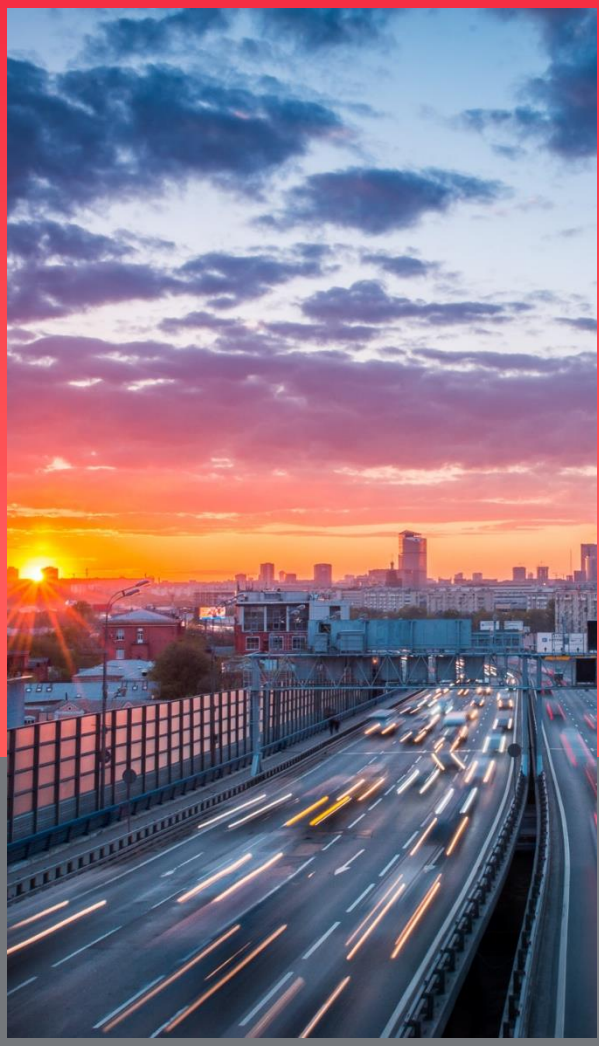
Банк России



Быстрая оценка байесовских моделей пространства состояний с использованием симуляций

Серия препринтов об экономических исследованиях

№ 104 / Декабрь 2022

Сергей Селезнев, Рамис Хабибуллин

Сергей Селезнев

Банк России, Департамент исследований и прогнозирования

E-mail: seleznevsm@cbr.ru

Рамис Хабибуллин

Независимый исследователь

Авторы выражают благодарность Дмитрию Горностаеву, Сергею Иващенко, Петру Милютину, Денису Шибитову, Александру Щеглову, Виталию Юфереву и участникам XV Международной конференции по вычислительной и финансовой эконометрике (CFE 2021), XXIII Ясинской (Апрельской) международной научной конференции по проблемам развития экономики и общества и II Международной конференции по эконометрике и бизнес-аналитике (iCEBA) за их комментарии и советы.

Препринты Банка России проходят процедуру анонимного рецензирования со стороны членов Консультативного исследовательского совета Банка России и внешних рецензентов.

Фото на обложке: Shutterstock/FOTODOM

Адрес: 107016, Москва, ул. Неглинная, 12
Телефоны: +7 (499) 300-30-00, +7 (495) 621-64-65 (факс)
Официальный сайт Банка России: www.cbr.ru

Настоящий материал подготовлен Департаментом исследований и прогнозирования Банка России.

Аннотация

Эта работа презентует быстрый алгоритм для оценки скрытых состояний байесовских моделей пространства состояний. Алгоритм является вариацией амортизированных симуляционных алгоритмов, где на первом этапе создается большое количество искусственных наборов данных, а затем обучается гибкая модель для прогнозирования интересующих переменных. Описанная в этой работе процедура, в отличие от предложенных ранее, позволяет обучать алгоритмы оценки для скрытых состояний за счет концентрации только на определенных характеристиках предельных апостериорных распределений, а также использования нейронной сети, отражающей специфику данных.

Иллюстрации на примере моделей стохастической волатильности, нелинейной динамической стохастической модели общего равновесия и процедуры сезонной оценки со сдвигами в сезонности показывают, что алгоритм имеет достаточную для практического использования точность. Более того, после предобучения, которое занимает несколько часов, нахождение апостериорного распределения для любого набора данных занимает от сотых до десятых секунды.

JEL-коды: C11, C15, C32, C45.

Ключевые слова: амортизированный симуляционный алгоритм оценки, байесовские модели пространства состояний, нейронные сети, сезонная корректировка, стохастическая волатильность, SV-DSGE.

Оглавление

1. Введение

Байесовские модели пространства состояний широко используются в прикладной макроэкономике. Они получили такое распространение ввиду того, что в виде моделей пространства состояний для последующей оценки на реальных данных может быть записано множество макроэкономических и эконометрических моделей. К примеру, в качестве моделей пространства состояний могут быть записаны различного рода фильтры и полуструктурные фильтры (см. Hodrick and Prescott (1997), Laubach and Williams (2003)), модели со стохастической волатильностью (см. Kim, Shepard and Chib (1998), Justiniano and Primiceri (2008), Carriero, Clark and Marcellino (2016)), модели с изменяющимися во времени параметрами (см. Hamilton (1989), Primiceri (2005), Koop and Korobilis (2012)), модели со смешанной частотностью (см. Chiu et al (2012), Schorfheide and Song (2015, 2021)), динамические факторные модели (см. Otrok and Whiteman (1998), Stock and Watson (2011)), динамические стохастические модели общего равновесия (см. Smets and Wouters (2003, 2007), Fernandez-Villaverde, Schorfheide and Rubio-Ramirez (2016)) и агент-ориентированные модели (см. Lux (2018), Deli Gatti and Grazzini (2020)). Байесовская же оценка параметров позволяет частично нивелировать отсутствие длинных временных рядов[1].

Несмотря на их гибкость, на практике оценка байесовских моделей пространства состояний является достаточно непростой задачей. Семплинговые алгоритмы основаны на итеративной схеме последовательного семплирования параметров и состояний модели и опираются на такие шаги, как семплирование по Гиббсу (см. Casella and George (1992)), алгоритм Метрополиса – Гастингса (см. Chib and Greenberg (1995)), гамильтоновы методы Монте-Карло (см. Neal (1996)) или последовательный алгоритм Монте-Карло (см. Del Moral, Doucet and Jasra (2006)). Даже в случаях когда модель является линейной и гауссовой или же дискретной относительно состояний, семплирование занимает от десятков минут до десятков часов. В системах же более общего вида исследователи используют фильтры частиц (см. Andrieu, Doucet and Holenstein (2010), Chopin, Jacob and Papaspiliopoulos (2012)), в результате чего время их оценки только увеличивается. Семплинговые алгоритмы являются точными в том смысле, что они сходятся к апостериорному распределению при стремящемся к бесконечности количестве итераций, правда, количество итераций, необходимое для близкой к апостериорному распределению оценки, может быть большим. Альтернатива им – оптимизационные алгоритмы, наиболее распространенным из которых в контексте моделей

[1] В этой работе мы фокусируемся на временных рядах, однако предложенный алгоритм может быть легко перенесен на другие модели со скрытыми переменными.

пространства состояний является вариационный байесовский алгоритм[2] (см. Wainwright and Jordan (2008), Hoffman et al. (2013)). На практике вариационный байесовский алгоритм, основанный на минимизации дивергенции Кульбака – Лейблера между аппроксимацией и настоящим апостериорным распределением, часто оказывается быстрее, чем семплинговые алгоритмы, однако время его работы, особенно в случаях когда оптимизационные шаги не могут быть записаны в аналитической форме, также велико.

В этой статье мы предлагаем быстрый алгоритм оценки байесовских моделей пространства состояний, который основан на принципах симуляционных алгоритмов оценки (см. Cranmer, Brehmer and Louppe (2020)). Быстрота работы алгоритма достигается путем амортизации задачи построения апостериорного распределения состояний, то есть предварительного обучения модели (нейронной сети в нашем случае), которая по набору данных прогнозирует его апостериорное распределение. При этом мы концентрируемся только на состояниях и делаем это по двум причинам. Во-первых, во многих задачах именно состояния, а не параметры модели представляют особый интерес. Так, например, в задаче выделения тренда (см. Orphanides and Van Norden (2002)) именно циклическая и трендовая составляющие, которые определяются скрытыми состояниями, представляют основной интерес. Во-вторых, оценка параметров на основе симуляционных алгоритмов была рассмотрена во множестве других работ (см. Приложение A из Lueckmann et al. (2021)) и может быть легко соединена с подходом, рассматриваемым в этой статье. Задача же построения апостериорного распределения скрытых состояний ввиду ее размерности является куда более сложной и практически не исследовалась в литературе.

В разделе 2 описывается алгоритм оценки апостериорного распределения модели. Раздел 3 посвящен изучению применения, практических характеристик и сравнению предложенного алгоритма с альтернативными алгоритмами, которые часто используются на практике. В разделе 4 описываются похожие направления в литературе. В разделе 5 затронуты вопросы, которые не были включены в работу, но являются важными в контексте использования алгоритма и требуют внимания при его применении в будущем. Заключение представлено в разделе 6.

[2] См. главы 3 и 5 из Beal (2003) и Gunawan, Kohn and Nott (2021) в качестве примеров использования вариационного байесовского алгоритма для оценки моделей пространства состояний.

2. Процедура оценки

2.1. Амортизированные симуляционные алгоритмы оценки

Наша методология оценки моделей пространства состояний основана на идее оценки параметров байесовских моделей, предложенной в Beaumont, Zhang and Balding (2002), Blum and Francois (2010) и развитой в Papamakarios and Murray (2016). Суть методологии состоит в симуляции совместного распределения параметров и данных и последующем прогнозировании параметров при заданных данных.

Формально для модели с априорным распределением $p(\theta)$ и функцией правдоподобия $p(y|\theta)$ на первом шаге симулируется набор данных, где i-я точка состоит из параметров $\theta_i \sim p(\theta)$ и наблюдаемых переменных $y_i \sim p(y|\theta_i)$. На втором шаге на основании полученного набора данных оценивается модель, позволяющая в том или ином виде оценивать распределение $p(\theta|y)$ или его характеристики, например, путем минимизации кросс-энтропии между искусственными данными и некоторым параметрическим семейством распределений $q_\varphi(\theta|y)p(y)$:

$$\varphi^* = argmin_\varphi\left(-\sum_{i=1}^{N}\left(log q_\varphi(\theta_i|y_i) + log\, p(y_i)\right)\right), \qquad (1)$$

где $\varphi -$ параметры распределения $q_\varphi(\theta|y)$, $N -$ количество симуляций. Далее мы будем пропускать слагаемое $log\, p(y_i)$ ввиду того, что оно не зависит от φ.

При стремящемся к бесконечности количестве симуляций и достаточно гибком параметрическом семействе q_φ оцененное распределение будет стремиться к апостериорному при любом y (см. неформальное доказательство в Приложении А.1). Это фактически означает, что алгоритм обладает свойством амортизации, то есть что оцененное единожды условное распределение $q_{\varphi^*}(\theta|y)$ может использоваться вне зависимости от данных, не требует переоценки модели и производится практически мгновенно.

Как можно заметить, данный метод оценки апостериорных распределений (далее, следуя Papamakarios and Murray (2016), мы будем называть его NPE) не требует знания $p(\theta)$ и $p(y|\theta)$ в явном виде, а основывается лишь на возможности симулирования данных, что является естественным для большинства моделей, и попадает в категорию симуляционных алгоритмов оценки *(simulation based inference (SBI),* или *likelihood-free inference*).

2.2. От байесовских моделей к моделям пространства состояний

Параметры моделей пространства состояний могут быть оценены с помощью процедуры, описанной в разделе 2.1, однако целью данной работы является оценка скрытых

состояний. Несложно заметить, что если заменить параметры на скрытые состояния в функции потерь, то процедура, описанная выше, останется валидной (см. Приложение A.2). Таким образом, в общем виде алгоритм для нахождения апостериорного распределения может быть записан, как показано ниже.

