DOI 10.37882/2223–2966.2022.05.13

МЕТОД ВЫПОЛНЕНИЯ АЛГОРИТМОВ ВИЗУАЛЬНОГО АНАЛИЗА ВИБРАЦИОННЫХ ДАННЫХ

METHOD FOR PERFORMING ALGORITHMS FOR VISUAL ANALYSIS OF VIBRATION DATA

V. Ilin

Summary. The article discusses a method for performing algorithms for visual analysis of vibration data, which allows optimizing the data processing process and increasing the efficiency of vibration diagnostics. The implementation of the method and the optimization approaches used are described. A comparative result of the work of this method and a standard script using the Python language is given, in which an algorithm for analyzing a filtered and unfiltered vibration signal is written by displaying the result of a fast Fourier transform.

Keywords: vibration data, visual programming language, python, analysis algorithms.

Ильин Виталий Алексеевич
Аспирант, МИРЭА — Российский технологический университет
vitaliy.a.ilin@gmail.com

Аннотация. В статье рассматривается метод выполнения алгоритмов визуального анализа вибрационных данных, который позволяет оптимизировать процесс обработки данных и повысить эффективность вибрационной диагностики. Описана реализация метода и используемые подходы оптимизации. Приводится сравнительный результат работы данного метода и стандартного скрипта с использованием языка Python, на котором написан алгоритма анализа фильтрованного и нефильтрованного вибрационного сигнала за счет вывода результата быстрого преобразования Фурье.

Ключевые слова: вибрационные данные, визуальный язык программирования, python, алгоритмы анализа.

Введение

Анализ вибрационных данных подразумевает частое изменение параметров алгоритмов и визуальное отображение полученных результатов. Популярными инструментами анализа являются пакет прикладных программ MATLAB, язык программирования Python с дополнительными библиотеками, а также специализированные программы [1]. Каждый из них имеет свои преимущества и недостатки, которые проявляются в определенных задачах анализа.

Специализированные программы имеют удобный интерфейс и оптимизированы для решения конкретных методов анализа вибрационных данных. Однако, это является и минусом, поскольку отсутствует возможность расширения функционала и добавление алгоритмов.

Написание скриптов на Python и MATLAB позволяет самостоятельно выбирать методы и алгоритмы обработки вибрационных данных. Но недостатком является то, что стандартные скрипты при запуске каждый раз выполняются полностью. При частом изменении параметров значительно снижается эффективность, поскольку обработка большого объема вибрационных данных производится продолжительное время. Оптимизация скриптов требует немалых навыков и ресурсов при разработке.

Одним из способов решения данной проблемы является разделение алгоритма на блоки, которые выполняют одну определенную функцию обработки данных. Такой подход позволяет обособить части кода и автоматически производить оптимизацию выполнения алгоритма анализа с помощью приведенного метода.

Реализация метода выполнения алгоритмов

Метод основан на двух базовых компонентах, с помощью которых строятся алгоритмы.

1. Блоки. Являются обособленным кодом, который выполняет одну функцию обработки данных. Имеет входные и выходные разъемы данных.
2. Связи. Обеспечивают передачу данных из выходных разъемов блоков во входные разъемы других блоков.

Данный подход напоминает парадигму потоко-ориентированного программирования и имеет те же преимущества, которые особенно проявляются при использовании этого метода совместно с визуальным языком

