СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Применение высокопроизводительных параллельных вычислительных систем (ВС) является одним из стратегических направлений развития современной компьютерной индустрии. Это обстоятельство вызвано не только принципиальным ограничением максимально возможного быстродействия обычных последовательных машин, но и постоянным появлением новых вычислительных задач, для решения которых возможностей существующих средств вычислительной техники всегда оказывается недостаточно. Моделирование реальных многомерных динамических процессов, описываемых системами обыкновенных дифференциальных уравнений (СОДУ), и представляет собой класс задач, для решения которых использование параллельных суперкомпьютеров не только оправдано, но и необходимо. Об этом свидетельствует известный список “большой вызов”, в котором такие задачи занимают одно из ведущих мест [1].

В последние годы максимальная производительность существующих параллельных вычислительных систем существенно выросла. Радикально изменилась технологическая база и архитектура. Но, по-прежнему, главным препятствием к внедрению практически всех параллельных архитектур является отсутствие эффективных параллельных методов. Как итог, проблема повышения эффективности функционирования параллельных вычислительных систем, несмотря на имеющиеся успехи, далека до полного разрешения, как в теоретическом, так и в практическом отношении.

Наиболее сложной является задача определения оптимальной алгоритмической и структурной организации параллельных вычислений с целью достижения достаточных характеристик эффективности [2-3]. На сегодня не вызывает сомнений, что реализовать потенциальные возможности параллельных компьютеров возможно прежде всего на основе проведения целенаправленной теоретической работы по построению новых и адаптации существующих методов решения сложных научно-технических задач, поскольку перспективы развития суперЭВМ в ближайшем будущем будут определяться успехами не столько электроники, сколько вычислительной математики.

Большой вклад в разработку теоретико-математических основ распараллеливания вычислений внесли Воеводин В.В., Воеводин Вл. В. [3-9], Гергель В.П. [10-11], Gupta A., Kumar D. [12-14]. Проблемам проектирования новых многопроцессорных архитектур и разработке эффективных топологий посвящены работы групп ученых под руководством Забродина А.В. [15-17], Самофалова К.Г. [18], Корнеева В.Д. [19-20]. Методы повышения эффективности специализированных параллельных вычислительных систем исследовались в работах Малиновского Б.Н. [21-23], Боюна В.П. [21-25], Каляева И.А. [26-27]. Несмотря на результаты обширных исследований в области параллельных вычислений, работы в этом направлении не утрачивают своей значимости и требуют дальнейшего развития в связи с массовым распространением параллельных вычислительных систем.

Вышеперечисленные проблемы и определяют направленность монографии, которая посвящена разработке и обоснованию параллельных методов решения широкого класса научно-технических динамических задач с сосредоточенными параметрами. Актуальность и своевременность исследований, предложенных в работе, обуславливается широким внедрением высокопроизводительных параллельных компьютеров на предприятиях и в организациях Украины.

В первой главе монографии приведен обзор современных архитектур и классов параллельных ВС, описаны базовые топологические структуры, введены основные принципы и методика распараллеливания; выполнен анализ динамических характеристик качества параллельных вычислений; проанализированы существующие методы решения динамических задач, описываемых СОДУ. На основе обзора известных работ сделаны выводы о состоянии проблемы в исследуемой области, обосновано принятое направление разработок, сформулирована цель исследований.

Во второй главе предложены и исследованы параллельные явные методы численного решения СОДУ с оценкой локальной апостериорной погрешности, используемые для моделирования динамических систем с сосредоточенными параметрами. Разработанные методы ориентированы на использование в многопроцессорных системах SIMD-, MIMD-, CLUSTER-архитектур с распределенной памятью и различными топологиями соединения

процессорных элементов и процессоров. В разделе использован подход, основанный на сочетании иерархической декомпозиционной методики и графов влияния, позволяющий эффективно распараллеливать последовательные алгоритмы сложных задач, а также осуществлять их отображение на современные многопроцессорные структуры. Исследована масштабируемость параллельных методов с использованием математического аппарата изоэффективного анализа.

Третья глава посвящена разработке, обоснованию и исследованию эффективности параллельных методов решения динамических задач на основе неявных одношаговых схем с контролем локальной погрешности. В главе предложены и теоретически обоснованы новые параллельные методы решения ОДУ с использованием неявных одношаговых разностных схем: k-точечный метод с правилом дублирования шага; вложенные блочные методы с применением двух независимых численных схем и на основе специально подобранных формул разных порядков точности; экстраполяционные блочные методы на основе технологии Ричардсона-Ромберга. Проведен сравнительный анализ неявных одношаговых методов решения нелинейной задачи Коши на основе блочных k-точечных и полностью неявных методов типа Рунге-Кутты. Осуществлено обобщение методов, реализующих разностные многоточечные одношаговые блочные схемы для решения систем ОДУ**.**

Для каждого из приведенных методов рассчитаны аналитические трудоемкости численного решения, получены экспериментальные характеристики параллелизма, которые свидетельствуют о высокой эффективности блочных методов. Исследована эффективность полученных параллельных алгоритмов в зависимости от размерности СОДУ, количества процессоров, типа параллельной ВС, латентности и времени передачи данных при различных топологиях.

В четвертой главе монографии предложены специальные параллельные методы интегрирования линейных динамических задач с постоянными коэффициентами. Получены вычислительные схемы интегрирования, соответствующие параллельным экспоненциальным методам решения линейной задачи с контролем погрешности на шаге. Для ускорения наиболее ресурсоемких операции экспоненциального метода матричного произведения, предложен параллельный алгоритм на основе комбинации быстрого рекурсивного и систолического

алгоритмов, а также модификация систолического алгоритма, которая снимает ограничения на порядок перемножаемых матриц.

Исследования, положенные в основу монографии, проводились Фельдманом Л.П. и Назаровой И.А. на кафедре прикладной математики и информатики ГВУЗ «Донецкий национальный технический университет» в рамках выполнения научно-исследовательских работ по госбюджетным темам фундаментальных исследований, утвержденных МОН Украины: «Научные основы оптимизации структур высокопроизводительных ВС и методы реализации параллельных алгоритмов», «Методы алгоритмизации, топологического отображения и оптимизации структур параллельных и распределенных ВС», «Разработка алгоритмических методов повышения эффективности моделирования сложных систем в параллельных вычислительных средах» [28-29]. Основные положения и результаты данной исследовательской работы докладывались, обсуждались и получили положительную оценку на следующих конференциях и семинарах: международные научно-практические семинары «Высокопроизводительные параллельные вычисления на кластерных системах», международные научно-технические конференции «Интеллектуальные и многопроцессорные системы», международные конференции по вычислительной механике и современным прикладным программным системам, международные научно-практические конференции студентов, аспирантов и молодых ученых “Системный анализ и информационные технологии”, международные научные конференции «Моделирование», международные конференции по неравновесным процессам в соплах и струях, международные конференции «Параллельные вычисления и задачи управления», международные конференции «Моделирование и компьютерная графика».

Для проведения моделирования, исследований и экспериментов использовались пакет Mathematica (Wolfram Research Inc.) [30-31] и стандарт MPI (Message Passing Interface) для языка C++ [30-36]. Тестирование алгоритмов проводилось на основе тестов для СОДУ, разработанных НИВЦ МГУ [37]. Результаты исследований, изложенные в монографии, были апробированы на международных конференциях и семинарах, авторами опубликовано более 50 печатных работ по тематике исследований [38-92].

ГЛАВА 1

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ И ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ

1.1. Основные классы современных параллельных компьютеров

Современный этап развития вычислительной техники характеризуется применением огромного количества высокопроизводительных параллельных вычислительных систем. В них используются различные аппаратные составляющие: процессоры (Intel, IBM, AMD, HP, NEC, Cray), сетевые карты (Ethernet, Myrinet, Infiniband, SCI). Они функционируют под управлением различных операционных систем (Unix/Linux, Windows) и реализуют различное прикладное программное обеспечение. Большое разнообразие параллельных систем породило необходимость введения для них эффективной системы классификации, представляющей возможность дифференциации существенно различных реальных и проектируемых вычислительных систем, а также обладающей наглядностью, простотой и практической целесообразностью.

Одной из первых и наиболее распространенных схем классификации архитектур вычислительных систем является **систематика Флинна** [93-94]. В ее основу положено понятие потоков выполняемых команд и обрабатываемых данных (рис. 1.1).

В результате такого подхода различают следующие основные типы параллельных систем:

1) **SISD (Single Instruction, Single Data)** – одиночный поток команд и одиночный поток данных;

2) **SIMD (Single Instruction, Multiple Data)** – одиночный поток команд и множественный поток данных;

3) **MISD (Multiple Instruction, Single Data)** – множественный поток команд и одиночный поток данных;

4) **MIMD (Multiple Instruction, Multiple Data)** – множественный поток команд и множественный поток данных.

SISD **Single Instruction, Single Data** **Одиночный поток команд, одиночный поток данных**	**SIMD** **Single Instruction, Multiple Data** **Одиночный поток команд, множественный поток данных**
MISD **Multiple Instruction, Single Data** **Множественный поток команд, одиночный поток данных**	**MIMD** **Multiple Instruction, Multiple Data** **Множественный поток команд, множественный поток данных**

Рисунок 1.1 – Классификация вычислительных систем по Флинну

Классификация Флинна до сих пор используется при начальной характеристике компьютерных систем, однако она имеет определенные недостатки. Практически все виды параллельных компьютеров, несмотря на их существенную разнородность, относятся к одной группе MIMD. Как результат, многими исследователями предпринимались неоднократные попытки детализации этой систематики [95-102]. Так, MIMD архитектуры далее классифицируются в зависимости от физического способа организации памяти:

1) компьютеры с распределенной памятью;

2) компьютеры с общей памятью;

3) компьютеры с виртуальной общей (разделяемой) памятью.

Некоторые авторы [11, 95] вычислительные системы с общей памятью называют **мультипроцессорами**, а системы с распределенной памятью – **мультикомпьютерами**. В соответствии с типом доступа к памяти вводятся основные классы параллельных компьютеров: SMP, MPP, PVP, NUMA- архитектуры и кластеры.

Симметричные мультипроцессорные системы (SMP) [103-105] обычно состоят из нескольких одинаковых процессоров и массива общей памяти. Все процессоры имеют равные возможности по доступу к единому адресному пространству, такие ВС называют системами с однородным доступом к памяти **(Uniform Memory Access – UMA**). Наличие общей памяти упрощает взаимодействие процессоров между собой, однако накладывает сильные ограничения на их число – не более 32-64 в реально существующих системах. Данная архитектура не

является хорошо масштабируемой и отказоустойчивой. Примерами SMP-систем могут служить HP 9000 V-class, N-class; SMP-сервера, рабочие станции на базе процессоров Intel.

Массивно-параллельные системы (MPP) [103-105] состоят из нескольких однородных вычислительных узлов, как правило, RISC-архитектуры. Каждый такой узел имеет свою локальную память, один или несколько центральных процессоров и иногда жесткий диск, причем прямой доступ к памяти других узлов невозможен **(No Remote Memory Access – NORMA)**. Узлы обычно связаны через коммуникационную среду. Доступ к удаленной памяти реализуется в рамках модели передачи сообщений на основе библиотек MPI (Message Passing Interface), PVM (Parallel Virtual Machine). Системы массового параллелизма хорошо масштабируются, общее число процессоров может достигать нескольких тысяч. Основной недостаток таких систем заключается в сложности организации обмена информацией между процессорами. Примерами MPP-систем служат: IBM RS/6000 SP2, Intel PARAGON, компьютеры серии МВС.

Основным признаком **параллельных векторных систем (PVP)** является наличие специальных векторно-конвейерных процессоров, в которых предусмотрены команды однотипной обработки векторов независимых данных, эффективно выполняющиеся на конвейерных функциональных устройствах. Примеры PVP-систем: NEC SX-4/SX-5, CRAY J90/T90.

Системы с неоднородным доступом к памяти (NUMA – Non Uniform Memory Access) представляют собой нечто среднее между SMP и MPP. В NUMA память физически распределена, но логически общедоступна. Поддерживается единое адресное пространство, аппаратно поддерживается доступ к удаленной памяти. При этом доступ к локальной памяти процессоров в несколько раз быстрее, чем к удаленной памяти. Масштабируемость NUMA-систем ограничивается объемом адресного пространства и возможностями операционной системы. Наиболее известные NUMA-системы: SGI Origin 2000, Sun HPC 10000, IBM/Sequent NUMA-Q 2000, SNI RM600.

В последние годы во всем мире широкое распространение получили **кластерные системы** как дешевая альтернатива суперкомпьютерам [10,103].

Система требуемой производительности собирается из готовых серийно выпускаемых компьютеров, объединенных с помощью некоторого серийно выпускаемого коммуникационного оборудования. Кластеры являются логическим продолжением идей, заложенных в архитектуре MPP-систем. Развитие коммуникационных технологий, а именно, появление высокоскоростного сетевого оборудования и специального программного обеспечения, реализующего механизм передачи сообщений над стандартными сетевыми протоколами, сделали кластерные технологии общедоступными.

Привлекательной чертой кластерных технологий является то, что они позволяют для достижения необходимой производительности объединять в единые вычислительные системы компьютеры самого разного типа, начиная от персональных компьютеров и заканчивая мощными суперкомпьютерами. При объединении в кластер компьютеров разной мощности или разной архитектуры, говорят о гетерогенных или неоднородных кластерах, в противном случае о гомогенных или однородных. Обычно используются либо простые однопроцессорные персональные компьютеры, либо 2-х или 4-х–процессорные SMP-серверы. При этом не накладывается никаких ограничений на состав и архитектуру узлов. Каждый из узлов может функционировать под управлением своей собственной операционной системы. Чаще всего применяются стандартные ОС: Linux, UNIX, Windows NT. Для связи узлов используется одна из стандартных сетевых технологий таких, как Fast/Gigabit Ethernet на базе шинной архитектуры или коммутатора. Ряд фирм предлагают специализированные решения на основе более скоростных сетей, таких как SCI фирмы Scali Computer, а также Mirynet и InfiniBand [106-109].

В настоящее время в **списке TOP500** [110] самых высокопроизводительных систем кластеры составляют большую часть, это Beowulf-кластеры, NCSA NT Supercluster, Thunder Intel Itanium2 [103]. В списке Top50 [111] для стран СНГ представлен кластер с пиковой производительностью 2.2 TFlop/s Института Кибернетики НАН Украины и компьютер Киевского политехнического института с реальной производительностью 1.5 TFlop/s. Сегодня не составляет большого труда создать небольшую кластерную систему, объединив вычислительные мощности компьютеров отдельной лаборатории или

учебного класса. Однако дешевизна кластерных систем оборачивается большими накладными расходами на взаимодействие параллельных процессов между собой, что сильно сужает потенциальный класс решаемых задач. Еще одним направлением развития современной параллельной индустрии является комбинирование различных архитектур в одной системе и построение **гибридных систем**.

Существуют несколько других классификаций ВС для параллельной обработки данных [5-9]. Более детальная систематизация компьютеров класса MIMD предложена Р. Хокни. Джонсон Е. проводит классификацию MIMD архитектур на основе структуры памяти и реализации механизма взаимодействия и синхронизации между процессорами. Т. Фенг классифицирует ВС на основе двух простых характеристик, числа бит n в машинном слове, обрабатываемых параллельно и числа слов m, обрабатываемых одновременно данной вычислительной системой. Особого внимания заслуживает способ структурной нотации [10-11], позволяющий с высокой точностью описать многие характерные особенности компьютерных систем.

В завершении из всего сказанного важно отметить, что огромное многообразие архитектурных решений для параллельных вычислительных систем приводит к тому, что создание эффективного и масштабируемого алгоритмического и программного приложений для них является трудоемким процессом. Поэтому на сегодня не вызывает сомнений, что реализовать потенциальные возможности параллельных компьютеров можно только на основе проведения целенаправленной теоретической работы по построению новых и адаптации существующих алгоритмов решения сложных научно-технических задач большой размерности.

1.2. Характеристика типовых топологий и коммуникационных примитивов для операций передачи данных

Параллельные ВС имеют в своем составе поле процессорных элементов, а также один или несколько модулей памяти. Эти функциональные компоненты должны быть связаны через соответствующие коммутационные системы.

Все высокоуровневые концепции коммуникаций, такие как общая память: реальная или виртуальная или обмен сообщениями, реализуются в параллельных системах некоторыми наличными физическими структурами. В силу специфики задач, решаемых на параллельных ВС, структура линий связи (топология сети передачи данных) должна удовлетворять различным требованиям. Во-первых, должна обеспечиваться связь между двумя любыми процессорами или модулями. Кроме этого, должно поддерживаться максимальное количество одновременных связей, чтобы средства коммутации не ограничивали параллельную обработку информации [10-11].

В качестве основных характеристик топологии сети передачи данных наиболее широко используется следующий ряд показателей:

1) p – число процессоров или процессорных элементов (ПЭ);
2) A – диаметр или коммуникационная длина;
3) V – количество линий связи у одного процессора или ПЭ;
4) связность;
5) ширина бинарного деления;
6) стоимость.

Диаметр – показатель, определяемый как максимальное расстояние между двумя процессорами сети (под расстоянием обычно понимается величина кратчайшего пути между процессорами). Данная величина может характеризовать максимально-необходимое время для передачи данных между процессорами, поскольку время передачи обычно прямо пропорционально длине пути.

Связность (*connectivity*) – показатель, характеризующий наличие разных маршрутов передачи данных между процессорами сети; конкретный вид данного показателя может быть определен, например, как минимальное количество дуг, которое надо удалить для разделения сети передачи данных на две несвязные области.

Ширина бинарного деления (*bisection width*) – показатель, определяемый как минимальное количество дуг, которое надо удалить для разделения сети передачи данных на две несвязные области одинакового размера.

Стоимость – показатель, который может быть определен, например, как общее количество линий передачи данных в многопроцессорной вычислительной системе.

К числу типовых топологий обычно относят следующие схемы: **линейка/кольцо, решетка/тор, гиперкуб, клика и звезда.** Для сравнения в таблице 1.1 приводятся значения перечисленных показателей для различных топологий сети передачи данных.

Таблица 1.1.

Характеристики топологий сети передачи данных
(p – количество процессоров)

Топология	Диаметр	Ширина бисекции	Связность	Стоимость
Полный граф	*1*	$p^2/4$	*p-1*	*p(p-1)/2*
Звезда	*2*	*1*	*1*	*p-1*
Полное двоичное дерево	*2log((p+1)/2)*	*1*	*1*	*p-1*
Линейка	*p-1*	*1*	*1*	*p-1*
Кольцо	$\lfloor p/2 \rfloor$	*2*	*2*	*p*
Решетка N=2	$2(\sqrt{p}-1)$	$\sqrt{p}$	*2*	$2(p-\sqrt{p})$
Решетка-тор N=2	$2\lfloor \sqrt{p}/2 \rfloor$	$2\sqrt{p}$	*4*	*2p*
Гиперкуб	*log p*	*p/2*	*log p*	*(p log p)/2*

Линейка (*linear array*) – представляет собой линейный массив процессоров. Каждый процессор соединен с двумя соседними, за исключением концевых, которые имеют по одному соединению. Достоинство этой схемы в ее простоте: из каждого процессора выходит не более двух линий связи. Недостаток состоит в том, что данные, прежде чем достичь своего конечного назначения могут пройти через несколько промежуточных процессоров. Максимальное расстояние между процессорами равно $p-1$.

Топология кольцо (*ring*) (рис. 1.1) получается из линейки путем соединения первого и последнего процессоров. Связи в кольце могут быть однонаправленными или двунаправленными. Кольцевая структура сохраняет достоинства линейки: число связей на один процессор

остается равным 2 и сокращается максимальное расстояние между процессорами – $p/2$.

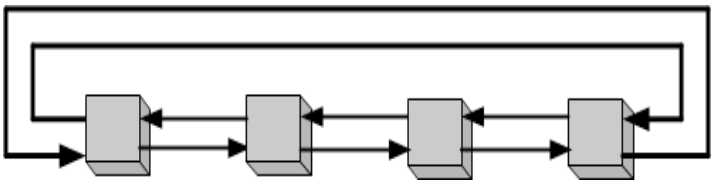


Рисунок 1.1 – Топология кольцо или 1D - тор

Решетка (матрица, сетка – *mesh*) представляет собой систему, в которой процессоры расположены в виде правильной двумерной решетки и каждый процессор (кроме крайних) соединен с четырьмя соседями. Эта схема соединений ”север-юг-восток-запад” была применена в компьютере IILIAC-IV, 64 процессора которого составляли массив 8×8. Достоинством схемы является простота, а недостаток состоит в том, что при обмене между отдаленными процессорами данные должны пройти через ряд промежуточных процессоров. Соединив в решетке граничные процессоры линиями связи, получим замкнутый вариант решетки - 2D-**тор** (рис. 1.2).

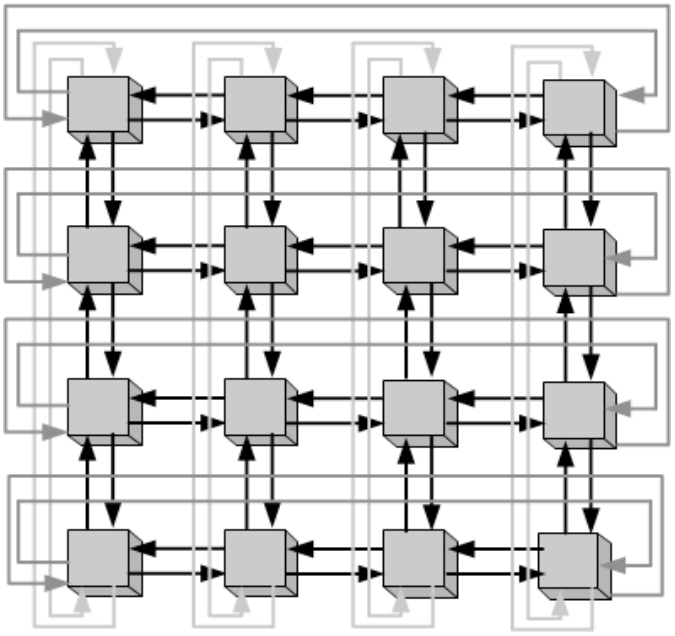


Рисунок 1.2 – Топология замкнутая решетка или 2D-тор

Тороидальные 2D-схемы соединения имеют диаметр, пропорциональный $\sqrt{p}$ [19]. Квадратный тор с четырьмя линиями связей у каждого процессора обладает следующими топологическими характеристиками: $V = 4$ и $A = \sqrt{p}$.

Кроме двумерных имеются схемы с трехмерным массивом процессоров, в которых каждый процессор соединен с шестью ближайшими соседями. В кубической 3D-решетке процессоры размещены в пространстве куба, максимальное расстояние между процессорными элементами пропорционально $\sqrt[3]{p}$.

Структура **гиперкуб (*hypercube*)** является частным случаем решетки, когда по каждой размерности сетки имеется только два процессора, то есть гиперкуб содержит $p = 2^N$ процессоров при размерности N. При трехмерной схеме соединения процессоры размещены в вершинах трехмерного куба, а ребра куба играют роль локальных связей между процессорами. Каждый процессор соединен с тремя ближайшими соседями. Схема соединения четырехмерного куба приведена на рис. 1.3.

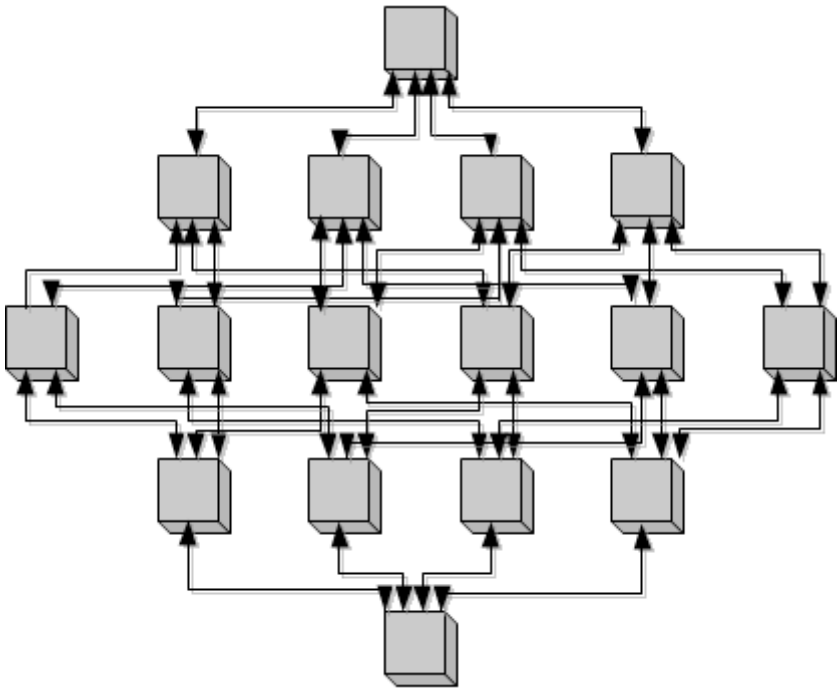


Рисунок 1.3 – Топология гиперкуб $p = 2^4$

При гиперкубовой архитектуре число связей между процессорами небольшое: в N-мерном гиперкубе каждый процессор имеет N соседей. Небольшим по сравнению с числом процессоров оказывается и диаметр системы, равный $log_2 p$ [10,103,112]. Недостатком описанной архитектуры является существование практического предела увеличения размерности гиперкуба. Топология гиперкуб предоставляет возможность моделировать некоторые важные типы связей: линейка, кольцо, решетка.

Гиперкуб – универсальная и одна из наиболее эффективных систем связи, ее концепции использованы в вычислительных системах Intel iPSC, Ncube Corp. и многих других.

Для кластерных ВС одним из широко используемых способов построения коммуникационной среды является применение переключателей. При этом топология сети представляет собой полный **граф или клику** (***completely-connected graph or clique***) (рис. 1.4). Такая коммуникационная структура предусматривает существование линий связи между любыми двумя процессорами в системе.

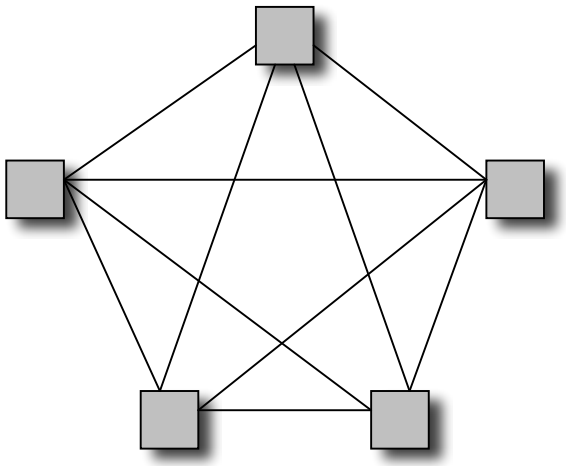


Рисунок 1.4 – Сеть с топологией клика

Полная связанность системы из p процессоров требует, чтобы из каждого процессора исходила $p-1$ линия связи, что делает стоимость такой схемы достаточно большой при большом p. Достоинством такой схемы соединения является минимальная величина диаметра, равная

единице. При применении в кластерных системах для топологии клика имеются ограничения на одновременность выполнения коммуникационных операций. Так, в каждый момент времени может выполняться взаимодействие между двумя процессорами, либо взаимодействие непересекающихся пар процессоров. Оценка времени выполнения коммуникационных операций также может быть выполнена по линейной модели при $l = 1$. Заметим, что существуют более точные модели оценки операций передачи данных в кластерах, использующие понятие пакета данных [10-11,112].

Звезда (*star*) – топологическая система, в которой все процессоры имеют линии связи с некоторым управляющим процессором. Данная топология является эффективной, например, при организации централизованных схем параллельных вычислений.

Реальные параллельные вычислительные системы обычно используют несколько различных схем соединения процессоров. Это позволяет сочетать лучшие качества известных топологий и минимизировать недостатки. Параллельная ВС MasPar MP-1 функционирует с использованием двух сетей: решетчатой структуры и трехступенчатой сети Клосса. Гибридная система Finite Element Mashine использует решетку для связей между соседними процессорами и шину для более отдаленных. Компьютер Connection Mashine имеет 16 полносвязных процессоров в группе и группы объединяются сетью топологии гиперкуб. Другим способом улучшить характеристики схем коммутации является построение систем с изменяемой конфигурацией. Изменять шаблон соединений можно как статически (до выполнения программы), так и динамически (в ходе ее выполнения).

Классическое определение параллельной вычислительной системы [5] обязательно включает в себя процессоры, модули памяти и коммутирующую сеть. Коммутирующая сеть чаще всего является сдерживающим фактором к достижению пиковой производительности вычислительной системы.

Обзор параметров существующих технологий, приведенный на рис. 1.5, основан на материалах [8-13] и дает основание проводить теоретический сравнительный анализ алгоритмов с параметрами, характерными для реальных систем.

Технология	**Топология сети**	**Латентность* (мкс)**	**Скорость передачи * Мб/с**	**Источники**
Gigabit Ethernet	Коммутируемая сеть, не блокирующая коммутация позволяет использовать нескольких десятков портов	50	70	[Т-платформы] [parallel.ru]
Myrinet 2000	Коммутируемая сеть, элементом коммутации является матрица 8x8. Коммутаторы на её основе поддерживают до 128 портов. Для построения больших сетей используются различные варианты топологии Fat Tree, максимум производительности достигается на Clos Network	10	200	[Т-платформы] [parallel.ru] [myri.com]
SCI (Scalable Coherent Interface)	Кольцо, двух- или трехмерный тор, а также коммутируемые кольца. В связи с такой топологией, при увеличении размеров тора происходит насыщение аппаратной пропускной способности, поэтому нецелесообразно строить кластеры с размером тора больше 6—8 по каждому измерению	4	325	[Т-платформы] [parallel.ru] [dolphin]
InfiniBand	Коммутируемая сеть, с использованием Fat Tree для больших конфигураций, коммутаторы поддерживают до 96 портов	7	1000 (пиковая)	[Т-платформы] [intel] [parallel.ru]

Рисунок 1.5 – Обзор параметров сетей коммутаций

Для оценки времени выполнения операции передачи одного сообщения объемом V байт между двумя процессами, локализованными на различных процессорах при распределенной памяти, используется следующая **линейная модель Хокни** [10-11]:

$$T_{p-p} = t_s + t_w \cdot V \cdot l, t_w = y / B, \quad (1.1)$$

где t_s – латентность, длительность подготовки сообщения для передачи;

l – длина маршрута;

t_w – время передачи одного байта;

y – число байт в слове;

B – пропускная способность канала передачи данных (байт/секунда).

Эта модель подразумевает использование способа передачи неделимых блоков информации, т.е. сообщений, если объемы пересылаемых данных невелики.

Поскольку параллельные методы решения динамических задач с сосредоточенными параметрами используют структурное информационное взаимодействие по типу коммуникаций, предоставляется возможность разработать единые коммуникационные примитивы для различных операций передачи данных в различных топологиях.

Рассмотрим наиболее используемые топологии: **линейка/кольцо, решетка/тор, гиперкуб** и три коммуникационные операции:

1) передача данных между двумя процессорами сети – операция типа “**точка-точка**” или “**point-point**”;

2) передача данных от одного процессора всем остальным процессорам сети – операция “**один-всем**” или “**one-to-all**”;

3) передача данных от всех процессоров сети всем процесорам сети – множественная пересылка “**все-всем**” или “**all-to-all**”.

Трудоемкость одиночной операции пересылки данных между двумя процесорам может быть получена путем подстановки длины максимального пути (диаметра сети) в выражение (1.1). Для вычисления времени выполнения множественной пересылки необходимо выбрать **алгоритм маршрутизации**. К числу наиболее распространенных оптимальных алгоритмов передачи данных

относятся **методы покоординатной маршрутизации [**11]. Идея этих методов заключается в том, что поиск путей передачи данных осуществляется последовательно для каждой размерности рассматриваемой топологии.

Для кольцевой топологии (рис. 1.1) каждый процессор может инициировать рассылку сообщения в каком-либо выбранном направлении по кольцу. Без существенных ограничений предполагаем, что любой процессор имеет возможность осуществлять одновременно прием и передачу данных (двунаправленные линки). Таким образом, для топологии кольцо время выполнения одиночной пересылки данных равно:

$$T_{p-p}^{R} = t_s + V \cdot t_w \cdot \lfloor p/2 \rfloor, \tag{1.2}$$

передача данных от одного процессора всем остальным определяется соотношением:

$$T_{one-to-all}^{R} = t_s + V \cdot t_w \cdot \lceil p/2 \rceil, \tag{1.3}$$

время выполнения множественной пересылки “все-всем” составляет:

$$T_{all-to-all}^{R} = (t_s + V \cdot t_w) \cdot (p-1). \tag{1.4}$$

Для топологии решетка-тор (рис. 1.2) одиночная операция пересылки данных требует следующего времени для исполнения с учетом модели (1.1) и диаметра топологии:

$$T_{p-p}^{M} = t_s + 2V \cdot t_w \cdot \lfloor \sqrt{p}/2 \rfloor. \tag{1.5}$$

Пересылка данных от одного процессора всем остальным в условиях топологии тор может быть получена из этого же способа передачи для кольцевой топологии, выполненного в два этапа:

$$T_{one-to-all}^{M} = t_s + 2V \cdot t_w \cdot \lceil \sqrt{p}/2 \rceil. \tag{1.6}$$

Множественная рассылка сообщений также может быть выполнена при помощи алгоритма, получаемого обобщением способа передачи данных для кольцевой структуры сети на основе идеи покоординатной маршрутизации:

$$T_{all-to-all}^{M} = 2t_s(\sqrt{p}-1) + V \cdot t_w \cdot (p-1). \tag{1.7}$$

Для топологии гиперкуб (рис. 1.3) время выполнения одиночной операции пересылки данных равно:

$$T_{p-p}^{H} = t_s + V \cdot t_w \cdot \log_2 p. \tag{1.8}$$

Время выполнения операции передачи данных от одного процессора всем остальным в топологии гиперкуб составляет:

$$T^{H}_{one-to-all} = \left(t_s + V \cdot t_w\right) \cdot log_2\, p. \qquad (1.9)$$

Алгоритм выполнения множественной рассылки сообщений для гиперкуба может быть получен путем обобщения способа передачи данных “все-всем” для топологии решетка на размерность гиперкуба $lp = log_2\, p$. Каждому процессору ставится в соответствие двоичный эквивалент его номера. Процессоры, имеющие непосредственное соединение в гиперкубе, будут иметь номера, отличающиеся друг от друга только одним разрядом. На каждом этапе $i, i = \overline{1,lp}$ алгоритма функционируют все процессоры сети, которые обмениваются данными со своими соседями по i размерности и формируют объединенные сообщения:

$$T^{H}_{all-to-all} = \sum_{i=1}^{lp} (t_s + 2^{i-1} \cdot V \cdot t_w) = t_s \cdot log_2\, p + t_w \cdot V \cdot (p - 1) \qquad (1.10)$$

Особенностью кластерных систем является использование **стандартных сетевых технологий**. В настоящий момент наиболее популярными по рейтингу TOP-50 для СНГ являются Gigabit Ethernet, InfiniBand, Myrinet (рис. 1.6). Список приведен в порядке убывания количества кластеров, использующих ту или иную технологию.

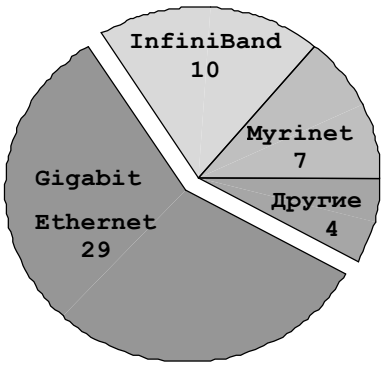


Рисунок 1.6 – Сетевые технологии в рейтинге ВС СНГ TOP-50

Аналогичная тенденция наблюдается и в мировом масштабе [97-103]. Наибольшее применение по-прежнему находит технология Gigabit Ethernet.

Эффективность использования той или иной сетевой технологии определяется ее числовыми параметрами. В таблице 1.1 на основании материалов [105-109] приведены коммуникационные константы обмена для наиболее известных коммуникационных сетей.

Таблица 1.1

Коммуникационные константы для сетевых технологий

Сетевая технология	Латентность (мкс)	Скорость передачи Мб/с
Gigabit Ethernet	50	70
Myrinet 2000	10	200
SCI (Scalable Coherent Interface)	4	325
InfiniBand	7	1000 (пиковая)

Обзор свойств типовых топологий соединения процессоров и их числовых характеристик, введение коммуникационных примитивов операций передачи данных для топологий, дают основания проводить теоретический сравнительный анализ выполнения различных параллельных алгоритмов на основе параметров, характерных для реальных высокопроизводительных многопроцессорных систем.

1.3. Иерархическая декомпозиционная технология разработки параллельного алгоритма

Развитым и общепризнанным математическим аппаратом разработки параллельных алгоритмов являются графовые модели: графы влияния, зависимостей, то есть некоторые **информационные графы алгоритмов** [3-5]. Однако для реальных задач применение этой модели сопряжено с большими трудностями. Для прямого описания всех вершин и дуг соответствующего графа необходимы большие затраты компьютерной памяти, а для анализа используются алгоритмы, которые практически все имеют высокую вычислительную сложность, чаще всего экспоненциальную. Поэтому для разработки параллельных алгоритмов задач практического уровня сложности используется многоэтапный процесс, начинающийся анализом задачи, выбором модели вычислений, декомпозицией задачи на независимые

параллельные процессы и заканчивающийся вопросами исследования эффективности. Такая технология распараллеливания получила название – **иерархической декомпозиционной методики** [11,97-98].

Первоначально вычислительная задача разбивается на подзадачи, анализируются информационные взаимосвязи между ними, определяется множество макроопераций, оценивается эффективность потенциального и реального параллелизма. В случае неудовлетворительного результата на любом из этапов схема разбиения изменяется, и весь процесс повторяется с начала до тех пор, пока характеристики полученного параллельного алгоритма не станут удовлетворительными или путем полного перебора не станет ясно, что добиться качественного распараллеливания невозможно. **Графы влияния** используются для построения параллельного алгоритма как на верхнем уровне для анализа взаимодействия подзадач, так и на нижнем уровне для распараллеливания отдельных макроопераций [3-9].

Введение **макроопераций** приводит к более компактному представлению графа алгоритма, значительно упрощает проблему выбора эффективных способов распараллеливания вычислений, позволяет использовать типовые параллельные методы выполнения макроопераций в качестве конструктивных элементов при разработке параллельных способов решения более сложных вычислительных задач. Процесс введения макроопераций осуществляется поэтапно с последовательно возрастающим уровнем детализации используемых операций.

Следующим этапом построения параллельного алгоритма является определение информационных потоков. Организация взаимодействия процессов, приводящая к формированию вполне определенных **схем коммуникации** (кольцо, тор, гиперкуб) – есть структурное взаимодействие. Неструктурное взаимодействие возникает, когда схема выполняемых операций передач данных имеет граф "нетипового" вида. Последний этап разработки параллельного алгоритма есть распределение вычислительной нагрузки по процессорам параллельной компьютерной системы и решение задачи балансировки.

Таким образом, в рамках данной исследовательской работы рассматриваются следующие этапы разработки параллельных

алгоритмов с использованием поэтапной декомпозиционной технологии:

1) анализ задачи и выявление ее потенциального параллелизма,

2) выбор модели программирования и схемы распараллеливания;

3) разделение процесса вычислений на части, которые могут быть выполнены одновременно и определение необходимых информационных взаимодействий для параллельных процессов обработки данных;

5) отображение параллельной схемы решения задачи на архитектуру компьютерной вычислительной системы.

Выбор декомпозиционной методики обусловлен преимуществами этого подхода, а именно: взаимозависимостью и итеративностью этапов построения параллельного алгоритма, возможностью обеспечить необходимый уровень масштабируемости вычислений за счет варьирования детальности декомпозиции.

1.4. Высокопроизводительные вычисления и динамические характеристики параллельных алгоритмов

При разработке параллельных методов решения задач принципиальным моментом является определение качества использования параллелизма. На сегодняшний день огромное количество исследований [10-14, 113-118] посвящено анализу существующих и введению новых динамических характеристик, определяющих эффективность параллельного алгоритма в сочетании с вычислительной системой, на которой он реализован.

Общепринятые и наиболее используемые в теории параллельных вычислений показатели:

1) T_1 – время, необходимое для решения задачи заданного размера на одном процессоре с помощью наилучшего последовательного алгоритма;

2) $T_{p,comp}$ – время решения задачи заданного размера с использованием параллельного алгоритма на компьютере из p процессоров без учета обменных операций (время реализации вычислений);

3) $T_{p,comm}$ – коммуникационная трудоемкость, сложность алгоритма или время выполнения межпроцессорных операций обмена;

4) T_p – общее время реализации параллельного алгоритма на параллельной архитектуре: $T_p = T_{p,comp} + T_{p,comm}$;

5) Z – доля времени обменных операций к общему времени выполнения параллельного алгоритма: $Z = T_{p,comm} / T_p$;

6) Dop – степень параллелизма или максимальное количество процессоров, используемых при выполнении параллельного алгоритма;

7) S_{pot}, E_{pot} – коэффициенты потенциального ускорения и эффективности (без учета операций обмена);

8) S, E – коэффициенты реального ускорения и эффективности параллельного алгоритма (с учетом обменных операций).

Коэффициент ускорения – динамическая характеристика, показывающая, во сколько раз быстрее параллельный алгоритм по сравнению с последовательным вариантом решения задачи [3, 11, 97]. Он определяется как отношение времени решения задачи на однопроцессорной вычислительной системе ко времени выполнения параллельного алгоритма. Различают потенциальный и реальный параллелизм, а, следовательно, и соответствующие коэффициенты ускорения. Коэффициент потенциального ускорения учитывает только внутренний, скрытый параллелизм метода решения задачи без учета операций обмена и других накладных расходов:

$$S_{pot} = T_1 / T_{p,comp},\ S = T_1 / T_p. \qquad (1.11)$$

Следующей важной характеристикой параллельного алгоритма является **коэффициент эффективности** [11,97], определяющий среднюю долю времени выполнения алгоритма, в течение которого процессоры реально используются для решения задачи, то есть качество загрузки оборудования:

$$E = T_1 / p \cdot T_p = S / p. \qquad (1.12)$$

В идеальном случае достигается линейное ускорение и единичная эффективность: $1 \le S \le p$, $1/p \le E \le 1$, при определенных обстоятельствах может наблюдаться явление **сверхлинейного ускорения**: $S > p$.

Разработка параллельных приложений в качестве своей цели может иметь не только уменьшение времени выполнения, но и обеспечение возможности решения задач большей размерности.

Способность параллельного алгоритма эффективно использовать процессоры при увеличении сложности расчетов является важной характеристикой параллельных вычислений и называется мерой масштабируемости [10-14, 97]. Параллельный алгоритм является масштабируемым, если при росте числа процессоров он обеспечивает увеличение ускорения при сохранении постоянного уровня эффективности использования процессоров.

Существует несколько подходов для количественной оценки свойств масштабируемости алгоритма. Наиболее применяемой является метрика, предложенная в [11-14] и основанная на введении **функции равной эффективности или изоэффективности**. Определяется новая динамическая характеристика – **накладные расходы на параллелизм**:

$$T_o(m,p) = p \cdot T_p - T_1. \tag{1.13}$$

Величина общих накладных расходов включает суммарные затраты всех процессоров параллельной системы: на реализацию обменов, последовательную часть распараллеленного алгоритма, непроизводительные расходы на синхронизацию и время простоя из-за несбалансированности загрузки процессоров. Используя введенное обозначение, получают новые выражения для коэффициентов ускорения и эффективности:

$$S = \frac{T_1}{T_p} = \frac{pT_1}{T_1 + T_o}, E = \frac{S}{p} = \frac{T_1}{T_1 + T_o} = \frac{1}{1 + T_o / T_1}. \tag{1.14}$$

Пусть $E = const$ задает необходимый уровень эффективности выполняемых вычислений, тогда:

$$T_o / T_1 = (1 - E)/E, \tag{1.15}$$

$$T_1 = E/(1 - E) \cdot T_0 = K(pT_p - T_1), \tag{1.16}$$

где $K = E/(1 - E)$ – есть коэффициент, зависящий только от значения показателя эффективности. Порождаемую последним соотношением зависимость $m = f_E(p)$ между сложностью решаемой задачи и числом процессоров называют **функцией изоэффективности**. Выражение (1.16) – центральное соотношение изоэффективного анализа. Функция f_E определяет, как надо увеличивать размер задачи при увеличении числа процессоров для поддержания постоянной эффективности. Изоэффективная функция некоторых параллельных систем есть полином от размерности процессорного поля: $O(p^x)$, где $x \geq 1$. При

$x = 1$ получаем **линейный масштабируемый параллельный алгоритм**. Малая степень p в функции изоэффективности свидетельствует о высокой масштабируемости. Если же имеется экспоненциальная (или любая другая быстро растущая функция) зависимость, то речь идет о слабо масштабируемых системах. Близким к линейно масштабируемому алгоритму считается **квазилинейный алгоритм** с функцией изоэффективности порядка $O(\log_x p)$ [11-14]. Таким образом, построение функции изоэффективности – это попытка соединить в едином аналитическом выражении характеристики задачи, параллельной архитектуры и оценить степень их влияния на качество параллельного алгоритма в целом.

При анализе эффективности параллельных алгоритмов необходимо учитывать временные машинно-зависимые параметры, влияющие на скорость выполнения параллельных вычислений. Таким параметром является время выполнения операции с плавающей точкой. Количество операций с плавающей точкой в секунду (Floating Point Operations Per Second – *FLOPS*) есть характеристика производительности процессора. Время выполнения операции с плавающей точкой будем обозначать:

$$t_{op} = 1 / FLOPS \,. \qquad (1.17)$$

При подсчете динамических характеристик алгоритмов считается, что $t_{ad} = t_{mul} = t_{op}$ – любая арифметическая операция с плавающей точкой выполняется за одно и тоже время независимо от вида операции. Это предположение справедливо для большинства современных компьютеров RISC архитектуры.

1.5. Анализ параллельных методов решения динамических задач для систем обыкновенных дифференциальных уравнений

Методы интегрирования динамических задач с сосредоточенными параметрами, описываемые системами дифференциальных уравнений,

$$\overline{y}' = \overline{f}(x, \overline{y}), \; \overline{y}(x_0) = \overline{y}_0, \qquad (1.18)$$

несмотря на наличие огромных и всесторонних знаний в этой области, продолжают оставаться источником активных исследований, особенно

в таком новом научном направлении, как параллельные вычисления. Распараллеливание известных и построение новых параллельных методов решения задачи Коши имеет своей целью ускорение вычислений при интегрировании СОДУ. Это особенно важно для научно-технических задач больших размерностей, сложных правых частей и для быстрого моделирования динамических процессов в реальном времени.

Параллелизм, используемый при решении СОДУ, можно классифицировать двумя способами:

а) системный параллелизм (параллелизм в пространстве);

б) параллелизм метода (параллелизм во времени).

Параллелизм за счет метода означает, что благодаря специальной структуре метода, некоторые значения функции могут быть вычислены параллельно внутри одного шага интегрирования.

Одношаговый *s*-стадийный метод типа Рунге-Кутты имеет вид:

$$\begin{cases} \overline{y}_{n+1} = \overline{y}_n + h_n \cdot \sum_{i=1}^{s} b_i \cdot \overline{k}_i; \\ \overline{k}_i = \overline{f}(x_n + c_i \cdot h_n; \overline{y}_n + h_n \cdot \sum_{j=1}^{s} a_{ij} \cdot \overline{k}_j); i = 1,...,s; \end{cases} \tag{1.19}$$

где s – число стадий метода, а $A = (a_{ij})_{i,j=1}^{s}$, $b = (b_1,...,b_s)^T$, $c = (c_1,...,c_s)^T$ – вещественные коэффициенты, определяющие уникальный вариант метода (выбираются из соображений точности). Параллельные методы Рунге-Кутты, как и последовательные, можно разбить на два больших класса: **явные (ЯМРК)** и **неявные (НМРК)** [119-120]. Классы отличаются друг от друга способом вычисления стадийных коэффициентов $\overline{k}_i = (k_{i1}, k_{i2},..., k_{im}), i = \overline{1,s}$, характеристиками устойчивости, а, следовательно, и областью применения.

Наиболее известными являются работы [119-126], основанные на реструктуризации явных методов Рунге-Кутты. При малом потенциальном параллелизме метода подходы к распараллеливанию ЯМРК эксплуатируют в основном системный параллелизм. В этом смысле интерес представляют так называемые ***P*-оптимальные методы** [119], реализующие параллелизм во времени. Если некоторые элементы **матрицы Батчера** равны нулю, то появляется возможность

вычислять соответствующие стадийные коэффициенты параллельно. Для этого строится ориентированный граф, в котором каждая дуга, проведенная из вершины i в вершину j, устанавливает ненулевой коэффициент матрицы коэффициентов ЯМРК.

Широкого распространения распараллеливание на основе P-оптимальности не получило, поскольку практически все распространенные явные методы Рунге-Кутты не обладают такими свойствами. С другой стороны, даже наличие таких свойств обеспечивает незначительную степень параллелизма. Явные одношаговые методы имеют ограниченную область применения из-за не способности решать жесткие задачи. В этой связи значительный интерес представляют **неявные схемы**, которые, несмотря на большую вычислительную сложность, не имеют альтернативы среди одношаговых методов при интегрировании **жестких уравнений**. В [123-128] предложены **устойчивые численные методы**, использующие неявные схемы.

Для вычисления решения каждый из этих методов требует разрешения систем нелинейных алгебраических уравнений. Описанный подход к решению задачи обладает высокой эффективностью, так как позволяет строить численные методы высокого порядка сходимости. Существенным недостатком этих методов являются значительные затраты машинного времени, связанные, во-первых, с увеличением размерности задачи и с ростом числа итераций, необходимых для обеспечения сходимости. В работах [129-132] предложены **параллельные диагонально и однократно-диагональные неявные методы**, существенно сокращающие процесс решения СНАУ на основе **модифицированного метода Ньютона**. Однако эти методы обладают худшими характеристиками устойчивости, и определяют решения меньшего порядка точности.

До сих пор речь шла о реструктуризированных методах. В материалах [133-139] предложены так называемые **блочные методы на основе неявных схем**, изначально ориентированные на выполнение в параллельных системах. Работы [135-136] описывают способы построения одношаговых k –точечных методов, в которых для блока из k точек новые k значений функции могут вычисляться одновременно. Авторами [136-138] получены расчетные формулы для m-шаговых k-

точечных блочных методов, определен порядок их точности. В [139] рассматриваются общие линейные многошаговые блочные методы, доказана сходимость методов к точному решению, определены условия **устойчивости по Далквисту**.

Одним из главных вопросов, возникающих при численном решении СОДУ, является проблема оценки погрешности приближенного решения. **Апостериорная оценка локальной погрешности**, получаемая на каждом шаге вычислений, позволяет автоматически выбирать шаг интегрирования, обеспечивающий заданную точность численного решения.

Известными и общепринятыми методами оценки пошаговой апостериорной погрешности решения СОДУ являются [140-141]: дублирование шага по правилу Рунге, технология локальной экстраполяции Ричардсона, вложенные методы или методы вложенных форм. Наиболее простой способ определения локальной погрешности – **дублирование шага по правилу Рунге**. Для точного решения при переходе из точки x_n к точке $x_n + 2h, \bar{y}(x_n + 2h)$ на основе одной и той же формулы порядка r получают две аппроксимации: $\bar{y}_{n+1}^{(1)}$ (один шаг длиной $2h$) и $\bar{y}_{n+1}^{(2)}$ (два шага h). Разность между этими значениями используется в качестве оценки локальной погрешности интегрирования, в качестве решения принимается аппроксимация, полученная с половинным шагом, как наиболее точная. Процесс уточнения решения с половинным шагом $\bar{y}_{n+1}^{(2)}$ повышает точность решения на единицу и носит название локальной экстраполяции [119].

Метод локальной экстраполяции Ричардсона является обобщением технологии удвоения шага по правилу Рунге [141]. Идея метода заключается в многократном измельчении шага интегрирования, и также в многократном применении процесса вычисления, названного выше локальной экстраполяцией. Решение задачи Коши рассматривается при переходе из точки x_n в точку $x_{n+1} = x_n + H$, где H – базовая длина шага (рис. 1.7). Выбирается ряд натуральных чисел $P_i = \{n_1, n_2, ..., n_k, ..\}$ такой, что: $n_1 < n_2 < ... < n_k < ...$ и, соответственно, последовательность шагов: $h_1 > h_2 > ... > h_{k-1} > h_k > ...$, где $h_i = H / n_i$. Задается опорный численный метод порядка r_0 и, выполняя

n_i шагов интегрирования длиной h_i, вычисляются приближенные решения исходной задачи в точке x_{n+1}:

$$T_{i,1} = \overline{y}_{h_i}(x_n + H) = \overline{y}^{(i)}_{n+1},\ i = \overline{1,k}. \tag{1.20}$$

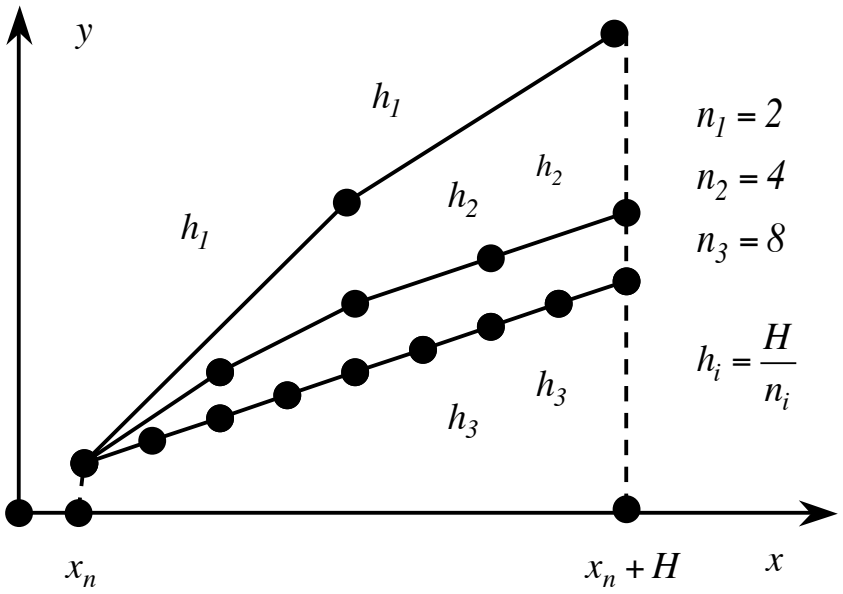


Рисунок 1.7 – Технология локальной экстраполяции Ричардсона

Выполнив вычисления для ряда последовательных значений *i*, по рекуррентному соотношению определяют $T_{i,j+1}$ – экстраполированные значения для произвольных *i,j*:

$$T_{i,j+1} := T_{i,j} + \frac{T_{i,j} - T_{i-1,j}}{(n_i / n_{i-j})^b - 1}. \tag{1.21}$$

Вычисления по (1.21) – многократно повторенный процесс локальной полиномиальной экстраполяции по **схеме Эйткена-Невилла** [119]. Величина *b* равна 1 в общем случае, для симметричных опорных методов *b* равно двум. T_{ij} – приближенное решение задачи Коши, полученное численным методом порядка $r_0 + (j-1)\cdot b$ с шагом h_i. Достоинство этого метода состоит в том, что он дает таблицу результатов вычислений (табл. 1.2), которые позволяют оценить локальную погрешность.

Таблица 1.2

Экстраполяционная таблица

r_0	$r_0 + b$	$r_0 + 2b$	…	$r_0 + (k-2)b$	$r_0 + (k-1)b$
T_{11}			…		
T_{21}	T_{22}		…		
…	…	...	…	T_{k-1k-1}	
T_{k1}	T_{k2}	T_{k3}	…	T_{k-1k}	T_{kk}

Альтернативным способом определения локальной апостериорной погрешности являются **вложенные методы Рунге-Кутты (ВМРК) или методы вложенных форм**. Этот способ основан на использовании двух приближенных значений решения в одной точке, но в отличие от правила Рунге приближения вычисляются не по одной, а по двум формулам различных порядков точности r и $\hat{r}$ с одним и тем же шагом [140].

$$\begin{cases} \bar{y}_{n+1} = \bar{y}_n + h_n \cdot \sum_{l=1}^{s} b_l \cdot \bar{k}_l, \\ \hat{\bar{y}}_{n+1} = \bar{y}_n + h_n \cdot \sum_{l=1}^{s'} b'_l \cdot \bar{k}_l, \; d = \left\| (\hat{\bar{y}}_{n+1} - \bar{y}_{n+1}) \right\|. \end{cases} \tag{1.22}$$

Для определения локальной погрешности менее точного результата и управления величиной шага интегрирования используется величина d. Вложенный метод Рунге-Кутты $r(\hat{r})$ – это схема, в которой метод $\hat{r}$ -го порядка (“оценщика погрешности”) получается как побочный продукт метода r-го порядка (рис. 1.8).

Порядок аппроксимаций: $\bar{y}_{n+1}$ и $\hat{\bar{y}}_{n+1}$ обычно отличаются на 1, т.е. $r = \hat{r} + 1$ или $r = \hat{r} - 1$. Оба приближения используют практически одни и те же значения стадийных коэффициентов, но комбинируют их по-разному. Вложенные методы Рунге-Кутты – это наименее трудоемкий из известных способов определения локальной апостериорной погрешности решения в последовательной реализации.

