САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет
Кафедра информатики

Варфоломеева Екатерина Дмитриевна

Аналитический модуль для прогнозирования доходности финансовых инструментов на основании GARCH-модели волатильности

Дипломная работа

Допущена к защите:
Зав. кафедрой
д. ф.-м. н., проф. Косовский Н.К.

Научный руководитель:
д. ф.-м. н., проф. Граничин О.Н.

Рецензент:
аспирант Власов В.С.

Санкт-Петербург
2009

Оглавление

Введение 4
Глава 1. Социальные институты и проблема неопределённости 6
1.1 Фондовый рынок как социальный институт 6
1.2 Измерение риска и прогнозирование как способ решения проблемы неопределённости на примере фондового рынка. Волатильность 8
1.2.1 Понятие доходности 8
1.2.2 Расчёт волатильности 10
1.2.2.1 Простая волатильность (простое скользящее среднее, SMA) 11
1.2.2.2 Экспоненциальная волатильность (экспоненциально взвешенное скользящее среднее, EWMA) 13
1.2.2.3 Волатильность, как комбинация нескольких распределений 14
1.2.2.4 Подразумеваемая/предполагаемая (implied) волатильность 14
1.2.2.5 Реализованная (realized) волатильность 15
1.2.2.6 ARCH/GARCH модели 16
ARCH модели 16
GARCH модель 19
1.3 Краткий обзор существующего ПО 20
1.3.1 Программы, применяемые трейдерами 20
1.3.2 Программы, позволяющие осуществлять прогнозирование на основе GARCH-моделей. 24
1.4 Цель работы и постановка задачи 25
Глава 2. Прогнозирование с помощью GARCH-модели 26
2.1 ARMA модель 26
2.2 Оценка параметров GARCH-модели 28
2.2.1 Функция правдоподобия 28

2.2.2 Начальные значения условной дисперсии 29
2.2.3 Начальные приближения 30
2.2.4 Метод оценивания 31
2.2.4.1 Метод внешнего произведения градиентов 31
2.2.4.1.1 Вычисление условной информационной матрицы t-го наблюдения и вклада в градиент t-го наблюдения 33
2.2.4.1.2 Искусственная регрессия 34
2.2.1.1.3 Вычисление градиентов остатков 35
2.2.4.1.4 Вычисление градиентов условных дисперсий 35
2.2.4.1.5 Алгоритм вычислений 36
2.2.4.2 Рандомизированный алгоритм стохастической аппроксимации 37
2.2.4.2.1 Алгоритм вычислений 37
2.2.4.3. Сравнение алгоритмов 38
2.3 Прогнозы и оценка доверительных интервалов для GARCH-модели. 39
Глава 3. Аналитический модуль, осуществляющий прогнозирование доходности 42
3.1 Описание реализации 42
3.2 Пример работы аналитического модуля 43
Заключение 46
Список использованной литературы 48
Приложения. 51
Приложение 1 51
Приложение 2 57

Введение

В работе рассматривается проблема устранения неопределённости, имеющая большое значение в социологии в силу того, что зачастую ситуация неопределённости может привести к невыполнению агентами установленных в рамках социального института норм и правил, что отрицательно сказывается на сложившихся в рамках института отношениях и может привести к кризису. В качестве примера социального института рассматривается фондовый рынок, для которого решается проблема неопределённости путём прогнозирования и оценки рисков, характерных для этого института.

Как правило, прогнозирование и оценка рисков производится с помощью информационной системы поддержки принятия решений. Входными данными для системы, осуществляющей оценку рисков финансовых активов, являются доходности рассматриваемого финансового актива за определённый период. Система осуществляет прогнозирование будущей доходности финансового актива. Прогнозирование осуществляется с использованием инструмента волатильности. В главе 1.2.2. рассмотрены различные подходы к расчёту волатильности и для решения поставленной задачи выбрана GARCH-модель расчёта, приведено обоснование выбора.

В главе 2 выбранная GARCH-модель расчёта волатильности дополняется ARMA-моделью, описание которой дано в пункте 2.1. В пункте 2.2 рассматриваются алгоритмы построения и оценки параметров для выбранной модели методом максимального правдоподобия. В пункте 2.2.4.1 приводится алгоритм, основанный на методе внешнего произведения градиентов. В пункте 2.2.4.2 для решения поставленной задачи предлагается рандомизированный алгоритм стохастической аппроксимации. В пункте 2.2.4.3. приводится сравнение алгоритмов и обосновывается выбор рандомизированного алгоритма стохастической для решения поставленной задачи. Далее в пункте 2.3 рассмотрен принцип построения прогноза

доходности на заданный период и доверительного интервала с учётом используемой модели.

На основании GARCH-модели расчёта волатильности в сочетании с ARMA-моделью, коэффициенты которой оцениваются с помощью рандомизированного алгоритма стохастической аппроксимации, приведённого в главе 2, разработан аналитический модуль, осуществляющий прогнозирование доходности финансовых инструментов. Описание и пример работы аналитического модуля приведены в главе 3, код модуля приведён в приложении 2.

В заключении подведены итоги и обозначены направления дальнейшего исследования.

Глава 1. Социальные институты и проблема неопределённости

В социологии остро встаёт проблема устранения неопределённости, т.е. прогнозирования и оценки рисков, так как ситуация риска или неопределённости будущего может отрицательно повлиять на поведение социальных агентов и привести к социальной нестабильности.

В рамках социологии ключевое значение имеет понятие института. Социальный институт (лат. institutum — установление, учреждение) — система созданных людьми ограничений, выполнение которых поддерживается механизмами принуждения. В качестве механизмов принуждения могут выступать как санкции за невыполнение правил, так и различные поощрения за следование им. Санкции и поощрения могут носить как материальный, так и нематериальный характер.

Также социальный институт можно определить как:

- совокупность лиц, организаций, учреждений, материальных средств, обеспечивающая определённую общественную потребность посредством функционирования системы взаимосогласованных, целесообразно ориентированных стандартов поведения;
- устойчивый комплекс норм, правил и символов, регулирующий какую-либо из сторон человеческой жизнедеятельности и организующий их в систему ролей и статусов [6].

Ситуация же неопределённости или риска ведёт к социальной нестабильности, так как участники испытывают страх перед будущим, что может привести к невыполнению ими установленных норм и правил, деформации сложившихся в рамках института отношений, а в последствии и к кризису.

1.1 Фондовый рынок как социальный институт

Н. Флигстин выдвигает концептуальное предположение о том, что социальные институты включают в себя и рынки. Он рассматривает

социологическую модель действия, согласно которой участники рынка стремятся к созданию стабильных миров и решению проблемы конкуренции социальными средствами. Также он обращается к анализу тесной связи, которая существует между рынком и государством. В ходе исследования Н. Флигстин приходит к заключению, что «Рынки – это социальные конструкции, отражающие уникальную политико-культурную комбинацию фирм и наций» [25, с. 45]. В. В. Радаев же в рамках экономической социологии определяет рынок как «систему регулярного, преимущественно денежного, взаимовыгодного, добровольного и состязательного обмена благами, в которой действия его участников регулируются (помимо цен) их структурными связями, институциональными формами, властными иерархиями и культурными конструкциями» [19, с. 50].

Также актуальной для нашей страны является проблема повышения финансовой грамотности населения, что ещё раз было доказано программой, ставящей своей целью исследование текущего уровня финансовой грамотности населения и путей её повышения, проведённой в декабре 2007 – марте 2008 гг. [15]. Одной из основных целей просвещения населения является «развитие у населения навыков планирования и эффективного распоряжения своими финансами», а также «обеспечение понимания населением основных принципов эффективного и доходного вложения денежных средств» [24, с. 67]. Стахович Л.В. отмечает, что отличительными чертами самого крупного и развитого на сегодняшний день фондового рынка США являются «понятное и обоснованное прогнозирование», «прозрачность и доступность» и «активное консультирование, выдача рекомендаций» [23, с. 68].

Для института фондового рынка характерны большие объёмы данных, для сбора и анализа которых становится необходимой информационная система. Проблему неопределённости же помогает решить оценка рисков, что само по себе является одной из основных задач финансовых институтов. Рыночный риск – это возможность несоответствия характеристик

экономического состояния объекта значениям, ожидаемым лицами, принимающими решения под действием рыночных факторов [21, с. 7]. Различают следующие виды рыночного риска: процентный риск (interest rate risk), валютный риск (foreign exchange risk), риск колебаний рыночных цен товаров (commodity price risk), риск колебаний цен акций (equity price risk), риск производных инструментов (derivative risk) [28, с. 11]. Одной из мер риска является волатильность (volatility, изменчивость), отражающая колебания значений переменной и обобщающая все наблюдавшиеся значения выбранной характеристики.

1.2 Измерение риска и прогнозирование как способ решения проблемы неопределённости на примере фондового рынка. Волатильность

Как в качестве непосредственной меры риска, так и в качестве одной из важных величин, применяемых в более сложных методиках измерения риска (например, при измерении чувствительности цен облигаций и опционов, а также при расчёте стоимости под риском (VaR)) на фондовом рынке используется волатильность. В широком смысле под волатильностью понимают изменчивость, вариацию во времени величины финансового или экономического показателя. Мерой риска удобнее считать волатильность доходности, т. к. величина дохода зависит от размера или стоимости актива на начало и конец отчётного периода, а также от характера изменения этой стоимости в течение всего отчётного периода [16, с. 25].

1.2.1 Понятие доходности

Рассмотрим некоторый период измерения. Обозначим через P_t стоимость ценной бумаги в момент времени *t*, где *t* обозначает конец некоторого периода. Тогда абсолютное изменение стоимости ценной бумаги между датами *t* и *t-1*, определяется следующим образом:

$$D_t = P_t - P_{t-1}.$$

Относительное изменение цены или процентный доход (арифметическая или дискретная доходность) для аналогичного временного периода определяется как:

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}}.$$

Следующая модель доходности характерна в частности для опционов. Также её использование удобно в случае, если данными являются, например, обменные валютные курсы, которые могут быть выражены через каждую из двух валют для каждой пары: курс доллара в рублях или курс рубля в долларах. Распределения для этой модели любого из этих курсов абсолютно симметричны, чего нельзя сказать о распределении арифметической доходности. Логарифм изменения цены (непрерывно начисляемая доходность или геометрическая доходность), ценной бумаги определяется как натуральный логарифм его процентной доходности, а именно:

$$r_t = \ln(\frac{P_t}{P_{t-1}}).$$

На практике, основная причина, по которой работа с доходностями активов является более предпочтительной, чем с непосредственными ценами активов, заключается в том, что доходности имеют более привлекательные статистические свойства. Кроме того, доходности (относительные и логарифмические) очень часто предпочитаются абсолютным изменениям стоимости, потому что последние не показывают изменения относительно некоторого заданного ценового уровня. Преимуществом же логарифмической доходности является то, что она очень легко применяется для множества периодов (например, доходность за двухмесячный интервал времени может быть представлена как сумма двух одномесячных доходностей). Таким образом, далее под доходностью будем понимать логарифмическую доходность.

1.2.2 Расчёт волатильности

В сущности, волатильность – это мера неопределенности инвестиции. Любое движение цены может быть разбито на две части:

1) ожидаемое движение цены и

2) движение, которое мы не ожидали.

Ожидаемая (не волатильная) часть движения цены инвестиции обычно описывается понятием «Ожидаемой ставки доходности» (expected growth rate). На коротком промежутке времени большинство инвесторов "предсказывают" будущую норму доходности исходя из недавней модели поведения роста цены.

Неожидаемая часть движения цены инвестиции и есть ее волатильность. Эта часть может как расти, так и падать с равной вероятностью, т. к. по определению она не предсказуема. Ввиду нашей неопределенности относительно движения цены, существует вероятность распределения цен вокруг своей "ожидаемой" цены. Это распределение вероятности лежит в основе концепции "волатильности". Описывая его форму и размер мы, по сути, описываем волатильность [29].

Волатильность является случайной составляющей изменения доходности финансового инструмента и, в соответствии с формулой Ито [30], рассматривается следующим образом:

$$r_t = \mu + \varepsilon_t , \quad (1)$$

где μ – среднее изменение цены (тренд), ожидаемая часть её движения;

ε_t – волатильность, случайная величина (временной ряд) с нулевым матожиданием, которая во многом характеризуется дисперсией.

То есть движение цены за некоторый интервал рассматривается как некое планируемое трендовое движение и случайное отклонение от тренда определяемое волатильностью. Волатильность таким образом является случайной величиной, или (при рассмотрении изменения цены за несколько интервалов) временным рядом.

Некоторые источники также определяют волатильность как среднеквадратическое отклонение вероятностного распределения доходности.

Далее приводятся основные способы моделирования волатильности.

1.2.2.1 Простая волатильность (простое скользящее среднее, SMA)

В рамках простейшего представления волатильность рассматривается как нормально распределённая случайная величина с дисперсией, равной дисперсии доходности за интервал. Этот метод также носит название SMA (Simple moving average). [В соответствии с требованиями Базельского комитета, волатильность и коэффициенты корреляции будущих периодов должны оцениваться с помощью взвешенных с равными весами значений волатильности в течение, как минимум, предшествующего года]. Как оценка волатильности используется стандартное отклонение этой величины, рассчитанное по некой исторической выборке:

$$\overline{\sigma}_{\varepsilon} = \sqrt{\sum_{t=1}^{T} \frac{(r_t - \frac{1}{T}\sum_{i=1}^{T} r_i)^2}{T-1}},$$

где T – длина исторической выборки.

Естественно для расчёта волатильности таким образом следует рассмотреть достаточно большое количество интервалов. Например, для расчета однодневной волатильности желательно использовать не менее чем трехмесячную выборку однодневных изменений цен.

Достоинством такого рода модели является то, что она крайне проста, как с точки зрения расчёта волатильности, так и с точки зрения её дальнейшего применения. Чтобы рассчитать максимально возможное отклонение цены от среднего ожидаемого значения с заданной вероятностью достаточно просто умножить волатильность на коэффициент, определяемый свойствами нормального распределения. Так, чтобы рассчитать предельное изменение с вероятностью 95% (стандарт RiskMetrics), волатильность

необходимо умножить на 1,65. Вероятности 99% (требования Базельского комитета) соответствует коэффициент 2,33.

Простую волатильность также крайне просто моделировать, что позволяет легко использовать её в оценках риска, использующих метод Монте-Карло.

Недостатками такого расчёта волатильности являются:

- Несоответствие нормального распределения реальному распределению случайных доходностей. Реальные случайные доходности в целом не так сильно склонны отклоняться относительно нуля, как это моделируется нормальным распределением, но совершают иногда резкие скачки (имеют т. н. "тяжелые хвосты"). Представленные нормальным распределением случайные изменения с одной стороны склонны к сравнительно большим колебаниям около нуля, но, с другой стороны, не склонны к резким выбросам. Последнее наиболее неприятно, т. к. именно резкие случайные движения цен представляют наибольший интерес при оценке потерь.
- Расчёт характеристик волатильности по значительному историческому массиву приводит к "запаздыванию" оценки – произошедшие в течение последних дней или недель изменения волатильности не найдут в полной мере свое отражение в оценке волатильности.
  С другой стороны, при регулярном (например, ежедневном) расчете волатильности с одной и той же длиной выборки выход из выборки резких скачков, имевших место в прошлом, будет приводить к резкому изменению текущей волатильности.
- Этот подход не учитывает возможную автокорреляцию случайных изменений цен – например, в случае резкого однодневного скачка цен в последующие дни случайные изменения цен будут также выше своей "средней нормы", что способно существенно повлиять на характер принимаемых рисков (кластеризация волантильности).

1.2.2.2 Экспоненциальная волатильность (экспоненциально взвешенное скользящее среднее, EWMA)

Экспоненциальная волатильность также представляет случайные доходности, как нормальное распределение. Особенность этого метода расчёта волатильности в том, что при расчете стандартного отклонения данные исторической выборки включаются в расчёт с весовыми коэффициентами, увеличивающими вес недавних движений цен в выборке по сравнению с давними движениями.

Этот подход является стандартной моделью RiskMetrics.

Для оценки экспоненциальной волатильности используется следующая формула:

$$\overline{\sigma}_{\varepsilon} = \sqrt{(1-\lambda)\sum_{t=1}^{T}\lambda^{t-1}(r_t - \frac{1}{T}\sum_{i=1}^{T} r_i)^2},$$

где $0 < \lambda < 1$, λ – коэффициент сглаживания – весовой коэффициент, определяющий степень влияния на волатильность последнего изменения цен по сравнению с ранними оценками.

