Инженерный вестник Дона, №3 (2016)
ivdon.ru/ru/magazine/archive/n3y2015/3749

Программное обеспечение моделирования и фильтрации сигналов сложной нелинейной природы

В.В. Мисюра, И.В. Мисюра

Донской государственный технический университет, Ростов-на-Дону

Аннотация: приведены результаты проектирования и реализации программного обеспечения для моделирования, фильтрации и интерполяции сигнала для моделей сложной нелинейной природы, в частности модели стохастической волатильности. Рассматриваются вопросы разработки программного комплекса: инструментальные средства, объектно-ориентированная модель, методы распараллеливания. Разработанная система классов может быть использована независимо от пользовательского интерфейса и применяться в качестве вспомогательного механизма в различных программах моделирования и фильтрации сигналов, написанных с использованием фреймворка Qt.
Ключевые слова: нелинейная модель, стохастическая волатильность, сигнал, фильтрация, генерация, абстрактный класс.

Введение. В настоящее время наблюдается рост интереса к задачам связанным с исследованием случайных сигналов, которые описывают как экономические, так и технические модели и процессы, учитывающие случайные воздействия. Например, задачи определения динамических параметров строительных конструкций, таких как, собственные частоты колебаний, декремент колебаний, физические свойства материалов, задачи идентификации и локализации повреждений строительных конструкций и т.д. [1,2]

Для решения такого класса задач широко применяются математические моделей, в основе которых лежит нелинейная модель стохастической волатильности. Непрерывная модель стохастической волатильности в общем виде определяется уравнениями:

$$dY_t = AX_t dt + dW_t,\ dX_t = a\left(b_t\right)X_t dt + \sigma\left(b_t\right)dU_t, \tag{1}$$

$$db_t = \mu\left(b_t\right)dt + \nu\left(b_t\right)dV_t.$$

В (1) процесс диффузионный процесс b_t аппроксимируется марковской цепью с множеством состояний $B^h = \{b_0 + ih\}, i = 0,...,N$, h – расстояние между соседними состояниями.

Использование модели стохастической волатильности обусловлено тем, что с ее помощью можно описать резкие изменения свойств траектории сигнала. Задачам фильтрации, интерполяции и экстраполяции для нелинейных моделей сигналов посвящены как монографии, так и многочисленные статьи [3-6]. В работах [7-9] приводятся результаты по построению и анализу моделей, основанных на потраекторной смеси условных гауссовских распределений. Такой подход позволяет исследовать широкий класс сигналов и предложить эффективные численные методы их анализа. Теоретическое обоснование построенных алгоритмов, непосредственно полное их описание, численные методы фильтрации, интерполяции сигнала, сочетающие вычислительные методы решения дифференциальных уравнений с имитационным моделированием представлены в работах [7,9].

1. Объектно-ориентированная модель программного обеспечения. Реализует предложенные модели и вычислительные методы эффективный расширяемый программный комплекс для моделирования и анализа сигналов, позволяющий легко управлять подключенными к нему библиотеками фильтров, свободно добавлять и использовать их.

В качестве средства разработки программного обеспечения выбран язык высокого уровня C++ с использованием кроссплатформенного фреймворка Qt в среде разработки Qt Creator. Выбор данной среды обусловлен такими факторами как: наличие объектно-ориентированного языка высокого уровня (C++); бесплатность среды разработки; наличие эффективной визуальной среды разработки; наличие компонентной

структуры, позволяющей без труда внедрять и использовать программные продукты сторонних разработчиков, а также компоненты, разработанные для расширения функционала и механизмов среды; наличие современных технологий, позволяющих с легкостью решать поставленные задачи работы с данными, многопоточности, параллельности и др.; наличие эффективных методов отладки, позволяющих отлавливать ошибки как при компиляции (несоответствие типов, аргументов и т.д.), так и во время работы программы (утечки памяти, необработанные исключения, неосвобожденные дескрипторы ресурсных объектов) [10]. При этом помимо стандартных компонентов Qt в разработке был использован сторонний компонент, созданный другими разработчиками, для визуализации результатов моделирования сигналов и их фильтрации, который также основан на стандартных классах Qt.