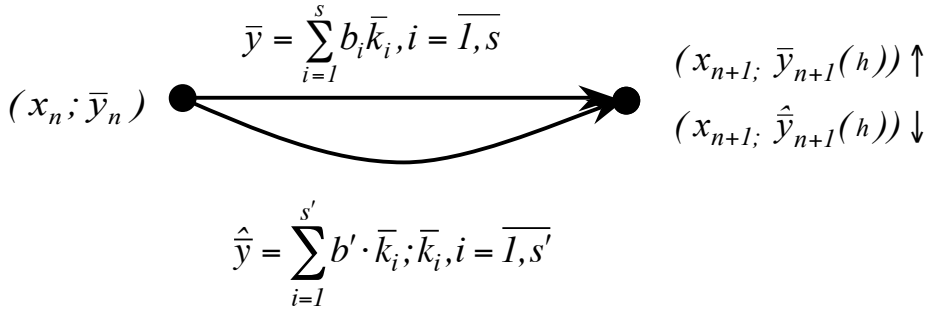


Рисунок 1.8 – Схема вычислений для вложенных методов

Обобщая известные теоретические исследования и вычислительные эксперименты [119-139] необходимо отметить, что, несмотря на то, что методы решения систем обыкновенных дифференциальных уравнений используются в вычислительной практике достаточно давно, разработка новых эффективных параллельных вычислительных схем для решения конкретных динамических задач до сих пор остается актуальной проблемой.

1.6. Выводы

Проведенный обзор и анализ литературы свидетельствуют о том, что повышение эффективности параллельных компьютерных систем – одна из наиболее актуальных задач современного этапа развития вычислительной техники. Очевидно, что ее решение в ближайшем будущем в значительной степени будет определяться результатами, полученными в области алгоритмических и программных приложений. Поэтому целью данной работы является повышение эффективности функционирования многопроцессорных вычислительных систем за счет разработки новых параллельных методов решения динамических задач и их реализации в современных вычислительных структурах различных архитектур и топологий.

Для достижения поставленной цели решены следующие задачи:

– разработаны параллельные вычислительные методы решения нелинейной задачи Коши на основе явных одношаговых схем с встроенными способами оценки локальной погрешности вложенные методы, правило дублирования шага и локальная экстраполяция

Ричардсона, получены оценки их вычислительной сложности с учетом различных способов разбиения данных по процессорам: равномерный, пропорциональный и комбинационный;

– разработаны новые параллельные методы, реализующие способы оценки шаговой погрешности для блочных одношаговых методов интегрирования начальной задачи Коши для ОДУ;

– выполнено обобщение разработанных параллельных блочных неявных одношаговых методов, реализующих способы оценки шаговой погрешности при интегрировании начальной задачи Коши для систем обыкновенных дифференциальных уравнений;

– проведено сравнение параллельных блочных методов решения задачи Коши с полностью неявными одношаговыми методами типа Рунге-Кутты;

– разработаны специальные параллельные экспоненциальные методы решения линейной задачи Коши со встроенными способами оценки локальной апостериорной погрешности, более эффективные, чем стандартные алгоритмы решения той же задачи;

– разработан эффективный рекурсивно-систолический метод умножения плотнозаполненных матриц, ускоряющий выполнение этой ресурсоемкой операции экспоненциального метода.

– решены задачи отображения предложенных методов на современные параллельные вычислительные структуры SIMD-, MIMD- и CLUSTER-архитектур с различными топологиями соединения процессоров: кольцо, 2D-тор и гиперкуб, произведена оценка эффективности отображения ;

– определены динамические характеристики качества параллелизма для разработанных методов: коэффициент ускорения, коэффициент эффективности и масштабируемость на основе функции изоэффективности;

– для каждого параллельного алгоритма разработаны рекомендации по выбору конкретного класса параллельной вычислительной системы, обоснование выбора выполнено посредством определения динамических характеристик параллелизма, учитывающих как параметры метода, так и исходной системы обыкновенных дифференциальных уравнений.

ГЛАВА 2

ПАРАЛЛЕЛЬНЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНОЙ ЗАДАЧИ КОШИ НА ОСНОВЕ ЯВНЫХ ОДНОШАГОВЫХ ЧИСЛЕННЫХ СХЕМ

Современные исследования и вычислительные эксперименты в области параллельных вычислений показывают [5], что практически все новые параллельные методы, даже те из них, которые эффективны в теоретическом отношении, на практике не конкурентно способны. Поэтому на текущий момент единственно надёжным источником создания параллельных методов является подходящая реструктуризация проверенных временем последовательных алгоритмов и математических описаний. Формально реструктуризация сводится к математически эквивалентным заменам в записях всех или части формульных выражений с целью явно указать обнаруженный в алгоритмах потенциальный, скрытый параллелизм.

В данном разделе рассматриваются параллельные алгоритмы явных методов Рунге-Кутты (ЯМРК) с встроенными способами оценки шаговой апостериорной погрешности для численного решения систем обыкновенных дифференциальных уравнений общего вида. Предлагаемые алгоритмы ориентированы на использование в многопроцессорных вычислительных системах SIMD, MIMD, кластерной архитектуры с различными топологиями соединения вычислительных устройств, будь то компьютеры, процессоры или процессорные элементы. Набор устройств однороден, известен до начала вычислений и не меняется в процессе счета, при этом каждый вычислитель может выполнить любую арифметическую операцию за один такт (флоп), временные затраты, связанные с обращением к запоминающему устройству, не анализируются.

Явные формулы типа Рунге-Кутты находятся среди старейших и хорошо изученных схем численного анализа [2]. Однако, несмотря на наличие огромных и всесторонних знаний в этой области, ЯМРК продолжают быть источником активных исследований, особенно в таком новом научном направлении как параллельные вычисления.

Потенциально явные методы Рунге-Кутты содержат два вида параллелизма:

– **параллелизм метода** (параллелизм во времени);

– **параллелизм системы** (параллелизм в пространстве).

Системный параллелизм реализуется через распределённые вычисления различных компонент системы, т.е. отдельных компонент шаговых векторов и векторов решений. Параллелизм метода связан с выделением независимых субвычислений внутри одного шага метода (например, параллельное умножение матрицы на вектор) и, как правило, значительно меньше системного. Распараллеливание явных одношаговых разностных методов, к которым относятся формулы Рунге-Кутты, концентрируется на выполнении одного шага интегрирования.

Пусть численно решается задача Коши для системы обыкновенных дифференциальных уравнений (СОДУ) первого порядка с известными начальными условиями:

$$\begin{cases} \dfrac{dy_1(x)}{dx} = f_1(x; y_1, y_2, ..., y_m), \quad y_1(x_0) = y_{10}, \\ \dfrac{dy_2(x)}{dx} = f_2(x; y_1, y_2, ..., y_m), \quad y_2(x_0) = y_{20}, \\ \qquad\qquad ... \\ \dfrac{dy_m(x)}{dx} = f_m(x; y_1, y_2, ..., y_m), \quad y_m(x_0) = y_{m0}. \end{cases} \tag{2.1}$$

Явный s-стадийный одношаговый метод Рунге-Кутты может быть определён следующим образом:

$$\begin{cases} \overline{y}_{n+1} = \overline{y}_n + h \cdot \sum\limits_{i=1}^{s} b_i \overline{k}_i, \\ \overline{k}_i = F(x_n + c_i h; \overline{g}_i), \\ \overline{g}_i = \overline{y}_n + h \cdot \sum\limits_{j=1}^{i-1} a_{ij} \overline{k}_j, \ i = 1, ..., s. \end{cases} \tag{2.2}$$

Рассмотрим различные вычислительные схемы параллельных алгоритмов на базе ЯМРК для СОДУ общего типа с учётом встроенных методов оценки локальной апостериорной погрешности.

2.1. Распараллеливание явных одношаговых численных схем с дублированием шага по правилу Рунге

При распараллеливании алгоритма решения нелинейной задачи Коши на основе ЯМРК с использованием правила Рунге, воспользуемся иерархической декомпозиционной методикой. Первоначально вычислительная задача разбивается на подзадачи, анализируются информационные взаимосвязи между ними, биективно к множеству подзадач определяется множество макроопераций, оценивается эффективность потенциального крупно или средне блочного параллелизма. Затем методика применяется для распараллеливания каждой из макроопераций, то есть используется мелкозернистый параллелизм. Для разработки параллельного алгоритма и проверки корректности его построения на каждом из уровней используется математический аппарат графов влияния [3-5].

Один шаг интегрирования СОДУ на основе ЯМРК с встроенным способом оценки локальной погрешности по правилу Рунге имеет вид:

$$\begin{cases} \bar{y}_{n+1}^{(1)} = \bar{y}_n + 2h \cdot \sum\limits_{i=1}^{s} b_i \cdot \bar{k}_i^{(1)}(2h), \quad \bar{k}_i^{(1)} = F\left(x_n + c_i \cdot 2h, \bar{y}_n + 2h \sum\limits_{j=1}^{i-1} a_{ij} \bar{k}_j^{(1)} \right), \\ \bar{y}_{n+\frac{1}{2}} = \bar{y}_n + h \cdot \sum\limits_{i=1}^{s} b_i \cdot \bar{k}_i(h), \quad \bar{k}_i = F\left(x_n + c_i h, \bar{y}_n + h \sum\limits_{j=1}^{i-1} a_{ij} \bar{k}_j \right), \ i = \overline{1,s}, \\ \bar{y}_{n+1}^{(2)} = \bar{y}_{n+\frac{1}{2}} + h \cdot \sum\limits_{i=1}^{s} b_i \cdot \bar{k}_i^{(2)}(h), \quad \bar{k}_i^{(2)} = F\left(x_{n+\frac{1}{2}} + c_i h, \bar{y}_{n+\frac{1}{2}} + h \sum\limits_{j=1}^{i-1} a_{ij} \bar{k}_j^{(2)} \right). \end{cases} \qquad (2.3)$$

Последовательный алгоритм, описываемый вычислительной схемой (2.3), можно представить, как решение трёх подзадач:

1) вычисление приближенного решения $\bar{y}_{n+1}^{(1)}(2h)$ в точке x_{n+1} с шагом $2h$;

2) вычисление приближенного решения в промежуточной точке $x_{n+\frac{1}{2}} = x_n + h$ с шагом h: $\bar{y}_{n+\frac{1}{2}}(h)$;

3) вычисление приближенного решения в точке x_{n+1} через промежуточную точку с шагом h: $\bar{y}_{n+1}^{(2)}(h)$.

Все эти три задачи являются применением явной одношаговой схемы, что позволяет ввести в качестве основной макрооперации на этом уровне – однократное вычисление аппроксимации точного решения в некоторой произвольной точке сетки с заданным шагом интегрирования. В методе функциональной декомпозиции сначала сегментируется вычислительный алгоритм, а затем уже под него подгоняется схема декомпозиции данных.

Рассмотрим три различные макрооперационные вычислительные схемы исследуемого алгоритма (рис. 2.1).

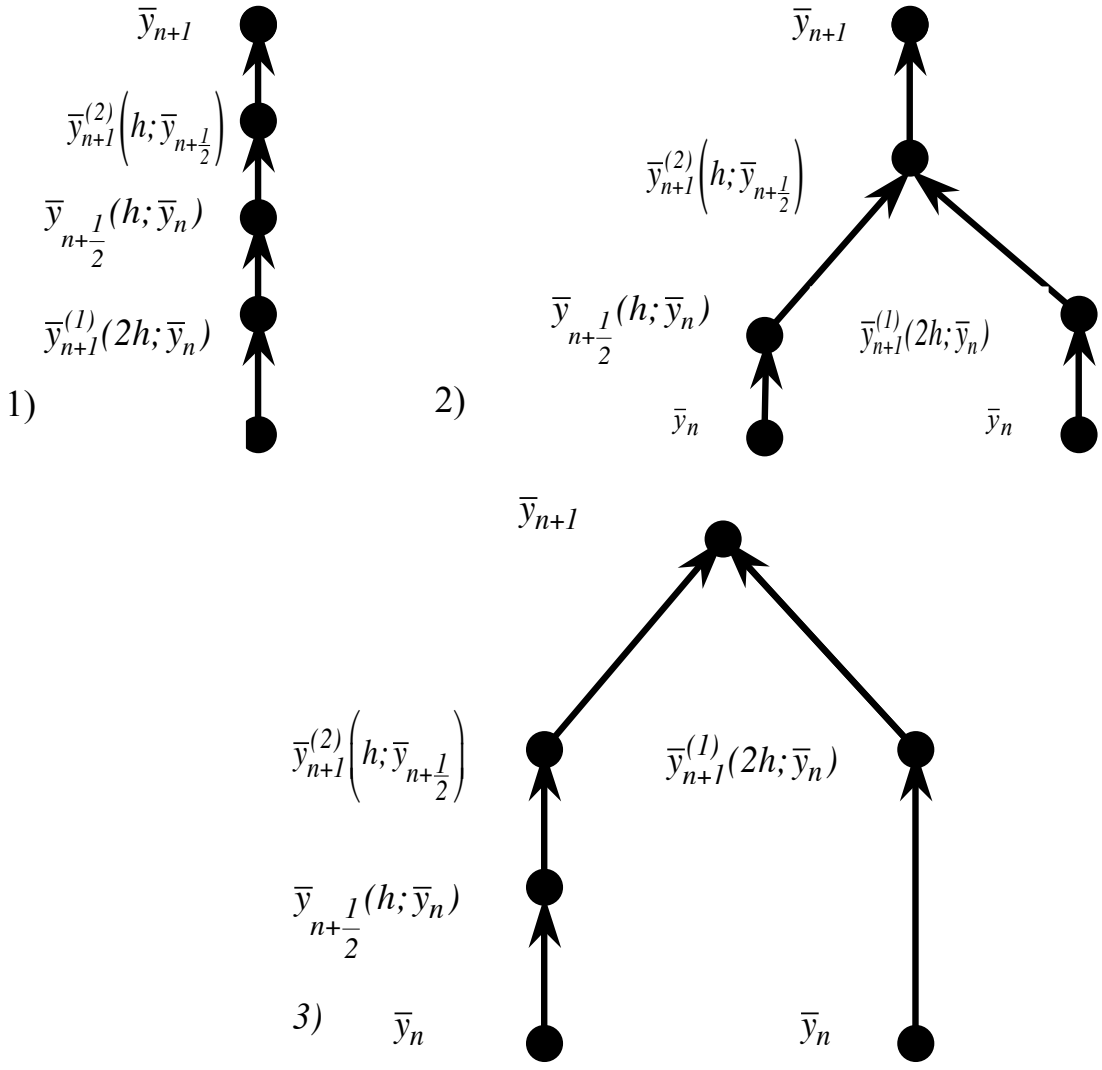


Рисунок 2.1 – Графы влияния макрооперационных вычислительных схем параллельного ЯМРК в сочетании с правилом Рунге,
1) схема № 1, 2) схема № 2 (равномерное разбиение),
3) схема № 3 (пропорциональное разбиение)

Первый вариант макрооперационной схемы конструируемого параллельного алгоритма представлен на рисунке 2.1. Все три подзадачи выполняются последовательно, данные между процессорами распределены равномерно, $Dop = m$.

Второй вариант макрооперационной схемы предполагает, что первая и вторая подзадачи выполняются параллельно, причём все множество компьютеров разделено на две равные части ($p_1 = \lceil p/2 \rceil$) для выполнения каждой из подзадач, а затем третья подзадача решается на всем процессорном поле. Исходные данные дублированы и равномерно распределены между двумя равными группами процессоров для решения первых двух подзадач, а затем перераспределены равномерно на все множество процессоров для решения третьей подзадачи. Для выполнения следующего шага интегрирования исходные данные должны быть снова перераспределены равномерно в каждой из двух групп процессоров.

Третий вариант предполагает параллельное выполнение первой подзадачи на меньшем числе процессоров, второй и третьей – на большем числе процессоров. Множество процессоров делится на две пропорциональные части: $p_1 = 2p_2; p_1 + p_2 = p$. На этом уровне детализации нет никаких оснований предпочесть одну из трёх вычислительных схем алгоритма. Последовательность выполнения трёх макроопераций в первой схеме компенсируется равномерной загрузкой и низкой интенсивностью взаимодействия процессоров. С другой стороны, вторая и третья вычислительные схемы алгоритма имеют максимальную степень параллелизма равную – $Dop(t) = p_{max} = const = 2m$, в то время как первая схема – $Dop(t) = m$, то есть в два раза меньше.

Перейдём к исследованию параллелизма на уровне отдельных операций, т.е. к мелкозернистому параллелизму. Параллельное решение СОДУ на базе ЯМРК концентрируется на параллельном выполнении одного шага интегрирования. Вычисление любой аппроксимации решения по схеме (2.3) состоит в вычислении множества шаговых коэффициентов $\bar{k}_i = (k_{i1}, k_{i2}, ..., k_{im})$, $i = \overline{1,s}$ и, собственно, приближенного решения $\bar{y}_{n+1}(h)$. Для явных схем вычисление шаговых коэффициентов есть сугубо последовательный процесс.

Равенство нулю определённых коэффициентов $a_{ij}, i = \overline{2,s};\ j = \overline{1,s-1}$ в матрице Батчера определяет возможность одновременного их вычисления. Однако такое свойство ЯМРК даёт очень незначительную степень потенциального параллелизма (порядка единиц). Поэтому воспользуемся так называемым системным параллелизмом, когда вычисление одного вектора коэффициентов схемы $\bar{k}_i$ распределяется на несколько независимых процессоров.

Последовательный алгоритм одного шага интегрирования по явной вычислительной схеме может быть представлен, как решение s подзадач вычисления шаговых коэффициентов и одной подзадачи вычисления приближенного решения. Поэтому в качестве первой макрооперации этого уровня детализации выделяется макрооперация – вычисление i -того шагового коэффициента, а в качестве второй – вычисление численной аппроксимации решения в некоторой точке.

Введение макроопераций приводит к более компактному представлению информационного графа алгоритма, значительно упрощает проблему выбора эффективных способов распараллеливания вычислений, позволяет использовать типовые параллельные методы выполнения макроопераций в качестве конструктивных элементов при разработке параллельных способов решения более сложных вычислительных задач. Как показано, процесс введения макроопераций осуществляется поэтапно с последовательно возрастающим уровнем детализации используемых операций.

Для разработки параллельного алгоритма рассматриваемой макрооперации использовался математический аппарат **графов влияния** [5-12]. Задача распараллеливания в такой постановке сводится к отысканию максимального независимого множества вершин орграфа. Причём в таком орграфе алгоритма вершинам сопоставляются выполняемые операции, вершины соединяются дугами, тогда и только тогда, когда результат выполнения одной операции влияет на результат смежной.

На рисунке 2.2 приведен граф влияния первой макрооперации: вычисления вектора значений i – того стадийного коэффициента в случае, если размерность СОДУ совпадает с числом процессоров, $m = p$.

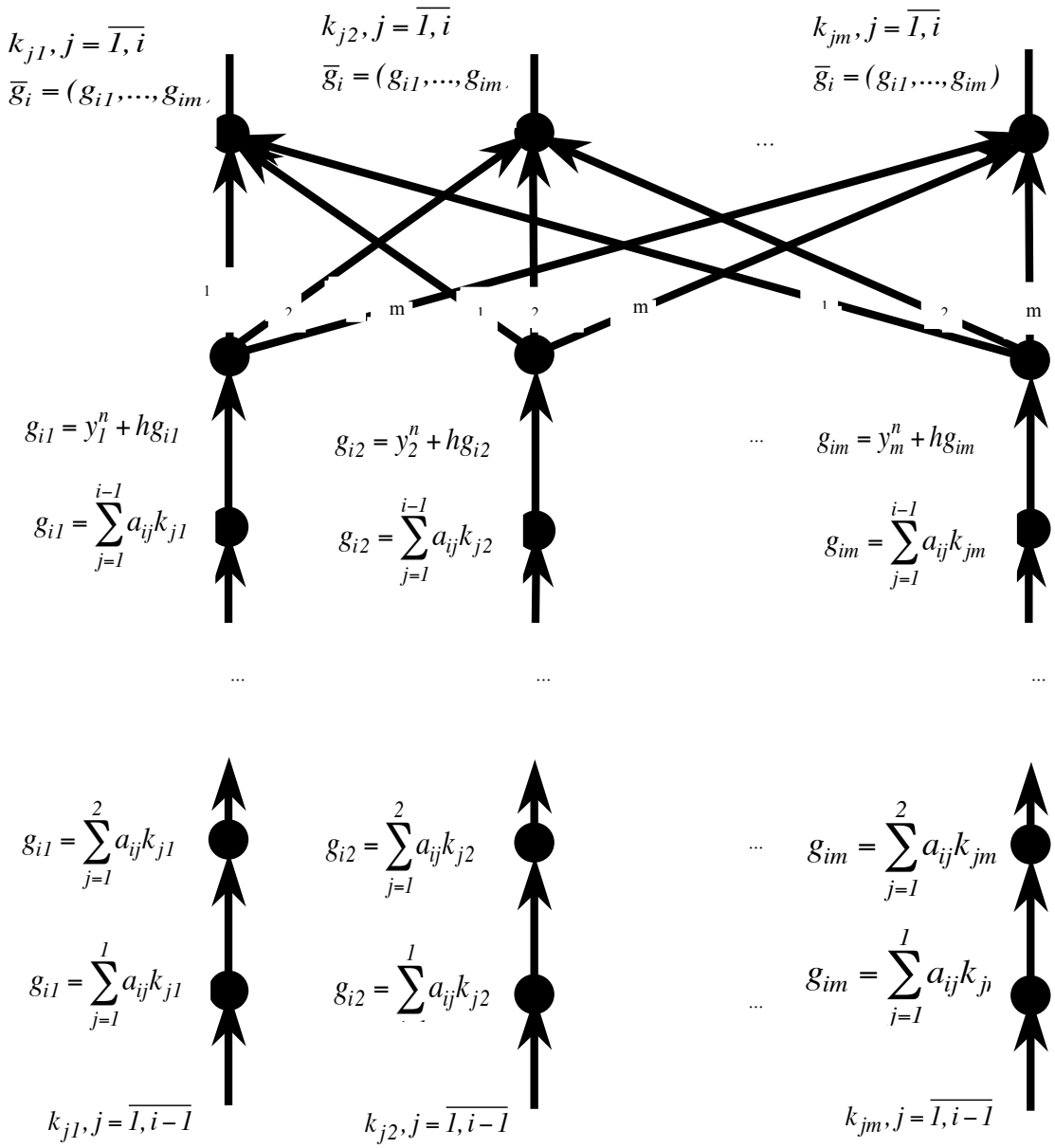


Рисунок 2.2 – Граф влияния для вычисления i – того стадийного коэффициента ЯМРК для СОДУ, $\overline{k}_i = (k_{i1}, k_{i2}, ..., k_{im})$, при $m = p$

Вторая макрооперация определяет вычисление приближенного решения на некотором шаге интегрирования и при имеющемся разбиении данных представлена графом влияния, изображённым на рис. 2.3. Обе макрооперации являются базовыми для всех явных схем и будут использованы в дальнейшем при рассмотрении альтернативных способов определения локальной погрешности.

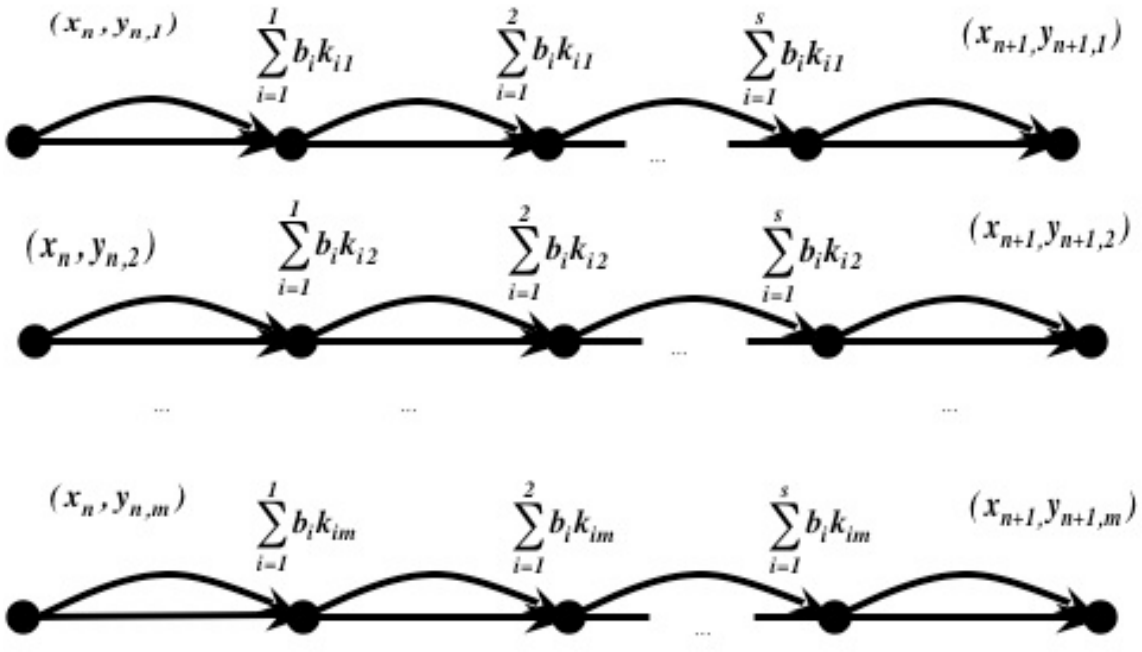


Рисунок 2.3 – Граф влияния второй макрооперации: вычисление численной аппроксимации решения в точке (x_{n+1}, y_{n+1})

Чтобы задача построения параллельных алгоритмов стала математически корректной, необходимо сделать предположения относительно свойств параллельной вычислительной системы. Применим **концепцию неограниченного параллелизма** [4]. В качестве идеализированной модели параллельной ВС используется мультикомпьютер с распределённой памятью и коммутационной схемой произвольной топологии.

Рисунок 2.4 представляет отображение первой макрооперационной схемы параллельного алгоритма ЯМРК с правилом Рунге на мультикомпьютер из p процессоров. Для СОДУ, состоящей из m уравнений, на одном временном шаге m_1 компонент векторов $\bar{g}_i$ и, соответственно, стадийных коэффициентов $\bar{k}_i$ могут быть вычислены параллельно.

Причём: 1) $m_1 = m$ при $m = p$; 2) $m_1 = \lceil m/p \rceil$ при $m > p$. Определение аппроксимаций решения и переход к следующему шагу не требуют дополнительных передач данных.

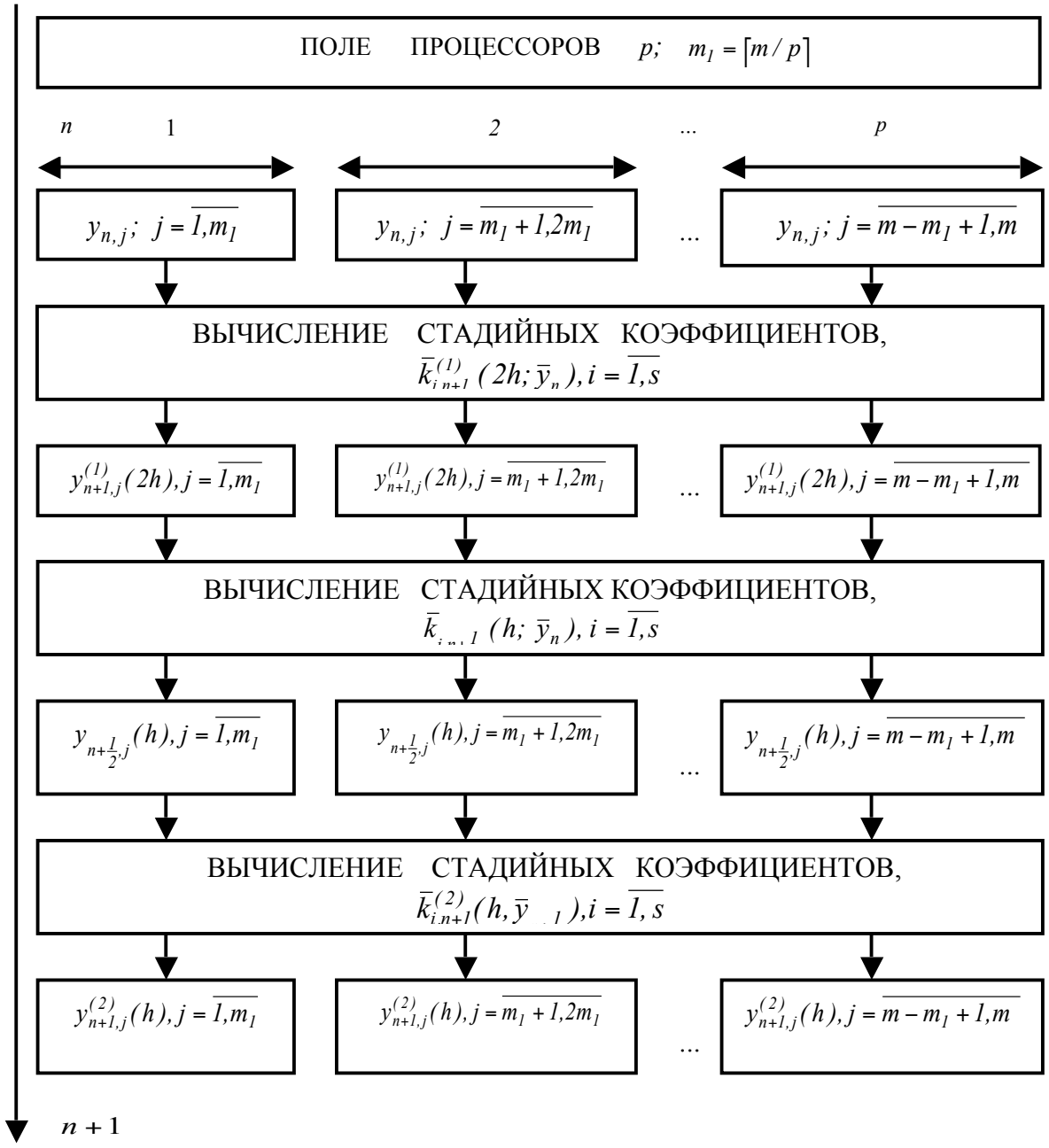


Рисунок 2.4 – Параллельный алгоритм ЯМРК + правило Рунге для численного решения нелинейной задачи Коши, вариант №1

Рисунок 2.5 описывает алгоритм решения той же задачи Коши для третьей макрооперационной схемы. Здесь разбиение процессоров произведено пропорционально объему вычислений. Базовая макрооперация описана графом влияния на рисунке 2.2. Первая группа процессоров вычисляет две аппроксимации решения: $\bar{y}_{n+\frac{1}{2}}$ и $\bar{y}^{(2)}_{n+1}(h)$ и

имеет размер процессорного поля в два раза больше, чем вторая группа. Вторая группа процессоров вычисляет одну аппроксимацию решения с удвоенным шагом: $\bar{y}^{(1)}_{n+1}(2h)$.

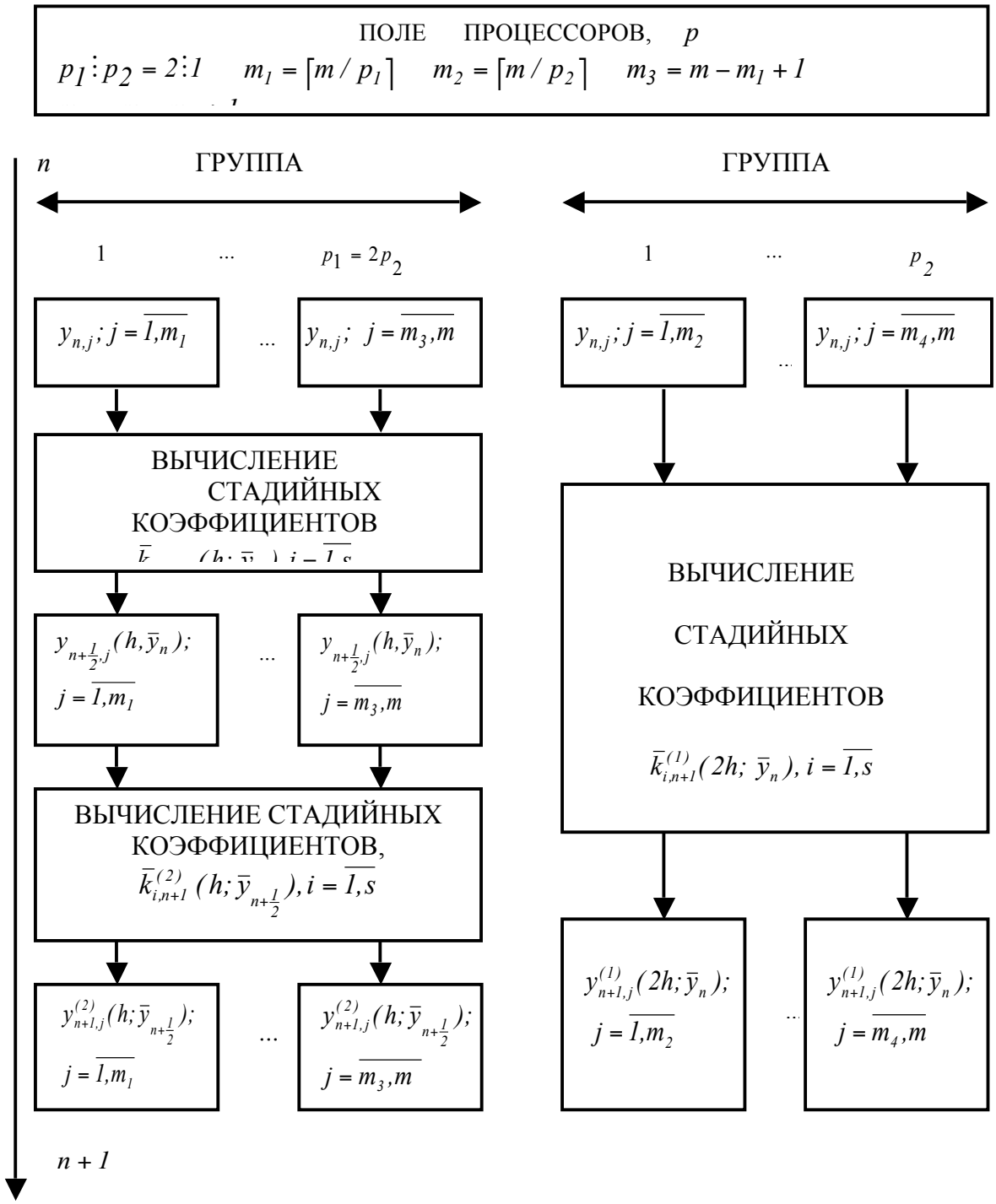


Рисунок 2.5 – Параллельный алгоритм ЯМРК+ правило Рунге , пропорциональное распределение данных по процессорам, вариант №3

Каждый процессор первой группы вычисляет $m_1 = \lceil m / p_1 \rceil$ компонент стадийных векторов и векторов решений, и каждый процессор второй группы вычисляет $m_2 = \lceil m / p_2 \rceil$ компонент тех же векторов. Таким образом, все вычисления производятся параллельно двумя группами с внутригрупповыми пересылками по типу “все-всем” и лишь для определения шага интегрирования потребуется множественная пересылка данных между группами. На данном этапе исследований очевидно, что вторая и третья схемы менее сбалансированы, более сложны в реализации и, по-видимому, менее эффективны. Однако для точного ответа на этот вопрос необходимо исследовать динамические характеристики потенциального и реального параллелизма для всех трех вычислительных схем.

Время выполнения последовательного алгоритма явного s - стадийного метода в применении к нелинейной задаче Коши с правилом Рунге состоит из времени вычисления трех аппроксимаций решения и определяется как:

$$T_1^1 = m(3s-1)T_F + 3(ms^2 + 2ms + 2s - 2)t_{op}, T_F = \sum_{i=1}^{m} T_{f_i}. \qquad (2.4)$$

Времена реализации расчетов для параллельных алгоритмов ЯМРК с правилом Рунге по вычислительным схемам равны:

1) вычислительная схема №1: $m \geq p$, $m_1 = \lceil m / p \rceil$:

$$T_{p,comp}^{11} = (3s-1)m_1 T_F + 3(m_1 s^2 + 2m_1 s + 2s - 2)t_{op}; \qquad (2.5)$$

2) вычислительная схема №2: $p_1 = \lceil p/2 \rceil$, $m_1 = \lceil m / p \rceil$, $m_2 = \lceil m / p_1 \rceil$:

$$T_{p,comp}^{12} = s(m_1 + m_2)T_F + [(m_1 + m_2)s^2 + 2(m_1 + m_2)s + 4s - 4)]t_{op}; \qquad 2.6)$$

3) вычислительная схема №3: $p_1 : p_2 = 2:1;\ m_1 = \lceil m / p_1 \rceil$;

4) $m_2 = \lceil m / p_2 \rceil : T_{p,comp}^{13} = m_2 s T_F + (m_2 s^2 + 2m_2 s + 4s - 4)t_{op}$. (2.7)

Величина T_F существенно зависит от рассматриваемой архитектуры параллельной ВС, для синхронной параллельности: $T_F = \sum_{i=1}^{m} T_{f_i}$; для асинхронной – $T_F = max_{i=1}^{m} T_{f_i}$. Для оценки потенциального параллелизма рассмотрим два альтернативных варианта: $T_F \gg t_{op}$ – время обращения к правой части СОДУ много

больше флопа; $T_F \sim t_{op}$ – время обращения к правой части соразмерно флопу:

$$1)\, T_F \gg t_{op} : T_1^1 \approx 3smT_F; \quad T_p^{11} \approx 3m_1 sT_F = 3\lceil m/p \rceil \cdot s \cdot T_F \quad \Rightarrow$$

$$\Rightarrow \quad S_{pot,1} = T_1^1 / T_p^{11} \approx p \Rightarrow E_{pot,1} \approx 1;$$

$$2)\, T_F \sim t_{op} : T_1^1 \approx (3s^2 m + 9sm)t_{op}; \; T_p^{11} \approx (3s^2 + 9s)m_1 t_{op} =$$

$$= (3s^2 + 9s)\lceil m/p \rceil \cdot t_{op} \Rightarrow \quad S_{pot,1} \approx p \Rightarrow E_{pot,1} \approx 1.$$

Аналогичные результаты для второй и третьей вычислительных схем позволяют сделать вывод, что характеристики потенциального параллелизма являются удовлетворительными: практически линейное ускорение, единичная эффективность и реально не зависят от выбора макрооперационной схемы алгоритма.

До сих пор нередко считается, что лучший вычислительный алгоритм тот, который имеет минимум арифметических операций, особенно трудоемких. Однако для параллельных вычислений важным, а иногда и определяющим, является учет сложности межпроцессорных связей. Поскольку для методов Рунге-Кутты информационное взаимодействие по типу коммуникаций является структурным, предоставляется возможность использовать единые коммуникационные примитивы для различных операций передачи данных в различных топологиях.

Определим время межпроцессорного обмена для трех вычислительных схем на основе введенных в подразделе 1.2 коммуникационных примитивов:

$$T_{p,comm}^{11} = 3sT_{all-to-all}(m_1; p), \tag{2.8}$$

$$T_{p,comm}^{12} = sT_{all-to-all}(m_2; p_1) + (s+1)T_{all-to-all}(m_1; p) + T_{all-to-all}(m_2; p), \tag{2.9}$$

$$T_{p,comm}^{13} = max[2sT_{all-to-all}(m_1, p_1); sT_{all-to-all}(m_2, p_2)] + T_{all-to-all}(m_2, p). \tag{2.10}$$

С учетом модели (1.1), вычислим время обмена для различных топологий и двух вычислительных схем, данные сведем в таблицу 2.1. Поскольку потенциальные характеристики вычислительных схем приближенно можно считать одинаковыми, произведем сравнение алгоритмов на основе коммуникационных составляющих.

Одним из основных параметров, влияющих на время межпроцессорного обмена, является топология соединения процессоров. По данным таблицы 2.1, для каждого из разработанных параллельных алгоритмов наиболее эффективной является топология гиперкуб. Худший вариант – линейка/кольцо, так как время обменов при таком соединении процессоров является определяющим и, как следствие, алгоритм имеет малые, а чаще всего незначительные оценки качества параллелизма.

Таблица 2.1

Время выполнения межпроцессорного обмена для трех вычислительных схем ЯМРК + правило Рунге и различных топологий

Вычислительная схема	Коэффициент при t_s	Коэффициент при t_w
Топология гиперкуб		
№1	$3s\log_2 p$	$3s\lceil m/p\rceil\cdot(p-1)$
№2	$(2s+2)\log_2 p$	$\lceil m/p\rceil\cdot(2sp-3s+3p-3)$
№3	$(2s+1)\log_2 p-2s\log_2 3+2s$	$\lceil m/p\rceil\cdot(2sp-3s+3p-3)$
Топология 2D-тор (замкнутая решетка)		
№1	$6s(\sqrt{p-1})$	$3s\lceil m/p\rceil\cdot(p-1)$
№2	$(2+\sqrt{2})s\sqrt{p}-4s+4\sqrt{p-4}$	$\lceil m/p\rceil\cdot(2sp-3s+3p-3)$
№3	$\left(4\sqrt{2/3}s+2\right)\sqrt{p-4s-2}$	$\lceil m/p\rceil\cdot(2sp-3s+3p-3)$
Топология 1D-тор (кольцо)		
№1	$3s(p-1)$	$3s\lceil m/p\rceil\cdot(p-1)$
№2	$1{,}5sp+2p-2s-2$	$\lceil m/p\rceil\cdot(2sp-3s+3p-3)$
№3	$(4/3)sp-2s+p-1$	$\lceil m/p\rceil\cdot(2sp-3s+3p-3)$

Вариант топологии 2D-тор в зависимости от значений коммуникационных параметров может приближаться по оценкам к гиперкубу, но не имеет перед ним преимуществ для предлагаемых вычислительных схем.

Более информативной величиной является отношение коммуникационных затрат к общим накладным временным затратам параллельного алгоритма. Эта величина некоторыми авторами [2] принимается в качестве оценки эффективности распараллеливания. Поскольку установлено, что топология гиперкуб наиболее оптимальна при организации обменов для рассматриваемых задач (рис. 2.7), оценим влияние машинно-зависимых коммуникационных констант на динамические параметры параллельных алгоритмов.

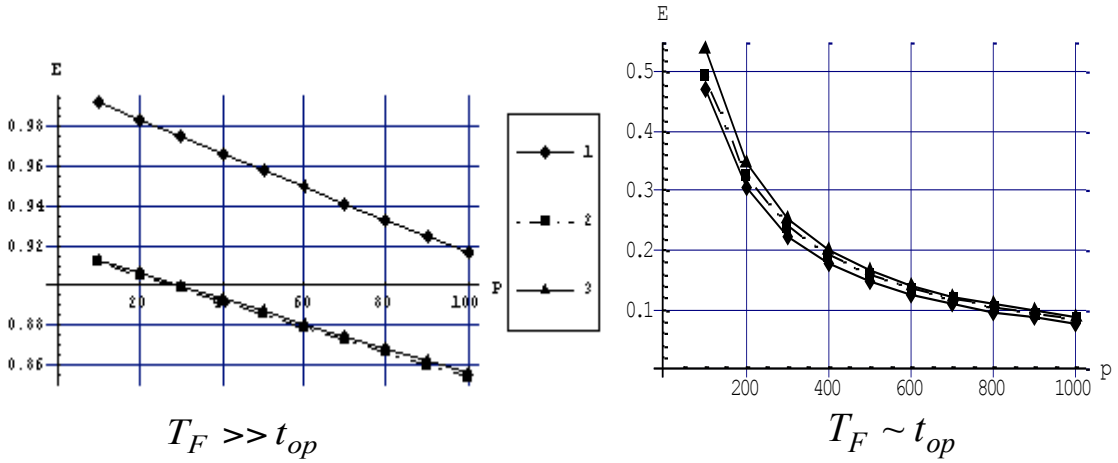
Рисунок 2.7 – Доля операций обмена в общем времени выполнения параллельного алгоритма №1 в различных топологиях

Временные характеристики обменных операций существенно зависят от используемого класса параллельных ВС. Согласно данным по соотношению между латентностью и временем передачи одного слова t_s / t_w выделим три различные группы мультикомпьютеров: кластерные, MIMD и SIMD-системы.

Кластерные параллельные системы с низкоскоростными сетями (типа RS/6000+Ethernet) имеют высокие значения латентности и малую скорость передачи данных, поэтому эффективны в алгоритмах с малой долей обменных операций. Параллельные MIMD-архитектуры имеют меньшую латентность и высокую скорость передачи данных, эффективность их применения определяется соотношением между вычислительной и обменной частью алгоритма [11].

Качество параллельного алгоритма оценивается коэффициентами реального ускорения и эффективности, которые являются функциями многих параметров: $\varphi(p, m, t_s, t_w, t_{op}, T_F)$.

Для MIMD-систем и СОДУ с тривиальными правыми частями эффективность распараллеливания для всех трех параллельных алгоритмов достаточно низкая (рис. 2.8). Для СОДУ с доминирующими правыми частями очевидным преимуществом обладает первая схема, а вторая и третья схемы дают очень близкие результаты. При увеличении порядка метода, коэффициенты ускорения и эффективности увеличиваются, но незначительно: $\uparrow r \Rightarrow \uparrow s \Rightarrow \uparrow S \Rightarrow \uparrow E$, с ростом размерности задачи существенно возрастают: $\uparrow m \Rightarrow \uparrow S \Rightarrow \uparrow E$.

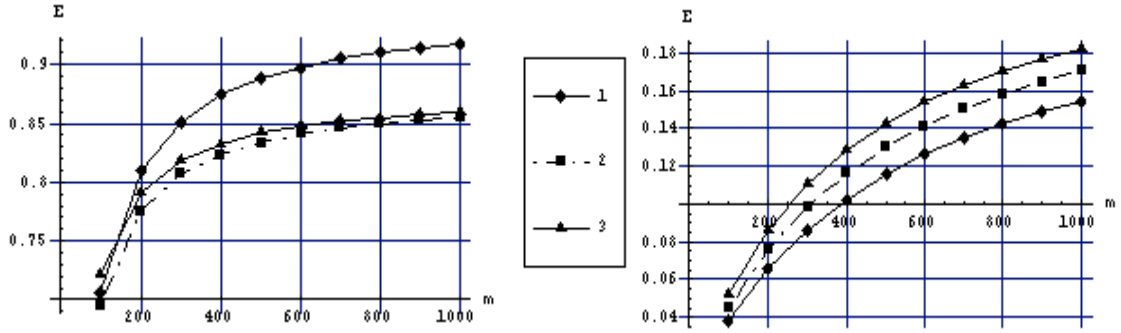


$T_F >> t_{op}$ $T_F \sim t_{op}$

Рисунок 2.8 – Коэффициенты эффективности вычислительных схем ЯМРК + правило Рунге для MIMD-систем

Описанные закономерности аналогичны для MIMD и кластерных систем (рис. 2.9), подтверждаются серией экспериментов, проведенных на системе тестов для нежестких СОДУ [37]. Этот факт свидетельствует о том, что улучшение временных характеристик современных коммуникационных сетей в кластере делает такие многопроцессорные архитектуры эффективными для разрабатываемых параллельных приложений.

Параллельные вычислительные системы типа SIMD отличаются от других типов ВС тем, что на каждом процессорном элементе выполняется один и тот же поток команд. Поскольку во второй и третьей схемах вычисления в разных группах процессоров не однородны, исследовать их отображения на SIMD-системы неправомерно. Проанализируем реальный параллелизм только первой вычислительной схемы.



$T_F >> t_{op}$ $T_F \sim t_{op}$

Рисунок 2.9 – Коэффициенты эффективности вычислительных схем ЯМРК + правило Рунге для кластерных систем

Для нее характерны невысокие показатели ускорения и эффективности, причем этот факт имеет место при любых значениях параметров: размерности задачи, числа процессорных элементов и сложности правых частей (рис. 2.10).

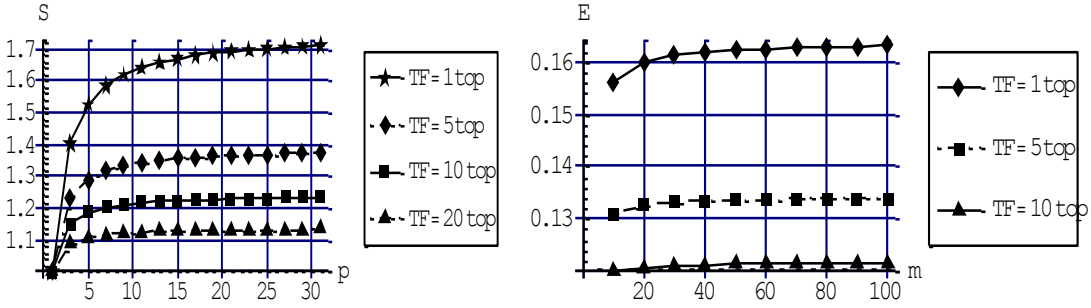


Рисунок 2.10 – Коэффициенты ускорения и эффективности вычислительной схемы №1 ЯМРК с правилом Рунге для SIMD-систем

Небольшие затраты на реализацию обменов при синхронной параллельности компенсируются последовательными вычислениями правой части СОДУ, поэтому распараллеливание нелинейной задачи Коши с правилом Рунге на ВС SIMD-архитектуры является мало эффективным. Таким образом, эффективное распараллеливание нелинейной задачи Коши на основе явных схем с правилом Рунге возможно при наличии сложной правой части СОДУ (порядка сотен и больше флопов) для асинхронных параллельных архитектур с высокоскоростными сетями обмена данными.

2.2. Параллельное решение нелинейных динамических задач на основе явных вложенных методов

Наиболее эффективным способом определения локальной погрешности при решении задачи Коши для однопроцессорных машин являются **методы вложенных форм**, которые дополнительно требуют вычисления одного или нескольких стадийных коэффициентов для определения более точной аппроксимации решения на шаге. Вложенные методы определяют две формулы Рунге-Кутты смежных порядков точности $r(\hat{r})$ и для интегрирования СОДУ имеют вид:

$$\begin{cases} \bar{y}_{n+1} = \bar{y}_n + h \cdot \sum_{i=1}^{s} b_i \cdot \bar{k}_i, \\ \hat{\bar{y}}_{n+1} = \bar{y}_n + h \cdot \sum_{i=1}^{s'} \hat{b}_i \cdot \bar{k}_i. \end{cases} \qquad (2.11)$$

Если основу вложенных форм составляют явные разностные схемы, то стадийные коэффициенты определяются по формулам:

$$\begin{cases} \bar{k}_i = F(x_n + c_i h; \bar{g}_i), \\ \bar{g}_i = \bar{y}_n + h \sum_{j=1}^{i-1} a_{ij} \cdot \bar{k}_j, i = \overline{1,s'}, \end{cases} \qquad (2.12)$$

с той разницей, что их количество определяется как $max(s,s')$. Для определенности предположим, что $s' > s$.

При разработке параллельного алгоритма вложенного метода Рунге-Кутты воспользуемся **иерархической декомпозиционной методикой** и результатами разработки параллельного алгоритма ЯМРК с правилом Рунге. Процесс вычислений по формулам (2.11)-(2.12) можно представить как решение двух подзадач: вычисление приближенного решения $\bar{y}_{n+1}(h)$ порядка r в точке x_{n+1} с шагом h и $\hat{\bar{y}}_{n+1}(h)$ порядка $\hat{r}$ в точке x_{n+1} с тем же шагом.

Граф влияния макрооперационной схемы алгоритма приведен на рисунке 2.11, максимальная степень параллелизма совпадает с размерностью СОДУ и равна $Dop = m$. В свою очередь макрооперации рисунка 2.11 можно представить, как решение s' подзадач вычисления стадийных коэффициентов и двух подзадач вычисления приближенного решения.

Очевидно, множество макроопераций для ВМРК совпадает с множеством макроопераций ЯМРК с правилом Рунге. Графы влияния обеих макроопераций приведены на рис. 2.2-2.3.

Время последовательного явного ВМРК $r(\hat{r})$ включает время вычисления стадийных коэффициентов и аппроксимаций решения:

$$T_1^2 = (s+1)mT_F + [s^2m + 6sm + 2s + 4m]t_{op}. \qquad (2.13)$$

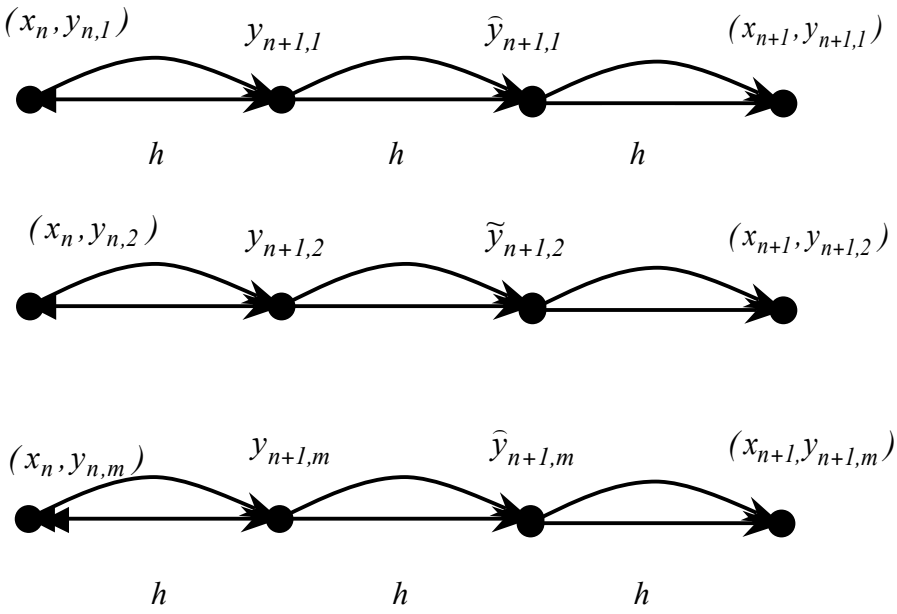


Рисунок 2.11 – Граф влияния макрооперационной схемы вложенных явных методов для нелинейной задачи Коши

На рисунке 2.12 представлен параллельный алгоритм решения задачи Коши по методу вложенных форм для мультикомпьютера из p процессоров. Пусть СОДУ имеет размерность, равную m; тогда m_1 компонента стадийных коэффициентов и векторов решений распределяется на каждый из процессоров: $m_1 = m$ при $m = p$ и $m_1 = \lceil m/p \rceil$ при $m > p$. Явные численные схемы определяют стадийные коэффициенты последовательно, поэтому для каждого из них, m_1 компонента дополнительного вектора $\bar{g}_i$ может быть вычислена параллельно. После реализации каждого такого вычисления компоненты вектора $\bar{g}_i, i = \overline{1,s'}$ должны быть доступны всем процессорам для вычисления очередного стадийного коэффициента.