Заметим, что для вышеприведённых оценок волатильностей не подразумевается зависимость от времени. Этот параметр определяют относительные веса, которые применяются к наблюдениям (доходностям), что позволяет учитывать только эффективное количество данных при оценке волатильности.

Важным преимуществом оценки по методу EWMA является возможность выражения в рекурсивной форме, которая, в свою очередь, используется в качестве основы при вычислениях прогнозов волатильности.

$$\overline{\sigma}_1 = r_1,$$

$$\overline{\sigma}_t = \sqrt{\lambda\overline{\sigma}_{t-1}^2 + (1-\lambda)r_t^2},$$

Здесь учитывается зависимость волатильности от времени.

Экспоненциальная волатильность при использовании на практике интерпретируется аналогично простой, но она в большей мере отражает

недавние изменения цен и не подвержена резким изменениям по факту выхода из выборки достаточно старых резких изменений цен.

Соответственно оценка по модели EWMA имеет следующие преимущества:

- Оценка, получаемая с помощью модели EWMA, намного быстрее адаптируется к изменениям конъюнктуры рынка и резким колебаниям курсов, так как недавние события имеют больший вес, чем произошедшие в далеком прошлом.
- Быстро среагировав на шоковые значения доходности, далее важность этого события падает тем больше, чем больше времени прошло с момента шокового события, то есть не происходит переоценки риска на достаточно большом интервале времени.

1.2.2.3 Волатильность, как комбинация нескольких распределений

Для того, чтобы не отказываться от удобного при моделировании и расчете нормального распределения, и учесть эффект тяжелых хвостов, можно моделировать волатильность в виде комбинации нескольких нормальнораспределённых случайных величин с различными дисперсиями, каждая из которых "срабатывает" с некоторой предопределённой вероятностью. При моделировании отдельного движения рынка для расчета случайной составляющей согласно выбранным вероятностям выбирается одно из этих распределений. Недостатком этого метода является субъективность выставления вероятностей появления рассматриваемых случайных величин.

1.2.2.4 Подразумеваемая/предполагаемая (implied) волатильность

Этот тип волатильности характерен в частности для фьючерсов, поскольку цена фьючерса не является единственным возможным критерием при решении вопроса о его приобретении. Некоторые практики ориентируются в своей работе на величину волатильности: покупают опцион

при низкой волатильности и продают при высокой. Таким образом, возникает обратная задача – определить величину волатильности по заданным параметрам и заданному уровню цены. При решении такого рода задач прибегают к итерационному и другим приближённом методам.

1.2.2.5 Реализованная (realized) волатильность

Под реализованной волатильностью рынка понимается оценка волатильности для некоторого горизонта, рассчитанная на основе изменений на меньших горизонтах. Например, исходя из предположения о броуновском характере движения цен, можно на основе внутридневной пятиминутной волатильности оценить однодневную волатильность или на основе однодневной волатильности рассчитать месячную.

В качестве самого простого способа (броуновское движение цен) расчета реализованной волатильности можно воспользоваться следующей формулой:

$$\sigma_{t1} = (t1/t2)^{1/2}\sigma_{t2},$$

где *t1* – больший (оцениваемый) временной горизонт;

t2 – меньший (с достаточной статистикой) временной горизонт.

Однако, для серьезного практического использования эта формула достаточна груба, требуется дополнительное исследование свойств движений цен.

Преимуществами использования реализованной волатильности являются:

- Возможность на основе незначительной выборки получить оценку волатильности инструмента на значительном горизонте.
- Возможность оперативно предсказывать возможные скачки волатильности в будущем, на основе роста волатильности на коротких интервалах.

1.2.2.6 ARCH/GARCH модели

ARCH модели

Напомним, что волатильность обладает следующими свойствами:

1) Тяжёлые хвосты: Доходности активов как правило являются лептокуртическими (островершинными, имеющими высокий куртозис, имеющими положительный (выше нормального) эксцесс). Они не подчиняются Гауссовому распределению, а описываются так называемыми распределениями с «толстыми» хвостами и высокими пиками.
2) Кластеризация: т. е., как правило, большие изменения влекут за собой большие изменения, малые изменения следуют за малыми изменениями, любого знака. Эффект кластеризации волатильности отмечен для таких рядов как изменение цен акций, валютных курсов, доходности спекулятивных активов.

Модель ARCH (autoregressive conditional heteroskedasticity) была предложена Р. Энглом для моделирования кластеризации волатильности.

Процесс ARCH порядка q, $\{\varepsilon_t\}_{t=-\infty}^{+\infty}$ задаётся следующими соотношениями:

$$\varepsilon_t \,|\, \Omega_{t-1} \sim N(0, \sigma_t^2),$$
$$\sigma_t^2 = \omega + \gamma_1 \varepsilon_{t-1}^2 + \ldots + \gamma_q \varepsilon_{t-q}^2,$$

где $\Omega_{t-1}=(\varepsilon_{t-1},\ \varepsilon_{t-2},\ \ldots)$ – предыстория процесса ε_t, а σ_t^2 – условная по предыстории дисперсия ε_t, т. е. $\sigma_t^2 = V(\varepsilon_t \,|\, \Omega_{t-1}) = E(\varepsilon_t^2 \,|\, \Omega_{t-1})$. Для того, чтобы условная дисперсия оставалась положительной, требуется выполнение соотношений $\omega > 0$, и $\gamma_1, \ldots, \gamma_q \geq 0$.

Доходность в данном случае также определяется по формуле (1).

Рассмотрим $\varepsilon_t \equiv r_t - E(r_t \,|\, \Omega_{t-1})$, тогда условные дисперсии ε_t и r_t совпадают, т. е. σ_t является в этом случае волатильностью.

Рассмотрим теперь следующий нормированный процесс:

$$\xi_t \sim NID(0,1),$$
$$\varepsilon_t = \xi_t \sigma_t,$$
$$\sigma_t^2 = \omega + \gamma_1 \varepsilon_{t-1}^2 + ... + \gamma_q \varepsilon_{t-q}^2.$$

Величины ξ_t нормально распределены и независимы. Такая запись удобна тем, что этот нормированный случайный процесс ξ_t не зависит от предыстории.

Смысл модели ARCH в том, что он характеризуется кластеризацией волатильности. Если абсолютная величина ε_t оказывается большой, то это приводит к повышению условной дисперсии в последующие периоды, а при высокой условной дисперсии более вероятно появление больших по абсолютной величине значений ε_t. И наоборот, если значения ε_t в течение нескольких периодов близки к нулю, то это приводит к понижению условной дисперсии в последующие периоды практически до уровня ω. В свою очередь при низкой условной дисперсии более вероятно появление малых по абсолютной величине значений ε_t.

Ещё одно свойство ARCH-процессов состоит в том, что безусловное распределение ε_t имеет более высокий куртозис (т. е. более тяжёлые хвосты и острую вершину), что хорошо соответствует финансовым временным рядам.

Таким образом, ARCH-модель обладает некоторыми преимуществами перед моделями, описанными ранее, т. к. учитывает кластерность и тяжёлые хвосты волатильности.

Можно показать, что процесс ARCH не автокоррелирован:

$$E(\varepsilon_t \varepsilon_{t-j}) = E(E(\varepsilon_t \varepsilon_{t-j} \mid \Omega_{t-1})) = E(\varepsilon_{t-j} E(\varepsilon_t \mid \Omega_{t-1})) = 0.$$

Поскольку этот процесс имеет постоянное (нулевое) матожидание и не автокоррелирован, то он является слабо стационарным в случае, если у него есть дисперсия.

Рассмотрим $\eta_t = \varepsilon_t^2 - \sigma_t^2$. Тогда можно переписать ARCH-процесс как:

$$\varepsilon_t^2 = \omega + \gamma_1 \varepsilon_{t-1}^2 + ... + \gamma_q \varepsilon_{t-q}^2 + \eta_t .$$

Так как условное матожидание η_t равно нулю, то и безусловное матожидание также равно нулю, кроме того, η_t не автокоррелирован. Таким образом, квадраты процесса ARCH(q) следуют авторегрессионному процессу q-го порядка, что позволяет получить состоятельные оценки коэффициентов, используя это представление.

Рассмотрим корни характеристического уравнения

$$1 - (\gamma_1 x + ... + \gamma_q x^q) = 0 .$$

Если они лежат за пределами единичного круга, что, в силу неотрицательности γ_j эквивалентно условию $\sum_{j=1}^{q} \gamma_j < 1$, то у процесса ARCH($q$) существует безусловная дисперсия и он является слабо стационарным. Если это условие не выполняется, то безусловной дисперсии не существует и процесс не будет слабо стационарным. Для вычисления безусловной дисперсии берётся матожидание от обеих частей уравнения условной дисперсии:

$$E(\sigma_t^2) = \omega + \gamma_1 E(\varepsilon_{t-1}^2) + ... + \gamma_q (\varepsilon_{t-q}^2) .$$

Учитывая, что $E(\sigma_t^2) = E(E(\varepsilon_t^2 \mid \Omega_{t-1}))$ $E(\varepsilon_t^2)$ $Var(\varepsilon_t)$ σ^2, получаем:

$$\sigma^2 = \frac{\omega}{1 - \gamma_1 - ... - \gamma_q} .$$

Таким образом, все ε_t имеют одинаковую безусловную дисперсию, т. е. имеет место гомокседаксичность. Однако условная дисперсия меняется, поэтому одновременно имеет место условная гетерокседаксичность.

Недостатком же ARCH модели можно считать то, что для корректного описания данных требуется довольно большая длина лага q, что создаёт трудности при оценивании (например, ARCH(1), не даёт достаточно

длительных кластеров волатильности а только порождает большое число выбросов), в частности нередко нарушается условие неотрицательности оценок γ_j. С этим помогает справиться ограничение, наложенное Энглом на коэффициенты лага, состоящее в том, что они линейно убывают до нуля. Веса лага задаются соотношением: $\varsigma_j = \frac{q+1-j}{0.5q(q+1)}$ так, чтобы их сумма равнялась 1, а коэффициенты берутся равными $\gamma_j = \gamma\omega_j$. Таким образом, получается модель с двумя параметрами $\sigma_t^2 = \omega + \gamma(\varsigma_1\varepsilon_{t-1}^2 + ... + \varsigma_q\varepsilon_{t-q}^2)$.

GARCH модель

Память ARCH(q) процесса ограничена q периодами. При использовании модели часто требуется длинный лаг q и большое число параметров. Обобщенный ARCH процесс (Generalized ARCH, GARCH), предложенный Т. Боллерслевом [18], имеет бесконечную память и допускает более экономную параметризацию, модель GARCH(p,q) выглядит следующим образом:

$$\sigma_t^2 = \omega + \sum_{j=1}^{p}\delta_j\sigma_{t-j}^2 + \sum_{j=1}^{q}\gamma_j\varepsilon_{t-j}^2 .$$

При этом предполагается, что ω >0, δ_1, …, $\delta_p \geq 0$ и γ_1, …, $\gamma_q \geq 0$.

Как и в модели ARCH σ_t^2 – условная дисперсия процесса:

$$\varepsilon_t \,|\, \Omega_{t-1} \square N(0, \sigma_t^2) .$$

По аналогии с процессом ARCH рассчитаем безусловную дисперсию GARCH:

$$\sigma^2 = \frac{\omega}{1 - \sum_{j=1}^{p}\delta_j - \sum_{j=1}^{q}\gamma_j} .$$

Таким образом, с точки зрения безусловной дисперсии GARCH-процесс гомокседастичен. Чтобы дисперсия была конечной, требуется:

$$\sum_{j=1}^{p}\delta_j + \sum_{j=1}^{q}\gamma_j < 1 .$$

По аналогии с ARCH процесс GARCH можно записать в эквивалентной форме:

$$\varepsilon_t^2 = \omega + \sum_{j=1}^{m}(\delta_j + \gamma_j)\varepsilon_{t-j}^2 + \eta_t - \sum_{j=1}^{p}\delta_j\eta_{t-j},$$

где $\eta_t = \varepsilon_t^2 - \sigma_t^2$, $m = \max(p,q)$.

(В этой записи подразумевается δ_j=0 про $j>p$ и γ_j=0 при $j>q$). Такая форма записи позволяет увидеть, что квадраты GARCH-процесса подчиняются модели ARMA(m,p), что позволяет получить автокорреляционную функцию квадратов GARCH-процесса.

GARCH-процесс, как и его частный случай ARCH-процесс имеет более высокий куртозис, чем нормальное распределение, причём безусловное распределение отдельного наблюдения GARCH-процесса является симметричным, поэтому все нечётные моменты, начиная с третьего, равны нулю.

Таким образом, для дальнейших расчётов мы будем рассматривать GARCH-модель волатильности, т. к. она наиболее полно отражает указанные выше свойства волатильности и допускает более экономную параметризацию, чем ARCH-модель. С точки зрения прогнозирования перспективной является модель, сочетающая GARCH с некоторой моделью, описывающей поведение условного или безусловного среднего наблюдаемого ряда. Перспективным считается сочетание ARMA и GARCH моделей, которое используется в рамках дипломной работы для решения поставленной задачи.

1.3 Краткий обзор существующего ПО

1.3.1 Программы, применяемые трейдерами

Компьютерные программы, применяемые трейдерами в процессе торговли, можно условно разделить на несколько категорий:

1) Программы для технического анализа исторической ценовой информации. Технический анализ, так же как и фундаментальный анализ используется при краткосрочном инвестировании на бирже. К наиболее популярным программам технического анализа, используемым российскими трейдерами и инвесторами, относятся: MetaStock, Omega Research Trade Station и WealthLab. Эти программы содержат в себе огромный комплекс средств и функций для анализа исторических цен. В них можно обрабатывать исторические данные с помощью различных методов и приемов технического анализа. Например, рисовать линии, уровни поддержки и сопротивления, анализировать ценовые движения с помощью всевозможных индикаторов, создавать собственные торговые стратегии и тактики, используя встроенный язык программирования.

Самой известной программой в России считается MetaStock2. Она достаточно давно существует и хорошо себя зарекомендовала, для нее можно найти и скачать в Интернете многие торговые стратегии и индикаторы. Система содержит в себе мощные инструменты для анализа рынка. Но у нее есть и ряд недостатков. Например, MetaStock не дает возможности реализовывать механизмы управления капиталом, а только помогает выставлять стоп-приказы. Более того, учет сделок, а значит и расчет прибылей и убытков, у нее осуществляется только по какому-то одному параметру. Например, только по цене открытия или только по цене закрытия, или только по максимальной или по минимальной цене. Также к существенным недостаткам можно отнести то, что пользователь не может легко предоставить программе свои собственные данные и, построив собственные показатели, проанализировать их.

Программа TradeStation3 очень схожа по внешнему виду и по функциональным возможностям с MetaStock. Полезными функциями этих программ является возможность самостоятельно создавать индикаторы, торговые идеи, системы, тестировать их работоспособность, оптимизировать и применять в торговле, в том числе и в режиме реального времени.

Программа TradeStation обладает более широкими возможностями по применению вышеприведенных функций, чем программа MetaStock. При этом реализовать многие функции в программе TradeStation существенно сложнее, чем в программе MetaStock. В MetaStock для создания индексов используется язык макрокоманд, а в TradeStation полноценный язык программирования с полной функциональностью, необходимой профессиональным трейдерам. В частности, в этом языке можно описывать переменные, константы, задавать многомерные массивы, использовать циклы, переходы по ссылке, сравнения и условия, работать с датами и временем. Кроме этого, язык позволяет выводить информацию на экран или напрямую в файл, что очень удобно, когда возникает необходимость создать собственную таблицу с заданными параметрами стратегии [32]. Этот язык более сложный, для работы с ним нужно иметь навыки программирования, но зато он позволяет делать такие вещи, которые невозможно сделать в программе MetaStock.

2) Программы для Интернет-торговли, позволяющие покупать и продавать акции на бирже и получать оперативную финансовую информацию. Они также существенно облегчают жизнь игрокам на бирже. До их появления, все торговые операции на бирже осуществлялись путем подачи брокеру поручений, которые передавались ему в бумажном виде, иногда по телефону, но с обязательным последующим письменным подтверждением. Это было крайне неудобно и требовало дополнительных затрат времени и сил. Интернет-технологии позволили участникам рынка самостоятельно совершать операции на бирже.

