ChatGPT

Сравнение симуляции RQSimForms с реляционно-квантовой гипотезой

Введение. Реляционно-квантовая (RQ) гипотеза в современной теоретической физике утверждает, что ни пространство, ни время не являются изначально заданными фундаментальными сущностями – они *эмергируют* из совокупности отношений (корреляций) между квантовыми системами [1]. В таком подходе физические события трактуются как корреляции между подсистемами без привязки к внешнему пространственно-временному фону [2]. Программа RQSimForms реализует численную симуляцию этой идеи, представляя Вселенную в виде динамического графа (сети), узлы которого – это «квантовые объекты», а ребра – меры их корреляции [3]. В данной статье проводится обзор того, как ключевые принципы реляционно-квантовой гипотезы воплощены (либо аппроксимированы) в коде RQSimForms, построенном на событийно-ориентированной архитектуре. Мы рассмотрим динамику графа, механизм формирования массы, реализацию калибровочных полей и гравитации на графе, а также модель **реляционного времени** (формализм Пейджа–Вуттерса). Для каждой темы отмечены соответствия между теорией и симуляцией, а также объясняются отличия, связанные с особенностями событийной модели, численными методами и масштабами вычислений.

Динамика графа: пространство из корреляций без абсолютного фона

Реляционный принцип: В RQ-гипотезе принимается, что **не существует изначального пространства-времени** – метрика и геометрия должны возникнуть на основе корреляционных связей между квантовыми объектами [1]. Узлы графа интерпретируются не как точки в пространстве, а как локальные квантовые состояния, а ребра – как квантовые корреляции или запутанность между ними [3]. Внешних координат нет: расстояния и структура «пространства» выводятся эндогенно, например, через свойства графа (спектральная размерность, метрика на графе) [4]. **Причинно-следственные связи** также должны быть не привнесены извне, а возникать из структуры графа – через его связность (отсутствие связи между двумя узлами означает отсутствие прямого влияния, аналогично причинной разобщенности) [2].

Реализация в коде: В RQSimForms вся Вселенная моделируется как граф `RQGraph` с N узлами. В коде явно отсутствует какая-либо глобальная система координат – массив координат узлов присутствует лишь для визуализации и **не участвует в физических расчётах** [5]. Вместо этого расстояния вычисляются по графу (например, кратчайший путь) или через веса ребер. Каждое ребро имеет вес `Weights[i,j]`, характеризующий «силу» связи или степень корреляции между узлами *i* и *j*. Метрика может быть выведена, например, как `distance ~ -\ln(\text{weight})` или иными способами через спектральную геометрию графа [4]. В процессе симуляции **граф эволюционирует**: веса ребер динамически обновляются по законам, имитирующим физику (см. раздел о гравитации ниже), а сами ребра могут добавляться или удаляться. Например, предусмотрены механизмы предотвращения фрагментации графа: если сеть грозит распасться на несвязные компоненты, симуляция может **добавлять новые ребра** между случайными узлами, чтобы восстановить единство пространства [6] [7]. Такой шаг – вынужденная мера в модели, гарантирующая, что «вселенная» остаётся связной (что соответствует предположению о едином континууме в реальности), даже если при некоторых настройках параметров граф начал

разбиваться на изолированные островки. В идеале сама физика должна предотвращать полную разрывность пространства, но в численной модели приходилось компенсировать это артефактами (например, из-за слишком сильной гравитационной связи или недостаточного времени релаксации, см. раздел о гравитации).

Отдельного упоминания заслуживает **событийно-ориентированная модель времени**, заложенная в архитектуру симуляции. Теория утверждает отсутствие глобального «сейчас» и абсолютного единого времени для всех частей системы [8]. Код RQSimForms построен так, чтобы не обновлять синхронно все узлы в фиксированные такты глобального времени (как это делается в наивной итерационной симуляции), поскольку такая схема противоречила бы реляционному принципу. Вместо этого реализован **дискретный событийный движок**. Каждый узел обладает собственным *собственным временем* τ_i, течение которого зависит от локального состояния узла (например, энергии, окружения), и эволюционирует независимо. В модуле `EventDrivenEngine` запускается цикл, выбирающий следующий узел, который «созреет» для обновления, на основе наименьшего значения его локального времени [9] [10]. Узел обновляется (что соответствует произошедшему **событию** – взаимодействию), после чего для него планируется следующая порция эволюции через определенный интервал $\Delta \tau_i$, рассчитанный по внутренней логике (см. раздел о времени). Затем этот узел снова помещается в очередь событий с обновленным временем следующего события [11] [12]. Благодаря этому **отсутствует глобальный шаг dt** – модели внешнего непрерывного времени нет вовсе, его роль статистически берёт на себя тот самый событийный механизм. Глобальные циклы по всем узлам всё же могут использоваться для удобства (например, когда опция событийной симуляции отключена), но при активированной RQ-модели времени движок использует принципиально иную схему, согласующуюся с реляционной картиной [13] [11]. Более того, код поддерживает **параллельное** выполнение независимых событий: перед запуском симуляции граф окрашивается в несколько цветов так, что узлы одного цвета не имеют прямых связей друг с другом [14] [15]. Это означает, что они могут обновляться одновременно, не нарушая причинности (ведь отсутствие ребра между ними подразумевает отсутствие непосредственного взаимодействия) [8]. Поток выполнения разбивается на группы узлов (по цветам), и для каждой группы физические обновления узлов выполняются параллельно потоками, после чего происходит синхронизация и переход к следующему «цвету» [14]. Этот приём повышает производительность на многопроцессорных системах и одновременно отражает идею *отсутствия единовременного глобального обновления*: пока одни части Вселенной взаимодействуют между собой, другие, не связанные с ними, могут эволюционировать параллельно без нарушения локальной причинности [8].