Алгоритм 1. Симуляционный алгоритм оценки байесовских моделей пространства состояний

Для $i = 1, \dots, N$:

1. Симулировать скрытые состояния и наблюдаемые данные:
   1.a. Семплировать параметры модели из априорного распределения:
   $$\theta_i \sim p(\theta).$$
   1.b. Семплировать состояния при условии параметров модели:
   $$s_i \sim p(s|\theta_i).$$
   1.c. Семплировать наблюдаемые данные при условии состояний и параметров:
   $$y_i \sim p(y|s_i, \theta_i).$$
2. Найти параметры аппроксимации апостериорного распределения состояний:
   $$\varphi^* = argmin_\varphi\left(-\sum_{i=1}^{N} log\, q_\varphi(s_i|y_i)\right). \quad (2)$$

Несмотря на простоту алгоритма 1, существует несколько практических сложностей при переходе от нахождения апостериорного распределения для параметров к распределению состояний. Во-первых, это большая размерность пространства скрытых состояний. Несмотря на наличие различного рода потоковых трансформаций (см. Rezende and Mohamed (2015)), которые часто используются в SBI и во многих случаях хорошо восстанавливают совместные распределения, их применение для задач такого размера, да еще и с амортизацией, оказывается вычислительно затратным и сопряженным с оптимизационными вызовами. Во-вторых, это большая размерность данных. Для скрытых состояний практически невозможно найти агрегированные статистики, которые снижали бы размерность данных, в отличие от задачи оценки параметров, где это является обычной практикой (см., например, SIR (T.9) и Lotka-Voltera (T.10) модели в Lueckmann et al. (2021)). Чтобы обойти эти проблемы и упростить задачу обучения, мы сосредотачиваемся на характеристиках предельных распределений, которые зачастую представляют наибольший практический интерес, не моделируя при этом зависимости между переменными, и подбираем архитектуру для $q_\varphi(s|y)$ исходя из особенностей данных.

2.3. Функция потерь для предельных распределений

Переход от полного распределения состояний к предельным распределениям фактически эквивалентен дроблению изначальной многомерной задачи на множество одномерных. Логарифм вероятности для параметрического семейства $q_\varphi(s|y)$ в этом случае запишется как:

$$log\ q_\varphi(s|y) = \sum_{t=1}^{T}\sum_{k=1}^{k} log\ q_{\varphi_{t,k}}(s^{t,k}|y), \qquad (3)$$

где t и $k-$ период времени, за который ищется апостериорное распределение и индекс состояния в векторе состояний. Отметим, что для каждого состояния вектор параметров $\varphi_{t,k}$ в общем случае оказывается своим, а φ состоит из набора этих векторов.

В качестве $q_{\varphi_{t,k}}(s^{t,k}|y)$ мы используем нормальное распределение со средним $m_{\varphi_{t,k}}(y)$ и стандартным отклонением $\sigma_{\varphi_{t,k}}(y)$. Несложно показать, что при достаточно гибких $m_{\varphi_{t,k}}$ и $\sigma_{\varphi_{t,k}}$ такая аппроксимация в точности восстанавливает среднее и стандартное отклонение истинного апостериорного распределения (см. Приложение А.3).

Более того, выбор нормального распределения и кросс-энтропии в качестве функции потерь могут быть легко изменены. Так, вместо нормального распределения для предельных плотностей может быть использована смесь нормальных или небольшая потоковая модель, а вместо кросс-энтропии – любая М-оценка (см. главу 5 в Van der Vaart (2000)), например квантильная регрессия.

2.4. Архитектура модели оценки

В качестве модели оценки выбирается нейронная сеть. Нейронные сети представляют собой класс гибких моделей, которые в теории могут аппроксимировать практически любые зависимости[3]. Естественной архитектурой, которая похожа на фильтрацию и сглаживание в моделях пространства состояний, является двухсторонняя рекуррентная нейронная сеть (Bidirectional RNN). Эта структура также позволяет для моделей с околостационарным процессом порождения данных[4,5] уйти от оценки модели для каждого состояния по отдельности и вместо этого использовать общие параметры и выполнять расчет функции потерь для всех состояний за один проход по данным.

[3] См. Goodfellow, Bengio and Courville (2016) для введения в нейронные сети и их свойства.
[4] Мы используем термин *околостационарный процесс порождения данных*, чтобы подчеркнуть возможность использований различных распределений состояний для нулевого периода времени.
[5] Если же происходит сдвиг в процессе порождения данных, то всегда можно соединить несколько таких сетей.

Чтобы сделать архитектуру сети более гибкой, мы дополнительно к сырым данным подаем на вход RNN их свертку различной длины, а выход в зависимости от конкретной задачи преобразуем с использованием линейного преобразования либо полносвязной нейронной сети.

3. Применение на практике

Чтобы проиллюстрировать свойства предложенного метода, мы оцениваем три модели: модель стохастической волатильности, нелинейную DSGE-модель и модель сезонной корректировки при наличии сдвига в сезонности.

3.1. Модель стохастической волатильности

Модель стохастической волатильности (см. Kim, Shepard and Chib (1998)) является классическим примером для тестирования различных алгоритмов оценки состояний в моделях пространства состояний (см. Tan, Bhaskaran and Nott (2020)). Спецификация модели в точности соответствует Tan, Bhaskaran and Nott (2020) и представлена в Приложении Б.1. Чтобы немного упростить задачу обучения, в качестве наблюдаемых данных в нейронную сеть подается логарифм модуля данных (полная архитектура сети вместе с алгоритмом обучения представлены в Приложении Б.2). Модель обучается на 20 000 000 сгенерированных серий длиной от 800 до 1200 наблюдений.

Сначала мы продемонстрируем качество оценки на данных, сгенерированных в процессе обучения. На рисунке 1 показаны примеры истинных значений логарифмов волатильностей и средние для аппроксимации их апостериорного распределения (± 2 стандартных отклонения). На рисунке видно, что истинные значения вполне хорошо оцениваются моделью. В качестве бенчмарка для последующих исследований мы также приводим в таблице 1 среднее значение отрицательного логарифма правдоподобия (далее – NLL) и среднеквадратичную ошибку (далее – MSE) на случайно сгенерированном 1 000 000 серий.

К сожалению, оценка соответствующих метрик для других алгоритмов, таких как MCMC или стохастический вариационный байесовский алгоритм, вычислительно трудна (см. дискуссию относительно метрик качества для SBI в Lueckmann et al. (2021)), поэтому мы аналогично Tan, Bhaskaran and Nott (2020) сосредоточимся на сравнении результатов для

датасетов NYSE и GBPUSD[6]. Результаты сравниваются с адаптивным MCMC-алгоритмом на основе аппроксимации хи-квадрат распределения смесью нормальных, предложенной в Kim, Shepard and Chib (1998), и стохастической вариационной гауссовской оценкой (далее – VB) с разреженной матрицей ковариации (оба алгоритма приведены в Приложении Б.3). Как видно из рисунка 2, несмотря на то что оценки с использованием нейронной сети являются немного зашумленными, они достаточно близки к MCMC-алгоритму, который служит золотым стандартом, и к вариационному алгоритму, который является одной из наиболее быстрых и точных аппроксимаций. Отметим также, что NYSE-датасет почти в два раза превышает максимальный размер симуляции и нейронная сеть никогда не видела данных такой длины. Тем не менее обученная модель справляется с этой задачей.

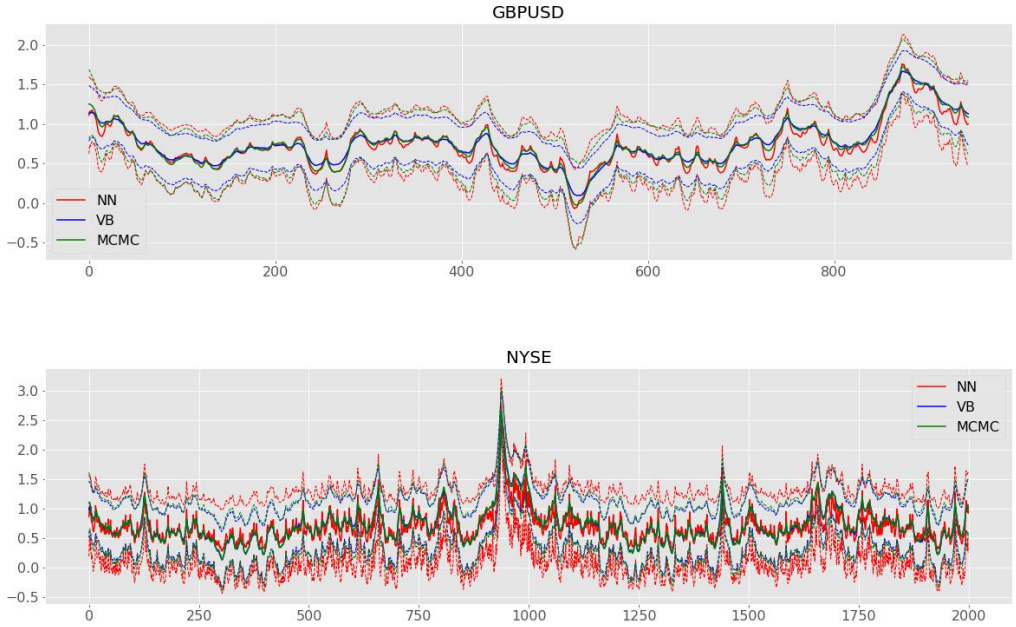
Рисунок 1. Истинные значения логарифмов волатильностей и аппроксимация апостериорного распределения на искусственных данных (среднее ± 2 ст. откл.)

Таблица 1. NLL и MSE для различных приложений (среднее ±2 ст. откл.)

	NLL	MSE
SV	$(8.17 \pm 0.21) \times 10^{-2}$	$(2.92 \pm 0{,}02) \times 10^{-1}$
SV-DSGE	$(-1.64 \pm 0{,}00) \times 10^{-1}$	$(4.77 \pm 0{,}00) \times 10^{-2}$
SA	$-3.00 \pm 0{,}01$	$(1.27 \pm 0{,}02) \times 10^{-3}$

[6] Так как вычисление таких метрик, как C2ST, неинформативно для совместного распределения (информация о корреляциях является важной с точки зрения классификации, а используемый алгоритм предполагает независимость распределения состояний), а расчет их для предельных распределений требует обучения равного количеству скрытых состояний моделей, далее используется только визуальный анализ.

Рисунок 2. Сравнение NPE с MCMC и VB на реальных данных для модели стохастической волатильности (среднее ±2 ст. откл.)



3.2. DSGE-модель со стохастической волатильностью

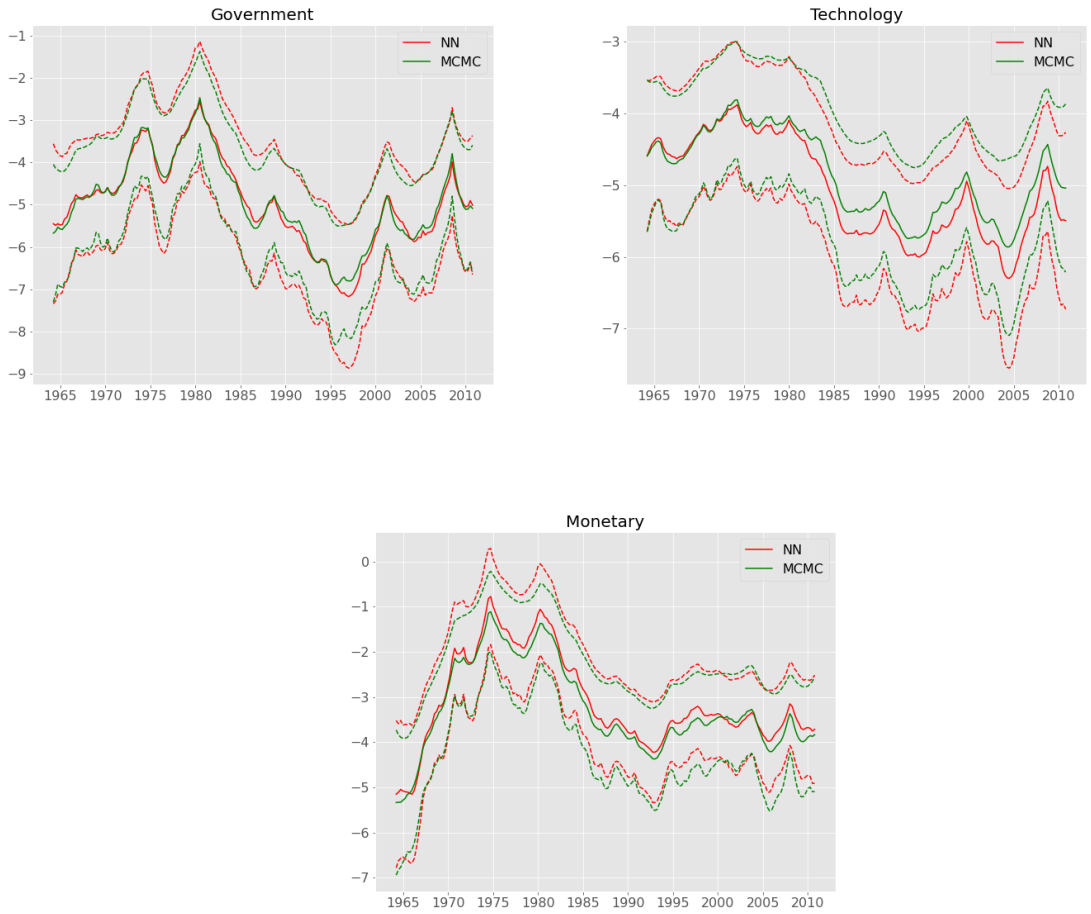
Модель стохастической волатильности, хоть и содержит множество скрытых состояний, тем не менее, является моделью с одним пространственным измерением, как с точки зрения данных, так и с точки зрения цели[7]. Чтобы протестировать SBI-алгоритм в условиях с несколькими пространственными измерениями, была выбрана DSGE-модель. Этот класс моделей широко используется макроэкономистами как в практических целях (см. Linde, Smets and Wouters (2016)), так и в академических исследованиях (см. Walsh (2010)). Обычно ввиду вычислительных сложностей с решением и оценкой нелинейных моделей используются их лог-линеаризованные версии. Решение нелинейных моделей лежит вне рамок этой работы, однако мы хотим продемонстрировать, что предложенный алгоритм хорошо работает и в нелинейных DSGE-моделях, где фильтрация и оценка правдоподобия не могут быть осуществлены с использованием фильтра Калмана. Поэтому была выбрана упрощенная[8] DSGE-модель со стохастической волатильностью из Diebold, Schorfheide and