Наиболее популярным программным продуктом, позволяющим совершать торговые операции на российских биржах, является программа Quik. Она появилась в 1995 году. Эту программу используют большинство брокеров. С ее помощью можно достаточно легко совершать операции купли/продажи с российскими ценными бумагами на бирже и параллельно

получать самую свежую и реальную информацию с финансовых площадок [20].

Главным элементом Интернет-торговли является сервер QUIK. Сервер через специальные "шлюзы" подключен к торговой системе биржи. Он передает или транслирует информацию о торгах на бирже всем активным (подключенным) пользователям. Сервер QUIK так же принимает и передает полученные от пользователей поручения на покупку/продажу ценных бумаг в торговую систему биржи. Таким образом, клиенты через сервер брокера получают возможность самостоятельно участвовать в торгах на бирже, посылая заявки из программы Quik непосредственно со своего компьютера.

3) Интеллектуальные программы для принятия торговых решений, которые представляют собой готовые торговые системы, дающие конкретные рекомендации на покупку и продажу акций. В отличие от программ первого типа, такие торговые системы выполняют всю аналитическую работу по поиску наиболее доходных акций, они избавляют своих пользователей от возможности совершить технические или психологические ошибки, из-за которых торговля на бирже считается весьма опасной. Их в России пока очень мало. Наиболее интересной и проработанной является программа PiAdviser (Personal Investment Adviser). Для многих новичков с самого начала работы на бирже достаточно трудно сориентироваться, какие акции покупать, какие продавать, когда и как правильно это делать, продавать акции по частям или одним пакетом. На эти и многие другие вопросы подобного рода может помочь найти ответ программа PIAdviser. Это одна из программ, которая дает четкие и подробные торговые рекомендации по купле/продаже акций. Программа прогнозирует движения цены и составляет инвестиционные рекомендации, предназначенные для покупки и продажи ценных бумаг. Программа PIAdviser помогает пользователю вести и анализировать свою инвестиционную и спекулятивную деятельность на рынке акций.

1.3.2 Программы, позволяющие осуществлять прогнозирование на основе GARCH-моделей.

Для построения GARCH-моделей можно воспользоваться такими программными средствами, как MatLab, EViews, SPSS, STATA. Оценки модели осуществляются методом максимального правдоподобия. Далее в работе будут рассмотрены алгоритмы построения оценок параметров GARCH-модели в рамках этого метода, в частности описан метод внешнего произведения градиентов и предложен рандомизированный алгоритм стохастической аппроксимации, ранее для решения этой задачи не использовавшийся. Соответственно, существующее программное обеспечение использует для оценки по методу максимального правдоподобия градиентный метод.

EViews позволяет построить оценки коэффициентов GARCH-модели и графическое отображение для условных дисперсий. В SPSS существует возможность работать с моделями типа ARIMA, однако такой же возможности для моделей типа GARCH в стандартном пакете нет. STATA имеет достаточно обширный набор функций для работы с GARCH и ARMA-моделями, однако в ней отсутствует возможность прогнозирования. Среди достоинств MatLab следует отметить доступность кода, возможность ознакомится с теорией, достаточное количество встроенных функций, возможность легко совмещать модели ARMA и GARCH, возможность строить прогноз на основе модели и графическое отображение. Однако необходимо заметить, что для использования MatLab необходимо иметь навыки программирования.

Для построения и оценки моделей с помощью указанных программных средств пользователю приходится самостоятельно выбирать и оценивать порядок модели, а также загружать данные, необходимые для построения прогноза, в систему. Аналитический модуль, являющийся целью дипломной работы, включает в себя процедуры оценивания порядка моделей ARCH и GARCH и, на основе оценки, выбора порядков моделей, аппроксимирующих

рассматриваемый набор данных. Также планируется встроить разработанный модуль в систему, имеющую удобный интерфейс и предоставляющую пользователю информацию, а также осуществляющую прогнозы на основе хранящихся в ней данных.

1.4 Цель работы и постановка задачи

Целью дипломной работы является создание аналитического модуля, позволяющего по имеющимся данным прогнозировать будущие значения доходности и строить доверительный интервал прогноза на основании GARCH-модели в сочетании с моделью ARCH, для проекта LiveTrade Online.

Пусть у нас имеется стохастический процесс r_t с дискретным временем, $t=1,\ldots,T$, и предысторией $\Omega_T = (r_{T-1}, r_{T-2}, \ldots, r_1, \ldots)$, где под r_t мы подразумеваем логарифмическую доходность некоторого финансового инструмента.

Требуется спрогнозировать значение r_t для некоторых периодов $t=T+1\ldots\ T+m,\ m\geq 1$ на основе GARCH-модели расчёта волатильности, совмещённой с моделью ARCH, и построить доверительный интервал прогноза.

Глава 2. Прогнозирование с помощью GARCH-модели

На практике модель GARCH обычно дополняют какой-либо моделью, описывающей поведение условного или безусловного среднего наблюдаемого ряда. Как уже было сказано, доходность определяется следующим образом:

$$r_t = \mu + \varepsilon_t .$$

Можно предположить, что наблюдаемый ряд имеет постоянное безусловное математическое ожидание μ, к которому добавляется ошибка ε_t в виде процесса GARCH.

Можно моделировать математическое ожидание с помощью линейной регрессии, т. е. $r_t = X_t\beta + \varepsilon_t$. Это позволяет учитывать линейный тренд, детерминированные сезонные переменные и т. д. С точки зрения прогнозирования перспективной является модель, сочетающая ARMA и GARCH. Модель ARMA используется для моделирования поведения условного математического ожидания ряда, GARCH – для моделирования условной дисперсии.

2.1 ARMA модель

ARMA(k,l) представляет собой комбинацию двух различных типов процессов AR(k) и MA(l) и принимает следующий вид:

$$X_t = \sum_{j=1}^{k} \beta_j X_{t-j} + \varepsilon_t + \sum_{j=1}^{l} \alpha_j \varepsilon_{t-j} ,$$

где ε_t представляет собой «белый шум», т. е. удовлетворяет условиям теоремы Гаусса-Маркова.

Стационарность ARMA процесса определяется только его AR частью. Поэтому условия те же самые, что у процесса AR: все корни характеристического уравнения $\lambda^k - \beta_1\lambda^{k-1} - ... - \beta_k = 0$ лежат внутри единичного круга, это условие необходимо и достаточно.

Заметим, что для нашей модели достаточно процесса ARMA(*k*, 0), т. к. в противном случае МА компоненты будут вносить свой вклад в условную дисперсию прогнозируемой величины r_t. Будем рассматривать следующую модель прогнозируемого процесса:

$$r_t = \beta_0 + \sum_{j=1}^{k} \beta_j r_{t-j} + \varepsilon_t ,$$

где $\beta_0 = \mu(1 - \beta_1 - ... \beta_p)$, μ – безусловное математическое ожидание.

Тогда для прогноза на один шаг вперёд можно записать:

$$\tilde{r}_{T+1} = E(r_{T+1} \mid \Omega_T) = \beta_0 + \sum_{j=0}^{k} \beta_j r_{t-j} .$$

Для прогноза на 2 шага вперёд:

$$E(r_{T+2} \mid \Omega_T) = E(\beta_0 + \beta_1 r_{T+1} + \sum_{j=1}^{k-2} \beta_j r_{t-j} + \varepsilon_{T+2} \mid \Omega_T) .$$

Матожидание ε равно нулю, условные матожидания $r_1, ..., r_T$ равны самим этим значениям, но в это выражение входит условное матожидание от r_{T+1}, полученное на предыдущем шаге. Рассмотрим рекуррентное соотношение, связывающее последовательные значения прогноза. Это соотношение является линейным разностным уравнением порядка *k*, и его решение стремится при увеличении *t* к $\mu = \frac{\beta_0}{1 - \beta_1 - ... - \beta_k}$, т. е. к безусловному прогнозу.

Выбор порядка модели будем осуществлять при помощи критерия Шварца, т. к. Шибата доказал, что для процессов ARMA(*k*,0) критерий Акаике переоценивает порядок модели [11]. Критерий Шварца в данном случае будет иметь вид:

$$BIC(k) = \ln \bar{\sigma}^2 + \ln k , \; \bar{\sigma}^2 = \frac{RSS}{T - k} .$$

Далее заметим, что выбор порядка модели GARCH мы так же будем осуществлять с помощью критерия Шварца, однако для этого случая:

$$BIC(p,q) = \ln\bar{\sigma}^2 + \ln(p+q)\,,\ \bar{\sigma}^2 = \frac{RSS}{T-p-q}\,.$$

2.2 Оценка параметров GARCH-модели

Оценим параметры GARCH-модели с помощью метода максимального правдоподобия, который даёт состоятельные и асимптотически эффективные оценки.

При оценивании в функции правдоподобия вместо ε_t используют r_t - $X_t\boldsymbol{\beta}$, где $X_t\boldsymbol{\beta} = E(r_t \mid \Omega_{t-1}) = \beta_0 + \sum_{j=1}^{k}\beta_j r_{t-j}$ моделируется с помощью ARMA(k,0).

2.2.1 Функция правдоподобия

Рассмотрим условие об условной нормальности, $\Omega_{t-1} = \{\varepsilon_{t-j}\}_{j=1}^{\infty}$:

$$\varepsilon_t \mid \Omega_{t-1} \sim N(0,\sigma_t^2)\,.$$

В предположении условной нормальности вклад в логарифмическую функцию правдоподобия t-го наблюдения равен

$$\ell_t = -\frac{1}{2}\ln 2\pi - \frac{1}{2}\ln\sigma_t^2 - \frac{1}{2}\frac{\varepsilon_t^2}{\sigma_t^2}\,,\ t = 1,\ ...,\ T.$$

Здесь ε_t и σ_t^2 следует рассматривать как функции от неизвестных параметров $\boldsymbol{\theta}$:

$$\varepsilon_t = \varepsilon_t(\boldsymbol{\theta}) \text{ и } \sigma_t^2 = \sigma_t^2(\boldsymbol{\theta})\,.$$

В вектор $\boldsymbol{\theta}$ должны войти параметры $\boldsymbol{\beta}$, ω, $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$ и любые другие неизвестные параметры (см. ниже).

Логарифмическая функция правдоподобия равна

$$\ell(\boldsymbol{\theta}) = -\frac{1}{2}(T\ln 2\pi + \sum_{t=1}^{T}[\ln\sigma_t^2(\boldsymbol{\theta}) + \frac{\varepsilon_t^2(\boldsymbol{\theta})}{\sigma_t^2(\boldsymbol{\theta})}])\,.$$

Оценки максимального правдоподобия по определению должны максимизировать логарифмическую функцию правдоподобия ℓ по неизвестным параметрам $\boldsymbol{\theta}$.

Оценки максимального правдоподобия являются корнями уравнения правдоподобия: $\frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{\mathbf{T}}} = \mathbf{0}$.

2.2.2 Начальные значения условной дисперсии

Для того чтобы использовать рекуррентную формулу GARCH-процесса для расчета условной дисперсии, используемой в функции правдоподобия, нужны начальные значения σ_t^2, $t = \min\{1, q + p - 1\}, ..., q$. Один из способов — рассматривать эти начальные значения как параметры, которые требуется оценить. Обозначим эти параметры через ζ_t (= σ_t^2), тогда рекуррентная формула примет вид:

$$\sigma_t^2(\boldsymbol{\theta}) = \begin{cases} \zeta_t, t = \min\{1, p + q - 1\}, \dots p \\ \omega + \sum_{j=1}^{p} \delta_j \sigma_{t-j}^2 + \sum_{j=1}^{q} \gamma_j \varepsilon_{t-j}^2, t \quad p+1, \dots, T \end{cases} \quad (2)$$

Можно также использовать вместо начальных значений безусловную дисперсию:

$$\sigma_t^2 = \sigma^2, \; t = \min\{1, q + p - 1\}, \dots, p, \text{ где } \sigma^2 = \frac{\omega}{1 - \sum_{j=1}^{p} \delta_j - \sum_{j=1}^{q} \gamma_j}.$$

При этом предполагается, что $\sum_{j=1}^{p} \delta_j + \sum_{j=1}^{q} \gamma_j < 1$, т. е., что GARCH-процесс стационарен. В дальнейшем рассмотрим только первый способ (с параметрами ζ_t). При этом в вектор $\boldsymbol{\theta}$ следует включить и ζ_t. Если GARCH-процесс стационарен, то с асимптотической точки зрения выбор начальных приближений не играет роли.

2.2.3 Начальные приближения

Большую роль играет выбор начальных приближений параметров $\boldsymbol{\theta} = \boldsymbol{\theta}^0$. Неправильный выбор может вызвать переполнение при вычислениях. Можно предложить следующие начальные приближения:

Для вектора коэффициентов регрессии $\boldsymbol{\beta}$ можно использовать оценки обычного метода наименьших квадратов, который даёт состоятельные оценки, а поскольку ε_t – гауссов процесс, то и асимптотически нормальные:

$$\boldsymbol{\beta}^0 = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{r}\,.$$

Затем на основе остатков из регрессии можно вычислить оценку безусловной дисперсии:

$$\overline{\sigma}^2 = \frac{1}{T}\sum_{t=1}^{T}\varepsilon_t^2\,.$$

Параметры γ_j^0 и δ_j^0 выбираются положительными и такими, что

$$\sum_{j=1}^{q}\gamma_j^0 + \sum_{j=1}^{p}\delta_j^0 < 1.$$

Можно взять, например,

$$\gamma_j^0 = \frac{1}{4q},\ \delta_j^0 = \frac{1}{4p}\ (\text{при } p>0).$$

Другой вариант — выбирать γ_j^0 и δ_j^0 случайно.

Затем на основе оценки безусловной дисперсии и коэффициентов γ_j и δ_j вычисляется начальное приближение для ω:

$$\omega^0 = \overline{\sigma}^2(1 - \sum_{j=1}^{q}\gamma_j^0 - \sum_{j=1}^{p}\delta_j^0)\,.$$

В качестве начальных приближений параметров ζ_t можно взять оценку безусловной дисперсии:

$$\zeta_t^0 = \overline{\sigma}^2\,.$$

2.2.4 Метод оценивания

2.2.4.1 Метод внешнего произведения градиентов

Для оценивания модели можно применить следующий общий итерационный метод, взятый из лекций А. Цыплакова [17]:

$$\boldsymbol{\theta}^{z+1} = \boldsymbol{\theta}^{z} + (\overline{\Gamma}(\boldsymbol{\theta}^{z}))^{-1}\mathbf{g}(\boldsymbol{\theta}^{z}), \quad (3)$$

где $\mathbf{g}(\boldsymbol{\theta}) = \dfrac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{\mathbf{T}}}$ — градиент логарифмической функции правдоподобия (вектор-столбец), $\overline{\Gamma}(\boldsymbol{\theta})$ — оценка информационной матрицы, z — номер итерации.

Стационарная точка итераций ($\boldsymbol{\theta}^{z+1} = \boldsymbol{\theta}^{z}$) соответствует решению уравнения правдоподобия:

$$\mathbf{g}(\boldsymbol{\theta}^{z}) = \mathbf{0}.$$

Алгоритм (3) на практике применяют обычно в несколько модифицированном виде:

$$\boldsymbol{\theta}^{z+1} = \boldsymbol{\theta}^{z} + \lambda(\overline{\Gamma}(\boldsymbol{\theta}^{z}))^{-1}\mathbf{g}(\boldsymbol{\theta}^{z}),$$

где $\lambda > 0$. Такая модификация нужна для повышения устойчивости алгоритма. Например, при выборе $\lambda = 1$ функция правдоподобия может уменьшится или шаг может попасть в недопустимую область (в GARCH-модели это случай, когда одна из оценок условной дисперсии становится отрицательной). В таких случаях множитель λ уменьшают до тех пор, пока не достигнут требуемого — увеличения функции правдоподобия или попадания в допустимую область.

Информационную матрицу можно оценить разными методами. Это, например, может быть матрица Гессе логарифмической функции правдоподобия со знаком минус:

$$\overline{\Gamma}(\boldsymbol{\theta}) \quad -\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^{\mathbf{T}}}.$$

При этом итерации (3) соответствуют классическому методу Ньютона. К сожалению, в модели GARCH матрицу Гессе довольно трудно вычислять.

