ПРОЦЕДУРЫ БЫСТРОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ И УОЛША

В.Д.Гусев

Появление алгоритма быстрого преобразования Фурье [I] привело к переоценке буквально всех подходов, связанных с численной реализацией идей и методов спектрального анализа и смежных областей (корреляционного анализа, теории фильтрации, поиска периодичностей и т.д.).

Алгоритм предназначен для эффективного вычисления дискретного преобразования Фурье в прямой

$$X(j) = \sum_{k=0}^{N-1} A(k) \exp(-2\pi i k j/N) \qquad (I)$$

и обратной форме

$$A(k) = \frac{1}{N} \sum_{j=0}^{N-1} X(j) \exp(2\pi i k j/N), \qquad (2)$$

где $A(k)$ – последовательность дискретных отсчетов исходной функции (действительной или комплексной) для $k = 0,I,2,\ldots, N-I$,

$X(j)$ – соответствующая ей последовательность спектральных отсчетов, определенная для $j = 0,I,2,..,N-I$,

i – мнимая единица,

N – число отсчетов исходной функции.

Эффективность алгоритма определяется максимально возможной степенью расщепления числа N на простые сомножители:

$$N = r_1 \times r_2 \times \ldots \times r_m^{*)}.$$

Рекуррентная схема счета коэффициентов преобразования, осуществление которой становится возможным лишь благодаря такому расщеплению, позволяет вместо N^2 комплексных операций типа "умножение плюс сложение", требующихся для выполнения преобразований (I), (2) при прямом методе счета, затратить всего лишь

*) Иногда целесообразно проводить разбиение и не на простые сомножители (см., например, [2]).

$N\sum_{i=1}^{m} r_i$ операций.*)

Таким образом, выигрыш, получаемый за счет использования алгоритма быстрого преобразования Фурье (БПФ), выражается отношением $N^2/N\sum_{i=1}^{m} r_i - N/\sum_{i=1}^{m} r_i$, которое тем больше, чем больше N и выше степень расщепления. Показано [I], что максимальный выигрыш получается при равных сомножителях $r_1 = r_2 = \ldots r_m = r$. Сравнение эффективности алгоритмов по r $(2 \leq r < 10)$ показало, что наибольший эффект дает выбор $N = 3^m$. Случай $N = 2^m$ незначительно уступает этому случаю по эффективности, зато дает определенные преимущества при программировании.

Подавляющее большинство алгоритмов БПФ написано для случая $N = 2^m$. Это обусловлено следующими соображениями. Во-первых, при произвольном N (если считать все значения N равновероятными) уменьшается вероятность выбора N, расщепляющегося на большое число сомножителей и тем самым уменьшается потенциальная эффективность алгоритма. Во-вторых, с точки зрения программирования алгоритм становится значительно более громоздким, что в конечном счете ведёт к увеличению времени его работы. И, наконец, случай произвольного N всегда можно свести к случаю $N^* = 2^m$, где N^* – ближайшая, превосходящая N целая степень двойки, добавлением $N^* - N$ нулей к правому концу анализируемой дискретной функции. Как известно, спектр такой функции, дополненной нулями, будет иметь ту же огибающую, что и спектр исходной функции, но с более часто расположенными отсчетами.

С точки зрения эффективности алгоритма такая процедура является оправданной в большинстве случаев. К примеру, вычисление преобразования Фурье при помощи алгоритма, ориентированного на произвольное значение N, для дискретной функции, содержащей $N = 59$ отсчетов, потребует даже большего времени, чем использование прямого метода счета. В то же время дополнение этой функции нулями до $N^* = 64$ и использование алгоритма

*) Для случая $r_1 = r_2 = \ldots r_m = 2$ эта оценка сильно завышена (примерно в 3 раза). Алгоритмы, предложенные в [6], требуют для своей реализации не более чем $m \times [N/2]$ комплексных операций типа "умножение+сложение" плюс $N/2$ комплексных операций типа "сложение"]. Все сравнения в дальнейшем, однако, проводятся для общего (наименее выигрышного) случая.