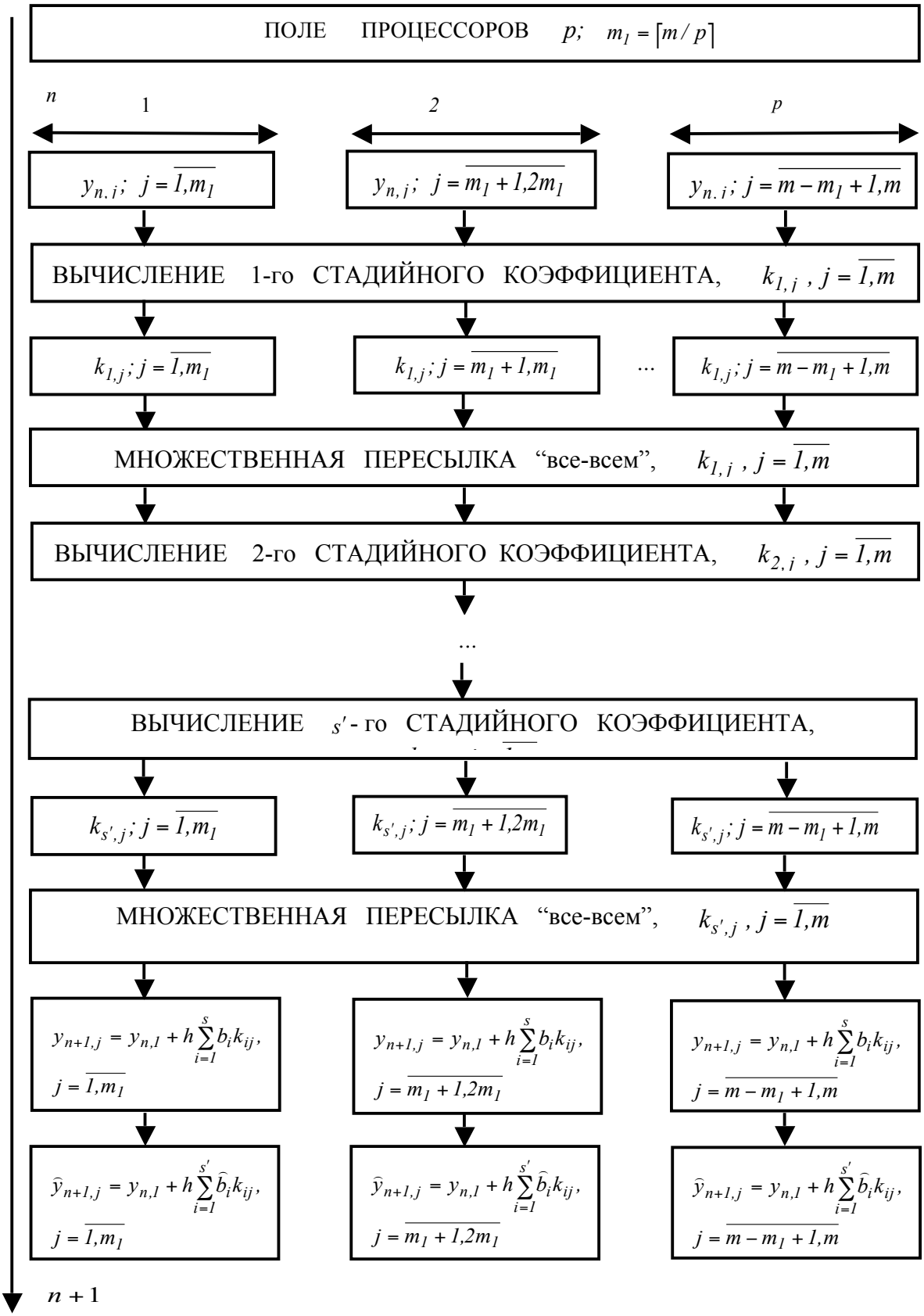


Рисунок 2.12 – Параллельный алгоритм решения нелинейной задачи Коши для СОДУ явными ВМРК $r(\hat{r})$

Затем каждый процессор вычисляет равную часть компонент вектора стадийных коэффициентов, путем соответствующего вычисления компонент правой части исходной СОДУ, и равное количество компонент векторов решений. Заметим, что дополнительных пересылок здесь не понадобится, так как каждый процессор на следующем временном шаге будет работать с одной и той же частью компонент, как стадийных векторов, так и векторов решений. Время выполнения параллельного алгоритма ВМРК $r(\hat{r})$ включает время выполнения арифметических операций и обменов:

$$T^2_{p,comp} = m_1(s+1)T_F + (s^2 m_1 + 6sm_1 + 2s + 4m_1)t_{op}. \quad (2.14)$$

$$T^2_{p,comm} = s' T_{all-to-all}(m_1, p), \text{ где } m_1 = \lceil m/p \rceil. \quad (2.15)$$

Оценки потенциального параллелизма ВМРК близки к оптимальным, то есть практически линейное ускорение и единичная эффективность. Определим время передачи данных для алгоритма ВМРК для различных топологий, объем передаваемых данных равен $m_1 = \lceil m/p \rceil$:

– кольцо: $T^2_{p,comm} = T^{2,R}_{all-to-all} = s' \cdot \left[\left(t_s + \lceil m/p \rceil \cdot t_w \right) \cdot (p-1) \right]$; (2.16)

– решетка: $T^2_{p,comm} = T^{2,M}_{all-to-all} = s' \cdot \left[2t_s(\sqrt{p}-1) + \lceil m/p \rceil \cdot t_w \cdot (p-1) \right]$; (2.17)

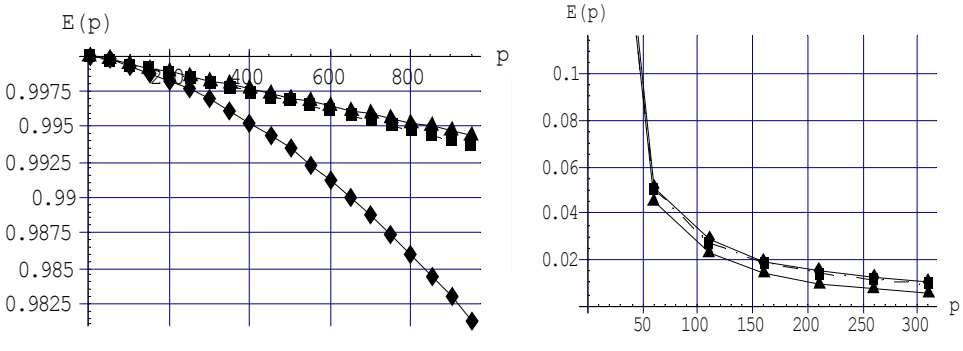
– гиперкуб: $T^2_{p,comm} = T^{2,H}_{all-to-all} = s' \cdot \left[t_s \cdot log_2 p + t_w \cdot \lceil m/p \rceil \cdot (p-1) \right]$. (2.18)

Поскольку время вычислений параллельного алгоритма для кластерных и MIMD систем идентично, то большое значение приобретает величина коммуникационной составляющей. Проанализируем изменение времени обмена от топологии соединения процессоров, коммуникационных констант и сложности правой части СОДУ (рис. 2.13). Как и для предыдущих алгоритмов, соединение по типу кольцо является наихудшим вариантом, топологии гиперкуб и тор дают практически идентичные результаты. Доля обменных операций уменьшается с ростом сложности правой части СОДУ для всех типов параллельных ВС и топологий.

Для кластерных и MIMD-систем (рис. 2.14) поведение динамических характеристик параллельного алгоритма ВЯМРК идентично. Определим, при каких условиях расходы на параллелизм окупаются и распараллеливание становится эффективным.

Рисунок 2.13 – Доля времени обменов в общем времени выполнения параллельных ВЯМРК для MIMD-систем в разных топологиях

Коэффициенты ускорения и эффективности увеличиваются с ростом размерности задачи: $\uparrow m \Rightarrow \uparrow S \Rightarrow \uparrow E$ и сложности правой части СОДУ $\uparrow T_F \Rightarrow \uparrow S \Rightarrow \uparrow E$. Величина коэффициента ускорения увеличивается, а коэффициента эффективности уменьшается с ростом числа процессоров $\uparrow p \Rightarrow \uparrow S \Rightarrow \downarrow E$.



$T_f >> t_{op}$ $T_f \sim t_{op}$

Рисунок 2.14 – Коэффициенты эффективности вложенных явных методов для топологий: кольцо, тор, гиперкуб в MIMD-архитектуре

Заметим, что это справедливо, пока число процессоров не превысит **максимальную степень параллелизма**. Таким образом, распараллеливание ВЯМРК эффективно при сложных правых частях, и расходы на параллелизм не окупаются при тривиальных правых частях СОДУ [48-50].

Для SIMD-систем обращения к правой части СОДУ выполняются последовательно, что существенно влияет на характеристики параллелизма. Даже, если вектор правых частей СОДУ является тривиальным по вычислениям, последовательная реализация обращения к нему при больших размерностях задачи становится определяющей. Коэффициенты реального ускорения и эффективности для синхронных систем незначительны, уменьшаются с ростом числа процессоров и практически совпадают для различных топологий.

2.3. Параллельная реализация технологии локальной экстраполяции Ричардсона для явных одношаговых схем

2.3.1. Повышение эффективности последовательной реализации технологии локальной экстраполяции

Построим эффективный с точки зрения минимизации вычислительных затрат метод интегрирования СОДУ на основе **технологии Ричардсона**. **Экстраполяционная технология** включает опорный численный метод решения задачи Коши, последовательность сеток интегрирования, рекуррентное правило вычисления значений приближенного решения. Эффективность ее применения напрямую зависит от правильного выбора и сочетания всех трех составляющих. Для получения сеток интегрирования применяются числовые последовательности, образованные гармоническим рядом, рядом четных чисел, степенями двойки [119]:

$$\begin{aligned} &1)\ P_1 = \{1,2,3,4,5,6,...\} - \quad n_j = j; \\ &2)\ P_2 = \{1,2,4,6,8,10,12,...\} - \quad n_j = 2(j-1), j > 1 \vee n_1 = 1; \\ &3)\ P_3 = \{1,2,4,8,16,32,64,...\} - \quad n_j = 2^{j-1}. \end{aligned} \tag{2.19}$$

Вычислительная сложность генерации сеток интегрирования определяется числом обращений к правой части СОДУ и равна:

$$\begin{aligned} &N(k) = n_1 + n_2 + ... + n_k, \\ &P_1: \quad N(k) = (k^2 + k)/2, \\ &P_2: \quad N(k) = k^2 - k + 1, \\ &P_3: \quad N(k) = 2^k - 1, \end{aligned} \tag{2.20}$$

где k – число строк экстраполяционной таблицы.

При одном и том же опорном методе наиболее затратной является последовательность P_3 , которая фактически повторяет процесс удвоения шага, последовательности P_1 и P_2 имеют приблизительно одинаковую вычислительную сложность [46,62]. Выбор опорного метода интегрирования базируется на исследовании влияния порядка метода и его свойств на объем необходимых вычислений.

Пусть опорный метод имеет порядок – r_0 , длина экстраполяционной таблицы равна k, а число стадий явного метода – s, определим необходимый размер экстраполяционной таблицы для получения аппроксимации решения порядка r . **При условии симметричности опорного явного одношагового метода, он имеет асимптотическое разложение по степеням h^2 , следовательно, каждая экстраполяция исключает две степени** h [119-121]. Формирование экстраполяционной таблицы для симметричных методов на основе четных рядов требует $k = \lceil (r - r_0)/2 + 1 \rceil$ строк для достижения точности порядка r , а для произвольных методов – $k = r - r_0 + 1$. Таким образом, h^2 -экстраполяция на основе симметричных опорных методов и второй четной последовательности для генерации сетки интегрирования является наиболее эффективной с точки зрения вычислительных затрат.

Например, если в качестве опорного метода выбирается симметричный явный метод второго порядка точности, $r_0 = 2$, тогда для P_2 число вычислений f равно: $N(r/2) = (r^2 - 2 \cdot r + 4)/4$. Для произвольных опорных методов и гармонического ряда имеем: $N(r-1) = (r^2 - r)/2.$

Время последовательных вычислений по схеме локальной экстраполяции состоит из времени определения k аппроксимаций решения с различными шагами интегрирования и времени составления экстраполяционной таблицы:

$$T_1^3 = T_{T11} + T_{T21} + \ldots + T_{Tk1} + T_1^{ext-tab}.$$

Каждая из аппроксимаций $T_{11},\ldots,T_{k1}$ получается за счет n_i раз примененной схемы явного одношагового опорного метода с разными шагами интегрирования:

$$\begin{cases} T_{T11} = T_{\bar{y}_{n+1}^{(1)}\left(\frac{h}{n_1}\right)} = n_1 \cdot T_1^{r_0}, \\ \quad \ldots \\ T_{Tk1} = T_{\bar{y}_{n+1}^{(k)}\left(\frac{h}{n_k}\right)} = n_k \cdot T_1^{r_0}, \end{cases} \tag{2.21}$$

где $T_1^{r_0}$ – время решения задачи Коши для СОДУ размерности m опорным методом порядка r_0. Очевидно, что $T_1^3 = T_1^{r_0} \cdot \sum_{i=1}^{k} n_i + T_1^{ext-tab}$.

Время вычисления экстраполяционной таблицы по каждой размерности СОДУ равно произведению количества экстраполированных значений $K_{elem-ext}$ (число элементов таблицы без первого столбца) на время вычисления одного экстраполированного значения по формуле (1.21), T_1^{ext}:

$$K_{elem-ext} = (k^2 - k)/2 \; ; \; T_1^{ext} = m(2t_{mul} + 3t_{ad}) = 5mt_{op}; \tag{2.22}$$

$$T_1^{ext-tab} = K_{elem-ext} \cdot T_1^{ext} = 5m(k^2 - k)/2 \cdot t_{op}. \tag{2.23}$$

Для симметричных опорных методов и P_2 имеем:

$$N_{P_2}(k) = k^2 - k + 1; k = \lceil (r - r_0)/2 + 1 \rceil. \tag{2.24}$$

Таким образом, время выполнения последовательного алгоритма в применении к нелинейной задаче Коши с оценкой апостериорной локальной погрешности по технологии локальной экстраполяции составляет:

$$\begin{cases} T_1^3 = \left(\frac{r^2 - 2r + 4}{4} + \frac{a^2 - 2ar + 2a}{4} \right) T_1^{r_o} + T_1^{ext-tab}, \\ T_1^{ext-tab} = 5m \left(\frac{r^2 - 2r}{8} + \frac{a^2 - 2ar + 2a}{8} \right) \cdot t_{op}, \; a = r_0 - 2. \end{cases} \tag{2.25}$$

Оценки времени вычисления одного шага интегрирования для явных опорных симметричных методов различного порядка точности ($r_0 = 2 \div 10$) и второй четной последовательности сведены в таблицу 2.2.

Таблица 2.2

Вычислительная сложность одного шага интегрирования по технологии локальной экстраполяции с симметричным опорным методом

Порядок /число стадий симметричного опорного ЯМРК, r_0 / s	Накладные расходы на 1 шаг интегрирования по ЯМРК порядка r_0	Длина экстраполяционной таблицы, k
2/2	$2mT_F + (8m+2)t_{op}$	$r/2$
4/4	$4mT_F + (24m+6)t_{op}$	$r/2-1$
6/7	$7mT_F + (63m+12)t_{op}$	$r/2-2$
8/11	$11mT_F + (143m+20)t_{op}$	$r/2-3$
10/17	$17mT_F + (323m+32)t_{op}$	$r/2-4$

Вычислительная сложность одного шага интегрирования по технологии локальной экстраполяции при одних и тех же способах получения сеток интегрирования и порядке полученного экстраполяционного метода r существенно зависит от порядка опорного метода и его свойств, времени вычисления правой части СОДУ (рис. 2.15). Так, длина экстраполяционной таблицы линейно уменьшается с ростом порядка опорного метода: $r_0 \uparrow \Rightarrow k \downarrow$ [63-65].

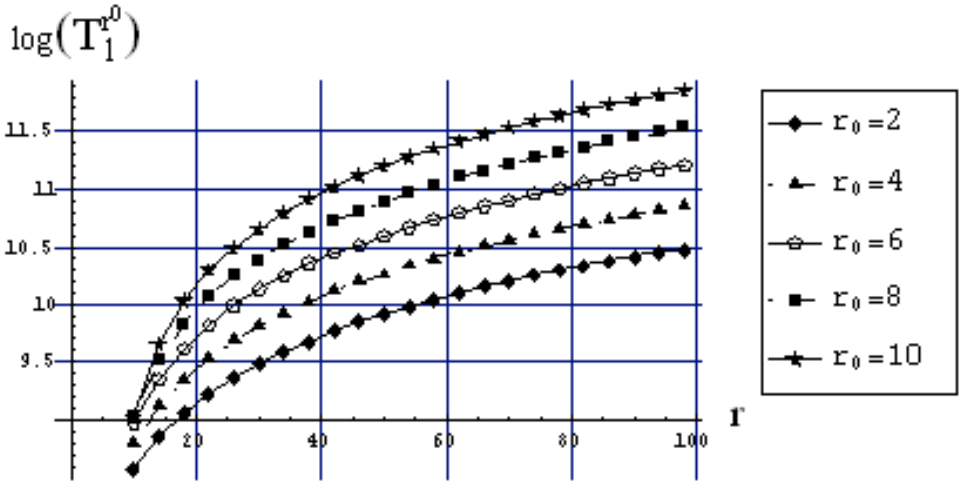


Рисунок 2.15 – Время выполнения одного шага интегрирования по технологии локальной экстраполяции с явным опорным методом порядка r_0

Однако, несмотря на уменьшение количества строк экстраполяционной таблицы, накладные расходы на экстраполяцию возрастают с ростом порядка опорного метода: $r_0 \uparrow \Rightarrow T_1^{r0} \uparrow$, причем при $T_F \gg t_{op}$ в $1{,}55 \div 2$ раза для методов смежных порядков, при $T_F \approx t_{op}$ в десятки/сотни раз. Кроме того, накладные расходы возрастают с ростом размерности задачи: $m \uparrow \Rightarrow T_1^{r0} \uparrow$.

Суммируя все полученные результаты, можно сделать вывод, что для уменьшения накладных расходов при применении технологии локальной экстраполяции следует выбирать опорный метод малого порядка точности.

Заметим, что **экстраполяционная таблица** вычисляется для каждой размерности исходной СОДУ. Теоретически нет никаких ограничений на длину экстраполяционной таблицы, но конечность памяти машины и накапливание ошибок округления, ограничивают длину экстраполяционной таблицы сверху и, как правило, используется: $k_{max} \le 10$. Поэтому для получения **высокоточных приложений** $(10^{-15} - 10^{-20})$, где вычисления правой части СОДУ достаточно сложны, не имеется другой возможности, как использовать технологию локальной экстраполяции Ричардсона на базе опорного метода высокого порядка.

Таким образом, выбирая симметричный явный метод Рунге-Кутты второго порядка, имеем время для выполнения одного шага интегрирования по схеме локальной экстраполяции равным:

$$\begin{cases} T_1^3 = \left(\dfrac{r^2 - 2r + 4}{4}\right) \cdot T_1^{r_o=2} + T_1^{ext-tab}, \\ T_1^{ext-tab} = 5m\left(\dfrac{r^2 - 2r}{8}\right) \cdot t_{op},\ T_1^{r_o=2} = 2m \cdot T_F + (8m+2) \cdot t_{op}, \end{cases} \Rightarrow$$

$$T_1^3 = \left(\frac{r^2 - 2r + 4}{2}\right) m \cdot T_F + \left[\left(\frac{21}{8} r^2 - \frac{42}{8} r + 8\right) m + \frac{r^2 - 2r + 4}{2}\right] \cdot t_{op}. \qquad (2.26)$$

2.3.2. Параллельные алгоритмы решения нелинейной задачи Коши на основе технологии локальной экстраполяции

Потенциально вычисления по технологии локальной экстраполяции содержат три источника внутреннего параллелизма:

– **системный параллелизм** (ограничен размерностью СОДУ);

– **параллелизм экстраполяции** (ограничен размерностью таблицы экстраполяции);

– **параллелизм опорного метода** (малая степень параллелизма).

Построение вычислительных схем параллельных алгоритмов решения нелинейной задачи Коши для СОДУ по технологии Ричардсона базируется на декомпозиционной методике. После того, как определены все составляющие технологии локальной экстраполяции Ричардсона, процесс решения можно разбить на две последовательно выполняемые подзадачи:

1) вычисление k аппроксимаций решения в точке x_{n+1} с разными шагами интегрирования: $\bar{y}_{h_i}(x_n + H) = \bar{y}_{n+1}^{(i)}$;

2) построение экстраполяционной таблицы.

Организация параллельных вычислений может производиться двумя способами, отсюда и две принципиально различные макрооперационные схемы алгоритма. Первый вариант подразумевает использование только системного параллелизма, второй – комбинацию параллелизма экстраполяции и системного.

Рассмотрим первую подзадачу алгоритма. В качестве основной макрооперации для обоих вариантов вводится однократное вычисление аппроксимации точного решения в точке сетки с заданным шагом интегрирования на базе явного опорного метода. Граф влияния первого варианта макрооперационной вычислительной схемы приведен на рисунке 2.16, здесь все аппроксимации решения вычисляются последовательно одновременно для каждой размерности системы.

По второй макрооперационной схеме параллельно вычисляется k независимых аппроксимаций решения в точке $x_n + H$, граф влияния для такой схемы приведен на рисунке 2.17. Второй вариант алгоритма имеет большую степень параллелизма, равную $Dop = m \cdot k$, в отличие от первого: $Dop = m$.

Однако, этот недостаток схемы №1 может быть компенсирован лучшей балансировкой загрузки многопроцессорной вычислительной системы, а также низкой интенсивностью обменов.

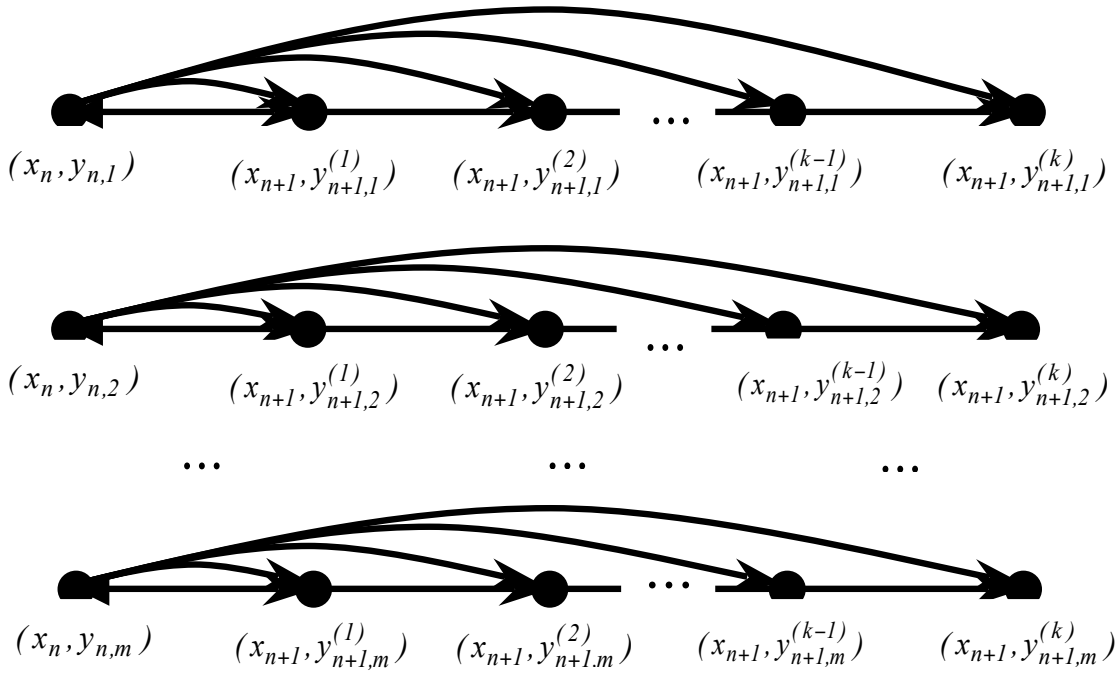


Рисунок 2.16 – Граф влияния макрооперационной схемы для вычисления аппроксимаций решения СОДУ, вариант №1

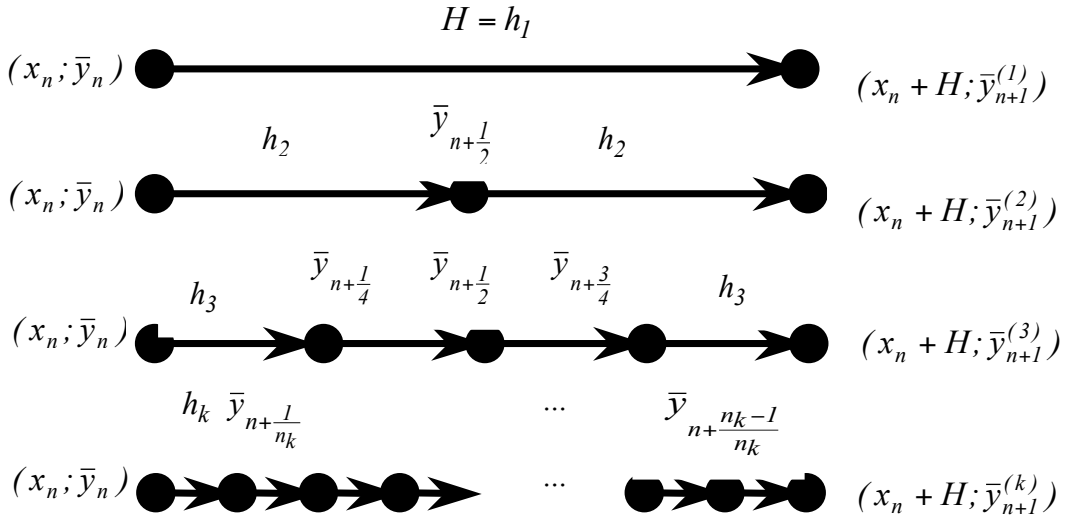


Рисунок 2.17 – Граф влияния макрооперационной схемы для вычисления аппроксимаций решения СОДУ, вариант № 2

Вычисление экстраполяционной таблицы в зависимости от способа решения первой подзадачи, также может быть распараллелено разными способами. В качестве основной макрооперации для этой подзадачи вводится вычисление одного значения $T_{il,j}, j = \overline{1,m}, i = \overline{2,k}; l = \overline{i,k}$ по формуле полиномиальной экстраполяции Эйткена-Невилла (1.21). При построении экстраполяционной таблицы распараллеливание может быть осуществлено за счет независимого выполнения вычислений по формуле полиномиальной экстраполяции по каждой размерности системы ($Dop = m$), а также за счет распределенного вычисления каждой строки таблицы ($Dop = k - 1$). Как правило, $m > k$, поэтому появляется возможность совместить оба вида параллелизма. Граф влияния для метода построения экстраполяционной таблицы приведен на рисунке 2.18.

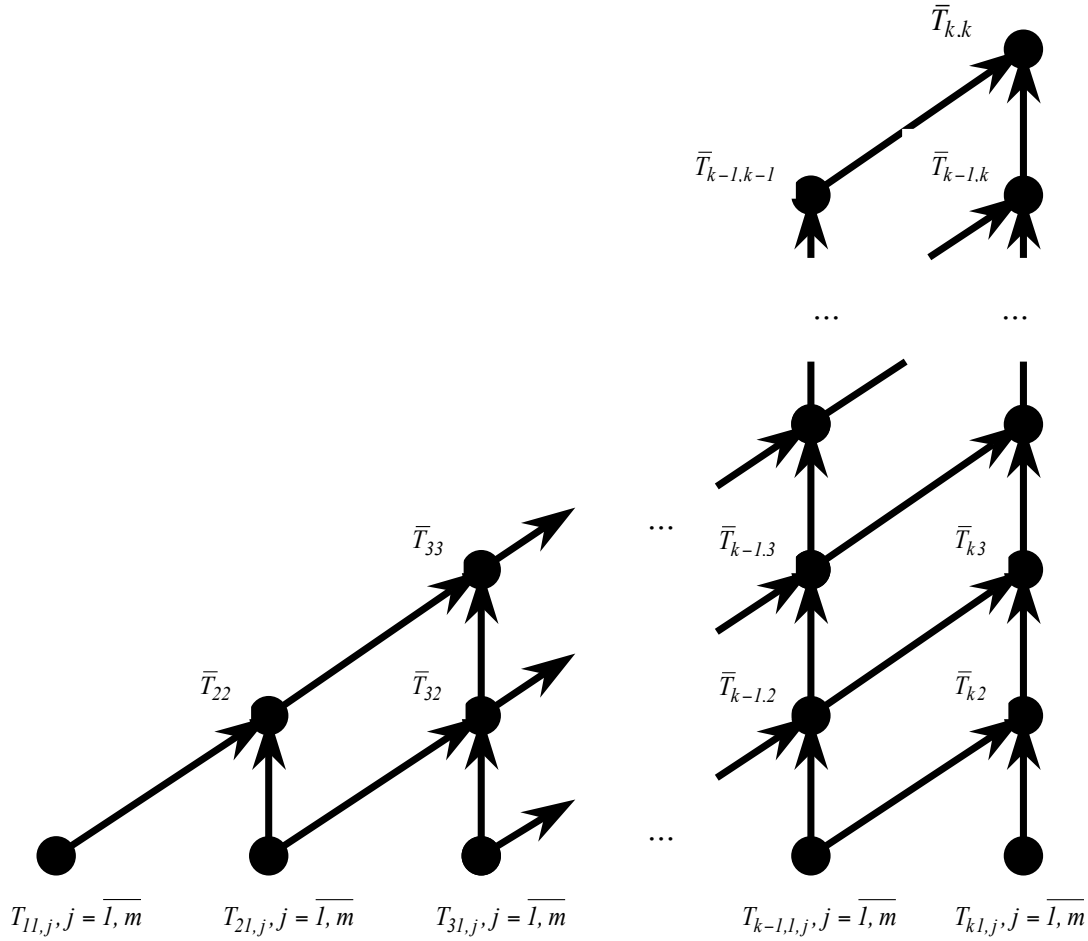


Рисунок 2.18 – Граф влияния для метода построения экстраполяционной таблицы в технологии Ричардсона

На рассмотренном этапе детализации дан предварительный анализ эффективности каждой из вычислительных схем. Перейдем к использованию внутреннего параллелизма метода. Первая макрооперационная схема алгоритма локальной экстраполяции повторяет макрооперационную схему №1 для правила Рунге, с тем различием, что теперь у нас не три, а k последовательно вычисляемых аппроксимаций по ЯМРК. Заметим, что, каждую макрооперацию схемы локальной экстраполяции можно представить, как n_i раз используемую макрооперацию однократного применения явной численной схемы одношагового метода.

Рисунок 2.19 представляет параллельный алгоритм решения системы обыкновенных дифференциальных уравнений на мультикомпьютере из p процессоров для первой макрооперационной схемы. Для СОДУ, состоящей из m уравнений, на одном временном шаге $m_1, m_1 = \lceil m/p \rceil$ компонент каждой аппроксимации решения могут быть вычислены параллельно. Соответственно, при построении экстраполяционной таблицы все значения $T_{il}, i = \overline{2,k}; l = \overline{i,k}$ определяются последовательно, но на каждом процессоре вычисляется одна при $p = m$ или $m_1 = \lceil m/p \rceil$ при $p < m$ компонент СОДУ.

Время вычислений по параллельной схеме №1 составляет:

1) $T_p^{31} = T_p^{r0} \cdot \sum_{i=1}^{k} n_i + T_p^{31,ext-tab}$, где $T_p^{r_0}$ – время параллельного вычисления опорного метода порядка r_0;

2) $T_{p,comp}^{31,ext-tab}$ – время вычисления экстраполяционной таблицы на p процессорах с использованием первой схемы локальной экстраполяции.

Как результат, при использовании h^2-экстраполяции имеем:

$$T_{p,comp}^{31} = \frac{(r^2 - 2\cdot r + 4)}{2} m_1 T_F + \left[\frac{r^2 - 2r + 4}{2} + \left(\frac{21}{8} r^2 - \frac{42}{8} r + 8 \right) m_1 \right] \cdot t_{op}; \qquad (2.27)$$

$$T_{p,comm}^{31} = \frac{(r^2 - 2\cdot r + 4)}{2} T_{all-to-all}(m_1, p). \qquad (2.28)$$

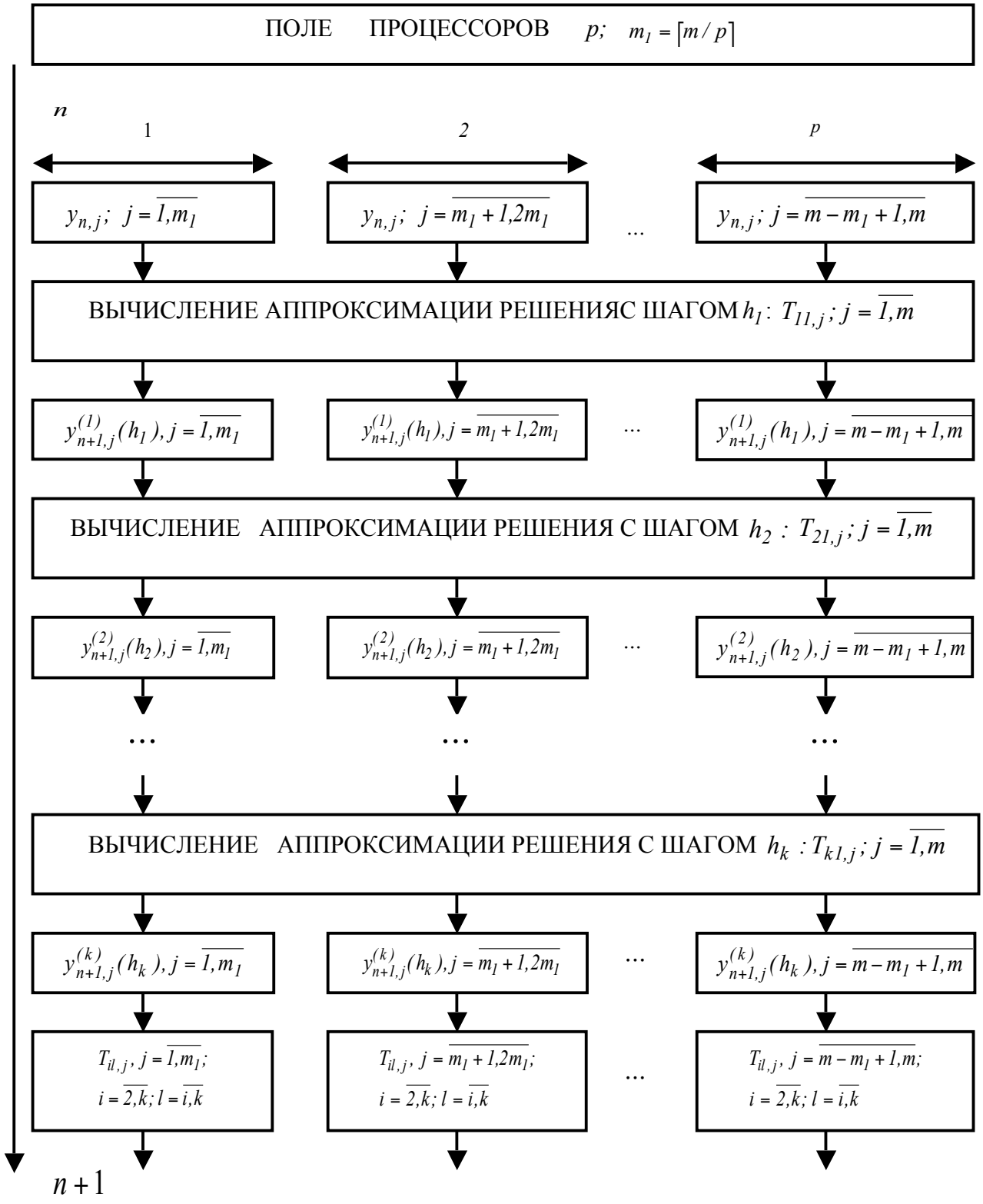


Рисунок 2.19 – Параллельный алгоритм решения нелинейной задачи Коши на основе локальной экстраполяции с явным одношаговым опорным методом, вариант №1

Межпроцессорный обмен при первом варианте распараллеливания определяется $N(k)$ раз выполненной операцией множественной пересылки данных, необходимой при вычислении коэффициентов $\bar{k}_l, l = \overline{1,s}$. Преимуществом рассмотренной схемы является отсутствие необходимости на каждом шаге перегруппировывать данные для следующего шага итерации, то есть все вычисления по m_1 компонентам локализованы на одном процессоре, включая и вычисления для таблицы экстраполяции.

На рисунке 2.20 представлен параллельный алгоритм решения СОДУ на мультикомпьютере из p процессоров макрооперационной схемы №2. Если размерность процессорного поля велика, появляется возможность выполнять макрооперации первой подзадачи параллельно. Для этого необходимо разбить процессоры на группы таким образом, чтобы обеспечить наиболее рациональную балансировку загрузки многопроцессорной системы.

Рассмотрим следующие способы разбиения процессоров, а соответственно и распределения данных по процессорам: **равномерный и пропорциональный**. При наиболее простом, равномерном способе, каждый процессор или процессорная группа будут отвечать за вычисление равного количества аппроксимаций. Если размерность процессорного поля равна p , количество строк в экстраполяционной таблице – k , то при таком способе будет k групп по $p_i = \lceil p / k \rceil$, $i = \overline{1,k}$ процессоров в каждой и каждый процессор в i-той группе будет содержать $m_i = \lceil m / p_i \rceil$ компонент соответствующей аппроксимации решения и соответствующего экстраполированного значения. Для равномерного разбиения: $p_1 = p_2 = ... = p_k$ и, соответственно, $m_1 = m_2 = ... = m_k$. Каждая группа будет отвечать за вычисление определенной аппроксимации решения: $T_{j1}, j = \overline{1,k}$, время решения определяется по $\max\limits_{j=1}^{k} T_{j1}$, и все группы, кроме последней, будут простаивать некоторое время. Каждый процессор в i-той группе будет содержать $m_i = \lceil m / p_i \rceil$ компонент соответствующей аппроксимации решения и соответствующего экстраполированного значения.

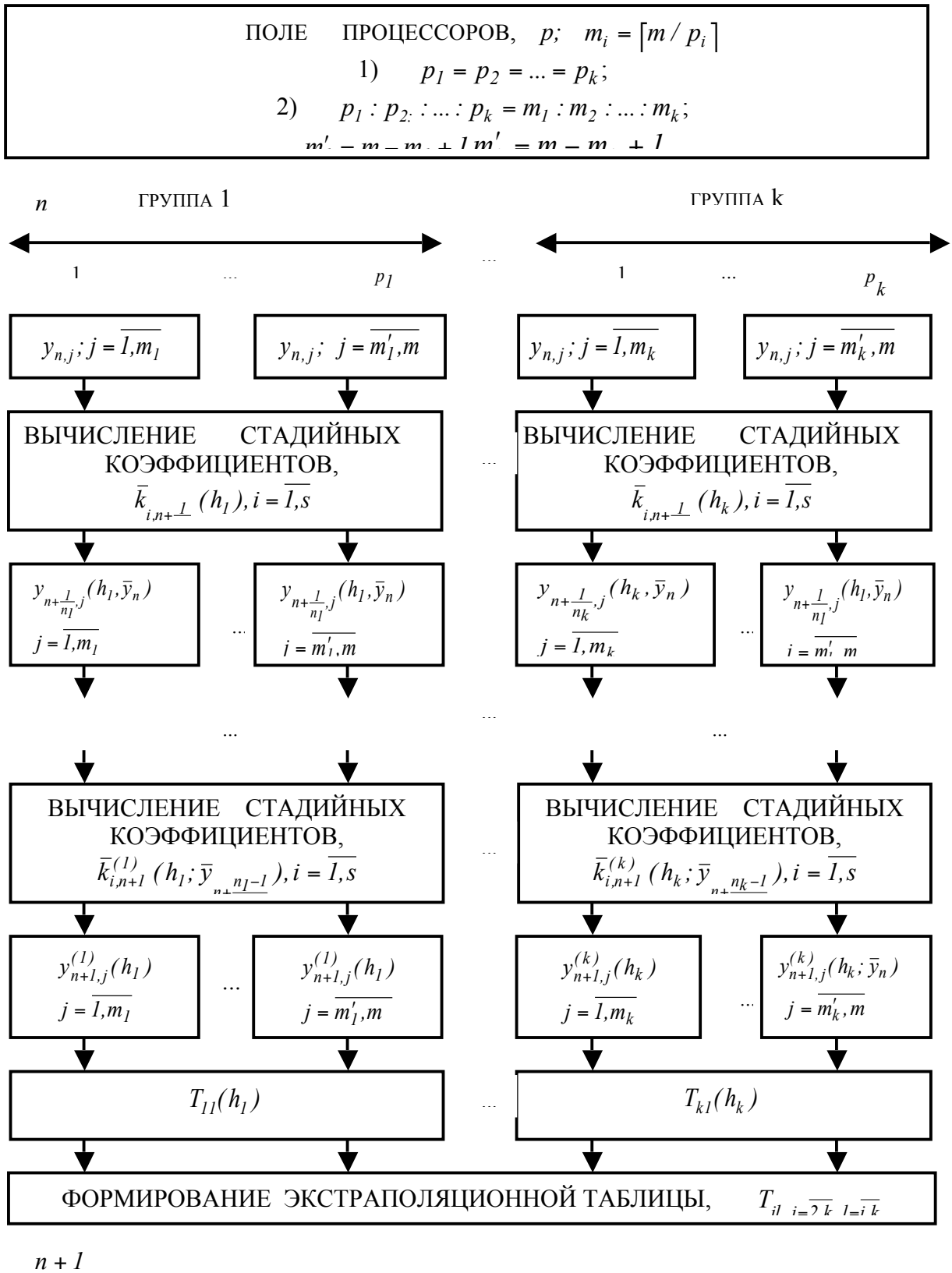


Рисунок 2.20 – Вычислительные схемы параллельного алгоритма №2 на основе локальной технологии Ричардсона, равномерное и пропорциональное разбиение

Каждая группа будет отвечать за вычисление определенной аппроксимации решения: $T_{j1}, j = \overline{1,k}$, время решения определяется как максимум из T_{i1} по $i = \overline{1,k}$ и все группы, кроме последней, будут простаивать некоторое время.

Определим время, необходимое для организации параллельного процесса выполнения первой подзадачи, как:

$$T_p^{32} = \max_{i=1}^{k}(n_i \cdot T_p^{r_0}) + T_p^{32,ext-tab}.$$

Поскольку при известном опорном методе время $T_p^{r_0=2}$ не зависит от номера аппроксимации, время выполнения арифметических операций равно: $T_{p,comp}^{32} = n_k \cdot T_p^{r_0=2} + T_{p,comp}^{32,ext-tab}$. Для последовательности $P_2 : n_k = n_{r/2} = r - 2$ и $m_k = m / p_k = mr / 2p$, следовательно,

$$T_{p,comp}^{32} = (r^2 - 2r)\frac{m}{p}T_F + (4r^2 - 8r)\frac{m}{p}t_{op} + (2r - 4)t_{op} + T_{p,comp}^{32,ext-tab}. \qquad (2.29)$$

Время на реализацию межпроцессорного обмена также определяется как максимальное значение времени обмена по всем аппроксимациям решения: $T_{p,comm}^{32} = n_k T_{all-to-all}(m_k, p_k) + T_{p,comm}^{32,ext-tab}$;

$$T_{p,comm}^{32} = (r - 2)T_{all-to-all}\left(\frac{mr}{2p}, \frac{2p}{r}\right) + T_{p,comm}^{32,ext-tab}. \qquad (2.30)$$

Рассмотрим различные варианты организации параллельных вычислений экстраполяционной таблицы при равномерном распределении для вычислительной схемы №2. После параллельного вычисления k аппроксимаций решения на k группах процессоров, имеется следующее распределение исходных данных для вычисления экстраполированных значений. Первая группа процессоров содержит вектор $T_{11,j}; j = \overline{1,m}$, вторая – $T_{21,j}; j = \overline{1,m}$ и, соответственно, k-тая – $T_{k1,j}; j = \overline{1,m}$, каждый процессор в группах содержит одинаковое m_i число компонент.

Для вычислений элементов экстраполяционной таблицы могут быть реализованы два варианта. Первый состоит в использовании системного параллелизма, каждый процессор группы передает соответствующему процессору соседней группы все компоненты своего вектора аппроксимации решения.

Так, первый процессор первой группы передаст первому процессору второй группы подвектор: $T_{11,j}; j = \overline{1,m_1}$ для вычисления $T_{22,j}; j = \overline{1,m_2}$; второй процессор первой группы передает $T_{11,j} j = \overline{m_1 + 1,2m_1}$ второму процессору второй группы для вычисления $T_{22,j}; j = \overline{m_2 + 1,2m_2}$ и так далее, первый процессор $(k-1)$ группы передает первому процессору k группы значения $T_{k-1,1,j} j = \overline{1,m_{k-1}}$ для вычисления $T_{k2,j}; j = \overline{1,m_k}$. Высота такого параллельного алгоритма равна $k-1$, то есть на первом шаге вычисляются подвекторы второй строки таблицы экстраполяции, на последнем $k-1$ шаге вычисляются соответствующие подвекторы k –той строки таблицы экстраполяции.

Время на организацию вычислений по этому алгоритму в процессе формирования экстраполяционной таблицы равно:

$$T_{p,comp}^{32,1,ext-tab} = (k-1)m_i \cdot T_1^{ext} = 5m_i(k-1)t_{op} = 5 \cdot \frac{mr}{2p}\left(\frac{r-2}{2}\right)t_{op}, \quad (2.31)$$

$$T_{p,comp}^{32,1} = (r^2 - 2r)\frac{m}{p}T_F + (4r^2 - 8r)\frac{m}{p}t_{op} + (2r-4)t_{op} + 5 \cdot \frac{mr}{2p}\left(\frac{r-2}{2}\right)t_{op} \quad (2.32)$$

В то же время межпроцессорный обмен потребует $k-1$ операцию передачи данных по типу “точка-точка” и, для подготовки следующего шага интегрирования, однократной передачи данных от каждого процессора последней группы его подвектора решения каждому соответствующему процессору других групп:

$$T_{p,comm}^{32,1,ext-tab} = (k-1)T_{p-p}(m_i, p) = \frac{r-2}{2} \cdot T_{p-p}\left(\frac{mr}{2p}, p\right). \quad (2.33)$$

$$T_{p,comm}^{32} = (r-2)T_{all-to-all}\left(\frac{mr}{2p}, \frac{2p}{r}\right) + \frac{r-2}{2} \cdot T_{p-p}\left(\frac{mr}{2p}, p\right) + T_{single-brodcast}(m_i, p_i) \quad (2.34)$$

Второй алгоритм вычисления экстраполяционной таблицы состоит в следующем: в каждой группе процессоров ровно один процессор будет отвечать за передачу данных между группами. Для этого после подсчета элементов первого столбца таблицы в каждой группе процессоров производится обмен значениями по типу “все-всем”, в результате каждый процессор в группе будет содержать весь вектор аппроксимации решения, а не его часть. Процессоры первой группы – $\overline{T}_{11}$, второй – $\overline{T}_{21}$ и k - той – $\overline{T}_{k1}$.

Затем i - тый процессор каждой группы, кроме последней, передает всем процессорам соседней группы вектор решений для подсчета экстраполированного значения. Далее каждый процессор каждой группы кроме первой (ширина алгоритма уменьшается с каждым шагом) считает свой подвектор (длиной m_i) своего вектора экстраполированного значения: процессоры второй группы – $\overline{T}_{22}$, третьей – $\overline{T}_{32}$ и k- той – $\overline{T}_{k2}$. На последнем $k-1$ шаге будут работать только процессоры последней группы, и каждый будет считать m_k значений вектора $\overline{T}_{kk}$.

Таким образом, время на формирование экстраполяционной таблицы по описанному алгоритму составит:

$$T_{p,comp}^{32,2,ext-tab} = (k-1)m_i \cdot T_1^{ext} = 5m_i(k-1)t_{op};.$$

$$T_{p,comp}^{32,2,ext-tab} = 5\frac{mr}{2p}\left(\frac{r-2}{2}\right)t_{op} = 1{,}25\frac{mr}{p}(r-2)t_{op}, \qquad (2.35)$$

а время коммуникаций равно:

$$T_{p,comm}^{32,2,ext-tab} = T_{p,comm}^{32,2,in-gr} + T_{p,comm}^{32,2,over-gr} =$$
$$= (k-1)T_{all-to-all}(m_i, p_i) + (k-1)T_{sin\,gle-brodcast}(m, p_i);$$

$$T_{p,comm}^{32,2,ext-tab} = \left(\frac{r-2}{2}\right)T_{all-to-all}\left(\frac{mr}{2p}, \frac{2p}{r}\right) + \left(\frac{r-2}{2}\right)T_{sin\,gle-brodcast}\left(m, \frac{2p}{r}\right). \qquad (2.36)$$

Чтобы увеличить балансировку загрузки, разобьем процессорное поле на такое же число групп, но неравномерно, а пропорционально числам $n_i, i = \overline{1,k}$: $p_1 : p_2 : \ldots : p_k = n_1 : n_2 : \ldots : n_k$, $\sum_{k=1}^{k} p_i = p$, $p_i = pn_i / N(k)$. Алгоритм разбиения на группы следующий: если $p \vdots N(k)$, то берется $p_i = \lfloor pn_i / N(k) \rfloor$, а затем оставшиеся процессоры по одному добавляются в каждую из групп. Таким образом, каждая группа будет содержать тем большее число процессоров, чем больше шагов интегрирования на базовом шаге ей предстоит выполнить для получения аппроксимации решения.

Оценим время выполнения параллельного алгоритма при условии, что $p \vdots N(k)$, тем самым заранее несколько увеличивая время выполнения по описанной схеме (для простоты рассуждений):

$$T_{p,comp}^{33} = \max_{i=1}^{k}(n_i \cdot T_{p_i}^{EMRK,r^0}) + T_{p,comp}^{33,ext-tab},$$

$$T_{p,comp}^{33} = \frac{m}{p}\left(\frac{r^2-2r+4}{2}\right)T_F + 2\frac{m}{p}(r^2-2r+4)t_{op} + 2(r-2)t_{op} + T_{p,comp}^{33,ext-tab}. \quad (2.37)$$

Из двух рассмотренных вариантов получения экстраполяционной таблицы, именно второй подходит для пропорционального разбиения процессоров, поскольку не требует дополнительной сортировки данных из-за неравномерности их распределения.

$$T_{p,comp}^{33,ext-tab} = \sum_{j=2}^{k}\left(\max_{i=2}^{k} t_{T_{ij}}\right) = \sum_{j=2}^{k} m_i \cdot T_1^{ext} = 5\frac{mN}{p}\cdot\sum_{i=2}^{k}\frac{1}{n_i}\cdot t_{op} =$$

$$= 2{,}5\frac{m}{p}\cdot\left(\frac{r^2-2r+4}{2}\right)\cdot\sum_{i=2}^{r/2}\frac{1}{n_i}\cdot t_{op} = 2{,}5\frac{m}{p}\cdot\left(\frac{r^2-2r+4}{2}\right)\cdot\sum_{i=2}^{r/2}\frac{1}{2(i-1)}\cdot t_{op}. \quad (2.38)$$

$$T_{p,comm}^{33,ext-tab} = (k-1)\cdot\max_{i=1}^{k} T_{all-to-all}(m_i, p_i) + (k-1)\cdot T_{single-broadcast}(m, p_k). \quad (2.39)$$

Тогда общее время на коммуникационные расходы при пропорциональном способе разбиения данных равно:

$$T_{p,comm}^{33} = n_k \cdot T_{all-to-all}(m_k, p_k) + T_{p,comm}^{33,ext-tab} =$$

$$= (r-2)\cdot T_{all-to-all}\left(\frac{mN}{pn_k}, \frac{pn_k}{N}\right) + T_{p,comm}^{33,ext-tab} \quad (2.40)$$

Время вычислений и обменов для третьей вычислительной схемы с учетом RISC-архитектуры составляет:

$$T_{p,comp}^{33} = \frac{m}{p}\left(\frac{r^2-2r+4}{2}\right)\cdot T_F + 2\frac{m}{p}(r^2-2r+4)\cdot t_{op} + 2(r-2)\cdot t_{op} +$$

$$+ 2{,}5\frac{m}{p}\cdot\left(\frac{r^2-2r+4}{2}\right)\cdot\sum_{i=2}^{r/2}\frac{1}{2(i-1)}\cdot t_{op}. \quad (2.41)$$

$$T_{p,comm}^{33,ext-tab} = (k-1)\cdot\max_{i=1}^{k} T_{all-to-all}(m_i, p_i) + (k-1)\cdot T_{single-broadcast}(m, p_k). \quad (2.42)$$

Максимальная степень параллелизма для второй и третьей схем совпадает и равна: $Dop = p_{max} = m \cdot k$. В общем виде проблема

разбиения процессорного поля на группы, обеспечивающие оптимальное решение задачи балансировки загрузки МПВС, может быть сформулирована в терминах комбинаторной оптимизации. Точное решение задач этого класса за приемлемое время невозможно, так как задача является NP-сложной. Поэтому будем искать близкое к оптимальному рациональное решение приближенными эвристическими методами. Сократить время простоя можно при использовании комбинационного способа.

Идея комбинационного метода состоит в использовании $\lceil k/2 \rceil$ групп процессоров, причем i-тая группа вычисляет i-тую и $(k-i+1)$-тую аппроксимации решения. Для получения i- той аппроксимации необходимо n_i раз выполнить обращение к опорному методу решения, для $(k-i+1)$ - той – n_{k-i+1} раз. То есть первая группа должна $n_1 + n_k$ раз обратиться к вычислению решения по опорному методу, вторая группа – $n_2 + n_{k-1}$ и, наконец, $k1 = \lceil k/2 \rceil$ группа – $n_{k1} + n_{k1+1}$. Напомним, что для P_2: $\{1,2,4,6,8,10,...\}$. В таблице 2.3 приведены данные разбиения процессорного поля на группы в случае, если длина экстраполяционной таблицы – четное число и используется вторая четная последовательность, формирующая сетки интегрирования.

Таблица 2.3

Разбиение процессоров на группы для P_2 и четной длины экстраполяционной таблицы, k

№ группы / k	1	2	3	4	5	6
6	**11**	**10**	**10**			
8	**15**	**14**	**14**	**14**		
10	**19**	**18**	**18**	**18**	**18**	
12	**23**	**22**	**22**	**22**	**22**	**22**

Очевидно, что имеется практически равномерное обеспечение загрузки процессорных групп, кроме первой, которая отличается от остальных вне зависимости от длины таблицы на единицу. В случае нечетной длины таблицы разбиение становится менее равномерным за счет непарной последней группы.

На рисунке 2.21 представлен параллельный алгоритм решения системы нелинейных обыкновенных дифференциальных уравнений на мультикомпьютере из p процессоров для макрооперационной схемы №2 с комбинационным способом разбиения процессоров и данных.

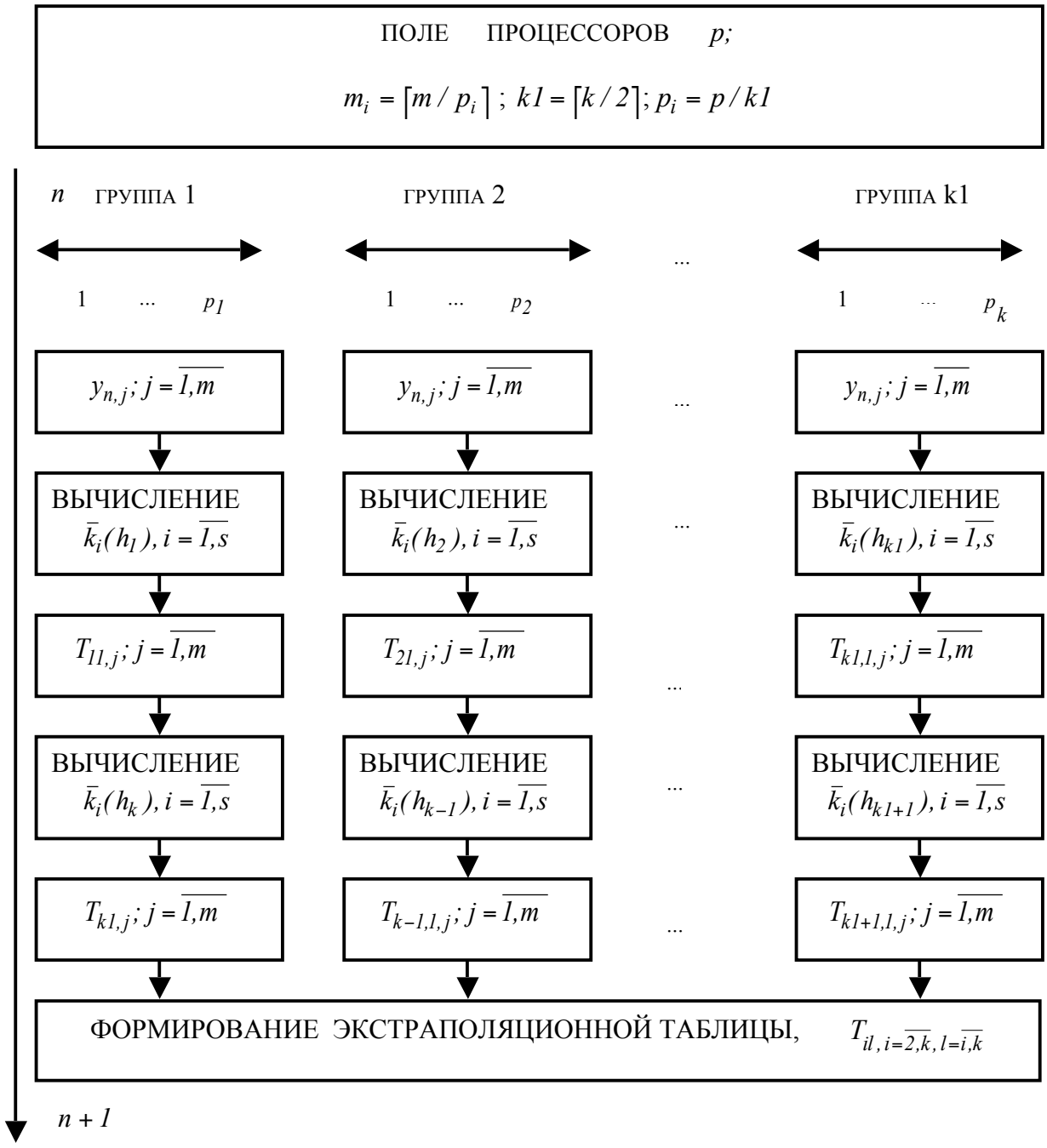


Рисунок 2.21 – Вычислительная схема параллельного алгоритма на основе технологии локальной экстраполяции, комбинационное разбиение

Число процессоров в каждой группе равно: $p_i = \lceil p/k1 \rceil = \lceil 2p/k \rceil = \lceil 4p/r \rceil$ и число компонент вектора решения для вычисления определенной аппроксимации решения: $m_i = \lceil m/p_i \rceil = \lceil mr/4p \rceil$. В этом случае время реализации параллельных вычислений аппроксимаций решения по схеме локальной экстраполяции с комбинационным способом разбиения равно:

$$T_{p,comp}^{34} = \max_{i=1}^{k/2}(n_i + n_{k-i+1}) \cdot T_{p_i}^{r^0=2} + T_{p,comp}^{34,ext-tab} =$$

$$= (r^2 - r) \cdot (m/2p) \cdot T_F + 2(r^2 - r)(m/p) \cdot t_{op} + 2(r-1) \cdot t_{op} + T_{p,comp}^{34,ext-tab}.$$

На вычисление экстраполяционной таблицы потребуется:

$$T_{p,comp}^{34,ext-tab} = [5(3r-8)/16p]mr \cdot t_{op}.$$

В целом время арифметических операций для третьей схемы равно:

$$T_{p,comp}^{34} = (r^2 - r) \cdot \frac{m}{2p} T_F + (17r^2 - 4.5r) \cdot \frac{m}{p} t_{op} + 2(r-1) \cdot t_{op}. \qquad (2.43)$$

Время обменов составляет:

$$T_{p,comm}^{34} = (n_1 + n_k) T_{all-to-all}(m_i, p_i) + T_{p,comm}^{34,ext-tab},$$

$$T_{p,comm}^{34} = (r-1) T_{all-to-all}\left(\frac{mr}{4p}, \frac{4p}{r}\right) + (1.5r - 4) T_{p-p}\left(\frac{mr}{4p}, \frac{4p}{r}\right) + T_{single-to-all}(m_1, p - p_1). \qquad (2.44)$$

Исследуем потенциальный параллелизм полученных вычислительных схем (рис. 2.22). Первая вычислительная схема вне зависимости от сложности правой части обладает лучшими характеристиками ускорения. Наихудшим потенциальным ускорением обладают варианты схемы 2 вне зависимости от сложности правой части.