Другой метод (его применяли те, кто предложил модель GARCH — Энгл, Боллерслев) [18] называется методом внешнего произведения градиентов (OPG). В этом методе информационная матрица оценивается следующим способом:

$$\overline{\Gamma}(\boldsymbol{\theta}) = \mathbf{G}(\boldsymbol{\theta})^{\mathrm{T}}\mathbf{G}(\boldsymbol{\theta}),$$

где **G** — матрица вкладов в градиент отдельных наблюдений:

$$\mathbf{G}(\boldsymbol{\theta}^r) = \{G_t(\boldsymbol{\theta})\}_{t=1,..T},$$
$$G_t(\boldsymbol{\theta}) = \frac{\partial \ell_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \quad .$$

Рассмотрим также следующий метод, который считается лучше рассмотренных ранее. В нем оценка информационной матрицы получается по формуле:

$$\overline{\Gamma}(\boldsymbol{\theta}) = \sum_{t=1}^{T} \overline{\Gamma}_t(\boldsymbol{\theta}),$$

где $\overline{\Gamma}_t(\boldsymbol{\theta})$ — информационная матрица для *t*-го наблюдения *условная* по прошлой информации Ω_t:

$$\overline{\Gamma}_t(\boldsymbol{\theta}) \quad \mathbf{E}\left(\frac{\partial \ell_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{\mathbf{T}}} \frac{\partial \ell_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \mid \Omega_t\right) \quad \mathbf{E}\left(G_t(\theta)^T G_t(\theta) \mid \Omega_t\right).$$

Для применения метода нужны формулы для расчета условной информационной матрицы *t*-го наблюдения, $\overline{\Gamma}_t(\boldsymbol{\theta})$, и для расчета градиента $\mathbf{g}(\boldsymbol{\theta})$. Градиент удобно разложить на вклады отдельных наблюдений:

$$\mathbf{g}(\boldsymbol{\theta})^{\mathrm{T}} = \mathbf{1}^{\mathrm{T}}\mathbf{G} = \sum_{t=1}^{T} G_t(\boldsymbol{\theta}) \quad .$$

Таким образом, требуется вычислить вклады в градиент $G_t(\boldsymbol{\theta})$.

2.2.4.1.1 Вычисление условной информационной матрицы t-го наблюдения и вклада в градиент t-го наблюдения

Продифференцируем $\ell_t(\boldsymbol{\theta})$, чтобы получить формулу для вклада в градиент *t*-го наблюдения:

$$G_t = \frac{\partial \ell_t}{\partial \boldsymbol{\theta}} = -\frac{1}{2}\frac{1}{\sigma_t^2}\frac{d\sigma_t^2}{d\boldsymbol{\theta}} + \frac{1}{2}\frac{\varepsilon_t^2}{\sigma_t^2}\frac{1}{\sigma_t^2}\frac{d\sigma_t^2}{d\boldsymbol{\theta}} - \frac{\varepsilon_t}{\sigma_t}\frac{1}{\sigma_t}\frac{d\varepsilon_t}{d\boldsymbol{\theta}}.$$

Здесь и в дальнейшем мы будем опускать аргумент $\boldsymbol{\theta}$.

Введем обозначения

$$\xi_t = \frac{\varepsilon_t}{\sigma_t},\ w_t = \xi_t^2 - 1,\ Q_t = \frac{1}{\sigma_t}\frac{d\varepsilon_t}{d\boldsymbol{\theta}},\ S_t = \frac{1}{\sigma_t^2}\frac{d\sigma_t^2}{d\boldsymbol{\theta}}.$$

В этих обозначениях вклад в градиент *t*-го наблюдения равен

$$G_t = \xi_t Q_t + \frac{1}{2} w_t S_t.$$

Поскольку условно по прошлой информации Ω_t вектора Q_t и S_t являются детерминированными, то

$$\bar{\Gamma}_t(\boldsymbol{\theta}) = \mathbf{E}(G_t^{\mathrm{T}} G_t \mid \Omega_t) = \mathbf{E}[(\xi_t Q_t + \frac{1}{2} w_t S_t)^{\mathrm{T}} (\xi_t Q_t + \frac{1}{2} w_t S_t) \mid \Omega_t].$$

Несложно показать, что

$$\mathbf{E}(\xi_t^2) = 1,\ \mathbf{E}(\xi_t w_t) = 0,\ \mathbf{E}(w_t^2) = 2.$$

Отсюда

$$\bar{\Gamma}_t(\boldsymbol{\theta}) = Q_t^{\mathrm{T}} Q_t + \frac{1}{2} S_t^{\mathrm{T}} S_t.$$

Ниже мы подробно разберем как вычислять градиенты $\frac{d\sigma_t^2}{d\boldsymbol{\theta}}$ и $\frac{d\varepsilon_t}{d\boldsymbol{\theta}}$, которые нужны для вычисления Q_t и S_t.

2.2.4.1.2 Искусственная регрессия

Эти преобразования показывают, что *r*-й шаг итерационного алгоритма имеет в данном случае вид

$$\boldsymbol{\Delta}^z = \boldsymbol{\theta}^{z+1} - \boldsymbol{\theta}^z \quad (\mathbf{Q}^{z\mathbf{T}}\mathbf{Q}^z + \frac{1}{2}\mathbf{S}^{z\mathbf{T}}\mathbf{S}^z)^{-1}(\mathbf{Q}^{z\mathbf{T}}\boldsymbol{\xi}^z + \frac{1}{2}\mathbf{S}^{z\mathbf{T}}\mathbf{w}^z),$$

где **Q**, **S**, $\boldsymbol{\xi}$, **w** — матрицы и вектора, составленные из Q_t, S_t, ξ_t и w_t соответственно. Шаг этого алгоритма, как несложно увидеть, можно вычислить с помощью следующей регрессии:

$$\begin{pmatrix} \boldsymbol{\xi} \\ \frac{1}{\sqrt{2}}\mathbf{w} \end{pmatrix} = \begin{bmatrix} \mathbf{Q} \\ \frac{1}{\sqrt{2}}\mathbf{S} \end{bmatrix} \boldsymbol{\Delta} + остатки\ .$$

Заметим также, что оценка ковариационной матрицы в этой регрессии, полученная на последней итерации (когда метод сошелся), является состоятельной оценкой ковариационной матрицы оценок максимального правдоподобия в GARCH-модели:

$$\overline{Var}(\bar{\boldsymbol{\theta}}) = \frac{\boldsymbol{\xi}^{\mathbf{T}}\boldsymbol{\xi} + \frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w}}{T}(\mathbf{Q}^{\mathbf{T}}\mathbf{Q} + \frac{1}{2}\mathbf{S}^{\mathbf{T}}\mathbf{S})^{-1}.$$

Таким образом, эта регрессия представляет собой то, что называется *искусственной регрессией*.

В качестве критерия остановки можно взять нецентральный коэффициент детерминации из искусственной регрессии:

$$R^2_{nc} = 1 - \frac{сумма\ |\ квадратов\ |\ остатков}{сумма\ |\ квадратов\ |\ зависимой\ |\ переменной}.$$

Можно считать, что метод сошелся, если регрессия практически не объясняет зависимую переменную, т. е. если величина R^2_{nc} мала ($R^2_{nc} < \phi$). На практике неплохо работает правило остановки с $\phi = 10^{-11}$.

2.2.1.1.3 Вычисление градиентов остатков

Для полного описания алгоритма требуется указать способ вычисления градиентов $\frac{d\sigma_t^2}{d\boldsymbol{\theta}}$ и $\frac{d\varepsilon_t}{d\boldsymbol{\theta}}$.

Поскольку по определению

$$\varepsilon_t(\boldsymbol{\theta}) = r_t - X_t\boldsymbol{\beta},$$

то

$$\frac{d\varepsilon_t}{d\boldsymbol{\theta}} = (X_t, \mathbf{0}^{\mathrm{T}}).$$

Если регрессоры отсутствуют, то есть, если r_t — это непосредственно GARCH-процесс ε_t с нулевым математическим ожиданием, то градиент равен нулю и, соответственно, $Q_t = \mathbf{0}^{\mathrm{T}}$. При этом искусственная регрессия упрощается до регрессии $\frac{1}{\sqrt{2}}\mathbf{w}$ на $\frac{1}{\sqrt{2}}\mathbf{S}$ (множитель $\frac{1}{\sqrt{2}}$ не влияет на оценки регрессии, но должен входить в ковариационную матрицу, поэтому его лучше не отбрасывать).

2.2.4.1.4 Вычисление градиентов условных дисперсий

Обозначим $\frac{d\sigma_t^2}{d\boldsymbol{\theta}} = H_t$.

Формулу для этих величин получим, дифференцируя по $\boldsymbol{\theta}$ соотношение (2).

При $t = \min\{1, q+p-1\}, \ldots, q$ вектор-строка H_t, будет состоять из нулей, только на месте параметра ζ_t будет стоять 1. При $t > q$ вектор H_t вычисляется рекуррентно:

$$H_t = \frac{d\sigma_t^2}{d\boldsymbol{\theta}} + 2\sum_{j=1}^{q}\gamma_j\varepsilon_{t-j}\frac{d\varepsilon_{t-j}}{d\boldsymbol{\theta}} + \sum_{j=1}^{p}\delta_t H_{t-j}, t \quad q+1,\ldots,T.$$

Здесь $\frac{d\sigma_t^2}{d\boldsymbol{\theta}}$ — непосредственная производная условной дисперсии по параметрам:

$$\qquad\qquad \boldsymbol{\beta} \quad \omega \qquad \boldsymbol{\gamma} \qquad\qquad \boldsymbol{\delta} \qquad \boldsymbol{\zeta}$$

$$\frac{d\sigma_t^2}{d\boldsymbol{\theta}} = (\mathbf{0}^{\mathbf{T}}, 1, \varepsilon_{t-1}^2, ..., \varepsilon_{t-p}^2, \sigma_{t-1}^2, ..., \sigma_{t-q}^2, \mathbf{0}^{\mathbf{T}}).$$

Сверху в формуле подписаны параметры, по которым берется производная.

2.2.4.1.5 Алгоритм вычислений

Сначала вычисляются начальные приближения параметров. Следует следить, чтобы они оказались в допустимой области, т. е. вычисленные на их основе условные дисперсии все были положительными.

После этого идут итерации:

1) Вычислить остатки $\varepsilon_t(\boldsymbol{\theta}) = r_t - X_t \boldsymbol{\beta}$.
2) Вычислить условные дисперсии σ_t^2 по рекуррентной формуле (2).
3) Вычислить градиенты остатков $\frac{d\varepsilon_t}{d\boldsymbol{\theta}}$
4) Вычислить градиенты условных дисперсий H_t.
5) Вычислить **Q**, **S**, ξ и **w**.
6) Оценить искусственную регрессию и получить направление изменения параметров $\boldsymbol{\Delta}$.
7) Проверить выполнение правила остановки ($R_{nc}^2 < \phi$).
8) Выбрать коэффициент λ так, чтобы новое значение параметров $\boldsymbol{\theta}^{z+1} = \boldsymbol{\theta}^z + \lambda \boldsymbol{\Delta}^z$ оказалось в допустимой области и давало больший уровень функции правдоподобия, чем $\boldsymbol{\theta}^z$, т. е. $\ell(\boldsymbol{\theta}^{z+1}) > \ell(\boldsymbol{\theta}^z)$.

   Начать новую итерацию.

После того, как итеративный алгоритм сойдется, на основе ковариационной матрицы из искусственной регрессии вычисляются стандартные ошибки параметров и t-статистики по обычным формулам.

2.2.4.2 Рандомизированный алгоритм стохастической аппроксимации.

2.2.4.2.1 Алгоритм вычислений

Для упрощения алгоритма заменим матрицу $(\overline{\Gamma}_t(\boldsymbol{\theta}))^{-1}$ на положительное число φ_n, получим алгоритм типа процедуры Робинсона-Монро. Градиент функции правдоподобия аппроксимируем с помощью $\Delta_n \frac{\ell_n(\boldsymbol{\theta}^{n-1}+\psi_n\Delta_n)-\ell_n(\boldsymbol{\theta}^{n-1}-\psi_n\Delta_n)}{2\psi_n}$, Δ_n – рандомизированный вектор направления изменения оценок, координаты которого являются реализацией независимых друг от друга случайных величин, принимающих с равной вероятностью значения +/-1. Таким образом, для построения последовательности оценок мы будем использовать рандомизированный алгоритм стохастической аппроксимации [7].

Вычисляя начальные приближения, следует следить за тем, чтобы они оказались в допустимой области, т. е. вычисленные на их основе условные дисперсии все были положительными.

В качестве φ_n рассмотрим последовательность $\frac{1}{n}$, $\psi_n = \frac{1}{\sqrt{n}}$, что удовлетворяет общим условиям $\sum_n \varphi_n = \infty$, $\sum_n \varphi_n^2 < \infty$, $\varphi_n \xrightarrow[n\to\infty]{} 0$, $\psi_n \xrightarrow[n\to\infty]{} 0$. Также определим проекторы $P_{\boldsymbol{\theta}}^{\,1}$ и $P_{\boldsymbol{\theta}}^{\,2}$, обеспечивающие попадание векторов в допустимую область.

Итерации:

1) Вычислить остатки $\varepsilon_t(\boldsymbol{\theta}) = r_t - X_t\boldsymbol{\beta}$, t=1,…,T
2) Вычислить условные дисперсии σ_t^2 по рекуррентной формуле (2).
3) Случайным образом сгенерировать вектора Δ_n.

4) Скорректировать вектор Δ_n так, чтобы $\boldsymbol{\theta}^{n-1}+\psi_n\Delta_n$ и $\boldsymbol{\theta}^{n-1}-\psi_n\Delta_n$ попали в допустимую область (найти $P_{\boldsymbol{\theta}}^{1}$)

5) Вычислить следующие величины:

$$z_n^+ = P_{\boldsymbol{\theta}}^{1}(\boldsymbol{\theta}^{n-1}+\psi_n\Delta_n) \quad \ell_n^+ = \ell(z_n^+)$$
$$z_n^- = P_{\boldsymbol{\theta}}^{1}(\boldsymbol{\theta}^{n-1}-\psi_n\Delta_n)\text{,} \quad \ell_n^+ = \ell(z_n^-)$$

6) Скорректировать вектор Δ_n так, чтобы $(\boldsymbol{\theta}^{n-1}-\Delta_n\varphi_n\frac{\ell_n^+-\ell_n^-}{2\psi_n})$ попал в допустимую область (найти $P_{\boldsymbol{\theta}}^{2}$)

7) Вычисляется оценка $\boldsymbol{\theta}^n = P_{\boldsymbol{\theta}}^{2}(\boldsymbol{\theta}^{n-1}-\Delta_n\varphi_n\frac{\ell_n^+-\ell_n^-}{2\psi_n})$.

8) Проверяем правило остановки $\|\boldsymbol{\theta}^n-\boldsymbol{\theta}^{n-1}\|<\phi$, ϕ – требуемая точность оценки.

Если $\|\boldsymbol{\theta}^n-\boldsymbol{\theta}^{n-1}\|^2<\phi$, то оценка параметров $\bar{\boldsymbol{\theta}}=\boldsymbol{\theta}^n$, иначе начинаем новую итерацию.

После того, как алгоритм сойдется, на основе ковариационной матрицы из искусственной регрессии вычисляются стандартные ошибки параметров и t-статистики по обычным формулам.

2.2.4.3. Сравнение алгоритмов

Сравнение алгоритмов было проведено с помощью MatLab, код программы, позволяющей сравнивать алгоритмы, можно найти в приложении 1. Были сгенерированы процессы с помощью встроенной функции garchsim и проведены оценки параметров для полученных рядов двумя способами – с помощью рандомизированного алгоритма стохастической оптимизации (функция Param) и градиентного алгоритма, реализованного с помощью встроенной функции garchfit.