[7] Хотя потенциально последнее может быть и преимуществом, так как информация от разных состояний может помогать при обучении общих параметров модели. Более того, технически можно обучать свою модель для каждого пространственного измерения, что, по сути, сводит задачу к предыдущей с точки зрения таргета.

[8] Чтобы не касаться вопросов пропущенных переменных и их влияния на результат, мы исключаем из модели шок таргета инфляции и инфляционные ожидания из наблюдаемых переменных. Ряд дополнительных экспериментов показал, что использование обучаемых значений для входов нейронной сети на месте пропущенных переменных (см. Lueckmann et al. (2017)) и добавление дополнительных дамми-переменных на вход рекуррентной сети способны справляться с этой задачей. Однако, чтобы отделить эффект множества наблюдаемых переменных от эффекта пропущенных

Shin (2017). Решение этой модели может быть легко получено с помощью стандартных алгоритмов линеаризации (см. Blanchard and Kahn (1980), Anderson an Moore (1985), Klein (2000), Sims (2002)) с последующим добавлением нелинейности при оценке модели. Модель оценивается на 50 000 000 сгенерированных наборах данных длиной от 180 до 200 точек и сравнивается с адаптивным MCMC-алгоритмом (описание модели, архитектура нейронной сети и реализация MCMC приведены в Приложении B).

Рисунок 3. Сравнение NPE с MCMC на данных США для DSGE-модели со стохастической волатильностью, стохастическая волатильность (среднее ±2 ст. откл.)



Для иллюстрации мы сконцентрировались на оценке стохастических волатильностей и приводим графики и метрики именно для этих состояний. Однако ненаблюдаемые шоки (точнее логарифмы их модулей) также были использованы нами при оценке в качестве подсказки на промежуточных выходах нейронной сети. На рисунке 3 видно, что на данных по США с 1964Q2 по 2011Q1 (последний винтаж в Diebold, Schorfheide and Shin (2017))

данных, мы сконцентрируемся здесь на более простой версии модели. Оценкам DSGE-моделей с помощью SBI будет посвящена отдельная работа, где мы также затронем и этот вопрос.

обученная нейронная сеть показывает близкие к MCMC результаты. Как и для предыдущей модели, для сравнений с нашей работой в будущем в таблице 1 приведены NLL и MSE на 1 000 000 случайно сгенерированных датасетов.

3.3. Модель сезонной корректировки со структурными сдвигами в сезонности

Помимо задач, в основе которых лежат хорошо выверенные формулы для уравнений перехода и уравнений наблюдений, SBI подходит и для тех моделей, где во главе угла стоят симуляции, а запись уравнений хоть и возможна, но слишком громоздка либо же совсем вычислительно невозможна. К таким случаям, например, можно отнести задачи, где легко сгенерировать множество различных данных, подсказывая модели, как она должна вести себя в тех или иных ситуациях.

В качестве примера мы покажем, как может быть построена модель сезонной корректировки на квартальных данных, которая учитывает структурные сдвиги в сезонной компоненте. Как будет показано ниже, традиционные процедуры сезонной корректировки в стиле X13-ARIMA-SEATS (см. US Census Bureau (2017)) плохо справляются с такой задачей. Дополнительным бонусом может служить наличие автоматически генерируемых доверительных интервалов.

В Приложении Г представлена процедура генерации искусственных серий длиной от 40 до 80 кварталов. По сути, она состоит из генерации сезонной и несезонной компонент с вероятностью появления сдвига в сезонной части. Таким образом, получающиеся серии могут как содержать сдвиг, так и не иметь его.

В этом подразделе мы при оценке качества, в отличие от двух прошлых моделей, сравниваем NPE с X13, а не с семплинговым алгоритмом. Целью этого эксперимента является демонстрация того, как можно легко сгенерировать примеры поведения модели, задав таким образом неявно байесовскую модель. На практике в таких случаях, как и в нашем примере, построение быстрого MCMC-алгоритма обычно оказывается затруднительным. Сравнение же с другими алгоритмами, решающими ту же практическую задачу, тем не менее, представляет интерес с точки зрения оценки качества выполнения поставленной задачи.

На рисунке 4 приведены случайные примеры, показывающие, как ведут себя предложенная процедура и X13 на сериях со сдвигом в сезонности (серая зона показывает 1,5 года в обе стороны от точки сдвига). X13 не справляется адекватно с задачей выделения сезонности в окрестности сдвигов, NPE же, наоборот, робастен. Мы посчитали MSE на 100 000 случайно сгенерированных сериях для X13 и на 1 000 000 серий для нейронной сети. Как видно из таблицы 2, ошибка для NPE меньше. Но сами по себе меньшие ошибки по сравнению с X13 не являются неожиданностью, так как оптимизация кросс-энтропии должна

в точности давать модель оценки с минимальной среднеквадратичной ошибкой на этом датасете. Однако разрыв между ошибками на сериях со сдвигами и без еще раз подчеркивает, что предложенный в работе алгоритм справляется с задачей намного лучше широко распространенных среди макроэкономистов альтернатив.

Рисунок 4. Сравнение NPE и X-13-ARIMA-SEATS на сериях со сдвигом в сезонности (среднее ± 2 ст. откл.)

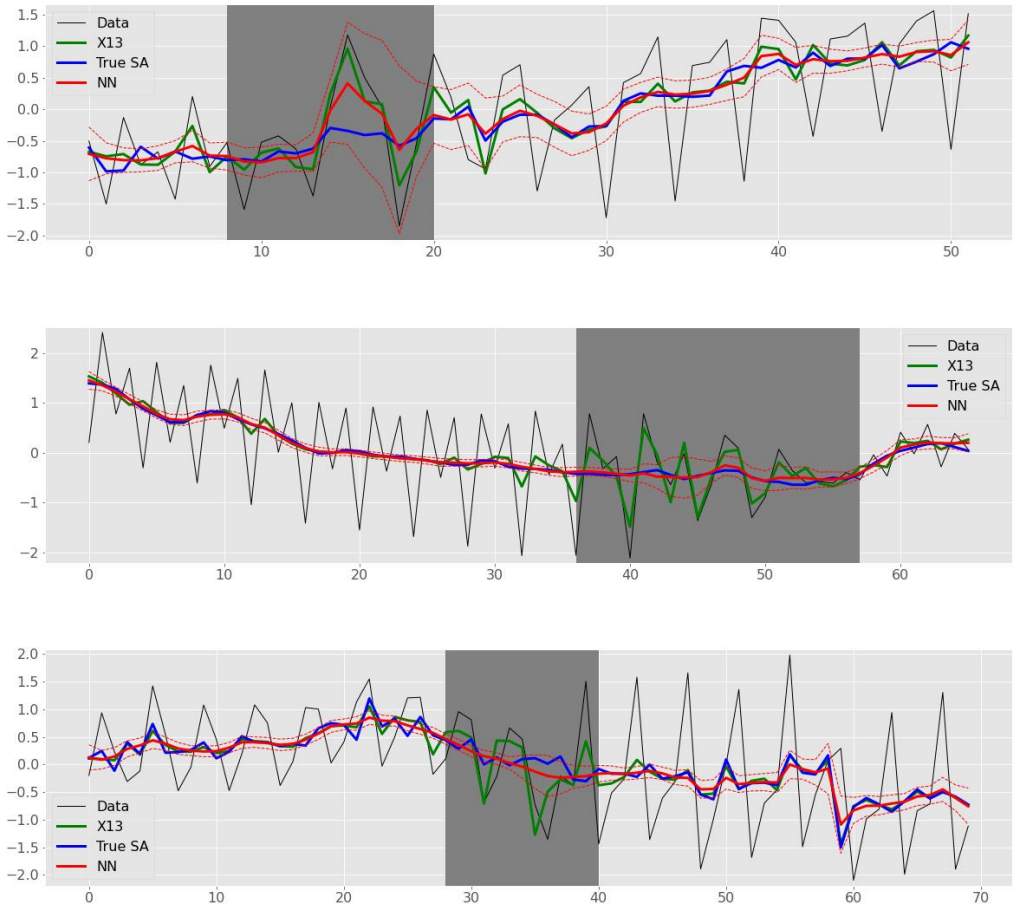


Таблица 2. Среднеквадратичные ошибки моделей сезонной корректировки для NPE и X13-ARIMA-SEATS на искусственных данных

	Full sample	With shifts	Without shifts
NPE	1.3×10^{-3}	3.0×10^{-3}	1.1×10^{-3}
X13-ARIMA-SEATS	5.4×10^{-3}	25.0×10^{-3}	3.1×10^{-3}

3.4. Время работы и имплементации

Как отмечалось выше, обладая свойством амортизации, NPE работает практически мгновенно. Построение аппроксимации апостериорного распределения для предобученной модели на CPU (Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, 16GB RAM) занимает десятые секунды. Оценка параметров нейронных сетей на Pytorch[9] (Paszke et al. (2019)) с использованием GPU (NVIDIA GeForce RTX 2070) длится около 12, 12 и 2 часов для модели стохастической волатильности, DSGE-модели и модели сезонной корректировки соответственно. Сравнение времени работы для амортизированных алгоритмов с альтернативами, которые не обладают такими свойствами, сталкивается с некоторыми трудностями. С одной стороны, в наших примерах оценка байесовской модели для фиксированного датасета занимает меньше времени, если используются MCMC- или VB-алгоритмы (см. таблицу 3), так как для амортизированного алгоритма требуется предварительное обучение нейронной сети. С другой стороны, обученная единожды нейронная сеть на практике может быть использована для различных наборов данных (в том числе данных разной длины), что является преимуществом в случае многократных обучений.

Таблица 3. Время оценки модели на одном наборе данных[10]

	NPE		VB		MCMC	Other
	CPU	GPU	CPU	GPU	CPU	CPU
SV	0.42 с	0.08 с	1 ч 27 м	16 м	19 м	–
SV-DSGE	0.14 с	0.02 с	–	–	9 ч 18 м	–
SA	0.14 с	0.05 с	–	–	–	0.27 с

Имплементация NPE-алгоритма практически всегда оказывается куда проще, чем имплементация MCMC и даже вариационной оценки, ведь для построения NPE-алгоритма требуется лишь написать генератор данных и воспользоваться уже готовыми пакетами оценки нейронных сетей (написание кода для архитектуры сети обычно занимает лишь десятки минут ввиду того, что слои нейронной сети уже реализованы в соответствующих библиотеках). Для реализации же MCMC требуются вывод алгоритма семплирования и написание программного кода, который зачастую куда сложнее, чем для NPE. Стохастические вариационные алгоритмы, как и NPE, так же просты для имплементации (если использовать готовые пакеты автоматического дифференцирования и не нужно писать код для дифференцирования

[9] Генерация искусственных данных для модели происходит на CPU, а обучение – на GPU.
[10] Для MCMC мы выбираем количество итераций исходя из сходимости параметров DSGE-модели (половина начальных итераций удаляется), а для VB – исходя из сходимости функции потерь.