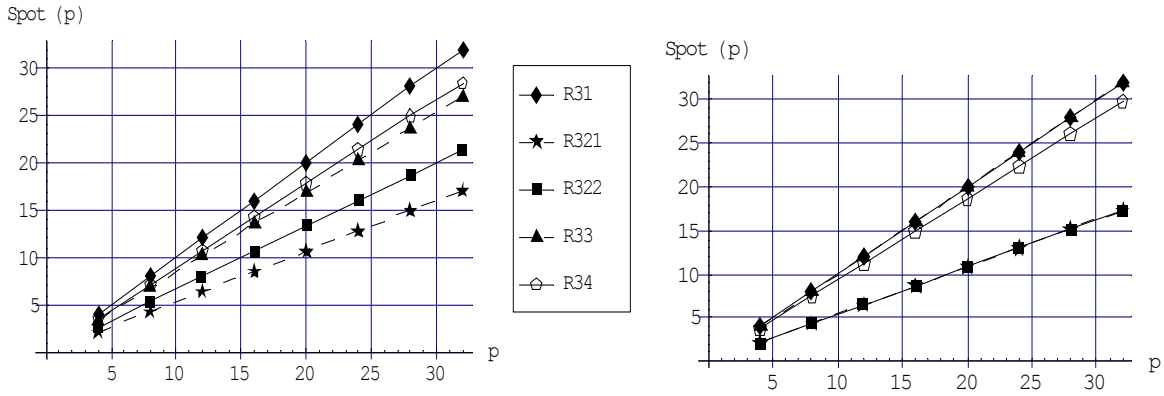


Рисунок 2.22 – Коэффициент потенциального ускорения методов от числа процессоров, 1) $T_f \sim t_{op}$; 2) $T_f >> t_{op}$

Оценим коммуникационную составляющую параллельных алгоритмов для различных типов параллельных ВС и топологий соединения процессоров.

На рисунке 2.23 приведен типичный график зависимости доли времени обменов к общим накладным расходам на параллелизм для первой вычислительной схемы. Вне зависимости от сложности правой части СОДУ и типа параллельной ВС топология гиперкуб также является наиболее эффективной для всех вычислительных схем, а топология кольцо требует больше всего времени на реализацию операций обмена.

Проведем сравнение вычислительных схем на основе реальных коэффициентов ускорения и эффективности для топологии гиперкуб. Равномерное разбиение процессоров на группы, несмотря на простоту реализации, обладает большими накладными расходами, в том числе и непроизводительными, имеет значительно худшие характеристики параллелизма и поэтому далее вычислительная схема №2 не рассматривается.

Рисунок 2.23 – Доля операций обмена алгоритма №1 к общим накладным расходам при различных топологиях

Первая схема, использующая системный параллелизм, имеет определенные преимущества, она может быть реализована на параллельном компьютере любого типа и обладает хорошей балансировкой загрузки процессоров. Из анализа рисунка 2.24 следует, что первая и третья схемы при сложных правых частях имеют хорошие показатели эффективности для систем, как с высокоскоростными каналами передачи информации, так и низкоскоростными.

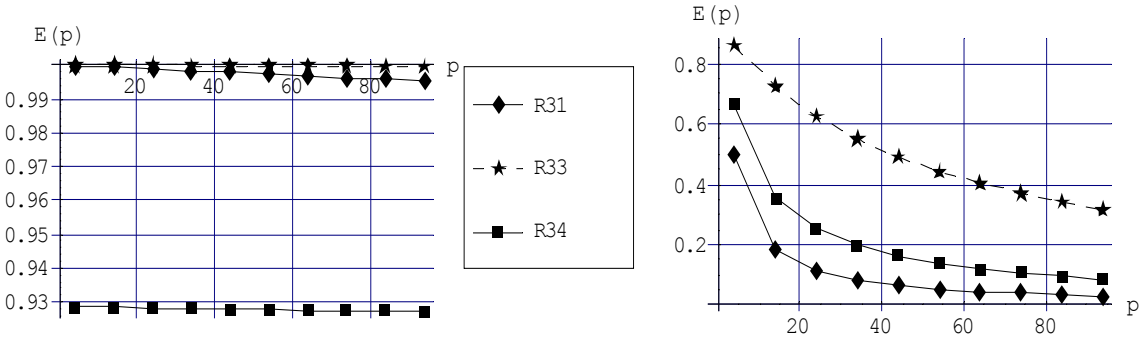


Рисунок 2.24 – Коэффициенты реальной эффективности методов от числа процессоров, $r = 8$, $m = 1000$; 1) $T_f >> t_{op}$; 2) $T_f \sim t_{op}$

Существенным параметром для этих схем является сложность правой части исходной СОДУ, обеспечивающая доминирование вычислений над обменами. Для несложных правых частей третья вычислительная схема, объединяющая параллелизм системы и экстраполяции, обладает явным преимуществом, и этот эффект с ростом числа процессоров увеличивается. Для SIMD-систем, как и ранее, может быть реализована только первая схема, которая имеет низкие показатели ускорения и эффективности.

2.4. Эффективность явных параллельных методов решения нелинейных СОДУ с альтернативными способами определения локальной погрешности

Сравнение вычислительной сложности трех различных параллельных алгоритмов решения нелинейной задачи Коши на основе явных численных схем с встроенными способами оценки локальной апостериорной погрешности корректно, если эти методы дают приближенное решение одного и того же порядка точности [52-54]. В явном виде порядок приближенного численного метода в оценках накладных расходов на параллелизм не присутствует, однако он может быть оценен через параметр s – число стадий явного метода типа Рунге-Кутты.

В таблице 2.4 указано минимальное число стадий s, необходимое для одношагового метода Рунге-Кутты на основе явной схемы порядка r. Очевидно, что для явных методов малых и средних порядков $r \le 6$ число стадий равно порядку или на единицу превышает его.

Для методов более высоких порядков эта разность больше единицы и возрастает с ростом порядка метода. Взаимосвязь между этими величинами для перечисленных методов определяется на основе барьеров Батчера [119-121].

Таблица 2.4

Связь порядка метода и числа стадий для известных ЯМРК

Порядок метода, r	$r \le 4$	5	6	7	8	10
Минимальное число стадий, s	r	6	7	9	11	17

В таблице 2.5 представлены известные вложенные методы типа Рунге-Кутты на основе явных схем разных порядков точности и соответствующее число стадий для каждого из них. Вычислительная сложность ВЯМРК зависит от порядков методов, составляющих вложенную пару $r(\hat{r})$, в свою очередь эти величины определяются числом используемых стадий.

Сравнение накладных расходов для реализации последовательных явных одношаговых методов с альтернативными способами оценки локальной апостериорной погрешности показывает, что вложенные методы обеспечивают наиболее экономный по временным характеристикам способ вычисления погрешности на шаге.

Таблица 2.5

Связь порядка и числа стадий для вложенных явных методов (ВЯМРК)

Вложенные явные методы Рунге-Кутты $r(\hat{r})$	Порядок метода, $r(\hat{r})$	Число стадий, s
Фельберга 4(5)	4(5)	6
Дормана-Принса 5(4)	5(4)	7
Фельберга 7(8)	7(8)	12
Дормана-Принса 8(7)	8(7)	13

Сравнение параллельных алгоритмов определения локальной погрешности на основе явных одношаговых схем производится по динамическим характеристикам, которые являются функциями машинно-зависимых констант и характеристик СОДУ (табл. 2.6-2.7).

От соотношения этих параметров и их комбинаций зависит эффективность полученных параллельных методов.

Таблица 2.6

Вычислительная составляющая ЯМРК с альтернативными способами оценки локальной апостериорной погрешности

Метод оценки локальной погрешности	Коэффициент при T_F	Коэффициент при t_{op}
Правило Рунге	$(3s-1)\frac{m}{p}$	$3\frac{m}{p}s^2+6\frac{m}{p}s+6s-6$
ВМРК	$(s+1)\frac{m}{p}$	$\frac{m}{p}s^2+6\frac{m}{p}s+2s+4\frac{m}{p}$
ЛЭР	$\left(\frac{s^2-2s+4}{2}\right)\cdot\frac{m}{p}$	$\left[\left(\frac{s^2-2s+4}{2}\right)+\left(\frac{21}{8}s^2-\frac{42}{8}s+8\right)\frac{m}{p}\right]$

Таблица 2.7

Коммуникационная составляющая ЯМРК с альтернативными способами оценки локальной апостериорной погрешности

Метод оценки локальной погрешности	Коэффициент при t_s	Коэффициент при t_w
Правило Рунге	$3s\log_2 p$	$3s\frac{m}{p}(p-1)$
ВМРК	$(s+1)\log_2 p$	$(s+1)\frac{m}{p}(p-1)$
ЛЭР	$\left(\frac{s^2-2s+4}{2}\right)\log_2 p$	$\left(\frac{s^2-2s+4}{2}\right)\frac{m}{p}(p-1)$

Наиболее эффективный последовательный метод – это ВМРК, приведем все характеристики к нему. Относительные коэффициенты эффективности вычисляются как:

$$T_1^{opt}=T_1^2;\quad E_i^{Re}=T_1^2/pT_p^i,$$

где T_1^{opt} – время выполнения наиболее оптимального последовательного алгоритма.

Анализ данных таблиц (2.6-2.7), теоретических и аналогичных экспериментальных зависимостей (рис. 2.26) для динамических характеристик трех способов определения локальной погрешности для явных одношаговых схем позволяют сделать выводы:

1) параллельные вложенные методы для явных схем наименее затратные для любых типов параллельных ВС и топологий соединения процессорных элементов;

2) практически единичная эффективность для ВМРК достигается для систем типа MIMD, топологии гиперкуб и сложных правых частей.

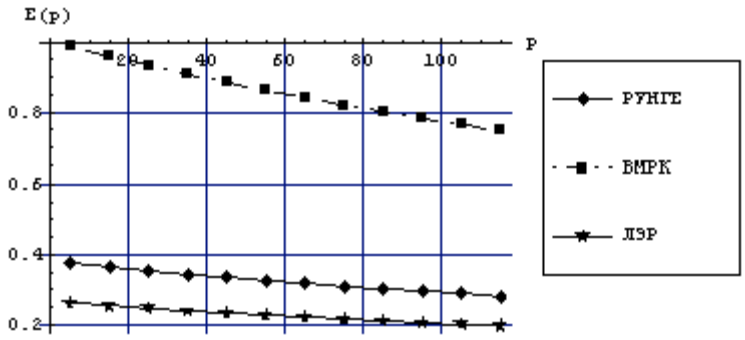


Рисунок 2.26 – Относительные коэффициенты эффективности параллельных алгоритмов ЯМРК

Несмотря на широкое применение, максимальный порядок известных методов, разработанных на основе явных разностных одношаговых схем, ограничен ($r \le 8$). Для интегрирования задач, требующих вычисления высокоточного решения ($10^{-15} - 10^{-20}$), не существует соответствующих вложенных схем. Таким образом, метод локальной экстраполяции Ричардсона имеет свою область приложения.

2.5. Исследование масштабируемости параллельных вложенных методов на основе явных одношаговых схем

Постановка вопроса об оценке эффективности распараллеливания при увеличении числа процессоров с сохранением объема вычислений не всегда является целью исследований, более того, некоторыми авторами [2] считается спорной. Действительно, рост производительности системы в целом обусловлен

сбалансированностью вычислительной работы и обменов на ее фоне. Невыполнение этого условия – одна из причин ухудшения качества параллелизма с увеличением размерности процессорного поля. Поэтому выводы следует делать на основе достижения приемлемой эффективности при распараллеливании на число процессоров, фактически необходимое для решения задачи в заданные временные сроки. Другими словами, речь идет об оценке масштабируемости параллельной системы, состоящей из алгоритма в комбинации с архитектурой, на которой он реализован.

В данном подразделе исследуется **масштабируемость параллельного алгоритма** вложенных методов, как наиболее эффективного способа оценки локальной погрешности для явных одношаговых схем. Математическим аппаратом, позволяющим решить поставленную задачу, является **теория изоэффективного анализа** [11-14]. Для построения **функции изоэффективности** необходимо определить общие накладные расходы на параллельный алгоритм:

$$T_o^2 = (s+1)\left(p\log_2 p\right)\cdot t_s + m(s+1)(p-1)\cdot t_w + 2s(p-1)\cdot t_{op}. \quad (2.43)$$

Тогда основное соотношение изоэффективного анализа для рассматриваемого алгоритма принимает вид:

$$T_1^2 = K\cdot T_o^2 \quad \Rightarrow$$

$$T_1^2 = (s+1)mT_F + (s^2m + 6sm + 2s + 4m)t_{op} = K\cdot T_o^2 \quad \Rightarrow \quad (2.44)$$

$$T_1^2 = K[(s+1)\left(p\log_2 p\right)\cdot t_s + m(s+1)(p-1)\cdot t_w + 2s(p-1)\cdot t_{op}]. \quad (2.45)$$

Используя (2.44-2.45), определим функцию изоэффективности, то есть найдем зависимость $m = f_E(p)$, на основе которой можно вычислить, как необходимо увеличивать размерность задачи при увеличении числа процессоров для поддержания постоянной эффективности. То есть, выразим размерность СОДУ, определяющую сложность исходной задачи, как функцию от следующих параметров:

1) количество процессорных элементов;
2) временные константы алгоритма и ВС;
3) порядок и число стадий явного метода;
4) коэффициент масштабируемости $m = f_E(p;T_F,t_{op},t_s,t_w;s;K)$.

Напомним, что $K = E/1-E$.

$$m = \frac{K[(s+1)(p\log_2 p)\cdot t_s + 2s(p-1)\cdot t_{op}] - 2s\cdot t_{op}}{(s+1)\cdot T_F + (s^2+6s+4)\cdot t_{op} - K(s+1)(p-1)\cdot t_w}. \qquad (2.46)$$

Проанализируем это соотношение с учетом характеристик алгоритма и параллельной вычислительной системы. Заметим, что из $Dop = p_{max} = m$ следует, что число процессоров всегда должно быть меньше либо равно размерности исходной СОДУ во избежание непроизводительных расходов (простоя оборудования). Вне зависимости от соотношения любых параметров, размерность СОДУ должна быть положительным числом. Исходя из (2.46) очевидно, что числитель дроби есть число положительное, так как в самом худшем случае при $p = s = 2$ это требование выполняется. Знаменатель (2.47) должен быть строго больше нуля, поэтому в качестве верхней границы для числа процессоров имеем следующее выражение:

$$p < \frac{(s+1)\cdot T_F + (s^2+6s+4)\cdot t_{op} + K(s+1)\cdot t_w}{K(s+1)\cdot t_w}. \qquad (2.47)$$

Проанализируем соотношение (2.46) с выведенными ограничениями. По приведенной в главе 1 классификации рассмотренный алгоритм является квазилинейным относительно числа процессоров, то есть обладает высокой масштабируемостью. Рисунок 2.27 представляет коэффициент ускорения при 4 различных значениях размерности задачи для асинхронных параллельных систем с низкоскоростными коммуникационными сетями: высокая латентность и небольшая скорость передачи данных. Как и ожидалось, для малой размерности $m \le O(100t_{op})$ ускорение достигает пика уже при небольшом числе процессоров: $p \approx 100$. За этой точкой увеличение размера процессорного поля существенного влияния на ускорение не оказывает. С другой стороны, ускорение для больших размерностей задачи близко к линейному.

Для MIMD систем с быстрыми каналами связи (рис. 2.28) масштабируемость алгоритма повышается, то есть ускорение с ростом числа процессоров становится практически линейным, особенно для задач большой размерности. Пик ускорения достигается при гораздо большем числе процессоров, величина максимального возможного ускорения возрастает.

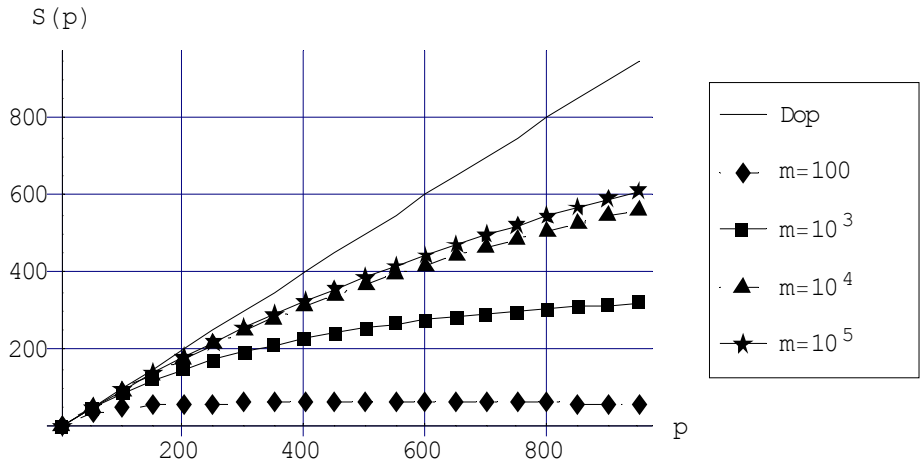


Рисунок 2.27 – Графики зависимости коэффициента ускорения ВЯМРК от числа процессоров для различных размерностей задачи в MIMD-системе с низкоскоростными коммуникационными средами топологии гиперкуб для вложенных методов $r = 4$ при $T_F >> t_{op}$

Число процессоров, при котором происходит насыщение ускорения, прямо пропорционально сложности исходной задачи. Эта тенденция проявляется для параллельных систем любой архитектуры, эффективности коммуникационных сетей и топологии соединения процессоров в них.

Рассмотрим влияние на величину функции изоэффективности нескольких параметров: коэффициента масштабируемости, а, следовательно, и приемлемой эффективности, порядка исследуемого метода, сложности правой части, при варьировании числа процессоров и характеристик коммутационной сети (рис. 2.28-2.30).

Зная функцию изоэффекивности, можно определить, какой размерности задачу необходимо решать на имеющейся параллельной ВС, чтобы эффективность использования оборудования достигала определенного заданного значения. Так, при размерности процессорного поля $p = 400$ и низкоскоростных коммутационных сетях для достижения эффективности 0.9 нужна задача размерности не менее $m \approx 4500$, в то же время для $E = 0.7$ этот параметр приблизительно в пять раз меньше ($m \approx 872$). Достижение тех же характеристик для быстрых каналов передачи данных требует решения задач практически в десять раз меньшей размерности.

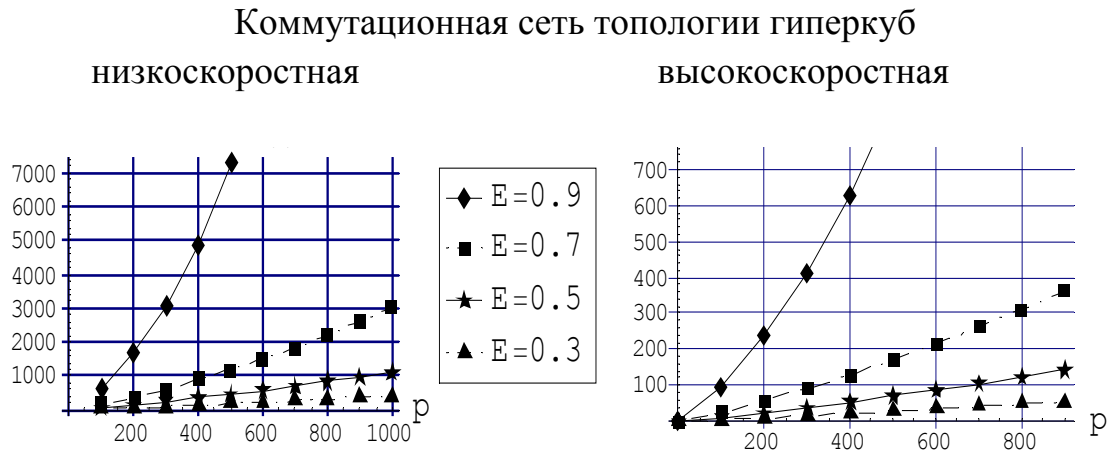


Рисунок 2.28 – Функция изоэффективности ВМРК для MIMD-систем при различных значениях коэффициента эффективности

На рисунке 2.29 приведены графики зависимости функции изоэффективности от порядка наиболее известных вложенных методов и различных коммуникационных сред, причем коэффициент масштабируемости равен $K = 9$, что соответствует $E = 0.9$.

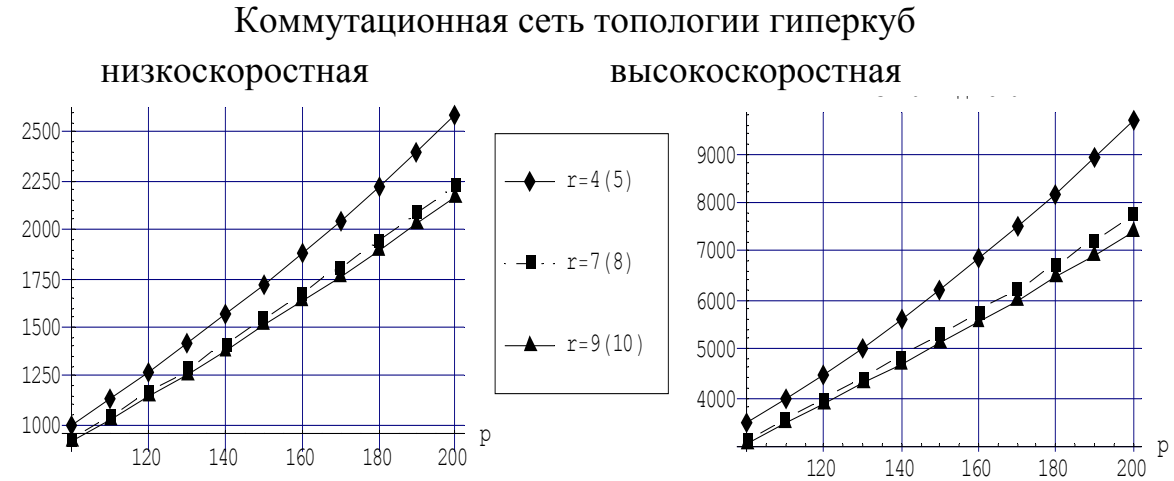


Рисунок 2.29 – Функция изоэффективности для MIMD-систем и вложенных явных методов различных порядков точности при $E = 0.9$, 1) Мерсона 4(5); 2) Фельберга 7(8); 3) Дормана-Принса 9(10)

Очевидно, что чем меньше порядок метода, тем большей размерности задачу необходимо решать для достижения одной и той же эффективности при одинаковом числе используемых процессоров.

Аналогично, с ростом сложности правой части СОДУ размерность решаемой задачи уменьшается.

Коммутационная сеть топологии гиперкуб

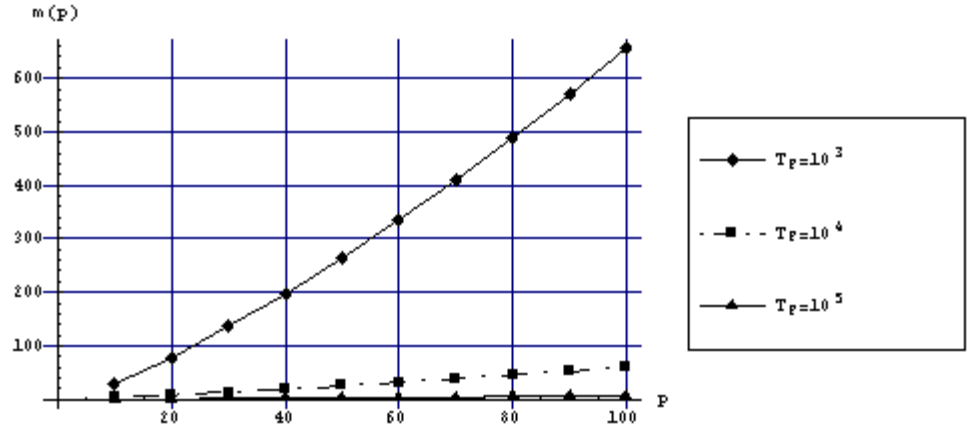


Рисунок 2.30 – Функция изоэффективности вложенных явных методов решения СОДУ с различной сложностью правой части для MIMD-систем с низкоскоростными коммутационными сетями

Применение изоэффективного анализа позволяет сделать выводы о качестве полученного параллельного алгоритма на основе единого аналитического выражения, сократив тем самым многочисленные эксперименты при различных комбинациях параметров от которых зависят характеристики параллелизма.

2.6. Выводы

Исследована эффективность альтернативных способов определения локальной апостериорной погрешности при решении нелинейной задачи Коши для СОДУ на основе явных одношаговых схем r -го порядка точности. Показано, что при параллельной реализации методы вложенных форм требуют минимальных вычислительных затрат: практически в три раза по сравнению с правилом дублирования шага и в r раз по сравнению с технологией локальной экстраполяции.

Предложен подход, основанный на сочетании иерархической функциональной поэтапной декомпозиционной методики и математического аппарата графов влияния, позволяющий эффективно распараллеливать последовательные алгоритмы.

Построен параллельный алгоритм решения нелинейной задачи Коши с контролем шаговой погрешности на основе технологии локальной экстраполяции Ричардсона. Установлено, что наименее трудоемкими являются численные схемы с симметричными опорными методами малых порядков точности в сочетании с четными последовательностями чисел для генерации сеток интегрирования. Оценена эффективность полученных параллельных алгоритмов на основе равномерного, пропорционального и комбинационного способа распределения данных по процессорам.

Разработаны вычислительные схемы отображения полученных параллельных алгоритмов на структуры параллельных вычислительных систем различных архитектур (SIMD, MIMD и кластеров) с распределенной памятью и следующими топологиями межпроцессорных связей: линейка/кольцо, решетка/тор, гиперкуб.

Исследована эффективность полученных топологических отображений в зависимости от типа архитектуры параллельной ВС, различных топологий и коммуникационных констант, а также размерности начальной задачи, сложности ее правой части и уникальных характеристик метода.

Для вложенных методов построена функция изоэффективности, определяющая аналитическую зависимость размерности задачи от числа процессоров для достижения необходимой эффективности и на ее основе определена степень масштабируемости алгоритма: $O(\log_2 p)$.

Определены приоритетные области применения разработанных параллельных методов решения динамических задач в сочетании с параллельной архитектурой:

– для методов средних и малых порядков наиболее эффективными с точки зрения вычислительных и коммуникационных затрат являются параллельные вложенные методы на основе явных одношаговых схем с топологией гиперкуб;

– преимущества явных схем, основанных на технологии локальной экстраполяции, проявляются при поиске высокоточных решений $r > 10$ и для сложных правых частей ($T_F > 100 t_{op}$).

ГЛАВА 3

ПАРАЛЛЕЛЬНЫЕ НЕЯВНЫЕ ОДНОШАГОВЫЕ МЕТОДЫ ЧИСЛЕННОГО РЕШЕНИЯ ЖЕСТКИХ ЗАДАЧ КОШИ

Исследование методов решения динамических задач с сосредоточенными параметрами показало, что параллельные свойства таких алгоритмов во многом определяются видом лежащей в их основе численной схемы. Наиболее изученными и наименее трудоемкими являются явные методы, однако присущие этим схемам недостатки, одним из которых является условная устойчивость, существенно ограничивают область их применения. В этой связи значительный интерес представляют неявные схемы, которые, несмотря на большую вычислительную сложность, не имеют альтернативы среди одношаговых методов при решении жестких задач Коши [94-95].

Рассматривается численное решение задачи Коши, ассоциируемое с решением СОДУ первого порядка с известными начальными условиями:

$$\begin{cases} \dfrac{d\bar{y}(x)}{dx} = \bar{f}(x, \bar{y}(x)), \\ \bar{y}(x_0) = \bar{y}_0, \end{cases} \tag{3.1}$$

где правая часть системы есть в общем случае нелинейная функция, задающая отображение $F = \bar{f} : R \times R^m \to R^m$:

$$F = \begin{pmatrix} f_1(x, y_1, y_2, ..., y_m) \\ f_2(x, y_1, y_2, ..., y_m) \\ ... \\ f_m(x, y_1, y_2, ..., y_m) \end{pmatrix}; \quad \bar{y} = \begin{pmatrix} y_1 \\ y_2 \\ ... \\ y_m \end{pmatrix}; \quad \bar{y}_0 = \begin{pmatrix} y_{0,1} \\ y_{0,2} \\ ... \\ y_{0,m} \end{pmatrix}.$$

Глава 3 посвящена разработке, обоснованию и оценке эффективности параллельных численных алгоритмов решения жестких задач Коши со встроенными способами оценки локальной погрешности на основе неявных одношаговых методов.

Класс неявных численных схем представлен двумя типами методов: блочные одношаговые многоточечные методы и ПНМРК.

3.1. Параллельные одношаговые блочные методы интегрирования ОДУ со встроенными способами оценки локальной погрешности

Блочные многоточечные методы решения задачи Коши особенно актуальны, поскольку хорошо согласуются с архитектурой параллельных ВС и не требуют вычисления значений в промежуточных точках, что значительно повышает эффективность счета. Данные методы обладают **высокой устойчивостью** и являются **изначально параллельными** [107-108], так как позволяют получать решение одновременно в нескольких точках сетки интегрирования.

Множество точек равномерной сетки Ω_h : $\left\{ x_j \right\}, j = \overline{1,M}$ разбивается на N блоков. Каждый блок содержит k точек и при этом $N \leq M$. Очевидно, что суммарное число точек по всем блокам равно: $M = k \times N$. Предполагается, что в пределах блока все точки равноудалены друг от друга, следовательно, справедливо следующее соотношение:

$$x_{n,i} = x_{n,o} + ih, i = \overline{1,k}, \tag{3.2}$$

где i – номер точки в блоке, $i = \overline{1,k}$;

n – номер блока, $n = \overline{1,N}$;

$x_{n,i}$ – точка с номером i, принадлежащая блоку n;

$x_{n,0}$ – начальная точка n-го блока;

$x_{n,k}$ – конечная точка n-го блока.

Множество точек n-го блока из k точек обозначается $T_n^{(k)}$. При этом имеет место равенство: $x_{n,k} = x_{n+1,0}$ (рис. 3.1).

Пусть $y_{n,0}$ – приближенное значение решения задачи Коши в точке $x_{n,0}$ – начальной точке обрабатываемого блока. Уравнения одношаговых блочных разностных методов в применении к ОДУ для блока из k точек могут быть записаны в следующем виде:

$$y_{n,i} = y_{n,0} + ih\left[b_i F_{n,0} + \sum_{j=1}^{k} a_{i,j} F_{n,j}\right]; i = \overline{1,k}; n = \overline{1,N}. \qquad (3.3)$$

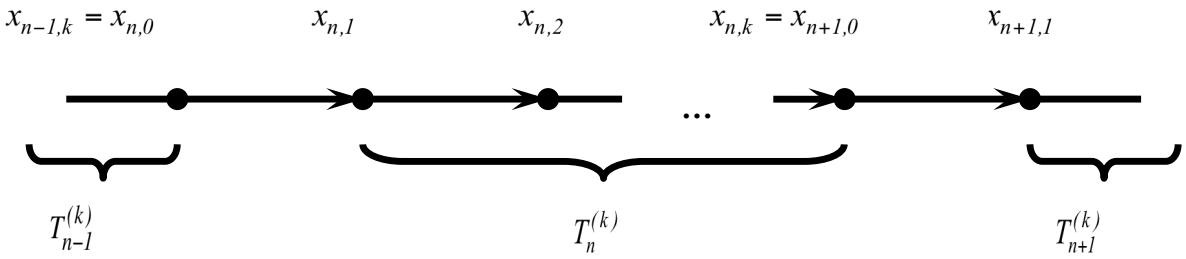


Рисунок 3.1 – Схема разбиения на блоки для одношагового k-точечного метода

Разложением в ряд Тейлора входящих в невязку функций можно показать, что **одношаговый *k*-точечный блочный метод имеет наивысший порядок аппроксимации, равный $k+1$, следовательно, локальная ошибка в узлах блока имеет порядок** $O(h^{k+2})$ [109-113]. Блочные параллельные методы относятся к классу неявных, поэтому для вычисления приближенных значений решения задачи Коши необходимо разрешить систему нелинейных уравнений.

Одним из способов получения решения является метод **простой функциональной итерации:**

$$\begin{cases} y_{n,i,0} = y_{n,0} + ihF_{n,0}, \; i = \overline{1,k}, \; n = 1,2,...,N, \\ y_{n,i,l+1} = y_{n,0} + ih(b_i F_{n,0} + \sum_{j=1}^{k} a_{i,j} F_{n,j,l}), \quad l = \overline{0,L-1}, \end{cases} \qquad (3.4)$$

где n – номер блока, $n = 1,2,...,N$; i – номер точки блока, $i = \overline{1,k}$;

l – номер текущей итерации $l = \overline{0,L-1}$;

L – максимальное число ненулевых итераций.

В отличие от рассмотренных явных методов решения СОДУ, реализация альтернативных способов оценки апостериорной локальной погрешности на основе блочных методов связана с рядом особенностей:

– нет соответствующих последовательных аналогов, следовательно, требуется разработать и обосновать метод оценки локальной погрешности;

– варьирование шага интегрирования возможно только после вычисления всех значений в k узлах текущего n-го блока,

– при условии неудовлетворительной оценки локальной погрешности практически все вычисления для точек блока окажутся напрасными (некоторые обращения к правой части могут быть использованы вновь).

3.1.1. Эффективность параллельной реализации правила дублирования шага в блочных одношаговых методах интегрирования ОДУ

Пусть решение задачи Коши для ОДУ выполняется на основе k-точечного одношагового блочного метода. При реализации **правила дублирования шага** необходимо произвести вычисления по одной и той же группе формул вида (3.3):

$$\begin{cases} y_{n,i}^{(1)} = y_{n,0}^{(1)} + ih \cdot \left[b_i f(x_{n,0}; y_{n,0}^{(1)}) + \sum_{j=1}^{k} a_{i,j} f(x_{n,j}; y_{n,j}^{(1)}) \right], n = \overline{1,N},\ i = \overline{1,k} \\ y_{n,i}^{(2)} = y_{n,0}^{(2)} + i\frac{h}{2} \cdot \left[b_i f(x_{n,0}; y_{n,0}^{(2)}) + \sum_{j=1}^{k} a_{i,j} f(x_{n,j}; y_{n,j}^{(2)}) \right], n = \overline{1,N1}, i = \overline{1,k} \end{cases} \quad (3.5)$$

на двух различных равномерных сетках:

1) $\Omega_h = \{ x_j \}, j = \overline{1,M}$ с шагом h в N блоках;

2) $\Omega_{h/2} = \{ x_j \}, j = \overline{1,M1}$ с половинным шагом $h/2$ в $N1$ блоках.

На рисунке 3.2 приведена схема вычислений при использовании правила Рунге для одношагового блочного k-точечного метода. Как и в предыдущей главе, аппроксимация решения с одинарным шагом обозначается $y_{n,i}^{(1)}$, а с половинным, соответственно: $y_{n,i}^{(2)}$. Точки n-го блока сетки Ω_h составляют множество $T_{n,k}^{(1)}$, а сетки $\Omega_{h/2}$ – $T_{n,k}^{(2)}$.

Поскольку количество точек в блоке для обеих сеток равно k, то для одного и того же интервала интегрирования число блоков второй сетки ровно в два раза больше, чем первой.

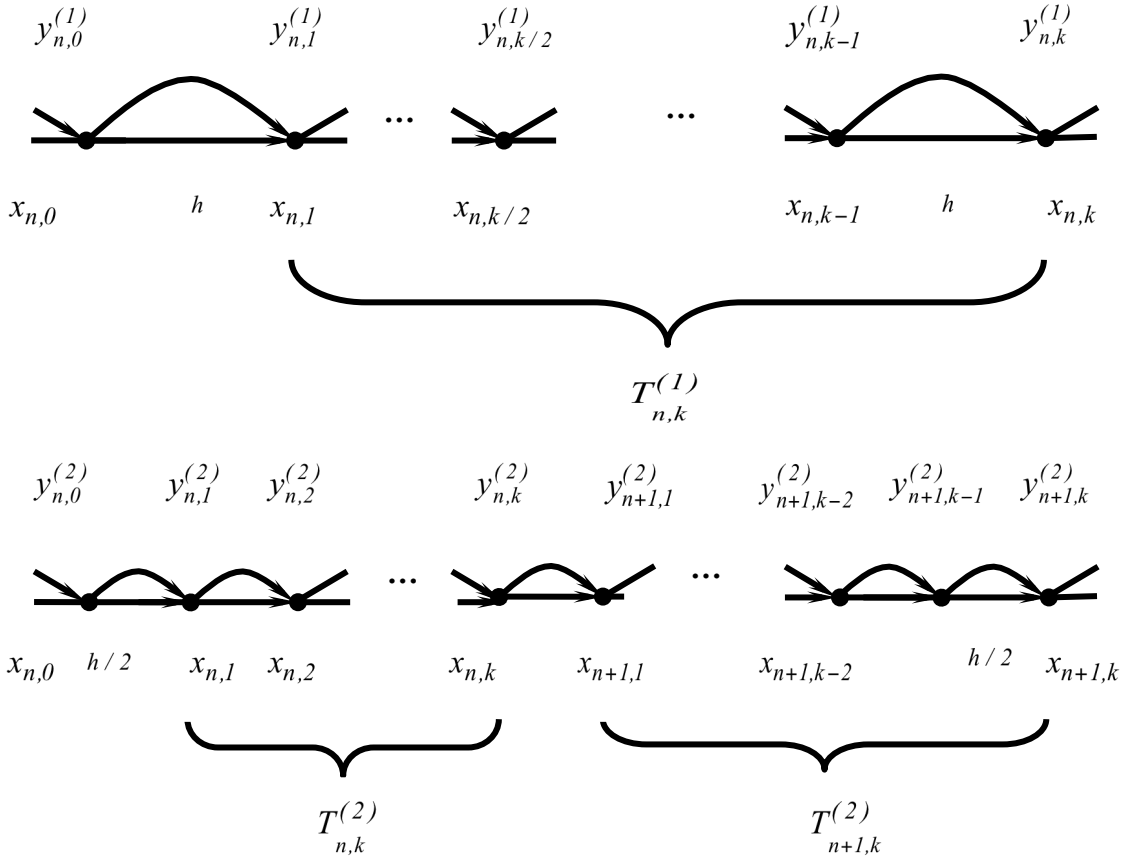


Рисунок 3.2 – Схема применения правила дублирования шага для одношагового k –точечного блочного параллельного метода

Основой счета при интегрировании является сетка Ω_h, при этом каждый узел с четным номером в блоках сетки $\Omega_{h/2}$ используется для вычисления оценки локальной погрешности на этом шаге. Более того, в качестве решения в этих узлах часто принимается аппроксимация, полученная с половинным шагом либо экстраполированная, как наиболее точная. Узлы сетки $\Omega_{h/2}$ с нечетными номерами используются только как вспомогательные. Рассматриваемые методы являются неявными, поэтому **применение правила Рунге к блочным одношаговым методам требует разрешения трех различных систем нелинейных алгебраических уравнений размерности k.**

Общее время последовательной реализации блочных методов с правилом Рунге T_1^{11} состоит из суммы времени вычисления решения с одинарным шагом в блоке n плюс времена решения с половинным шагом в n-том и $(n+1)$-вом блоках. Поскольку для получения каждого из трех решений реализуется свой итерационный процесс, введем следующие обозначения.

1) $L1-$ предельное количество итераций для нахождения аппроксимации решения $y_{n,i}^{(1)}$;

2) $L2-$ предельное количество итераций для решения $y_{n,i}^{(2)}$;

3) $L3-$ предельное количество итераций для решения $y_{n+1,i}^{(2)}$.

Тогда, соответственно, текущее число итераций, обеспечивающее достаточную для каждой из данных задач точность, обозначим: $li,\ li \le Li,\ i = \overline{1,3}$. Время вычислений последовательного алгоритма блочных методов плюс правило Рунге включает времена на определение нулевых и последующих итераций решения:

$$T_1^{11} = \underbrace{6kt_{mul} + 3kt_{ad} + 2T_F}_{y_{n,i;0}^{(1)} + y_{n,i;0}^{(2)} + y_{n+1,i;0}^{(2)}} +$$

$$+(l1+l2+l3)\cdot \underbrace{[(k^2+2k)\cdot t_{mul} + (k^2+2k)\cdot t_{ad} + k\cdot T_F]}_{y_{n,i;l1}^{(1)} + y_{n,i;l2}^{(2)} + y_{n+1,i;l3}^{(2)}},$$

$$T_1^{11} = [k(l1+l2+l3)+2]\cdot T_F + [(l1+l2+l3)(2k^2+4k)+9k]\cdot t_{op} \qquad (3.6)$$

где T_F- время вычисления правой части ОДУ.

Вычислительная схема параллельного блочного k-точечного метода с контролем локальной апостериорной погрешности по правилу Рунге приведена на рисунке 3.3. Здесь каждый процессор вычисляет решение в одном узле сетки, то есть максимальная степень параллелизма вычислительной схемы составляет: $Dop = k$.

Для каждой из трех задач последовательно выполняются вычисления нулевой и последующих итераций.

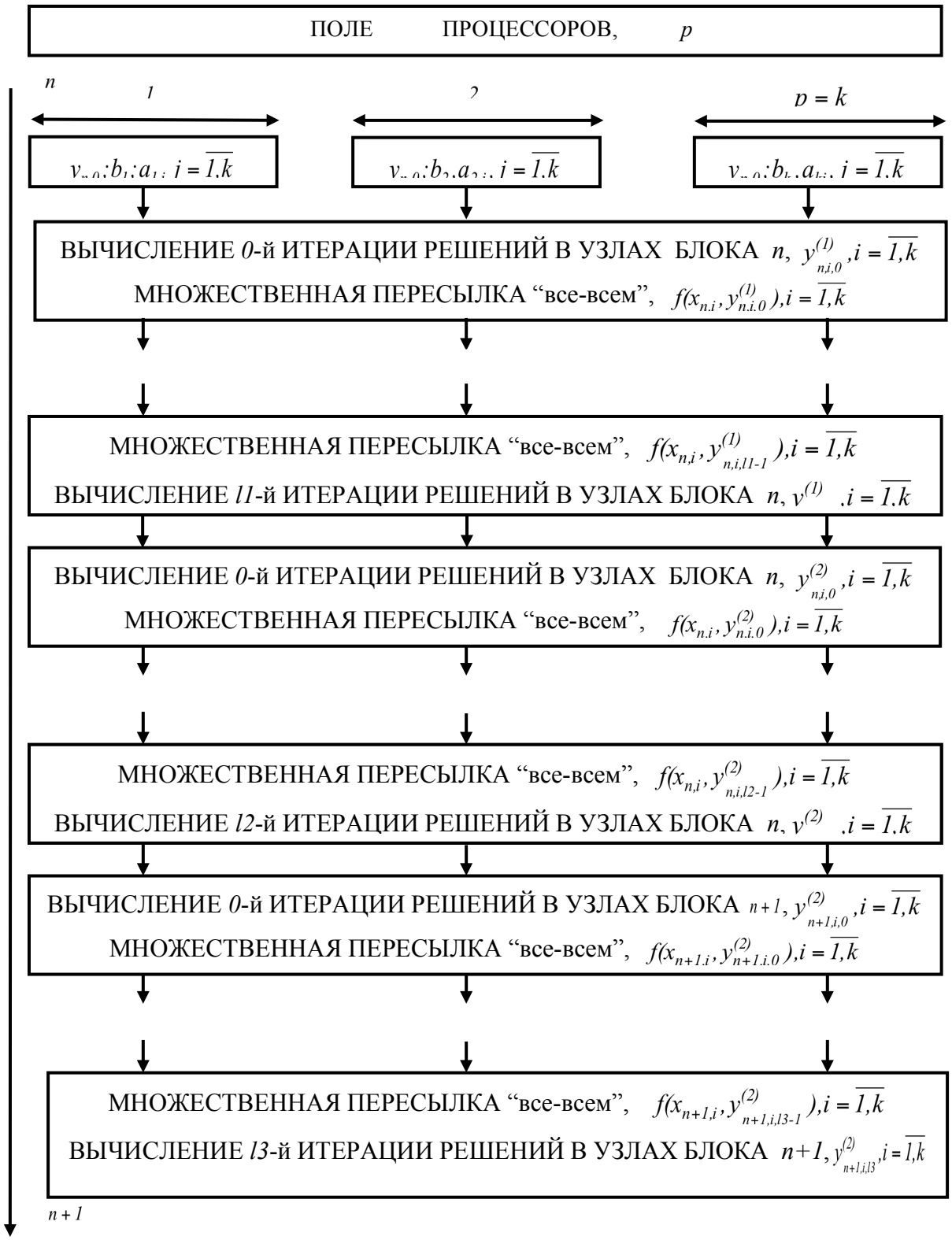


Рисунок 3.3 – Вычислительная схема параллельного алгоритма блочного метода с контролем локальной погрешности по правилу Рунге

При этом нулевая итерация состоит из следующих шагов:

1) вычисление нулевого приближения параллельно в каждом узле нового блока по первой формуле системы (3.4);

2) вычисление правой части ОДУ от нулевого приближения;

3) множественный обмен вычисленными значениями правой части по типу “все-всем”.

Затем li раз выполняется аналогичная группа операций для последующих итераций. Перечень операций включает:

1) вычисление очередного приближения в каждом узле нового блока по второй формуле (3.4), базовой операцией является умножение матрицы A на вектор значений правых частей ОДУ;

2) вычисление правой части ОДУ от полученного приближения и множественный обмен значениями правой части по типу “все-всем”.

Таким образом, время параллельных вычислений по схеме (3.5) с локальной точностью $O(h^{k+li+2})$ в узлах соответствующих сеток составляет:

$$T_{p,comp}^{11} = (\sum_{i=1}^{3} li + 2) \cdot T_F + [\sum_{i=1}^{3} li \cdot (k+3) + 3] \cdot t_{mul} + [\sum_{i=1}^{3} li \cdot (k+2) + 1] \cdot t_{ad}.$$

Для RISC-архитектур, соответственно:

$$T_{p,comp}^{11} = (\sum_{i=1}^{3} li + 2) \cdot T_F + [\sum_{i=1}^{3} li \cdot (2k+5) + 4] \cdot t_{op}. \qquad (3.7)$$

Реализация обменов требует выполнение групповых операций пересылок по типу “все-всем”:

$$T_{p,comp}^{11} = \left(\sum_{i=1}^{3} li + 2 \right) \cdot T_{all-to-all}(p). \qquad (3.8)$$

Потенциальные характеристики параллелизма предложенного метода можно оценить по числу обращений к правой части ОДУ. При сложной правой части ОДУ, то есть в случае, если выполняется соотношение $T_F \gg t_{op}$:

$$S_{pot}^{11} \approx \left[\left(k \sum_{i=1}^{3} li + 2 \right) \cdot T_F \right] / \left[\left(\sum_{i=1}^{3} li + 2 \right) \cdot T_F \right] \approx k, \; E_{pot}^{11} \approx \frac{S_{pot}^{11}}{p} \approx 1,$$

имеет место практически линейное ускорение и единичная эффективность. Такие же потенциальные характеристики могут быть получены и в случае, когда правая часть по времени вычисления соизмерима со временем выполнения одной операции с плавающей точкой. Реальные динамические характеристики полученного параллельного алгоритма существенно зависят не только от параметров задачи и алгоритма, но и от эффективности организации межпроцессорных связей и определяются соотношениями:

$$S^{11} = \frac{(k\cdot\sum_{i=1}^{3} li + 2)\cdot T_F + [\sum_{i=1}^{3} li\cdot(2k^2+4k)+5k]\cdot t_{op}}{(\sum_{i=1}^{3} li + 2)\cdot T_F + [\sum_{i=1}^{3} li\cdot(2k+5)+4]\cdot t_{op} + \left(\sum_{i=1}^{3} li + 2\right)\cdot T_{all-to-all}(p)}, \quad (3.9)$$

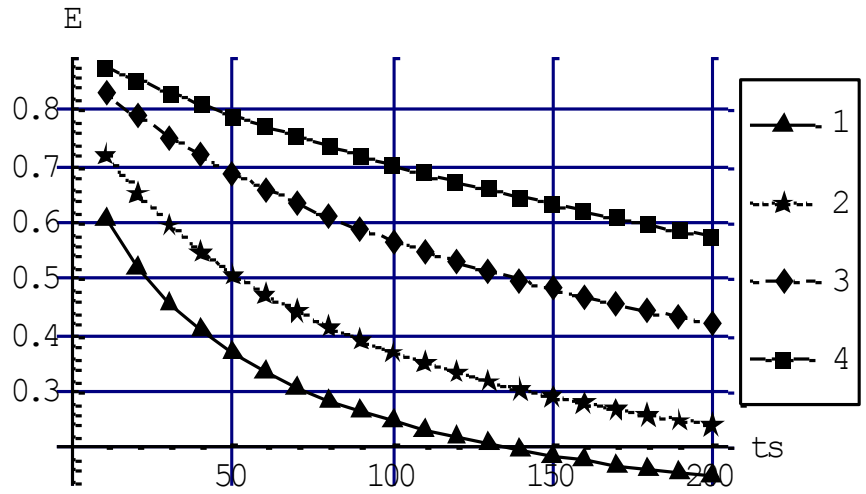
$$E^{11} = \frac{(k\cdot\sum_{i=1}^{3} li + 2)\cdot T_F + [\sum_{i=1}^{3} li\cdot(2k^2+4k)+5k]\cdot t_{op}}{k(\sum_{i=1}^{3} li + 2)\cdot T_F + [\sum_{i=1}^{3} li\cdot(2k+5)+4]\cdot t_{op} + \left(\sum_{i=1}^{3} li + 2\right)\cdot T_{all-to-all}(p)}. \quad (3.10)$$

Анализ теоретического выполнения и вычислительный эксперимент показывают, что для выполнения групповых обменных операций в предложенном алгоритме эффективными являются топологии гиперкуб и тор (рис. 3.4), худший вариант соединения процессоров – кольцо.

Рисунок 3.4 – Доля обменов к общему времени выполнения блочного метода с правилом Рунге для различных топологий

Кроме топологии соединения, на величину времени межпроцессорных обменов и динамические характеристики параллелизма существенное влияние оказывают тип параллельной архитектуры и определяемые им машинно-зависимые константы обмена, такие, как латентность и время передачи слова (рис. 3.5).

Заметим, что с ростом величины латентности коммуникационной среды время на реализацию обменов увеличивается, а ускорение и эффективность алгоритма уменьшаются $\uparrow t_s \Rightarrow \uparrow T_{p,comm} \Rightarrow \downarrow S \Rightarrow \downarrow E.$ Аналогичные зависимости связывают динамические характеристики и время передачи одного слова: $\uparrow t_w \Rightarrow \uparrow T_{p,comm} \Rightarrow \downarrow S \Rightarrow \downarrow E.$ Однако величина латентности является более существенным параметром, степень влияния скорости передачи данных, как правило, возрастает при увеличении объемов передаваемых данных.



$1 - T_F = 100t_{op}; 2 - T_F = 200t_{op}; 3 - T_F = 500t_{op}; 4 - T_F = 1000t_{op}$

Рисунок 3.5 – Коэффициент эффективности блочного метода с контролем локальной погрешности по правилу Рунге

Для рассматриваемого алгоритма вычисления являются однородными, и, как результат, коэффициент эффективности для SIMD-архитектур при прочих равных условиях выше, чем для MIMD-систем за счет меньшего значения латентности сети (рис. 3.6).

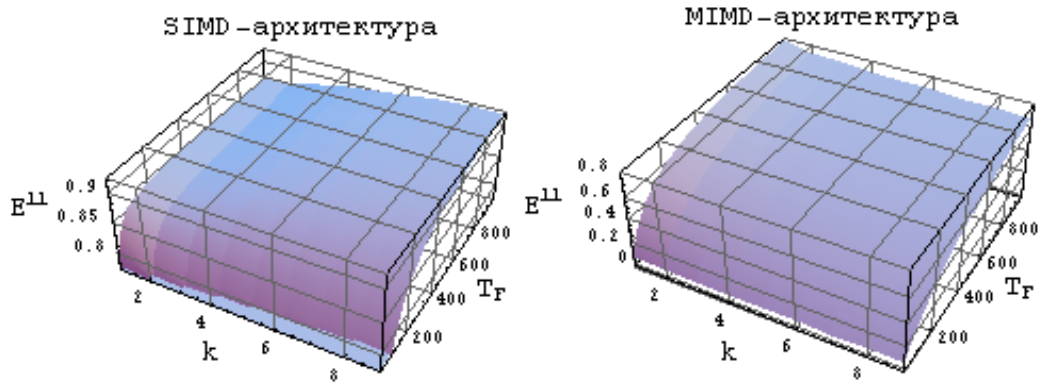
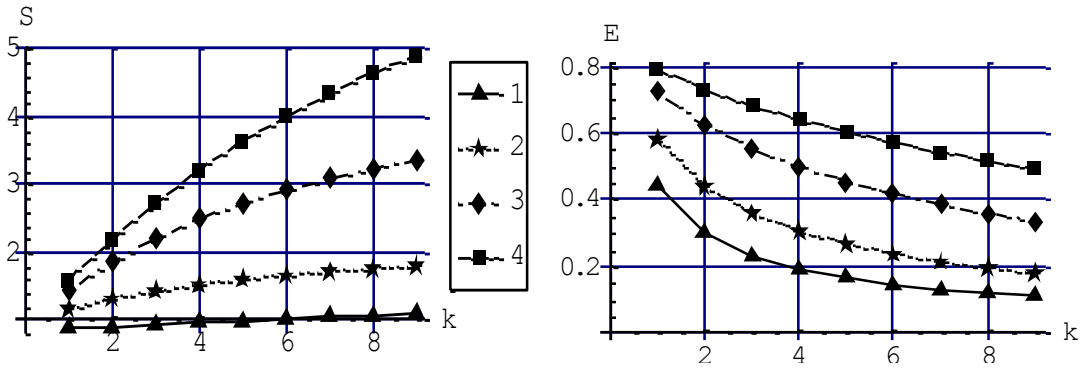


Рисунок 3.6 – Коэффициент эффективности блочного метода с контролем локальной погрешности по правилу Рунге

Из временных характеристик алгоритма и исходной задачи качество параллелизма наиболее чувствительно к необходимому объему вычислений приходящихся на реализацию правой части (3.1) и количеству точек в одном блоке. Зависимости реальных коэффициентов ускорения и эффективности параллельного процесса контроля локальной погрешности на основе правила Рунге от числа точек блока при росте сложности правых частей ОДУ представлены с помощью графиков рис. 3.7.



$1-T_F=100t_{op};2-T_F=200t_{op};3-T_F=500t_{op};4-T_F=1000t_{op}$

Рисунок 3.7 – Коэффициенты ускорения и эффективности блочного метода с правилом Рунге, $p=k$

Очевидно, чем сложнее правая часть ОДУ, тем лучше характеристики параллелизма: $\uparrow T_F \Rightarrow \uparrow S \Rightarrow \uparrow E$ и, одновременно, чем

больше размерность блока, совпадающая с числом процессоров, тем больше ускорение и меньше эффективность рассматриваемого метода: $\uparrow k \Rightarrow \uparrow p \Rightarrow \uparrow S \Rightarrow \downarrow E$.

Таким образом, наилучшие характеристики параллелизма при решении нелинейной задачи Коши для одного уравнения блочными методами с контролем локальной погрешности по правилу Рунге достигаются для любых параллельных архитектур, большой размерности задачи, сложной правой части и высокоскоростных сетей передачи информации.

3.1.2. Разработка и анализ эффективности параллельных вложенных блочных одношаговых методов решения задачи Коши

Идея вложенных форм, предложенная для оценки локальной погрешности численного решения обыкновенных дифференциальных уравнений методами типа Рунге-Кутты, может быть использована и **для одношаговых блочных многоточечных методов**. В данном подразделе приведено обоснование применения этого способа оценки погрешности для параллельных вложенных блочных методов решения нелинейной задачи Коши для ОДУ на основе двух различных подходов:

1) **комбинация независимых формул** разных порядков точности;

2) **комбинация специально подобранных формул** разных порядков точности.