БПФ для случая $N = 2^m$ даст выигрыш более чем в 5 раз по сравнению с прямыми методами счета. Более того, дополнение функции нулями иногда является целесообразным и из некоторых физических соображений.

В настоящее время появилось множество модификаций алгоритма БПФ и сфера его применимости расширена на некоторые другие преобразования (Хаара, Уолша [3] и др.).

Из модификаций алгоритма следует отметить следующие две. Первая из них связана с некоторой переоценкой эффективности алгоритма при равных сомножителях по сравнению с начальной оценкой, полученной Кули и Тьюки [I]. Показано [2], что в случае $r = 4, 8, 16$ можно изыскать дополнительные возможности для повышения эффективности алгоритма по сравнению со случаем $r = 2, 3$. Максимальный выигрыш, получаемый с помощью 8- и 16-точечных схем преобразования, составляет примерно полтора раза по сравнению с обычной двухточечной схемой преобразования, однако сам алгоритм при этом несколько усложняется.

Вторая модификация реализует идею, высказанную Кули и Тьюки относительно преобразования действительных чисел. Массив исходных данных длиной $N = 2^m$ делится при этом на два подмассива длины $N/2$, каждый из которых рассматривается как действительная и мнимая части комплексного массива длины $N/2$. Вместо преобразования Фурье исходного действительного массива длины N осуществляется преобразование комплексного массива длины $N/2$, за счет чего время преобразования сокращается примерно в 2 раза. По спектру полученной функции потом легко восстанавливается спектр исходной функции.

В настоящей работе реализована двухточечная схема БПФ для $N = 2^m$, позволяющая получить как прямое, так и обратное преобразование Фурье, а также вторая из упоминавшихся выше модификаций, позволяющая получить преобразование последовательности действительных чисел. На основании первой из этих процедур реализовано также быстрое преобразование Уолша (БПУ), которое в соответствии с [3] определяется следующим образом:

$$F(m) = \sum_{n=0}^{N-1} f(n)\, wal(m, n) \qquad (3)$$

$$m = 0, 1, 2, \ldots N-1,$$

где функции Уолша $wal(m, n)$ в свою очередь определяются рекуррентным соотношением

$$wal(m,n) = wal([m/2], 2n)\cdot wal(m-2[m/2], n) \quad (4)$$

при начальных условиях

$$wal(0,n) = 1 \quad \text{для} \quad n = 0,1,2,\ldots, N-1$$

$$wal(1,n) = \begin{cases} 1 & \text{для } n = 0,1,2,\ldots,(N/2)-1 \\ -1 & \text{для } n = N/2, N/2+1,\ldots, N-1. \end{cases} \quad (5)$$

Соответственно этому обратное преобразование будет иметь вид

$$f(n) = \frac{1}{N}\sum_{m=0}^{N-1} F(m)\quad wal(n,m) \quad (6)$$

$$n = 0, 1, 2, \ldots, N-1.$$

Поскольку при этом как сама функция f, так и её преобразование F являются действительными и функции Уолша могут принимать значения только +I и -I, то комплексные операции заменяются на действительные, а операции умножения на операции сложения и вычитания, за счет чего достигается значительная экономия по времени по сравнению с БПФ.

Описание процедур БПФ и БПУ

Процедуры для реализации БПФ и БПУ написаны на "Входном языке", созданном в Вычислительном центре СО АН СССР (4,5) и являющемся расширением языка "Алгол-60". Наиболее существенным и принципиальным моментом в выборе языка для написания процедур явилась возможность использовать описание типа "комплексный", имеющаяся во "Входном языке", что делает сами процедуры гораздо более компактными и обозримыми. Процедуры ориентированы на ЭВМ М-220 и БЭСМ-6. Ниже приводятся описания процедур.