Разработанный программный комплекс имеет расширяемую архитектуру и позволяет добавлять в него новые функциональные возможности: алгоритмы моделирования и обработки сигналов, методы ускорения работы алгоритмов и др. Разработанная система классов может быть использована независимо от пользовательского интерфейса и применяться в качестве вспомогательного механизма в различных программах, написанных с использованием фреймворка Qt. Реализованное в программном обеспечении распараллеливание нелинейных алгоритмов фильтрации, требующих больших вычислительных затрат, повышает быстродействие вычислительных процедур.

Программный комплекс состоит из четырех основных подсистем. На рисунке 1 представлена схема их взаимодействия.

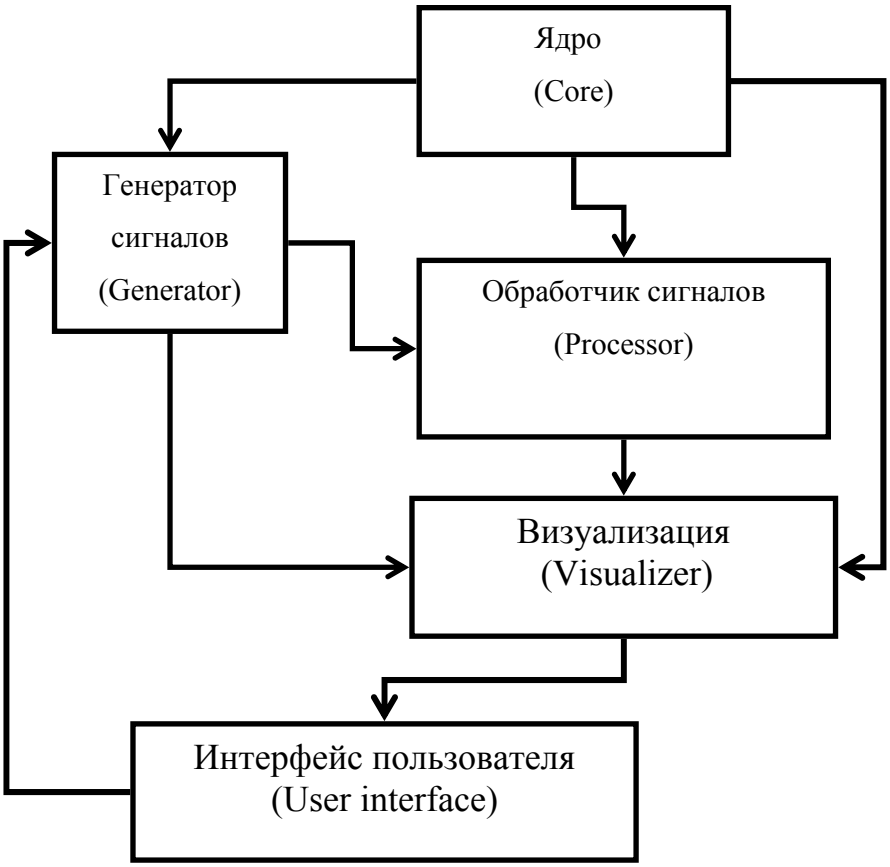

Рис. 1. Схема взаимодействия подсистем программного комплекса

Ядро системы отвечает за выполнение основных математических операций, как то генерация псевдослучайных чисел, генерация нормально распределенной случайной величины и т.д. Также в ядре определены необходимые для работы системы классы, методы, отвечающие за операции с векторами, в которых хранится информация о сигнале, а также за генерацию случайных отрезков на оси времени, отвечающих требованиям модели стохастической волатильности.

Генератор сигналов отвечает за моделирование зашумленных сигналов. Для непрерывного и дискретного случаев генерация происходит с помощью различных функций. Данные, необходимые для моделирования зашумленных процессов, т.е. параметры генерации, подсистема получает из интерфейса пользователя.

Обработчик сигналов принимает данные от генератора, т.е. получает в качестве входящей информации зашумленный сигнал, и передает на обработку модулям, реализующим алгоритмы фильтрации сигнала, соответствующие непрерывным и дискретным моделям.

Подсистема визуализации получает в качестве входящих данных сигналы от генератора или обработчика сигналов, подготавливает ее для отображения и передает информацию пользовательскому интерфейсу.

Интерфейс пользователя предоставляет возможность управления данными, необходимыми для моделирования сигналов, а также графически отображает данные, полученные от подсистемы визуализации.