Первый подход заключается в применении двух различных независимых блочных методов смежных порядков точности $r(\hat{r}), \hat{r} = r \pm 1$ на одной и той же сетке интегрирования Ω_h:

$$\begin{cases} y_{n,i} = y_{n,0} + ih\left[b_i f(x_{n,0}; y_{n,0}) + \sum\limits_{j=1}^{k} a_{i,j} f(x_{n,j}; y_{n,j})\right]; i = \overline{1,k}; n = \overline{1,N}, \\ \hat{y}_{n,i} = \hat{y}_{n,0} + ih\left[\hat{b}_i f(x_{n,0}; \hat{y}_{n,0}) + \sum\limits_{j=1}^{\hat{k}} \hat{a}_{i,j} f(x_{n,j}; \hat{y}_{n,j})\right]; i = \overline{1,\hat{k}}; n = \overline{1,\hat{N}}. \end{cases} \quad (3.11)$$

При этом первый метод определяет аппроксимацию решения на основе k -точечного одношагового метода, а второй $\hat{k}$ -точечного, также одношагового метода. Второе приближенное решение в

совпадающих узлах блоков $T_{n,i}^{(k)}$ и $T_{n,j}^{(\hat{k})}$ сетки Ω_h используется для оценки апостериорной локальной погрешности (рис. 3.8). Пусть для определенности основным является блочный метод низшего порядка точности, то есть $\hat{k} = k + 1$. Локальная погрешность приближенного решения одношаговым *k*-точечным методом в *i*-том узле блока для одного уравнения определяется следующей формулой: $y(x_{n,i}) - y_{n,i}^{r} = O(h^{k+2}); i = \overline{1,k}$, и для (*k+1*)-точечного метода в том же узле равна: $y(x_{n,i}) - y_{n,i}^{\hat{r}} = O(h^{k+3}); i = \overline{1,\hat{k}}$. Из полученных соотношений следует, что оценка локальной погрешности формулы меньшего порядка точности, *k*-точечного метода, приближенно может быть вычислена, как: $y_{n,i}^{r} - y_{n,i}^{\hat{r}}; i = \overline{1,k}$.

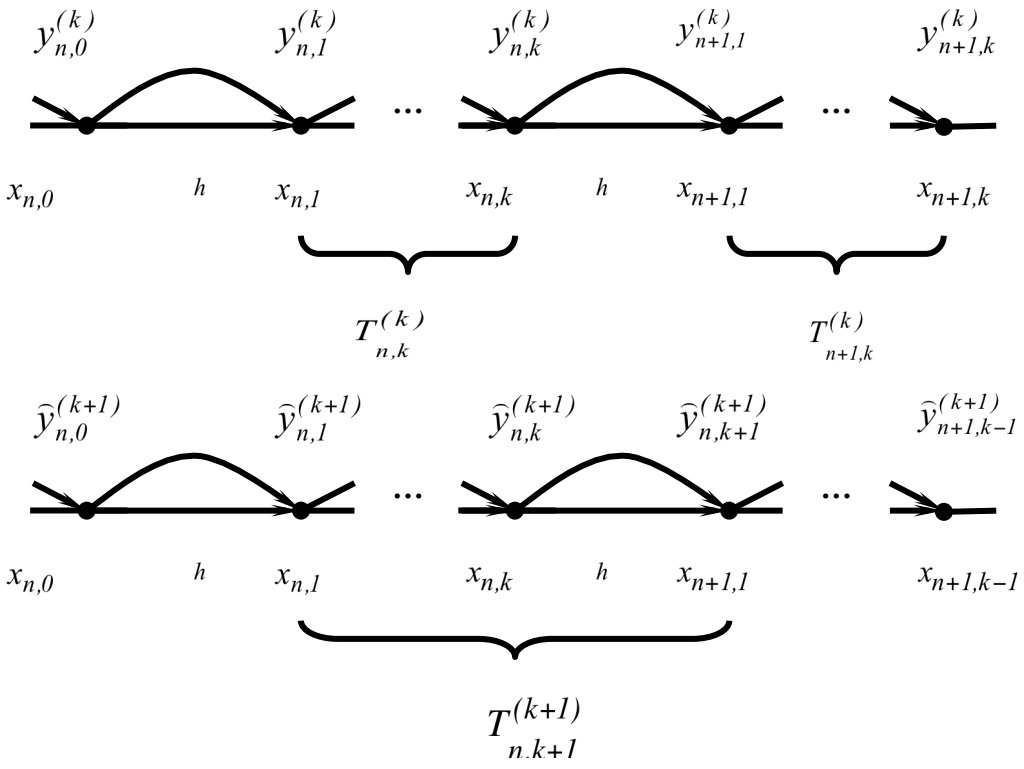


Рисунок 3.8 – Схема вычислений для вложенных одношаговых $k(\hat{k})$-точечных блочных методов, $\hat{k} = (k+1)$

Такой подход к оценке локальной погрешности является более эффективным по сравнению с правилом Рунге, поскольку при достаточной простоте позволяет уменьшить вычислительные затраты.

Так, для применения правила дублирования шага необходимо решить три СНАУ размерности k, а для применения вложенного метода две системы: одну той же размерности, другую размерности $(k+1)$. Определим время последовательной реализации рассматриваемого метода в предположении вычислительной и емкостной равномощности однопроцессорной и многопроцессорной ВС. Пусть $T_1^{12}(k)$ – время последовательных вычислений по k-точечному методу, а $T_1^{12}(k+1)$ – аналогичное время, полученное по $(k+1)$-точечному методу. Общее время последовательного выполнения схемы (3.12) равно: $T_1^{12} = T_1^{12}(k) + T_1^{12}(k+1)$. Каждый из блочных методов требует применения итерационного процесса для вычисления решения соответствующих систем нелинейных уравнений.

Пусть L – предельное количество итераций при реализации k-точечного метода, l – текущее число итераций, обеспечивающее достаточную для данной задачи точность, $l \le L$. Аналогично, для вложенного метода более высокого порядка имеем следующие обозначения: $\hat{L}$ и $\hat{l}$, $\hat{l} \le \hat{L}$. Тогда,

$$T_1^{12}(k) = \underbrace{k[t_{mul} + t_{ad}] + T_F}_{y_{n,i,0}} + l \cdot \underbrace{[[k^2 + 2k] \cdot (t_{mul} + t_{ad}) + k \cdot T_F] + kt_{mul}}_{y_{n,i,l}},$$

$$T_1^{12}(\hat{k}) = \underbrace{\hat{k}[t_{mul} + t_{ad}] + T_F}_{\hat{y}_{n,i,0}} + \hat{l} \cdot \underbrace{\{[\hat{k}^2 + 2\hat{k}](t_{mul} + t_{ad}) + \hat{k}T_F] + \hat{k}t_{mul}}_{\hat{y}_{n,i,\hat{l}}}, \quad (3.12)$$

при $t_{mul} = t_{ad} = t_{op}$ и $\hat{k} = k+1$ время выполнения последовательного алгоритма:

$$T_1^{12} = (lk + \hat{l}k + \hat{l} + 2) \cdot T_F + [2l \cdot [(k^2 + 2k) + 2\hat{l}(k^2 + 4k + 3) + 6k + 3] \cdot t_{op}. \quad (3.13)$$

Для обеспечения более высокого порядка точности, необходимо, чтобы $\hat{l} = l + 1$:

$$T_1^{12} = (2lk + k + l + 3) \cdot T_F + [(4l + 2) \cdot k^2 + 12lk + 6l + 14k + 9] \cdot t_{op}. \quad (3.14)$$

Вычислительная схема параллельного алгоритма №1 на основе формул (3.11) представлена на рис. 3.9. Поскольку $\hat{k} = k+1$, то число процессоров равно: $p = \hat{k}$ (при реализации k-точечного метода один процессор простаивает).

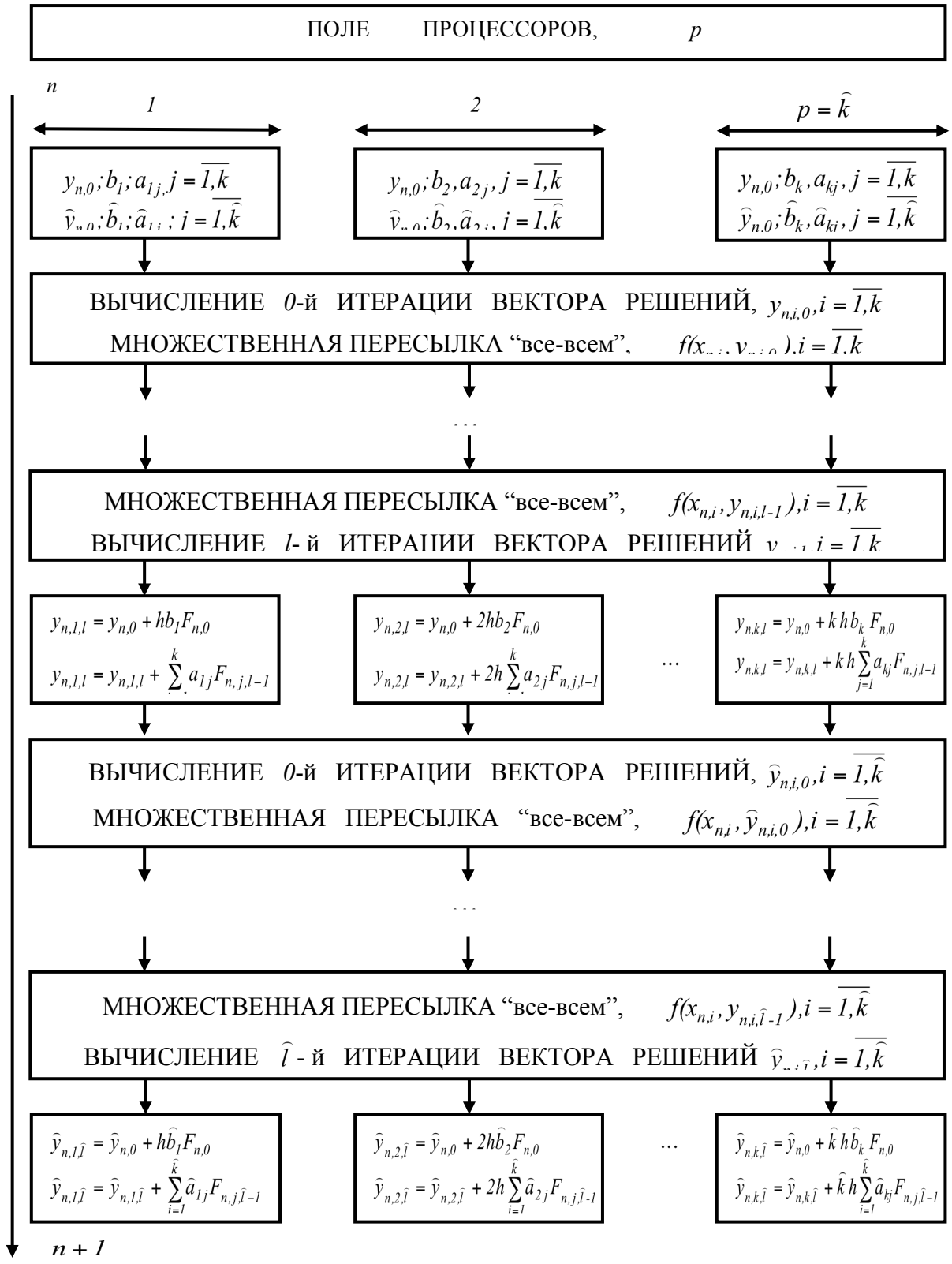


Рисунок 3.9 – Вычислительная схема параллельного алгоритма №1 вложенного одношагового $k(\hat{k})$-точечного блочного метода для ОДУ

Каждый процессорный элемент вычисляет решение в одном узле сетки, то есть для такой вычислительной схемы $Dop = \hat{k} = k + 1$. Первоначально определяется аппроксимация решения на основе k-точечного метода, затем – $(k+1)$-точечного. Для каждой из двух решаемых задач последовательно выполняются вычисления нулевой и последующих итераций. Тогда время на вычисления и обменные операции для параллельного вложенного блочного метода №1 составляет ($\hat{l} = l + 1$ и $t_{op} = t_{mul} = t_{ad}$):

$$T_{p,comp}^{12} = (2l+3)\cdot T_F + (4lk + 10l + 2k + 12)\cdot t_{op} \qquad (3.15)$$

$$T_{p,comm}^{12} = (l + \hat{l} + 2)\cdot T_{all-to-all}(p) = (2l+3)\cdot T_{all-to-all}(k+1). \qquad (3.16)$$

Потенциальный параллелизм полученного метода при сложной правой части ОДУ составляет:

$$S_{pot}^{12} = [k\cdot(l+\hat{l}) + \hat{l} + 2]\cdot T_F / [(l+\hat{l}+2)\cdot T_F] \approx k,$$

$$E_{pot}^{12} = S_{pot}^{12} / p \approx 1,$$

аналогичные характеристики имеем и при $T_F \approx t_{op}$.

Для вложенных методов влияние топологии соединения процессоров и параметров коммуникационной среды на динамические характеристики параллелизма аналогично, как и для блочных методов с правилом Рунге. Второй подход к разработке блочных вложенных методов предполагает использование **идеи последовательного повышения порядка точности** [116-117], и имеет своей целью сокращение вычислительных затрат на основе комбинации специально подобранных формул разных порядков.

Пусть решение задачи Коши для ОДУ на некотором интервале интегрирования выполняется на основе k – точечного одношагового блочного метода. Покажем, что в качестве оценки локальной погрешности в каждом узле текущего n-го блока может быть принята следующая величина:

$$d_{n,i} = \hat{y}_{n,i} - y_{n,i} = y_{n,i,\hat{l}} - y_{n,i,\breve{l}}, i = \overline{1,k}, \hat{l} = \breve{l} \pm 1, \qquad (3.17)$$

где $y_{n,i,\hat{l}}$ и $y_{n,i,\breve{l}}$ – $\hat{l}$ и $\breve{l}$-тая аппроксимации, полученные при решении (3.3) итерационным методом (3.4).

Введем следующие обозначения:

1) $y_{n,i}^{[v]}, i = \overline{1,k}$ – значение численного решения в i-том узле $x_{n,i}$ n-го блока, вычисленное с локальной погрешностью $O(h^v)$,

2) $F_{n,i}^{[v]} = f\left(x_{n,i}, y_{n,i}^{[v]}\right)$ – обращение к правой части исходного дифференциального уравнения, вычисленное в точке $\left(x_{n,i}, y_{n,i}^{[v]}\right)$.

Пусть приближенное значение решения $y_{n,0}$ в начальной точке n-го блока вычислено с некоторой локальной ошибкой $O(h^v)$, обеспечивающей достаточную для поставленной задачи точность. Тогда, обращение к правой части исходного дифференциального уравнения может быть вычислено с такой же погрешностью: $F_{n,0}^{[v]}$. Выполнив вычисления для нулевой итерации по первой формуле итерационного метода, получим $y_{n,i,0}^{[2]} = y_{n,0}^{[v]} + ihF_{n,i,0}^{[v]}, i = \overline{1,k}$, так как локальная ошибка формулы Эйлера имеет порядок $O(h^2)$. Каждое последующее вычисление по второй формуле (3.4) дает повышение порядка точности для метода простых итераций на 1:

$$l = 0: \; y_{n,i,1}^{[3]} = y_{n,0}^{[v]} + ih \cdot \left(b_i F_{n,0}^{[v]} + \sum_{j=1}^{k} a_{i,j} F_{n,j,0}^{[2]} \right), i = \overline{1,k},$$

$$l = 1: \; y_{n,i,2}^{[4]} = y_{n,0}^{[v]} + ih \cdot \left(b_i F_{n,0}^{[v]} + \sum_{j=1}^{k} a_{i,j} F_{n,j,1}^{[3]} \right), i = \overline{1,k},$$

…,

так как $F_{n,j,0}^{[v]} = f(x_{n,j}, y_{n,j,0}^{[v]}), j = \overline{1,k}$. Этот процесс не может быть продолжен бесконечно, при $l = k - 1$, получим результаты, соответствующие предельным локальным точностям приближенных формул (3.4). Поскольку разностные схемы, соответствующие блочным одношаговым k-точечным методам, аппроксимируют дифференциальное уравнение (3.1) с порядком $O(h^{k+1})$, то последующие итерации повышения порядка точности результатов не дают:

$$l = k - 1: y_{n,i,k}^{[k+2]} = y_{n,0}^{[v]} + ih \cdot \left(b_i F_{n,0}^{[v]} + \sum_{j=1}^{k} a_{i,j} F_{n,j,k-1}^{[k+1]} \right), i = \overline{1,k}. \qquad (3.18)$$

Таким образом, для оценки локальной погрешности могут быть выбраны две произвольные, подряд идущие, аппроксимации решения $y_{n,i,\hat{l}}$ и $y_{n,i,\check{l}}$ с учетом соображений точности и ограничений: $\check{l} < \hat{l} \le L - 1 = k - 1$. Как правило, $\check{l} = l,\ \hat{l} = l + 1$. При описанном вложенном блочном k-точечном методе, все дополнительные вычислительные расходы на определение локальной погрешности сводятся к дополнительному итерированию (в рамках предельных значений) при решении СНАУ размерности k.

Расчетные формулы для одного, n-го, блока вложенного многоточечного алгоритма №2 имеют вид:

$$\begin{cases} y_{n,i,0} = y_{n,0} + ihF_{n,0},\ i = \overline{1,k}, \\ y_{n,i,l+1} = y_{n,0} + ih(b_i F_{n,0} + \sum\limits_{j=1}^{k} a_{i,j} F_{n,j,l}),\ \ l = \overline{0,\check{l} - 1}, \\ \hat{y}_{n,i,\hat{l}} = y_{n,0} + ih(b_i F_{n,0} + \sum\limits_{j=1}^{k} a_{i,j} F_{n,j,\check{l}}),\ \ \check{l} = l, \hat{l} = l + 1. \end{cases} \tag{3.19}$$

Оценим время выполнения последовательного алгоритма, определяемого вычислительной схемой (3.19):

$$T_1^{13} = \underbrace{kt_{mul} + kt_{ad} + T_F}_{y_{n,i,0}} + \underbrace{\hat{l} \cdot [(k^2 + 2k) \cdot t_{mul} + (k^2 + 2k) \cdot t_{ad} + k \cdot T_f] + k \cdot t_{mul}}_{y_{n,i,\check{l}} + \hat{y}_{n,i,\hat{l}}}.$$

При $t_{mul} = t_{ad} = t_{op}$ имеем:

$$T_1^{13} = (\hat{l}k + 1) \cdot T_F + [2\hat{l}(k^2 + 2k) + 3k] \cdot t_{op}. \tag{3.20}$$

Вычислительная схема параллельного вложенного блочного метода №2 приведена на рис. 3.10, как и ранее показаны вычисления в рамках одного n-го блока. Для простоты изложения рассматривается случай, когда количество процессоров совпадает с размерностью блока и за каждым узлом блока закреплен один процессор. Для одного дифференциального уравнения внутренний параллелизм исчерпывается независимыми вычислениями каждой точки блока и ограничен количеством точек в блоке. Итерационный процесс решения полученных СНАУ является сугубо последовательным. После каждого из $l = \overline{0,\hat{l}}$ шагов необходим обмен данными по типу “все-всем”.

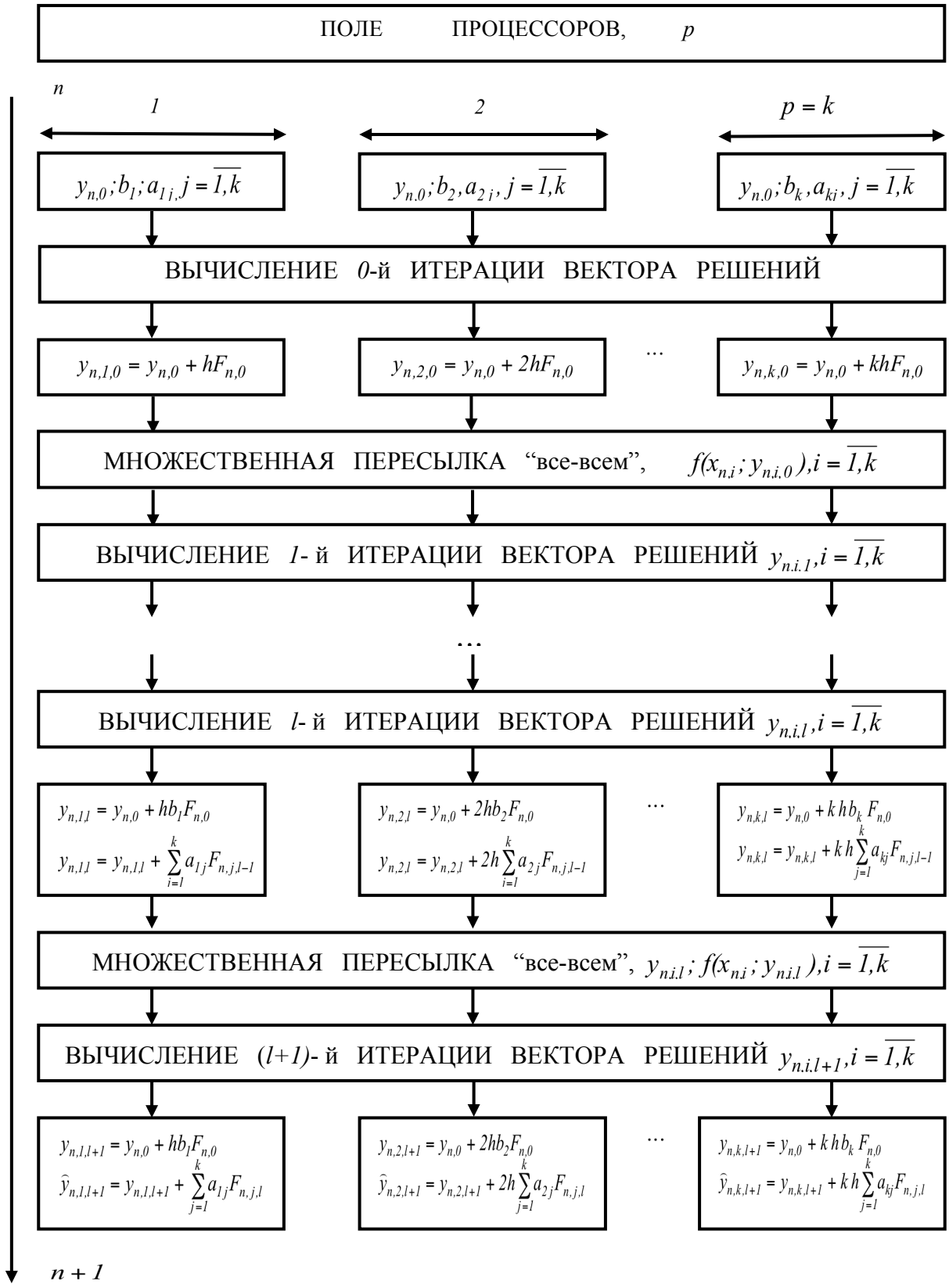


Рисунок 3.10 – Вычислительная схема №3 параллельного алгоритма вложенного одношагового k-точечного блочного метода для ОДУ

При реализации алгоритма на k процессорах можно одновременно вычислять значения $F_{n,i,l}$, а затем также одновременно получить по формулам значения $y_{n,i,l}$ для каждого фиксированного l. Время параллельного вычисления приближенных значений решения с точностью $O(h^{\hat{l}+2})$ для всех узлов блока при выполнении $\hat{l}$ шагов итерационного процесса равно:

$$T_{p,comp}^{13} = (\hat{l}+1)\cdot T_F + [\hat{l}(k+2)+2]\cdot t_{mul} + [\hat{l}(k+2)+1]\cdot t_{ad},$$

$$T_{p,comp}^{13} = (\hat{l}+1)\cdot T_F + [2\hat{l}k + 4\hat{l} + 3]\cdot t_{op}. \qquad (3.21)$$

Время обменов при этом с использованием введенных коммуникационных примитивов будет равно:

$$T_{p,comm}^{13} = (\hat{l}+1)\cdot T_{all-to-all}(1,p) = (k+1)\cdot T_{all-to-all}(1,p). \qquad (3.22)$$

Потенциально оба вложенных блочных метода обладают высокой степенью внутреннего параллелизма: практически линейным ускорением и единичной эффективностью. Это следует из приведенных ниже соотношений для коэффициентов потенциального ускорения и эффективности алгоритма в случае, если время обращения к правой части ОДУ доминирует над другими вычислениями:

$$S_{pot}^{13} = T_1^{13} / T_p^{13} \approx [(\hat{l}k+1)T_F] / [(\hat{l}+1)T_F] \approx k = p,\ E_{pot}^{13} = S_{pot}^{13} / p \approx 1.$$

Аналогичный результат имеем для тривиальной правой части. По-прежнему наилучшей топологией для рассмотренных методов является топология гиперкуб. В целом влияние параметров коммуникационной среды на динамические характеристики параллелизма этих двух алгоритмов совпадают. Проведем сравнение двух различных вложенных блочных методов (рис. 3.11-3.13) с целью определить минимум накладных расходов при оценке локальной погрешности решения на основе идеи вложенности.

Поскольку скорость сходимости итерационного процесса при решении систем нелинейных алгебраических уравнений существенно зависит от свойств конкретной системы, а, следовательно, коэффициентов самого блочного метода, для общности рассуждений проведем сравнение при предельно допустимом числе итераций.

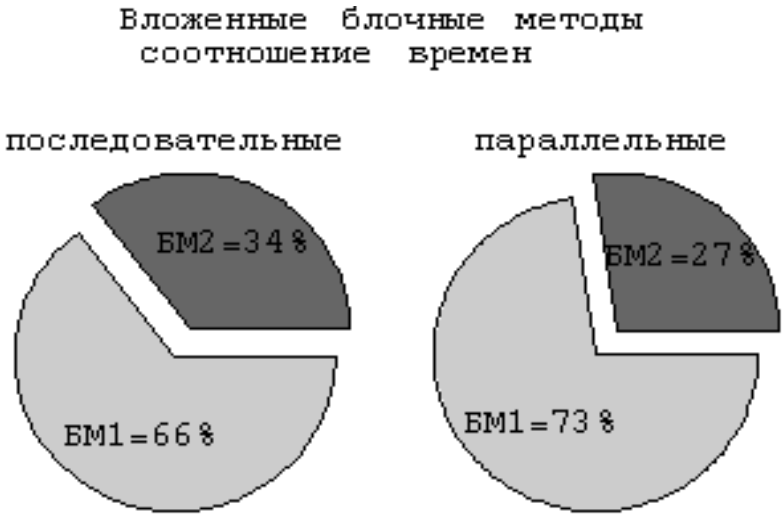


Рисунок 3.11 – Сравнение временной сложности двух вложенных блочных методов для ОДУ в последовательной и параллельной реализациях

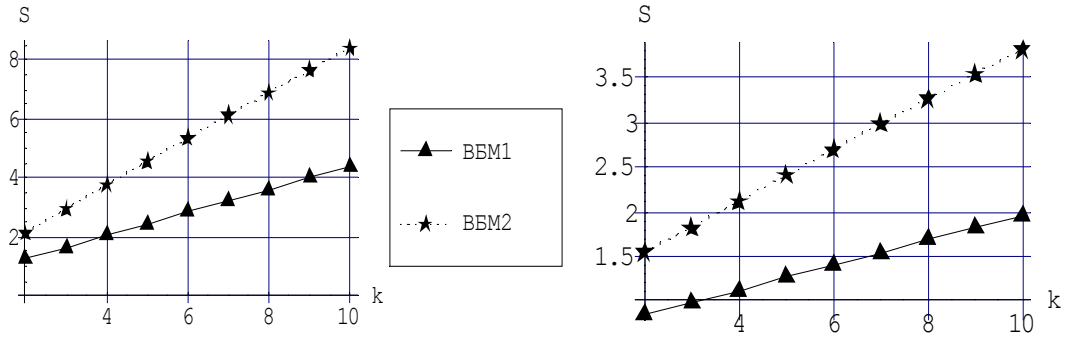
Анализ теоретического выполнения и проведенный эксперимент позволяют сделать следующие выводы:

1) время выполнения первого вложенного блочного алгоритма больше времени выполнения второго, как в последовательной, так и параллельной реализациях: $T_1^{12} > T_1^{13}$ и $T_p^{12} > T_p^{13}$, причем для параллельных алгоритмов эта разница больше, чем для последовательных;

2) коэффициенты ускорения и эффективности первого вложенного блочного метода меньше соответствующих коэффициентов второго метода при различных значениях параметров задачи, метода и параллельной системы: $S^{12} < S^{13}$, $E^{12} < E^{13}$ (рис. 3.12-3.13).

Заметим, что влияние фактора “сложность правой части ОДУ” является во многом определяющим для качества параллелизма. При одинаковых значениях коммуникационных констант и параметров, задающих уникальный блочный метод, коэффициенты ускорения и эффективности уменьшаются практически в два раза при переходе от доминирующих к тривиальным правым частям. Степень влияния коммуникационных констант традиционна для рассматриваемых методов и операций обмена.

Увеличение числа точек в блоке приводит к росту коэффициента ускорения $\uparrow k \Rightarrow \uparrow S$ и к уменьшению коэффициента эффективности $\uparrow k \Rightarrow \downarrow E$.



$T_F >> t_{op}$ $T_F \approx t_{op}$

Рисунок 3.12 – Коэффициент ускорения двух вложенных блочных методов от числа точек блока и сложности правой части ОДУ

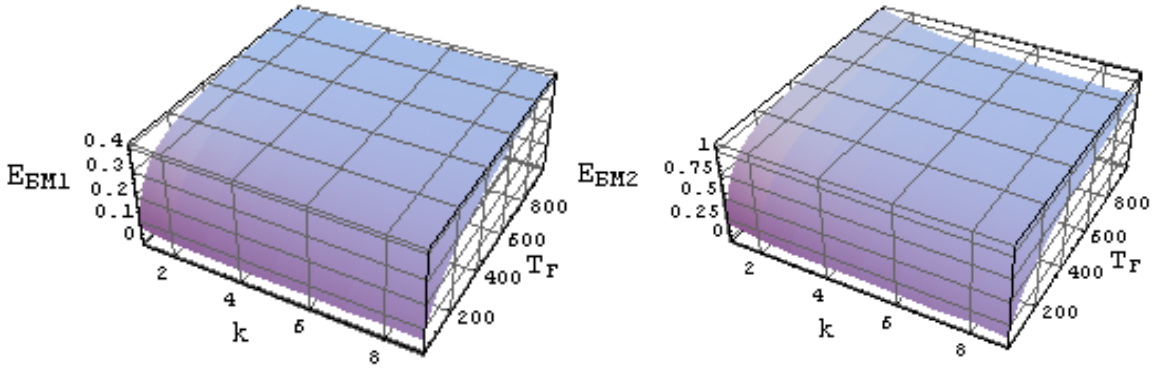


Рисунок 3.13 – Коэффициент эффективности двух вложенных одношаговых блочных методов от числа точек блока и сложности правой части ОДУ

Такая зависимость объясняется тем, что число точек блока связано с числом используемых процессоров. Суммируя все полученные результаты, можно сделать вывод, что из двух рассмотренных методов вложенных форм для блочных одношаговых способов решения нелинейной задачи Коши для ОДУ, второй метод обладает несомненными преимуществами.

3.1.3. Технология локальной экстраполяции на основе одношаговых блочных методов для ОДУ

Реализация **технологии локальной экстраполяции Ричардсона для блочных методов** требует многократных вычислений на одном и том же интервале интегрирования с использованием опорного блочного k_0-точечного метода порядка r_0 на сгущающихся равномерных сетках:

а) $\Omega_{h/n_1} = \{ x_j \}, j = \overline{1,M_1}$ с шагом $h_1 = h / n_1$ в N_1 блоках;

б) $\Omega_{h/n_2} = \{ x_j \}, j = \overline{1,M_2}$ с шагом $h_2 = h / n_2$ в N_2 блоках;

……

в) $\Omega_{h/n_k} = \{ x_j \}, j = \overline{1,M_k}$ с шагом $h_k = h / n_k$ в N_k блоках, где k – число строк экстраполяционной таблицы.

Базовый шаг интегрирования равен $h_1 = h$, $n_1 = 1$, то есть основой счета является сетка: Ω_h. Для генерации вспомогательных сеток выбирается гармонический ряд $P_1 = \{n_2,...,n_k,...\} = \{2,3,4,5,...\}$, как наименее затратный в случае опорного метода произвольного типа (см. 2.20). Блочный опорный метод должен иметь малый порядок точности:

$$T_{11,i} = y_{n,i} = y_{n,0} + ih\left[b_i F_{n,0} + \sum_{j=1}^{k_0} a_{i,j} F_{n,j} \right]; i = \overline{1,k_0}, k_0 \le 4, \qquad (3.23)$$

так как с ростом r_0 вычислительные затраты на технологию в целом существенно возрастают, несмотря на линейное уменьшение длины экстраполяционной таблицы. Для неявных блочных методов это особенно важно, поскольку увеличение порядка, а, следовательно, и числа точек блока, влечет за собой увеличение порядка многократно решаемых СНАУ. Схема разбиения базового интервала на блоки при применении технологии локальной экстраполяции для двухточечного опорного метода и гармонической последовательности приведена на рис. 3.14. Заметим, что для блочных методов базовый интервал интегрирования – это некоторый блок с номером n основной сетки длины $H = k_0 h$. Общее число точек, вычисляемое с разными шагами интегрирования $h_i = h / i, i = \overline{1,k}$ и определяющее узлы сеток Ω_{h_i} по P_1,

равно $M_i = H / h_i = (k_0 h) / h_i = k_0 i$. Соответственно, количество блоков i – той сетки $N_i = M_i / k_o = i$ и $n_i = N_i, i = \overline{1,k}$.

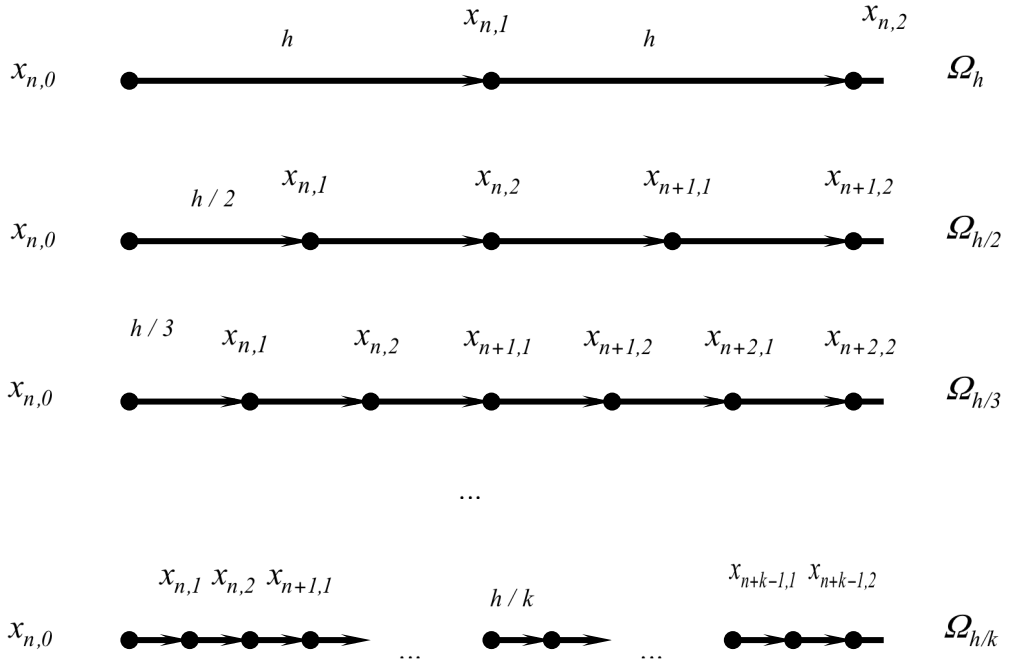


Рисунок 3.14 – Схема разбиения на блоки по технологии локальной экстраполяции для одношаговых блочных методов

Поскольку экстраполируются значения в узлах базовой сетки, необходимо установить механизм соответствия узлов сетки Ω_h основного счета узлам сеток для экстраполяции $\Omega_{h_i}, i = \overline{2,k}$. Для двухточечного опорного блочного метода и гармонического ряда:

1) i - четное число: $x_{n,1}(h) \Leftrightarrow x_{n+\frac{i}{2}-1,2}(h_i)$;

2) i - нечетное число: $x_{n,1}(h) \Leftrightarrow x_{n+\left\lceil \frac{i}{2} \right\rceil,1}(h_i)$;

3) для любых i: $x_{n,2}(h) \Leftrightarrow x_{n+i-1,2}(h_i)$.

Значения первого столбца экстраполяционной таблицы определяются на основе следующих формул:

$$\begin{cases} T_{11,n,i} = y^{(1)}_{n,i} = y^{(1)}_{n,0} + ih \cdot \left[b_i F_{n,0} + \sum_{j=1}^{k_0} a_{i,j} F_{n,j} \right]; i = \overline{1,k_0}; n = \overline{1,N_1}, \\ \ldots \\ T_{k1,n,i} = y^{(k)}_{n,i} = y^{(k)}_{n,0} + i\frac{h}{k} \cdot \left[b_i F_{n,0} + \sum_{j=1}^{k_0} a_{i,j} F_{n,j} \right]; i = \overline{1,k_0}; n = \overline{1,N_k}. \end{cases} \quad (3.24)$$

Каждая из аппроксимаций решения получается за счет N_i раз примененной схемы одношагового блочного k_0-точечного метода с разными шагами интегрирования:

$$\begin{cases} T_{T_{11}} = T_{\bar{y}_{n,i}^{(1)}(h)} = N_1 \cdot T_1^{r_0}(h), \\ \quad\quad\quad ... \\ T_{T_{k1}} = T_{\bar{y}_{n,i}^{(k)}\left(\frac{h}{k}\right)} = N_k \cdot T_1^{r_0}(h/k), \end{cases} \tag{3.25}$$

где $T_1^{r_0}(h/i), i = \overline{1,k}$ – время, необходимое для решения задачи Коши для ОДУ опорным методом порядка r_0 с шагом h_i. Затем по формуле Эйткена-Невилла вычисляются приближения $T_{22}, T_{33,\ldots}$ и $T_{k,k}$. Время последовательных вычислений по схеме локальной экстраполяции состоит из времени определения k аппроксимаций решения с различными шагами интегрирования и времени вычисления элементов экстраполяционной таблицы:

$$T_1^{14} = T_{T_{11}} + T_{T_{21}} + ... + T_{T_{k1}} + T_1^{ext-tab}, \quad T_1^{14} = T_1^{r_0} \cdot \sum_{i=1}^{k} n_i + T_1^{ext-tab}.$$

Для блочных опорных методов порядка $r_0 = k_0 + 1$ и P_1 имеем:

$$N_{P_1}(k) = \sum_{i=1}^{k} n_i = (k^2 + k)/2; \; k = r - r_0 + 1 \;\Rightarrow$$

$$\Rightarrow \quad N_{P_1(k)}^{r_0} = [r^2 - r(2r_0 - 3) + (r_0 - 2)(r_0 - 1)]/2 \;\Rightarrow \tag{3.26}$$

$$N_{P_1(k)}^{r_0=3} = (r^2 - 3r + 2)/2.$$

Время вычисления экстраполяционной таблицы равно:

$$\begin{aligned} T_1^{ext-tab} &= 0.5(k^2 - k)(2t_{mul} + 3t_{ad}) = \\ &= 2.5(r^2 - r(2r_0 - 1) + r_0(r_0 - 1)) \cdot t_{op} \end{aligned} \tag{3.27}$$

при $r_0 = 3$: $T_1^{ext-tab} = 2.5 \cdot (r^2 - 5r + 6) \cdot t_{op}$.

Пусть L_0 – предельное, максимальное число итераций, необходимое для решения СНАУ (3.24) итерационным методом. Тогда время выполнения последовательного алгоритма решения ОДУ на базе

блочных одношаговых методов со встроенным методом оценки погрешности по технологии локальной экстраполяции составляет:

$$T_1^{14} = T_1^{r0} \cdot \left(\frac{r^2 - 3r + 2}{2} \right) + 5 \left(\frac{r^2 - 5r + 6}{2} \right) \cdot t_{op}, \qquad (3.28)$$

где $T_1^{r0} = (L_0 k_0 + 1) \cdot T_F + [2L_0 (k_0^{\,2} + 2k_0) + 3k_0] \cdot t_{op}$,

при $L_0 = k_0$: $T_1^{r0} = (k_0^2 + 1) \cdot T_F + [2k_0^{\,3} + 4k_0^{\,2} + 3k_0] \cdot t_{op}$,

$$T_1^{14} = (k_0^2 + 1) \left(\frac{r^2 - 3r + 2}{2} \right) \cdot T_F +$$

$$+ \left[(2k_0^{\,3} + 4k_0^{\,2} + 3k_0) \left(\frac{r^2 - 3r + 2}{2} \right) + 5 \left(\frac{r^2 - 5r + 6}{2} \right) \right] \cdot t_{op}. \qquad (3.29)$$

Тогда, при $k_0 = 2 \;\Rightarrow\; L_0 = 2 \;\Rightarrow r_0 = 3$ имеем: $T_1^{r_0 = 3} = 5T_F + 38t_{op}$.

Как результат: $T_1^{14} = (5T_F + 38t_{op}) \left(\frac{r^2 - 3r + 2}{2} \right) + 5 \left(\frac{r^2 - 5r + 6}{2} \right) \cdot t_{op}$,

$$T_1^{14} = 5 \cdot \left(\frac{r^2 - 3r + 2}{2} \right) \cdot T_F + \left(\frac{43r^2 - 89r + 106}{2} \right) \cdot t_{op}. \qquad (3.30)$$

Для построения параллельного алгоритма локальной экстраполяции на основе блочного одношагового неявного метода (рис. 3.15) воспользуемся результатами второй главы. Реализуем первую макрооперационную схему, граф влияния которой приведен на рис. 2.20 в применении к ОДУ. В качестве основной макрооперации выбирается однократное вычисление некоторой аппроксимации решения на основе k_0-точечного опорного метода малого порядка. Каждая макрооперация, как и в предыдущих случаях, представляет собой итерационный процесс, выполненный максимально возможное число раз, то есть L_0. Затем формируется остальная часть экстраполяционной таблицы. Определим оценки времени выполнения параллельного алгоритма:

$$T_p^{14} = T_p^{r0} \cdot \sum_{i=1}^{k} n_i + T_p^{ext-tab}.$$

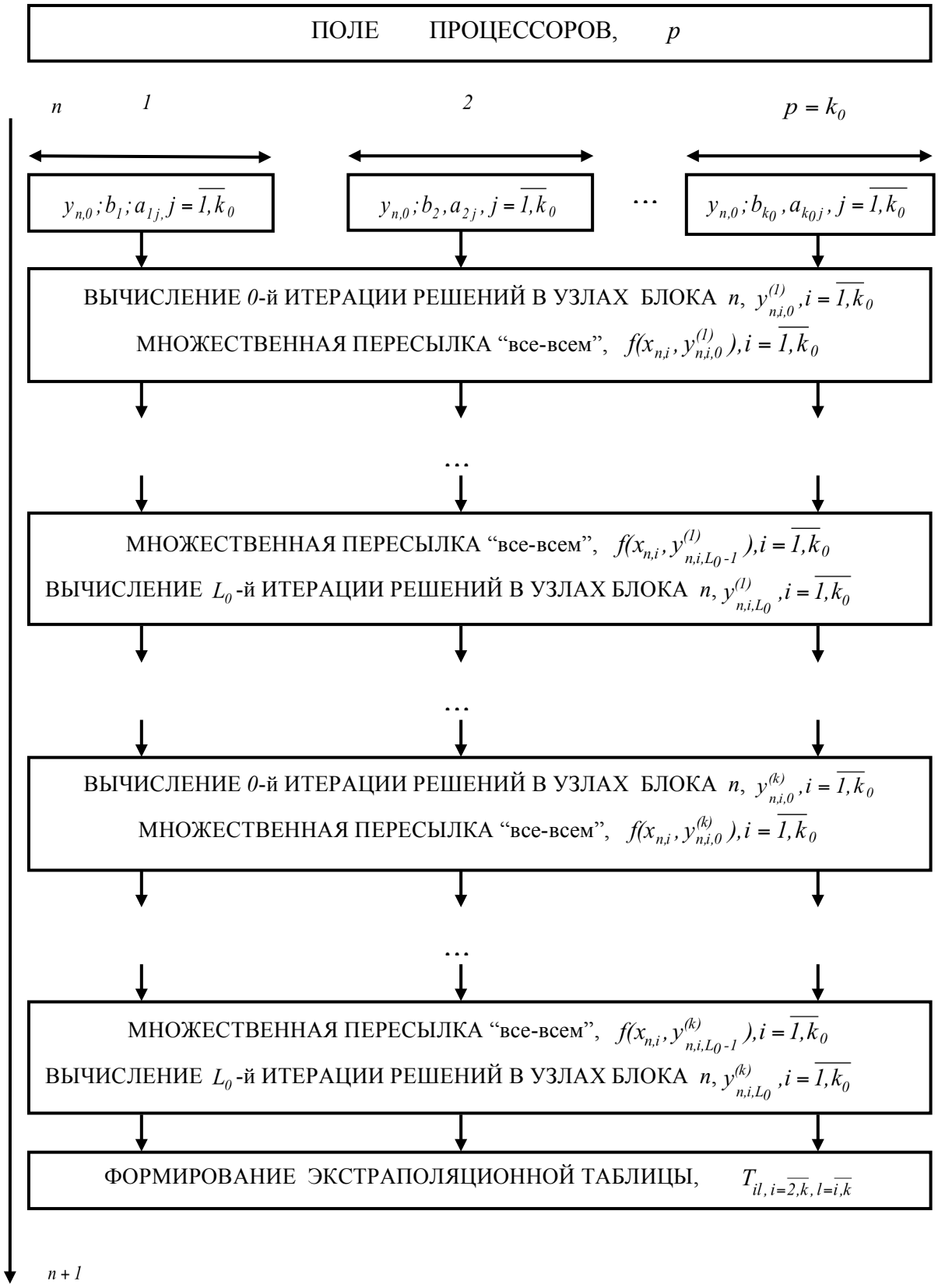


Рисунок 3.15 – Параллельный алгоритм технологии локальной экстраполяции на основе одношагового k_0-точечного метода для ОДУ

Реализация параллельных вычислений для опорного блочного метода потребует следующего времени вычислительных и обменных операций:

$$T_{p,comp}^{r_0} = (L_0 + 1)\cdot T_F + [2L_0(k_0 + 2) + 3]\cdot t_{op}, \tag{3.31}$$

$$T_{p,comm}^{r_0} = (L_0 + 1)\cdot T_{all-to-all}(1, p). \tag{3.32}$$

Время паралельного выполнения технологии Ричардсона составит:

$$T_p^{r_0}\cdot\sum_{i=1}^{k} n_i = \left(\frac{r^2 - r(2r_0 - 3) + (r_0 - 2)(r_0 - 1)}{2}\right)\cdot\left((L_0 + 1)\cdot T_F + [2L_0(k_0 + 2) + 3]\right)\cdot t_{op},$$

$$T_p^{ext-tab} = 5\cdot\frac{r^2 - r(2r_0 - 1) + r_0(r_0 - 1)}{2}\cdot t_{op}.$$

Для двухточечного опорного метода:

$$T_{p,comp}^{14} = T_{p,comp}^{r_0=3}\cdot\sum_{i=1}^{k} n_i + T_{p,comp}^{ext-tab} = \\ = (3T_F + 19t_{op})\cdot\left(\frac{r^2 - 3r + 2}{2}\right) + T_{p,comp}^{ext-tab}. \tag{3.33}$$

Соответственно, коэффициент потенциального ускорения алгоритма в случае, если время обращения к правой части ОДУ доминирует над другими вычислениями, определяется, как:

$$S_{pot}^{14} = T_1^{14} / T_p^{14} \approx [(L_0k_0 + 1)\cdot T_F] / [(L_0 + 1)\cdot T_F] \approx k_0 = p,$$

$$E_{pot}^{14} = S_{pot}^{14} / p \approx 1.$$

Аналогичный результат имеем для тривиальной правой части, то есть потенциально метод локальной экстраполяции для многоточечных блочных методов обладает высокой степенью внутреннего параллелизма. Обменные операции рассмотренного параллельного алгоритма составляют:

$$T_{p,comm}^{14} = \left(\frac{r^2 - r(2r_0 - 3) + (r_0 - 2)(r_0 - 1)}{2}\right)(L_0 + 1)\cdot T_{all-to-all}(p),$$

$$T_{p,comm}^{14} = \left(\frac{r^2 - 3r + 2}{2}\right)\cdot(L_0 + 1)\cdot T_{all-to-all}(p), \tag{3.34}$$

Анализ коммуникационной составляющей дает основание утверждать, что наилучшей для рассмотренного метода по-прежнему является топология гиперкуб. Для сравнения параллельных алгоритмов различных способов оценки локальной апостериорной погрешности на основе многоточечных методов оценим их динамические характеристики (рис. 3.16-3.17).

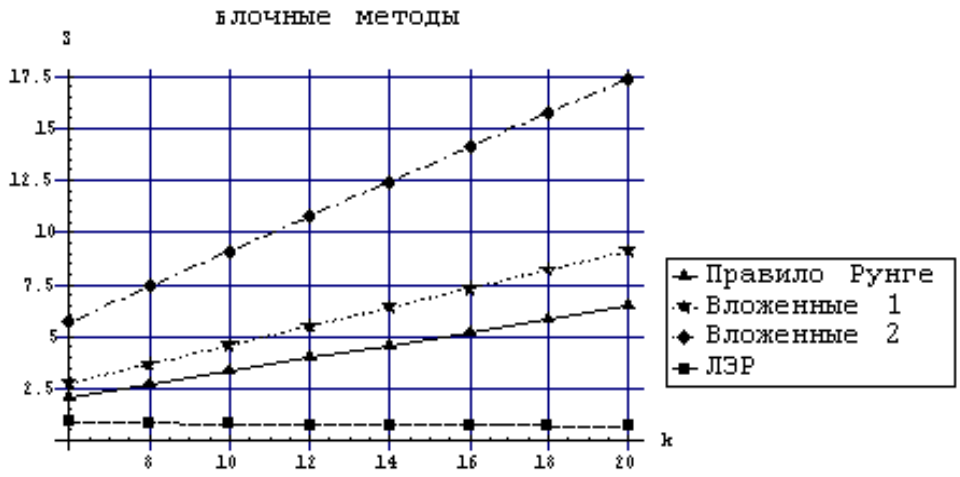


Рисунок 3.16 – Коэффициент ускорения блочных методов с различными способами оценки локальной погрешности, $p = k$

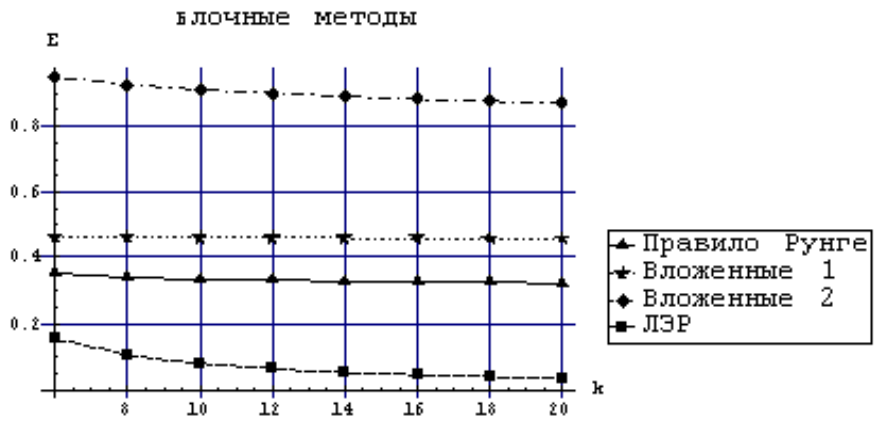


Рисунок 3.17 – Коэффициент эффективности блочных методов с различными способами оценки локальной погрешности, $p = k$

Установлено, что наиболее простыми и наименее затратными являются методы вложенных форм для систем любых архитектур, различных параметров задачи и метода. Это имеет особое значение, так

как, несмотря на трудоемкость, теоретически нет никаких ограничений для получения блочных методов более высоких порядков.

Однако, вложенный блочный метод на основе комбинации независимых формул, как показали исследования, имеет практически такой же порядок сложности, что и метод с использованием правила Рунге. Параллельный алгоритм технологии локальной экстраполяции имеет худшие характеристики параллелизма, областью его применения по-прежнему остаются высокоточные решения.

3.1.4. Особенности параллельных вложенных блочных многоточечных методов при интегрировании систем ОДУ

Разработанные и обоснованные в предыдущих подразделах одношаговые блочные методы решения задачи Коши для обыкновенного дифференциального уравнения с оценкой локальной погрешности можно перенести и на случай системы уравнений. При переходе от одного уравнения к системе появляется возможность использовать системный параллелизм, который, как правило, гораздо больше параллелизма метода.

Блочный одношаговый k-точечный метод для СОДУ имеет вид:

$$\bar{y}_{n,i} = \bar{y}_{n,0} + ih\left[b_i F_{n,0} + \sum_{j=1}^{k} a_{i,j} F_{n,j}\right]; i = \overline{1,k}; n = \overline{1,N}, \qquad (3.35)$$

и требует решения системы нелинейных алгебраических уравнений размерности $m \times k$:

$$\begin{cases} y_{n,q,i;0} = y_{n,q,0} + ihf_q(x_{n,0}; \bar{y}_{n,0}), \\ i = \overline{1,k},\ n = \overline{1,N},\ q = \overline{1,m}, \\ y_{n,q,i;l+1} = y_{n,q,0} + ih[b_i f_q(x_{n,0}; \bar{y}_{n,0}) + \sum_{j=1}^{k} a_{i,j} \cdot f_q(x_{n,j}; \bar{y}_{n,j,l})], \\ l = \overline{0,L-1}, \end{cases} \qquad (3.36)$$

где $F_{n,q,j} = f_q[x_{n,j}; y_1(x_{n,j}), y_2(x_{n,j}), ..., y_m(x_{n,j})]$, $q = \overline{1,m}$ есть $q-$ тая компонента вектора правой части СОДУ.

Особенности параллельного интегрирования систем обыкновенных дифференциальных уравнений рассмотрим на основе

вложенных многоточечных блочных одношаговых методов, как наиболее эффективных из существующих способов оценки локальной апостериорной погрешности:

$$\begin{cases} y_{n,q,i;0} = y_{n,q,0} + ihF_{n,q,0},\ i = \overline{1,k}, \\ y_{n,q,i} = y_{n,q,i;l+1} = y_{n,q,0} + ih(b_i F_{n,q,0} + \sum\limits_{j=1}^{k} a_{i,j} F_{n,q,j;l}), \\ l = \overline{0, L-2}, \\ \hat{y}_{n,q,i} = y_{n,q,i;L} = y_{n,q,0} + ih(b_i F_{n,q,0} + \sum\limits_{j=1}^{k} a_{i,j} F_{n,q,j;L-1}). \end{cases} \tag{3.37}$$

Последовательный алгоритм интегрирования СОДУ по численной схеме (3.37) имеет следующую вычислительную сложность:

$$T_1 = (Lk+1)\sum_{i=1}^{m} T_{f_i} + m[2L(k^2+2k)+3k]t_{op}. \tag{3.38}$$

Итерационный процесс, описываемый вычислительной схемой (3.37), может быть реализован только последовательно. Однако он позволяет распределить вычисление текущей l - той итерации векторов решения следующими способами:

1) на k независимых процессов, по числу точек в каждом блоке ($Dop_1 = k$);

2) на m по числу компонент в системе ($Dop_2 = m$);

3) на mk процессов по максимальной возможной ширине параллельного метода и системы ($Dop_3 = mk$).

Вычислительные схемы параллельного вложенного блочного многоточечного метода для СОДУ, использующие описанные способы распараллеливания, приведены на рисунках 3.18-3.20.

Для сокращения записи введены следующие обозначения. Вектор решений $y_{n,q,i;l}$ имеет четыре индекса, n – номер блока, q – номер компоненты исходной СОДУ, i – номер точки в блоке, l – номер итерации при решении соответствующей системы нелинейных алгебраических уравнений. Наличие прочерка для одного из индексов будет обозначать, что берется соответствующий вектор значений.