I. Процедура БПФ2 реализует двухточечную схему БПФ для случая $N = 2^m$. Процедура позволяет выполнить как прямое, так и обратное преобразование Фурье и не содержит операций в кодах, т.е. пригодна для реализации как на М-220, так и на БЭСМ-6.

Обращение к процедуре имеет вид: БПФ2 (X, T, Tx, Ty, m, ключ). Здесь X [0: N-I] - массив исходных данных, имеющий описание типа "комплексный". Результат преобразования помещается в этот же массив, т.е. исходная информация не сохраняется; T[0: $N/2$-I] - комплексный массив для хранения значений тригонометрических функций. Для работы процедуры необходимо ввести также описание действительной и мнимой частей компонент массива T, например : T [к] = Tx [к] + i Ty [к]. Описания действительной (Tx) и мнимой (Ty) частей массива T входят в число параметров процедуры;

m - степень двойки в представлении числа N ($N = 2^m$);

ключ - параметр, определяющий вид преобразования и имеющий описание типа "логический". Значению "истина" этого параметра соответствует прямое преобразование Фурье, а значению "ложь" - обратное.

Наиболее часто в практических задачах встречается ситуация, когда требуется получать спектральное представление для последовательности функций одинаковой длины. Тригонометрический массив при этом либо сохраняется, либо заменяется на сопряженный. Если повторное обращение к процедуре происходит с тем же самым значением m, то для счета используются значения тригонометрических функций, полученные (или уже имевшиеся) при предыдущем обращении. При этом о сохранении массива T в промежутке между двумя обращениями должен позаботиться сам программист.

2. Процедура БПУ 2 реализует быстрое преобразование Уолша для дискретной функции, содержащей $N = 2^m$ отсчетов. Процедура написана на основе процедуры БПФ2 и также пригодна для реализации как на М-220, так и на БЭСМ-6.

Обращение к процедуре имеет вид: БПУ2 (X, m, ключ). Здесь X[0: N-I]- массив исходных данных, имеющий тип "вещественный" (он же является массивом результата);

"m" и "ключ" - параметры, имеющие тот же смысл, что и соответствующие параметры в процедуре БПФ 2.

С целью избежать привязки к конкретной машине требуемая алгоритмом перетасовка исходных данных в процедуре БПУ2, также как и в процедуре БПФ2 осуществляется не в кодах, хотя это было бы наиболее естественно. Для этого внутри процедур заказывается дополнительная память (3 m ячеек). Максимально возможное значение m принимается равным I4.

3. Процедура БПФ2 Д осуществляет быстрое преобразование Фурье массива действительных чисел длиной $N = 2^m$. Обращение к процедуре имеет вид: БПФ2Д (X, T, Tx, Ty, m). Здесь X [0: $N/2$-I] - комплексный массив, который в 2 раза короче исходного массива действительных чисел. Перед обращением к процедуре в действительную часть этого массива помещаются все чётные

члены исходной последовательности ($K = 0,2,4,\ldots,N-2$), а в мнимую – все нечетные ($K = 1,3,5,\ldots,N-1$).

Результат преобразования помещается также в массив X. При этом нам достаточно знать лишь половину (точнее $N/2+1$) спектральных составляющих исходного массива, поскольку вторая половина может быть найдена из соотношения

$$A_{N-K} = A^*_K \quad , \qquad K = 1,2,\ldots, N/2-1 ,$$

где значок (*) означает комплексно-сопряженную величину. Поскольку A_0 и $A_{N/2}$ являются действительными числами, то для их хранения отводится одна (нулевая) компонента массива X. При этом постоянная составляющая A_0 представляет действительную часть нулевой компоненты массива X, а составляющая $A_{N/2}$ – её мнимую часть;

T [0: $N/4$ – 1] – комплексный массив для хранения значений тригонометрических функций, Tx и Ty соответственно – описания его действительной и мнимой частей. Как и в случае процедуры БПФ2, о сохранении массива T в промежутке между двумя последовательными обращениями к процедуре с одинаковым значением m должен позаботиться сам программист.