Приближения и ограничения: В целом, динамический граф в RQSimForms довольно близко следует философии реляционного взгляда – **никаких внешних пространственных координат или абсолютного времени**, только изменяющаяся сеть корреляций. Тем не менее, некоторые аспекты осуществлены приближённо. Например, на практике симуляция стартует с некоего начального графа, заданного явно (будь то случайный граф или специальная конфигурация, например «Buckyball» или «Wormhole» в папке `Experiments/Definitions`) – то есть начальные условия заложены извне. Разрешённый размер графа конечно (сотни-тысячи узлов), поэтому эмергентная метрика лишь приближенно подчиняется тем свойствам, которые ожидаются в континуальном пределе. Например, **спектральная размерность** смоделированного «пространства» (эффективное фрактальное измерение графа) не всегда оказывается близка к 3+1=4, как того требует гипотеза на больших масштабах; возможны отклонения. В одном из диагностических прогонов получена даже отрицательная спектральная размерность $d_S \approx -0.86$ [16] [17], что явно противоречит физическому смыслу. Разбор показал, что это произошло из-за фрагментации графа на компоненты: случайный блуждатель на таком графе практически не возвращается в исходную точку, нарушая нормальное поведение $P(t)\sim t^{-d_S/2}$ и

приводя к некорректному наклону кривой (отсюда формально отрицательное d_S) [17]. Это явное расхождение с теорией вызвано техническими причинами – недостаточное время симуляции и неподходящие параметры (слишком быстрое охлаждение, чрезмерно сильная гравитация, см. ниже), – и было устранено подбором режимов (увеличением длительности, плавным вводом гравитации и т.п.). Таким образом, хотя базовая картина «граф вместо пространства» соблюдается строго, практические нюансы (размер, границы, дискретизация событий) могут приводить к временным конфликтам между теорией и симуляцией. Разработчики предусмотрели некоторые «предохранители», как упомянутая подкачка ребер для связности или сглаживание расчёта размерности [18] [19], чтобы по возможности уменьшить эти артефакты.

Формирование массы: кластеры как частицы

Принцип гипотезы: В реляционно-квантовой картине **масса не является неким изначальным параметром**, а должна выводиться из энергий корреляций внутри системы. Интуитивно, стабильные связные структуры графа – «кластеры» – выступают аналогами частиц, и их *внутренняя энергия связи* проявляется как инерционная масса этих образований [20]. Нет внешней «массы покоя» или гравитационного поля, которое нужно закладывать извне; вся масса – *внутренняя*, обусловленная степенью коррелированности узлов в связанных группах. Таким образом, частицы – это устойчивые подграфы, топологически защищённые, а их масса пропорциональна энергии связи, удерживающей этот подграф цельным [20].

Реализация в коде: В RQSimForms описанный подход реализован через вычисление так называемой **корреляционной массы** (`Correlation Mass`) для узлов графа. Процедура `ComputePerNodeCorrelationMass()` находит кластеры узлов, сильно связанных между собой (т.е. с весами ребер выше некоторого порога) [21]. Каждый такой кластер рассматривается как кандидат на «частицу». Для него суммируется избыточная энергия связей: грубо говоря, для каждой пары узлов *v* и *u* в кластере берётся вес ребра $w_{v,u}$ сверх порогового значения и складывается ко вкладу энергии кластера [22]. Полученная суммарная энергия связи распределяется между узлами кластера поровну, и этот вклад добавляется к корреляционной массе каждого узла [22]. Таким образом, если узел входит в крупный сильно связанный компонент, ему приписывается значительная корреляционная масса, отражающая «связанную энергию» этого образования. Узлы, не принадлежащие никаким плотным кластерам, имеют близкую к нулю корреляционную массу (весьма «легки»). Параметры порога и минимального размера кластера настраиваются (по умолчанию кластер из <3 узлов не считается «частицей») [21]. Такое вычисление повторяется динамически по ходу симуляции и используется, например, в гравитационных уравнениях для учета распределения массы (см. раздел о гравитации ниже, где корреляционная масса выступает как источник «стресс-энергии» в дискретном аналоге уравнений Эйнштейна) [23].

В ходе разработки симуляции выявилось, что при некоторых параметрах система не образует тяжелых кластеров – граф остается диффузным, однородным, то есть материя не конденсируется. Например, при слишком слабом притяжении узлы не успевают сгруппироваться, а при слишком сильном – граф может быстро фрагментироваться, прежде чем кластеры стабилизируются [24]. В одном из обновлений был решён этот вопрос: **усилен гравитационный коэффициент и введён “отжиг” (annealing)** – на ранних этапах гравитация делается искусственно сильнее, стимулируя образование кластеров, а затем плавно ослабевает до физически разумного уровня [25]. Например, базовое гравитационное сцепление увеличили ~в 25 раз и постепенно снижали в течение первых 1000 шагов симуляции [25]. После такой калибровки модель стала формировать «тяжелые кластеры» – аналог материи – вместо того, чтобы оставаться разреженной или полностью распадаться [26]. Это показательный момент: **масса действительно является**

эмергентным свойством в симуляции, но для её проявления нужны подходящие условия; их подбор – часть работы, чтобы численная Вселенная повела себя подобно реальной.

Стоит отметить, что помимо чисто топологической массы, в симуляции присутствуют и другие вклады. Код содержит поля для масс фермионов, распределённые по узлам (`_fermionMassField`) и потенциально «массу», связанную с калибровочными полями [27]. Например, если на узел помещён фермион (см. ниже про спиноры), ему может приписываться некий вклад массы. Анализ реализации показал проблему: возникло **три разных определения массы** – топологическая корреляционная масса, масса фермионных полей и вклад от полей (например, энергия калибровочного поля) [27]. Такое дублирование нежелательно, поскольку в идеале масса объекта должна быть единой величиной. В документации отмечен этот конфликт и предложено вычислять **итоговую массу узла как сумму** всех компонентов: топологической (связанной с кластером), фермионной и калибровочной [28] [29]. В дальнейшем кодом предусмотрена функция `TotalMass(i)` для получения единого значения массы узла на основе всех вкладов [30]. Это важный момент расхождения с «чистой» RQ-гипотезой: в идеальном мире вся масса была бы топологической, но в модели пришлось ввести вспомогательные компоненты (скажем, исходная масса уравнения Дирака, параметр в потенциале скалярного поля и т.д.) для корректной имитации известных полей Стандартной модели. Эти компоненты можно считать **приближением** – они вводятся из прагматичных соображений (проще задать ненулевую массу фермиону, чем получить её исключительно из корреляций при малых размерах графа). Тем не менее, сам факт, что стабильная связная структура графа получает добавочный вес (массу) *в коде напрямую реализован*. Таким образом, RQSimForms в значительной степени соответствует гипотезе: **«частицы = кластеры, масса = энергия связи»** – и различия возникают в основном из-за дополнительных полевых степеней свободы, нужных для богатства модели, и из-за конечности системы. В большой предельной модели эти различия могли бы исчезнуть, объединившись в единый механизм.