вручную), только вместо процедуры семплирования данных используется совместная плотность параметров, скрытых состояний и данных.

Важным с практической точки зрения моментом для стохастических оптимизационных алгоритмов (NPE и VB) также является возможность почти что без усилий перенести вычисления на GPU.

4. Похожие направления исследований

Предложенный в работе алгоритм тесно связан с несколькими направлениями в литературе. Наша работа является частью литературы по SBI-алгоритмам (SBI – likelihood-free inference) (см. Crammer, Brehmer and Louppe (2020)). Байесовское направление в этой области до недавнего времени развивалось в основном как приближенные байесовские вычисления (approximate Bayesian computations – ABC), которые аппроксимируют функции правдоподобия путем введения вспомогательной функции правдоподобия, зависящей от расстояния между некоторыми характеристиками данных и соответствующими характеристиками симулированных данных, а затем используют стандартные алгоритмы семплирования (см. Sisson, Fan and Beaumont (2018)). Развитие алгоритмов машинного обучения и в особенности нейронных сетей поспособствовало появлению целого семейства алгоритмов, которые напрямую обучают апостериорные распределения (см. Papamakarios and Murray (2016), Lueckmann et al. (2017), Greenberg, Nonnenmacher and Macke (2019), Durkan, Murray and Papamakarios (2020)), функции правдоподобия (см. Wood (2010), Lueckmann et al. (2019), Brehmer et al. (2020), Papamakarios, Sterratt and Murray (2019)) или отношение функций правдоподобия (см. Brehmer et al. (2020), Hermans, Begy and Louppe (2020), Durkan, Murray and Papamakarios (2020)) и в отличие от ABC-алгоритмов не чувствительны к гиперпараметру точности аппроксимации (tolerance) и метрикам близости между характеристиками симуляций и данных. Однако, как нам известно, за исключением нескольких работ по вероятностному программированию (см. Le, Baydin and Wood (2017), Baydin et al. (2019), Munk et al. (2022)), исследователи концентрируются в основном на параметрах, а не на пространствах состояний. Работы же по вероятностному программированию имеют два ключевых отличия от подхода, предложенного в данной статье. Во-первых, они используют обученную нейронную сеть в качестве вспомогательного распределения для алгоритма выборки по значимости, а не напрямую для аппроксимации апостериорного распределения. Во-вторых, нейронная сеть в вероятностном программировании моделирует зависимость между переменными, чтобы веса алгоритма выборки по значимости имели меньший разброс,

что значительно сложнее с точки зрения оптимизации[11]. Более того, в отличие от нашей работы реализация с нуля или модификация алгоритмов вероятностного программирования под задачи, не укладывающиеся в рамки стандартных пакетов[12], весьма сложна, так как требует достаточно глубоких знаний об адресации случайных переменных.

Наша работа также тесно связана с симуляционной оценкой экономических моделей. Симуляционный метод моментов и его модификации (см. McFadden (1989), Duffy and Singleton (1993), Gallant and Tauchen (1966)) распространены при частотной оценке структурных параметров моделей[13]. Существует похожее направление работ, которые оценивают параметры моделей на основе минимизации различного рода дивергенций между симулированными и реальными данными (см., например, Nickl and Pötscher (2010), Kaji, Manresa and Pouliot (2022)). Gallant and McCalloch (2009)[14] предложили байесовскую версию симуляционного метода моментов. Недавняя работа Fen (2022) для байесовской оценки параметров использует последовательный (не амортизированный) NPE. Симуляционной оценке не параметров, а состояний, как и в SBI, посвящено не так много работ. Наиболее близкой из нам известных к этой работе является статья Deli Gatti and Grazzini (2020), где авторы оценивают состояния (прогнозы разрывов выпуска и инвестиций) на искусственных данных с использованием непараметрической ядерной оценки. Этот метод идеологически очень близок к SBI и тому, что предлагается в нашей статье, но, как отмечают сами авторы, вычислительно сложен при большом количестве скрытых состояний.

Метаобучение (см. Finn and Levine (2019)) похоже на SBI по математической формулировке. Как и в SBI, в основе метаобучения лежит идея обучения на множестве похожих заданий (см. Vinyals et al. (2016)). Ключевыми отличиями служат цель и данные. В отличие от SBI метаобучение концентрируется на задаче прогнозирования, а не на нахождении апостериорного распределения. Также метаобучение обычно работает с реальными данными, а не симулированными.

Амортизации нахождения распределения скрытых состояний посвящено множество работ по вариационным автоэнкодерам (см. Kingma and Welling (2019)). Однако существует

[11] Также стоит отметить, что вероятностное программирование использует семплирование состояний, которое зависит от семплированных состояний предыдущего периода, что может приводить к накоплению ошибок аппроксимации на длинных периодах. Такая архитектура не совсем подходит для прямой аппроксимации, однако это не критично при последующем ресемплинге, особенно если вместо алгоритма выборки по значимости использовать его последовательный аналог.
[12] См., например, PyProb.
[13] См. список приложений из Carrasco and Florens (2002).
[14] Gallant, Giacomini and Ragusa (2013) также разработали версию байесовского симуляционного метода моментов на основе фильтра частиц для моделей со скрытыми состояниями.

ряд отличий от этой работы. Во-первых, модель порождения данных обычно задается с помощью достаточно гибкой модели, такой как нейронная сеть (см. Kingma and Welling (2014)) или гауссовский процесс (см. Dai et al. (2016)), а не более классических моделей, где пространство скрытых состояний обладает б*о*льшей идентифицируемостью и интерпретируемостью. Кроме того, модель обычно имеет не байесовский характер[15]. Во-вторых, функция потерь, которая минимизируется, – дивергенция Кульбака – Лейблера между приближенным апостериорным и апостериорным распределением, в то время как в SBI – это дивергенция Кульбака – Лейблера между апостериорным и приближенным апостериорным распределением. Несимметричность дивергенции Кульбака – Лейблера приводит к тому, что за редким исключением (см. Tran, Ranganath and Blei (2017)) для обучения вариационных автоэнкодеров недостаточно симуляций и приходится рассчитывать вероятности для модели порождения данных. Также в случае диагональной аппроксимации (как в разделе 2.3) это приводит к недооценке дисперсии (см. Blei, Kucukelbir and McAuliffe (2018)). В-третьих, для обучения вариационных автоэнкодеров, как и для метаобучения, используются реальные, а не искусственные данные.

5. Дискуссия

Как было показано во многих работах (см. Lueckmann et al. (2021)), амортизация приводит к необходимости более длительного обучения SBI-алгоритмов, чем их последовательных аналогов. Несмотря на это, мы используем именно амортизированный NPE-алгоритм по двум причинам. Во-первых, обычно последовательные SBI-алгоритмы работают с задачами небольшой размерности (с точки зрения таргета) и их адаптация к высокоразмерной задаче нахождения апостериорного распределения состояний нетривиальна и требует решения большего количества практических трудностей. В частности, если попробовать сконцентрироваться на предельных плотностях, как делается в этой статье, сгенерированные для новых раундов состояния будут непохожи на апостериорное распределение из-за отсутствия зависимости между отдельными измерениями (состояния будут достаточно шумны), что в большинстве случаев будет ухудшать сходимость. Во-вторых, в отличие от некоторых работ по SBI мы не ставим задачу найти наилучший алгоритм

[15] В основном в качестве модели порождения данных используются нейронные сети, которые по своей природе относятся к частотным моделям. Более того, несмотря на то что в большинстве работ для регуляризации применяют метод прореживания (см. Srivastava et al. (2014)), который имеет байесовскую интерпретацию (см. Kingma, Salimans and Welling (2015) и Gal and Ghahramani (2016)), параметры являются общими для всех данных, что идеологически отличается от идеи амортизации моделей.

при ограничении на количество симуляций (см. Lueckmann et al. (2021)), где последовательные алгоритмы дают значительный выигрыш. Наша основная задача – построение алгоритма, который бы заменил долго работающие альтернативы (как в первых двух примерах) либо же помогал оценивать модели там, где другие алгоритмы не справляются (как в третьем примере)[16]. Амортизация является отличным свойством, которое в случае частого переиспользования модели помогает решить эту задачу.

Множество вопросов касательно оценки распределения состояний модели остались за рамками этой статьи и требуют дальнейшего исследования. Некоторые из них обсуждаются ниже.

Оцененные с помощью предложенного в данной работе алгоритма апостериорные распределения, хоть и близки к результатам MCMC, но, тем не менее, немного отличаются. В задаче оценки модели стохастической волатильности результаты чуть зашумлены, а в DSGE-модели смещены. Это сигнализирует о возможности дальнейшего улучшения нейронных сетей за счет повышения гибкости архитектуры нейронной сети, количества примеров или процедуры обучения. Как известно, процедуры стохастической оптимизации сходятся к одному из локальных оптимумов и только в асимптотике и при определенном расписании на скорость обучения (см. главу 5 из Kushner and Yin (2003)). При конечном же числе итераций (и, как следствие, скорости обучения, не стремящейся к нулю) точный (даже локальный) оптимум не достигается[17]. Недостаточно же гибкая сеть и небольшое количество наблюдений в окрестности реальных данных могут приводить к тому, что модель в принципе неспособна хорошо предсказывать апостериорное распределение даже в оптимуме[18]. Более того, при прочих равных качество аппроксимации апостериорного распределения наверняка будет ухудшаться с увеличением размерности задач, поэтому одной из основных задач на будущее видится изучение взаимосвязи масштабируемости, качества аппроксимации и времени оценки.

Отсутствие зависимости между переменными и факторизуемая гауссовская аппроксимация, которые рассматриваются в статье, обычно не проблема с практической точки зрения, ведь в большинстве случаев исследователям интересны первые и вторые

[16] При этом неявно предполагается, что большое количество симуляций модели может быть выполнено за адекватное время.
[17] Mandt, Hoffman and BLei (2017) дают интуицию относительно поведения процедуры оценки при конечных скоростях обучения.
[18] Ярким примером такого улучшения в области текстового анализа может служить модель GPT-3 (Brown et al. (2020)), которая за счет на порядок большего, чем использовалось ранее, числа параметров и огромного датасета вышла на принципиально новый уровень по сравнению с предыдущими моделями.

моменты предельных распределений состояний. Несмотря на то что оба вопроса имеют понятные теоретические решения (М-оценки для исследования других характеристик и более гибкие семейства распределений), их практическая реализация требует дальнейших исследований[19].

Мы обошли стороной вопросы прогнозирования и пропущенных переменных, которые связаны между собой в том смысле, что задача прогнозирования может рассматриваться как задача построения апостериорного распределения для пропущенных переменных на горизонте прогнозирования. Для работы с пропущенными переменными модели могут быть расширены путем введения в качестве одного из входов нейронной сети дополнительных дамми-переменных, показывающих наличие пропуска, и/или заполнением пропусков обучаемыми параметрами, как это сделано в работе Lueckmann et al. (2017). Подобный метод или же альтернатива, основанная на идеях метаобучения (см. Harrison, Sharma and Pavone (2020)), где в качестве выхода модели-оценщика используются только прогнозируемые переменные, могут быть применены для построения моделей прогноза.

Как показано в разделе 3.4, у NPE с точки зрения скорости работы есть как свои недостатки, так и преимущества. Выбор, использовать NPE или нет, должен зависеть от ситуации. Мы советуем использовать NPE-алгоритм, если планируется частая переоценка модели или альтернативные алгоритмы медленны либо же совсем не справляются с задачей. При этом мы также советуем перед использованием проводить верификацию обученного алгоритма путем сравнения работы алгоритма с альтернативными алгоритмами аппроксимации апостериорного распределения (когда они не слишком медленны), а в случае если такая проверка невозможна, хотя бы визуальный анализ на искусственно сгенерированных данных.

6. Заключение

Предложенный в данной работе амортизированный симуляционный алгоритм для оценки скрытых состояний байесовских моделей пространства состояний представляет альтернативу для уже существующих алгоритмов в этой области. В отличие от предыдущих работ мы рассматриваем принципиально новый подход, который аппроксимирует предельные апостериорные распределения и основывается не на использовании функций

[19] В ряде предварительных экспериментов, которые не вошли в работу, мы увидели, что для предельных распределений квантильные функции потерь также показывают неплохие результаты.

плотности вероятности для априорных распределений, уравнений перехода и наблюдений, а использует лишь симуляции искусственных данных.