Восстановление спектра исходной последовательности по спектрам четной и нечетной подпоследовательностей объединено в процедуре с последним шагом быстрого преобразования Фурье. Если обозначить через λ_K ($K = 0,1,2,\ldots, N/2 - 1$) спектральные составляющие комплексной последовательности, у которой действительные части являются четными членами исходной последовательности, а мнимые – нечетными, то спектральные составляющие A_K исходной последовательности будут выражаться через λ_K следующим образом:

$$A_0 = Re\,\lambda_0 + Im\,\lambda_0 ,$$

$$A_K = \frac{1}{2}\left[(\lambda_K + \lambda^*_{N/2-K}) + W^{K+N/4}(\lambda_K - \lambda^*_{N/2-K})\right],$$

$$K = 1,2,\ldots, N/4-1; \quad W = \exp\left(-i\,\frac{2\pi}{N}\right). \qquad (7)$$

$$A_{N/4} = \lambda^*_{N/4} ,$$

$$A_{N/2-K} = \frac{1}{2}\left[(\lambda_K + \lambda^*_{N/2-K}) - W^{K+N/4}(\lambda_K - \lambda^*_{N/2-K})\right]^*,$$

$$A_{N/2} = Re\,\lambda_0 - Im\,\lambda_0 .$$

Ниже помещены α – схемы процедур БПФ2, БПУ2 и БПФ2Д.

Оценки процедур по памяти и быстродействию

Сравнение процедур по быстродействию проводилось на массиве исходных данных фиксированной длины ($N = 128$ точек). С изменением N абсолютное время обработки будет естественно меняться, но в одинаковой пропорции для всех процедур.

Относительно выбора самого N можно сказать следующее. Тот факт, что выигрыш в связи с использованием БПФ вместо обычного преобразования Фурье растет с увеличением N, создает иногда иллюзию того, что, чем большее N мы возьмем, тем больший выигрыш по времени мы получим. Иными словами, на вопрос, что выгоднее, обработать K сигналов по N точек или один сигнал с $K \times N$ точками, дается ответ – один сигнал с $K \times N$ точками. Применительно к речевому сигналу, скажем, это может привести к выводу, что выгоднее получать спектральную картинку для всего слова в целом, нежели совокупность спектральных картинок фонем, составляющих данное слово, или выгоднее получать спектральную картинку всей фонемы, нежели совокупность спектральных картинок сегментов, составляющих фонему, и т.д.

Ошибочность такого заключения становится очевидной, если учесть, что с увеличением N растет не только выигрыш, даваемый алгоритмами БПФ по сравнению с алгоритмами обычного ПФ, но и число самих операций, требующихся для реализации преобразования. Так в случае обычного преобразования Фурье для обработки одного сигнала с $K \times N$ точками требуется K^2N^2 комплексных операций типа "умножение + сложение", а для обработки K сигналов по N точек в каждом – лишь KN^2 операций. Таким образом, уже в случае обычного ПФ при обработке длинного массива информации его выгодно разбивать на куски такой минимальной длины, которую ещё допускает физика задачи.

Применение БПФ несколько сглаживает этот эффект, но не

устраняет его, поскольку выигрыш за счет увеличения N, даваемый алгоритмом Кули-Тьюки, растет медленнее, чем проигрыш за счет увеличения числа операций. Действительно, положив K в вышеприведенном примере равным 2^n, а N равным 2^m, получим, что для обработки одного сигнала с $K \times N$ точками потребуется $2^{n+m} \cdot (n+m) \times 2$ комплексных операций типа "умножение + сложение", а для обработки K сигналов по N точек $2^n \cdot 2^m \cdot m \cdot 2$ операций. Иными словами, затраты в I-м случае относятся к затратам во 2-м случае как $\frac{n+m}{m}$, т.е. проигрыш имеется, хотя и менее явный.