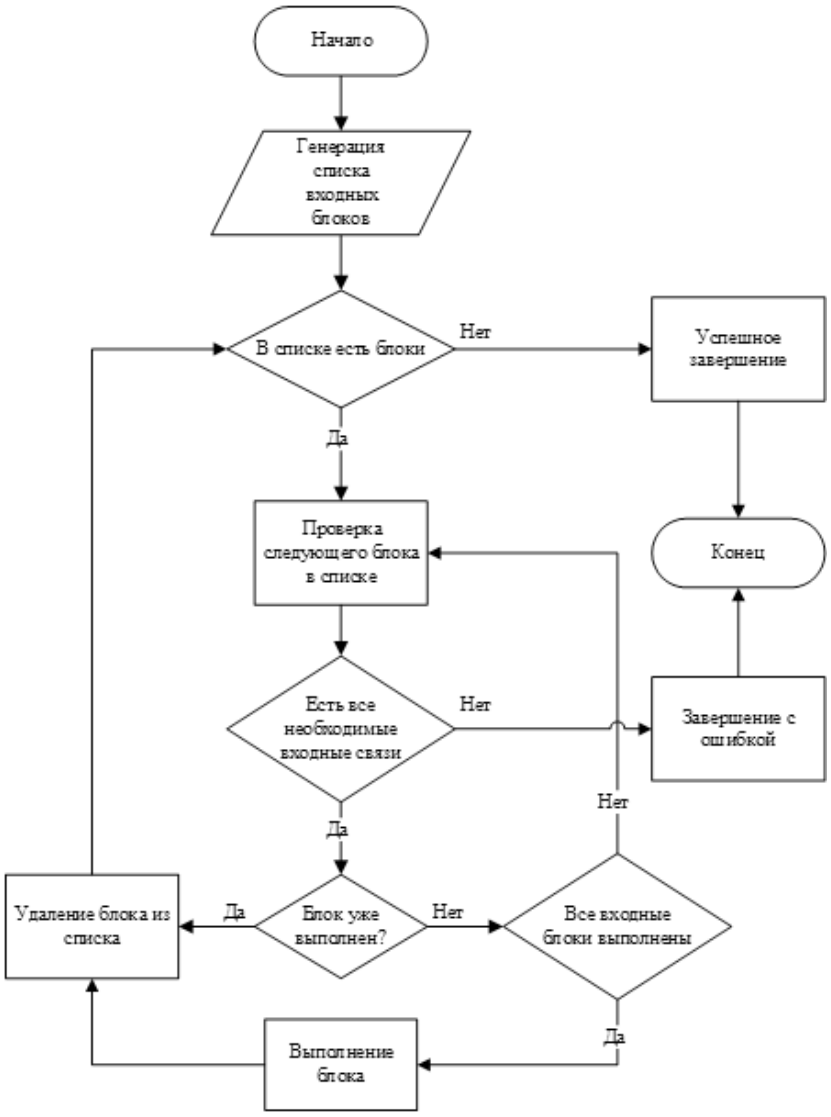

Рис. 1. Блок-схема выполнения блока

программирования. Однако есть и отличия: способ обмена данных между блоками и порядок их выполнения [2].

Блоки делятся на три типа:

1. Блоки входных данных. Обеспечивают поступление данных в алгоритм с помощью генерации или загрузки из внешних источников (файл, база данных и т.д.).
2. Блоки обработки данных. Производят вычислительные функции, необходимые для обработки данных для последующего анализа.
3. Блоки визуализации. Передают данные для различных компонентов визуализации, с помощью которых производится анализ данных.

Также блоки имеют следующие компоненты:

- Входные разъемы данных. Разъемы, которые соединяются с выходными разъемами других блоков и получают данные.
- Входные разъемы параметров. Разъемы, в которые поступают данные, позволяющие изменять параметры блока.
- Выходные разъемы данных. Разъемы, из которых поступают данные в другие блоки.
- Дополнительные выходные разъемы. Разъемы, из которых поступают промежуточные данные, получаемые в ходе выполнения блока, но не являющиеся основным результатом работы блока (к примеру, среднеквадратическое отклонение, получаемое при вычислении коэффициента эксцесса).

Таблица 1. Время выполнения алгоритмов при изменении параметров

Измененный параметр	Время выполнения стандартного алгоритма (сек.)	Время выполнения с использованием нового метода (сек.)
Фильтруемые даты	3.24	4.06
Тип фильтра	3.21	2.74
Фильтруемый диапазон частот	3.26	2.76
Параметры графика	3.22	1.65

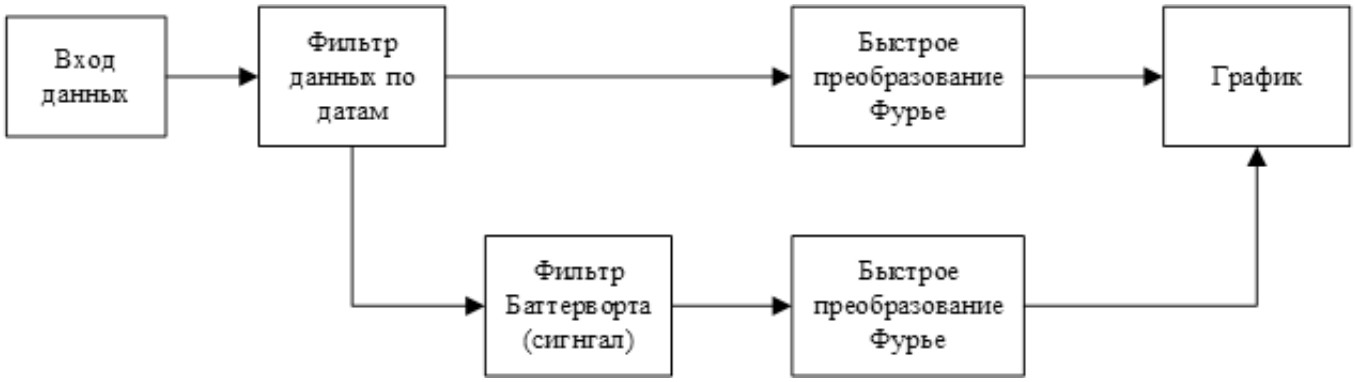

Рис. 2. Алгоритм обработки данных

- Параметры. Это переменные, которые влияют на выполнения блока и могут редактироваться извне (к примеру, порядок и частоты фильтра).
- Внутренние данные. Это данные, которые сохраняются между циклами выполнения алгоритма.