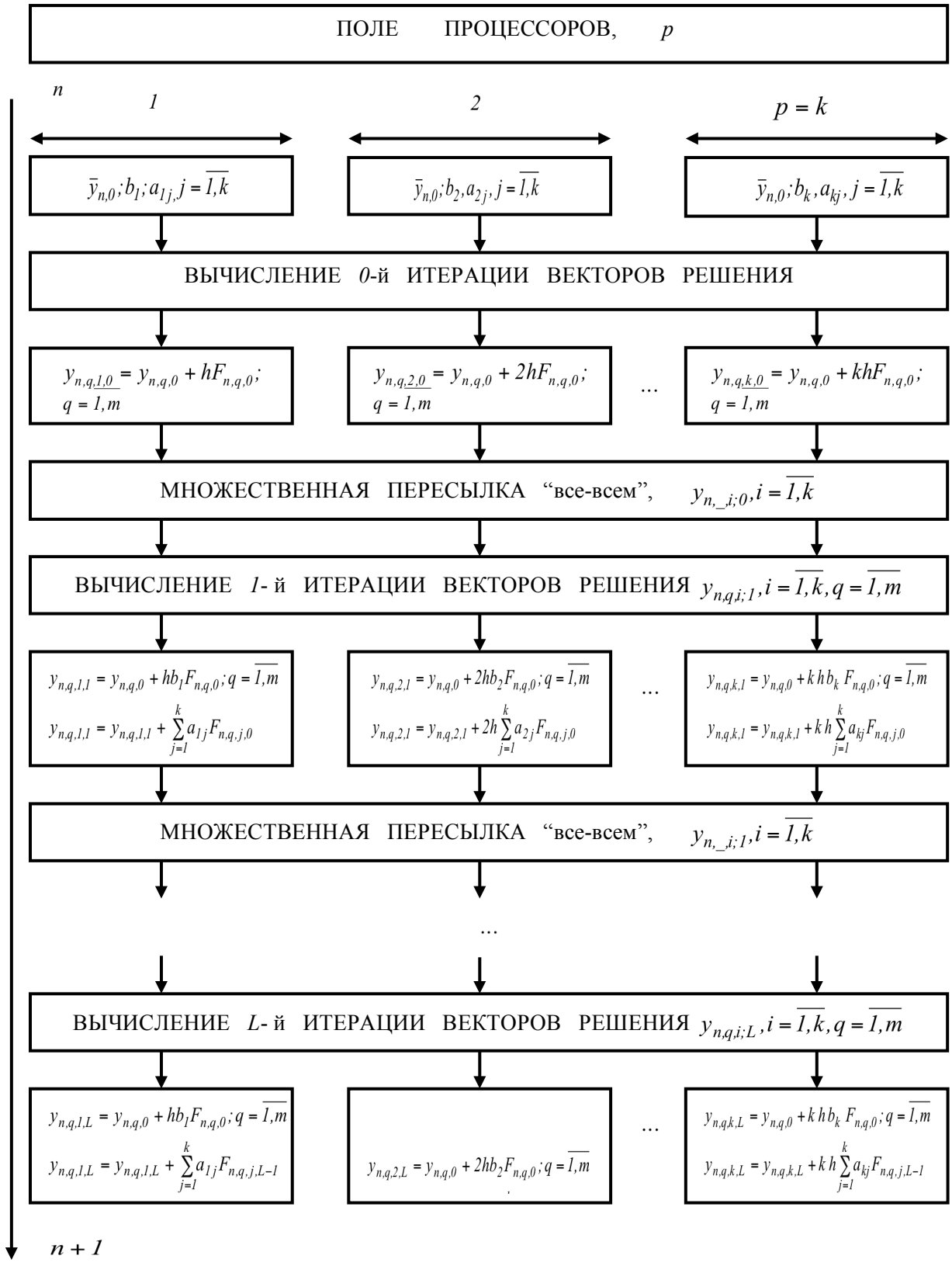


Рисунок 3.18 – Параллельный алгоритм №1 вложенного одношагового k-точечного блочного метода интегрирования СОДУ

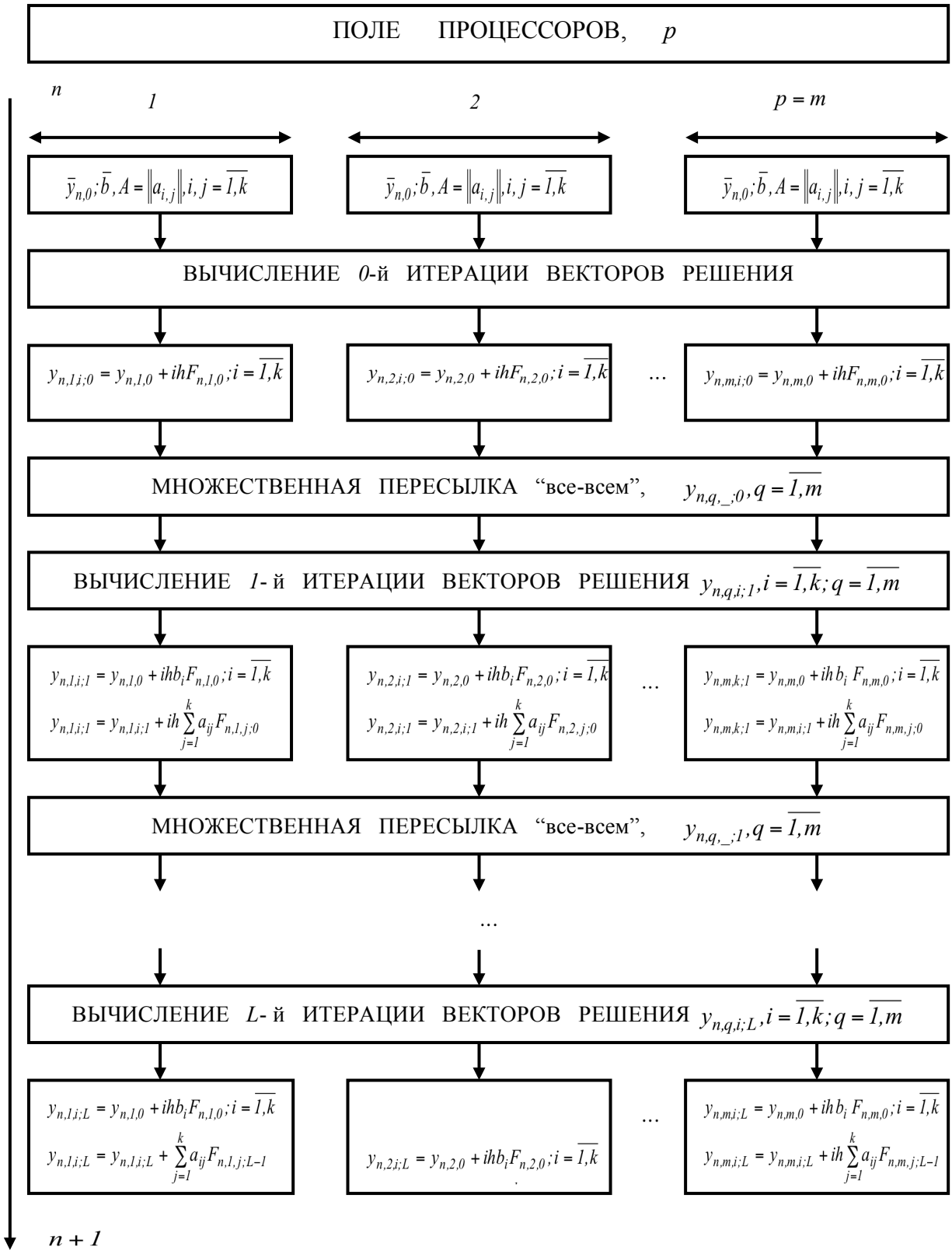


Рисунок 3.19 – Параллельный алгоритм №2 вложенного одношагового k-точечного блочного метода интегрирования СОДУ

Так, $y_{n,\_,i;l}$ – обозначает вектор решений размерности m для СОДУ в i-той точке n-го блока, вычисленный на l-той итерации: $(y_{n,1,i;l}, y_{n,2,i;l}, \ldots, y_{n,m,i;l})$, а $y_{n,q,\_;l}$ – вектор решения размерности k для q-той компоненты СОДУ, вычисленный во всех точках n-го блока на l-той итерации: $(y_{n,q,1;l}, y_{n,q,2;l}, \ldots, y_{n,q,k;l})$.

Оценим качество полученных параллельных алгоритмов и определим приоритетные области использования каждого из них:

$$1)\ T^1_{p,comp} = (L+1)\cdot \sum_{i=1}^{m} T_{f_i} + m[2Lk + 4L + 3]\cdot t_{op}, \tag{3.40}$$

$$T^1_{p,comm} = (L+1)\cdot T_{all-to-all}(m,p) = (L+1)\cdot T_{all-to-all}(m,k), \tag{3.41}$$

$$2)\ T^2_{p,comp} = (Lk+1)T_{F_{max}} + [2L(k^2+2k)+3k]t_{op}, \tag{3.42}$$

$$T^2_{p,comm} = (L+1)\cdot T_{all-to-all}(k,p) = (L+1)\cdot T_{all-to-all}(k,m). \tag{3.43}$$

По первой схеме каждый процессор вычисляет все компоненты вектора решения некоторой точки блока для каждого шага итерационного процесса.

По второй схеме каждый процессор вычисляет одну компоненту вектора решения всех точек блока для каждого шага итерационного процесса. Определим аналитические выражения для расчета времени коммуникаций с учетом наилучшей для данного типа операций топологии гиперкуб:

$$T^1_{p,comm} = (L+1)\cdot [t_s \cdot log_2 k + m(k-1)\cdot t_w],$$

$$T^2_{p,comm} = (L+1)\cdot [t_s \cdot log_2 m + k(m-1)\cdot t_w].$$

Вторая схема имеет время вычислений и общее время выполнения меньшие, чем первая. Количественное выражение коэффициентов ускорения варьируется при изменении машинно-зависимых коммуникационных констант, сложности правой части. Однако при этом коэффициент ускорения вычислений для первой вычислительной схемы всегда ниже, чем для второй.

Исследованные вычислительные схемы блочного вложенного метода решения СОДУ имеют ограничения на число используемых процессоров.

Разработаем параллельный алгоритм, не содержащий таких ограничений и учитывающий как параллелизм метода, так и системный (рис. 3.20).

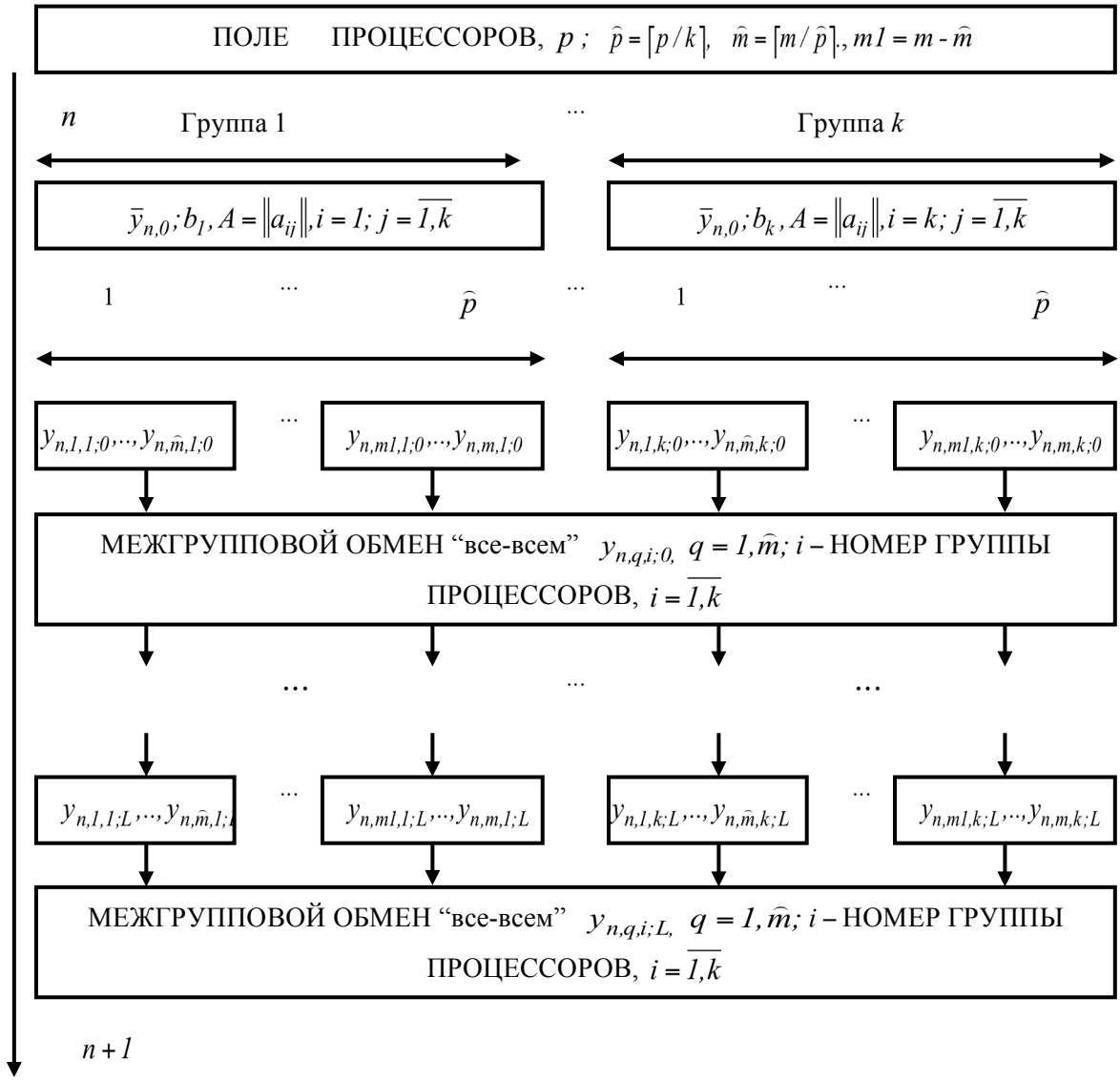


Рисунок 3.20 – Вычислительная схема №3 параллельного алгоритма вложенного одношагового k-точечного блочного метода для СОДУ

Пусть имеется параллельная ВС с p процессорами; разобьем процессорное поле на k групп по числу точек в блоке. Число процессоров в одной группе равно $\hat{p} = \lceil p/k \rceil$, число компонент системы, вычисляемых одним процессором, составляет $\hat{m} = \lceil m/\hat{p} \rceil$.

Каждая группа отвечает за вычисление решения в одной точке блока для всей СОДУ на одной итерации.

Время последовательной реализации этого алгоритма вычисляется по (3.38), время параллельной для топологии гиперкуб равно:

$$T^3_{p,comp} = (L+1)\cdot \sum_{i=1}^{\lceil mk/p \rceil} T_{f_i} + \lceil mk/p \rceil \cdot [2Lk + 4L + 3] \cdot t_{op}, \qquad (3.44)$$

$$T^3_{p,comm} = (L+1)\cdot [t_s \cdot log_2 mk + \lceil mk/p \rceil (mk-1)\cdot t_w] \qquad (3.45)$$

Потенциальные характеристики всех трех алгоритмов при произвольных правых частях близки к идеальным (линейное ускорение и единичная эффективность) при числе процессоров, равном максимальной степени параллелизма каждого метода. В силу различной сложности обменных операций и максимально возможного числа процессоров: первая вычислительная схема, использующая параллелизм метода, имеет наихудшие характеристики реального параллелизма. Третья вычислительная схема обладает несомненным преимуществом перед двумя другими (рис. 3.21-3.22).

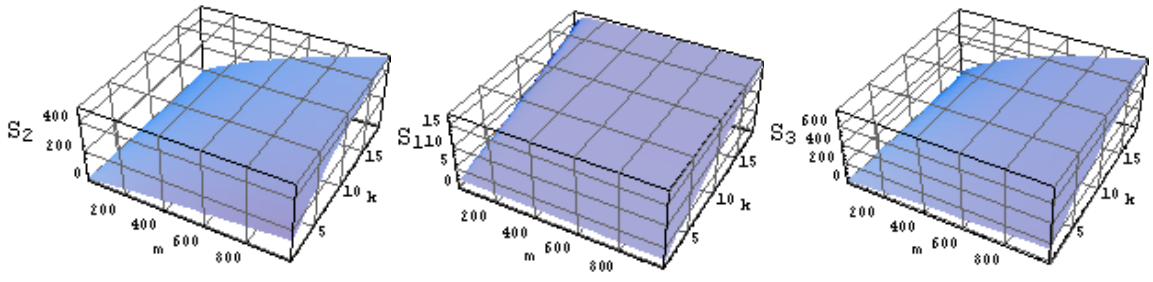


Рисунок 3.21 – Коэффициент ускорения параллельной реализации трех вычислительных схем от числа точек в блоке и размерности задачи

Качество рассмотренных алгоритмов решения СОДУ существенно зависит от соотношения параметров задачи, численного метода и параллельной системы.

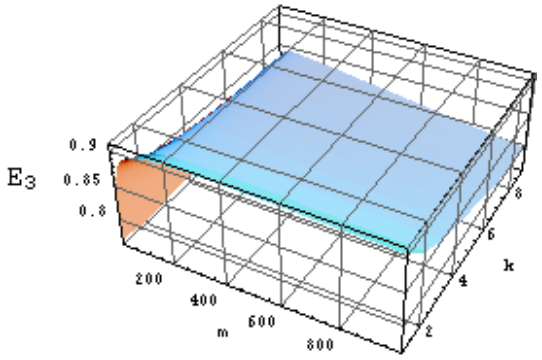


Рисунок 3.22 – Коэффициент эффективности параллельного алгоритма №3

Наилучшие характеристики параллелизма имеют место для многомерных задач, сложных правых частей, высокоскоростных сетей передачи данных для схемы №3.

3.2. Параллельные алгоритмы решения нелинейной задачи Коши полностью неявными одношаговыми методами Рунге-Кутты (ПНМРК)

Численное решение (3.1-3.2) **полностью неявным s-стадийным методом Рунге-Кутты** можно получить последовательно по шагам:

$$\begin{cases} \bar{y}_{n+1} = \bar{y}_n + h \cdot \sum_{i=1}^{s} b_i \bar{k}_i, \\ \bar{k}_i = \bar{f}[x_n + c_i h;\ y_n + h \cdot \sum_{j=1}^{s} a_{ij} \bar{k}_j], i = \overline{1,s}, \end{cases} \tag{3.46}$$

где s-размерные вектора $c = (c_1, c_2, ..., c_s)^T$, $b = (b_1, b_2, ..., b_s)^T$ и матрица A описывают уникальный вариант метода. При решении СОДУ полностью неявными методами ms стадийных коэффициентов могут быть получены на основе итерационной схемы:

$$\begin{cases} \bar{k}_i^{(0)} = \bar{f}[x_n + c_i h;\ \bar{g}_i^{(0)}], \\ \bar{k}_i^{(l+1)} = \bar{f}[x_n + c_i h;\ \bar{g}_i^{(l)}], \end{cases} \tag{3.47}$$

где $\bar{g}_i^{(0)} = \bar{y}_n$, $\bar{g}_i^{(l)} = \bar{y}_n + h \sum_{j=1}^{s} a_{ij} \bar{k}_j^{(l)}$, $i = \overline{1,s}$, $l = \overline{1,N}$. (3.48)

3.2.1. Разработка и оценка эффективности параллельного алгоритма ПНМРК для ОДУ

Параллельное решение ОДУ на базе ПНМРК концентрируется на выполнении одного шага интегрирования. Поскольку в неявных методах для одного уравнения имеется возможность использовать параллелизм метода, распределим вычисление каждого из стадийных векторов на отдельный процессор в пределах одного шага итерации. В качестве основной макрооперации вводится вычисление коэффициентов $K^{(l)} = \left(k_1^{(l)}, k_2^{(l)}, ..., k_s^{(l)}\right)$ на l-том шаге итерационного процесса. Параллельный алгоритм введенной макрооперации разработан с использованием графа влияния (рис. 3.23).

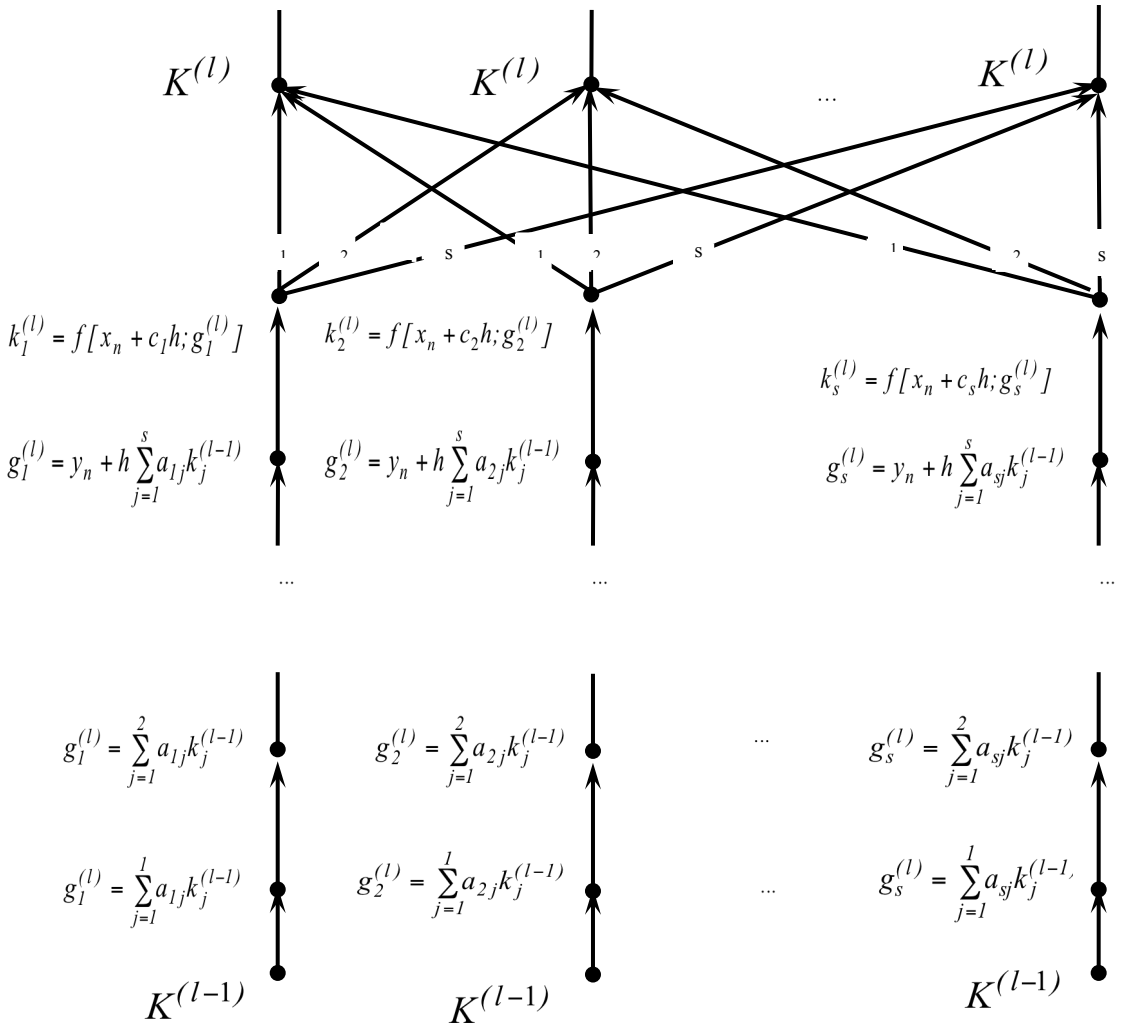


Рисунок 3.23 – Граф влияния вычисления l – той итерации стадийных векторов $K^{(l)} = (k_1^{(l)}, k_2^{(l)}, ..., k_s^{(l)})$ ПНМРК для ОДУ

Каждый из p процессоров отвечает за вычисление l-той итерации i-той компоненты вспомогательного вектора $g_i^{(l)}, i = \overline{1,s}$. Затем производится обмен между процессорами по типу “все-всем” для вычисления очередной итерации коэффициентов $k_i, i = \overline{1,s}$.

Время последовательного метода для одного уравнения при реализации ПНМРК составляет:

$$T_1^{21} = \underbrace{= s \cdot (T_f + t_{mul} + t_{ad})}_{k_i^{(0)}} + N \cdot \underbrace{[s^2 \cdot t_{mul} + s^2 \cdot t_{ad} + s \cdot T_f]}_{k_i^{(l)}} + \underbrace{(s+1) \cdot (t_{mul} + t_{ad})}_{y_{n+1}} \quad (3.49)$$

Рисунок 3.24 представляет параллельный алгоритм решения одного дифференциального уравнения полностью неявным методом на мультикомпьютере из p процессоров, объединенных однородной коммутационной сетью [43-44]. Для параллельного алгоритма ПНМРК время выполнения вычислений на p процессорах без учета обменов и других накладных расходов равно:

$$T_{p,comp}^{21} = \underbrace{T_f + t_{mul} + t_{ad}}_{k_i^{(0)}} + N \cdot \underbrace{[s \cdot t_{mul} + s \cdot t_{ad} + T_f]}_{k_i^{(l)}} + \underbrace{(s+1) \cdot (t_{mul} + t_{ad})}_{y_{n+1}}. \quad (3.50)$$

Коэффициент потенциального ускорения для ОДУ со сложной правой частью определяется следующим образом: $S_{pot}^{21} = T_1^{21} / T_p^{21} \approx \frac{s(N+1)T_f}{(N+1)T_f} = s = p$. Аналогичный результат имеем при $t_{ad} = t_{mul} = t_{op}$. Максимальная степень параллелизма алгоритма составляет: $Dop = s$. Время, необходимое на реализацию межпроцессорных обменов рассматриваемого алгоритма, определяется $N+1$ раз повторенной множественной операцией по типу “все-всем” для p процессоров $T_{p,comm}^{21} = (N+1) \cdot T_{all-to-all}(p)$:

– кольцо: $T_{p,comm}^{21,R} = (N+1) \cdot (t_s + t_w) \cdot (p-1);$

– тор: $T_{p,comm}^{21,M} = (N+1) \cdot \left[2t_s \cdot (\sqrt{p} - 1) + t_w (p-1) \right];$

– гиперкуб: $T_{p,comm}^{21,H} = (N+1) \cdot \left[t_s \log_2 p + t_w \cdot (p-1) \right].$

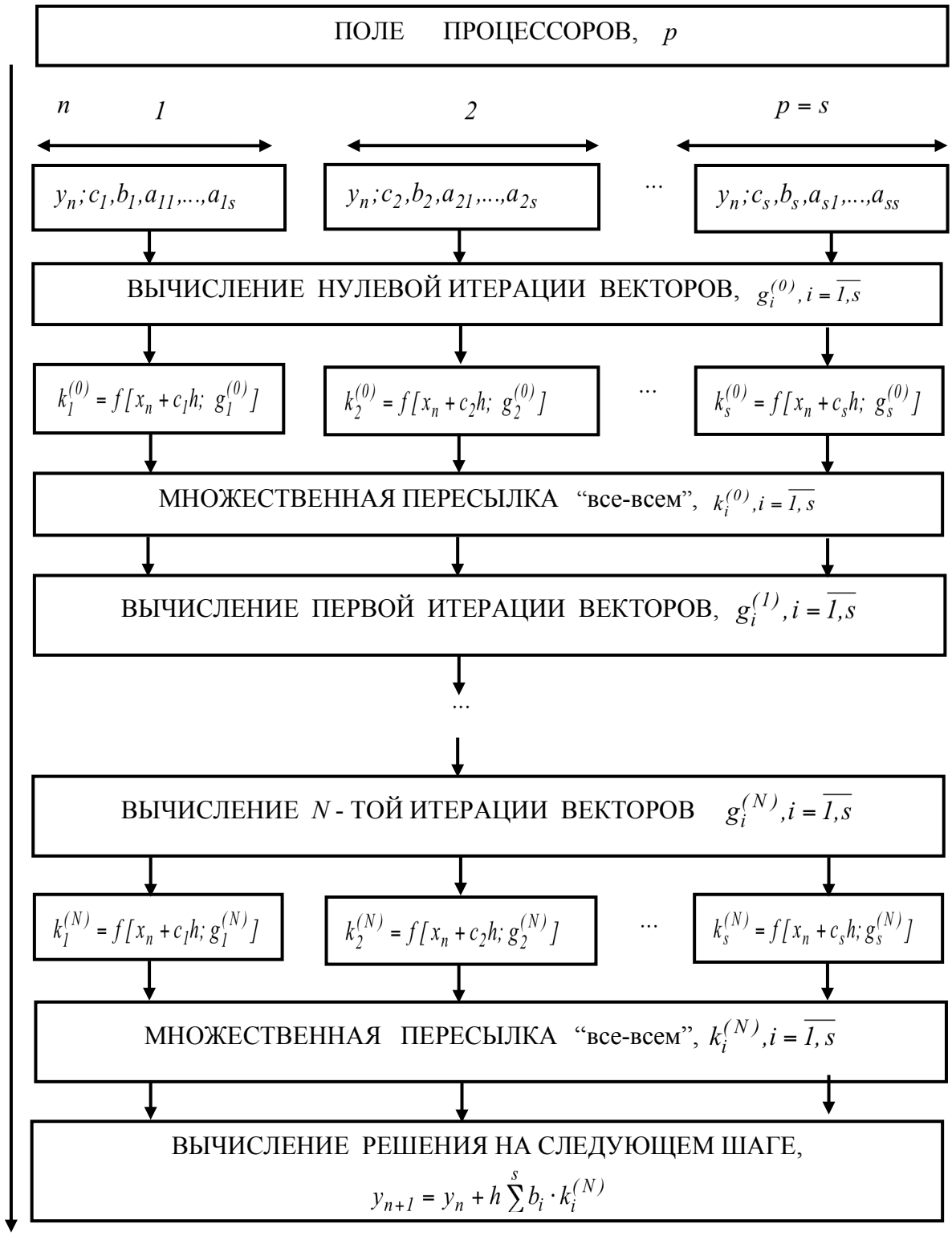


Рисунок 3.24 – Вычислительная схема параллельного алгоритма полностью неявного одношагового метода типа Рунге-Кутты для одного обыкновенного дифференциального уравнения

Реальная эффективность (рис. 3.25-3.26) параллельных вычислений во многом определяется трудоемкостью коммуникационных операций. Анализ предложенных топологий показал, что, безусловно, наихудшим вариантом для рассматриваемого алгоритма является соединение типа кольцо, наилучшим - гиперкуб (рис. 3.26).

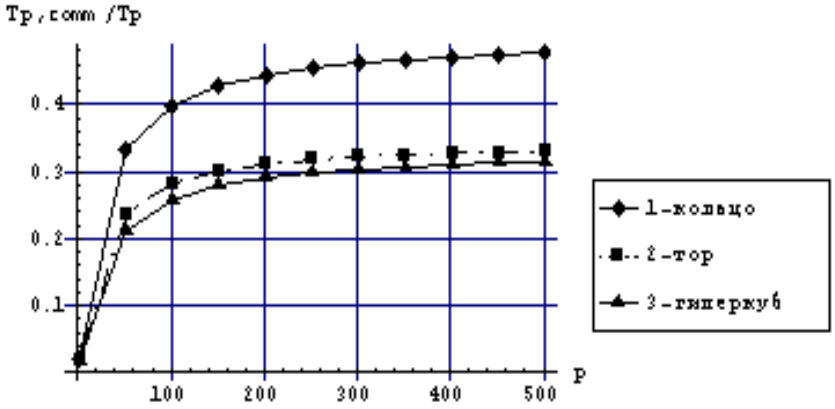


Рисунок 3.25 – Доля коммуникационных операций к общим накладным расходам параллельного алгоритма ПНМРК для ОДУ

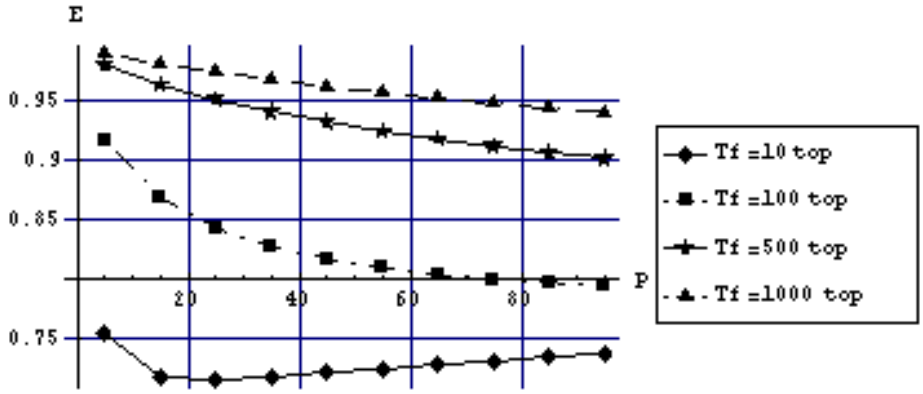


Рисунок 3.26 – Коэффициент эффективности параллельного ПНМРК для ОДУ при различной сложности правой части

Динамические характеристики параллельных ПНМРК для ОДУ имеют большой разброс по значениям. Поэтому их эффективное применение возможно только при тщательном учете параметров параллельной системы, алгоритма и исходной задачи [58-59].

3.2.2. Особенности распараллеливания полностью неявного метода Рунге-Кутты для СОДУ при решении нелинейной задачи Коши

При параллельном решении систем обыкновенных дифференциальных уравнений с использованием полностью неявных методов Рунге-Кутты помимо параллелизма метода появляется возможность использовать и системный параллелизм. Применение ПНМРК к СОДУ требует решения системы нелинейных уравнений для определения $\overline{k_i} = (k_{i1}, k_{i2}, ..., k_{im}), i = \overline{1,s}$:

$$\begin{cases} \overline{g}_i^{(0)} = 0,\ \overline{k}_i^{(0)} = F(x_n + c_i h, \overline{y}_n), \\ \overline{g}_i^{(l)} = a_{i1}\overline{k}_1^{(l)} + a_{i2}\overline{k}_2^{(l)} + ... + a_{is}\overline{k}_s^{(l)}, \\ \overline{k}_i^{(l)} = F[x_n + c_i h, \overline{y}_n + h \cdot g_i^{(l-1)}], \\ i = 1,...,s; l = 1,...,N. \end{cases} \tag{3.51}$$

Вычислительная схема параллельного ПНМРК для решения СОДУ приведена на рисунке 3.27. Число нелинейных уравнений, как и число неизвестных компонент стадийных векторов $k_{ij}\left(i = \overline{1,s}; j = \overline{1,m}\right)$ равно sm. Разобьем все множество процессоров на s групп, и каждая группа будет отвечать за вычисление одного коэффициента $\overline{k_i} = (k_{i1}, k_{i2}, ..., k_{im}), i = \overline{1,s}$, количество процессоров в группах одинаково и равно $p_i = \lceil p / s \rceil = c, \forall i = \overline{1,s}$. Далее каждый из p_i процессоров s-той группы будет параллельно вычислять $d = \lceil m / p_i \rceil$ компонент вектора $\overline{g}_i$ на каждой из $N + 1$ итераций.

Внутри итерационного процесса каждый процессор в группе обменивается своими данными со всеми другими (обмен “все-всем” в группе) процессами группы компонентами вектора $\overline{g}_i$. Заметим, что нет необходимости во внутригрупповом обмене векторами $\overline{k}_i, i = \overline{1,s}$, так как на следующем шаге итерации каждый процессор в группе будет вычислять те же компоненты стадийных векторов. После окончания итерационного процесса необходимо перераспределить стадийные вектора между процессорами, для этого должен быть произведен обмен данными между группами – межгрупповой обмен по типу “все-всем”.

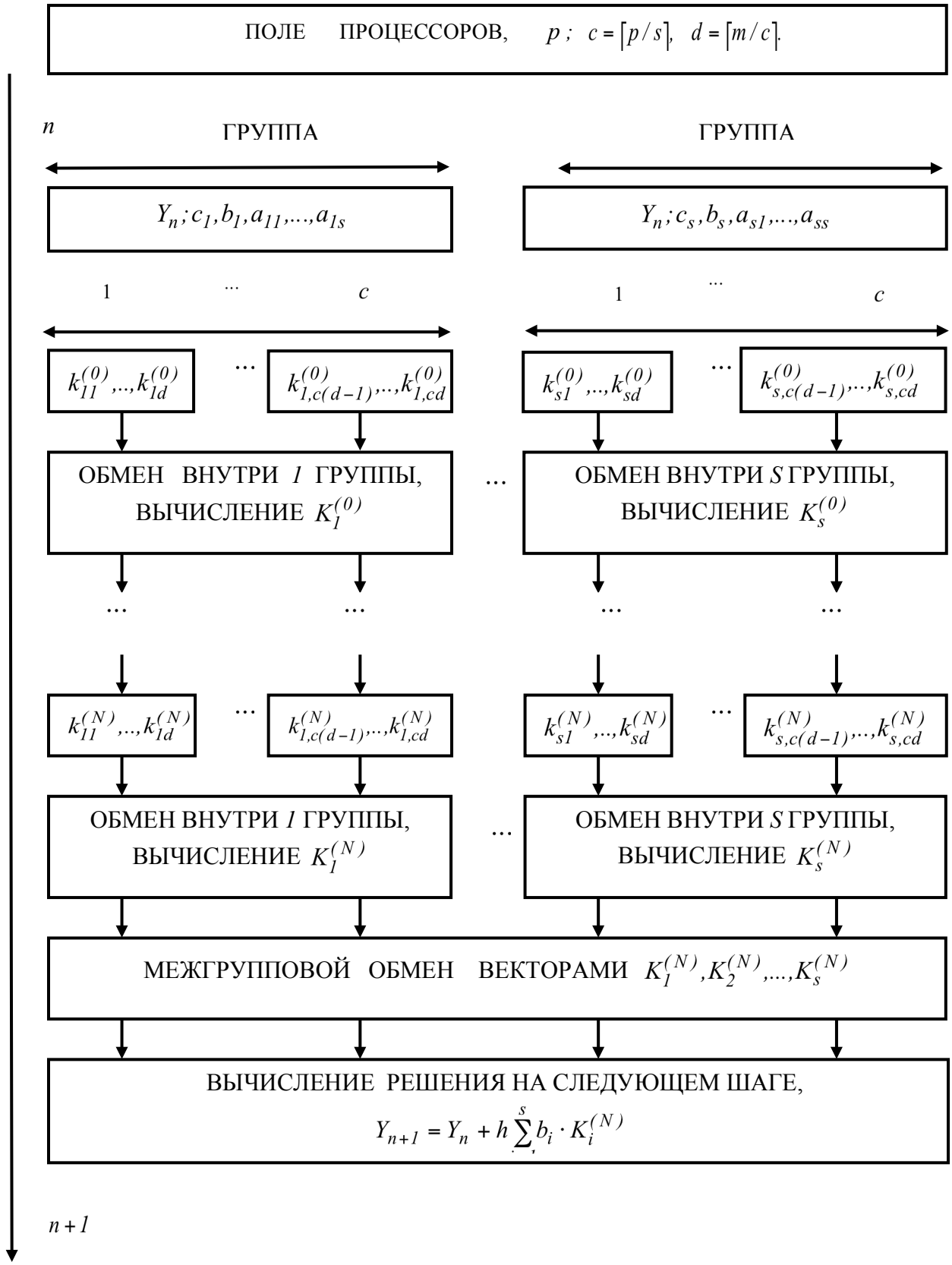


Рисунок 3.27 – Вычислительная схема параллельного алгоритма полностью неявного одношагового метода типа Рунге-Кутты для СОДУ

При вычислении аппроксимации решения на каждом временном шаге на одном процессоре располагается $\lceil m/p \rceil$ компонент вектора решения.

Межгрупповой обмен может быть осуществлен двумя способами:

1) каждый первый процессор в группе передает вектор $\bar{k}_i$ в первый элемент каждой другой группы и производится групповой обмен в группе;

2) межгрупповая передача по типу “все-всем”.

Определим динамические характеристики полученного алгоритма. Время вычислений при последовательной реализации равно:

$$T_1^{22} = (N+1)s \cdot \sum_{i=1}^{m} T_{f_i} + (2Nms^2 + 2ms + 2s + 2m) \cdot t_{op}. \qquad (3.52)$$

Время вычислений для алгоритма рис. 3.27 зависит от типа параллельной вычислительной системы. Для SIMD архитектур вычисление в общем случае различных компонент вектора функций F не может быть выполнено параллельно, следовательно, на это потребуется: $T_F = \sum_{i=1}^{m} T_{f_i}$. В то же время для MIMD систем с учетом возможности параллельной реализации обращения к правой части СОДУ, время вычислений ПНМРК равно:

$$T_{p,comp}^{22} = \underbrace{t_{mul} + t_{ad} + \left\lceil \frac{ms}{p} \right\rceil T_F}_{\bar{k}^{(0)}} + N \cdot \underbrace{\left\lceil \frac{ms}{p} \right\rceil [T_F + s(t_{mul} + t_{ad})]}_{\bar{k}^{(l)}} + \underbrace{\left\lceil \frac{m}{p} \right\rceil (s+1)(t_{mul} + t_{ad})}_{\bar{y}_{n+1}},$$

где $T_F = \max_{i=1}^{m} T_{f_i}$ (3.53)

Накладные расходы на обменные операции включают время на внутригрупповые и межгрупповые пересылки данных:

$$T_{p,comm}^{22} = T_{p,comm;1}^{22} + T_{p,comm;2}^{22}.$$

Проанализируем сложность коммуникационной составляющей параллельного алгоритма ПНМРК для СОДУ. Внутригрупповая операция обмена есть $N+1$ раз повторенная операция множественной

пересылки по типу “все-всем” на p_i процессорах, объем пересылаемых данных равен d: $T^{22}_{p,comm;1} = (N+1)\cdot T_{all-to-all}(d, p_i)$.

Для различных топологий внутригрупповой обмен требует:

– кольцо: $T^{22,R}_{p,comm;1} = (N+1)\cdot\left(t_s + \left\lceil\frac{ms}{p}\right\rceil\cdot t_w\right)\cdot\left(\left\lceil\frac{p}{s}\right\rceil - 1\right)$;

– тор: $T^{22,M}_{p,comm;1} = (N+1)\cdot\left[2\left(\sqrt{\left\lceil\frac{p}{s}\right\rceil} - 1\right)\cdot t_s + \left\lceil\frac{ms}{p}\right\rceil\cdot\left(\left\lceil\frac{p}{s}\right\rceil - 1\right)\cdot t_w\right]$;

– гиперкуб: $T^{22,H}_{p,comm;1} = (N+1)\cdot\left[log_2\left(\left\lceil\frac{p}{s}\right\rceil\right)\cdot t_s + \left\lceil\frac{ms}{p}\right\rceil\cdot\left(\left\lceil\frac{p}{s}\right\rceil - 1\right)\right]\cdot t_w$.

Межгрупповой обмен выполняется дважды для обмена стадийными коэффициентами и перегруппировки данных после вычисления очередной аппроксимации решения; он требует следующих временных затрат:

– кольцо: $T^{22,R}_{p,comm;21} = \left[2t_s + \left(\left\lceil\frac{m}{p}\right\rceil + \left\lceil\frac{ms}{p}\right\rceil\right)\right]\cdot t_w\cdot(p-1)$;

– тор: $T^{22,M}_{p,comm;21} = \left[4\left(\sqrt{p} - 1\right)\cdot t_s + \left(\left\lceil\frac{m}{p}\right\rceil + \left\lceil\frac{ms}{p}\right\rceil\right)\cdot(p-1)\cdot t_w\right]$;

– гиперкуб: $T^{22,H}_{p,comm;21} = 2\cdot log_2 p\cdot t_s + \left(\left\lceil\frac{m}{p}\right\rceil + \left\lceil\frac{ms}{p}\right\rceil\right)\cdot(p-1)\cdot t_w$.

Динамические характеристики полученного параллельного метода решения систем уравнений ПНМРК аналогичны соответствующим характеристикам интегрирования ОДУ.

3.3. Сравнительный анализ неявных одношаговых методов решения задачи Коши для ОДУ

Анализ эффективности определения апостериорной локальной погрешности в данном разделе базируется на применении следующих неявных одношаговых численных схем:

1) распараллеленных полностью неявных методов типа Рунге-Кутты;

2) параллельных блочных многоточечных методов.

Проведем сравнение двух классов неявных методов при последовательной и параллельной реализациях на основе наиболее эффективного способа оценки локальной погрешности, а, именно, вложенных формул. Основными параметрами, характеризующими ПНМРК, являются взаимосвязанные величины – порядок метода r и число стадий s.

Кроме того, при решении систем нелинейных алгебраических уравнений для определения стадийных коэффициентов появляется такой параметр, как необходимое число итераций L. Вычислительная сложность блочных методов зависит от порядка метода r, числа точек в блоке k, а также от числа итераций $\hat{L}$ для получения решения СНАУ необходимой точности. Временные затраты на вычисления, а также обменные операции при параллельной реализации описанных методов приведены в таблице 3.1.

Таблица 3.1

Временные характеристики s-стадийных ПНМРК и k-точечных блочных одношаговых методов

Методы		Коэффициент при T_F	Коэффициент при t_{op}	Коэффициент при $T_{all-to-all}$
ПНМРК (k - точек)	T_1	$ks(L+1)$	$2k(Ls^2+2s+1)$	
	T_p	$k(L+1)$	$2k(Ls+s+2)$	$k(L+1)$
Блочные методы	T_1	$\hat{L}k+1$	$2\hat{L}(k^2+2k)+3k$	
	T_p	$\hat{L}+1$	$2\hat{L}k+4k)+3$	$\hat{L}+1$

Для того, чтобы сравнение численных методов на базе неявных одношаговых схем было корректным, необходимо:

а) обеспечить один и тот же порядок точности, r;

б) получить решение в одинаковом числе новых точек, k;

в) порождаемые итерационные процессы должны реализовывать предельное число итераций, предусмотренное каждым из рассматриваемых методов: $L=2s$ и $\hat{L}=k$.

Для широко применяемых ПНМРК по теоремам Батчера число стадий связано с порядком метода одним из следующих соотношений

[94]: метод Гаусса $r = 2s$; методы Радо $r = 2s - 1$; методы Лобатто $r = 2s - 2$. Возьмем соотношение $r = 2s$, тем самым намеренно выбирая лучший вариант для ПНМРК. В то же время для блочных одношаговых методов порядок может быть определен через число точек одного блока: $r = k + 1$. Тогда число стадий ПНМРК и число точек в блоке многоточечного метода связаны следующим соотношением: $r = 2s = k + 1 \Rightarrow k = 2s - 1$. Используя полученные соотношения, приведем все временные характеристики к одному параметру. Пусть это будет число стадий s (табл. 3.2).

Таблица 3.2

Временные характеристики ПНМРК и блочных одношаговых методов, приведенные к одному параметру s

Методы		Коэффициент при T_F	Коэффициент при t_{op}	Коэффициент при $T_{all-to-all}$
ПНМРК (k - точек)	T_1	$4s^3 - s$	$8s^4 - 4s^3 + 8s^2 - 2$	
	T_p	$4s^2 - 1$	$8s^3 + 6s - 4$	$4s^2 - 1$
Блочные методы	T_1	$4s^2 - 4s + 2$	$16s^3 - 8s^2 + 2s - 1$	
	T_p	$2s$	$8s^2 + 1$	$2s$

Заметим, что среди множества неявных методов типа Рунге-Кутты для сравнения выбраны именно полностью неявные методы, поскольку они обладают идентичными характеристиками устойчивости, что и блочные одношаговые методы [113-115], а также в силу оптимального соотношения между порядком и числом стадий методов. Сравним времена выполнения последовательных алгоритмов рассматриваемых одношаговых неявных методов. При доминировании в вычислениях времени обращения к правой части ОДУ $T_F >> t_{op}$ объем вычислений для ПНМРК в s раз больше, чем для блочных методов:

$$T_1^{RK} / T_1^{BM} \approx \frac{ks(L+1)\cdot T_F}{(\hat{L}k+1)\cdot T_F} \approx \frac{s(4s^2-1)}{k^2+1} = \frac{4s^3-s}{4s^2-4s+2} \approx s.$$

Аналогично, для параллельного выполнения разработанных алгоритмов имеем следующее соотношение:

$$\frac{T_p^{RK}}{T_p^{BM}} \approx \frac{k(L+1)\cdot T_F}{(\hat{L}+1)\cdot T_F} \approx \frac{4s^2-1}{k+1} = \frac{4s^2-1}{2s} \approx 2s.$$

Для тривиальной правой части получается аналогичный результат, последовательная и параллельная реализации вычислений решения в новых k точках сетки интегрирования требуют больших накладных расходов для неявных методов Рунге-Кутты по сравнению с блочными одношаговыми методами (рис. 3.28). Заметим, что для параллельных алгоритмов эта разница увеличивается почти в два раза, что подтверждается экспериментом на тестовых задачах для СОДУ.

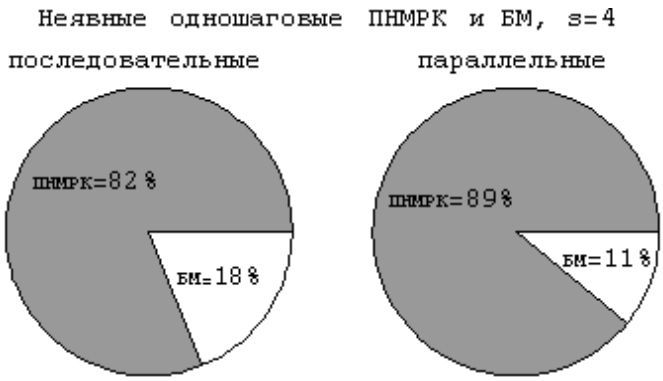
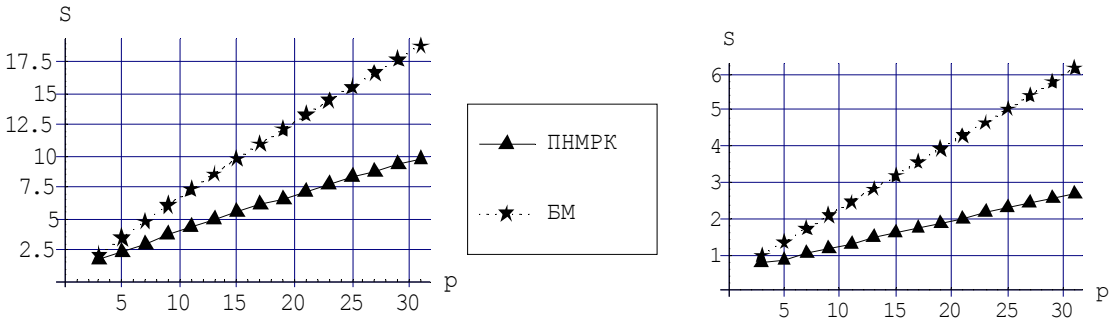


Рисунок 3.28 – Соотношение времен s-стадийных ПНМРК и k-точечных блочных вложенных одношаговых методов

Оценка эффективности распараллеливания производится на основе абсолютных коэффициентов ускорения вычислений (рис. 3.29), вычисленных по сравнению со своим последовательным алгоритмом.



$T_F >> t_{op}$ $T_F \approx t_{op}$

Рисунок 3.29 – Абсолютные коэффициенты ускорения вложенных ПНМРК и блочных одношаговых методов

Кроме того, с использованием относительных коэффициентов ускорения (рис. 3.30) по сравнению с наилучшим из рассмотренных последовательных алгоритмов, методом вложенных форм.

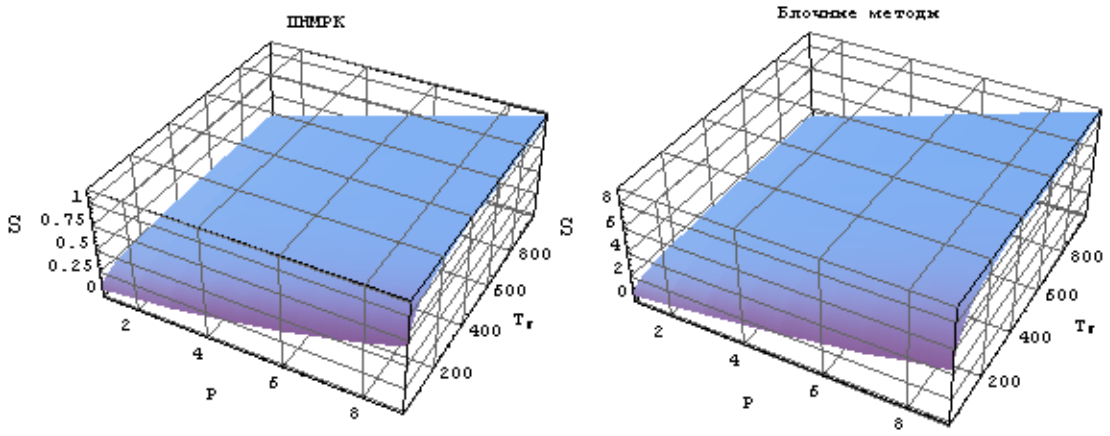


Рисунок 3.30 – Относительные коэффициенты ускорения вложенных ПНМРК и блочных одношаговых методов

В обоих случаях теоретические исследования, проведенный вычислительный эксперимент на системе тестов и реальной динамической задаче моделирования динамики шахтных подъемных установок позволяют сделать вывод, что для любых способов оценки локальной погрешности блочные многоточечные одношаговые методы являются наиболее эффективными с точки зрения временных затрат по сравнению с полностью неявными одношаговыми методами Рунге-Кутты.

3.4 Выводы

В третьей главе предложены и теоретически обоснованы новые параллельные методы оценки локальной апостериорной погрешности численного решения жестких задач Коши для одного дифференциального уравнения на основе блочных неявных одношаговых разностных схем:

1) блочный k-точечный метод с правилом дублирования шага;

2) вложенные формы на основе k и $(k+1)$-точечных блочных методов и с использованием метода последовательного повышения порядка точности;

3) локальная экстраполяция с блочным опорным методом.

Осуществлена модификация алгоритмов, реализующих разностные блочные методы распараллеливания для решения СОДУ, которая с точки зрения временных затрат позволяет с большей эффективностью по сравнению с существующими алгоритмами находить численное решение задачи Коши.

Построены итерационные параллельные алгоритмы численного решения нелинейной разностной задачи Коши, позволяющие получать результаты с заданной степенью точности. Приведены сравнительные характеристики численного решения тестовых задач и оценки параллелизма. Решение тестовых задач показало, что экспериментальные оценки ускорения и эффективности близки к потенциальным.

Приведены схемы отображения алгоритмов на параллельные вычислительные структуры, иллюстрирующие процессы сокращения объема последовательных вычислений. Для каждого из приведенных алгоритмов рассчитаны аналитические трудоемкости численного решения, получены экспериментальные характеристики параллелизма, которые свидетельствуют о высокой эффективности $(E > 0.7)$ предложенных методов.

Разработаны параллельные методы полностью неявных одношаговых схем типа Рунге-Кутты для одного дифференциального уравнения. Осуществлена модификация полученных методов с целью обобщения ПНМРК на системы ОДУ. Проведен сравнительный анализ эффективности неявных одношаговых методов решения общей задачи Коши на основе блочных k-точечных методов и s-стадийных ПНМРК одного и того же порядка точности для последовательной и параллельной реализаций. Установлено, что динамические характеристики блочных многоточечных методов превосходят соответствующие характеристики многостадийных одношаговых методов практически в s раз для последовательной и в $2s$ для параллельной реализаций.

Исследована эффективность полученных вычислительных схем отображения параллельных алгоритмов на структуры ВС в зависимости от размерности СОДУ, количества процессоров, типа параллельной ВС (SIMD, MIMD и кластерные системы), латентности и времени передачи данных в сетях различных топологий.

ГЛАВА 4

ПАРАЛЛЕЛЬНЫЕ МЕТОДЫ РЕШЕНИЯ ЛИНЕЙНОЙ ЗАДАЧИ КОШИ С ОЦЕНКОЙ ШАГОВОЙ ПОГРЕШНОСТИ ДЛЯ МУЛЬТИКОМПЬЮТЕРОВ

Данная глава посвящена построению и анализу эффективности параллельных алгоритмов решения **задачи Коши для систем линейных** или линеаризованных **обыкновенных дифференциальных уравнений (СЛОДУ) со встроенными способами оценки локальной апостериорной погрешности**. В частности, предложены специальные экспоненциальные методы решения однородных и неоднородных СЛОДУ с постоянными коэффициентами. Как показывают многочисленные исследования, учет специфики задачи позволяет строить более эффективные параллельные вычислительные алгоритмы, чем в случае применения общепринятых численных схем.

Общий вид задачи Коши для однородных СЛОДУ с постоянными коэффициентами:

$$\begin{cases} \overline{y}'(x) = A \cdot \overline{y}(x), \\ \overline{y}(x_0) = \overline{y}_0, \\ A = const, \end{cases} \tag{4.1}$$

где $\overline{y} = (y_1, y_2, ..., y_m)^T$ – вектор неизвестных,

$\overline{y}_0 = (y_{10}, y_{20}, ..., y_{m0})^T$ – вектор начальных условий,

$A = \|a_{ij}\|, i, j = \overline{1,m}$ – матрица коэффициентов линейной системы ОДУ, элементы матрицы являются константами.

Точным решением задачи Коши для СЛОДУ вида (4.1) является **матричная экспонента**:

$$\begin{cases} \overline{y}(x_n + h) = e^{hA} \cdot \overline{y}(x_n), \\ e^{hA} = F(hA) = \sum\limits_{i=0}^{\infty} \dfrac{(hA)^i}{i!}. \end{cases} \tag{4.2}$$

Приближенное решение можно построить, аппроксимировав матричную экспоненту отрезком её ряда Тейлора при малом h:

$$\begin{cases} \bar{y}_{n+1} = F_r(hA) \cdot \bar{y}_n, \\ F_r(hA) = \sum\limits_{i=0}^{r} \dfrac{(hA)^i}{i!}, \end{cases} \tag{4.3}$$

а затем, используя некоторый алгоритм умножения матриц, вычислить численное решение (4.1). Полученная вычислительная схема интегрирования однородных СЛОДУ с постоянными коэффициентами соответствует разностному методу решения порядка $O(h^r)$ [115].