С этой точки зрения следует весьма критически относиться к утверждениям типа "использование БПФ для N = I024 точек дает выигрыш более чем в 50 раз по сравнению с прямым методом счета", имеющимся в большинстве публикаций по алгоритмам БПФ. Дело действительно обстоит так (а при большем N ещё более благополучно), если для анализа принципиально необходим участок в I024 точки. Однако во многих задачах нужная информация может извлекаться и из участков меньшей длины. С этой точки зрения исследователь должен был разбить свой сигнал на участки минимально допустимой длины, скажем, на 4 участка по 256 точек или на 8 участков по I28 точек.

Такое разбиение было бы экономически наиболее целесообразным в случае использования обычного ПФ. Именно на участках такой минимально допустимой условиями задачи длины и следует оценивать выигрыш, даваемый БПФ по сравнению с обычным ПФ. В нашем случае этот выигрыш составит I6 раз для N = 256 и 9 раз для N = I28 точек.

И, наконец, использование больших значений N нецелесообразно ввиду потери точности, что характерно для всех рекурентных методов счета. Это может оказаться принципиальным в некоторых задачах, особенно при невысокой точности представления числа в машине.

Как уже говорилось выше, сравнение процедур по быстродействию проводилось на реализациях длиной N = I28 точек. Среднее время работы процедуры получалось усредненным по 200 реализациям. Время работы процедуры при прямом преобразовании получалось несколько меньшим, чем при обратном за счет отсутствия деления на N (см. формулы 2,6). Для сравнения с процедурами БПФ была написана также процедура обычного преобразования Фурье (ПФ). Результаты сравнения по памяти и быстродействию приводятся в нижеследующей таблице.

Таблица

Сравнение процедур БПФ и БПУ
по памяти и быстродействию

Наименование процедуры	Время обработки I реализации в секундах (N = I28)		Объем памяти, занимаемый процедурой (число ячеек МОЗУ - десятичное)	
	М-220	БЭСМ-6	М-220	БЭСМ-6
ПФ	I8	I,5	202	226
БПФ-2	0,68 -прямое 0,72 - обратное	$0{,}53\ I0^{-I}$ - прямое $0{,}57 \cdot I0^{-I}$ - обратное	336	356
БПФ2Д	0,39	$0{,}3I\ I0^{-I}$	505	509
БПУ2	0,28 -прямое 0,34- обратное	$0{,}23\ I0^{-I}$ - прямое $0{,}26\ I0^{-I}$ - обратное	I45	I50

По поводу приведенной таблицы можно заметить следующее. Объем памяти, занимаемый соответствующими процедурами на обеих машинах, примерно одинаков. Это объясняется тем, что при одноадресной системе команд в машине БЭСМ-6 в одной ячейке памяти помещаются 2 команды, а в М-220 - одна команда, но зато трехадресная.

Далее, следует ещё раз обратить внимание на то, что процедуры написаны без использования кодов с целью избежать привязки к конкретной машине. Это ведет к некоторым неудобствам при осуществлении перетасовки исходных данных, требующейся в соответствии с алгоритмом Кули-Тьюки. Наиболее эффективно такую перетасовку можно осуществить в кодах. Особенно существенно это сказывается при реализации процедуры БПУ-2, где время перетасовки составляет примерно четыре десятых от времени работы всей процедуры.

И, наконец, время работы всех процедур можно несколько

уменьшить за счет использования отдельного массива памяти для результатов анализа. В приведенных процедурах с целью экономии памяти результаты анализа засылаются в исходный массив памяти.

Литература

I. Cooley I.W, Tukey I.W. An Algorithm for the Machine Calculation of Complex Fourier Series. Mathematics of Computation, v.19, N 90, April, 1965.

2. Bergland G.D. A Fast Fourier Transform Algorithm Using Base 8 Iterations. Mathematics of Computation, v. 22, N 102, April 1968.