Оптимизация выполнения алгоритма

Одним из наиболее важных способов оптимизации для данного метода является мемоизация. В данном случае это подразумевает сохранение результатов обработки данных каждого блока, при выполнении которых используются следующие правила:

- если блок не выполнялся, то он выполняется, а результат его выполнения сохраняется;
- если блок выполнялся, то используется сохранённый результат.

При выполнении блоков используется подход с использованием хэш-таблиц, который является частью парадигмы программирования потоков данных. Отличием же является то, что список блоков строится только из тех, которые являются зависимостью блока, к данным которого производится запрос. Также используется пропагация алгоритма "Pull" из парадигмы реактивного программирования [3]. Выполнение блоков производится только тогда, когда происходит обращение к данным. Если имеются входные связи, то выполняемый блок обращается к подключенным к этим разъемам блокам.

При изменении параметров блоков используется инкрементное вычисление. То есть происходит выполнение не всего алгоритма, а только тех блоков, которые зависят от выходных данных изменяемого блока. Но поскольку метод в данной работе использует пропагацию Pull, то выполнятся должны только те блоки, к которым в момент изменения параметров происходит обращение. Для вызова блоков используется стандартная функция, приведенная на рисунке 1. Поэтому необходимо исключить те блоки, которые являются входными блоками для других выполняемых. Полученные для выполнения блоки можно представить с помощью множества:

$$A = \{a_i | 0 < i \leq n\} \quad (1)$$

$$OR(x) = \bigcup \{OR(y) | y \in O(x)\} \cup O(x) \quad (2)$$

$$IR(x) = \bigcup \{IR(y) | y \in I(x)\} \cup I(x) \quad (3)$$

$$B = \bigcup \{I(x_i) | i \in G\} \cup \{y_i | i \in G\} \quad (4)$$

$$C = OR(x_e) \cup \{x_e\} \quad (5)$$

$$D = B \cap C \quad (6)$$

$$E = D \setminus \bigcup \{IR(d) | d \in D\} \quad (7)$$

Где: n — количество блоков в алгоритме;

x — блок алгоритма;

O(x) — функция получения множества блоков, которые подключены к выходным разъемам блока x;

I(x) — функция получения множества блоков, которые подключены к входным разъемам блока x;

e — индекс блока, чьи параметры были изменены;

Результаты

Для демонстрации эффективности данного метода был проведен эксперимент, при котором подсчитывается время выполнения при изменении параметров стандартного скрипта и оптимизированного с помощью приведенного метода выполнения. На рисунке 2 показана блок-схема потока данных. Для эксперимента было взято 100 данных, а алгоритмы были выполнены 1000 раз с усреднением времени. Оба алгоритма написаны на языке Python с использованием библиотек Numpy, Scipy и Matplotlib [4]. Блоки реализованы с помощью внешних скриптов Python, которые имеют параметры, описывающие их компоненты.

На основе результатов, приведенных в таблице 1, можно сделать вывод, что оптимизированный метод работает медленнее при изменении параметров, влияющих на весь алгоритм, но значительно быстрее при изменении промежуточных параметров.

Заключение

В этой статье был приведен метод выполнения алгоритмов визуального анализа вибрационных данных и сделан эксперимент, в котором сравнивалась скорость выполнения алгоритма при использовании метода и стандартного скрипта. Результаты показали, что оптимизированный метод выполнения алгоритмов с помощью блоков позволяет повысить эффективность анализа данных при частом изменении параметров [5]. Он может использоваться совместно с визуальным языком программирования и графическим интерфейсом, который позволит удобно выводить элементы изменения параметров и отображение графиков. Это предоставит гибкость и функционал, аналогичный написанию простых скриптов, а также обеспечит оптимизацию, схожу с той, которая применяется при написании специализированных программ.

ЛИТЕРАТУРА

1. Mukhopadhyay M. Computer Programs in Vibration Analysis / M. Mukhopadhyay. — Text: electronic // Structural Dynamics. — Cham: Springer International Publishing, 2021. — P. 563–606.
2. Trancón y Widemann B. Foundations of Total Functional Data-Flow Programming / B. Trancón y Widemann, M. Lepper // Electronic Proceedings in Theoretical Computer Science. — 2014. — Vol. 153. — P. 143–167.
3. Hedgpeth R. The Case for Reactive Programming / R. Hedgpeth. — Text: electronic // R2DBC Revealed. — Berkeley, CA: Apress, 2021. — P. 3–16.
4. Kwak M.K. Dynamic modeling and active vibration control of structures / M.K. Kwak. — 2022.
5. Hu H. Vibration Mechanics: a research-oriented tutorial. Vibration Mechanics / H. Hu. — S.L.: Springer Verlag, Singapor, 2022.

Журнал «Современная наука: актуальные проблемы теории и практики»