Несложно заметить, что итерация алгоритма, использующего метод внешнего произведения градиентов, имеет большую вычислительную сложность, чем итерация рандомизированного алгоритма стохастической аппроксимации. Однако в среднем алгоритм, использующий метод внешнего произведения градиентов, сходится за 20 итераций, рандомизированный алгоритм стохастической аппроксимации же за 150-2000. Таким образом, за счёт того, что рандомизированный алгоритм стохастической аппроксимации требует большее количество итераций, однако имеет меньшую вычислительную сложность итерации, время работы алгоритмов примерно одинаково (10-20 секунд – в зависимости от требуемой точности и размерности вектора $\boldsymbol{\theta}$). Максимальные значения функции правдоподобия, получаемые с помощью алгоритмов, различаются максимум на $6*10^{-1}$. Однако следует заметить, что рандомизированный алгоритм стохастической аппроксимации теоретически будет сходиться и для процесса с почти произвольными помехами, тогда как алгоритм, основанный на методе внешнего произведения градиентов, сходится только для помех, представляющих собой процесс, удовлетворяющий теореме Гаусса-Маркова.

В разрабатываемом в рамках дипломной работы аналитическом модуле для оценки коэффициентов GARCH-модели был выбран рандомизированный алгоритм стохастической аппроксимации.

2.3 Прогнозы и оценка доверительных интервалов для GARCH-модели.

Наличие условной гетерокседаксичности позволяет найти более эффективные (т. е. более точные) оценки среди нелинейных и смещённых оценок, чем оценки метода наименьших квадратов. Действительно, метод максимального правдоподобия даёт асимптотически эффективные оценки более точные, чем МНК. Когда мы делаем прогноз на следующий (T+1) период, то в ошибку прогноза вносит свой вклад, во-первых, ошибка ε_{T+1}, а, во-вторых, разница между оценками параметров и истинными значениями

параметров. Использование более точных оценок позволяет уменьшить в некоторой степени вторую составляющую ошибки прогноза.

Как уже было замечено, сами по себе прогнозы условной дисперсии, т. е. волатильности, могут иметь практическое применение как меры рискованности финансового актива, например, при принятии решений об инвестициях в финансовые активы.

Далее покажем, что доверительный интервал прогноза зависит от предыстории $\Omega_T = (r_T, r_{T-1}, ..., r_1, ...)$.

Пусть нам известны истинные параметры процесса. Прогноз на *m* периодов – это матожидание прогнозируемой величины r_{T+m}, условное относительно имеющейся на момент *T* информации Ω_T. Он равен

$$r_{T+m}^p = E(r_{T+m} \mid \Omega_T) = E(X_{T+m}\beta + \varepsilon_{T+m} \mid \Omega_T) = X_{T+m}\beta .$$

Здесь мы учли, что, поскольку информация Ω_T содержится в информации Ω_{T+m-1} при $m \geq 1$, то по правилу повторного взятия ожидания выполнено:

$$E(\varepsilon_{T+m} \mid \Omega_T) = E(E(\varepsilon_{T+m} \mid \Omega_{T+m-1}) \mid \Omega_T) = 0 .$$

Таким образом, при известных истинных параметрах присутствие GARCH-ошибок не отражается на том, как строится точечный прогноз, – он оказывается таким же, как для обычной линейной регрессии.

Ошибка предсказания равна:

$$d_m = r_{T+m} - r_{T+m}^p = \varepsilon_{T+m} .$$

Условная дисперсия ошибки предсказания равна:

$$\sigma_{d_m}^2 = E(d_m^2 \mid \Omega_T) \quad E(\varepsilon_{T+m}^2 \mid \Omega_T) .$$

То есть она зависит как от горизонта прогноза *m*, так и от предыстории Ω_T.

Заметим, что при $t>T$ выполнено $E(\varepsilon_t^2 \mid \Omega_T) \quad E(\sigma_t^2 \mid \Omega_T)$, т. к. $E(\varepsilon_t^2 - \sigma_t^2 \mid \Omega_T) = E(E(\varepsilon_t^2 - \sigma_t^2 \mid \Omega_{T-1}) \mid \Omega_T) = 0$ (правило повторного

взятия матожидания с учётом того, что $E(\varepsilon_t^2 - \sigma_t^2 \mid \Omega_{T-1}) = 0$, информация Ω_T содержится в информации Ω_{t-1} при $t>T$)

Значит, $\sigma_{d_m}^2 = E(\varepsilon_{T+m}^2 \mid \Omega_T) \quad E(\sigma_{T+m}^2 \mid \Omega_T)$.

Таким образом, фактически дисперсия прогноза r_{T+m} – это прогноз волатильности на m шагов вперёд.

Возьмём от обеих частей рекуррентного уравнения для GARCH-процесса матожидание, условное относительно Ω_T. Получим

$$E(\sigma_t^2 \mid \Omega_T) \quad \omega + \sum_{j=1}^{p} \delta_j E(\sigma_{t-j}^2 \mid \Omega_T) + \sum_{j=1}^{q} \gamma_j E(\varepsilon_{t-j}^2 \mid \Omega_T) \quad (3)$$

Можно использовать эту формулу для расчёта $E(\sigma_t^2 \mid \Omega_T)$ при $t>T$. При этом следует учесть, что $E(\varepsilon_t^2 \mid \Omega_T) = \varepsilon_t^2$ при $t \leq T$, поскольку информация о ε_t содержится в Ω_T, и по аналогичной причине $E(\sigma_t^2 \mid \Omega_T) = \sigma_t^2$ при $t \leq T+1$. Кроме того, мы доказали $E(\varepsilon_t^2 \mid \Omega_T) \quad E(\sigma_t^2 \mid \Omega_T)$ при $t>T$.

Таким образом, имеются все данные для того, чтобы с помощью формулы рассчитать дисперсию ошибки прогноза для r_{T+m} в модели GARCH (При m=1 можно сразу записать $\sigma_{d_1}^2 = E(\sigma_{T+1}^2 \mid \Omega_T) = \sigma_{T+1}^2$).

Чтобы обойти проблему, связанную с тем, что d_m имеет более толстые хвосты (т. к. условное относительно Ω_T распределение ε_{T+m} имеет более толстые хвосты), чем нормальное распределение, можно использовать прогнозные интервалы в виде плюс/минус двух среднеквадратических ошибок прогноза без выяснения того, какой именно доверительной вероятности это соответствует (для нормального распределения это примерно 95% двусторонний квантиль).

Глава 3. Аналитический модуль, осуществляющий прогнозирование доходности

MetaEco представляет собой программно-аппаратный комплекс, кластер базы данных которого функционирует на основе Oracle и Oracle Application Server. Основными задачами комплекса является:

- накопление всевозможной финансовой (биржевой) информации;
- анализ;
- принятие решений;

На базе платформы функционирует автоматизированная торговая система, осуществляющая следующие действия:

- принятие решения в режиме реального времени;
- предоставление конечному пользователю сложной аналитической информации на основе входных данных (например, биржевых индикаторов);

В рамках дальнейшего развития системы предполагается включить в неё систему прогнозирования, предоставляющую информацию пользователю, а также системе принятия решений.

3.1 Описание реализации

Аналитический модуль, являющийся целью дипломной работы, предполагается включить в аналитическую часть системы. Этим модулем сможет пользоваться система принятия решений, а также конечные пользователи. Модуль позволяет осуществлять прогнозы, основанные на GARCH-модели волатильности для оценки ошибки прогноза и ARMA-модели для построения условного математического ожидания. Входными данными для модуля являются:

- ✓ ряд цен $P_t, t = 1,...,T$
- ✓ горизонт прогнозирования m
- ✓ требуемый порядок точности прогноза ϕ

- ✓ максимальные величины порядков моделей ARMA и GARCH *k_max, p_max, q_max*

Выходные данные:

- ✓ $r_t, t = T+1,...,T+m$ – прогноз доходности выбранного финансового инструмента
- ✓ доверительный интервал прогноза

Аналитический модуль состоит из набора аналитических функций, взаимосвязанных между собой и выполняющих следующие задачи:

- построение критерия Шварца для модели ARMA;
- построение критерия Шварца для модели GARCH;
- оценка параметров модели GARCH методом максимального правдоподобия с использованием рандмизированного алгоритма стохастической аппроксимации;
- построение прогноза доходности на *m* периодов;

Аналитическая система разработана на основе СУБД Oracle. Для реализации модуля был выбран язык PL/SQL, поскольку он позволяет находиться максимально близко к данным и осуществлять реализацию с минимальным временем отклика.

Код реализованного модуля можно увидеть в приложении 2.

3.2 Пример работы аналитического модуля

Для апробации работы аналитического модуля были выбраны данные временного ряда цен закрытия для акций АО «ВТБ» на период со 2 марта 2009 года по 1 июня 2009 года.

На вход разработанного аналитического модуля были поданы следующие значения: *T*=63, *m*=4, *k_max*=2, *p_max*=*q_max*=2, $\phi = 10^{-7}$, а также представленный в табл. 1 ряд котировок $P_t, t = 1,...,T$.

Таблица 1

АО ВТБ

Дата	Цена закрытия	Дата	Цена закрытия	Дата	Цена закрытия	Дата	Цена закрытия
2009.03.02	0	2009.03.25	0,0911148	2009.04.16	0,0392207	2009.05.12	0,0957451
2009.03.03	0,025318	2009.03.26	-0,036701	2009.04.17	0,0291566	2009.05.13	-0,081493
2009.03.04	0,067659	2009.03.27	-0,057708	2009.04.20	-0,077651	2009.05.14	-0,068319
2009.03.05	0	2009.03.30	-0,104261	2009.04.21	-0,031548	2009.05.15	0,0347429
2009.03.06	0,032187	2009.03.31	0,0253178	2009.04.22	0,0284379	2009.05.18	0,063776
2009.03.10	0,150913	2009.04.01	0,0281709	2009.04.23	0,0215725	2009.05.19	0,0158914
2009.03.11	0,019268	2009.04.02	0,0638873	2009.04.24	-0,018462	2009.05.20	0,0505047
2009.03.12	-0,019268	2009.04.03	-0,026404	2009.04.27	-0,0125	2009.05.21	-0,084872
2009.03.13	0	2009.04.06	-0,010084	2009.04.28	-0,035204	2009.05.22	0,029853
2009.03.16	0,053043	2009.04.07	0,0167508	2009.04.29	0,032054	2009.05.25	0,0022599
2009.03.17	0,011009	2009.04.08	0,0197375	2009.04.30	-0,006329	2009.05.26	-0,034447
2009.03.18	-0,064052	2009.04.09	0,0477038	2009.05.04	0,04652	2009.05.27	0,0069849
2009.03.19	0,064052	2009.04.10	-0,00936	2009.05.05	0,0060423	2009.05.28	0,0138251
2009.03.20	0,073847	2009.04.13	-0,009449	2009.05.06	0,0208651	2009.05.29	0,0203858
2009.03.23	0,09685	2009.04.14	0,0553988	2009.05.07	0,0628779	2009.06.01	0,0858897
2009.03.24	-0,066797	2009.04.15	-0,027316	2009.05.08	0,1466035	-	-

По входным данным несложно рассчитать логарифмическую доходность по формуле:

$$r_t = \ln(\frac{P_t}{P_{t-1}}).$$

График доходности представлен на рис. 1.

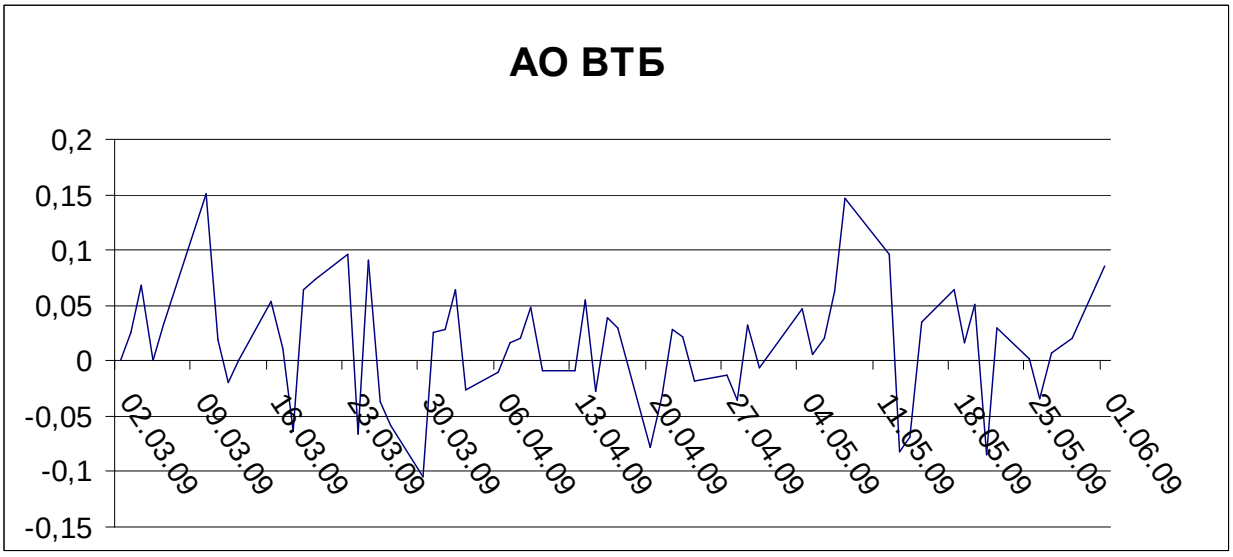

Рис. 1

На выходе аналитического модуля были получены: прогноз доходности $r_t, t = T+1,...,T+m$ на m периодов, верхняя и нижняя границы доверительного интервала для прогноза. Данные были выгружены в MS Excel, на их основании был построен следующий график прогноза доходности, представленный на рис. 2:

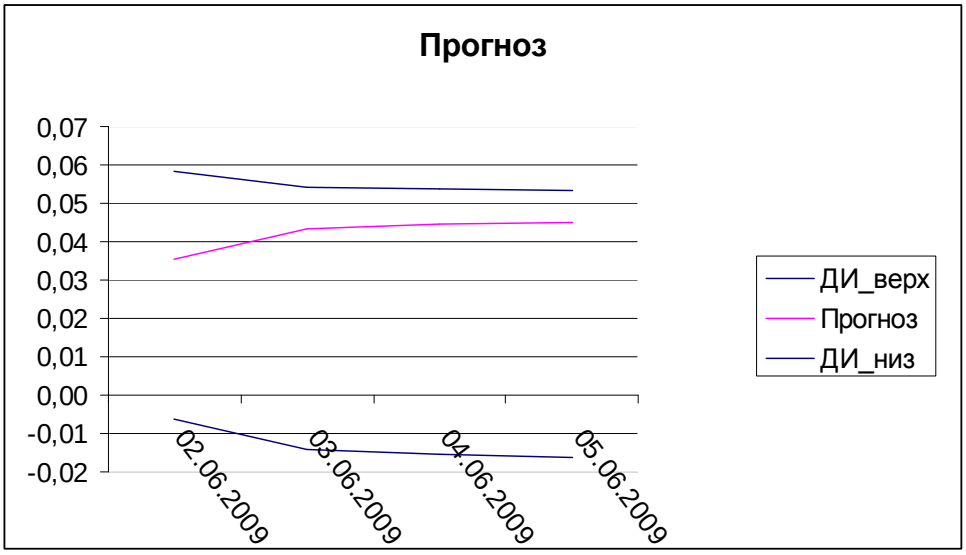

Рис. 2

Заключение

1) В ходе дипломной работы была показана значимость решения проблем неопределённости и риска для социологии. В качестве социального института был рассмотрен институт фондового рынка, проблема неопределённости и характерные риски в рамках этого института. В качестве меры риска была рассмотрена волатильность финансовых инструментов, используемая также для осуществления прогнозирования, которое в свою очередь тоже является способом решения проблемы неопределённости. Были исследованы различные способы расчёта волатильности и выбрана GARCH-модель расчёта волатильности в сочетании с ARMA-моделью, в силу того, что эта модель учитывает такие свойства волатильности, как тяжёлые хвосты и кластеризация.
2) Был предложен рандомизированный алгоритм для оценки параметров GARCH-модели в сочетании с моделью ARMA дающим асимптотически точные и несмещённые оценки методом максимального правдоподобия. Был проведён сравнительный анализ работоспособности предложенного алгоритма и стандартного метода внешнего произведения градиентов, и для решения поставленной задачи был выбран рандомизированный алгоритм стохастической аппроксимации. В качестве критерия выбора порядка моделей был выбран критерий Шварца. Был реализован метод построения прогноза и доверительного интервала для прогноза, основанный на GARCH-модели в сочетании с ARMA, полученной в результате оценки порядка GARCH и ARMA с помощью критерия Шварца и оценки параметров модели методом максимального правдоподобия с использованием рандомизированного алгоритма стохастической аппроксимации.