3. Shanks I.L. Computation of the Fast Walsh-Fourier Transform IEEE Trans. on Computers, vol. C-18, N 5, May, 1969.

4. Ершов А.П., Кожухин Г.Н., Волошин Ю.М. Входной язык для систем автоматического программирования. Новосибирск, 1964.

5. Ершов А.П., Кожухин Г.И., Поттосин И.В. Руководство к пользованию системой Альфа. Новосибирск, "Наука", 1968.

6. Cochran W.T., Cooley I.W. et al. What is the Fast Fourier Transform? Proceedings of the IEEE, vol.55, N 10, 1967.


```
ПРОЦЕДУРА ~Б-П~*2(~Х,~Т,~ТХ,~ТУ,М,КЛЮЧ); НАЧАЛО ЦЕЛЫЙ I,J,K,L,P,P1,~N;
СОБСТ ЦЕЛЫЙ *АЛЬФА*; *АЛЬФА*:=0; СОБСТ ЛОГИЧ *БЗТА*; СОБСТ ВЕЩ *ПИ*2; *ПИ*2:=6.28318530
718;
ВЕЩ **И*; КОМПЛ Z; Z:=X*1У; ЦЕЛЫЙ МАССИВ ~А,~В,~С[1:14];
~N:=2↑M; P1:=~N/4; **И*:=*ПИ*2/~N; X:=COS(**И*); У:=SIN(**И*);
ЕСЛИ *АЛЬФА*=~N ТО НАЧАЛО ~ТХ[0]:=1; ~ТУ[0]:=0; ~ТХ[P1]:=0; ~ТУ[P1]:=1; *АЛЬФА*:=~N; ЕСЛИ
КЛЮЧ ТО НАЧАЛО У:=-У; ~ТУ[P1]:=-1 КОНЕЦ ;
ДЛЯ К:=1,....,P1-1 ЦИКЛ НАЧАЛО ~Т[К]:=~Т[К-1]*Z; ~ТХ[P1*И]:=~ТХ[К]; ~ТУ[P1*И]:=~ТУ[К];
~ТУ[P1+К]:=-~ТХ[К] КОНЕЦ ; НА ТАС КОНЕЦ ИНАЧЕ ЕСЛИ *БЗТА*АНДКЛЮЧ ОТ *БЗТА*АНДNOТКЛЮЧ
ТО НА ТАС ИНАЧЕ НАЧАЛО ДЛЯ К:=1,....,2*P1-1 ЦИКЛ ~ТУ[К]:=-~ТУ[К] КОНЕЦ ; ТАС:=БЗТА:=КЛЮ
Ч; **И*:=~N; L:=1; ДЛЯ P:=1,....,M ЦИКЛ НАЧАЛО P1:=~А[P]:=**И*/2;
ДЛЯ К:=1,....,P ЦИКЛ ~С[К]:=P1; ЕСЛИ L<P1 ТО НАЧАЛО Z:=~Х[P1]; ~Х[P1]:=
~Х[L]; ~Х[L]:=Z КОНЕЦ ; L:=L*1; ~В[P]:=1; ПОВТ: ДЛЯ К:=1,....,P ЦИКЛ ЕСЛИ ~В[К]=0 ТО НАЧАЛО
P1:=~С[К]:=~С[К]-~А[К]; ЕСЛИ L<P1 ТО НАЧАЛО Z:=~Х[P1];
~Х[P1]:=~Х[L]; ~Х[L]:=Z КОНЕЦ ; L:=L+1; ~В[К]:=~В[К]+1; ДЛЯ I:=1,....,К-1 ЦИКЛ