Для модели стохастической волатильности и DSGE-модели NPE-алгоритм показывает близкие к другим алгоритмам результаты, однако после обучения работает практически мгновенно. Помимо этого, как показано в примере с сезонной корректировкой, он также хорошо справляется с задачами, где байесовская модель задается не напрямую, а путем симуляций различного рода ситуаций и правильного поведения в них.

Литература

Anderson, G. and G. Moore (1985). A Linear Algebraic Procedure for Solving Linear Perfect Foresight Models. *Economics Letters*, 17(3): 247–252.

Andrieu, C., A. Doucet and R. Holenstein (2010). Particle Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society Series B*, 72(3): 269–342.

Baydin, A.G., L. Shao, W. Bhimji, L. Heinrich, L.F. Meadows, J. Liu, A. Munk, S. Naderiparizi, B. Gram-Hansen, G. Louppe, M. Ma, X. Zhao, P. Torr, V. Lee, K. Cranmer, Prabhat and F. Wood (2019). Etalumis: Bringing Probabilistic Programming to Scientific Simulators at Scale. *In Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC19)*.

Beal, M.J. (2003). Variational Algorithms for Approximate Bayesian Inference. *PhD Thesis, Gatsby Computational Neuroscience Unit, University College London*.

Beaumont, M.A., W. Zhang and D.J. Balding (2002). Approximate Bayesian Computation in Population Genetics. *Genetics*, 162(4): 2025–2035.

Blanchard, O.J. and C.M. Kahn (1980). The Solution of Linear Difference Models under Rational Expectations. *Econometrica*, 48(5): 1305–1311.

Blei, D.M., A. Kucukelbir and J.D. McAuliffe (2018). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518): 859–877.

Blum, M.G.B. and O. François (2010). Non-linear Regression Models for Approximate Bayesian Computation. *Statistics and Computing*, 20: 63–73.

Brehmer, J., G. Louppe, J. Pavez, and K. Cranmer (2020). Mining Gold from Implicit Models to Improve Likelihood-Free Inference. *Proceedings of the National Academy of Sciences*, 117(10): 5242–5249.

Brown, T.B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D.M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei (2020). Language Models are Few-Shot Learners. *arXiv:2005.14165v4*.

Carrasco, M. and J.-P. Florens (2002). Simulation-Based Method of Moments and Efficiency. *Journal of Business and Economic Statistics*, 20(4): 482–492.

Carriero, A., T.E. Clark and M. Massimiliano (2019). Large Bayesian Vector Autoregressions with Stochastic Volatility and Non-Conjugate Priors. *Journal of Econometrics*, 212(1): 137–154.

Casella, G. and E.I. George (1992). Explaining the Gibbs Sampler. *The American Statistician*, 46: 167–174.

Chib, S. and E. Greenberg (1995). Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4): 327–335.

Chiu, C.W., B. Eraker, A.T. Foerster, T.B. Kim and Hernan D. Seoane (2012). Estimating VAR's Sampled at Mixed or Irregular Spaced Frequencies: a Bayesian Approach. *Federal Reserve Bank of Kansas City RWP*, 11–11.

Chopin, N., P.E. Jacob and O. Papaspiliopoulos (2012). SMC^2: An Efficient Algorithm for Sequential Analysis of State Space Models. *Journal of the Royal Statistical Society Series B*, 75(3): 397–426.

Cranmer, K., J. Brehmer and G. Louppe (2020). The Frontier of Simulation-Based Inference. *Proceedings of the National Academy of Sciences*, 117(48): 30055–30062.

Dai, Z., A. Damianou, J. González and N. Lawrence (2016). Variational Auto-encoded Deep Gaussian Processes. *International Conference on Learning Representations*.

Del Moral, P., A. Doucet and A. Jasra (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society Series B*, 68(3): 411–436.

Deli Gatti, D. and J. Grazzini (2020). Rising to the Challenge: Bayesian Estimation and Forecasting Techniques for Macroeconomic Agent Based Models. *Journal of Economic Behavior and Organization*, 178: 875–902.

Diebold, F.X., F. Schorfheide and M. Shin (2017). Real-time Forecast Evaluation of DSGE Models with Stochastic Volatility. *Journal of Econometrics*, 201(2): 322–332.

Duffie, D. and K. Singleton (1993) Simulated Moments Estimation of Markov Models of Asset Prices. *Econometrica*, 61: 929–1052.

Durbin J. and S.J. Koopman (2002). A Simple and Efficient Simulation Smoother for State Space Time Series Analysis. *Biometrika*, 89(3): 603–615.

Durkan, C., I. Murray and G. Papamakarios (2020). On Contrastive Learning for Likelihood-Free Inference. *In Proceedings of the 36th International Conference on Machine Learning*.

Fen, C. (2022). Fast Simulation-Based Bayesian Estimation of Heterogeneous and Representative Agent Models using Normalizing Flow Neural Networks. *arXiv:2203.06537v1*.

Fernandez-Villaverde, J., J.F. Rubio-Ramirez and F. Schorfheide (2016). Solution and Estimation Methods for DSGE Models. *Handbook of Macroeconomics*, 2: 527–724.

Finn, C. and S. Levine (2019). Meta-Learning: from Few-Shot Learning to Rapid Reinforcement Learning. *The International Conference on Machine Learning*, Tutorial.

Gal, Y. and Z. Ghahramani (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *Neural Information Processing Systems*.

Gallant, A.R., R. Giacomini and G. Ragusa (2013). Generalized Method of Moments with Latent Variables. *CEPR Discussion Papers*, DP9692.

Gallant, A.R. and R.E. McCulloch (2009). On the determination of general statistical models with application to asset pricing. *Journal of the American Statistical Association*, 104: 117–131.

Gallant, A.R. and G. Tauchen (1996). Which moments to match? *Econometric Theory*, 12: 657–681.

Goodfellow, I., Y. Bengio and A. Courville (2016). Deep Learning. *MIT Press*.

Greenberg, D., M. Nonnenmacher, and J. Macke (2019). Automatic Posterior Transformation for Likelihood-Free Inference. *In Proceedings of the 36th International Conference on Machine Learning*.

Gunawan, D., R. Kohn and D. Nott (2021). Variational Bayes Approximation of Factor Stochastic Volatility Models. *International Journal of Forecasting*, 37(4): 1355–1375.

Hamilton, J, (1989). A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. *Econometrica*, 57(2): 357–384.

Harrison, J., A. Sharma and M. Pavone (2020). Meta-Learning Priors for Efficient Online Bayesian Regression. *Algorithmic Foundations of Robotics XIII*, 14: 318–337.

Hermans, J., V. Begy and G. Louppe (2020). Likelihood-Free MCMC with Approximate Likelihood Ratios. *In Proceedings of the 37th International Conference on Machine Learning*.

Hodrick, R. and E. Prescott (1997). Postwar U.S. Business Cycles: An Empirical Investigation. *Journal of Money, Credit and Banking*, 29(1): 1–16.

Hoffman, M.D, D.M. Blei, C. Wang and J. Paisley (2013). Stochastic Variational Inference. *Journal of Machine Learning Research*, 14(1): 1303–1347.

Justiniano, A. and G.E. Primiceri (2008). The Time-Varying Volatility of Macroeconomic Fluctuations. American Economic Review, 98(3): 604–641.

Kaji, T., E. Manresa and G. Pouliot (2022). An Adversarial Approach to Structural Estimation. *arXiv:2007.06169v2*.

Kim, S., N. Shepard and S. Chib (1998). Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models. *The Review of Economic Studies*, 65(3): 361–393.

Kingma, D.P. and J. Ba (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

Kingma, D.P., T. Salimans and M. Welling (2015). Variational Dropout and the Local Reparametrization Trick. *In Advances in Neural Information Processing Systems*, 2575–2583.

Kingma, D.P. and M. Welling (2014). Auto-Encoding Variational Bayes. *International Conference on Learning Representations*.

Kingma, D.P. and M. Welling (2019). An Introduction to Variational Autoencoders. *Foundations and Trends in Machine Learning*, 12(4): 307–392.

Klein, P. (2000). Using the Generalized Schur Form to Solve a Multivariate Linear Rational Expectations Model. *Journal of Economic Dynamics and Control*, 24(10): 1405–1423.

Koop, G. and D. Korobilis (2012). Forecasting Inflation Using Dynamic Model Averaging. *International Economic Review*, 53(3): 867–886.

Kushner, H.J. and G.G. Yin (2003). Stochastic Approximation Algorithms and Recursive Algorithms and Applications. *Springer Science and Business Media*.

Laubach, T. and J.C. Williams (2003). Measuring the Natural Rate of Interest. *The Review of Economics and Statistics*, 85(4): 1063–1070.

Le, T.A., A.G. Baydin and F. Wood (2017). Inference compilation and universal probabilistic programming. *In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Linde, J., F. Smets and R. Wouters (2016). Challenges for Central Banks' Macro Models. *Handbook of Macroeconomics*, 2: 2185–2262.

Lueckmann, J.-M., G. Bassetto, T. Karaletsos and J.H. Macke (2019). Likelihood-free Inference with Emulator Networks. *In Proceedings of the 1st Symposium on Advances in Approximate Bayesian Inference.*

Lueckmann, J.-M., J. Boelts, D.S. Greenberg, P.J. Gonçalves and J.H. Macke (2021). Benchmarking Simulation-Based Inference. *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Lueckmann, J.-M., P.J. Goncalves, G. Bassetto, K. Öcal, M. Nonnenmacher and J.H. Macke (2017). Flexible Statistical Inference for Mechanistic Models of Neural Dynamics. *In Advances in Neural Information Processing Systems*, 30: 1289–1299.

Lux, T. (2018). Estimation of Agent-Based Models Using Sequential Monte Carlo Methods. *Journal of Economic Dynamics and Control*, 91: 391–408.

Mandt, S., M.D. Hoffman and D.M. Blei (2017). Stochastic Gradient Descent as Approximate Bayesian Inference. *Journal of Machine Learning Research*, 18: 1–35.

McFadden, D. (1989). A Method of Simulated Moments for Estimation of Discrete Response Models Without Numerical Integration. *Econometrica*, 57(5): 995–1026.

Munk, A., B. Zwartsenberg, A. Scibior, A.G. Baydin, A.L. Stewart, G. Fernlund, A. Poursartip and F. Wood (2022). Probabilistic Surrogate Networks for Simulators with Unbounded Randomness. *arXiv:1910.11950v2*.

Neal, R.M. (1996). Bayesian Learning for Neural Networks. *Springer-Verlag, Lecture Notes in Statistics*, № 118.

Nickl, R. and B.M. Pötscher (2010). Efficient Simulation-Based Minimum Distance Estimation and Indirect Inference. *Mathematical Methods of Statistics*, 19: 327–364.

Orphanides, A. and S. Van Norden (2002). The Unreliability of Output-gap Estimates in Real Time. *Review of Economics and Statistics*, 84(4): 569–583.

Otrok, C. and C.H. Whiteman (1998). Bayesian Leading Indicators: Measuring and Predicting Economic Conditions in Iowa. *International Economic Review*, 39(4): 997–1014.

Papamakarios, G. and I. Murray (2016). Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation. *In Advances in Neural Information Processing Systems*, 29: 1028–1036.

Papamakarios, G., D. Sterratt and I. Murray (2019). Sequential Neural Likelihood: Fast Likelihood-Free Inference with Autoregressive Flows. *In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *In Advances in Neural Information Processing Systems*, 32: 8024–8035.

Primiceri, G.E. (2005). Time Varying Structural Vector Autoregressions and Monetary Policy. *The Review of Economic Studies*, 72(3): 821–852.

Rezende, D. and S. Mohamed (2015). Variational Inference with Normalizing Flows. *In Proceedings of the 32nd International Conference on Machine Learning*.

Schorfheide, F. and D. Song (2015), Real-Time Forecasting with a Mixed-Frequency VAR, *Journal of Business and Economic Statistics*, 33(3): 366–380.

Schorfheide, F. and D. Song (2021). Real-Time Forecasting with a (Standard) Mixed-Frequency VAR During a Pandemic. *NBER Working Papers*, № 29535.

Sims, C.A. (2002). Solving Linear Rational Expectations Models. *Computational Economics*, 20: 1–20.

Sisson, S.A., Y. Fan, M. Beaumont (2018). Handbook of Approximate Bayesian Computation. *Chapman and Hall/CRC*.

Smets, F. and R. Wouters (2003). An Estimated Dynamic Stochastic General Equilibrium Model of the Euro Area. *Journal of the European Economic Association*, 1(5): 1123–1175.