Разработана система классов, реализующих подсистемы программного комплекса.

1. Классы DataSignal, Graph, AbstractSingleton (абстрактный класс) и FilterCore отвечают подсистеме ядра. Класс FilterCore отвечает за выполнение основных математических операций, используемых при моделировании и обработке сигнала. При его создании был использован шаблон проектирования «Одиночка» («Singleton»), что позволяет иметь только один экземпляр этого класса. Это создает условия для удобного и быстрого подключения ядра к другим классам, что позволяет использовать все его возможности без излишних объявлений класса и ссылок на его представителей. Реализация происходит с помощью абстрактного класса AbstractSingleton. Классы Signal и Graph наследованы от глобального класса Qt QObject. Это позволяет использовать при работе с объектами этих классов сигналы и слоты.

2. Классы AbstractGenerator (абстрактный класс), InfiniteGenerator и DiscreteGenerator соответствуют подсистеме генератора сигналов.

Абстрактный класс AbstractGenerator наследуется от глобального класса QObject для обеспечения возможности внедрения и использования сигналов и слотов. Класс InfiniteGenerator отвечает за моделирование сигналов, соответствующих непрерывной модели. Класс DiscreteGenerator моделирует сигналы, соответствующие модели с дискретным временем. Общедоступные (public) функции этих классов возвращают в качестве результата объекты класса DataSignal, которые содержат информацию о смоделированном сигнале. Вся остальная логика работы скрыта, т.е. прочие функции объявлены как закрытые (private) методы класса. Листинг программного кода приведен в приложении 1 (Листинг №2).

3. Классы SignalProcessor (абстрактный класс), DiscreteProcessor и InfiniteProcessor соответствуют подсистеме обработчика сигналов.

DiscreteProcessor обрабатывает объекты типа DataSignal, соответствующие дискретной модели, т.е. полученные в результате моделирования с помощью методов класса DescreteGenerator из подсистемы генератора сигналов. Соответственно InfiniteProcessor обрабатывает сигналы, соответствующие непрерывной модели. Основные (общедоступные) методы этих классов возвращают в качестве результата объект класса DataSignal.

4. Классы Visualizer и QCustomPlot (заимствованный класс) отвечают подсистеме визуализации. QCustomPlot – класс, созданный сторонними разработчиками, позволяющий использовать инструменты визуализации Qt более гибко. Используя этот класс, можно максимально эффективно и быстро демонстрировать результаты работы генератора и обработчика сигналов. Класс Visualiser осуществляет взаимодействие объектов классов QCustomPlot, DataSignal и Graph. Листинг приведен в приложении 1 (Листинг №4).

5. Пользовательский интерфейс реализован как класс MainWindow. Он наследуется от родного класса Qt QMainWindow. Преимущество этого наследования заключается в быстром реагировании на события, происходящие во время моделирования и обработки сигналов и мгновенной перерисовкой графических представлений сигналов. Также интерфейс позволяет работать не только с визуальным представлением моделированных и обработанных сигналов, но и с табличным представлением данных. Для этого был создан класс DetailedView, который тоже наследуется от QMainWindow.

2. Описание графического интерфейса. Графический интерфейс программного обеспечения генерации модели стохастической волатильности представлен на рисунке 2.

