весьма большое количество членов ряда в выражениях (4.1) или (4.2), что может оказаться неприемлемым с точки зрения эффективности алгоритма сжатия.

К другому классу алгоритмов сжатия данных по времени относятся так называемые адаптивные алгоритмы сжатия, отличающиеся тем, что обработка данных в блоке сжатия осуществляется в темпе их поступления, или, другими словами, в реальном масштабе времени. Адаптивные алгоритмы сжатия строятся таким образом, что в ходе их работы находятся приближения $f_j^*(t)$ вида (4.1), удовлетворяющие выбранному критерию верности и относящиеся к подынтервалу времени T_j, который входит в интервал времени T : $T_j \in T$, $j = 1, \ldots, k$. Количество подынтервалов T_j определяется как особенностями динамики процесса $f(t)$, так и количеством и видом функций $x_i(t)$. Кроме этого, определенное влияние оказывает признак, по которому осуществляется адаптация кусочной модели $f^*(t) = f_1^*(t) \cap f_2^*(t) \cap \ldots \cap f_k^*(t)$ к поступающим экспериментальным данным. Можно построить три следующих варианта построения адаптивных алгоритмов сжатия.

1. Фиксируется количество слагаемых в представлении (4.1) и автоматически подбирается максимальное значение интервала аппроксимации T_j, на котором выполняются требования по ограничению на выбранный показатель верности.

2. Фиксируется значение интервала аппроксимации T_j, на котором отыскивается аппроксимирующая функция вида (4.1) минимальной степени m, обеспечивающей выполнение требований по ограничению на выбранный показатель верности.

3. Интервал аппроксимации T_j и степень аппроксимирующей функции m выбираются автоматически.

Использование адаптивных алгоритмов сжатия предполагает кусочное представление контролируемого процесса и для его восстановления с заданной точностью вместо $2N$ измеренных значений (плюс время) на интервале времени $T(f(t_1), f(t_2), \ldots, f(t_N); t_1, t_2, \ldots, t_N \in T)$ достаточно заполнить значения коэффициентов $A_1^j, A_2^j, \ldots, A_{m_j}^j$, определенных на каждом подынтервале разбиения T_j, и значения моментов времени t_j, соответствующих началу этих подынтервалов. Коэффициент сжатия оказывается равным значению

$$K_{сж} = \frac{2N}{K + \sum_{j=1}^{K} m_j} .$$

Здесь K – количество подынтервалов разбиения, m_j – количество аппроксимирующих функций, используемых на подынтервале T_j.

На практике наибольшее применение нашло использование в качестве аппроксимирующего базиса функций следующего вида:

$$x_i(t) = t^{i-1} . \qquad (4.4)$$

На рис. 4.1 приводится иллюстрация процесса сжатия данных при использовании первого признака адаптации аппроксимирующей зависимости. В качестве используемого процесса рассматривается переходный процесс по изменению во времени тепловой мощности реактора РБМК-1000, при этом количество функций типа (4.4) жестко выбрано равным двум. В темпе поступления новых временных отсчетов происходит разбиение рассматриваемого интервала на подынтервалы T_j.

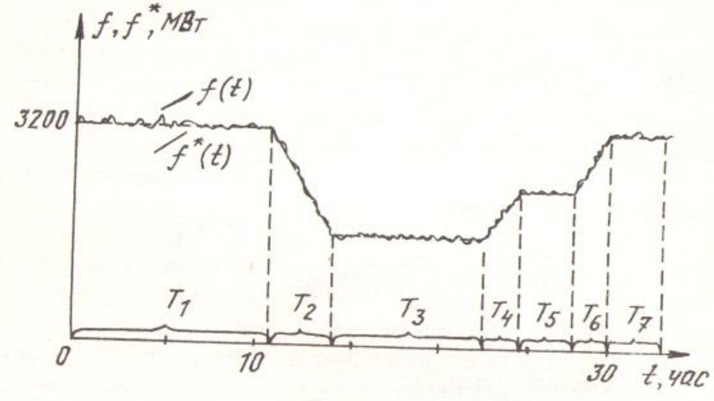

Рис. 4.1. Иллюстрация алгоритма сжатия с однопараметрической адаптацией по интервалу наблюдения

Применение адаптивных алгоритмов сжатия не ограничивается системами реального времени, поскольку можно программно имитировать процесс поступления данных в блок сжатия из промежуточного буфера их накопления. В настоящее время адаптивные алгоритмы сжатия получили широкое распространение в технике, поскольку при их использовании требуется минимальный объем памяти, необходимый для хранения промежуточных данных [8].

Выбор конкретных алгоритмов сжатия данных, предназначенных для обработки интересующих пользователя временных зависимостей, определяется следующими требованиями:

1. Алгоритмы сжатия и восстановления данных должны обеспечивать достаточное быстродействие программ с точки зрения получения функции $f^*(t)$. Иными словами, чем проще алгоритм расчета коэффициентов A_i в выражении (4.1) и чем проще собственно модель (4.1), тем лучше.

2. Алгоритм сжатия данных должен обеспечивать высокую эффективность с точки зрения получения как можно более высокого коэффициента сжатия, определяемого по формуле:

$$K_{сж} = \frac{n}{n^*} , \qquad (4.5)$$