Smets, F. and R. Wouters (2007). Shocks and Frictions in US Business Cycles: A Bayesian DSGE Approach. *American Economic Review*, 97(3): 586–606.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15: 1929–1958.

Stock, J.H. and M.W. Watson (2011). Dynamic Factor Models. *In Clements M.J. and D.F. Hendry Oxford Handbook on Economic Forecasting*.

Tan, L., A. Bhaskaran and D. Nott (2020). Conditionally Structured Variational Gaussian Approximation with Importance Weights. *Statistics and Computing*, 30(5).

Tan, L. and D. Nott (2018). Gaussian Variational Approximation with Sparse Precision Matrix. *Statistics and Computing*, 28(2): 259–275.

Tran, D., R. Ranganath and D.M. Blei (2017). Hierarchical Implicit Models and Likelihood-Free Variational Inference. *Neural Information Processing Systems*.

US Census Bureau (2017). X-13 ARIMA-SEATS Reference Manual. *US Census Bureau*.

Van der Vaart, A.W. (2000). Asymptotic Statistics. *Cambridge University Press*.

Vinyals, O., C. Blundell, T. Lillicrap, K. Kavukcuoglu and D. Wierstra (2016). Matching Networks for One Shot Learning. *In Advances in Neural Information Processing Systems*.

Wainwright, M.J. and M. Jordan (2008). Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1–2): 1–305.

Walsh C.E. (2010). Monetary Theory and Policy. *MIT Press*, 3rd Edition.

Wood, S.N. (2010). Statistical Inference for Noisy Nonlinear Ecological Dynamic Dystems. Nature, 466(7310): 1102–1104.

Приложение А. Неформальные доказательства

А1. Асимптотика NPE

При гибком семействе параметров $q_\varphi(\theta|y)$ и стремящемся к бесконечности количестве наборов данных задача (1) становится эквивалентна следующей задаче:

$$q^* = argmin_q E_{p(\theta,y)}(-\log q(\theta|y) - \log p(y)) = argmin_q \int\int(-\log q(\theta|y) - \log p(y))p(x,\theta)d\theta dy = argmin_q \int(-\int \log q(\theta|y)\, p(\theta|y)d\theta)\, p(y)d.$$

Заметим, что теперь задача оптимизации разделяется на множество отдельных минимизаций кросс-энтропии для каждого отдельного набора данных $-\int log\, q(\theta|y)\, p(\theta|y)d\theta$. Минимум кросс-энтропии достигается при совпадении распределений, и это означает, что

$$q^* = p(\theta|y).$$

А2. NPE для состояний

Единственное, что нужно доказать для валидности алгоритма 1, что совместное распределение семплированных состояний и данных является предельным распределением процесса порождения данных. Тогда, переобозначая s через θ, мы можем воспользоваться результатами Приложения А.1.

Семпл θ_i, s_i, y_i в результате алгоритма 1 по построению в точности совпадает с семплом из процесса порождения данных, то есть $\theta_i, s_i, y_i \sim p(\theta, s, y)$. Интегрируя по θ, получаем что $s_i, y_i \sim p(s, y)$ – предельное распределение процесса порождения данных.

А3. NPE для функции потерь на основе предельных распределений

Рассмотрим функцию потерь для М-оценки $m(x, s, y)$, где, как и ранее, $y -$ набор наблюдаемых переменных, $s -$ набор скрытых состояний, а $x -$ набор характеристик апостериорного распределения, которые оцениваются. Предположим, что $f_\varphi(y) -$ параметрическое семейство функций, которое в соответствие каждому набору данных ставит оценку характеристик апостериорного распределения. Тогда аналогично Приложению А.1 при гибкой функции f и стремящемся к бесконечности количестве симуляций получаем:

$$f^* = argmin_f E_{p(s,y)} m(f(y), s, y) = argmin_f \int(\int m(f(y), s, y)p(s|y)ds)\, p(y)dy.$$

Как и в Приложении А.1, задача распадается на множество минимизаций для отдельных наборов данных $\int m(f(y), s, y)p(s|y)ds$, а $f^*(y)$ совпадает с предельным значением М-оценки для каждого y. Таким образом, множество классических М-оценок может быть использовано для оценки характеристик апостериорного распределения.

В случае когда, как в этой статье, используется независимое нормальное распределение в качестве приближения к апостериорному распределению, средние и стандартные отклонения сходятся к своим истинным значениям. Таким образом, если функция f достаточно гибкая, а количество симуляций стремится к бесконечности, среднее и стандартные отклонения стремятся к своим истинным значениям.

Приложение Б. Модель стохастической волатильности

Б1. Модель

Априорное распределение:

$$\alpha \sim N\left(0, \sqrt{10}\right), \quad \kappa \sim N\left(0, \sqrt{10}\right), \quad \psi \sim N\left(0, \sqrt{10}\right),$$

$$\sigma = \log(1 + e^{\alpha}), \rho = \frac{1}{1+e^{-\psi}}.$$

Уравнение перехода:

$$SV_t \sim N\left(\frac{\kappa}{2}(1-\rho) + \rho SV_{t-1}, \frac{\sigma}{2}\right), t = 2, \dots, T,$$

$$SV_1 \sim N\left(\frac{\kappa}{2}, \frac{\sigma}{2\sqrt{1-\rho^2}}\right).$$

Уравнение наблюдений:

$$y_t \sim N(0, e^{SV_t}).$$

Б2. Архитектура и алгоритм обучения

Алгоритм Б1. Предобучение модели стохастической волатильности ($B = 100$, $N_{sim} = 200\,000$, $T_{lb} = 800$, $T_{ub} = 1200$, $c = 10^{-30}$)

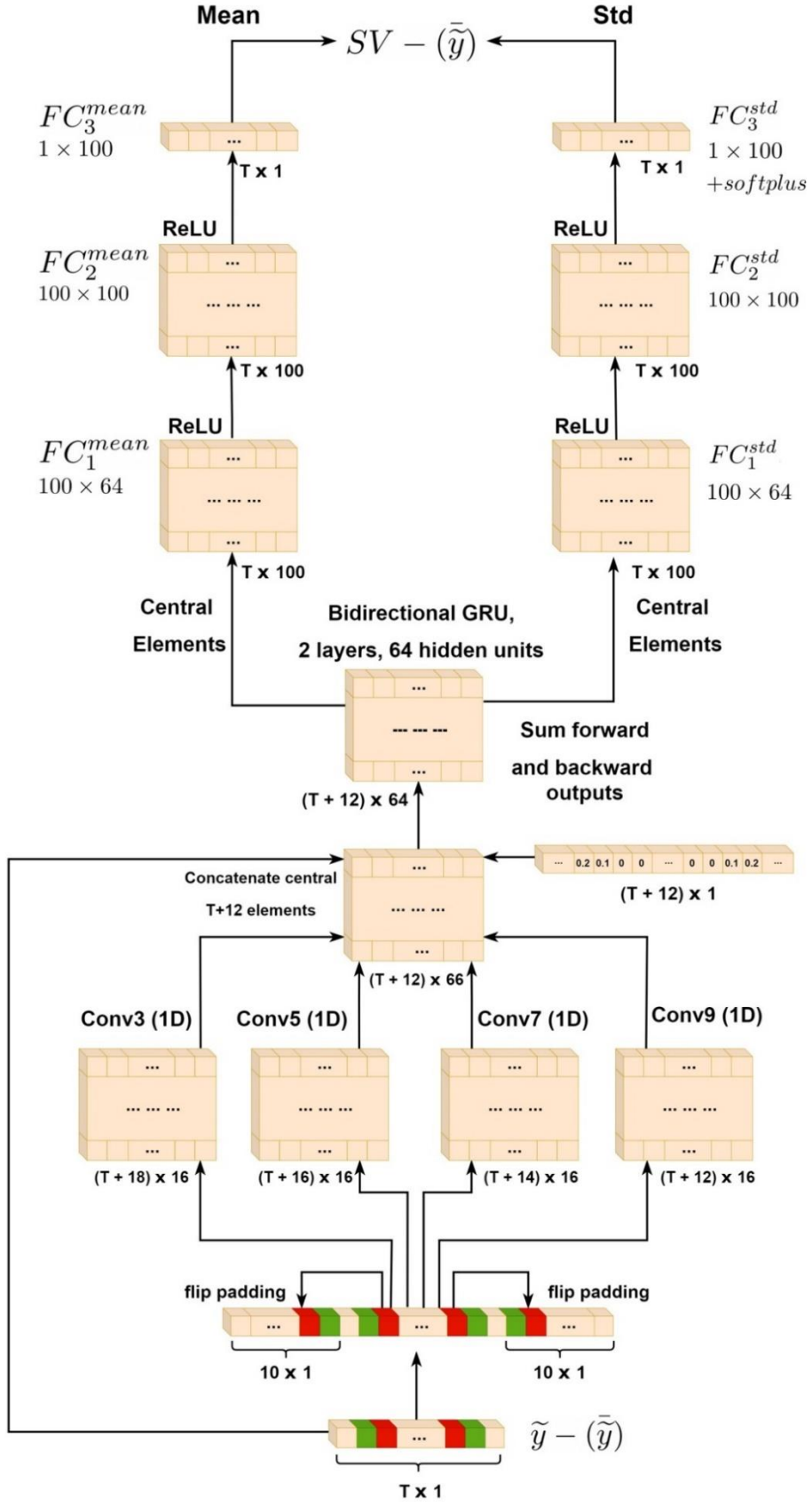
Для $n = 1, \dots, N_{sim}$:

1. Семплировать T_n из равномерного дискретного распределения $T \sim U(T_{lb}, T_{ub})$.
2. Симулировать n-й батч:

   Для $b = 1, \dots, B$:

   2.а. Семплировать $\sigma^b, \kappa^b, \rho^b$ из априорного распределения.

Рисунок Б1. Архитектура нейронной сети для расчета среднего и стандартного отклонения для модели стохастической волатильности



2.b. Семплировать $SV^b = \{SV_1^b, \dots, SV_{T_n}^b\}$ при условии $\sigma^b, \kappa^b, \rho^b$.

2.c. Семплировать $\tilde{y}^b = \{log(c + |y_1^b|), \dots, log(c + |y_{T_n}^b|)\}$ при условии SV^b.

$SV_n^{batch} = \{\{SV^1, \tilde{y}^1\}, \dots, \{SV^B, \tilde{y}^B\}\}$.

3. Рассчитать функцию потерь для архитектуры, изображенной на рисунке Б1:

$$L_n = -\frac{1}{BT_n}\sum_{b=1}^{B}\sum_{t=1}^{T_n} log\, p\left(SV_t^b | m_t(\tilde{y}^b, \varphi), \sigma_t(\tilde{y}^b, \varphi)\right)$$

4. Сделать шаг оптимизации для обновления φ, используя ADAM-алгоритм.

ADAM (Kingma and Ba (2014)) применяется со стандартными настройками, за исключением скорости обучения, ε_n:

$$\varepsilon_n = \begin{cases} 10^{-3}, \text{если } n < 3 \times 10^4 \\ 10^{-4}, \text{если } 3 \times 10^4 \le n < 10^5 \\ 10^{-5}, \text{если } n \ge 10^5. \end{cases}$$

Б3. Альтернативные алгоритмы для модели стохастической волатильности

В качестве алгоритмов для сравнения используются адаптивный MCMC и VB-алгоритмы. MCMC-алгоритм основан на идее приближенного представления распределения логарифма квадрата гауссовской случайной величины в виде смеси семи нормальных распределений (см. Kim, Chib and Shepard (1998)). Для этого уравнение наблюдений переписывается в виде:

$$log\, y_t^2 = 2SV_t + \sum_{k=1}^{7} z_t e_t^k,$$

$$e_t^k \sim N(\mu_k, \sigma_k),$$

$$p(z_t = k) = \omega_k,$$

где $\mu_k, \sigma_k, \omega_k$ – константы, определенные в Kim, Chib and Shepard (1998). Алгоритм Б2 описывает полную процедуру.