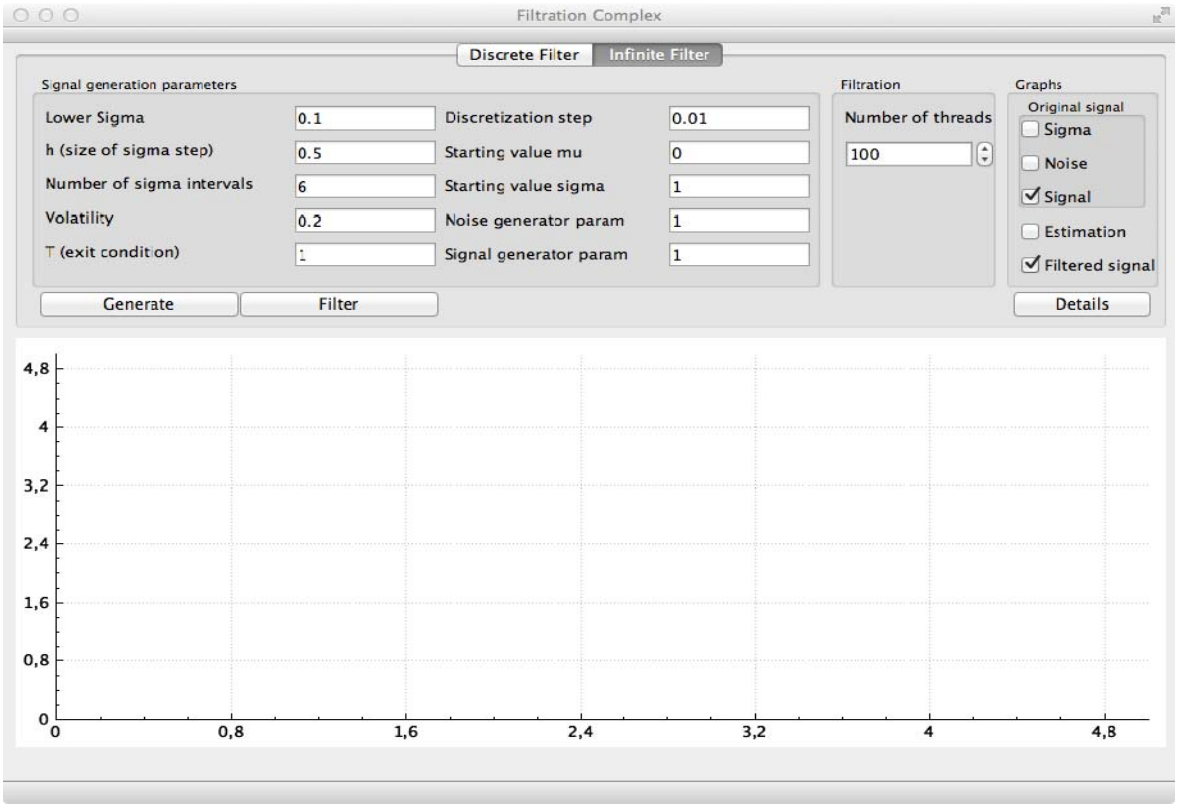

Рис. 2. Графический интерфейс, соответствующий модели стохастической волатильности

Область «Signal generation parameters» отвечает за ввод данных, необходимых для генерации случайных кусочно-постоянных функций θ, представляющих собой двумерные марковские последовательности. (см. [9]) и для генерации самого сигнала. Область «Filtration» отвечает за количество итераций нахождения оценки сигнала. Область «Graphs» отвечает за отображение различных графиков, что позволяет подробно изучать результаты моделирования и фильтрации.

Входные параметры: Lower Sigma– минимальное значение среднего квадратического отклонения σ , h_s – шаг изменения σ (size of sigma step), Number of Sigma Intervals – количество возможных значений σ , Volatility– волатильность g, T(exit condition) – длина горизонта, Discretization step– шаг дискретизации сигнала , Noise generator param – параметр генерации шума (1), Signal generator param – параметр генерации сигнала A (1), Starting value mu, Starting value sigma– математическое ожидание и дисперсия начального значения шума m_s, n_s.

Алгоритм моделирования сигнала и шума реализует формулу (1). В (1) диффузионный процесс b_t аппроксимируется марковской цепью с множеством состояний $B^h = \{b_0 + ih\}, i = 0,...,N$, h – расстояние между соседними состояниями. Матрица интенсивностей должна удовлетворять условию локальной согласованности, подробнее см. [11]. Одна из возможных приближенных схем с трехдиагональной матрицей интенсивностей Λ была построена в [12].

На рисунке 3 показан результат моделирования ненаблюдаемого шума и зашумленного сигнала при параметрах $a = 1,\ \sigma_1 = 0.1,\ h_s = 0.5,\ m = 7,$ $g = 0.2,\ T = 1,\ m_s = 0,\ n_s = 1,\ a = 1,\ A = 1,\ h = 0.01$.