Калибровочные поля на графе: фазы на ребрах

Принцип гипотезы: Одним из постулатов RQ-гипотезы является то, что привычные нам поля взаимодействий (электромагнитное, слабое, сильное) можно описать **в терминах графа** как дополнительные внутренние параметры на узлах или ребрах. В частности, **калибровочные поля** могут быть реализованы в виде фазовых факторов на ребрах графа [31]. Аналогично решеточным теориям, каждое ребро несёт «связь» – элемент группы симметрии (например, U(1) для электромагнетизма или SU(2)/SU(3) для ядерных взаимодействий). Динамика этих фазовых переменных должна подчиняться уравнениям Янг–Миллса, но без апелляции к внешнему континууму – только исходя из структуры графа и распределения материи на нём.

Реализация в коде: В RQSimForms присутствует явная модель **U(1)-калибровочного поля** на графе. При инициализации графа вызывается `InitEdgeGaugePhases()`, которая присваивает каждому ребру небольшую случайную фазу θ_{ij} (симметрично $\theta_{ji} = -\theta_{ij}$) [32] [33]. Эта фаза – аналог потенциала вдоль ребра – и представляет, к примеру, электромагнитный фазовый фактор. В ходе симуляции фазы обновляются методом `UpdateGaugePhases(dt)`, который реализует дискретный аналог уравнений для электромагнитного поля [34] [35]. Алгоритм следующий: для каждого ребра (i,j) суммируется *криволинейность* поля вокруг элементарных циклов, содержащих это ребро. На графе наименьший цикл – это треугольник (i–j–k). Сумма фаз вокруг такого треугольника $\theta_{ij} + \theta_{jk} + \theta_{k i}$ характеризует поток калибровочного поля через этот контур (аналоги компоненты тензора напряжённости поля $F_{\mu\nu}$) [36]. Берётся синус этой суммы (что соответствует непрерывному пределу $F \sim \sin(\text{curl }A)$ на решётке). Далее учитывается

ток через данное ребро – наличие заряда или токов материи, которые могут влиять на эволюцию фазы [37]. В коде ток вычисляется на основе разности «плотностей» на концах ребра: если узел i более «возбуждён» или имеет больший заряд, чем j, то по ребру течёт ток от i к j [38] [39]. Фактически, функция `ComputeEdgeCurrent(i,j)` использует состояние узлов (`State` – например, возбужден/не возбужден) и локальный потенциал, чтобы определить направление и величину тока, умноженного на вес ребра [40] [41]. Кроме того, добавлен возможный вклад от *скалярного поля* (наподобие Хиггса): функция `ComputeScalarFieldCurrent(i,j)` учитывает произведение значений скалярного поля $\phi_i \phi_j$ и синус фазового угла, что даёт дополнительный ток – так симуляция пытается учесть обратное влияние скалярного поля на калибровочное (эффект Хиггса) [42]. После суммирования всех составляющих **фаза на ребре обновляется**: $\dot{\theta}*{ij} \propto -(J)$ [43]. Знак минус перед током соответствует тому, что возрастающий ток снижает разность фаз (как в уравнениях Максвелла $\partial_t A \sim - j$), а добавочный член с кривизной приводит к эффекту типа уравнения для магнитного потока (стремится аннулировать локальный криволинейный дефект, аналог закона Фарадея на решётке). После вычисления приращений все фазы корректируются и нормируются обратно в диапазон $[-\pi, \pi]$ [44] [45] .} + c \cdot \text{сумма синусов по плакеткам

Таким образом, **электромагнитное поле на графе** моделируется как совокупность фаз на ребрах, эволюционирующих под действием своих «уравнений движения». Эти уравнения сформулированы *операционно* – без обращения к каким-либо координатам или непрерывному пространству, а только в терминах топологии графа (петли, треугольники) и внутренних свойств узлов (плотности возбуждения, поля) [36] [42]. Это соответствует требованию **background independence** для калибровочных полей [46]. Более сложные калибровочные поля также учтены: код содержит структуры для SU(2) и SU(3) полей (например, массив `_gluonField` размерности $N\times N\times 8$ – по 8 цветовых компонент на каждое ребро для поля сильного взаимодействия) и многокомпонентную волновую функцию `_waveMulti` с размерами, соответствующими gauge-степеням свободы [47] [48]. В реализацию включён модуль `EvolveYangMillsRelational(dt)`, который должен обновлять эти поля по реляционно-инвариантным уравнениям (включая коммутаторы полей, токи цвета и т.д.) [49] [50]. Код достаточно сложен и, судя по комментариям, находится в разработке, но важный факт: **везде исключена ссылка на привычные координаты**. Например, токи цвета вычисляются по разности плотностей $\rho_i - \rho_j$ узлов *i* и *j*, умноженной на вес ребра – т.е. поток цвета течёт вдоль ребра по градиенту плотности, *модулированному фазовым сдвигом*, если таковой имеется [51] [52]. Это чисто *реляционный* расчет тока, опирающийся на граф (в противоположность, скажем, токам в решёточном QCD, где вычисления ведутся с учётом фиксированной решётки).

Ещё один значимый элемент – соблюдение **калибровочной инвариантности**. В численной симуляции могут накапливаться ошибки, нарушающие условие $\nabla\cdot E = \rho$ (закон Гаусса) или калибровочные ограничения. В RQSimForms добавлен модуль коррекции `GaussLawProjection`: он периодически проверяет выполненность закона Гаусса, вычисляя дивергенцию электрического поля $div(E)$ на каждом узле и сравнивая с локальным зарядом ρ [53]. Решая соответствующее уравнение (аналог уравнения Пуассона) и найдя скалярную функцию χ_i, код выполняет **калибровочное преобразование** фаз: $\theta'*{ij} = *theta* - (\chi_i - \chi_j)$ [54]. Это эквивалентно тому, как поправляют решётку в Kogut-Susskind подходе, убирая нерегулярные скапливания магнитного потока. Данный шаг гарантирует, что после него $\nabla\cdot E - \rho = 0$ на узлах (в пределах вычислительной погрешности), восстанавливая **локальную калибровочную инвариантность**. Подобная процедура – вынужденное численное приближение: в идеале уравнения движения сами сохраняли бы Constraints, но из-за конечного шага интегрирования требуются поправки. Тем не менее, сама возможность такой корректировки свидетельствует, что модель учитывает **зарядово-нейтральные токи** и старается поддерживать первые интегралы (закон сохранения заряда) даже в дискретном мире.