~С[I]:=P1; НА ПОВТ КОНЕЦ ИНАЧЕ ~В[К]:=0; **И*:=**И*-~А[P] КОНЕЦ ; ДЛЯ К:=6 ШАГ 2 ДО ~N-2 ЦИКЛ
НАЧАЛО Z:=~Х[К]*~Х[К+1]; ~Х[К+1]:=~Х[К]-~Х[К+1]; ~Х[К]:=Z КОНЕЦ ; **И*:=1; M-1 РАЗ ЦИКЛ НАЧАЛО
К:=**И*:=**И*/2; J:=~N/P1; ДЛЯ I:=0
ШАГ P1 ДО ~N-P1 ЦИКЛ НАЧАЛО L:=0; ДЛЯ P:=I+К,....,I+К+К-1 ЦИКЛ НАЧАЛО Z:=
ЕСЛИ L=П ТО ~Х[P] ИНАЧЕ ~Т[L]:=~Х[P]; ~Х[P]:=~Х[P-К]-Z; ~Х[P-К]:=
~Х[P-К]+Z; L:=L+J КОНЕЦ КОНЕЦ ; ЕСЛИ НОТКЛЮЧ ТО НАЧАЛО **И*:=1/~N; ~Х[I]:=~Х[I]**И*
КОНЕЦ КОНЕЦ ;
```

```
ПРОЦЕДУРА "Б-П"У2("Х,М,КЛЮЧ); НАЧАЛО ЦЕЛЫЙ I,K,L,P,P1,"N; ВЕЩ Z,T;
ЦЕЛЫЙ МАССИВ "A,"B,"C[1:14]; T:="N:=2↑M; L:=1; ДЛЯ P:=1,...,M ЦИКЛ НАЧАЛО P1:="A[P]:=T/
2; ДЛЯ K:=1,...,P ЦИКЛ "C[K]:=P1; ЕСЛИ L<P1 ТО
НАЧАЛО Z:="X[P1]; "X[P1]:="X[L]; "X[L]:=Z КОНЕЦ ; L:=L+1; "B[P]:=1;
ПОВТ: ДЛЯ K:=1,...,P ЦИКЛ ЕСЛИ "B[K]=0 ТО НАЧАЛО P1:="C[K]:="C[K]+"A[K];
ЕСЛИ L<P1 ТО НАЧАЛО Z:="X[P1]; "X[P1]:="X[L]; "X[L]:=Z КОНЕЦ ; L:=L+1; "B[K]:="B[K]
+1; ДЛЯ I:=1,...,K-1 ЦИКЛ "C[I]:=P1; НА ПОВТ КОНЕЦ ИНАЧЕ "B[K]:=0; T:="A[P] КОНЕЦ ;
Z:=.5; M РАЗ ЦИКЛ НАЧАЛО L:=Z:=Z×2; P1:=Z×2; ДЛЯ I:=0 ШАГ P1
ДО "N-P1 ЦИКЛ ДЛЯ P:=I+L,...,I+L+L-1 ЦИКЛ НАЧАЛО T:="X[P]; "X[P]:=
"X[P-L]-T; "X[P-L]:="X[P-L]+T КОНЕЦ КОНЕЦ ; Z:=1/"N; ЕСЛИ NOТКЛЮЧ ТО "X[]:="X[]×Z КОНЕЦ ;
```

```
ПРОЦЕДУРА "Б-П"Ф2"Д("Х,"Т,"Тх,"ТУ,М); НАЧАЛО ЦЕЛЫЙ I,J,K,L,P,P1,"N;
СОБСТ ЦЕЛЫЙ ∘АЛЬФА∘; ∘АЛЬФА∘:=0; СОБСТ ВЕЩ ∘ПИ∘2; ∘ПИ∘2=6.28318530718; ВЕЩ ∘ФИ∘,∘ПСИ∘;
КОМПЛ Z,S,T; Z=X∘IУ; ЦЕЛЫЙ МАССИВ "A,"B,"C[1:14];
"N:=2↑(M-1); P1:="N/4; ∘ФИ∘:=∘ПИ∘2/"N; X:=COS(∘ФИ∘); У:=-SIN(∘ФИ∘);