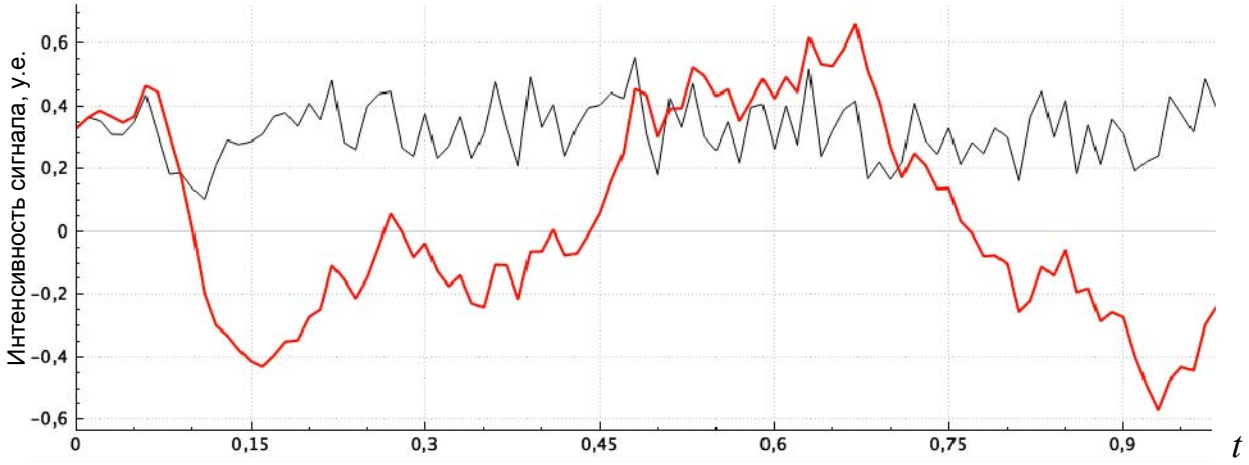

Рис. 3. Графическое представление зашумленного сигнала

$$a = 1, \quad A = 1, \quad h = 0.01$$

Основная задача фильтрации сигнала – найти оптимальную усредненную оценку сигнала, очищенную от шума. В работах [7,9] представлены алгоритмы фильтрации. Для её эффективного нахождения необходимо задать количество итераций, равное $N = T/h$. Выполнить генерацию случайной кусочно-постоянной функции θ.

В итоге работы программного модуля получаем эффективный фильтр, соответствующий модели стохастической волатильности. На рис. 4 продемонстрирован результат фильтрации.

Рис. 6**.** Результат фильтрации зашумленного сигнала в модели стохастической волатильности

3. Методы распараллеливания вычислений. Каждый шаг итерации включает в себя сложные вычислительные процессы, например, численное интегрирование для нахождения интеграла Римана-Стилтьеса. В этой ситуации целесообразно использовать средства распараллеливания вычислений. Был подключен модуль QtConcurrent, позволяющий запускать любую нестатическую функцию в отдельном потоке QThread. Помимо удобства внедрения, использование средств распараллеливания Qt предоставляет широкий спектр возможностей контроля каждого из потоков, а также возможность отслеживания используемых потоком ресурсов, мониторинга времени работы и многое другое. Так как QThread является представителем QObject, мы можем использовать сигналы для передачи информации о том, что поток завершил работу, и посылать уже готовые данные на пост-обработку, т.е. передать их в качестве входящего параметра для слота, отвечающего за усреднение оценок сигнала. В свою очередь, пост-обработка также происходит в отдельном потоке, не зависимом от потоков, отвечающих за нахождение оценок сигнала. Такой подход к распараллеливанию возможен только благодаря тому, что мы точно знаем, сколько необходимо итераций для нахождения оптимальной усредненной оценки. На рис. 7. показана схема распараллеливания фильтра.