Приближения и расхождения: Модель калибровочных полей в RQSimForms, безусловно, является упрощённой по сравнению с полной квантовой теорией поля. Во-первых, реализована в явном виде пока только группа U(1) с явным обновлением фаз; SU(2) и SU(3) – частично (в коде есть задел, но их динамика сложнее и, возможно, в текущей версии используется ограниченно). Во-вторых, поля эволюционируют по классическим уравнениям на графе (синусы фаз, токи) – **квантовые флуктуации** поля напрямую не моделируются. Это значит, что пока мы работаем скорее с средними значениями полей (как в классическом пределе), а квантование могло бы потребовать, например, стохастическое включение/выключение плакеттных переменных или распределения вероятностей фаз, чего нет. Тем не менее, для целей исследования **калибровочная теория на динамическом графе** реализована: важнейшие понятия (фазовые переменные на ребрах, плакетты, токи от фермионных полей, закон Гаусса) воспроизведены в соответствиях, близких к теории решёточного gauge-поля. Таким образом, *RQSimForms напрямую воплощает идею, что взаимодействия (силовые поля) – это дополнительные отношения на графе*, а не поля на внешнем пространстве. Расхождения состоят в том, что для поддержания устойчивости модели разработчики вынуждены были вручную вводить некоторые параметры (например, коэффициенты сопряжения g, масштаб вклада плакетт c [55], которые подгоняются) и выполнять коррекции (GaussLawProjection) – это связано с конечной точностью и дискретностью симуляции. В реальности такие параметры должны быть либо фундаментальными константами, либо возникать масштабно, но в модели они настройваемы. Калибровочные поля в RQSimForms можно считать **приближенным аналогом** полей Стандартной модели, пригодным для качественных экспериментов (например, изучить, как присутствие фаз на ребрах влияет на структуру графа, возникновение массового разрыва у фермионов и т.д.), но не претендующим на количественное совпадение с физическими константами.

Гравитация на графе: эйнштейновы уравнения в дискретном виде

Принцип гипотезы: В реляционной картине гравитация не рассматривается как отдельное поле, а отождествляется с **динамикой самой структуры связей** (геометрии) системы. Уравнения Эйнштейна в общем смысле должны возникнуть как следствие экстремума некоторого действия, зависящего от «корреляционной геометрии». То есть граф должен эволюционировать так, чтобы аналог кривизны стремился уравновеситься с аналогом распределения массы/энергии (стресс-энергии), не прибегая к существованию фонового пространства. Гипотеза ожидает, что в пределе большое число узлов, большое число корреляций – мы получим поведение близкое к классической ОТО (4-мерной) [56] [57].

Реализация в коде: RQSimForms содержит специальный модуль `RQGraph.NetworkGravity` – **«сетевая гравитация»**, реализующий эволюцию весов ребер под действием эффективных гравитационных законов [58]. Ключевая часть – функция `EvolveNetworkGeometry(dt)`, выполняющая одномоментное обновление всех весов за шаг *dt* согласно вариационным уравнениям. Чтобы задать эти уравнения, сначала считается **дискретная кривизна** каждого ребра. В текущей реализации по умолчанию используется приближение Формана для риш curvature: для ребра (i,j) кривизна вычисляется, сравнивая количество (и суммарную силу) треугольников, содержащих это ребро, с «пенальти» за валентности узлов [59] [60]. Формула в коде:

$$Ric_{Forman}(i,j) = w_{ij}\Big(\sum_{\triangle ijk} \sqrt{w_{ik}w_{jk}w_{ij}} \;-\; \alpha\,(W_i + W_j - 2w_{ij})\Big),$$

где сумма по всем треугольникам *i–j–k*, W_i и W_j – суммарные веса, сходящиеся в узлы i и j соответственно, а α – некоторая малая постоянная (коэффициент штрафа за степень узла) [61] [62]. В результате $Ric(i,j)$ положителен, если ребро окружено множеством треугольников (кластеризованная область, «сферическая» геометрия), и отрицателен, если ребро топологически принадлежит «дереву» с малым числом циклов (что аналогично гиперболической геометрии) [63]. Далее, высчитывается **аналог действия** S. В комментарии к коду указано:

$$S = S_{\text{geometry}} + S_{\text{matter}},$$

где

$$S_{\text{geometry}} = \sum_{(i,j)} w_{ij}\, R_{ij}$$

(дискретный аналог действия Эйнштейна–Гильберта на графе [64]), а

$$S_{\text{matter}}$$

содержит вклад материи – в данном случае через корреляционную массу узлов. Кроме того, есть и Λ – *космологический член*, пропорциональный числу узлов (эффективный объём) [65], который введён с небольшим положительным значением для стабильности (разрастания графа). Из вариационного принципа выводятся псевдо-«уравнения Эйнштейна» в форме градиентного спуска:

$$\frac{\partial S}{\partial w_{ij}} = 0 \;\Rightarrow\; \Delta w_{ij} \propto -\frac{\partial S}{\partial w_{ij}}.$$

Код реализует именно это: изменение веса Δw за шаг времени берётся **противоположным градиенту** S [66] [23]. Градиент включает три слагаемых для каждого ребра: - $\partial S_{\text{geometry}}/\partial w_{ij} = Ric_{ij}$, то есть саму дискретную кривизну ребра. Положительная кривизна даёт положительную производную (с ростом веса действие возрастает), отрицательная – отрицательную. - $\partial S_{\text{matter}}/\partial w_{ij}$ пропорциональна $-(m_i + m_j)$, где m_i и m_j – корреляционные массы в узлах *i* и *j*. Знак минус отражает то, что наличие массы *уменьшает* действие при больших весах (т.е. энергетически выгодно наращивать связь между массивными узлами). В коде фактически используется $T_{ij} = \frac{1}{2}(m_i + m_j)$ с неким коэффициентом [67]. - $\partial S_{\Lambda}/\partial w_{ij} = \text{const}$, обусловленная космологической постоянной Λ, которая пытается *увеличить* все веса равномерно (как положительное давление/фоновая энергия) [23].

Сворачивая эти члены, изменение веса задаётся как:

$$\Delta w_{ij} = -\Gamma \left(Ric_{ij} + dS_{\text{matter}} + dS_{\Lambda} \right) dt,$$

где Γ – «скорость обучения», аналог гравитационной постоянной (чем больше Γ, тем активнее веса подстраиваются под кривизну) [68] [67]. В базовой реализации выбрана $\Gamma = 2.0$ (достаточно большое значение, после упомянутых калибровок) [69].