Алгоритм Б2. Адаптивный MCMC-алгоритм для модели стохастической волатильности ($N_{sim} = 2000, c = 1.5, \Sigma_0 = 0.1I$)

Для $n = 1, \dots, N_{sim}$:

1. Для $t = 1, \dots, T$ семплировать дискретные переменные аппроксимации для хи-квадрат распределения:

$$z_t^n \sim p(z_t | SV_t^{n-1}, y_t).$$

2. Семплировать параметры $\theta = \{\alpha, \kappa, \psi\}$ с использованием алгоритма Метрополиса – Гастингса с адаптивной вспомогательной плотностью (см. Roberts and Rosenthal (2009)):

$$q(\theta' | \theta_{n-1}) = 0.95N\left(\theta^{n-1}, \frac{c^2}{3}\Sigma_{n-1}\right) + 0.05N\left(\theta^{n-1}, \frac{0.01^2}{3}I\right),$$

и вероятностью принятия нового значения:

$$ar_n = \min\left(\frac{p(y_1,\dots,y_T | z_1^n,\dots,z_T^n,\theta)p(\theta)}{p(y_1,\dots,y_T | z_1^n,\dots,z_T^n,\theta^{n-1})p(\theta^{n-1})}, 1\right).$$

3. Семплировать стохастические волатильности:

$$SV_1^n, \dots, SV_T^n \sim p(SV_1, \dots, SV_T | z_1^n, \dots, z_T^n, y_1, \dots, y_T, \theta^n).$$

Заметим, что после введения переменных $z_1, \dots, z_T$ шаги 2 и 3 алгоритма Б2 могут быть сделаны с помощью стандартных процедур фильтра Калмана и семплирования (см. Durbin and Koopman (2002)).

Для VB-оценки используется алгоритм с нормальной аппроксимацией, где обратная матрица ковариации является разреженной (см. Tan and Nott (2018)). При обучении мы использовали 20 000 итераций алгоритма ADAM со скоростью обучения 0.001 и размером батча 100.

Приложение В. DSGE-модель со стохастической волатильностью

В1. Модель

Мы используем DSGE-модель, аналогичную Diebold, Schorfheide and Shin (2017), но с несколькими модификациями. Во-первых, чтобы не касаться вопросов работы с пропущенными переменными, из наблюдаемых переменных убираются инфляционные ожидания, а также из модели исключается шок в динамике таргетируемой инфляции. Во-вторых, чтобы сделать симуляции чуть более реалистичными, мы немного изменили априорные распределения на процессы, связанные со стохастической волатильностью.

Априорное распределение:

$$\tau \sim N(1.5, 0.36), \quad \nu_l \sim G(2, 0.75), \quad \iota \sim B(0.5, 0.15), \quad \zeta \sim B(0.5, 0.1), \quad \psi_1 \sim N(1.5, 0.25),$$

$$\psi_2 \sim N(0.12, 0.05), \quad -400\, log\, \beta \sim G(1, 0.4), \quad 400\, log\, \pi_* \sim G(2.48, 0.4),$$

$$100\, log\, \gamma \sim N(0.4, 0.1), \quad \rho_R \sim B(0.5, 0.2), \quad \rho_g \sim B(0.5, 0.2), \quad \varphi_z \sim U(-1, 1),$$

$$(100\sigma_R)^2 \sim IG(0.1, 2), \qquad \left(10\sigma_g\right)^2 \sim IG(0.1, 2), \qquad (10\sigma_z)^2 \sim IG(0.1, 2),$$

$$\left(0.2\sigma_g^{SV}\right)^2 \sim IG(0.05, 2), \qquad (0.2\sigma_z^{SV})^2 \sim IG(0.05, 2), \qquad (0.2\sigma_R^{SV})^2 \sim IG(0.05, 2),$$

$$\rho_g^{SV} \sim N(0.9, 0.07),\ \rho_z^{SV} \sim N(0.9, 0.07),\ \rho_R^{SV} \sim N(0.9, 0.07),$$

где $N(a,b), B(a,b), G(a,b) -$ нормальное, бета- и гамма-распределения со средним a и стандартным отклонением b, $U(a,b) -$ равномерное распределение с верхней и нижней границами a и b, $IG(a,b) -$ обратное гамма-распределение с плотностью вероятности $p(x) \sim x^{-b-1} e^{-\frac{ba^2}{2x}}$.

Уравнения перехода:

Уравнения перехода задаются в виде:

$$s_t \sim N\left(A(\theta)s_{t-1}, B(\theta)diag(e^{SV_t})B^T(\theta)\right), t = 2, \dots, T,$$

$$SV_t = \{SV_t^g, SV_t^z, SV_t^R\},$$

$$s_1 \sim N\left(0, P(\theta)\right),$$

$$SV_t^i \sim N\left(\rho_i^{SV} SV_{t-1}^i, \sigma_i^{SV}\right), t = 2, \dots, T, i \in \{g, z, R\},$$

$$SV_1^i \sim \left(0, \frac{\sigma_i^{SV}}{\sqrt{1-\left(\rho_i^{SV}\right)^2}}\right), i \in \{g, z, R\},$$

где $\theta = \left\{\tau, \nu_l, \iota, \zeta, \psi_1, \psi_2, \beta, \pi_*, \gamma, \rho_R, \rho_g, \varphi_z, \sigma_R, \sigma_g, \sigma_z\right\}$, $s_t = \{y_t, c_t, g_t, \pi_t, R_t, z_t, dy_t\}$, $P(\theta) -$ решение уравнения:

$$P(\theta) = A(\theta)P(\theta)A^T(\theta) + B(\theta)B^T(\theta),$$

а $A(\theta)$ и $B(\theta) -$ стабильное решение[20] следующей линейной системы стохастических дискретных уравнений:

$$c_t = E_t(c_{t+1} + z_{t+1}) - \frac{1}{\tau}(R_t - E_t\pi_{t+1}),$$

$$\pi_t = \frac{\iota}{(1+\beta\iota)}\pi_{t-1} + \frac{\beta}{(1+\beta\iota)}E_t\pi_{t+1} + \frac{(1-\zeta\beta)(1-\zeta)}{(1+\beta\iota)\zeta}(c_t + \nu_l y_t),$$

[20] Мы исключаем параметры, при которых существует множество стабильных решений либо же стабильного решения нет.

$$y_t = c_t + g_t,$$

$$R_t = \rho_R R_{t-1} + (1-\rho_R)\big(\psi_1 \pi_t + \psi_2 (y_t - y_{t-1} + z_t)\big) + \sigma_R e_t^R,$$

$$z_t = -\varphi_z z_{t-1} + \sigma_z e_t^z,$$

$$g_t = \rho_g g_{t-1} + \sigma_g e_t^g,$$

$$dy_t = y_t - y_{t-1} + z_t,$$

где $y_t, c_t, g_t, \pi_t, R_t, z_t, dy_t -$ переменные, которые соответствуют отклонениям от стационарного состояния выпуска, потребления, экзогенного процесса, отвечающего за долю госпотребления, инфляции, процентной ставки, экзогенного технологического процесса и прироста ВВП, $e_t^R, e_t^z, e_t^g -$ шок денежно-кредитной политики, технологический шок и шок госпотребления.

Уравнения наблюдений:

Уравнения наблюдений задаются в виде:

$$obs_t = \begin{bmatrix} dy_t^{obs} \\ \pi_t^{obs} \\ R_t^{obs} \end{bmatrix} = \begin{bmatrix} 100\, log\, \gamma \\ 100\, log\, \pi_* \\ 100(log\, \gamma + log\, \pi_* - log\, \beta) \end{bmatrix} + \begin{bmatrix} 100\, dy_t \\ 100\, \pi_t \\ 100\, R_t \end{bmatrix},$$

где $dy_t^{obs} -$ квартальный рост реального ВВП, $\pi_t^{obs} -$ квартальный темп роста цен, $R_t^{obs} -$ ставка процента в квартальных терминах.

B2. Архитектура и алгоритм обучения

Алгоритм B1. Предобучение DSGE-модели со стохастической волатильностью ($N_{presim} = 1\,000\,000$, $B = 100$, $N_{sim} = 500\,000$, $T_{lb} = 180$, $T_{ub} = 200$, $w = 1$, $c = 10^{-30}$)

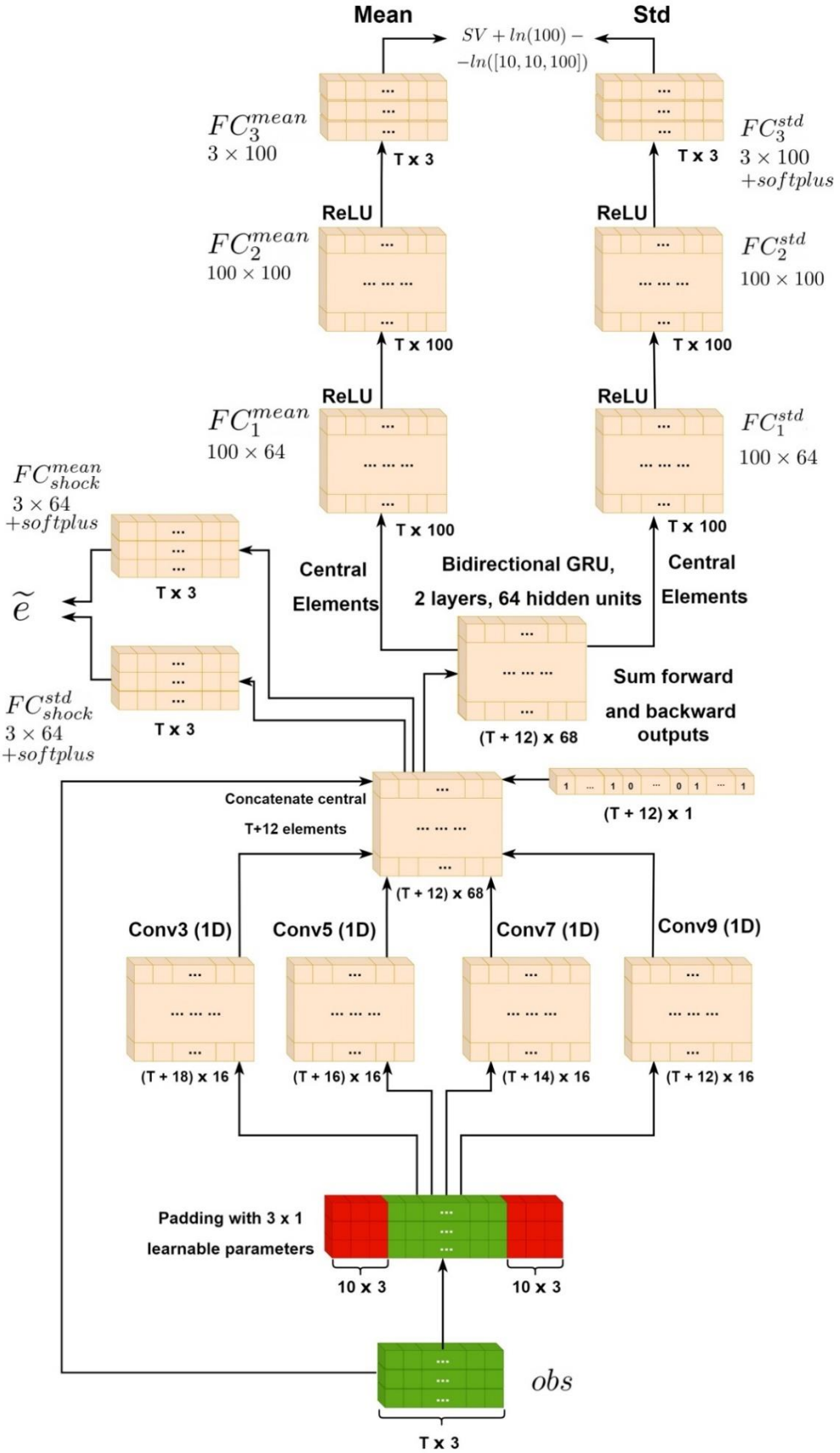
Установить $n_{presim} = 0, \theta = \{\}, A = \{\}, B = \{\}$.

Пока $n_{presim} < N_{presim}$:

1. Семплировать $\theta_{n_{presim}}$ из априорного распределения.
2. Найти решение DSGE-модели[21].

[21] Используется алгоритм из Anderson and Moore (1985).

Рисунок B1. Архитектура нейронной сети для расчета среднего и стандартного отклонения для SV-DSGE-модели



3. Если решение стабильно[22], добавить $\theta_{n_{presim}}, A_{n_{presim}}, B_{n_{presim}}$ в θ, A, B и увеличить n_{presim} на 1.

Для $n = 1, \dots, N_{sim}$:

1. Семплировать T_n из непрерывного дискретного распределения $T \sim U(T_{lb}, T_{ub})$.
2. Симулировать n-й батч:

   Для $b = 1, \dots, B$:

   2.a. Семплировать $\{\theta^b, A^b, B^b\}$ равномерно из $\{\theta, A, B\}$ и $\sigma_g^{SV,b}, \sigma_z^{SV,b}, \sigma_R^{SV,b}$, $\rho_g^{SV,b}, \rho_z^{SV,b}, \rho_R^{SV,b}$ из априорного распределения.

