
11. B. Gold and C. M. Rader, *Digital Signal Processing* (McGraw-Hill, New York, 1969; Sov. Radio, Moscow, 1973).
12. V. I. Belov and E. I. Panimaskin, "The Fast Fourier Transform (FFT) with Decimation in Time," https://docplayer.ru/25793573-Belov-v-i-panimaskin-e-i-bystroe-preobrazovanie-fure-bpf-s-prorezhivaniem-po-vremeni.html. Cited May 15, 2021.
13. N. A. Galanina, "Synthesis of FFT Functional Modules in RNS," Vestn. Chuvash. Gos. Univ., No. 2, 124–127 (2005).
14. J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," Math. Comput. **19** (90), 297–301 (1965).
15. D. F. Vydrin, Yu. R. Abzalilova, and A. K. Vdovin, "Fast Fourier Transformation at Digital Signal Processing," Teor. Prakt. Sovremen. Nauki, No. 2, 161–163 (2017).
16. L. I. Timoshenko, "Discrete Transformation of Fourier and His Fast Algorithms," Sovremen. Naukoemk. Tekhnol., No. 12-2, 188–193 (2014).

2021, **22** (2), 123–137. DOI: 10.26089/NumMet.v22r209

17. A. I. Solonina, D. A. Ulakhovich, and L. A. Yakovlev, *Algorithms and Processors for Digital Signal Processing* (BHV-Petersburg, St. Petersburg, 2001) [in Russian].
18. B. Lerner, *Writing Efficient Floating-Point FFTs for ADSP-TS201 TigerSHARC® Processors.* https://www.analog.com/media/en/technical-documentation/application-notes/EE-218.pdf. Cited May 15, 2021.
19. V. E. Loginov and P. A. Ishin, "32-bit Floating-Point Fast Fourier Transform Optimization for Elbrus Processor," Voprosy Radioelektron. **4** (3), 108–118 (2012).

Received
January 11, 2021

Accepted for publication
March 4, 2021

Information about the author

Oleg V. Osipov — Ph.D., Associate Professor, Shukhov Belgorod State Technological University, ulitsa Kostyukova, 46, 308012, Belgorod, Russia.