3) Разработан аналитический модуль, позволяющий осуществлять прогнозирование доходности финансовых инструментов и строить доверительный интервал для полученного прогноза, который в дальнейшем предполагается включить в аналитическую часть системы программно-аппаратного комплекса MetaEco в рамках проекта LiveTrade Online.

Список использованной литературы

1. Агаев И.А., Куперин Ю.А. Инструментальные методы экономики: Нелинейное моделирование статистических свойств доходностей финансовых инструментов // Управление экономическими системами. 2006. №4(8) / http://uecs.mcnip.ru/
2. Барышников И.В., Голембиовский Д.Ю. Ценообразование опционов: «улыбка» по-русски // Управление риском. 2004. №3(31). с. 44-49.
3. Бригхем Ю., Гопенски Л. Финансовый менеджмент. Том 1. – СПб.: Экономическая школа, 1997 – 497 с.
4. Бригхем Ю., Гопенски Л. Финансовый менеджмент. Том 2. – СПб.: Экономическая школа, 1997 – 668 с.
5. Бруссер П.А. Волатильность и информационная эффективность фондового рынка // Вестник СПбГУ. 2004. №21. с. 144-148.
6. Глоссарий / http://www.glossary.ru
7. Граничин О.Н., Поляк Б.Т. Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах. – М.: Наука, 2003 – 290 с.
8. Григорьева И.Л., Филиппов Л.А. Оценка экономического риска // Управление риском. 2001. №3. с. 6-12.
9. Егорова Е.Е. Ещё раз о сущности риска и системном подходе // Управление риском. 2002. №2. с. 9-12.
10. Канторович Г.Г. Анализ временных рядов // Экономический журнал ВШЭ. 2002. Том 6. №1. с. 85-117.
11. Канторович Г.Г. Анализ временных рядов // Экономический журнал ВШЭ. 2002. Том 6. №2. с. 251-273.
12. Канторович Г.Г. Анализ временных рядов // Экономический журнал ВШЭ. 2002. Том 6. №3. с. 379-401.

13. Корников В.В., Серёгин И.А., Хованов Н.В. Многокритериальное оценивание финансовых рисков в условиях неопределённости. – СПб.: Издательство С.-Петербургского университета, 2002 – 93 с.
14. Крутякова Ю. Время риска и риски времени // РИСК. 2006. №4. с. 46-48.
15. Кузина О.Е. Критерии финансовой грамотности населения и пути её повышения / http://www.isras.ru/abstract_bank/1209884227.pdf
16. Кулаков А.Е. Волатильность доходности и подход к построению системы контроля и управления рисками // Банковское дело. 2004. №6. с. 35-38.
17. Кулаков А.Е. Волатильность доходности как интегральный показатель риска // Финансы и кредит. 2004. № 16(154). с. 25-30.
18. Материалы по GARCH-моделям / http://www.nsu.ru/ef/tsy/ecmr/garch/index.htm
19. МакЛафлин М., Урдман С., Хардман Р. Oracle Database 10g. Программирование на языке PL/SQL. – СПб: Питер, 2007 – 816 с.
20. Пустовит Д., Подколзина И. История биржевой торговли. – СПб.: Вектор. 2009 – 256 с.
21. Радаев В.В. Современные экономико-социологические концепции рынка // Экономическая социология. 2008. №1. с. 20-50.
22. Рогов М.А. Риск-менеджмент – М.: Финансы и статистика, 2001 – 118 с.
23. Сенько В. Меняющийся подход к риск-менеджменту в крупных компаниях // Управление риском. 2001. №3. с. 3-5.
24. Середа А.Ю Оценка VaR портфеля ценных бумаг с применением ARCH-моделей // Финансовый менеджмент. 2008. № 16(304). с. 16-21.

25. Стахович Л.В. Формирование финансовой грамотности населения в сфере финансовых рынков: анализ международного опыта // Финансы и кредит. 2008. №16 (304). с. 67-71.
26. Флигстин Н. Рынки как политика: политико-культурный подход к рыночным институтам // Экономическая социология. 2003. №1. с. 45-63.
27. Хованов Н.В. Математические модели риска и неопределённости. – СПб.: Издательство С.-Петербургского университета, 1998 – 199 с.
28. Чекулаев М. Волатильность умом не понять // Валютный спекулянт. 2004. № 07(57). с. 22-25.
29. Четыркин Е.М. Финансовые риски. – М.: Дело, 2008 – 175 с.
30. Ito formula Kyrwiki / http://www.theponytail.net/wiki/pmwiki.php/Main/ItoFormula
31. Sekvester's HomePage / http://sekvestr-trade.narod.ru/
32. Trade Station Securities / http://www.omegaresearch.com/default_2.shtm

Приложения.

Приложение 1

```
spec=garchset('R',1,'C',0.5,'AR',0.25,'P', 2, 'Q', 2, 'K', 0.5, 'GARCH',[0.05
0.025],'ARCH',[0.01 0.025])
r=garchsim(spec, 10)
garchfit(spec, r)
Param

function Param

format long e

T = 10;
k = 1;
p = 2;
q = 2;


r = [2.6428, 2.1682, -1.1814, 2.5120, 0.5855, -0.0532, 0.5351, -0.1518, -
0.0920, 1.0965];

for u = 1:1:T-k
    for u1 = 1:1:k
        X(u, u1+1) = r(k+u-u1);
    end
end

for u = 1:1:T-k
    X(u, 1) = 1;
end

for u = 1:1:T-k
    r1(u) = r(u);
end

betta = (((X.'*X)^-1)*X.')*r1.';

summa1 = 0;
summa2 = 0;
for u = 1:1:k
    summa1 = summa1 + betta(u+1);
end

for u = 1:1:T
    if(u <= k)
        epsilon(u) = r(u)-betta(1)/(1-summa1);
    end

    if(u > k)
        for u2 = 1:1:k
            summa2 = summa2 + betta(u2+1)*r(u-u2);
        end
        epsilon(u) = r(u)-(betta(1)+summa2);
        summa2 = 0;
    end
end
```

```
for u = 1:1:q
    gamma(u) = 1/(4*q);
end

for u = 1:1:p
    delta(u) = 1/(4*p);
end

omega = 0;

for u = 1:1:T
    omega = omega + (epsilon(u)^2)*(1-sum(gamma)-sum(delta));
end

omega = (1/T)*omega;


summa3 = 0;

for u = 1:1:T
    summa3 = summa3 + epsilon(u)^2;
end


summa4 = 0;
summa5 = 0;

for u = 1:1:T
    if(u <= p)
        sigma_kvadrat(u) = (1/T)*summa3;
    end

    if(u > p)
        for u2 = 1:1:p
            summa4 = summa4 + delta(u2)*sigma_kvadrat(u-u2);
        end
        for u3 = 1:1:q
            summa5 = summa5 + gamma(u3)*((epsilon(u-u3))^2);
        end

        sigma_kvadrat(u) = omega + summa4 + summa5;
        summa4 = 0;
        summa5 = 0;
    end
end

for u = 2:1:p+1
    sigma1(u-1) = sigma_kvadrat(u);
end

tetta = [betta; omega; gamma.'; delta.'; sigma1.'];

N = 1;
j = 1;

%--------------------------------------------------

while j == 1
```

```
    Dn = randint(k+2*p+q+2,1);

    for u = 1:1:(k+2*p+q+2)
        if(Dn(u) == 0)
            Dn(u) = -1;
        end
    end

    Dn = Dn/sqrt(k+2*p+q+2);

    fi = 1/N;

    psi = 1/sqrt(N);

    change = 1;

    while change == 1

        Zplus = tetta + psi*Dn;
        Zminus = tetta - psi*Dn;

        change = 0;

        for u = 1:1:p %delta
            if(Zplus(k+q+2+u) < 0)
                Dn(k+q+2+u) = tetta(k+q+2+u)/2;
                change = 1;
            end
            if(Zminus(k+q+2+u) < 0)
                Dn(k+q+2+u) = tetta(k+q+2+u)/2;
                change = 1;
            end
        end

        for u = 1:1:q %gamma
            if(Zplus(k+2+u) < 0)
                Dn(k+2+u) = tetta(k+2+u)/2;
                change = 1;
            end

            if(Zminus(k+2+u) < 0)
                Dn(k+2+u) = tetta(k+2+u)/2;
                change = 1;
            end
        end

%omega
        if(Zplus(k+2) < 0)
            Dn(k+2) = tetta(k+2)/2;
            change = 1;
        end
        if(Zminus(k+2) < 0)
            Dn(k+2) = tetta(k+2)/2;
            change = 1;
        end

        for u = 1:1:p %sigma
            if(Zplus(k+p+q+2+u) < 0)
                Dn(k+p+q+2+u) = tetta(k+p+q+2+u)/2;
                change = 1;
            end
```

```
            if(Zminus(k+p+q+2+u) < 0)
                Dn(k+p+q+2+u) = tetta(k+p+q+2+u)/2;
                change = 1;
            end
        end
    end

    summa1 = 0;
    summa2 = 0;
    summa3 = 0;
    summa4 = 0;
    for u = 1:1:k
        summa1 = summa1 + Zplus(u+1);
        summa3 = summa3 + Zminus(u+1);
    end

    for u = 1:1:T
        if(u <= k)
            epsilonZplus(u) = r(u)-Zplus(1)/(1-summa1);
           epsilonZminus(u) = r(u)-Zminus(1)/(1-summa3);
        end

       if(u > k)
            for u2 = 1:1:k
                summa2 = summa2 + Zplus(u2+1)*r(u-u2);
                summa4 = summa4 + Zminus(u2+1)*r(u-u2);
            end
            epsilonZplus(u) = r(u)-(Zplus(1)+summa2);
            epsilonZminus(u) = r(u)-(Zminus(1)+summa4);
            summa2 = 0;
            summa4 = 0;
       end
    end

    summa3P = 0;
    summa3M = 0;
    for u = 1:1:T
        summa3P = summa3P + epsilonZplus(u)^2;
        summa3M = summa3M + epsilonZminus(u)^2;
    end

    summa4P = 0;
    summa5P = 0;
    summa4M = 0;
    summa5M = 0;

    for u = 1:1:T
        if(u <= p)
           sigma_kvadratZplus(u) = (1/T)*summa3P;
           sigma_kvadratZminus(u) = (1/T)*summa3M;
        end

        if(u > p)
            for u2 = 1:1:p
                summa4P = summa4P + Zplus(k+q+2+u2)*sigma_kvadratZplus(u-u2);
                summa4M = summa4M + Zminus(k+q+2+u2)*sigma_kvadratZminus(u-
u2);
            end
            for u3 = 1:1:q
                summa5P = summa5P + Zplus(k+2+u3)*(epsilonZplus(u-u3)^2);
                summa5M = summa5M + Zminus(k+2+u3)*(epsilonZminus(u-u3)^2);
            end
```

```
            sigma_kvadratZplus(u) = Zplus(k+2) + summa4P + summa5P;
            sigma_kvadratZminus(u) = Zminus(k+2) + summa4M + summa5M;

            summa4P = 0;
            summa5P = 0;
            summa4M = 0;
            summa5M = 0;
        end
    end

    summaTP = 0;
    summaTM = 0;

    for u = 1:1:T
        summaTP = summaTP + log(sigma_kvadratZplus(u)) +
[epsilonZplus(u)^2]/sigma_kvadratZplus(u);
        summaTM = summaTM + log(sigma_kvadratZminus(u)) +
[epsilonZminus(u)^2]/sigma_kvadratZminus(u);
    end

    Lplus = -0.5*(T*log(2*pi)+summaTP);
    Lminus = -0.5*(T*log(2*pi)+summaTM);

    change2 = 1;

    sum1 = 0;
    sum2 = 0;
    sum3 = 0;
    sum4 = 0;

    while change2 == 1;

        tettaN = tetta-Dn*fi*[ (Lplus - Lminus)/(2*psi) ];

        change2 = 0;

        for u = 1:1:p %delta
            sum1 = sum1 + tettaN(k+q+2+u);
            sum3 = sum3 + tetta(k+q+2+u);
            Ma(u) = tettaN(k+q+2+u);
            if(tettaN(k+q+2+u) < 0)
                Dn(k+q+2+u) = tetta(k+q+2+u)/2;
                change2 = 1;
            end
        end

        for u = 1:1:q %gamma
            sum2 = sum2 + tettaN(k+2+u);
            sum4 = sum4 + tetta(k+2+u);
            Ma(p+u) = tettaN(k+2+u);
            if(tettaN(k+2+u) < 0)
                Dn(k+2+u) = tetta(k+2+u)/2;
                change2 = 1;
            end
        end

        %omega
        if(tettaN(k+2) < 0)
            Dn(k+2) = tetta(k+2)/2;
            change2 = 1;
```

```
    end

    for u = 1:1:q %sigma
        if(tettaN(k+p+q+2+u) < 0)
            Dn(k+p+q+2+u) = tetta(k+p+q+2+u)/2;
            change2 = 1;
        end
    end

    if(sum1 + sum2 > 1)

        for u = k+2+1:1:k+q+2+p
            Dn(u) = (1-sum3-sum4)/(p+q+1);
        end
        change2 = 1;
    end
    sum1 = 0;
    sum2 = 0;
    sum3 = 0;
    sum4 = 0;
end

%norma raznosti

j = 0;

for u = 1:1:(k+2*p+q+2)
    if(sqrt(tettaN(u)*tettaN(u)-tetta(u)*tetta(u))>10^(-3))
        j = 1;
    end
end

summa1 = 0;
summa2 = 0;
for u = 1:1:k
    summa1 = summa1 + tettaN(u+1);
end

for u = 1:1:T
    if(u <= k)
        epsilontettaN(u) = r(u)-tettaN(1)/(1-summa1);
    end

    if(u > k)
        for u2 = 1:1:k
            summa2 = summa2 + tettaN(u2+1)*r(u-u2);
        end
        epsilontettaN(u) = r(u)-(tettaN(1)+summa2);
        summa2 = 0;
    end
end

summatettaN = 0;
for u = 1:1:T
    summatettaN = summatettaN + epsilontettaN(u)^2;
end

summatettaN1 = 0;
summatettaN2 = 0;

for u = 1:1:T
```

```
        if(u <= p)
            sigma_kvadrattettaN(u) = (1/T)*summatettaN;
        end

        if(u > p)
            for u2 = 1:1:p
                summatettaN1 = summatettaN1 +
tettaN(k+q+2+u2)*sigma_kvadrattettaN(u-u2);
            end
            for u3 = 1:1:q
                summatettaN2 = summatettaN2 +
tettaN(k+2+u3)*(epsilontettaN(u-u2)^2);
            end
            sigma_kvadrattettaN(u) = tettaN(k+2) + summatettaN1 +
summatettaN2;
            summatettaN1 = 0;
            summatettaN2 = 0;
        end
    end

    summaLtetta = 0;
    for u = 1:1:T
        summaLtetta = summaLtetta + log(sigma_kvadrat(u)) +
[epsilon(u)^2]/sigma_kvadrat(u);
    end

    Ltetta = -0.5*(T*log(2*pi)+summaLtetta);

    summaLtettaN = 0;

    for u = 1:1:T
        summaLtettaN = summaLtettaN + log(sigma_kvadrattettaN(u)) +
[epsilontettaN(u)^2]/sigma_kvadrattettaN(u);
    end

    LtettaN = -0.5*(T*log(2*pi)+summaLtettaN);

    if(LtettaN > Ltetta)
        tetta = tettaN;
        N = N + 1;
    end

    if(LtettaN <= Ltetta)
        tetta = tetta;
    end
end

N
tetta
Ltetta
```