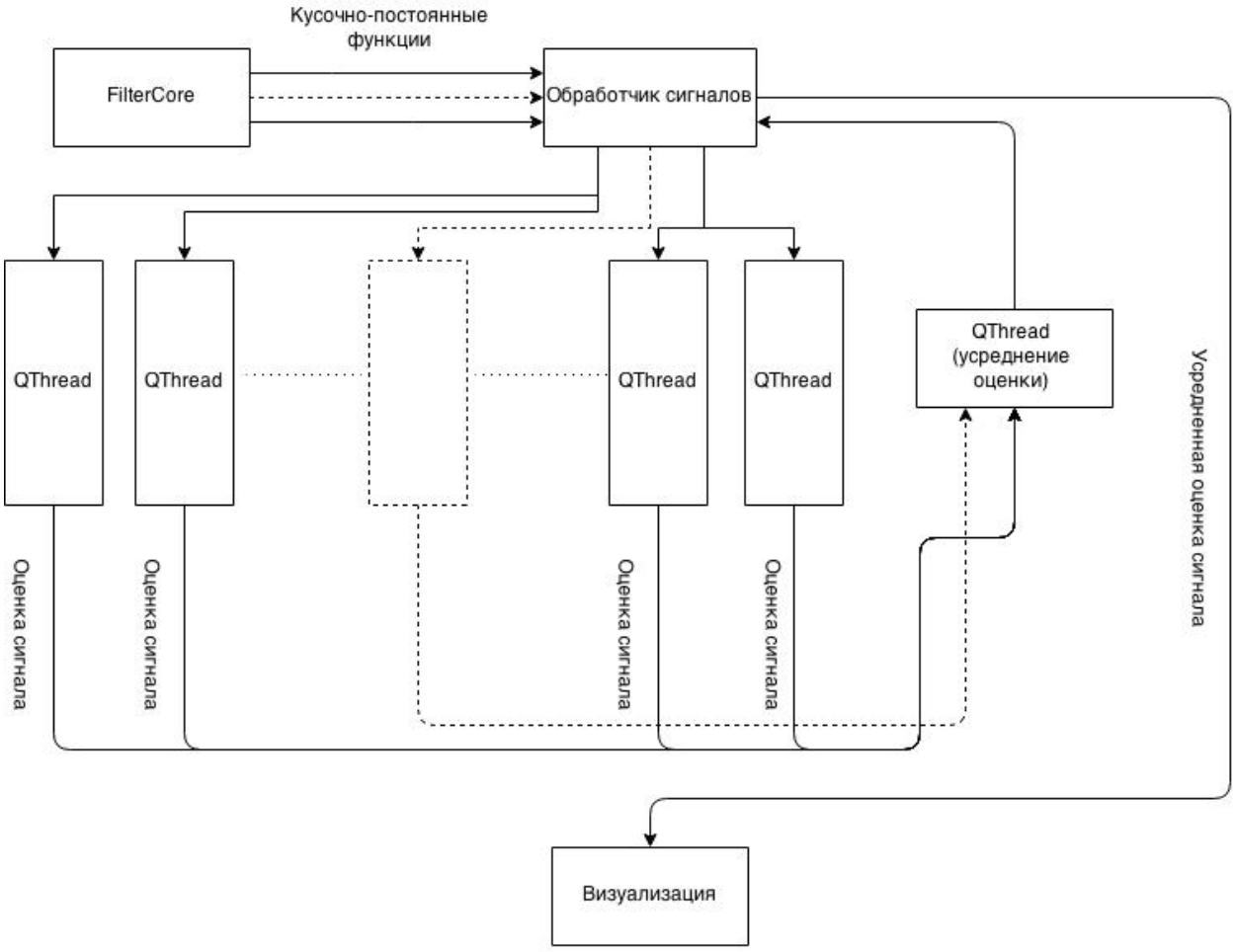

Рис. 7. Схема распараллеливания фильтра

Заключение. Из проведенных экспериментов следует, что программный комплекс позволяет получать хорошие оценки для сигналов сложной нелинейной природы. Например, линейная обработка сигнала при параметрах $A=1,\ \ h=0.01$, т.е. при 100 отсчетах сигнала, распараллеленный алгоритм проводил вычисления 12.5 секунд, в то время как линейное нахождение оценки заняло 34 секунды. Более того, существуют дальнейшие ступени распараллеливания. Например, можно каждый отрезок $[\tau_i, \tau_{i+1}]$ функции θ также обрабатывать в отдельном потоке, что должно привести к еще более эффективному ускорению работы фильтра.

Расширяемая архитектура программного обеспечения позволяет добавлять в него новые функциональные возможности моделирования и

обработки сигналов сложной нелинейной природы, например, авторегрессионных моделей условной неоднородности (ARCH, GARCH и т.д.). Для этого может быть использована разработанная система классов независимо от пользовательского интерфейса.