Экспоненциальный метод относится к частным методам численного интегрирования задачи Коши для СЛОДУ, основан на точном представлении решения в аналитической форме и приближенном вычислении матричной экспоненты [115, 118]. Заметим, что в этой главе элементы матрицы Батчера будут обозначены, как c_{ij}, с целью сохранения устоявшихся обозначений коэффициентов при неизвестных для линейных систем ОДУ типа (4.1).

На основе экспоненциального метода построены три различных параллельных алгоритма с учетом альтернативных способов оценки локальной погрешности :

1) экспоненциальный и правило Рунге;

2) вложенный - экспоненциальный;

3) экспоненциальный с локальной экстраполяцией.

Матричная экспонента $F(hA)$ обладает следующим свойством: $F(hA) = F\left(\frac{h}{2}A\right) \cdot F\left(\frac{h}{2}A\right)$, что позволяет достаточно легко встраивать алгоритмы определения локальной апостериорной погрешности на основе дублирования шага и локальной экстраполяции в вычислительные схемы явных численных методов для линейных СОДУ. Применение экспоненциального метода для вложенных форм делает практически ненужными вычисления по формуле высшего порядка точности.

Предлагаемые алгоритмы особенно эффективны при решении задач с большой константой Липшица и требуют меньшего объема вычислений, чем стандартные методы решения линейной задачи Коши.

4.1. Параллельное решение линейной задачи Коши для СОДУ с оценкой шаговой погрешности по правилу Рунге

Для разработки параллельных алгоритмов решения линейной задачи Коши с альтернативными способами оценки локальной погрешности, используются специальные последовательные вычислительные схемы, основанные на определении матричной экспоненты (4.3-4.3).

Интегрирование **однородных СЛОДУ с постоянными коэффициентами на основе экспоненциального метода** и встроенного способа определения локальной погрешности на основе **дублирования шага** по правилу Рунге требует вычисления решения с удвоенным и дважды с половинным шагом:

$$\begin{cases} \overline{y}^{(1)}(x_n + 2h) = F(2hA) \cdot \overline{y}(x_n), \\ \overline{y}(x_n + h) = F(hA) \cdot \overline{y}(x_n), \\ \overline{y}^{(2)}(x_n + 2h) = F(hA) \cdot \overline{y}(x_n + h). \end{cases} \tag{4.4}$$

Используя аппроксимацию матричной формы F, а также ее свойства для половинного шага, получим следующую преобразованную вычислительную схему численного решения при переходе из точки x_n в точку x_{n+1}:

$$\begin{cases} \overline{y}_{n+1}^{(1)} = F^{(1)} \cdot \overline{y}_n = F_r(2hA) \cdot \overline{y}_n = \left(\sum_{i=0}^{r} \frac{(2hA)^i}{i!} \right) \cdot \overline{y}_n, \\ \overline{y}_{n+1}^{(2)} = F^{(2)} \cdot \overline{y}_n = F_r^2(hA) \cdot \overline{y}_n = \left(\sum_{i=0}^{r} \frac{(hA)^i}{i!} \right)^2 \cdot \overline{y}_n, \end{cases} \tag{4.5}$$

где $F_r(hA) = \left(E + hA + \frac{h^2 A^2}{2!} + \ldots + \frac{h^r A^r}{r!} \right)$ (4.6)

Основными преимуществами предложенного метода являются:

1) фактическое отсутствие вычислений для промежуточной точки с половинным шагом;

2) возможность введения подготовительного этапа, который будет выполняться до начала интегрирования, и вынесение в него

наиболее ресурсоемких операций нахождения матричных форм, а именно матричного умножения, не зависящего от шага интегрирования.

Процесс вычисления по формулам (4.5-4.6) можно представить, как решение следующих подзадач:

1) до начала интегрирования однократно выполняется вычисление степеней матрицы A и умножение их на скаляры;

2) на каждом из N шагов интегрирования вычисляются две матричные формы: $F^{(1)}$, $F^{(2)}$ и две аппроксимации решения: $\bar{y}_{n+1}^{(1)}$ и $\bar{y}_{n+1}^{(2)}$, а также величина d_n для оценки локальной погрешности.

Время выполнения последовательного алгоритма по экспоненциальной схеме с правилом Рунге состоит из времени на реализацию подготовительного этапа и времени на N шагов интегрирования. Подготовительный этап содержит вычисление степеней матриц коэффициентов: $\frac{A^2}{2!},\ldots,\frac{2^r A^r}{r!}$, время его выполнения в базовых операциях линейной алгебры составляет:

$$T_{1,0}^{11} = (r-1)\cdot T_1^{A\times A} + (2r-1)\cdot T_1^{c\cdot A},$$

где $T_{1,0}^{11}$ – время последовательной реализации подготовительного этапа для экспоненциального метода с правилом Рунге;

$T_1^{A\times A}$ – время вычисления скалярного произведения матриц;

$T_1^{c\cdot A}$ – время умножения матрицы на скаляр.

Один шаг интегрирования на однопроцессорной ВС по экспоненциальному методу + правило Рунге требует:

$$T_1^{11} = 3\cdot T_1^{A\times Y} + 2r\cdot T_1^{c\cdot A} + 2r\cdot T_1^{A+A} + T_1^{Y+Y},$$

где T_1^{11} – время последовательного выполнения одного шага интегрирования на базе экспоненциального метода с правилом Рунге;

$T_1^{A\times Y}$ – время умножения матрицы на вектор;

T_1^{Y+Y} – время сложения двух векторов;

T_1^{A+A} – время сложения двух матриц.

В таблице 4.1 приведены времена последовательной реализации матрично-векторных операций, используемые в разработанных вычислительных схемах, причем порядок матриц (векторов) равен m.

Таблица 4.1

Время последовательного выполнения матричных (векторных) операций

Вид операции	Время последовательного выполнения операции
Умножение матриц, $A \times A$	$m^3 \cdot t_{mul} + m^3 \cdot t_{ad}$
Умножение матрицы на вектор, $A \times Y$	$m^2 \cdot t_{mul} + m^2 \cdot t_{ad}$

Время последовательной реализации экспоненциального метода и правила Рунге с учетом подготовительного этапа и при условии использования RISC архитектуры равно:

$$T_{1,0}^{11} = [2(r-1)\cdot m^3 + (2r-1)\cdot m^2]\cdot t_{op}, \quad (4.7)$$

$$T_1^{11} = T_{1,0}^{11} + N\cdot(4rm^2 + 6m^2 + m)\cdot t_{op}, \quad (4.8)$$

где N – число шагов интегрирования.

Для построения параллельного экспоненциального алгоритма решения линейной задачи Коши для однородных СОДУ с постоянными коэффициентами + правило Рунге воспользуемся результатами главы 2. Из трех вариантов макросхем далее в рассмотрении будет анализироваться только первый вариант, поскольку он может быть применен для любой параллельной архитектуры (рис. 2.1а). Поскольку в линейной задаче Коши вид правой части СОДУ известен, то алгоритм решения может быть разбит на известные подзадачи линейной алгебры.

Совокупность подзадач реализуется через следующее множество макроопераций:

1) выполнение однократного матричного умножения;

2) вычисление матричной формы как функции шага интегрирования;

3) вычисление аппроксимации решения.

Все три макрооперации выполняются последовательно, первая $(r-1)$ раз до начала интегрирования, вторая и третья – на каждом шаге интегрирования дважды.

Параллельная реализация двух основных наиболее ресурсоемких макроопераций требует распараллеливания операций умножения двух матриц и умножения матрицы на вектор. Для рассматриваемых параллельных приложений топологическим решением, адекватно отображающим логическую связь между независимыми процессами, является квадратная сетка или ее замкнутый эквивалент – 2D-тор. На такой топологической схеме эффективно выполняются матричные операции, составляющие основу преобразованной формы (4.5-4.6). Здесь вычисление матричного умножения будет выполнено по **блочному варианту систолического алгоритма** или одной из модификаций **алгоритма Кеннона** (сдвигаются матрицы A и B, матрицы, являющиеся операндами матричного умножения) [70, 119].

Алгоритм первой основной макрооперации экспоненциального метода с правилом Рунге, а именно, умножения двух квадратных матриц $C = A \times B$, для мультикомпьютера из $p \times p$ процессоров приведен на рисунке 4.1.

Используемые в алгоритме решения СЛОДУ матрицы размерности $m \times m$ разбиваются на $q^2, q = p$ квадратных блоков порядка $k = \lceil m/q \rceil$. Для простоты рассуждений предположим, что $m \vdots q$ и $m \vdots k$. Заметим, что в нашем случае, $B = A$, то есть эта операция используется для нахождения степеней матрицы коэффициентов СЛОДУ. Тогда, с учетом обозначений операцию **умножения матриц** A и B в **блочном виде** можно представить следующим образом:

$$\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1q} \\ \dots & \dots & \dots & \dots \\ A_{q1} & A_{q2} & \dots & A_{qq} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} & \dots & B_{q1} \\ \dots & \dots & \dots & \dots \\ B_{q1} & B_{q2} & \dots & B_{qq} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & \dots & C_{q1} \\ \dots & \dots & \dots & \dots \\ C_{q1} & C_{q2} & \dots & C_{qq} \end{pmatrix},$$

где каждый блок C_{ij} матрицы C определяется в соответствии с выражением: $C_{ij} = \sum_{l=1}^{q} A_{il} \cdot B_{lj}$.

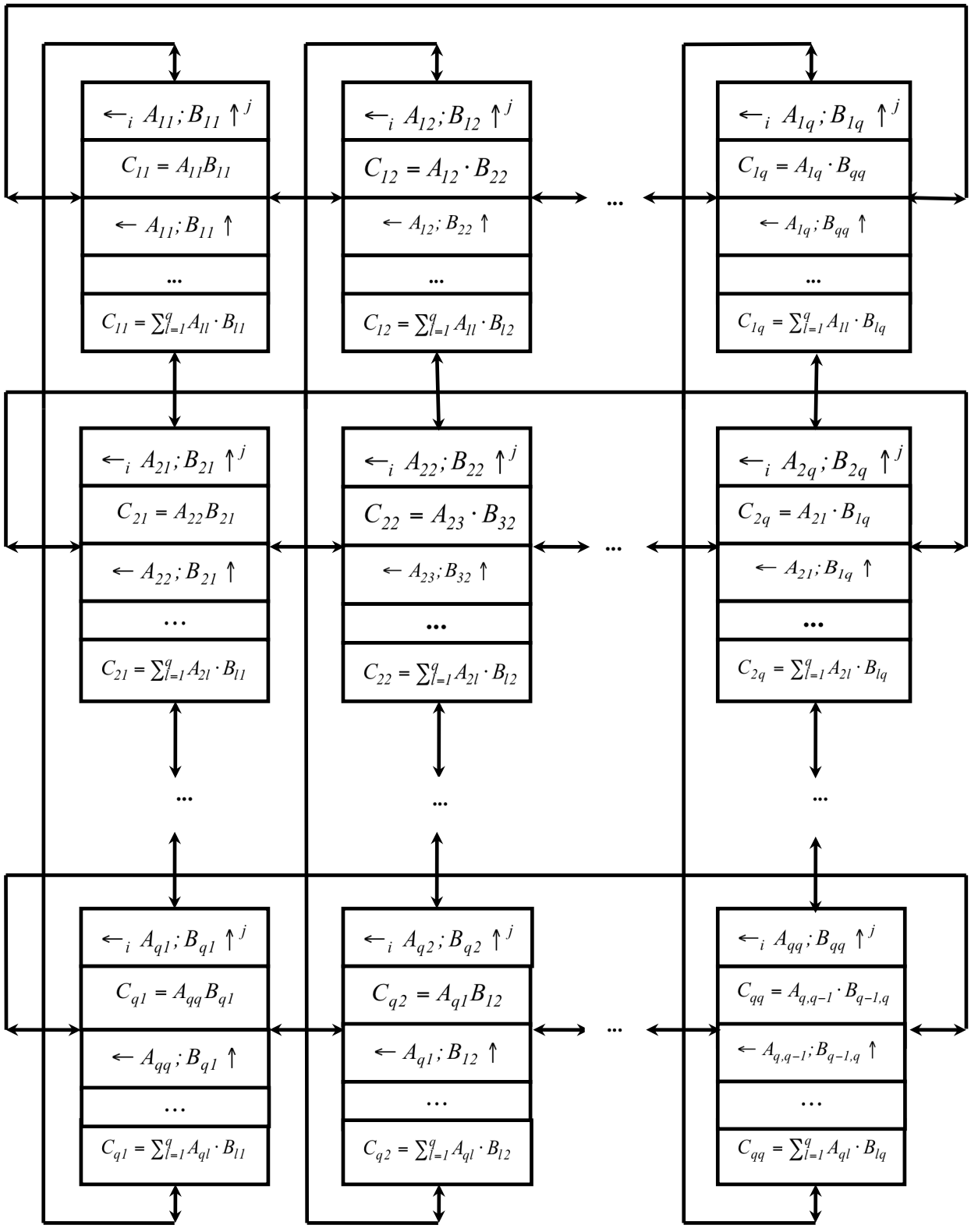


Рисунок 4.1 – Параллельный алгоритм блочного систолического умножения матриц, $A \underset{bl-sys}{\times} B = C$

Распределение исходных данных и результатов по процессорам следующее: каждый процессор решетки с номером $< i, j >$ содержит два блока исходных матриц A_{ij}, B_{ij}, и отвечает за вычисление блока матрицы результата C_{ij}. Предварительно, осуществляется косой сдвиг влево по строкам для блоков матрицы $A : \leftarrow_i A$ и косой сдвиг по столбцам вверх для блоков матрицы $B : B \uparrow^j$.

Иллюстрация операций **косой сдвиг влево по строкам** для блоков матрицы A:

$$\leftarrow_i A = \leftarrow_i \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1q} \\ A_{21} & A_{22} & \dots & A_{2q} \\ \dots & \dots & \dots & \dots \\ A_{q1} & A_{q2} & \dots & A_{qq} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1q} \\ A_{22} & A_{23} & \dots & A_{21} \\ \dots & \dots & \dots & \dots \\ A_{qq} & A_{q1} & \dots & A_{qq-1} \end{bmatrix},$$

и **косой сдвиг по столбцам вверх** для блоков матрицы B:

$$B \uparrow^j = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1q} \\ B_{21} & B_{22} & \dots & B_{2q} \\ \dots & \dots & \dots & \dots \\ B_{q1} & B_{q2} & \dots & B_{qq} \end{bmatrix} \uparrow^j = \begin{bmatrix} B_{11} & B_{22} & \dots & B_{qq} \\ B_{21} & B_{32} & \dots & B_{1q} \\ \dots & \dots & \dots & \dots \\ B_{q1} & B_{12} & \dots & B_{q-1q} \end{bmatrix}.$$

Далее над блоками выполняются в точности те же действия, которые выполняются систолическим алгоритмом над элементами матриц, причем сложение и умножение матричных блоков выполняются на одном процессоре. То есть на каждом из $q = p$ шагов выполняется умножение блоков матриц A и B по стандартному последовательному алгоритму. Затем производится одиночный сдвиг влево для блоков матрицы $A : \leftarrow A$ и вверх для блоков матрицы B: $B \uparrow$, и описанные действия повторяются. В результате на каждом процессоре с номером $< i, j >$ получается блок матрицы результата C_{ij}.

Время реализации арифметических операций при блочном систолическом умножении определяется, как время q раз выполненной операции умножения блоков матриц размерности $k \times k$ и равно:

$$T_{p,comp}^{A \times B\ bl-sys} = q \cdot \left(T_{p,comp}^{A_{ij} \times B_{ij}} \right),$$

где $T_{p,comp}^{A_{ij} \times B_{ij}}$ – время умножения блоков матриц исходных данных размерности $k \times k$.

В свою очередь, время вычисления умножения блоков матриц равно:

$$T_{p,comp}^{A_{ij} \times B_{ij}} = k^2 \cdot \left(k \cdot t_{mul} + k \cdot t_{ad} \right) = \frac{m^3}{p^3} \cdot (t_{mul} + t_{ad}),$$

где $k^2 = m^2 / p^2$ – количество элементов блока,

$k \cdot \left(t_{mul} + t_{ad} \right)$ – время, необходимое для вычисления одного элемента блока.

Заметим, что нет необходимости в отдельном выполнении операции сложения блоков для формирования одного блока результата C_{ij}, она совмещается со сложением внутри процедуры умножения блоков. Таким образом, общее время на реализацию вычислений блочного систолического алгоритма умножения матриц равно:

$$T_{p,comp}^{A \times B\ bl-sys} = (2m^3 / p^2) \cdot t_{op}.$$

Время на предварительную рассылку данных состоит из времени выполнения косого сдвига по строкам влево для матрицы A и косого сдвига по столбцам вверх для матрицы B:

$$T_{p,comm,0}^{A \times B\ bl-sys} = 2(p-1) \cdot [t_s + (m^2 / p^2) \cdot t_w].$$

Дополнительно, на каждом из $p-1$ шагов алгоритма производится пересылка блоков матрицы A влево и матрицы B вверх соседним процессорным элементам замкнутой решетки.

Общее время на обменные операции по блочному систолическому алгоритму составляет:

$$T_{p,comm}^{A \times B\ bl-sys} = 4(p-1) \cdot [t_s + (m^2 / p^2) \cdot t_w].$$

Алгоритм второй основной макрооперации экспоненциального метода решения СЛОДУ с правилом Рунге заключается в выполнении операции **блочного систолического умножения матрицы** A размерности $m \times m$ **на вектор** Y размерности m при наличии $p \times p$ процессоров (рис. 4.2).

Пусть $C_i = \sum_{l=1}^{q} A_{il} \cdot Y_l, i = \overline{1,q}$,

где q^2 – количество блоков, $q = p$,

k – размерность блока $k = \lceil m / q \rceil$,

$Y_1 = (y_1,...,y_k)$, ..., $Y_q = (y_{m-k+1},...,y_m)$ – подвектора размерности k вектора $Y = \|y_i\|, i = \overline{1,m}$.

Первоначально, блоки вектора Y распределяются на все процессоры исходной вычислительной системы. Процессор с номером $< i, j >$ содержит блок матрицы $A : A_{ij}$ и часть вектора $Y : Y_i$. Производится косой сдвиг вверх по столбцам для блоков вектора Y: $Y_i \uparrow^j$, косой сдвиг влево по строкам для блоков матрицы $A : \leftarrow_i A_{ij}$. После этого выполняется умножение каждого блока матрицы на подвектор параллельно всеми процессорами. В итоге на основе **алгоритма каскадного суммирования** получаем матрицу, каждая строка которой содержит соответствующий подвектор результата.

Время выполнения арифметических операций для описанного алгоритма состоит из времени вычисления однократного умножения $A_{ij} \cdot Y_i$ и реализации каскадного суммирования подвекторов и равно (см. табл. 4.2):

$$T_{p,comp}^{A \times Y\ bl-sys} = [(2m^2 / p^2) + (m / p) \cdot \lceil log_2 p \rceil] \cdot t_{op}.$$

Время обменов включает первоначальный этап, состоящий из пересылок элементов при дублировании подвекторов Y и обменов блоками при косом сдвиге для матрицы A:

$$T_{p,comm,0}^{A \times Y\ bl-sys} = 2(p-1) \cdot [t_s + \left(m^2 / p^2 + m / p\right) \cdot t_w].$$

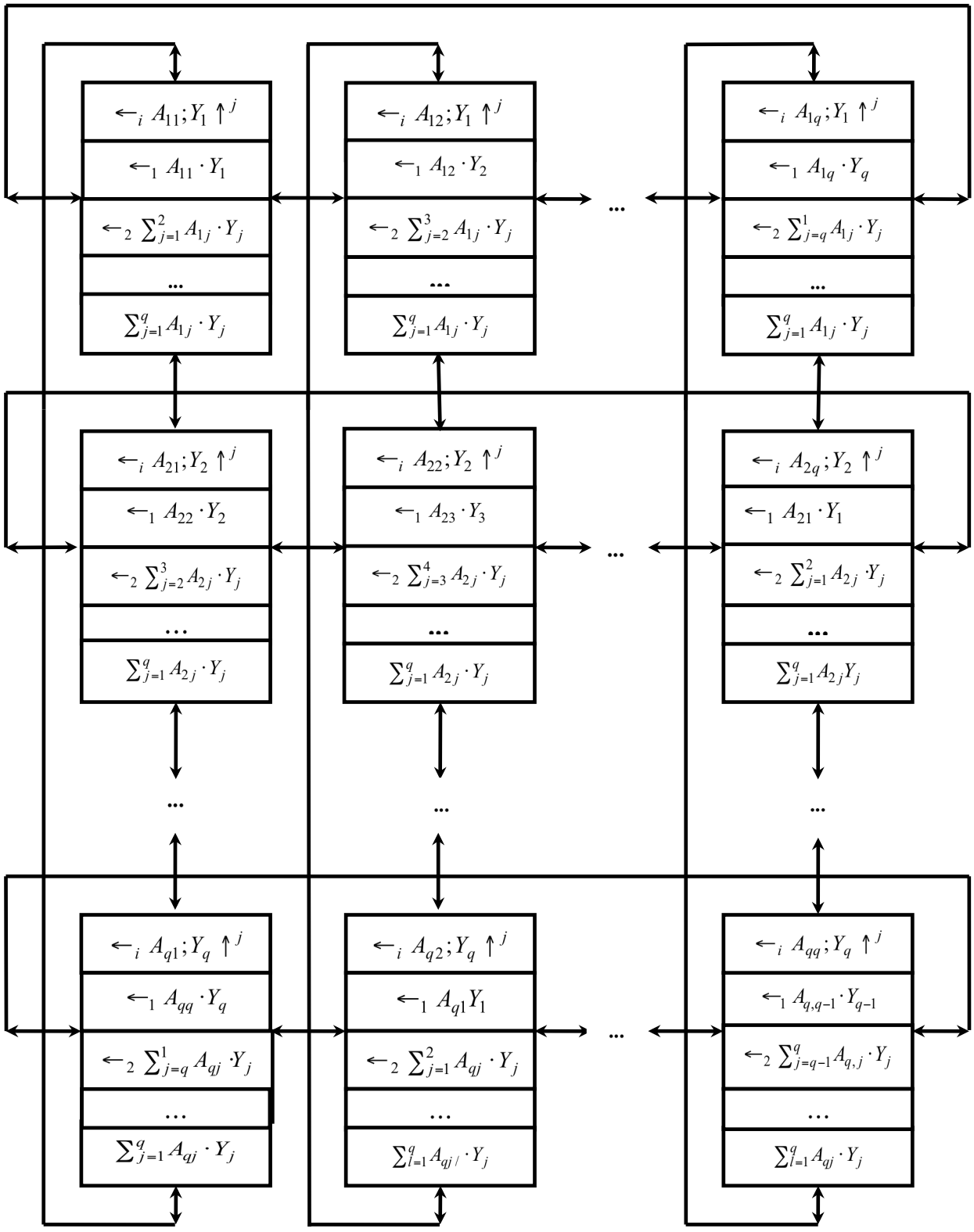


Рисунок 4.2 – Параллельный алгоритм блочного систолического умножения матрицы на вектор, $A \underset{bl-sys}{\times} Y$

Таблица 4.2

Время параллельного выполнения матричных/векторных умножений для блочного систолического алгоритма

Вид операции	Время вычислений в t_{op}
Умножение матриц, $A \times A$	$2m^3 / p^2$
Умножение матрицы на вектор, $A \times Y$	$\frac{2m^2}{p^2} + \frac{m}{p}\lceil log_2\, p \rceil$

Для данного алгоритма на каждом шаге интегрирования осуществляется косой сдвиг по столбцам подвекторов вектора решения Y и обмены при каскадном суммировании (табл. 4.3):

$$T^{A \times Y}_{p,comm\ bl-sys} = \left(2^{\lceil log_2\, p \rceil} + p - 2\right) \cdot [t_s + (m/p) \cdot t_w].$$

Таблица 4.3

Время коммуникационных операций параллельного выполнения матричных /векторных умножений на основе блочного систолического алгоритма при топологии 2D-тор

Вид операции	Время коммуникационных операций	
	0 шаг	i – й шаг
Умножение матриц, $A \times A$	$2(p-1)\left(t_s + \frac{m^2}{p^2} t_w\right)$	$2(p-1)\left(t_s + \frac{m^2}{p^2} t_w\right)$
Умножение матрицы на вектор, $A \times Y$	$2(p-1)\left(t_s + \left(\frac{m^2}{p^2} + \frac{m}{p}\right) t_w\right)$	$\left(2^{\lceil log_2\, p \rceil} + p - 2\right)\left(t_s + \frac{m}{p} t_w\right)$

Рассмотрим параллельные методы решения линейных систем ОДУ в совокупности с правилом Рунге на базе введенных выше алгоритмов базовых макроопераций.

Первоначальное распределение данных определяется особенностями выполнения операций блочного систолического умножения. Параллельный алгоритм решения СЛОДУ + правило Рунге на основе экспоненциального метода (рис. 4.3) содержит 0-ой этап, который выполняется один раз до начала интегрирования и вычисляет блоки матриц: $A^2, A^3, ..., A^r$. Для этого на каждом процессоре с номером $< i, j >$ замкнутой решетки $p \times p$ располагаются блоки матрицы коэффициентов при неизвестных A_{ij}, $i, j = \overline{1,q}; q = p$.

По блочному систолическому алгоритму умножения, $(r-1)$ раз выполняется вычисление соответствующих степеней блоков матрицы A и $(2r-1)$ раз умножение матриц на константы. Причем процессор p_{ij} будет содержать блоки $< i, j >$ матриц $A^2/2!, ..., (2A)^r/r!$.

Время вычислений и время обменов для подготовительного этапа:

$$T^{11}_{p,comp,0} = \left[(r-1)\left(\frac{2m^3}{p^2}\right) + (2r-1)\frac{m^2}{p^2}\right] \cdot t_{op}, \tag{4.9}$$

$$T^{11}_{p,comm,0} = (r-1)(3p-2) \cdot \left(t_s + \left(m^2/p^2\right) \cdot t_w\right). \tag{4.10}$$

Затем выполняется N шагов интегрирования, реализующих вычисление аппроксимаций решения через матричные формы. На рисунке 4.3 продемонстрирован один такой шаг. Последовательное вычисление двух матричных форм и соответствующих аппроксимаций решения компенсируется параллельными вычислениями над блоками матриц/векторов исходных и промежуточных данных.

На каждом процессоре с номером $< i, j >$ замкнутой решетки $p \times p$ располагаются вычисленные блоки матрицы коэффициентов при неизвестных, а также блок вектора решения предыдущего шага интегрирования. Сначала на процессоре p_{ij} вычисляется блок матричной формы $F^{(1)}(2hA)$ и, используя операцию блочного систолического умножения матрицы на вектор, получаем аппроксимацию решения с удвоенным шагом $Y^{(1)}_{n+1}$.

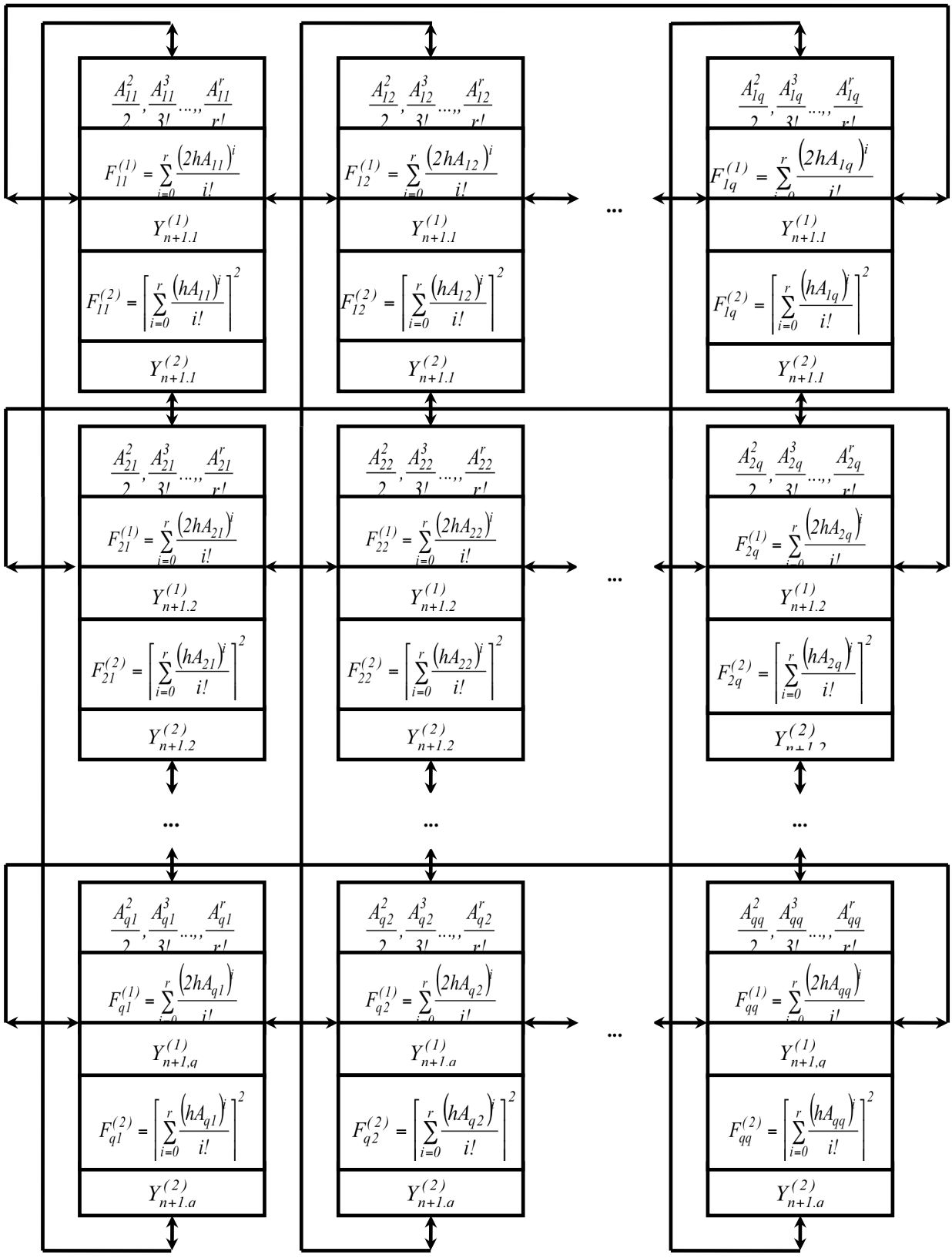


Рисунок 4.3 – Параллельный экспоненциальный алгоритм решения СЛОДУ с оценкой погрешности по правилу Рунге

Причем процессор p_{ij} будет содержать только $i-$й блок вектора решения за счет выполнения операции каскадного суммирования. Такое разбиение данных позволяет избежать дополнительных пересылок и обеспечить равномерную загрузку процессоров. Аналогично, вычисляются блоки матричной формы $F^{(2)}$ и аппроксимация решения с половинным шагом $Y_{n+1}^{(2)}$.

Вычислительная сложность параллельного алгоритма определяется выражением (с учетом 3 умножений матрицы на вектор,):

$$T_{p,comp}^{11} = \left[(4r+6)\frac{m^2}{p^2} + (1+3\lceil log_2 p\rceil)\frac{m}{p}\right]\cdot t_{op}, \qquad (4.11)$$

в свою очередь, коммуникационная составляющая равна:

$$T_{p,comm}^{11} = 2(p-1)\cdot\left(t_s + \frac{m^2}{p^2}t_w\right) + \left(2^{\lceil log_2 p\rceil+1} + p - 3\right)\cdot\left(t_s + \frac{m}{p}t_w\right). \qquad (4.12)$$

Для оценки эффективности разработанного экспоненциального алгоритма решения линейной задачи Коши проведем сравнение с общепринятой схемой вычислений на базе $s-$ стадийного ЯМРК аналогичного порядка точности $O(h^r)$. Правило Рунге для стандартного явного одношагового метода интегрирования в применении к однородной СЛОДУ с постоянными коэффициентами реализуется через схему:

$$\begin{cases} \overline{y}_{n+1}^{(1)} = \overline{y}_n + 2h\cdot\sum\limits_{i=1}^{s} b_i\cdot\overline{k}_i^{(1)}(2h), \quad \overline{k}_i^{(1)} = A\cdot\left(\overline{y}_n + 2h\cdot\sum\limits_{l=1}^{i-1} c_{il}\cdot\overline{k}_l^{(1)}\right), \\ \overline{y}_{n+\frac{1}{2}} = \overline{y}_n + h\cdot\sum\limits_{i=1}^{s} b_i\cdot\overline{k}_i(h), \quad \overline{k}_i = A\cdot\left(\overline{y}_n + h\cdot\sum\limits_{l=1}^{i-1} c_{il}\cdot\overline{k}_l\right), \\ \overline{y}_{n+1}^{(2)} = \overline{y}_{n+\frac{1}{2}} + h\cdot\sum\limits_{i=1}^{s} b_i\cdot\overline{k}_i^{(2)}(h), \quad \overline{k}_i^{(2)} = A\cdot\left(\overline{y}_{n+\frac{1}{2}} + h\cdot\sum\limits_{l=1}^{i-1} c_{il}\cdot\overline{k}_l^{(2)}\right), \\ d_n = \left\|\overline{y}^{(2)} - \overline{y}^{(1)}\right\|; \ i = \overline{1,s}. \end{cases} \qquad (4.13)$$

Распараллеливание вычислительной схемы (4.13) концентрируется на выполнении 1 шага интегрирования (рис. 4.4).

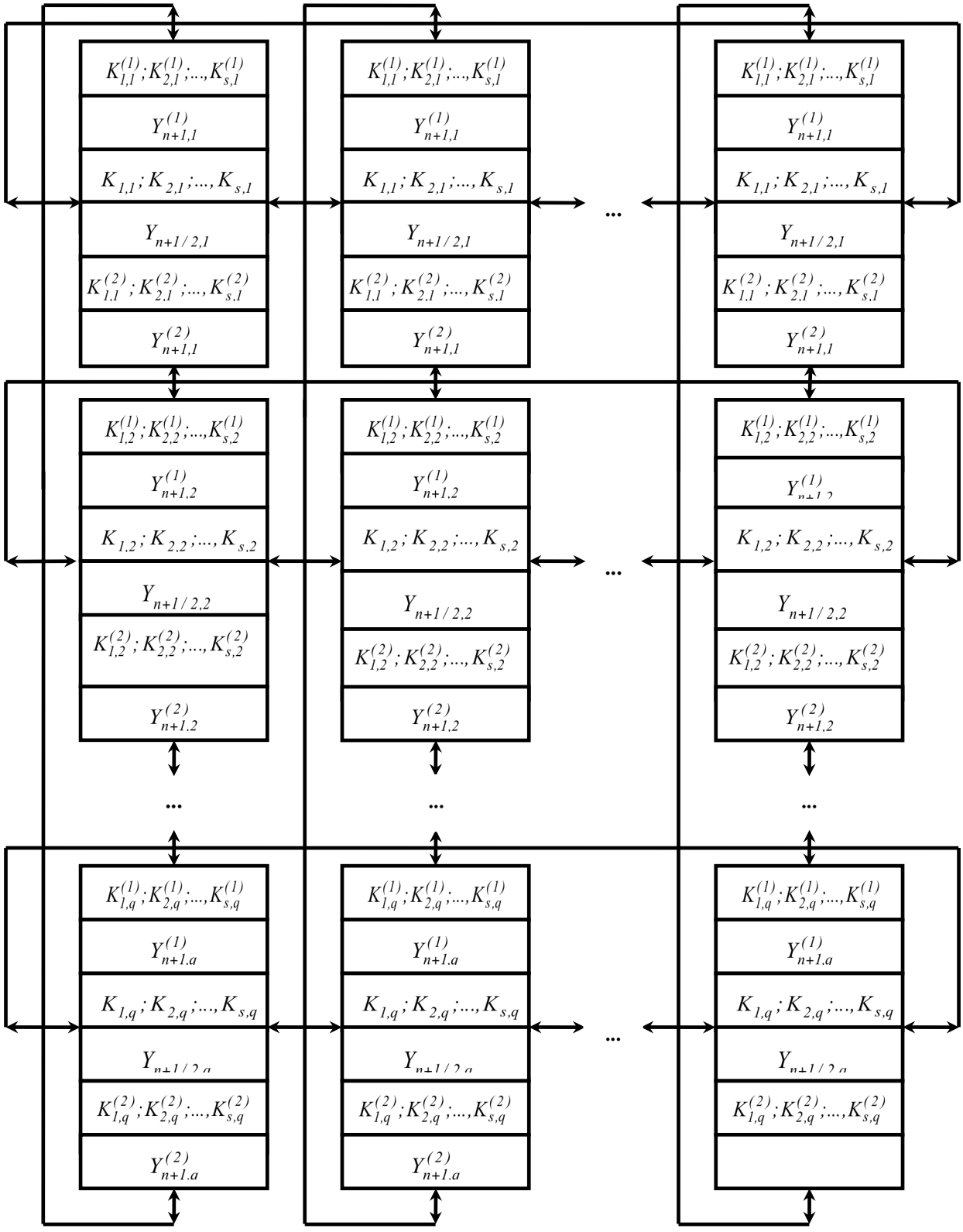


Рисунок 4.4 – Параллельный алгоритм решения СЛОДУ с правилом Рунге на базе стандартного явного одношагового метода

На каждом процессоре с номером $<i,j>$ замкнутой решетки располагаются блоки матрицы коэффициентов при неизвестных $A_{ij},\ i,j=\overline{1,q};\ q=\sqrt{p}$ и блок вектора решений предыдущего шага интегрирования: $Y_{n,i}$. Используя операцию блочного систолического умножения матрицы на вектор, получаем пошагово s стадийных векторов. Причем процессор p_{ij} будет вычислять только $i-$й блок векторов: $K_{1,i}^{(1)},K_{2,i}^{(1)},...,K_{s,i}^{(1)}$, который участвует в вычислении соответствующего блока вектора решения $Y_{n+1,i}^{(1)}$. Аналогично, вычисляются аппроксимации решения $Y_{n+1/2},Y_{n+1}^{(2)}$.

Время реализации параллельного алгоритма, описываемого схемой рисунка 4.4, в базовых операциях линейной алгебры составляет:

$$T_p^{12}=(3s-1)\cdot T_p^{A\times Y}+(1.5s^2+2)\cdot T_p^{Y+Y}+(1.5s^2+4.5s+1)\cdot T_p^{c\cdot Y},\quad (4.14)$$

где $T_p^{A\times Y}$ – время параллельного умножения матрицы на вектор,

T_p^{Y+Y} – время параллельного сложения двух векторов,

$T_p^{c\cdot Y}$ – время параллельного умножения вектора на скаляр.

Для сравнения асимптотической сложности двух параллельных алгоритмов решения СЛОДУ с правилом Рунге (рис. 4.3-4.4) ограничимся учетом наиболее ресурсоемкой операции, а, именно, умножения матрицы на вектор. Тогда, справедливо соотношение:

$$T_p^{12}/T_p^{11}\approx\frac{3s-1}{3}\approx s,$$

то есть **стандартная схема решения обладает вычислительной сложностью в s раз большей, чем экспоненциальная**.

Формула (4.14) определяет нижнюю границу времени выполнения параллельного алгоритма из-за соотношения между порядком, r и числом стадий s явной одношаговой схемы:

$$s=r+c,\ c=\begin{cases}0 & r\le 4;\\ 1,2,.. & r>4.\end{cases}$$

Аналогичный результат имеет место при учете полного времени выполнения рассматриваемых алгоритмов, вычисляемого экспериментальным путем (рис. 4.5).

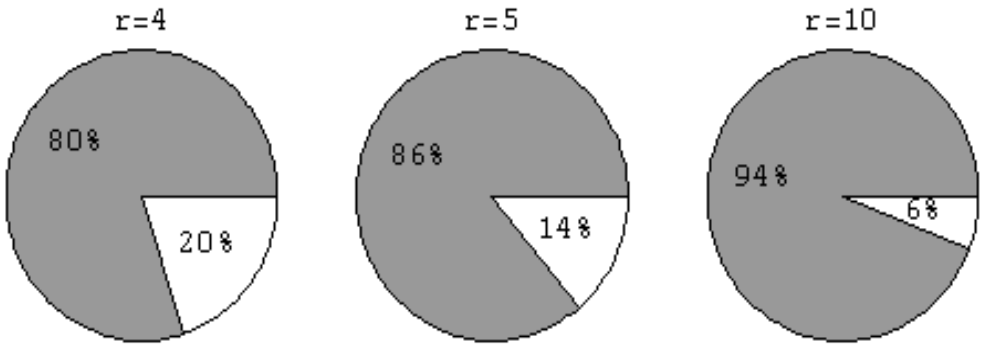


Рисунок 4.5 – Соотношение времен реализаций явной одношаговой, T_p^{12} и экспоненциальной, T_p^{11} схем с правилом Рунге от порядка метода

Таким образом, выражение (4.14) определяет нижнюю границу времени выполнения данного параллельного алгоритма и для методов высоких порядков временные затраты на параллельную реализацию стандартной схемы больше, чем в s раз превышают соответствующие затраты на экспоненциальную схему. Более того, это расхождение увеличивается с ростом порядка метода: $r\uparrow \Rightarrow s\uparrow \Rightarrow c\uparrow \Rightarrow \left(T_p^{12}/T_p^{11}\right)\uparrow$.

4.2. Вложенные параллельные методы численного решения систем линейных обыкновенных дифференциальных уравнений

Применение **идеи вложенных форм** в сочетании с экспоненциальным методом решения предполагает вычисление двух аппроксимаций решения смежных порядков точности в одной точке интегрирования: $\overline{y}_{n+1}$ порядка $O(h^r)$ и $\widehat{\overline{y}}_{n+1}$ порядка $O(h^{r+1})$ с использованием матричной экспоненты:

$$\begin{cases} \widehat{\overline{y}}_{n+1}(x_n+h;\ \overline{y}_n) = F_{r+1}(hA)\cdot \overline{y}_n, \\ \overline{y}_{n+1}(x_n+h;\ \overline{y}_n) = F_r(hA)\cdot \overline{y}_n,. \end{cases} \qquad (4.15)$$

Как правило, формула более высокого, $(r+1)$-го, порядка используется только для оценки локальной погрешности, а в качестве аппроксимации решения на шаге интегрирования берется формула r-го порядка. Поэтому вычислительная схема вложенного экспоненциального метода после элементарных преобразований принимает вид:

$$\begin{cases} \bar{y}_{n+1} = \left(E + hA + \dfrac{h^2A^2}{2!} + \ldots + \dfrac{h^rA^r}{r!} \right) \cdot \bar{y}_n; \\ d_n = \left\| \dfrac{h^{r+1}}{(r+1)!} A^{r+1} \cdot \bar{y}_n \right\|. \end{cases} \tag{4.16}$$

Вычислительная схема (4.16) позволяет уменьшить время каждого шага интегрирования за счет создания подготовительного этапа, который будет выполняться до начала основного счета и содержать операции, не связанные непосредственно с номером шага. Для вложенных методов величина шага интегрирования h – не есть постоянная величина. Следовательно, основными операциями этого этапа будут только вычисления степеней матрицы коэффициентов однородной системы A и умножение их на константы. Заметим, что эти операции являлись наиболее трудоемкими в схеме (4.16).

Время вычисления решения в точке x_{n+1} по преобразованной схеме вложенного экспоненциального метода для однопроцессорной ВС можно представить, как сумму двух слагаемых: времени выполнения подготовительного этапа и, собственно, шага интегрирования. Время подготовительного этапа вычисляется, как:

$$T_{1,0}^{21} = \left(2m^3 + m^2\right) \cdot r \cdot t_{op}. \tag{4.17}$$

Время последовательного выполнения одного шага интегрирования по схеме (4.16) определяется с учетом того факта, что нет необходимости в вычислении обоих приближений, достаточно знать $\bar{y}^{n+1}$ и d_n:

$$T_1^{21} = 2 \cdot T_1^{A \times Y} + (r+1) \cdot T_1^{A+A} + (r+1) \cdot T_1^{c \cdot A},$$

$$T_1^{21} = \left(2m^2 r + 6m^2\right) \cdot t_{op}. \tag{4.18}$$

Построение параллельного алгоритма для рассмотренной вычислительной схемы, как и в предыдущих случаях, основывается на декомпозиционной методике. Вне зависимости от наличия или отсутствия свойства линейности для задачи Коши множество макроопераций, а также граф влияния для макрооперационной схемы алгоритма явных вложенных методов (рис. 2.14) и схемы (4.16) совпадают. В свою очередь каждая из макроопераций может быть разбита на подзадачи линейной алгебры, параллельные алгоритмы которых приведены в подразделе 4.1 (рис. 4.1, 4.2).

Параллельная реализация вложенного экспоненциального метода также включает подготовительный этап вычисления степеней матрицы коэффициентов и, собственно, N шагов интегрирования. На рисунке 4.6 показан один шаг интегрирования алгоритма, состоящий из вычисления матричной формы, аппроксимации решения $r-$го порядка и величины d_n. До начала интегрирования (на нулевом шаге, ровно один раз) каждый процессор p_{ij} вычисляет соответствующий блок $<i,j>$ степеней $i=\overline{2,r+1}$ матрицы коэффициентов СЛОДУ на основе параллельного блочного систолического алгоритма умножения квадратных матриц и умножения их на константы:

$$\begin{pmatrix} \frac{A_{11}^2}{2!},...,\frac{A_{11}^r}{r!},\frac{A_{11}^{r+1}}{(r+1)!} & ... & \frac{A_{1q}^2}{2!},...,\frac{A_{1q}^r}{r!},\frac{A_{1q}^{r+1}}{(r+1)!} \\ ... & ... & ... \\ \frac{A_{q1}^2}{2!},...,\frac{A_{q1}^r}{r!},\frac{A_{q1}^{r+1}}{(r+1)!} & ... & \frac{A_{qq}^2}{2!},...,\frac{A_{qq}^r}{r!},\frac{A_{qq}^{r+1}}{(r+1)!} \end{pmatrix}.$$

На произвольном шаге интегрирования каждый процессор с номером $<i,j>$ вычисляет соответствующий блок формы F_r, а каждый процессор строки $<i>$ вычисляет подвекторы вектора решения $Y_{n+1}=(Y_{n+1,1},Y_{n+1,2},...,Y_{n+1,q})$ и величины d_n: $D_n=(D_{n,1},....,D_{n,q})$.

Время вычислений подготовительного этапа рассмотренного алгоритма:

$$T_{p,comp,0}^{21}=(2m^3/p^2)r\cdot t_{op}, \qquad (4.19)$$

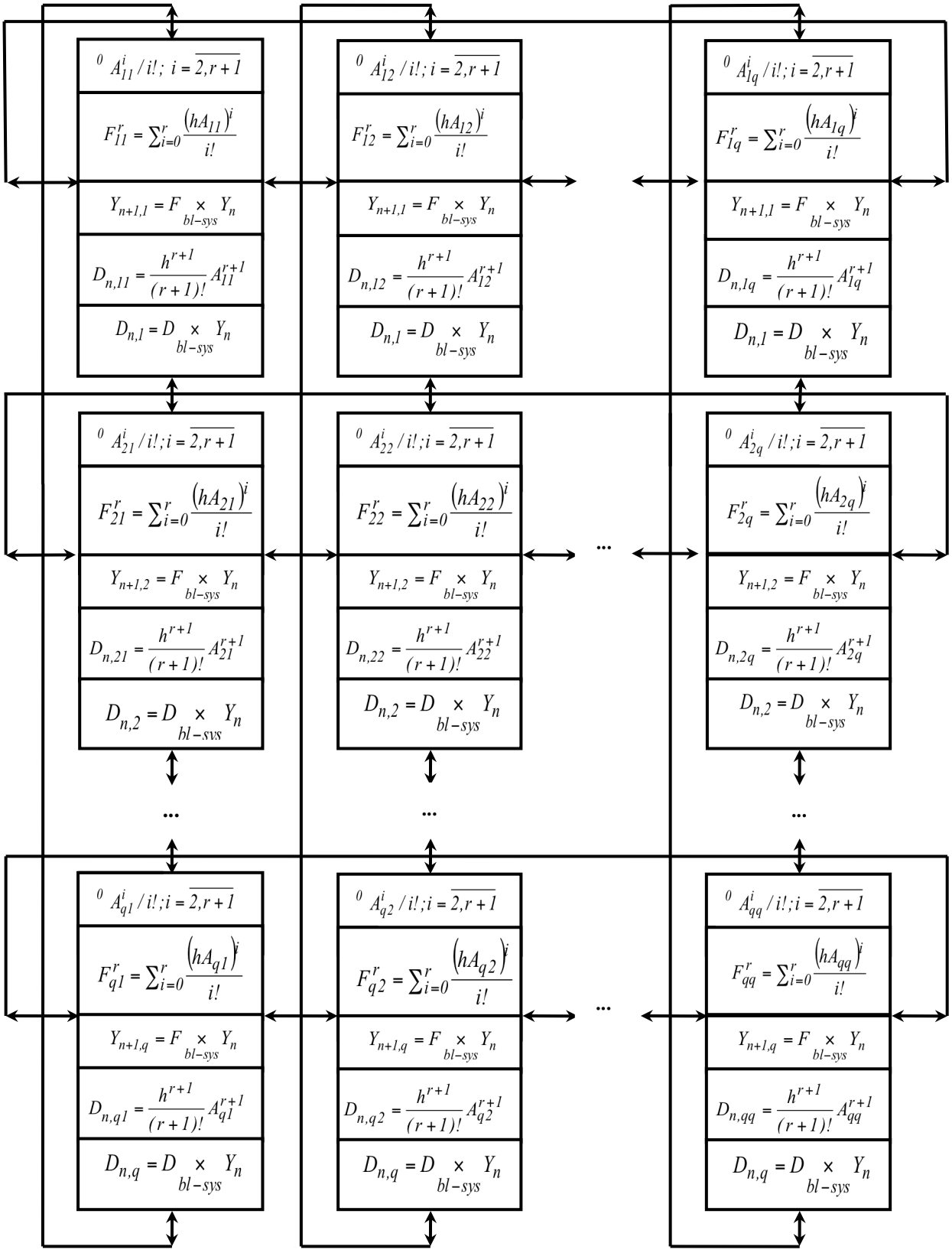


Рисунок 4.6 – Вычислительная схема параллельного алгоритма решения СЛОДУ на базе вложенного экспоненциального метода

Коммуникационная составляющая подготовительного этапа рассмотренного алгоритма составляет:

$$T_{p,comm,0}^{21} = 2(p-1)r \cdot [t_s + (m^2/p^2) \cdot t_w]. \quad (4.20)$$

Время параллельной реализации одного шага интегрирования для вложенного экспоненциального метода в операциях линейной алгебры:

$$T_p^{21} = 2 \cdot T_p^{A \times Y} + (r+1) \cdot T_p^{A+A} + (r+1) \cdot T_p^{c \cdot A}, \quad (4.21)$$

где T_p^{A+A} – время параллельного сложения матриц;

$T_p^{c \cdot A}$ – время параллельного умножения матрицы на константу.

Соответственно, время вычислений одного шага для вложенного экспоненциального метода равно:

$$T_{p,comp}^{21} = \left[(2r+6) \cdot \frac{m^2}{p^2} + 2m\lceil log_2 p \rceil \cdot / p \right] \cdot t_{op}, \quad (4.22)$$

а время обменов составляет:

$$T_{p,comm}^{21} = 2 \cdot \left(2^{\lceil log_2 p \rceil} + p - 2 \right) \cdot \left(t_s + m \cdot t_w / p \right). \quad (4.23)$$

Для сравнения приведем второй алгоритм решения СЛОДУ с постоянными коэффициентами по методу вложенных форм, основанный на использовании стандартной явной одношаговой схемы:

$$\begin{cases} \bar{y}_{n+1} = \bar{y}_n + h \cdot \sum_{i=1}^{s} b_i \cdot \bar{k}_i, \;\; \hat{\bar{y}}_{n+1} = \bar{y}_n + h \cdot \sum_{i=1}^{s'} \hat{b}_i \cdot \bar{k}_i, \\ \bar{k}_l = A \cdot \bar{y}_n + h \cdot \bar{g}_l, \;\; \bar{g}_l = A \cdot \sum_{i=1}^{l-1} c_{li} \cdot \bar{k}_i, l = \overline{1,s'}. \end{cases} \quad (4.24)$$

Последовательный алгоритм явного ВМРК $r(\hat{r})$ по схеме (4.24) включает вычисление коэффициентов $\bar{k}_i$, определение двух аппроксимаций решения $\bar{y}_{n+1}$ и $\hat{\bar{y}}_{n+1}$, вычисление величины d_n.

Соответственно, время его реализации в базовых операциях линейной алгебры равно:

$$T_1^{22} = (s+1) \cdot T_1^{A \times Y} + \left(\frac{s^2+5s+10}{2} \right) \cdot T_1^{c \cdot Y} + \left(\frac{s^2+3s+6}{2} \right) \cdot T_1^{Y+Y}.$$

Для правомерности сравнения вложенных методов решения СЛОДУ одного и того порядка точности $O(h^r)$ необходимо учесть

соотношение между числом стадий и порядком явного метода. Для вложенного метода Фельберга 4(5) число стадий равно 6, а для метода Дормана-Принса 8(7) $s=13$. Поэтому далее число стадий метода будет вычисляться по соотношению: $s := s + c$, где величина $c = 1,2,...$ и зависит от порядка используемого метода [95].

Тогда время выполнения алгоритма по схеме (4.24) равно:

$$T_1^{22} = [(2s+2)m^2 + (s^2 + 3s + 7)m] \cdot t_{op}. \qquad (4.25)$$

Рассмотрим построение параллельного алгоритма стандартной вычислительной схемы явного вложенного метода (рис. 4.7).

Первоначально вычисляются стадийные коэффициенты: $\bar{k}_1, \bar{k}_2, ..., \bar{k}_{s'}$ с использованием алгоритма систолического блочного умножения матрицы на вектор, затем обе аппроксимации решения смежных порядков $r(r')$ и d_n . Для исключения дополнительных передач данных при вычислении обеих аппроксимаций и величины d_n поддерживается блочное разбиение этих величин на подвекторы:

$$\bar{y}_{n+1} = (Y_{n+1,1}, ..., Y_{n+1,q}),$$
$$\hat{\bar{y}}_{n+1} = (\hat{Y}_{n+1,1}, ..., \hat{Y}_{n+1,q}),$$
$$D_n = (D_{n,1}, ..., D_{n,q}).$$

Время арифметических операций и операций обмена для алгоритма ВЯМРК определяется с использованием данных таблиц (4.1)-(4.2):

$$T_{p,comp}^{22} = \left\lceil (2s+2)\frac{m^2}{p^2} + \left(s^2 + 4s + 7 + \lceil log_2 p \rceil\right)\frac{m}{p} \right\rceil \cdot t_{op}, \qquad (4.26)$$

$$T_{p,comm}^{22} = (s+1) \cdot \left(2^{\lceil log_2 p \rceil} + p - 2\right) \cdot \left(t_s + \frac{m}{p} t_w\right). \qquad (4.27)$$

Оценим асимптотическую сложность обеих вычислительных схем (рис. 4.6-4.7) на основе вложенных форм. Ограничимся учетом наиболее ресурсоемкой операции умножения матрицы на вектор, тогда $T_p^{22} / T_p^{21} \approx (s+c)/2,\ c = 1,2...,$, то есть стандартная схема обладает большей вычислительной сложностью, чем экспоненциальная.