Алгоритмически, `EvolveNetworkGeometry` пробегает все ребра и применяет вычисленные Δw параллельно [70] [71], симметрично обновляя w_{ij} и w_{ji}. Есть защита от выхода весов за разумные пределы: `ApplySoftWalls` накладывает плавные ограничения – веса не могут расти сверх заданного порога или падать ниже минимального положительного значения, во избежание численных проблем [72] [73]. Постепенно граф «релаксирует» к состоянию, близкому к экстремуму действия. В таком состоянии достигается нечто аналогичное стационарному решению уравнений Эйнштейна: локально баланс $Ric_{ij} \approx -

(dS_{\text{matter}} + dS_{\Lambda})$, что можно трактовать как **дискретное уравнение $Ric \sim T + \Lambda$** (кривизна равна вкладу материи с противоположным знаком, с учетом знаковой конвенции) [74].

Чтобы симуляция могла не только найти статическое решение, но и отразить *динамические эффекты гравитации* (например, гравитационные волны, колебания метрики), в RQSimForms введена альтернативная схема – **гамильтонова** эволюция геометрии. В модуле `EvolveGeometryHamiltonian` веса ребер наделяются «моментами» π_{ij} – аналогами импульсов связей, хранящимися в матрице `_geometryMomenta` [75]. Используя подход вращательного интегратора (алгоритм Верле), код осуществляет поэтапное обновление: 1. Полушаг: $w_{ij}(t+\frac{1}{2}\Delta t) = w_{ij}(t) + \frac{\pi_{ij}(t)}{2M} \Delta t$ – веса изменяются под действием имеющегося импульса (M – условная «инерционная масса» для геометрии) [76] [77]. 2. Расчёт сил: $F_{ij} = -\partial H/\partial w_{ij}$, где H – гамильтониан (энергетический функционал = $T + V$). Здесь $-\partial H/\partial w$ аналогично градиенту действия, но с учетом демпфирования. В коде рассчитываются два вклада: *кривизна* (как и раньше, положительная кривизна стремится увеличить вес, отрицательная – уменьшить, то есть сила $F_{curv} \approx + Ric$) и *материя* (массы всегда дают притяжение, увеличивают вес: $F_{mass} \sim + (m_i+m_j)$) [78]. Затем вводится диссипативный член $- \gamma \pi_{ij}$ (сила сопротивления пропорциональна импульсу, коэффициент демпфирования γ), чтобы геометрия не разгонялась беспредельно [79]. В результате $F_{ij} = Ric_{ij} + \frac{1}{2}\Gamma (m_i+m_j) - \gamma \,\pi_{ij}$ [80] [79]. 3. Полный шаг для импульса: $\pi_{ij}(t+\Delta t) = \pi_{ij}(t) + F_{ij}\, \Delta t$ – импульс ребра корректируется по найденной суммарной силе [81]. 4. Второй полушаг для веса: $w_{ij}(t+\Delta t) = w_{ij}(t+\frac{1}{2}\Delta t) + \frac{\pi_{ij}(t+\Delta t)}{2M}\Delta t$ [82] [77].

При таком подходе веса могут колебаться вокруг экстремума, обмениваясь энергией с моментами π_{ij}, что эквивалентно **гравитационным волнам** на графе (возмущения метрики). Введённый коэффициент демпфирования γ постепенно рассеивает энергию, так что при длительном времени система всё равно стабилизируется (затухающие колебания). Данная схема более физична, чем чистый градиентный спуск, и позволяет исследовать вопросы вроде: **образуется ли «кольцо» колебаний при схлопывании связи** (например, аналог излучения при слиянии кластеров) – конечно, в сильно упрощённом варианте. С точки зрения RQ-гипотезы, появление таких колебаний – это проявление **динамической гравитации** в эмергентном пространстве: само пространство-время (граф) не жёсткое, а способно нести волны кривизны.

Приближения и ограничения: Решение гравитационной динамики на графе – одна из самых амбициозных частей симуляции, и очевидно, что она работает приближенно. Прежде всего, использованная кривизна Формана – лишь одна из возможных; она учитывает только локальные треугольники и степени узлов. Этот подход ловит качественные черты (кластеризация vs ветвистость), но упускает метрические аспекты. В планах была реализация более точной **кривизны Олливье–Рикки** – основанной на идее оптимального транспорта и расстояний на графе [83]. Такая кривизна чувствительнее к глобальной геометрии, но её вычисление трудоёмко (требует оценки расстояний типа Вазерштейна между окружностями узлов). В репозитории даже есть функция для её расчёта с приближением через индекс Жаккара [84], однако по умолчанию используется более простой и быстрый Форман. Это компромисс: грубая кривизна Формана позволяет запустить симуляцию на сотнях узлов с приемлемой скоростью, тогда как более строгий расчет замедлил бы её на порядки. Следовательно, гравитация в RQSimForms – скорее **модель**, чем точное восстановление ОТО. Она воспроизводит качественный принцип: веса связей перераспределяются так, чтобы сеть стремилась к «оптимальному» распределению корреляций, где нет ненужной избыточной кривизны без материи (пустой граф расправляется, Ric стремится к 0, веса уравновешены) и нет провалов без поддержки (большая масса сгибает

граф, увеличивая рядом веса). Но количественно метрика, вытекающая из этих весов, может отличаться от настоящей метрики OTO, особенно на малых масштабах (эффекты порядка длины ребра, решётки).

Кроме того, требование **четырёхмерности** явно не наложено – наоборот, цель была посмотреть, получится ли нужная спектральная размерность ~ 4 сама собой. Как уже упоминалось, без специальных мер она не получилась (граф был слишком мелкий и фрагментировался). Разработчики встроили **валидатор спектральной размерности**: каждые N шагов код вычисляет d_S графа и проверяет, близка ли она к 4.0 в пределах 10% [85]. Если отклонение велико (например, граф сильно фрактален с $d_S \approx 2.5$), симуляция может вывести предупреждение или даже **приглушить часть физики** (например, временно отключить обновление спинорных полей, требующих 4-мерной структуры) [85]. Такой эмпирический подход – признание того, что на малых графах может не сформироваться классическое 4-мерное поведение, и тогда эволюция некоторых полей некорректна. Это опять же ограничение масштаба: в реальной Вселенной число степеней свободы гигантское, позволяя средним величинам стабилизироваться; в симуляции же 300 узлов могут случайно образовать «фрактал». К счастью, с усиленным притяжением и отжигом удаётся довести d_S ближе к 3.5–4 уже при нескольких сотнях узлов [16] [17].