Приложение 2

```
FUNCTION vec_norm(
   vec1_ core_math.t_vector,
   vec2_ core_math.t_vector
```

```
) RETURN NUMBER
IS
  numb NUMBER;
BEGIN
  numb := 0;
  FOR i IN 1..vec1_.count
  LOOP
    numb := numb + (vec2_(i) - vec1_(i)) * (vec2_(i) - vec1_(i));
  END LOOP;
  RETURN SQRT(numb);
END vec_norm;



--return k
FUNCTION BIC_1(
  profit_vector IN core_math.t_vector,
  k PLS_INTEGER
) RETURN NUMBER
IS
  t PLS_INTEGER;
  oc core_math.t_vector;
  T_count PLS_INTEGER;

  X  core_math.t_matrix;
  x_transpose core_math.t_matrix;
  x_inverse core_math.t_matrix;
  x_mul_xt core_math.t_matrix;
  x_in_mul_xt core_math.t_matrix;
  B  core_math.t_vector;
  temp core_math.t_vector;

  numb NUMBER;  --|for tempory computing
  numb2 NUMBER; --|

  i PLS_INTEGER;      --| error code
BEGIN
  T_count := profit_vector.count;

  --compute B vector. from 0! to k + 1-------------------------------
--------------------------
    --compute X matrix
```

```
--    x := core_math.mat_new(T_count - k - 1, k + 1, 0);
      x := core_math.mat_new(k + 1, T_count - k - 1, 0);

      FOR t IN 1..T_count - k - 1
      LOOP
        x(1)(t) := 1;
        FOR i IN 2..k+1
        LOOP
          x(i)(t) := profit_vector(k + t - (i - 1));
        END LOOP;
      END LOOP;
      x := core_math.mat_transp(x);

      --compute (X * Xt)^-1 * Xt * r
      x_transpose := core_math.mat_transp(x);
      x_mul_xt := core_math.mat_mul(x_transpose, x);

      i := core_math.mat_inverse(x_mul_xt, x_inverse);

      x_in_mul_xt := core_math.mat_mul(x_inverse, x_transpose);

      temp := core_math.vec_new(T_count - k - 1, 0);
      FOR i IN 1..T_count - k - 1
      LOOP
        temp(i) := profit_vector(k + i);
      END LOOP;
      b := core_math.mat_mul(x_in_mul_xt, temp);

      --compute oc vector 1..k-----------------------------------------
--------------------------
      oc := core_math.vec_new(T_count);
      FOR t IN 1..k
      LOOP
        numb := 0;
        --sum by a
        FOR j IN 2..k + 1
        LOOP
          numb := numb + b(j);
        END LOOP;
        oc(t) := b(1) / (1 - numb);
      END LOOP;
```

```
          --compute oc vector k+1 to T-----------------------------------------
--------------------------
          FOR t IN k+1..T_count
          LOOP
            numb := 0;
            --sum aj*rj-1
            FOR j IN 2..k + 1
            LOOP
              numb := numb + b(j) *  profit_vector(t - (j - 1));
            END LOOP;
            oc(t) := b(1) + numb;
          END LOOP;
          --compute result-----------------------------------------------------
--------------------------
            --compute sum by r
          numb := 0;
          FOR j IN 1..T_count
          LOOP
            numb := numb + profit_vector(j) / T_Count;
          END LOOP;
          --compute denaminator
          numb2 := 0;
          FOR i IN 1..T_count
          LOOP
            numb2 := numb2 + (oc(i) - numb) * (oc(i) - numb);
          END LOOP;
          --compute fraction
          numb := numb2 / (T_count - k);
          RETURN ln(numb) + ln(k);
        END BIC_1;

        FUNCTION param(
          p_ IN PLS_INTEGER,
          q_ IN PLS_INTEGER,
          lapse IN NUMBER,
          k PLS_INTEGER,
          profit_vector core_math.t_vector
        ) RETURN t_pq_struct
        IS
          qt2 core_math.t_vector;
          w NUMBER;
          T_count PLS_INTEGER;
          y core_math.t_vector;
```

```
          eps core_math.t_vector;
          delta core_math.t_vector;
          zplus core_math.t_vector;
          zminus core_math.t_vector;
          lplus NUMBER;
          lminus NUMBER;

          X  core_math.t_matrix;
          x_transpose core_math.t_matrix;
          x_inverse core_math.t_matrix;
          x_mul_xt core_math.t_matrix;
          x_in_mul_xt core_math.t_matrix;
          B  core_math.t_vector;
          temp core_math.t_vector;

          numb NUMBER;   --|for tempory computing
          numb2 NUMBER;  --|
          sum_y NUMBER;
          sum_d NUMBER;
          last_l NUMBER;
          next_l NUMBER;

          big_fi core_math.t_vector;
          big_fi_next core_math.t_vector;
          i PLS_INTEGER;      --| error code

          n PLS_INTEGER;
          random_vec core_math.t_vector;
          fi NUMBER;
          teta NUMBER;

          res t_pq_struct;
          boo BOOLEAN;
        BEGIN
          --init
          T_count := profit_vector.count;
          --compute B vector. from 0! to k + 1-------------------------------
--------------------------
            --compute X matrix
           x := core_math.mat_new(k + 1, T_count - k - 1, 0);

          FOR t IN 1..T_count - k - 1
          LOOP
```

```
            x(1)(t) := 1;
            FOR i IN 2..k+1
            LOOP
              x(i)(t) := profit_vector(k + t - (i - 1));
            END LOOP;
          END LOOP;
          x := core_math.mat_transp(x);

          --compute (X * Xt)^-1 * Xt * r
          x_transpose := core_math.mat_transp(x);
          x_mul_xt := core_math.mat_mul(x_transpose, x);

          i := core_math.mat_inverse(x_mul_xt, x_inverse);

          x_in_mul_xt := core_math.mat_mul(x_inverse, x_transpose);

          temp := core_math.vec_new(T_count - k - 1, 0);
          FOR i IN 1..T_count - k - 1
          LOOP
            temp(i) := profit_vector(k + i);
          END LOOP;
          b := core_math.mat_mul(x_in_mul_xt, temp);

          --compute eps vector 1..k-----------------------------------------
---------------------------
          eps := core_math.vec_new(T_count);
          FOR t IN 1..k
          LOOP
            numb := 0;
            --sum by a
            FOR j IN 2..k+1
            LOOP
              numb := numb + b(j);
            END LOOP;
            eps(t) :=profit_vector(t) - (b(1) / (1 - numb));
          END LOOP;

          --compute eps vector k+1 to T-------------------------------------
---------------------------
          FOR t IN k+1..T_count
          LOOP
            numb := 0;
            --sum aj*rj-1
```

```
            FOR j IN 2..k + 1
            LOOP
              numb := numb + b(j) * profit_vector(t -  (j - 1));
            END LOOP;
            eps(t) := profit_vector(t) - (b(1) + numb);
          END LOOP;

          --compute y-------------------------------------------------------
--------------------------
          y := core_math.vec_new(q_);
          FOR i IN 1..q_
          LOOP
            y(i) := 1 / (4 * q_);
          END LOOP;

          --compute delta---------------------------------------------------
--------------------------
          delta := core_math.vec_new(p_);
          FOR i IN 1..p_
          LOOP
            delta(i) := 1 / (4 * p_);
          END LOOP;

          --compute w-------------------------------------------------------
--------------------------
          numb := 0;

            --compute sum_y
            sum_y := 0;
            FOR i IN 1..q_
            LOOP
              sum_y := sum_y + y(i);
            END LOOP;

            --compute sum_d
            sum_d := 0;
            FOR i IN 1..p_
            LOOP
              sum_d := sum_d + delta(i);
            END LOOP;

          FOR t IN 1..T_count
          LOOP
```

```
            numb := numb + eps(t)*eps(t)*(1 - sum_y - sum_d);
          END LOOP;
          w := numb / T_count;

          --compute qt^2 1..q-----------------------------------------------
--------------------------
            --compute sum
            numb := 0;
            FOR i IN 1..T_count
            LOOP
              numb := numb + eps(i)*eps(i);
            END LOOP;
            numb := numb / T_count;

          qt2 := core_math.vec_new(T_count);
          FOR i IN 1..p_
          LOOP
            qt2(i) := numb;
          END LOOP;

          --compute qt^2 q+1..T-----------------------------------------------
--------------------------
          FOR t IN p_ + 1..T_count
          LOOP
            --compute first sum
            numb := 0;
            FOR j IN 1..p_
            LOOP
              numb := numb + delta(j) * qt2(t - j);
            END LOOP;

            --compute second sum
            numb2 := 0;
            FOR j IN 1..q_
            LOOP
              numb2 := numb2 + y(j) * eps(t - j) * eps(t - j);
            END LOOP;

            qt2(t) := w + numb + numb2;
          END LOOP;

          --compute first last_l---------------------------------------------
-------------------------------
```

```
          numb := 0;
          FOR i IN 1..T_count
          LOOP
            numb :=numb + ln(qt2(i)) + (eps(i) * eps(i) / qt2(i));
          END LOOP;
          last_l := -(T_Count * ln(2 * 3.141593) + numb) / 2;

          --compute big fi-----------------------------------------------------
--------------------------
          big_fi := core_math.vec_new(k+2+q_+p_+p_, 0);
          --write B
          FOR j IN 1..k+1
          LOOP
            big_fi(j) := b(j);
          END LOOP;
          --write w
          big_fi(k + 2) := w;

          --write y
          FOR j IN k+3..k+2+q_
          LOOP
            big_fi(j) := y(j - (k + 3) + 1);
          END LOOP;

          --write delta
          FOR j IN k+2+q_+1..k+2+q_+p_
          LOOP
            big_fi(j) := delta(j - (k+2+q_+1) + 1);
          END LOOP;

          --write qt2
          FOR j IN k+2+q_+p_+1..k+2+q_+p_+p_
          LOOP
            big_fi(j) := qt2(j - (k+2+q_+p_+1) + 1);
          END LOOP;

          dbms_random.initialize(588585858);
          --main_loop-----------------------------------------------------------
--------------------------
          n := 1;
          LOOP
            <<begin_of_loop>>
            big_fi_next := core_math.vec_new(k+2+q_+p_+p_, 0);
```

```
            --match_vector-----------------------------------------------------
---------------------------
              random_vec := core_math.vec_new(k+2+q_+p_+p_, 0);
              FOR i IN 1..k+2+q_+p_+p_
              LOOP
                numb := dbms_random.value;
                IF numb >= 0.5
                THEN
                  random_vec(i) := 1 / SQRT(big_fi.count);
                ELSE
                  random_vec(i) := -1 / SQRT(big_fi.count);
                END IF;
              END LOOP;

              fi := 1 / n;
              teta := 1 / SQRT(n);
              zplus := core_math.vec_new(k+2+q_+p_+p_);
              zminus := core_math.vec_new(k+2+q_+p_+p_);
              lplus := 0;
              lminus := 0;
              <<compute_z>>
              --compute zplus zminus----------------------------------------
---------------------------
              FOR i IN 1..k+2+q_+p_+p_
              LOOP
                zplus(i) := big_fi(i) + teta * random_vec(i);
                zminus(i) := big_fi(i) - teta * random_vec(i);
              END LOOP;


              --check componetn of zplus and zminus--------------------------
---------------------------
              boo := TRUE;
              IF zplus(k+2) <= 0
                THEN
                  boo := FALSE;
                  random_vec(k+2) := big_fi(i) / (2 * teta);
              END IF;

              IF zminus(k+2) <= 0
                THEN
                  boo := FALSE;
```

```
                random_vec(k+2) := big_fi(i) / (2 * teta);
              END IF;

              IF zplus(k+2+q_+p_+1) <= 0
                THEN
                  boo := FALSE;
                  random_vec(k+2+q_+p_+1) := big_fi(i) / (2 * teta);
              END IF;

              IF zminus(k+2+q_+p_+1) <= 0
                THEN
                  boo := FALSE;
                  random_vec(k+2+q_+p_+1) := big_fi(i) / (2 * teta);
              END IF;

              FOR i IN k+3..k+2+q_+p_+p_
              LOOP
                IF zplus(i) < 0  AND i <> k+2+q_+p_+1
                THEN
                  boo := FALSE;
                  random_vec(i) := big_fi(i) / (2 * teta);
                END IF;

                IF zminus(i) < 0  AND i <> k+2+q_+p_+1
                THEN
                  boo := FALSE;
                  random_vec(i) := big_fi(i) / (2 * teta);
                END IF;
              END LOOP;
              IF NOT boo THEN GOTO compute_z; END IF;

              --compute eps vector 1..k from zplus----------------------------
-------------------------
              eps := core_math.vec_new(T_count, 0);
              FOR t IN 1..k
              LOOP
                numb := 0;
                --sum by a
                FOR j IN 2..k+1
                LOOP
                  numb := numb + zplus(j);
                END LOOP;
                eps(t) :=profit_vector(t) - (zplus(1) / (1 - numb));
```

```
            END LOOP;

            --compute eps vector k+1 to T from zplus-----------------------
-------------------------
            FOR t IN k+1..T_count
            LOOP
              numb := 0;
              --sum aj*rj-1
              FOR j IN 2..k+1
              LOOP
                numb := numb + zplus(j) * profit_vector(t-(j-1));
              END LOOP;
              eps(t) := profit_vector(t) - (zplus(1) + numb);
            END LOOP;

            --compute w from fi from zplus---------------------------------
-------------------------
            w := zplus(k + 2);

            --compute y from fi from zplus---------------------------------
-------------------------
            y := core_math.vec_new(q_);
            FOR i IN 1..q_
            LOOP
              y(i) := zplus(k + 2 + i);
            END LOOP;

            --compute delta from zplus-------------------------------------
-------------------------
            delta := core_math.vec_new(p_);
            FOR i IN 1..p_
            LOOP
              delta(i) := zplus(k + 2 + q_ + i);
            END LOOP;

            --compute qt^2 1..q--------------------------------------------
-------------------------
            FOR t IN 1..p_
            LOOP
              qt2(t) := zplus(k+2+q_+p_ + t);
            END LOOP;
```

```
              --compute qt^2 q+1..T-------------------------------------------
--------------------------
              FOR t IN p_+1..T_count
              LOOP
                --compute first sum
                numb := 0;
                FOR j IN 1..p_
                LOOP
                  numb := numb + delta(j) * qt2(t - j);
                END LOOP;

                --compute second sum
                numb2 := 0;
                FOR j IN 1..q_
                LOOP
                  numb2 := numb2 + y(j) * eps(t - j) * eps(t - j);
                END LOOP;

                qt2(t) := w + numb + numb2;
              END LOOP;

              ----------------------------------------------------------------
--------------------------
              --compute lplus-------------------------------------------------
--------------------------
              numb := 0;
              FOR i IN 1..T_count
              LOOP
                numb :=numb + ln(qt2(i)) + (eps(i) * eps(i) / qt2(i));
              END LOOP;

              lplus := -(T_Count * ln(2 * 3.141593) + numb) / 2;

              --compute eps vector 1..k from zminus--------------------------
--------------------------
              eps := core_math.vec_new(T_count, 0);
              FOR t IN 1..k
              LOOP
                numb := 0;
                --sum by a
                FOR j IN 2..k+1
                LOOP
                  numb := numb + zminus(j);
```

```
              END LOOP;
              eps(t) :=profit_vector(t) - (zminus(1) / (1 - numb));
            END LOOP;

            --compute eps vector k+1 to T from zminus----------------------
-------------------------
            FOR t IN k+1..T_count
            LOOP
              numb := 0;
              --sum aj*rj-1
              FOR j IN 2..k+1
              LOOP
                numb := numb + zminus(j) * profit_vector(t-(j-1));
              END LOOP;
              eps(t) := profit_vector(t) - (zminus(1) + numb);
            END LOOP;
            --compute w from fi from zminus---------------------------------
-------------------------
            w := zminus(k + 2);

            --compute y from fi from zminus---------------------------------
-------------------------
            y := core_math.vec_new(q_);
            FOR i IN 1..q_
            LOOP
              y(i) := zminus(k + 2 + i);
            END LOOP;

            --compute delta from zminus-------------------------------------
-------------------------
            delta := core_math.vec_new(p_);
            FOR i IN 1..p_
            LOOP
              delta(i) := zminus(k + 2 + q_ + i);
            END LOOP;

            --compute qt^2 1..q---------------------------------------------
-------------------------
            FOR t IN 1..p_
            LOOP
              qt2(t) := zminus(k+2+q_+p_ + t);
            END LOOP;
```

```
              --compute qt^2 q+1..T-----------------------------------------