Работа выполнена при финансовой поддержке РФФИ в рамках научного проекта № 14-01-00579 а

Литература

1. Мисюра В.В., Мисюра И.В. Обработка и фильтрация сигналов. Современное состояние проблемы // Инженерный вестник Дона, 2013. №4. URL: ivdon.ru/magazine/archive/n4y2013/2130
2. Кадомцев М.И., Ляпин А.А., Шатилов Ю.Ю. Вибродиагностика строительных конструкций // Инженерный вестник Дона, 2012. №3 URL: ivdon.ru/ru/magazine/archive/n3y2012/941
3. Липцер Р. Ш., Ширяев А. Н. Нелинейная фильтрация диффузионных марковских процессов // Труды МИАН им. В.А. Стеклова АН СССР, 1968. Том 104. С. 135-180.
4. Bain A., Crisan D. Fundamentals of stochastic filtering. Springer, 2009. 390 p.
5. Shuss, Z. Nonlinear filtering and optimal tracing. Springer, 2012. 262 p.
6. Демин Д., Чуликов А. Фильтрация монотонных выпуклых сигналов, искаженных шумом, и оценка положения особых точек // Фундаментальная и прикладная математика, 2009. том 15, № 6. С. 15—31.
7. Мисюра И.В. Фильтрация сигнала для модели стохастической волатильности // Известия высших учебных заведений. Северо-Кавказский регион, 2014. №6. С. 27–31.

8. Белявский Г.И. Мисюра И.В. Фильтрация сигналов со скачками, возникающими в дискретном времени и с конечным горизонтом / Г.И. Белявский, // Научно-технические ведомости СПбГПУ. Физико-математические науки, 2014. Выпуск 2. С. 137 – 143.

9. Beliavskiy G. I., Misyura I. V. The signal filtration under switching regime model. The Monte-Carlo evaluation // Applied Mathematical Sciences, 2014. Vol. 8, no. 177. pp. 8833-8840.

10. Шлее М. Qt 5.3. Профессиональное программирование на C++. Спб: БХВ-Петербург, 2015. 928 с.

11. Rogers, L. C. G. A stochastic volatility alternative to SABR / L. C. G. Rogers and L. A. M. Veraart. // Journal of Applied Probability, 2008. vol. 45, no. 4. pp. 1071–1085.

12. Kushner, H. J. Numerical methods for stochastic control problems in continuous time // SIAM Journal of Control and Optimization, 1990. vol. 28, no. 5. pp. 999–1048.

References

1. Misjura V.V., Misjura I.V. Inženernyj vestnik Dona (Rus), 2013. №4 URL: ivdon.ru/magazine/archive/n4y2013/2130

2. Kadomcev M.I., Ljapin A.A., Shatilov Ju.Ju. Inženernyj vestnik Dona (Rus), 2012. №3 URL: ivdon.ru/ru/magazine/archive/n3y2012/941

3. Lipcer R. Sh., Shirjaev A. N. Trudy MIAN im. V.A. Steklova AN SSSR, 1968. Tom 104. pp. 135-180.

4. Bain A., Crisan D. Fundamentals of stochastic filtering. Springer, 2009. 390 p.

5. Shuss, Z. Nonlinear filtering and optimal tracing. Springer, 2012. 262 p.

6. Demin D., Chulikov A. Fundamental'naja i prikladnaja matematika, 2009. tom 15, № 6. pp. 15—31.

7. Misjura I.V. Izvestija vysshih uchebnyh zavedenij. Severo-Kavkazskij region, 2014. №6. pp. 27–31.

8. Beljavskij, G.I. Misjura I.V. Nauchno-tehnicheskie vedomosti SPbGPU. Fiziko-matematicheskie nauki, 2014. Vypusk 2 . pp. 137 – 143.

9. Beliavskiy G. I., Misyura I. V. Applied Mathematical Sciences, 2014. Vol. 8, no. 177. pp. 8833-8840.

10. Shlee M. Qt 5.3. Professional'noe programmirovanie na C++ [Qt 5.3. Advanced Programming in C ++.]. Spb: BHV-Peterburg, 2015. 928 p.

11. Rogers , L. C. G. Journal of Applied Probability, 2008. vol. 45, no. 4. pp. 1071–1085.

12. Kushner, H. J. SIAM Journal of Control and Optimization, 1990. vol. 28, no. 5. pp. 999–1048.