Наконец, хоть гравитация целиком эмергентна в модели, константы ей сопутствующие (`GravitationalCoupling`, `CosmologicalConstant`) – это входные параметры. Их подбирают для лучшего поведения, как упомянуто выше. В идеале RQ-гипотеза хотела бы вывести их значения, но пока симуляция не настолько фундаментальна – она принимает их как настройки. Тем не менее, важно подчеркнуть: **никакого гравитационного потенциала или метрического тензора на фоне** не вводится. Всё, что «ощущают» узлы – это изменения весов их ребер. Например, локальное замедление времени (эффект гравитации) возникает автоматически: если узел окружён большой массой или высокой кривизной, веса ребер меняются так, что при расчёте локального собственного времени получается меньший шаг (см. следующий раздел о времени). Таким образом, *принцип эквивалентности и гравитационное замедление* имитируются: узлы с большим потенциалом имеют меньший прирост времени $\Delta \tau$ за цикл [86] [87]. Это напрямую следует из формул `ComputeLocalProperTime` и `GetTimeDilation` – они смотрят на локальную потенциальную энергию и кривизну вокруг узла и уменьшают его эффективный шаг времени [86] [88]. Так что даже без явного гравитационного поля, **в коде присутствует гравитационное замедление**: та часть алгоритма, где каждому событию назначается интервал Δt с поправкой на энергию/массу, служит аналогом замедления хода часов в гравитационном поле.

Резюмируя, симуляция *воплотила ключевой тезис:* гравитация = динамика геометрии графа. Элементы общих уравнений реализованы явно (кривизна, источник от массы, космологический член). Отличия – в упрощённости этих элементов и необходимости численных ухищрений (градиентный спуск вместо точного решения, диссипация, ограничение размеров). Но качественно RQSimForms **не вводит никакой гравитации извне** – она рождается из тех же корреляций, что и всё остальное, что и требовалось согласно RQ-гипотезе.

Реляционное время: внутренняя стрелка по Пейджу–Уоттерсу

Принцип гипотезы: В традиционной квантовой механике эволюция задаётся внешним параметром времени t, и решение проблемы времени в квантовой гравитации – одно из труднейших. Формализм Пейджа–Уоттерса предлагает решить это, введя **«часы» как часть**

замкнутой системы [89]. Общая волновая функция Вселенной $|\Psi\rangle$ подчиняется уравнению вида $\hat{H}*{total}|\Psi\rangle=0$ *(общее гамильтониановое ограничение, аналог уравнения Уилера–ДеВитта без внешнего времени)* [90]*. Однако внутри системы можно выделить подсистему (часы) с состояниями $|t\rangle_C$, по которым относительно остальных степеней свободы определять «время». Проще говоря, если состояние часов меняется предсказуемо и монотонно, то его можно использовать как параметр для состояния остальной системы: $|\psi(t)\rangle_S = \langle t|_C |\Psi\rangle$* [91]. Время здесь – не внешняя ось, а метка на состоянии квантовых часов. Количественно его можно определить, например, через расстояние Фубини–Стади между последовательными состояниями часов: $d\tau \sim \arccos|\langle \psi_C(t)|\psi_C(t+dt)\rangle|$ [92]. Для малых промежутков это расстояние пропорционально изменению состояния часов и может быть интерпретировано как приращение внутреннего времени [93]. Таким образом, гипотеза постулирует **полностью внутреннее время** – глобального параметра нет, но система сама может «отчитывать» время своим состоянием.

Реализация в коде: RQSimForms сначала была написана как типичная симуляция с внешним циклом по шагам *t*. Однако для соответствия RQ-принципам был внедрён механизм **реляционного времени**. Он состоит из двух связанных подходов: 1. **Внутренние часы на графе:** в графе выделяется набор узлов, которые играют роль часовой подсистемы. В коде есть функция `InitInternalClock(clockSize)`, которая выбирает определённое число узлов (обычно 5% от всех узлов) с самой высокой степенью (т.е. высокосвязанные «hub»-узлы) и помечает их как `IsClock` [94] [95]. Идея выбора хабов состоит в том, что такие узлы имеют наибольшее число связей и сильнее всего участвуют в динамике, поэтому их состояния могут служить репрезентативным «фоном» для эволюции всей сети. После выбора часов их текущее квантовое состояние сохраняется: например, берутся значения первой компонент волновой функции `_waveMulti` для каждого из узлов-часов и записываются в массив `_lastClockStateVector` [96] [97]. Это фиксирует «показание часов» в данный момент. 2. **Вычисление шага времени из изменения состояния часов:** когда приходит момент эволюционировать систему (например, в цикле или в событийном движке), симуляция не пользуется заранее заданным dt, а **вычисляет его исходя из изменений состояния часов**. В коде реализовано в методе `ComputeRelationalDtExtended()`: он снова берёт текущие значения волновой функции на узлах-часах (`_waveMulti[i*d]` для каждого узла-часа *i*, где *d* – размерность калибровочного пространства) и сравнивает с сохранёнными последними значениями [98] [99]. Рассчитывается суммарная квадратичная разность $\sum_k | \psi_k^{\text{current}} - \psi_k^{\text{last}}|^2$ – это фактически квадрат расстояния в гильбертовом пространстве между состоянием часов «сейчас» и «в прошлый раз» [99]. Затем вычисляется dt пропорционально корню из этой величины, с нормировкой на амплитуды [100]. То есть $dt \approx \sqrt{\sum |\Delta \psi|^2} / N_{\text{norm}}$. Эта формула – прямое численное воплощение идеи Fubini–Study: если состояние часов почти не изменилось, то прошло очень маленькое dt; если сильно изменилось – dt больше. Код ограничивает разумные пределы: dt *принудительно* ограничивается снизу 0.001 и сверху 0.1 [101], чтобы ни слишком замедлять симуляцию, ни допускать больших шагов, могущих нарушить точность. После получения этого dt он используется как **эффективный шаг эволюции** для всех процессов. А за тем состояние часов обновляется – вызывается `AdvanceInternalClockState()`, сохраняющая новый «прошлый» вектор часов на будущее [102].