"Тх[0]:=1; "Ту[0]:=0; "Тх[P1]:=0; "ТУ[P1]:=-1; ЕСЛИ ∘АЛЬФА∘≠"N ТО НАЧАЛО ∘АЛЬФА∘:="N; ДЛЯ
K:=1,...,P1-1 ЦИКЛ НАЧАЛО "Т[K]:="Т[K-1]×Z;
"Тх[P1+K]:="Ту[K]; "ТУ[P1+K]:=-"Тх[K] КОНЕЦ КОНЕЦ ;
∘ПСИ∘:="N; L:=1; ДЛЯ P:=1,...,M-1 ЦИКЛ НАЧАЛО P1:="A[P]:=∘ПСИ∘/2; ДЛЯ K:=1,...,P ЦИКЛ "С
[K]:=P1; ЕСЛИ L<P1 ТО НАЧАЛО Z:="X[P1]; "X[P1]:="X[L]; "X[L]:=Z КОНЕЦ ;
L:=L+1; "B[P]:=1; ПОВТ: ДЛЯ K:=1,...,P ЦИКЛ ЕСЛИ "B[K]=0 ТО НАЧАЛО P1:="C[K]+"A[K]; ЕСЛИ
L<P1 ТО НАЧАЛО Z:="X[P1]; "X[P1]:="X[L]; "X[L]:=Z КОНЕЦ ;
L:=L+1; "B[K]:="B[K]+1; ДЛЯ I:=1,...,K-1 ЦИКЛ "C[I]:=P1; НА ПОВТ КОНЕЦ
ИНАЧЕ "B[K]:=0; ∘ПСИ∘:="A[P] КОНЕЦ ;
ДЛЯ K:=0 ШАГ 2 ДО "N-2 ЦИКЛ НАЧАЛО Z:="X[K]+"X[K+1]; "X[K+1]:="X[K]-"X[K+1]; "X[K]:=Z КОНЕЦ
; ∘ПСИ∘:=1; M-2 РАЗ ЦИКЛ НАЧАЛО K:=∘ПСИ∘; ∘ПСИ∘=2; P1:=∘ПСИ∘=2; J:="N/P1; ДЛЯ I:=0
ШАГ P1 ДО "N-P1 ЦИКЛ НАЧАЛО L:=0; ДЛЯ P:=I+K,...,I+K+K-1 ЦИКЛ НАЧАЛО Z:=
ЕСЛИ L=0 ТО "X[P] ИНАЧЕ "Т[L]="X[P]; "X[P]:="X[P-K]-Z; "X[P-K]:=
"X[P-K]+Z; L:=L+J КОНЕЦ КОНЕЦ КОНЕЦ ; X:="RE("X[0])+"IM("X[0]); У:="RE("X[0])-
"IM("X[0]); "X[0]:=Z; ∘ФИ∘:=∘ФИ∘/2; X:=COS(∘ФИ∘); У:=-SIN(∘ФИ∘); "X["N/2]:="CON("X("N/2));
I:="N/4 ДЛЯ K:=1 ШАГ 2 ДО "N/2-3 ЦИКЛ НАЧАЛО S:="X[K]+"CON("X["N-K]);
T:="Т[I]=Z×("X[K]-"CON("X["N-K])); "X[K]:=.5×(S+T); "X["N-K]:=.5×"CON(S-T);
P:=K+14-I:=I+1; S:="X[P]+"CON("X["N-P])); T:="Т[I]×("X[P]-"CON("X["N-P])));
"X[P]:=.5×(S+T); "X["N-P]:=.5×"CON(S-T) КОНЕЦ ; S:="X["N/2-1]+"CON("X["N/2+1]);
T:="Т[I]=Z×("X["N/2-1]-"CON("X["N/2+1]))); "X["N/2-1]:=.5×(S+T); "X["N/2+1]:=.5×"CON(S-T) КОНЕЦ;
```