   2.b. Семплировать $SV^b = \{SV_1^b, \dots, SV_{T_n}^b\}$, $s^b = \{s_1^b, \dots, s_{T_n}^b\}$ и $\tilde{e}^b = \left\{\left\{log\left(c + \left|e_1^{R,b}\right|\right), log\left(c + \left|e_1^{z,b}\right|\right), log\left(c + \left|e_1^{g,b}\right|\right)\right\}, \dots, \left\{\left\{log\left(c + \left|e_{T_n}^{R,b}\right|\right), log\left(c + \left|e_{T_n}^{z,b}\right|\right), log\left(c + \left|e_{T_n}^{g,b}\right|\right)\right\}\right\}$ при условии данных из 2a.

   2.c. Семплировать $obs^b = \{obs_1^b, \dots, obs_{T_n}^b\}$ при условии SV^b, s^b и θ^b.

   $SV_n^{batch} = \{\{SV^1, obs^1\}, \dots, \{SV^B, obs^B\}\}$.

3. Рассчитать функцию потерь для архитектуры, изображенной на рисунке B1:

$$L_n = -\frac{1}{3BT_n}\sum_{b=1}^{B}\sum_{t=1}^{T_n}\sum_{i\in\{g,z,R\}}\left(log\, p\left(SV_t^{i,b} | m_{t,i}^{SV}(obs^b, \varphi), \sigma_{t,i}^{SV}(obs^b, \varphi)\right) + w\, log\, p\left(\tilde{e}_t^{i,b} | m_{t,i}^{e}(obs^b, \varphi), \sigma_{t,i}^{e}(obs^b, \varphi)\right)\right).$$

4. Сделать шаг оптимизации для обновления φ, используя ADAM-алгоритм.

Скорость обучения ε_n для алгоритма ADAM задается следующим расписанием:

$$\varepsilon_n = \begin{cases} 10^{-3}, \text{если } n < 3 \times 10^4 \\ 10^{-4}, \text{если } 3 \times 10^4 \le n < 10^5 \\ 10^{-5}, \text{если } 10^5 \le n < 2 \times 10^5 \\ 10^{-6}, \text{если } 2 \times 10^5 \le n < 3.5 \times 10^5 \\ 3 \times 10^{-7}, \text{если } n \ge 3.5 \times 10^5. \end{cases}$$

[22] В нашем случае практически нет ситуаций, когда возникают множественные стабильные или только взрывные решения. См. Lueckmann et al. (2017) как один из примеров того, что делать в ситуациях, когда определенные области пространства параметров являются запрещенными.

B3. Альтернативный алгоритм для DSGE-модели со стохастической волатильностью

Алгоритм B1 сравнивается с адаптивным MCMC-алгоритмом, похожим на тот, что был использован в Justiniano and Primiceri (2008) и Diebold, Schorfheide and Shin (2017). Ключевым отличием является лишь использование алгоритма Метрополиса – Гастингса на шаге семплирования параметров стохастических волатильностей (с проинтегрированными состояниями) вместо семплирования по Гиббсу.

Алгоритм B2. Адаптивный MCMC-алгоритм для DSGE-модели со стохастической волатильностью ($N_{sim} = 100\,000, c = 1.5, \Sigma_0^{\theta} = 0.1I, \Sigma_0^{\theta^{SV}} = 0.1I$)

Для $n = 1, \dots, N_{sim}$:

1. Семплировать параметры θ с использованием алгоритма Метрополиса-Гастингса с адаптивной вспомогательной плотностью:
$$q(\theta'|\theta_{n-1}) = 0.95N\left(\theta^{n-1}, \frac{c^2}{15}\Sigma_{n-1}^{\theta}\right) + 0.05N\left(\theta^{n-1}, \frac{0.01^2}{15}I\right),$$
и вероятностью принятия нового значения:
$$ar_n = \min\left(\frac{p(obs_1, \dots, obs_T|SV_1^{n-1}, \dots, SV_T^{n-1}, \theta')p(\theta')}{p(obs_1, \dots, obs_T|SV_1^{n-1}, \dots, SV_T^{n-1}, \theta^{n-1})p(\theta^{n-1})}, 1\right).$$
2. Семплировать ошибки $e_1, \dots, e_T$:
$$e_1^n, \dots, e_T^n \sim p(e_1, \dots, e_T|obs_1, \dots, obs_T, SV_1^{n-1}, \dots, SV_T^{n-1}, \theta^n).$$
3. Для $t = 1, \dots, T$ семплировать дискретные переменные аппроксимации для хи-квадрат распределения:
$$z_t^{i,n} \sim p\left(z_t^i|SV_1^{n-1}, \dots, SV_T^{n-1}, e_1^n, \dots, e_T^n\right), \qquad i \in \{g, z, R\}.$$
4. Семплировать параметры $\theta^{SV} = \left\{\sigma_g^{SV}, \sigma_z^{SV}, \sigma_R^{SV}, \rho_g^{SV}, \rho_z^{SV}, \rho_R^{SV}\right\}$ с использованием алгоритма Метрополиса – Гастингса с адаптивной вспомогательной плотностью:
$$q(\theta^{SV\prime}|\theta_{n-1}^{SV}) = 0.95N\left(\theta^{SV,n-1}, \frac{c^2}{6}\Sigma_{n-1}^{\theta^{SV}}\right) + 0.05N\left(\theta^{SV,n-1}, \frac{0.01^2}{6}I\right),$$
и вероятностью принятия нового значения:
$$ar_n = \min\left(\frac{p(e_1^n, \dots, e_T^n|z_1^n, \dots, z_T^n, \theta^{SV\prime})p(\theta^{SV\prime})}{p(e_1^n, \dots, e_T^n|z_1^n, \dots, z_T^n, \theta^{SV,n-1})p(\theta^{SV,n-1})}, 1\right).$$
5. Семплировать стохастические волатильности:
$$SV_1^n, \dots, SV_T^n \sim p(SV_1, \dots, SV_T|z_1^n, \dots, z_T^n, e_1, \dots, e_T, \theta^{SV,n}).$$

Приложение Г. Модель сезонной корректировки со структурными сдвигами в сезонности

В отличие от предыдущих моделей здесь не задается процесс порождения данных напрямую. Вместо этого мы описываем процедуру генерации данных:

Алгоритм Г1. Генерация данных со сдвигами в сезонности ($T_{lb} = 40$, $T_{ub} = 80$, $T^{period} = 4, B = 100$)

1. Семплировать T_n из непрерывного дискретного распределения $T \sim U(T_{lb}, T_{ub})$.
2. Симулировать компоненты батча:

   Для $b = 1, \dots, B$:

   2.a. Сгенерировать несезонную компоненту NS^b:

$$e_t^b \sim Student(0, 1, 3 + |\eta_t|), \quad t = -199, \dots T_n,$$

$$\eta_t \sim N(0,3), \quad t = -199, \dots T_n,$$

$$e_t^{shift,b} \sim N(0,20) Bernouli(0.01), \quad t = -196, \dots T_n,$$

$$\rho_{AR,1}^b, \rho_{AR,2}^b, \rho_{AR,3}^b, \rho_{MA,1}^b, \rho_{MA,2}^b, \rho_{MA,3}^b \sim Bernouli(0.5) U[-0.5, 0.98],$$

$$\varepsilon_t = e_t^b + \left(\rho_{MA,1}^b + \rho_{MA,2}^b + \rho_{MA,3}^b\right) e_{t-1}^b + \left(\rho_{MA,1}^b \rho_{MA,2}^b + \rho_{MA,2}^b \rho_{MA,3}^b + \rho_{MA,1}^b \rho_{MA,3}^b\right) e_{t-2}^b + \rho_{MA,1}^b \rho_{MA,2}^b \rho_{MA,3}^b e_{t-3}^b + e_t^{shift,b}, \quad t = -196, \dots T_n,$$

$$x_t^b = \left(\rho_{AR,1}^b + \rho_{AR,2}^b + \rho_{AR,3}^b\right) x_{t-1}^b - \left(\rho_{AR,1}^b \rho_{AR,2}^b + \rho_{AR,2}^b \rho_{AR,3}^b + \rho_{AR,1}^b \rho_{AR,3}^b\right) x_{t-2}^b + \rho_{AR,1}^b \rho_{AR,2}^b \rho_{AR,3}^b x_{t-3}^b + \varepsilon_t, \quad t = -196, \dots T_n,$$

$$x_{-199}^b, x_{-198}^b, x_{-197}^b = 0,$$

$$c^b \sim N(0, 0.005), scale^b \sim N\left(0, \frac{0.007}{std(x^b)}\right),$$

$$NS^{*b} = \left\{c^b + scale^b x_{-199}, \dots, c^b + scale^b x_{T_n}\right\},$$

$$I^{integrated,b} \sim Bernouli(0.5),$$

$$NS^b = I^{integrated,b} cumsum(NS^{*b}) + \left(1 - I^{integrated,b}\right) NS^{*b}.$$

   2.b. Сгенерировать сезонную компоненту S^b:

$$\sigma^b \sim N\left(0, \frac{0.2}{\sqrt{40}}\right),$$

$$I_t^{shift,b} \sim Bernouli(0.01), \quad t = -199, \dots T_n,$$

$$z_t^b = I_{t-3}^{shift,b} I_{t-2}^{shift,b} I_{t-1}^{shift,b} I_t^{shift,b}, \quad t = -196, \dots T_n,$$

$$e_t^{S,b} \sim N(0,1), \quad t = -196, \dots T_n,$$

$$s_{-199}^b, s_{-198}^b, s_{-197}^b \sim N(0,1),$$

$$s_t^b = -\left(1 - z_t^b\right)\left(s_{t-1}^b + s_{t-2}^b + s_{t-3}^b + \sigma^b e_t^{S,b}\right) + z_t^b e_t^{S,b},$$

$$scale^{S,b} \sim N\left(0, 3std(NS^{*b})\right),$$

$$S^b = \left\{scale^{S,b} s_{-199}^b, \dots, scale^{S,b} s_{T_n}^b\right\}.$$

3. Создать батч размера $2B$:

   Для $b = 1, \dots, B$:

$$y^{*b} = S^b + NS^b,$$

$$y^b = \left\{\frac{y_1^b - mean(y^{*b})}{std(y^{*b})}, \dots, \frac{y_T^b - mean(y^{*b})}{std(y^{*b})}\right\},$$

$$y^{B+b} = \left\{\frac{y_T^b - mean(y^{*b})}{std(y^{*b})}, \dots, \frac{y_1^b - mean(y^{*b})}{std(y^{*b})}\right\},$$

$$sa^b = \left\{\frac{NS_1^b - mean(y^{*b})}{std(y^{*b})}, \dots, \frac{NS_T^b - mean(y^{*b})}{std(y^{*b})}\right\},$$

$$sa^{B+b} = \left\{\frac{NS_T^b - mean(y^{*b})}{std(y^{*b})}, \dots, \frac{NS_1^b - mean(y^{*b})}{std(y^{*b})}\right\},$$

   $y = \{y^1, \dots, y^{2B}\}, \quad sa = \{sa^1, \dots, sa^{2B}\}.$

Алгоритм оценки нейронной сети, аппроксимирующей среднее и стандартное отклонение апостериорного распределения, подобен тем, что были описаны для других моделей.

Алгоритм Г2. Сезонная корректировка со структурными сдвигами в сезонности ($N_{sim} = 100\,000$)

Для $n = 1, \dots, N_{sim}$:

1. Симулировать n-й батч с использованием алгоритма Г1:

$$sa_n^{batch} = \{\{sa^1, y^1\}, \dots, \{sa^{2B}, y^{2B}\}\},$$

2. Рассчитать функцию потерь для архитектуры, изображенной на рисунке Г1:

$$L_n = -\frac{1}{2BT_n}\sum_{b=1}^{2B}\sum_{t=1}^{T_n} log\, p\left(sa_t^b | m_t(y^b, \varphi), \sigma_t(y^b, \varphi)\right).$$

3. Сделать шаг оптимизации для обновления φ, используя ADAM-алгоритм.

Расписание для алгоритма ADAM задается, как:

$$\varepsilon_n = \begin{cases} 10^{-3}, \text{если } n < 1.5 \times 10^4 \\ 10^{-4}, \text{если } 1.5 \times 10^4 \leq n < 5 \times 10^4 \\ 10^{-5}, \text{если } n \geq 5 \times 10^4. \end{cases}$$

Рисунок Г1. Архитектура нейронной сети для расчета среднего и стандартного отклонений для SA-модели