Проиллюстрируем: допустим, часы – несколько узлов с их комплексными амплитудами. На предыдущем шаге у них был набор амплитуд ${\psi_i^{old}}$, теперь – ${\psi_i^{new}}$. Если изменения $\Delta \psi_i = \psi_i^{new} - \psi_i^{old}$ малы, то dt получится очень маленьким. Это означает, что эволюция системы *внутри себя* почти не продвинулась – времени прошло мало. Если же часы «сдвинулись» заметно (их волновая функция изменилась сильно), то это и есть мерило прошедшего времени. Таким образом, **симуляция избавляется от внешнего счётчика шагов**: каждый цикл определяет свою длительность адаптивно. В сочетании с

событийным движком описанная логика приводит к следующему: цикл `while` извлекает ближайшее событие, обновляет один узел (и его соседей) на рассчитанное для него локальное время $\Delta \tau$ [10] [103], затем корректирует глобальную переменную «время» (но только для логов) и перекидывает событие на следующий момент, рассчитанный через `ComputeLocalProperTime` и `GetTimeDilation` [11] [104] (эти функции учитывают локальную энергию и массу, как упоминалось выше). В результате глобальное время получается *неявным* – оно отслеживается только статистически как возрастание метки событий, но нигде в физических формулах не участвует напрямую. Такое время нельзя задать извне или контролировать линейно – оно «идёт» с той скоростью, с какой меняется конфигурация самой системы.

Приближения и ограничения: В теории Page–Wootters часы – квантовый объект, который должен запутаться с остальной системой по уравнению $\hat{H}_{total}|\Psi\rangle=0$. В симуляции RQSimForms, конечно, *прямо* не решается уравнение Уилера–ДеВитта (что было бы эквивалентно перебору всего состояния Вселенной). Вместо этого реализован практически удобный критерий: берём что-то вроде «главной компоненты» состояния часов (первую компоненту спиноров или внутренней волновой функции узла) [47] [105] и считаем её изменение. Это упрощение – *проекция* сложного множественного состояния на 1-мерную метрику. Оно работает, если часы изменяются достаточно гладко и доминирующая компонента может служить представителем общей динамики. В принципе, возможны случаи, когда часы изменили фазу, но первая компонента вернулась почти к исходному значению, и тогда симуляция ошибочно решит, что времени прошло мало (хотя на самом деле фаза сдвинулась на 2π). Такие тонкости – следствие компромисса: полное отслеживание $|\langle \Psi_C(t)|\Psi_C(t+dt)\rangle|$ требовало бы хранить *весь вектор состояния часов*, что накладно, поэтому берётся первый компонент как прокси. Для минимизации ошибок часы берут *хабами*, надеясь, что они не будут возвращаться циклически к одному и тому же состоянию слишком быстро.

Ещё один момент – **нормировка и масштаб времени**. Теория даёт лишь относительную шкалу через углы в пространстве состояний. Чтобы перевести это в число dt, код вводит некий нормировочный знаменатель (в реализации назван просто `normalization`) [106]. По сути, суммируется норма состояний до и после, и применяется $\sqrt{\Delta}/(\text{norm}+ \epsilon)$. Это фиксирует масштаб времени в произвольных единицах. Пороговые ограничения 0.001 и 0.1 на dt тоже означают, что модель не позволит внутреннему времени остановиться совсем или рвануть скачком. Эти ограничения введены сугубо для численной устойчивости, а не вытекают из теории (в реальности время может замедляться сколь угодно сильно вблизи гравитационного коллапса, например). Здесь же, если часы практически перестали меняться (скажем, вся система пришла в стационар), код всё равно будет отсчитывать $dt = 0.001$ минимально – иначе симуляция не продвинется. Это можно понимать как нижний предел разрешения времени в данной модели.

В событийном движке роль relational-тайм механизма несколько иная. Там нет единого dt на цикл – вместо этого *каждый узел* получает своё $\Delta \tau$ при срабатывании события [11] [104]. Но сама идея схожа: функция `ComputeLocalProperTime(node)` даёт базовый шаг 0.01, умноженный на фактор в зависимости от локальной потенциальной энергии узла [86]. Если у узла высокая энергия (например, он в плотном кластере или сильно возбужден), то шаг уменьшается (модель гравитационного замедления: $dt \propto 1/(1+ E\cdot0.1)$) [86]. Далее `GetTimeDilation(node)` также учитывает вклад кривизны (через корреляционную массу): чем больше локальная масса, тем больше дополнительное замедление [88] [107]. В итоге, когда событие (обновление) для данного узла завершено, его следующая активация планируется на глобальном времени $t + \Delta t * \text{dilation}$ [104]. Если вокруг узла много массы и энергии, $\text{dilation}<1$ и его следующее событие будет отнесено дальше в будущее (с его точки зрения время идёт медленнее, он реже обновляется). Тем самым, даже не прибегая к явным «часам»,

событийная система достигает эффекта **относительности одновременности**: разные узлы могут обновляться с разной частотой в глобальной последовательности, соответствуя тому, что в сильном гравитационном поле процессы идут медленнее.

В сравнении с оригинальной гипотезой, код RQSimForms явно демонстрирует правильный подход к проблеме времени: **исключено наличие внешнего t в физических уравнениях** – всё вращается вокруг внутренних переменных. Конечно, сама симуляция как программа всё равно запускается на шаги или события, но эти шаги калибруются по внутренним часам. Это огромный сдвиг от стандартных симуляций, где обычно жёстко задан dt. Здесь dt плавает, а при использовании чистого событийного режима – и вовсе неоднороден по узлам. Различия с идеальной картиной заключаются в том, что не решается полное уравнение $\hat{H}|\Psi\rangle=0$ (система эволюционирует не совершенно унистарно – в неё заложено некоторое эффективное время через алгоритм, пусть и внутреннее). Тем не менее, на том уровне, на каком численная модель способна, *время сделано относительным и зависимым от состояния системы*.

Отдельно можно упомянуть, что в теории Пейджа–Вуттерса подразумевается: если взять два различных набора часов (две разные подсистемы), их «время» не обязано идти синхронно – зависит от того, как они движутся относительно друг друга в квантовом состоянии. В RQSimForms часов выделяют одних (хабы), то есть имеется привилегированная подсистема, относительно которой всё остальное отсчитывается. Это упрощение: предположим, что эти часы можно ассоциировать с каким-то усреднённым космологическим временем (например, крупные кластеры, условно – галактики с часами). В принципе, можно было бы экспериментировать, что если выбрать другой набор часов, будет другой отсчёт. Но в рамках одной симуляции выбор фиксирован, и относительных расхождений между разными часами внутри одной вселенной не моделируется. Опять же, это ограничение техническое – для его обхода потребовалось бы параллельно вести несколько независимых clock-subsystem и сравнивать, что усложнило бы систему управления.