--------------------------
              FOR t IN p_+1..T_count
              LOOP
                --compute first sum
                numb := 0;
                FOR j IN 1..p_
                LOOP
                  numb := numb + delta(j) * qt2(t - j);
                END LOOP;

                --compute second sum
                numb2 := 0;
                FOR j IN 1..q_
                LOOP
                  numb2 := numb2 + y(j) * eps(t - j) * eps(t - j);
                END LOOP;
                qt2(t) := w + numb + numb2;
              END LOOP;

              --compute lminus-------------------------------------------------
-----------------------
              numb := 0;
              FOR i IN 1..T_count
              LOOP
                numb :=numb + ln(qt2(i)) + (eps(i) * eps(i) / qt2(i));
              END LOOP;
              lminus := -(T_Count * ln(2 * 3.141593) + numb) / 2;
              <<compute_next_bf>>
              --compute bif_fi_next-------------------------------------------
----------------------
              FOR i IN 1..k+2+q_+p_+p_
              LOOP
                big_fi_next(i) := big_fi(i) - random_vec(i) * fi * ((lplus -
lminus) / (2 * teta));
              END LOOP;

              boo := TRUE;
              --check sum-----------------------------------------------------
----------------------
              numb := 0;
              FOR i IN 1..p_
              LOOP
```

```
                numb := numb + big_fi_next(k+2+q_ + i);
              END LOOP;
              FOR i IN 1..q_
              LOOP
                numb := numb + big_fi_next(k+2 + i);
              END LOOP;

              IF numb >= 1
              THEN
                boo := FALSE;
                FOR i IN k+3..k+2+q_+p_
                LOOP
                  random_vec(i) := big_fi(i) / ((p_ + q_ + 1) * fi * ((lplus -
lminus) / (2 * teta)));
                END LOOP;
              END IF;

              IF big_fi_next(k + 2) < 0
              THEN
                boo := FALSE;
                random_vec(k + 2) := big_fi(k+2) / ((2 * teta) * fi * ((lplus
- lminus) / (2 * teta)));
              END IF;

              FOR i IN 1..p_
              LOOP
                IF big_fi_next(k+2+q_ + i) <= 0
                THEN
                  boo := FALSE;
                  random_vec(k+2+q_ + i) := big_fi(k+2+q_ + i)  / ((2 * teta)
* fi * ((lplus - lminus) / (2 * teta)));
                END IF;
              END LOOP;

              FOR i IN 1..q_
              LOOP
                IF big_fi_next(k+2 + i) <= 0
                THEN
                  boo := FALSE;
                  random_vec(k+2 + i) := big_fi(k+2 + i)  / ((2 * teta) * fi *
((lplus - lminus) / (2 * teta)));
                END IF;
              END LOOP;
```

```
              FOR i IN k+2+q_+p_+1..k+2+q_+p_+p_
              LOOP
                IF big_fi_next(i) <= 0
                THEN
                  boo := FALSE;
                  random_vec(i) := big_fi(i)  / ((2 * teta) * fi * ((lplus -
lminus) / (2 * teta)));
                END IF;
              END LOOP;
              IF NOT boo THEN GOTO compute_next_bf; END IF;

            --!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
------------------
            --EXIT;
            --!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
------------------
            n := n + 1;
            EXIT WHEN vec_norm(big_fi_next, big_fi) < lapse;

            --choose new vector-----------------------------------------------
------------------
            --compute l for big_fi_next
            /*
              core_math.put_vector(big_fi);
              dbms_output.put_line('----------');
              core_math.put_vector(big_fi_next);
            */
              --compute eps vector 1..k from big_fi_next--------------------
--------------------------------
              eps := core_math.vec_new(T_count, 0);
              FOR t IN 1..k
              LOOP
                numb := 0;
                --sum by a
                FOR j IN 2..k+1
                LOOP
                  numb := numb + big_fi_next(j);
                END LOOP;
                eps(t) :=profit_vector(t) - (big_fi_next(1) / (1 - numb));
              END LOOP;
```

```
            --compute eps vector k+1 to T from zplus-----------------------
-------------------------
            FOR t IN k+1..T_count
            LOOP
              numb := 0;
              --sum aj*rj-1
              FOR j IN 2..k+1
              LOOP
                numb := numb + big_fi_next(j) * profit_vector(t-(j-1));
              END LOOP;
              eps(t) := profit_vector(t) - (big_fi_next(1) + numb);
            END LOOP;

            --compute w from fi from zplus---------------------------------
-------------------------
            w := zplus(k + 2);

            --compute y from fi from zplus---------------------------------
-------------------------
            y := core_math.vec_new(q_);
            FOR i IN 1..q_
            LOOP
              y(i) := big_fi_next(k + 2 + i);
            END LOOP;

            --compute delta from zplus-------------------------------------
-------------------------
            delta := core_math.vec_new(p_);
            FOR i IN 1..p_
            LOOP
              delta(i) := big_fi_next(k + 2 + q_ + i);
            END LOOP;

            --compute qt^2 1..q--------------------------------------------
-------------------------
            FOR t IN 1..p_
            LOOP
              qt2(t) := big_fi_next(k+2+q_+p_ + t);
            END LOOP;

            --compute qt^2 q+1..T------------------------------------------
-------------------------
            FOR t IN p_+1..T_count
```

```
            LOOP
              --compute first sum
              numb := 0;
              FOR j IN 1..p_
              LOOP
                numb := numb + delta(j) * qt2(t - j);
              END LOOP;

              --compute second sum
              numb2 := 0;
              FOR j IN 1..q_
              LOOP
                numb2 := numb2 + y(j) * eps(t - j) * eps(t - j);
              END LOOP;

              qt2(t) := w + numb + numb2;
            END LOOP;

            ------------------------------------------------------------
---------------------------
            --compute next_l-----------------------------------------------
---------------------------
            numb := 0;
            FOR i IN 1..T_count
            LOOP
              numb :=numb + ln(qt2(i)) + (eps(i) * eps(i) / qt2(i));
            END LOOP;

            next_l := -(T_Count * ln(2 * 3.141593) + numb) / 2;
            IF next_l > last_l
            THEN
              big_fi := big_fi_next;
              last_l := next_l;
            END IF;
          END LOOP;

          dbms_output.put_line(n);
          dbms_output.put_line(last_l);
          dbms_output.put_line('----------');
          big_fi := big_fi_next;

          --fill result---------------------------------------------------
---------------------------
```

```
  res.B := core_math.vec_new(k + 1, 0);
  FOR i IN 1..k+1
  LOOP
    res.B(i) := big_fi_next(i);
  END LOOP;
  core_math.put_vector(res.B);
  dbms_output.put_line('----------');

  --
  res.w := big_fi_next(k + 2);
  dbms_output.put_line(w);
  dbms_output.put_line('----------');

  --
  res.Y := core_math.vec_new(q_, 0);
  FOR i IN k+3..k+2+q_
  LOOP
    res.Y(i - (k+2)) := big_fi_next(i);
  END LOOP;
  core_math.put_vector(res.Y);
  dbms_output.put_line('----------');
  --
  res.delta := core_math.vec_new(p_, 0);
  FOR i IN k+2+q_+1..k+2+q_+p_
  LOOP
    res.delta(i - (k+2+q_)) := big_fi_next(i);
  END LOOP;
  core_math.put_vector(res.delta);
  dbms_output.put_line('----------');
  --
  res.qt2 := core_math.vec_new(p_);
  FOR i IN k+2+q_+p_+1..k+2+q_+p_+p_
  LOOP
    res.qt2(i - (k+2+q_+p_)) := big_fi_next(i);
  END LOOP;
  core_math.put_vector(res.qt2);
  dbms_output.put_line('----------');
  RETURN res;
END param;

FUNCTION BIC_2(
  p_ IN PLS_INTEGER,
  q_ IN PLS_INTEGER,
```

```
          lapse IN NUMBER,
          k PLS_INTEGER,
          profit_vector core_math.t_vector
        ) RETURN NUMBER
        IS
          pq_struct t_pq_struct;
          eps core_math.t_vector;
          oct2 core_math.t_vector;
          T_count PLS_INTEGER;
          numb NUMBER;
          numb2 NUMBER;
        BEGIN

          T_count := profit_vector.count;
          pq_struct   :=   param(p_,   q_,   lapse,   k,   profit_vector   =>
profit_vector);

          --compute eps vector 1..k from pq_struct-----------------------------
---------------------------
          eps := core_math.vec_new(T_count);
          FOR t IN 1..k
          LOOP
            numb := 0;
            --sum by a
            FOR j IN 2..k+1
            LOOP
              numb := numb + pq_struct.B(j);
            END LOOP;
            eps(t) :=profit_vector(t) - (pq_struct.B(1) / (1 - numb));
          END LOOP;

          --compute eps vector k+1 to T from pq_struct-----------------------
---------------------------
          FOR t IN k+1..T_count
          LOOP
            numb := 0;
            --sum aj*rj-1
            FOR j IN 2..k+1
            LOOP
              numb := numb + pq_struct.B(j) * profit_vector(t-(j-1));
            END LOOP;
            eps(t) := profit_vector(t) - (pq_struct.B(1) + numb);
          END LOOP;
```

```
          --compute oct^2 1..p-----------------------------------------------
--------------------------
          oct2 := core_math.vec_new(T_count);
          FOR i IN 1..p_
          LOOP
            oct2(i) := pq_struct.qt2(i);
          END LOOP;
          --compute oct^2 p..T------------------------------------------------
--------------------------
          FOR t IN p_ + 1..T_count
          LOOP
            --compute first sum
            numb := 0;
            FOR j IN 1..p_
            LOOP
              numb := numb + pq_struct.delta(j) * oct2(t - j);
            END LOOP;

            --compute second sum
            numb2 := 0;
            FOR j IN 1..q_
            LOOP
              numb2 := numb2 + pq_struct.Y(j) * eps(t - j) * eps(t - j);
            END LOOP;
            oct2(t) := pq_struct.w + numb + numb2;
          END LOOP;

          --result-------------------------------------------------------------
--------------------------
            --compute sum by r
          numb := 0;
          FOR j IN 1..T_count
          LOOP
            numb := numb + eps(j) * eps(j);
          END LOOP;
          numb := numb / T_count;
          --compute denaminator
          numb2 := 0;
          FOR i IN 1..T_count
          LOOP
            numb2 := numb2 + (oct2(i) - numb) * (oct2(i) - numb);
          END LOOP;
```

```
          --compute fraction
          numb := numb2 / (T_count - p_ -  q_);
          RETURN ln(numb) + ln(p_ + q_);
        END BIC_2;

        FUNCTION Main_Compute(
          m PLS_INTEGER,
          Pt core_math.t_vector,
          lapse_ NUMBER
        ) RETURN PLS_INTEGER
        IS
          mu core_math.t_vector;
          eps core_math.t_vector;
          profit_vector core_math.t_vector;
          qt2 core_math.t_vector;

          T_count PLS_INTEGER;
          k_min NUMBER;
          k PLS_INTEGER;
          pq_min NUMBER;
          p_ PLS_INTEGER;
          q_ PLS_INTEGER;
          numb NUMBER;
          numb2 NUMBER;
          x NUMBER;
          y NUMBER;
          Ee NUMBER;
          pq_struct t_pq_struct;

          trust_interval_down core_math.t_vector;
          trust_interval_up core_math.t_vector;
        BEGIN
          T_count := Pt.count;
          profit_vector := core_math.vec_new(T_count + m, 0);

          --compute profit_vector----------------------------------------------
---------------------------
          FOR i IN 2..T_count
          LOOP
            profit_vector(i) := ln(Pt(i) / Pt(i - 1));
          END LOOP;
```

```
          --find min BIC_1---------------------------------------------------
--------------------------
          k_min := 9999999999999;
          FOR i IN 1..2
          LOOP
            numb := BIC_1(profit_vector, i);
            IF k_min > numb
            THEN
              k_min := numb;
              k := i;
            END IF;
          END LOOP;

          --find min BIC_2---------------------------------------------------
--------------------------
          pq_min := 9999999999999;
          FOR i IN 1..2
          LOOP
            FOR j IN 1..2
            LOOP
              numb := BIC_2(i, j, lapse => lapse_, k => k, profit_vector =>
profit_vector);
              IF pq_min > numb
              THEN
                pq_min := numb;
                p_ := i;
                q_ := j;
              END IF;
            END LOOP;
          END LOOP;

          pq_struct := param(p_, q_, lapse_, k, profit_vector);
          RETURN 0;
          --compute mu-------------------------------------------------------
--------------------------
          mu := core_math.vec_new(T_count + m);

          FOR t IN 1..k
          LOOP
            numb := 0;
            --sum by a
            FOR j IN 2..k+1
            LOOP
```

```
            numb := numb + pq_struct.b(j);
          END LOOP;
          mu(t) :=(pq_struct.b(1) / (1 - numb));
        END LOOP;

        --compute mu-----------------------------------------------------
-------------------------
        FOR t IN k+1..T_count
        LOOP
          numb := 0;
          --sum aj*rj-1
          FOR j IN 1..k
          LOOP
            numb := numb + pq_struct.b(j) * profit_vector(t-j);
          END LOOP;
          mu(t) := (pq_struct.b(1) + numb);
        END LOOP;

        --compute eps-----------------------------------------------------
-------------------------
        eps := core_math.vec_new(T_count + m);
        FOR t IN 1..T_count
        LOOP
          eps(t):= profit_vector(t) - mu(t);
        END LOOP;

        --compute qt^2 1..q-----------------------------------------------
-------------------------
        qt2 := core_math.vec_new(T_count+m);

        FOR i IN 1..q_
        LOOP
          qt2(i) := pq_struct.qt2(i);
        END LOOP;

        --compute qt^2 q+1..T---------------------------------------------
-------------------------
        FOR t IN q_ + 1..T_count
        LOOP
          --compute first sum
          numb := 0;
          FOR j IN 1..p_
          LOOP
```

```
        numb := numb + pq_struct.delta(j) * qt2(t - j);
      END LOOP;

      --compute second sum
      numb2 := 0;
      FOR j IN 1..q_
      LOOP
        numb2 := numb2 + pq_struct.y(j) * eps(t - j) * eps(t - j);
      END LOOP;

      qt2(t) := pq_struct.w + numb + numb2;
    END LOOP;

    --compute e
    LOOP
      IF ROUND(dbms_random.value, 1) >= 0.5
      THEN
        x := dbms_random.value;
      ELSE
        x := 0 - dbms_random.value;
      END IF;

      IF ROUND(dbms_random.value, 1) >= 0.5
      THEN
        y := dbms_random.value;
      ELSE
        y := 0 - dbms_random.value;
      END IF;
      EXIT WHEN x*x + y*y != 0 AND x*x + y*y < 1;
    END LOOP;
    ee := x * SQRT(-2 * ln(x*x + y*y) / (x*x + y*y));

    FOR i IN 1..m
    LOOP
      --compute qt2-----------------------------------------------------
------------------
        --compute first sum
      numb := 0;
      FOR j IN 1..p_
      LOOP
        numb := numb + pq_struct.delta(j) * qt2(T_count + i - j);
      END LOOP;
```

```
            --compute second sum
            numb2 := 0;
            FOR j IN 1..q_
            LOOP
              numb2 := numb2 + pq_struct.y(j) * eps(T_count + i - j) *
eps(T_count + i - j);
            END LOOP;
            qt2(T_count + i) := pq_struct.w + numb + numb2;
            eps(T_count + i) := ee * qt2(T_count + i);

            --compute mu-----------------------------------------------
            numb := 0;
            --sum aj*rj-1
            FOR j IN 1..k
            LOOP
              numb := numb + pq_struct.b(j) * profit_vector(T_count + i -j);
            END LOOP;
            mu(T_count + i) := (pq_struct.b(1) + numb);
          END LOOP;

          FOR i IN 1..m
          LOOP
            profit_vector(T_count + i) := mu(T_count + i) + eps(T_count + i);
          END LOOP;

          --compute trust interval_up and down-------------------------------
-----
          trust_interval_down := core_math.vec_new(m);
          trust_interval_up := core_math.vec_new(m);
          FOR t IN 1..m
          LOOP
            --compute sum
            numb := 0;
            FOR i IN 1..T_count
            LOOP
              numb := numb + profit_vector(i);
            END LOOP;
            numb := numb / T_count;

            numb2 := 2 * SQRT(qt2(T_count + m)) / SQRT(T_count);

            trust_interval_down(t) := numb - numb2;
            trust_interval_up(t) := numb + numb2;
```

```
    END LOOP;

  END Main_Compute;

BEGIN
  -- Initialization
  NULL;
```