Выводы и соответствие теории

Симулятор RQSimForms продемонстрировал, как основные идеи реляционно-квантовой гипотезы могут быть воплощены в виде *событийно-ориентированной* вычислительной модели. Мы видим почти прямые соответствия: **вместо статичного пространства – граф корреляций**, меняющийся по внутренним законам; **вместо внешнего времени – внутренний механизм часов и событий**; **масса рождается из связей**, **взаимодействия – фазы на ребрах**, **гравитация – эволюция структуры графа**. Код написан таким образом, что **не опирается на фиксированные глобальные структуры**: нигде в физических обновлениях не используются абсолютные координаты или глобальные часы [5] [11]. Все вычисления – локальные и относительные, как того требует принцип реляционности.

Там, где симуляция отходит от идеалов гипотезы, это можно объяснить её собственными ограничениями: - **Ограниченность размеров графа и времени симуляции.** Идеи RQ рассчитаны на (в идеале) безграничную Вселенную или по крайней мере очень большую, где статистические свойства графа приближаются к непрерывным. В модели же 100–1000 узлов – это «крохотная Вселенная», склонная к флуктуациям и аномалиям. Многие несовпадения (фрагментация графа, неправильная размерность, необходимость принудительных мер) – следствие того, что малые системы ведут себя не как континуум. Разработчики вынуждены компенсировать это искусственно: вводить **отжиг гравитации**, подстраивать коэффициенты, контролировать размер шагов, чтобы выправить тенденции. Это не изъян теории, а издержки симуляции. Со временем, по мере увеличения N и улучшения алгоритмов, ожидается, что

поведение будет ближе к прогнозам (например, уже при $N\sim 300$ удалось добиться $d_S \approx 3.9$ и появления стабильных кластеров-масс [16] [26]). - **Численные приближения физических законов.** Формулы, заложенные в код (для кривизны, для обновления фаз, для шага времени), выбраны из соображений разумного баланса между физической обоснованностью и простотой. Они могут отличаться от «настоящих» – например, Форман против настоящей кривизны, диссипативный спуск вместо точно гамильтоновой эволюции в некоторых режимах, усечение фаз. Эти отличия могут приводить к конфликтам: в анализе отмечалось, что присутствуют «магические константы без физического обоснования» и искусственные ограничения, которые желательно бы убрать [108]. Их наличие – плата за управляемость модели. Впрочем, в обновлениях виден курс на сокращение этих допущений: например, добавлена проверка спектральной размерности для согласования размерности спиноров [85], намечено устранение дублирующихся определений массы [28] [29], подчистка остатков координатного подхода (центр масс и т.п. уже помечены как deprecated) [109]. Каждая такая доработка делает симуляцию ближе к самосогласованной реляционной модели. - **Отсутствие полного квантового описания.** RQSimForms пока по сути *квазиклассическая* симуляция: квантовые величины (волновые функции, спиноры) эволюционируют по усреднённым уравнениям, без случайного коллапса или суперпозиции нескольких графовых конфигураций. Это значит, что она проверяет скорее **консистентность исходных идей на уровне средних полей**, чем эмулирует конкретные результаты квантовых измерений. Например, нет реализации квантовых измерений над графом (хотя закладываются какие-то элементы – `ApplyMeasurementBackAction()` имитирует обратное действие измерения, снижая веса связей в подсистемах [110] [111]). Но в рамках самой гипотезы реляционная природа – это мета-уровень, а не рецепт, как имитировать коллапс волновой функции, так что отсутствие этого в симуляции не является отклонением от принципов, скорее просто следующий этап разработки.

Подведём итог: RQSimForms строго следует структуре **оригинального обзора** принципов реляционно-квантовой гипотезы, реализуя их в коде. Графовая динамика, массовые кластеры, калибровочные связи, геометрия-гравитация и внутреннее время – все эти компоненты нашли своё отражение. Некоторые реализованы почти один-в-один (напрямую вычисляется то, что задумано теорией), другие – с приближениями. В местах расхождений понятно, откуда они берутся: либо из вычислительных ограничений, либо из стремления включить больше физики (например, реальные поля) в модель. Важно, что **архитектура симуляции – событийная, асинхронная** – сама по себе является большим шагом к соответствию реляционному мировоззрению, где нет абсолютного времени. Это отличие от классических симуляторов (например, клеточных автоматов на жёсткой решётке с глобальным тактом) принципиально: RQSimForms «живет» по часам, сделанным из неё самой. Таким образом, проект демонстрирует возможность построения целостной *реляционной Вселенной* внутри компьютера и служит рабочей лабораторией для изучения последствий столь необычных фундаментальных предположений. Несмотря на упомянутые трудности, уже достигнуты результаты, качественно подтверждающие гипотезу (формирование кластеров-частиц, возникновение эффективной размерности пространства, локальное замедление времени и т.д.), что обнадёживает в дальнейшей разработке и усложнении модели.

---

1 2 3 4 16 17 20 24 27 28 29 30 31 56 57 61 63 65 74 89 90 91 92 93 108 109
DEEP_ANALYSIS_RQ_HYPOTHESIS.md
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/MD/DEEP_ANALYSIS_RQ_HYPOTHESIS.md

5 18 19 21 22 110 111 RQGraph.cs
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/RQSimulation/Core/RQGraph.cs

6 7 RQGraph.CoreHelpers.cs
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/RQSimulation/Topology/RQGraph.CoreHelpers.cs

8 14 15 ParallelEventEngine.cs
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/RQSimulation/GPUOptimized/ParallelEventEngine.cs

9 10 11 12 103 104 EventDrivenEngine.cs
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/RQSimulation/GPUOptimized/EventDrivenEngine.cs

13 25 26 53 54 83 84 85 README.md
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/RQSimulation/GPUOptimized/README.md

23 58 59 60 62 64 66 67 68 69 70 71 72 73 75 76 77 78 79 80 81 82
RQGraph.NetworkGravity.cs
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/RQSimulation/Gravity/RQGraph.NetworkGravity.cs

32 33 34 35 36 37 38 39 40 41 42 43 44 45 55 RQGraph.GaugePhase.cs
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/RQSimulation/Gauge/RQGraph.GaugePhase.cs

46 48 49 50 51 52 RQGraph.YangMills.Relational.cs
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/RQSimulation/Gauge/RQGraph.YangMills.Relational.cs

47 94 95 96 97 98 99 100 101 102 105 106 RQGraph.RelationalTime.cs
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/RQSimulation/Spacetime/RQGraph.RelationalTime.cs

86 87 88 107 RQGraph.EventDrivenExtensions.cs
https://github.com/yakforkgen/RqSimForms/blob/8ff45d5c3e0e5539b0b24d966343e21ee2da60c0/RQSimulation/GPUOptimized/RQGraph.EventDrivenExtensions.cs