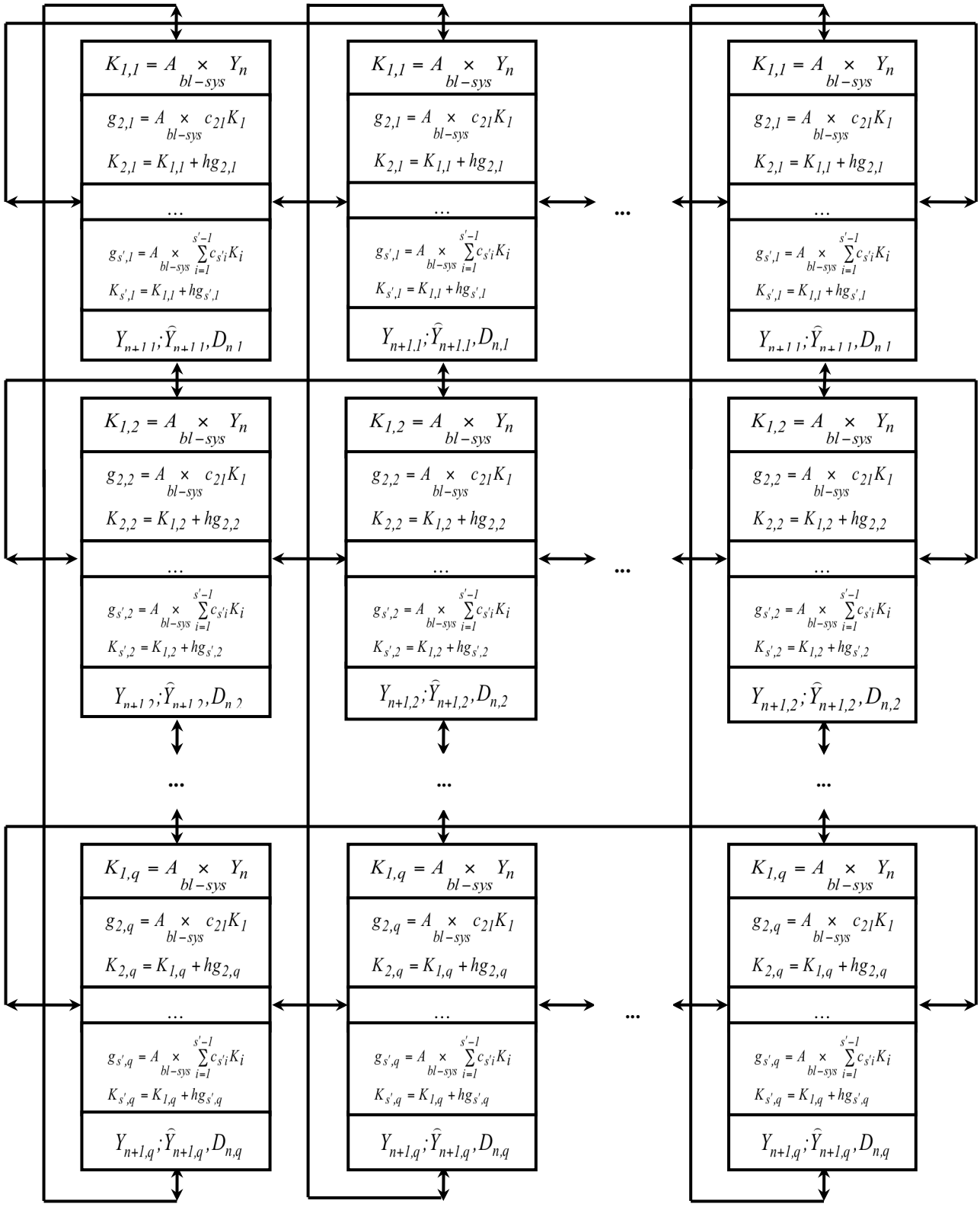


Рисунок 4.7 – Параллельный алгоритм решения СЛОДУ на базе вложенного ЯМРК (стандартный метод)

Сравнение реальных динамических характеристик этих вычислительных схем также позволяет сделать выводы о преимуществах предложенного экспоненциального подхода (рис. 4.8-4.9). Как результат, время реализации параллельного алгоритма для экспоненциальной схемы меньше, чем для стандартной и традиционно возрастает с ростом размерности задачи: $m \uparrow \Rightarrow \left(T_p^{21} \uparrow\right) < \left(T_p^{22} \uparrow\right)$ и уменьшается с ростом числа процессоров: $p \uparrow \Rightarrow \left(T_p^{21} \downarrow\right) < \left(T_p^{22} \downarrow\right)$.

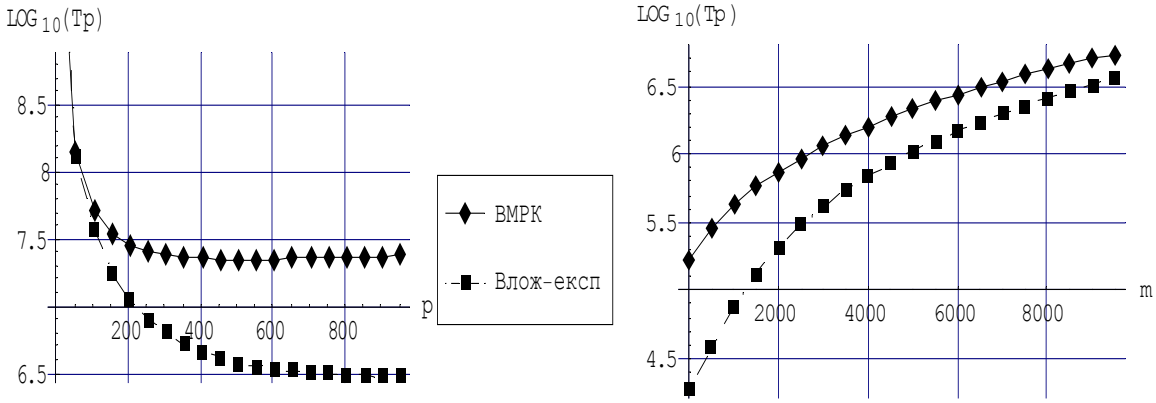


Рисунок 4.8 – Время реализации параллельных вложенных методов на основе стандартной, ВМРК и экспоненциальной схем

Рисунок 4.9 – Доля операций обмена для параллельных вложенных методов решения СЛОДУ на основе стандартной и экспоненциальной схем

Доля обменных операций для традиционной схемы превышает долю обменов для схемы с экспонентой (рис. 4.9). Таким образом, **вложенные методы на основе экспоненты требуют меньших накладных расходов по сравнению с традиционными схемами вычислений** для любых размерностей задачи и процессорного поля.

4.3. Реализация технологии локальной экстраполяции при параллельном численном решении линейной задачи Коши

Ускорение вычислений по **технологии локальной экстраполяции при решении линейной однородной задачи Коши с постоянными коэффициентами** также может быть получено на основе экспоненциального метода. По результатам, изложенным в главе 2, минимизировать вычислительные затраты можно реализацией h^2-экстраполяции для симметричных опорных методов малого порядка точности. Использование в качестве формы F частичной суммы из первых r_0 членов разложения матричной экспоненты в ряд Тейлора соответствует применению определенного метода Рунге-Кутты r_0 – го порядка точности:

$$\bar{y}_{n+1} = \left(E + hA + ... + \frac{h^{r_0} A^{r_0}}{r_0!} \right) \cdot \bar{y}_n = F_{r_0}(hA) \cdot \bar{y}_n. \qquad (4.28)$$

Если h – базовый шаг интегрирования, то $T_{11} = F_{r_0}(hA) \cdot \bar{y}_n$ – является аппроксимацией решения $\bar{y}(x_n + h)$ порядка $O(h^{r_0})$. При получении решения на шаге $n+1$ заданного порядка точности r необходимо провести вычисления для шагов $h/n_2, ..., h/n_k$ и получить соответствующие аппроксимации решения $T_{21}, ..., T_{k1}$:

$$T_{21} = F_{r_0}^{n_2}\left(hA/n_2\right) \cdot \bar{y}_n, ..., T_{k1} = F_{r_0}^{n_k}\left(hA/n_k\right) \cdot \bar{y}_n.$$

Затем, используя соотношение для полиномиальной экстраполяции (1.21), надо вычислить приближения T_{kk} порядка $O(h^r)$ и $T_{k-1,k}$ порядка $O(h^{r-1})$. Необходимое количество строк экстраполяционной таблицы равно $k = r/2$ при симметричном опорном методе и четной последовательности P_i.

Время вычисления по последовательной схеме для симметричного опорного метода порядка точности r_0 равно:

$$\begin{cases} T_{1,0}^{r_0,exp} = (r_0-1)\cdot T_1^{A\times A} + (r_0-1)\cdot T_1^{c\cdot A} = (r_0-1)\left(2m^3+m^2\right)\cdot t_{op}, \\ T_1^{r_0,exp} = n_k\cdot T_1^{A\times Y} + r_0\cdot T_1^{c\cdot A} + r_0\cdot T_1^{A+A} = 2m^2\left(N(k)+kr_0\right)\cdot t_{op}. \end{cases} \quad (4.29)$$

При определении времени последовательной реализации технологии локальной экстраполяции с опорным методом на базе матричной экспоненты воспользуемся (2.25), данными таблицы (4.1) и соотношениями:

$$T_1^{31} = T_{T_{11}} + T_{T_{21}} + \ldots + T_{T_{k1}} + T_1^{ext-tab},$$

$$T_1^{31} = \sum_{i=1}^{k} n_i\cdot T_1^{A\times Y} + kr_0\cdot T_1^{c\cdot A} + kr_0\cdot T_1^{A+A} + T_1^{ext-tab}.$$

Для числовой последовательности P_2 и **симметричного опорного метода** второго порядка имеем: $N(k)=k^2-k+1$. Тогда время вычислений по схеме опорного метода, $T_1^{r_0,exp}$ и общее время вычислений по технологии локальной экстраполяции с учетом определения $r/2$ строк экстраполяционной таблицы T_1^{31} равны:

$$T_1^{r_0,exp} = \frac{m^2}{2}\left(r^2+6r+4\right)\cdot t_{op}, \quad (4.30)$$

$$T_1^{31} = \left[\frac{m^2}{2}\left(r^2+6r+4\right)+\frac{5}{8}m\left(r^2-2r\right)\right]\cdot t_{op}. \quad (4.31)$$

При разработке параллельного алгоритма на базе экспоненциального метода с экстраполяцией по Ричардсону (рис. 4.10) необходимо выделить две подзадачи и, соответственно, макрооперации первого уровня. Первая подзадача: вычисление k аппроксимаций решения $T_{11},T_{21},\ldots,T_{k1}$ по опорному методу с шагами $h_1,h_2,\ldots,h_k$ и вторая: построение экстраполяционной таблицы по схеме Эйткена-Невилла, вычисление результирующих аппроксимаций решения T_{k-1k},T_{kk} . Параллельное выполнение первой макрооперации по аналогии с другими рассмотренными алгоритмами будет содержать подготовительный этап.

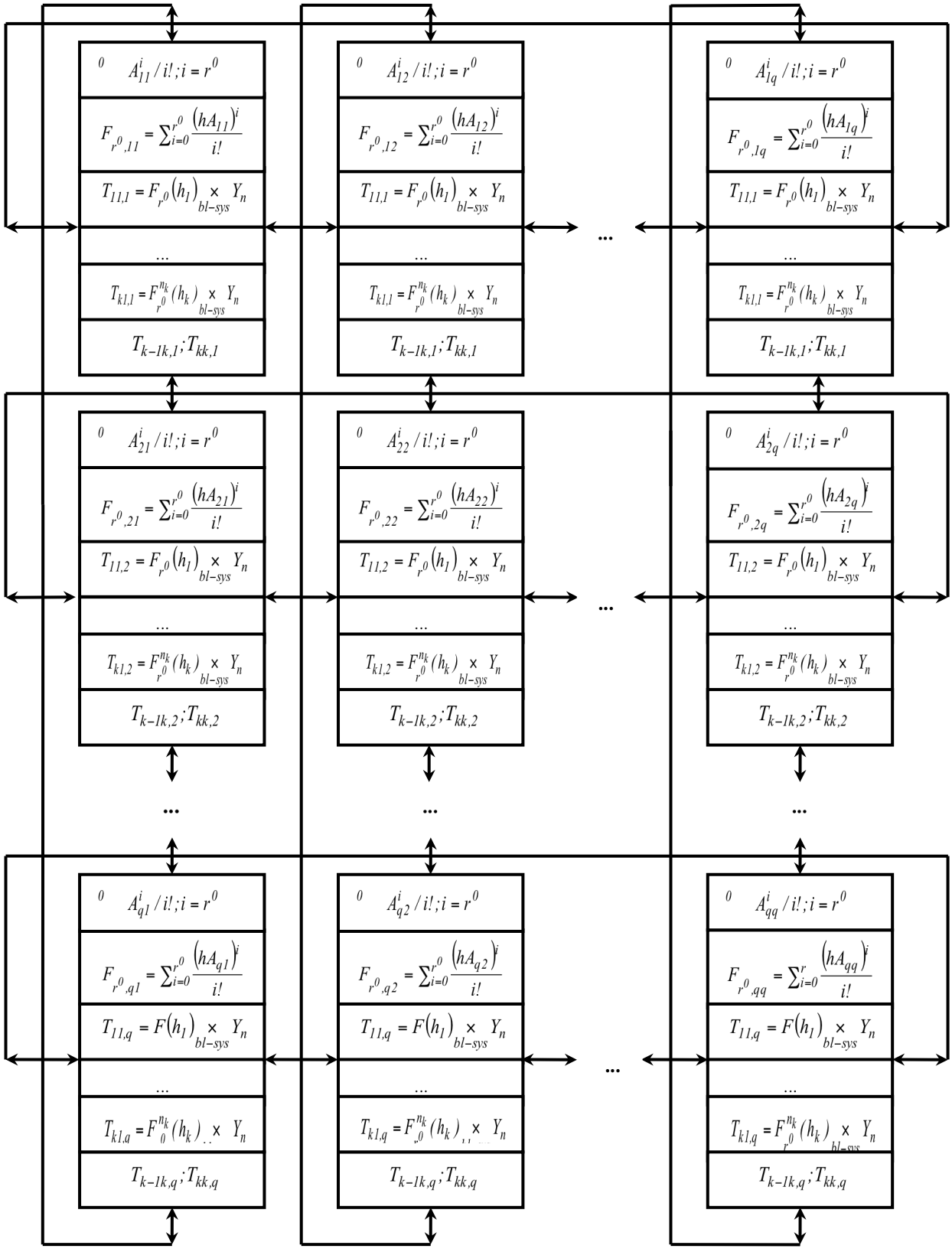


Рисунок 4.10 – Параллельный алгоритм решения СЛОДУ
с локальной экстраполяцией на базе экспоненциального метода

Время вычислений и время обменов для него, соответственно, равны:

$$\begin{cases} T_{p,0,comp}^{r_0,exp} = (r_0 - 1)\cdot\left(2m^3 / p^2\right)\cdot t_{op}, \\ T_{p,0,comm}^{r_0,exp} = 2(r_0 - 1)(p-1)\cdot[t_s + (m^2 / p^2)\cdot t_w]. \end{cases} \quad (4.32)$$

Далее вычисляются матричные формы $F_{n_i}^{r_0}, i = \overline{1,k}$ и аппроксимации решения, соответствующие первому столбцу экстраполяционной таблицы, затем на основе полиномиальной экстраполяции вычисляются остальные элементы таблицы. Первая макрооперация реализуется через известные операции линейной алгебры плюс параллельное вычисление экстраполяционной таблицы:

$$T_p^{31} = \sum_{i=1}^{k} n_i \cdot T_p^{A\times Y} + kr_0 \cdot T_p^{c\cdot A} + kr_0 \cdot T_p^{A+A} + T_p^{ext-tab}. \quad (4.33)$$

Разбиение данных по процессорам соответствует топологии 2D-тор и блочному систолическому алгоритму выполнения матричного умножения, следовательно, время вычислений в t_{op} равно:

$$T_{p,comp}^{31} = \left\lceil \frac{m^2}{p^2}\left(\frac{r^2+6r+4}{2}\right) + \frac{m}{p}\left(\frac{r^2-2r+4}{4}\lceil \log_2 p\rceil + \frac{5}{8}\left(r^2-2r\right)\right)\right\rceil \cdot t_{op}. \quad (4.34)$$

Коммуникационная составляющая описанного алгоритма определяется только обменами при вычислении k аппроксимаций решения на основе опорного метода. Вычисление элементов экстраполяционной таблицы при выбранном способе распараллеливания не требует межпроцессорных обменов и перегруппировки данных для следующего шага интегрирования.

Общее время на реализацию операций передачи данных равно:

$$T_{p,comm}^{31} = [(r^2-2r+4)/4]\cdot\left(2^{\lceil \log_2 p\rceil} + p - 2\right)\cdot\left[t_s + (m/p)\frac{m}{p}t_w\right]. \quad (4.35)$$

Проведем сравнение разработанного экспоненциального параллельного алгоритма решения линейной задачи Коши с оценкой шаговой погрешности по технологии локальной экстраполяции (рис. 4.10) со стандартной схемой вычислений по схеме Экстраполяции Ричардсона на базе явного опорного метода типа Рунге-Кутты.

Для этого в качестве опорного метода принимается явный одношаговый s_0- стадийный метод порядка $O\left(h^{r0}\right)$:

$$\begin{cases} \bar{y}_{n+1} = \bar{y}_n + h \cdot \sum_{i=1}^{s_0} b_i \cdot \bar{k}_i, \\ \bar{k}_l = A \cdot (\bar{y}_n + h \cdot \sum_{i=1}^{l-1} c_{li} \cdot \bar{k}_i), l = \overline{1, s_0}. \end{cases} \quad (4.36)$$

Время параллельной реализации экстраполяции Ричардсона при опорном методе (4.36) в базовых операциях линейной алгебры равно:

$$T_p^{32} = \sum_{i=1}^{k} n_i \cdot \left(s_0 \cdot T_p^{A \times Y} + \frac{s_0^2 + s_0}{2} \cdot T_p^{c \cdot Y} + \frac{s_0(s_0 - 1)}{2} \cdot T_p^{Y+Y} \right) + T_p^{ext-tab}. \quad (4.37)$$

Для сравнения вычислительной сложности двух параллельных алгоритмов решения СЛОДУ на базе локальной экстраполяции ограничимся учетом наиболее ресурсоемкой операции умножения матрицы на вектор. Стандартная схема решения обладает вычислительной сложностью в s_0 раз большей, чем экспоненциальная, в силу справедливости соотношения:

$$T_p^{32} / T_p^{31} \approx \frac{s_0 N(k)}{N(k)} = s_0. \quad (4.38)$$

В результате проведенных экспериментов, получена аналогичная зависимость (рис. 4.11) и при учете полного времени выполнения стандартной и экспоненциальной схем интегрирования СЛОДУ.

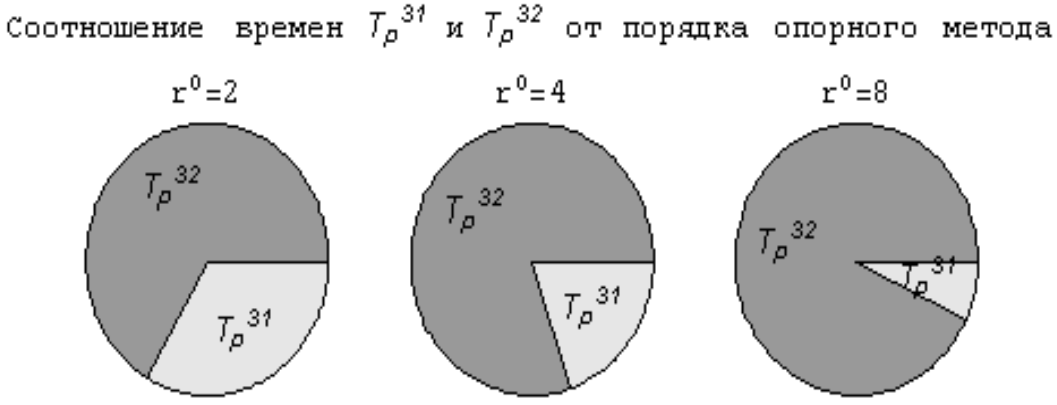


Рисунок 4.11 – Время реализации параллельных алгоритмов с локальной экстраполяцией для экспоненциальной, T_p^{31} и стандартной, T_p^{32} схем

Таким образом, как и при других способах оценки апостериорной шаговой погрешности, применение экспоненциального метода интегрирования в качестве опорного для технологии локальной экстраполяции Ричардсона, сокращает вычислительные и коммуникационные затраты приблизительно в s_0 раз.

Экспоненциальный метод также может быть применен для ускорения решения **неоднородных линейных СОДУ с постоянными коэффициентами**:

$$\begin{cases} \bar{y}'(x) = A \cdot \bar{y}(x) + b, \\ \bar{y}(x_0) = \bar{y}_0, \\ A, b = const. \end{cases} \tag{4.39}$$

Численное решение задачи Коши для неоднородной СЛОДУ с постоянными коэффициентами (4.39) имеет вид:

$$\bar{y}_{n+1} = \bar{y}_n + R_r(hA) \cdot (A\bar{y}_n + b), \tag{4.40}$$

где $R_r(hA)$ – матричная функция, которая может быть аппроксимирована отрезком ряда Тейлора:

$$R_r(hA) = \sum_{i=1}^{r} (A^{i-1} h^i / i!).$$

Как и в случае однородных СЛОДУ, для решения таких систем может быть применен экспоненциальный метод со встроенными способами оценки локальной апостериорной погрешности с целью управления шагом интегрирования, что особенно важно при решении жестких задач Коши. Более того, этот прием может быть распространен на случай общих линейных систем, в частности с переменными коэффициентами, если на каждом малом отрезке интегрирования исходную задачу аппроксимировать задачей (4.40).

Разработка параллельных алгоритмов со встроенными способами оценки локальной погрешности, теоретический анализ сложности позволяют сделать вывод, что применение экспоненциального метода для решения неоднородной линейной задачи Коши, как и для однородной, позволяет уменьшить вычислительные и коммуникационные затраты на параллелизм по сравнению со стандартными схемами решения СЛОДУ.

Предлагаемые алгоритмы особенно эффективны при решении задач с большой константой Липшица, в частности для жестких систем уравнений и требуют меньшего объема вычислений, чем стандартные методы решения линейной задачи Коши.

4.4. Повышение эффективности решения линейной задачи Коши на основе метода быстрого умножения матриц

Матричное умножение (МУ) – доминирующая вычислительная часть экспоненциальных методов решения СЛОДУ со встроенными способами оценки локальной апостериорной погрешности. На основании этого факта – ускорение параллельного выполнения этой базовой операции линейной алгебры, определяет уменьшение времени реализации всего метода решения линейной задачи Коши для систем обыкновенных дифференциальных уравнений в целом.

При огромном многообразии методов вычисления матричного произведения [14, 120-122] **для плотнозаполненных матриц** имеется два принципиально различных класса алгоритмов:

1) **традиционные методы матричного умножения**;

2) **рекурсивные методы на основе быстрого умножения Штрассена-Винограда** [123-125].

Практически все алгоритмы или их варианты имеют приблизительно линейное ускорение для матриц больших размерностей, и не существует алгоритма, который был бы существенно лучше других.

Вычислительная сложность матричного умножения $C = A \times B$ для исходных квадратных матриц размерности $m \times m$ составляет:

1) для традиционных последовательных алгоритмов – $O(m^3)$;

2) для быстрого умножения по Штрассену – $O(m^{2.81})$;

3) по методу Винограда – $O(m^{2.376})$.

В оригинале алгоритм Штрассена – это рекурсивный алгоритм умножения блочных матриц половинного размера, где каждый блок квадратный, т.е. размерности матриц должны быть четными числами. Это ограничение легко обходится за счет окаймления нулевыми элементами.

В данном подразделе исследуются вопросы повышения эффективности разработанных параллельных алгоритмов решения СЛОДУ со встроенными способами оценки локальной апостериорной погрешности за счет рационального выбора алгоритма матричного умножения.

Метод Штрассена-Винограда [123] состоит из 7 блочных умножений и 15 блочных сложений матриц (рис. 4.12).

$$
\begin{array}{lll}
S_1 = A_{21} + A_{22}, & M_1 = S_2 S_6, & T_1 = M_1 + M_3, \\
S_2 = S_1 - A_{11}, & M_2 = A_{11} B_{11}, & T_2 = T_1 + M_4, \\
S_3 = A_{21} - A_{12}, & M_3 = A_{12} B_{21}, & T_3 = M_5 + M_6, \\
S_4 = A_{12} - S_2, & M_4 = S_3 S_7, & C_{11} = M_2 + M_3, \\
S_5 = B_{12} - B_{11}, & M_5 = S_1 S_5, & C_{12} = T_1 + T_3, \\
S_6 = B_{22} - S_5, & M_6 = S_4 B_{22}, & C_{21} = T_2 - M_7, \\
S_7 = B_{22} - B_{12}, & M_7 = A_{22} S_8, & C_{22} = T_2 + M_5, \\
S_8 = S_6 - B_{12}, & &
\end{array}
$$

Рисунок 4.12 – Метод быстрого рекурсивного умножения матриц Штрассена-Винограда

Идея Штрассена может быть применена рекурсивно для нахождения произведений блоков матриц $M_i, i = \overline{1,7}$. Если исходные матрицы A и B порядка m, то алгоритм быстрого умножения можно использовать многократно, получая на самом нижнем уровне рекурсии блоки $k \times k = 1 \times 1$.

Однако нет необходимости опускаться вниз до уровня блоков единичного порядка. При достаточно малых размерах блока ($k \le k_{min}$) может оказаться выгодным вычислять блоки, используя стандартный алгоритм матричного умножения. На первом шаге алгоритм предусматривает 7 обращений к самому себе с матрицами порядка $m/2$ и 15 операций типа сложение матриц того же порядка. Далее идет развертка рекурсии до достижения минимального размера блока. Вычислительная сложность предложенной схемы алгоритма быстрого умножения определяется функцией размерности исходных матриц и минимального порядка умножаемых блоков: m, k_{min}.

Пусть при выполнении матричного умножения алгоритм Штрассена-Винограда рекурсивно вызывался d раз, тогда порядок умножаемых блоков матриц равен $k_{min} = m / 2^d$.

Время реализации последовательного алгоритма при этом составит:

$$T_1^{Str}(m) = \left[2\left(\frac{7}{8}\right)^d m^3 + \frac{15}{4} m^2 \left(1 + \frac{7}{4} + \frac{7^2}{4^2} + \ldots + \frac{7^{d-1}}{4^{d-1}} \right) \right] \cdot t_{op}. \qquad (4.41)$$

Известный параллельный алгоритм классического метода Штрассена был применен на *Intel Paragon* и показал хорошие результаты по сравнению с традиционными алгоритмами МУ [126-127]. Существенным недостатком этого алгоритма является отсутствие масштабируемости, так как требуемое число процессоров для него кратно семерке (по количеству умножений блоков матриц $M_i, i = \overline{1,7}$). Это ограничение является жестким и не естественным для большинства параллельных архитектур.

Для преодоления указанного недостатка воспользуемся полиалгоритмическим подходом. **Под полиалгоритмом подразумевается комбинация двух или более алгоритмов в одну вычислительную схему, реализующую поставленную задачу с целью сокращения вычислительных затрат** [14].

Алгоритм быстрого умножения рекурсивен, поэтому имеется возможность построить полиалгоритм из некоторого традиционного алгоритма умножения матриц на верхнем уровне рекурсии и метода Штрассена на нижнем уровне, и наоборот. Алгоритм Штрассена является блочным, поэтому естественно комбинировать его с алгоритмом МУ, использующим соответствующее разбиение данных. Построим полиалгоритм из блочного систолического алгоритма матричного умножения между процессорами и серии применений рекурсивного метода Штрассена на каждом процессоре (рис. 4.13).

Пусть имеется замкнутая двумерная сетка процессоров размерности $p \times p$, исходные матрицы A и B распределены на блоки $k = m / p$, количество блоков равно $q^2 = p^2$. Блоки исходных матриц и результата с координатами $< i, j >$ хранятся в соответствующем процессоре с теми же координатами.

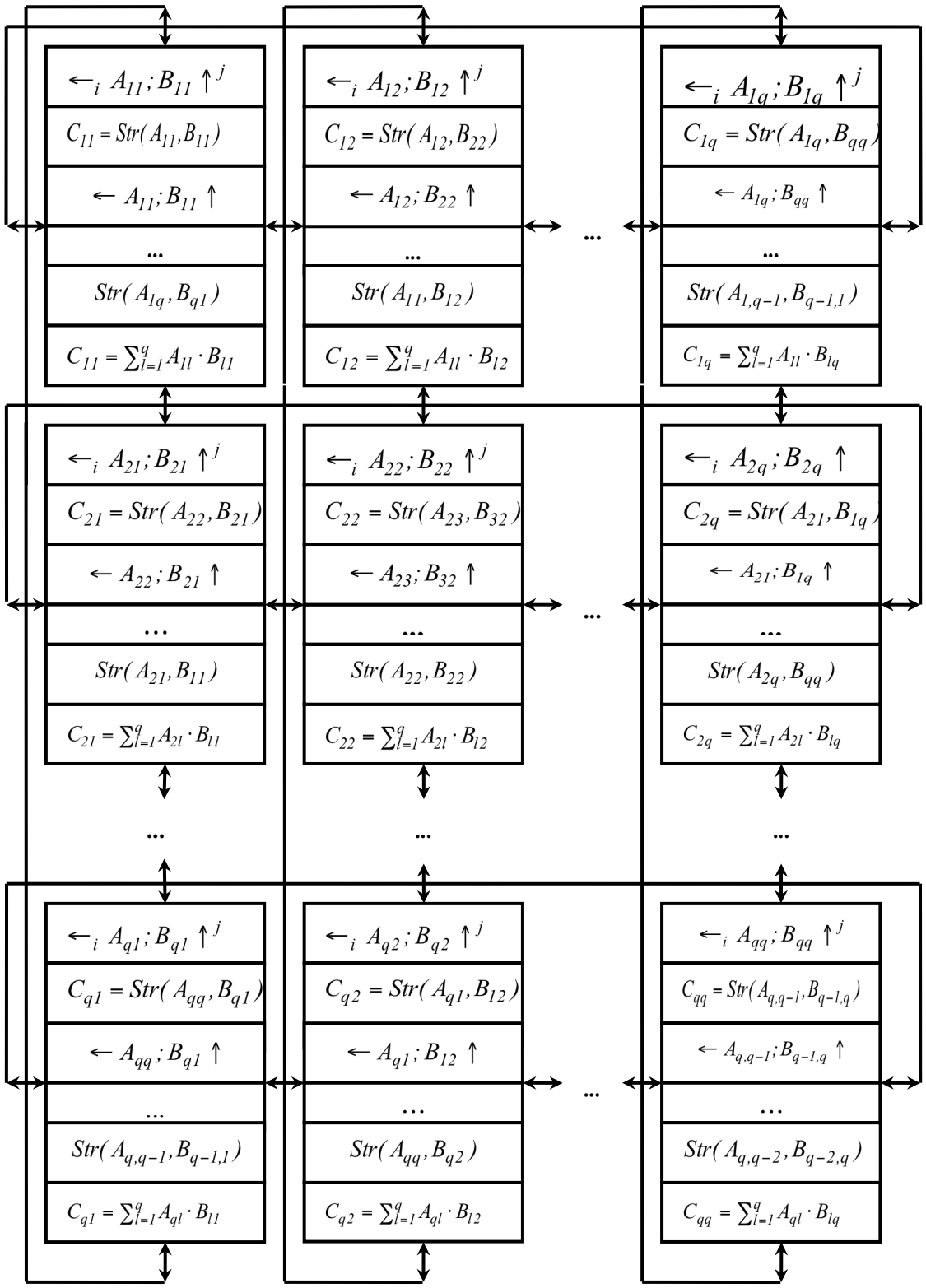


Рисунок 4.13 – Полиалгоритм умножения матриц:
блочное систолическое умножение + алгоритм Штрассена-Винограда

Как и ранее, предполагаем, что $m \ mod \ p = 0$, в противном случае используется **окаймление нулевыми элементами**. Предварительно, по вычислительной схеме блочного систолического умножения, выполняется косой сдвиг влево по строкам для блоков матрицы A и косой сдвиг вверх по столбцам для блоков матрицы B.

При этом время выполнения вычислений по схеме рис. 4.13 равно:

$$T_{p,comp}^{BSys-Str} = q \cdot \left(T_{A_{ij} \underset{Str}{\times} B_{ij}} + T_{A_{ij}+B_{ij}} \right), \tag{4.42}$$

где $T_{A_{ij} \underset{Str}{\times} B_{ij}}$ – время умножения блоков матриц порядка $k = m / p$ по рекурсивному алгоритму;

$T_{A_{ij}+B_{ij}} = k^2 \cdot t_{ad} = \left(m^2 / p^2 \right) \cdot t_{ad}$ – время сложения блоков матриц той же размерности.

Время выполнения умножения блоков выполняется по алгоритму Штрассена-Винограда и удовлетворяет рекуррентному соотношению:

$$T_{A_{ij} \underset{Str}{\times} B_{ij}} = T_p^{Str}(k) = 7T_p^{Str}\left(\frac{k}{2}\right) + 15\left(\frac{k}{2}\right)^2 \cdot t_{ad}. \tag{4.43}$$

В свою очередь, время реализации алгоритма для матриц порядка $k/2,...,k/2^{d-1}$ соответственно равно:

$$T_p^{Str}\left(\frac{k}{2}\right) = 7T_p^{Str}\left(\frac{k}{4}\right) + 15\left(\frac{k}{4}\right)^2 t_{ad},$$

$$\ldots,$$

$$T_p^{Str}\left(\frac{k}{2^{d-1}}\right) = 7T_p^{Str}\left(\frac{k}{2^d}\right) + 15\left(\frac{k}{2^d}\right)^2 t_{ad}.$$

Выполнив подстановки и элементарные преобразования, получим:

$$T_p^{Str}(k) = 7^d T_p^{Str}\left(\frac{k}{2^d}\right) + \frac{15}{4}k^2\left(1 + \frac{7}{4} + \frac{7^2}{4^2} + \ldots + \frac{7^{d-1}}{4^{d-1}}\right)t_{ad}. \tag{4.44}$$

До сих пор была определена вычислительная сложность развертки рекурсии. Рассмотрим внутренний уровень рекурсии, поскольку именно там выполняются операции умножения блоков матриц минимального порядка по стандартному методу матричного умножения:

$$T_p^{Str}\left(\frac{k}{2^d}\right) = k_{min}^3 \cdot \left(t_{mul} + t_{ad}\right) = 2\left(\frac{k}{2^d}\right)^3 \cdot t_{op}. \qquad (4.45)$$

Таким образом, время реализации быстрого рекурсивного умножения блоков матриц для мультикомпьютера равно:

$$T_p^{Str}(k) = \left[2\left(\frac{7}{8}\right)^d k^3 + \frac{15}{4}k^2\left(\frac{1-\frac{7^d}{4^d}}{1-\frac{7}{4}}\right)\right] \cdot t_{op}. \qquad (4.46)$$

Тогда общее время выполнения арифметических операций для комбинации блочного систолического и рекурсивного алгоритмов матричного умножения равно:

$$T_{p,comp}^{BSys-Str} = \left[2\left(\frac{7}{8}\right)^d \frac{m^3}{p^2} + \left(5\left(\frac{7}{4}\right)^d - 4\right)\frac{m^2}{p}\right] \cdot t_{op}. \qquad (4.47)$$

Время обменных операций для описанной схемы (рис. 4.13) определяется, как и для блочного систолического алгоритма:

$$T_{p,comm}^{BSys-Str} = 4(p-1) \cdot \left(t_s + \frac{m^2}{p^2} \cdot t_w\right). \qquad (4.48)$$

Очевидно, что динамические характеристики параллельных вычислительных схем МУ зависят от соотношения между числом процессоров и размерностью матриц. Для рекурсивного алгоритма умножения матриц существенным параметром является также величина глубины рекур**сии**, d (рис. 4.14-4.17).

Определение **оптимальной величины глубины рекурсии**, а, следовательно, и **размерности минимального обрабатываемого блока умножаемых матриц**, производилось стандартным математическим аппаратом в пакете *Mathematica.* Для этого необходимо найти значение d, доставляющее минимум функции T_1^{Str}, причем диапазон изменения глубины рекурсии ограничен следующим неравенством:

$$k_{min} = \left(m / 2^d\right) \geq 1 \Rightarrow\ m \geq 2^d \Rightarrow d \leq log_2 m.$$

Очевидно, что при больших размерностях СЛОДУ каждая из функций $V(d) = T_1^{Str} / t_{op}$ рисунка 4.15 имеет явно выраженный

локальный минимум, отношение же размерности системы к глубине рекурсии при этом остается постоянным.

```
(* Определение оптимальной глубины рекурсии *)

Round[Simplify[Log[2, 2*m*(Log[8/7])] - Log[2, (5*Log[7/4])]]]
```

$$d_{opt} = \text{Round}\left[\frac{\text{Log}\left[2\, m \,\text{Log}\left[\frac{8}{7}\right]\right] - \text{Log}\left[5\,\text{Log}\left[\frac{7}{4}\right]\right]}{\text{Log}[2]}\right]$$

$$m_{opt} = N\left[m \Big/ 2^{d_{opt}}\right] = 8$$

Рисунок 4.14 – Определение оптимального значения глубины рекурсии рекурсивного алгоритма МУ

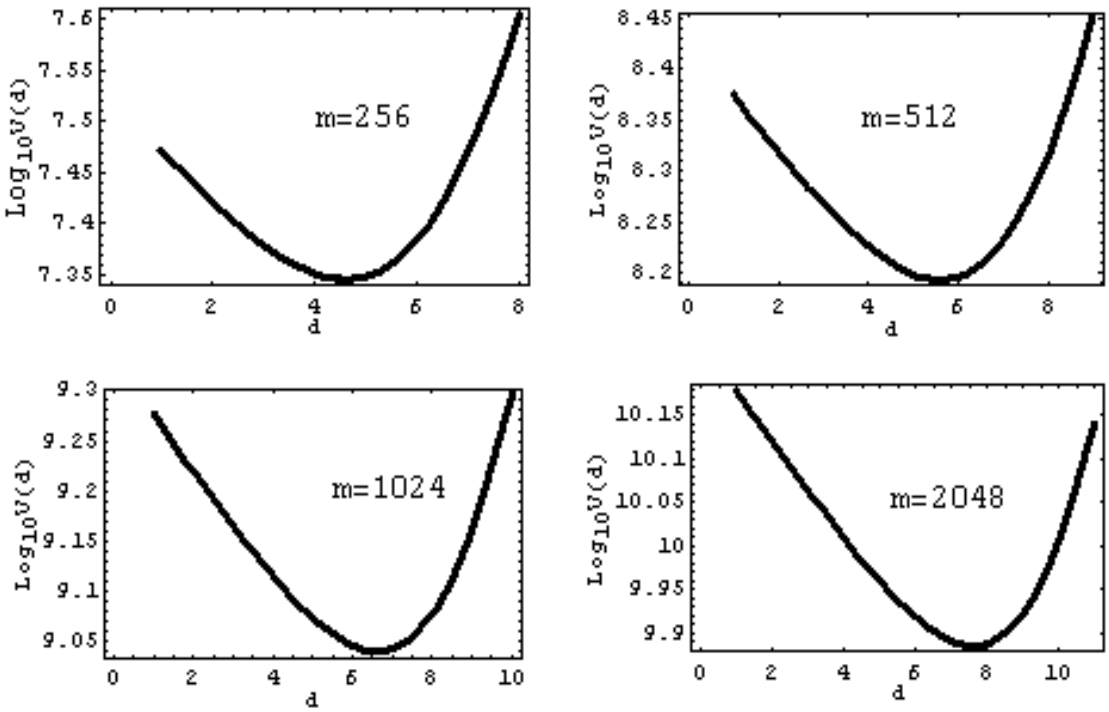


Рисунок 4.15 – Вычислительная сложность рекурсивно-систолического алгоритма от глубины рекурсии в логарифмическом масштабе

Например, при $m = 1024$, $d_{min} = 7$ и $m / 2^{d_{min}} = 1024 / 128 = 8$, а при $m = 256$ имеем $d_{min} = 5 \quad \Rightarrow m / 2^{d_{min}} = 8$.

В тоже время при $m = 16$ минимальная глубина рекурсии равна 1, то есть для достижения минимума вычислительных затрат необходимо выполнить всего один шаг развертки рекурсии, для $m = 8$ развертки рекурсии нет, так как $d_{min} = 0$. Аналитические результаты подтверждаются результатами экспериментов (рис. 4.16).

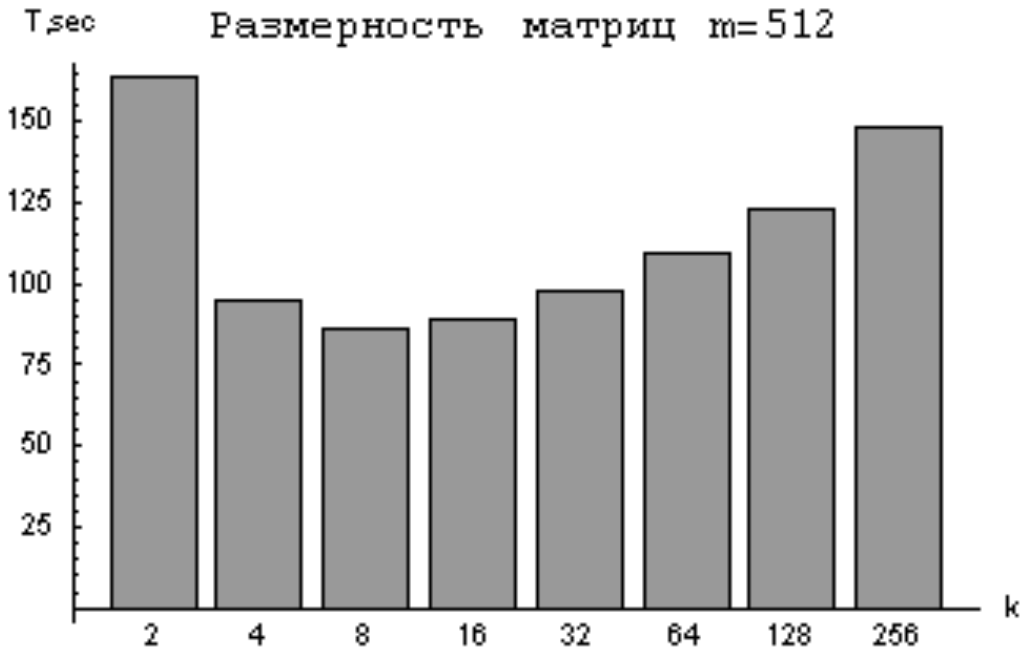


Рисунок 4.16 – Экспериментальное определение величины минимального блока для рекурсивно-систолического алгоритма умножения матриц

Для параллельного варианта предложенного алгоритма блочного рекурсивно-систолического умножения кроме глубины рекурсии необходимо учесть соотношение между числом процессоров, p и размерностью матриц, m (рис. 4.17). Ограничения на диапазон изменения глубины рекурсии определяются следующим неравенством:

$$k_{min} = \frac{m}{2^d p} \geq 1 \Rightarrow \frac{m}{p} \geq 2^d \Rightarrow d \leq log_2 \frac{m}{p}.$$

При оценке глубины рекурсии временные характеристики приведены ко времени реализации одной операции с плавающей точкой, для различных мультикомпьютеров величина оптимальной

глубины рекурсии должна быть поправлена с учетом этой машинно-зависимой константы.

```
(* Определение значения оптимальной глубины рекурсии *)
pd_min = Round [Simplify[Log[2, 2*m/p*(Log[8/7])] - Log[2, (5*Log[7/4])]]]
```

$$pd_{min} = \text{Round}\left[\frac{\text{Log}\left[\frac{\frac{2\,m\,\text{Log}\left[\frac{8}{7}\right]}{p}}{\text{Log}\left[5\,\text{Log}\left[\frac{7}{4}\right]\right]}\right]}{\text{Log}[2]}\right]$$

$$k_{min} = (m/p)_{min} = 8 * 2^{pd_{min}}$$

Рисунок 4.17 – Определение значения оптимальной глубины рекурсии для параллельного блочного рекурсивно-систолического алгоритма

Анализ аналитических выражений, характеризующих выполнение параллельных алгоритмов, а также проведенный численный эксперимент, позволяют сделать следующие выводы (рис. 4.18-4.19):

1) параллельный блочный метод рекурсивно-систолического умножения матриц обладает меньшей асимптотической:

$$T_p^{bl-sys} / T_p^{Bsys-Str} \approx (8/7)^d$$

и реальной, подтверждаемой экспериментальным путем, сложностью по сравнению с традиционными алгоритмами умножения матриц [12-14] (рис. 4.18);

2) динамические характеристики предложенного алгоритма на основе быстрого умножения превосходят соответствующие характеристики стандартных аналогов, особенно для матриц больших размерностей, при этом коэффициенты ускорения и эффективности возрастают с ростом размерности задачи: $m \uparrow \Rightarrow S \uparrow \Rightarrow E \uparrow$ и уменьшаются с ростом числа процессоров $p \uparrow \Rightarrow S \downarrow \Rightarrow E \downarrow$ (рис. 4.19).

Кроме того, рекурсивный характер разработанного алгоритма позволяет свести многомерную исходную задачу к решению подзадач меньшей размерности, и, за счет использования быстрой памяти компьютера, достичь ускорения операции МУ.

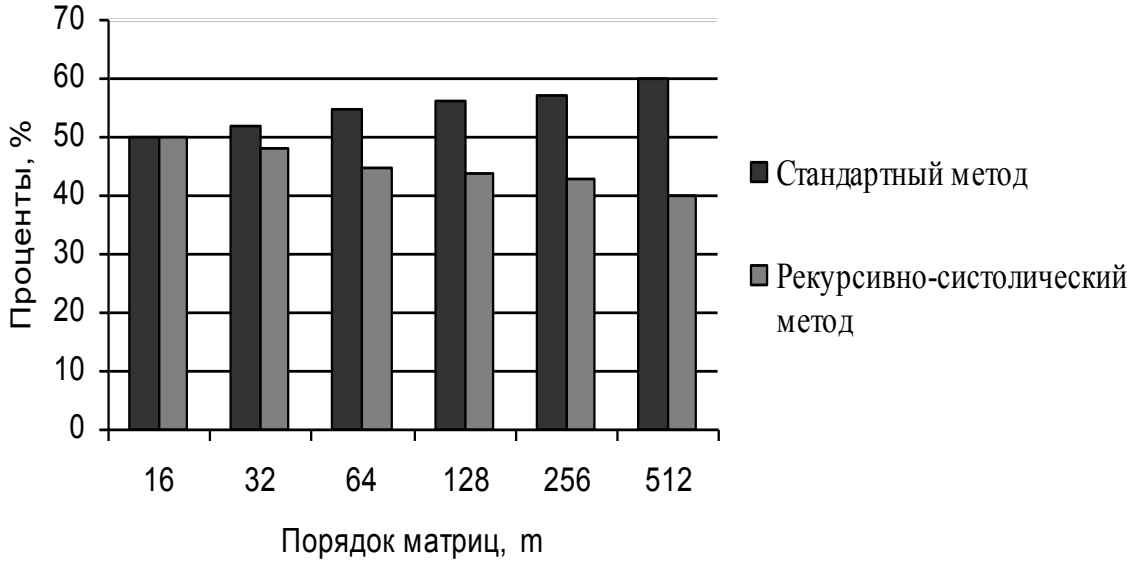


Рисунок 4.18 – Соотношение экспериментальных времен реализации параллельных методов матричного умножения

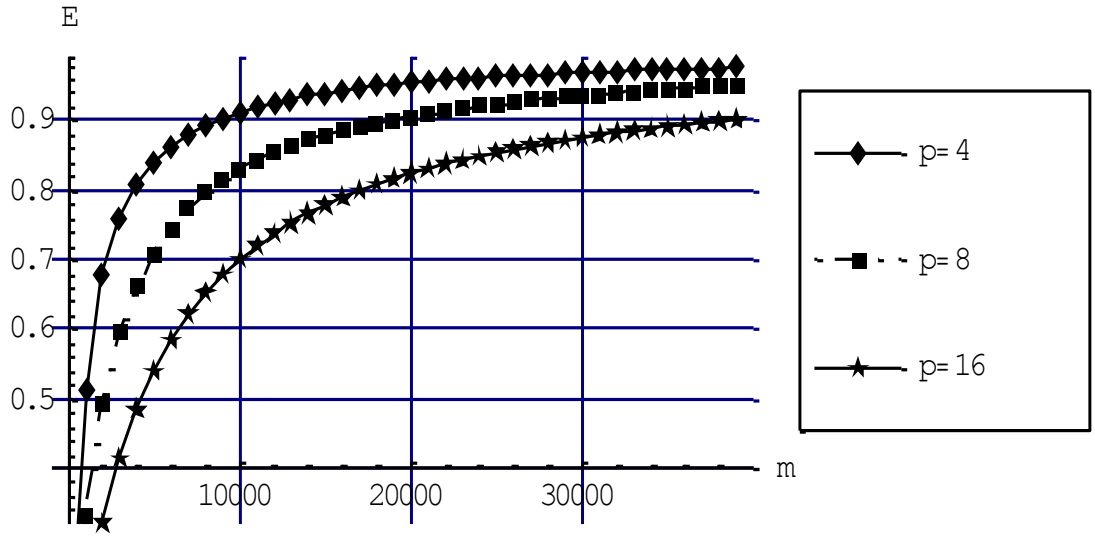


Рисунок 4.19 – Коэффициенты эффективности параллельного блочного рекурсивно-систолического метода для мультикомпьютера с шириной процессорной решетки, равной p

К недостаткам этого подхода следует отнести несколько худшую численную устойчивость, хотя и достаточную для большинства практических задач [126-127].

Вторая модификация систолического алгоритма с блочным разбиением данных выполняет последовательную реализацию систолического умножения блоков матриц. Использование блочного разбиения матриц позволяет на решетке размерностью $p \times p$ перемножать матрицы порядка ip, где $i = 1,2,...$

Перемножаемые квадратные матрицы размерности $m \times m$ разбиваются на блоки, кратные ширине матрицы процессорного поля: $k = ip$. Тогда, $m = q \cdot k$, где k – это размер ширины блока, а $q^2 = \left[m / ip\right]^2$ – количество блоков. Затем для каждого блока находится результат при помощи систолического алгоритма. При получении одного блока матрицы результата C_{ij} необходимо q раз применить обычный систолический алгоритм.

Время вычислений для систолического алгоритма №2 с блочным разбиением определяется, как:

$$T_{p,comp}^{A\times B,bl-sys,2} = q^3 i^2 (2ip + 1) = \frac{m^3}{ip^3}(2ip + 1). \quad (4.49)$$

Коммуникационная составляющая этого алгоритма равна:

$$T_{p,comm}^{A\times B,bl-sys,2} = \frac{m^3}{i^3 p^3} \cdot \left(4(p-1) \cdot \left(t_s + i^2 t_w\right)\right). \quad (4.50)$$

Такая вычислительная схема умножения матриц преодолевает недостаток систолического алгоритма, заключающийся в жестком ограничении на размерность перемножаемых матриц, накладываемом размерностью решетки процессорных элементов.

4.5. Сравнение эффективности параллельных методов оценки шаговой погрешности при решении линейных задач Коши

Исследование эффективности альтернативных способов оценки апостериорной шаговой погрешности при решении линейной однородной задачи Коши с постоянными коэффициентами проводилось на следующем множестве методов одного и того же порядка r:

1 – экспоненциальный метод и правило Рунге;

2 – явный метод Рунге-Кутты и правило Рунге;

3 – экспоненциальный метод с локальной экстраполяцией;

4 – ЯМРК и локальная экстраполяция Ричардсона;

5 – вложенные методы на основе экспоненциального;

6 – вложенные методы Рунге-Кутты.

Заметим, что динамические характеристики параллелизма существенно зависят от параметров как исходной задачи, так и параллельной системы, на которой они реализованы. Влияние же численного метода определяется наличием внутреннего параллелизма и порядком разностной схемы, положенной в его основу. Поэтому сравнение эффективности предлагаемых параллельных методов есть сравнение не уникальных вычислительных схем, а двух классов решения линейной задачи Коши [55-56].

Проведем оценку асимптотической сложности полученных параллельных алгоритмов на основе анализа времени их реализации на мультикомпьютере из p^2 процессоров с учетом самой трудоемкой операции (табл. 4.3):

а) для методов, основанных на применении матричной экспоненты, выполняется следующее соотношение: $T_p^{31} < T_p^{11} < T_p^{21}$; аналогичное соотношение имеет место и для стандартной схемы: $T_p^{32} < T_p^{12} < T_p^{22}$;

б) наибольшей временной сложностью обладают методы локальной экстраполяции и, прежде всего, на основе стандартной явной схемы типа Рунге-Кутты:

1) $O[sr^2m^2/p^2]$ – для вычислений;

2) $O\left[sr^2p\left(t_s+\frac{m}{p}\cdot t_w\right)\right]$ – для обменных операций;

в) лучшими показателями обладают алгоритмы, основанные на сочетании экспоненты и вложенных формул; так, например, вычислительная сложность:

1) вложенного экспоненциального метода составляет – $O[2\cdot m^2/p^2]$;

2) экспонента с правилом Рунге – $O[3\cdot m^2/p^2]$;

3) ВМРК – $O[(s+1)\cdot m^2/p^2]$;

г) все параллельные алгоритмы, основанные на применении частного, специального метода, а именно экспоненциального метода решения, имеют меньшую временную сложность по сравнению со своими стандартными аналогами.

Таблица 4.3

Оценки асимптотической временной сложности способов определения локальной апостериорной погрешности, приведенные ко времени выполнения наиболее ресурсоемкой операции

Способы оценки локальной апостериорной погрешности	Асимптотическая временная сложность	
	Экспоненциальная схема	Стандартная схема
1.Правило Рунге	$3 \cdot T_p^{A \times Y}$	$(3s-1) \cdot T_p^{A \times Y}$
2.Локальная экстраполяция	$\left(\frac{r^2 - 2 \cdot r + 4}{4} \right) \cdot T_p^{A \times Y}$	$\left(\frac{r^2 - 2 \cdot r + 4}{4} \right) \cdot s \cdot T_p^{A \times Y}$
3.Вложенные формы	$2 \cdot T_p^{A \times Y}$	$(s+1) \cdot T_p^{A \times Y}$

Традиционно сравнение качества полученных параллельных алгоритмов производится с использованием коэффициентов ускорения и эффективности (рис. 4.20-4.21). Часто, при анализе качества параллельных алгоритмов, рассчитываются как относительные, так и абсолютные коэффициенты ускорения и эффективности. В данном случае это означает, что для каждого способа оценки локальной погрешности использовался либо наименее затратный последовательный алгоритм, либо алгоритм, который был распараллелен.

Наилучшими динамическими характеристиками обладают вложенный экспоненциальный метод и экспоненциальный с правилом Рунге, для них характеристики реального параллелизма практически не различимы. Для всех алгоритмов, базирующихся на экспоненциальной схеме, абсолютные показатели качества параллельных алгоритмов лучше, чем для стандартных схем.

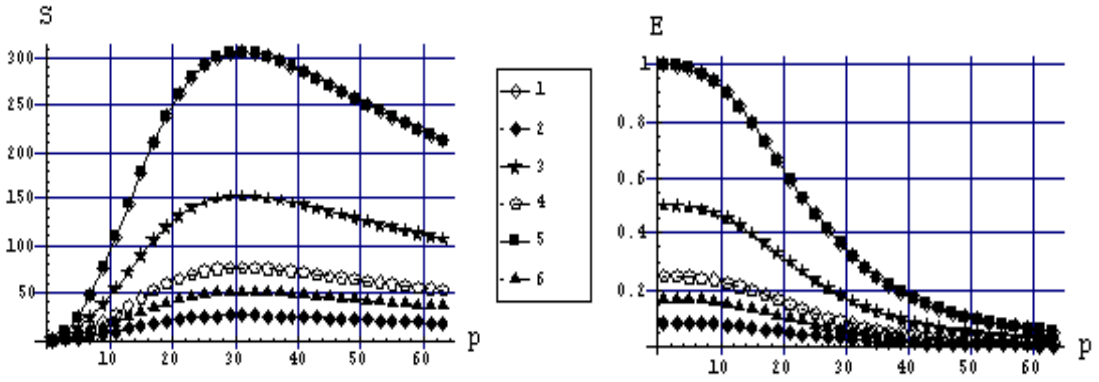


Рисунок 4.20 – Относительные коэффициенты ускорения и эффективности параллельных алгоритмов решения линейной задачи Коши с контролем локальной погрешности от ширины процессорной решетки

Например, при использовании вложенных форм коэффициент эффективности экспоненциальной схемы практически в два раза выше, чем стандартной (рис. 4.21).

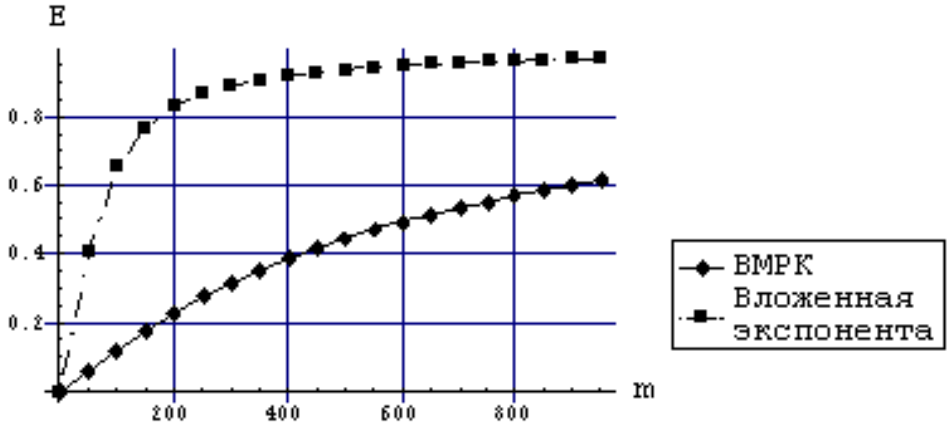


Рисунок 4.21 – Коэффициенты эффективности вложенных методов решения линейной задачи Коши от размерности СЛОДУ

До сих пор при построении параллельных алгоритмов главы 4 в качестве топологического решения использовался 2D-тор. При переходе на физическую коммутационную сеть иной топологии необходимо иметь гибкий программный механизм логического отображения одной топологии на другую.

Математическим аппаратом, способным решить поставленную задачу, являются **двоичные рефлексивные коды Грея** [10, 128].

Логическое отображение двумерной замкнутой решетки на гиперкуб выполняется по правилу: каждому процессорному элементу решетки с координатами (i, j) будет соответствовать процессор гиперкуба с номером $G(i,p)||G(j,p)$, где $G(i,p)$ – значение кода Грея, i – порядковый номер значения в коде, p – его длина, $||$ – операция конкатенации. Например, рисунок 4.22 иллюстрирует отображение решетки 4×4 на гиперкуб размерности 4.

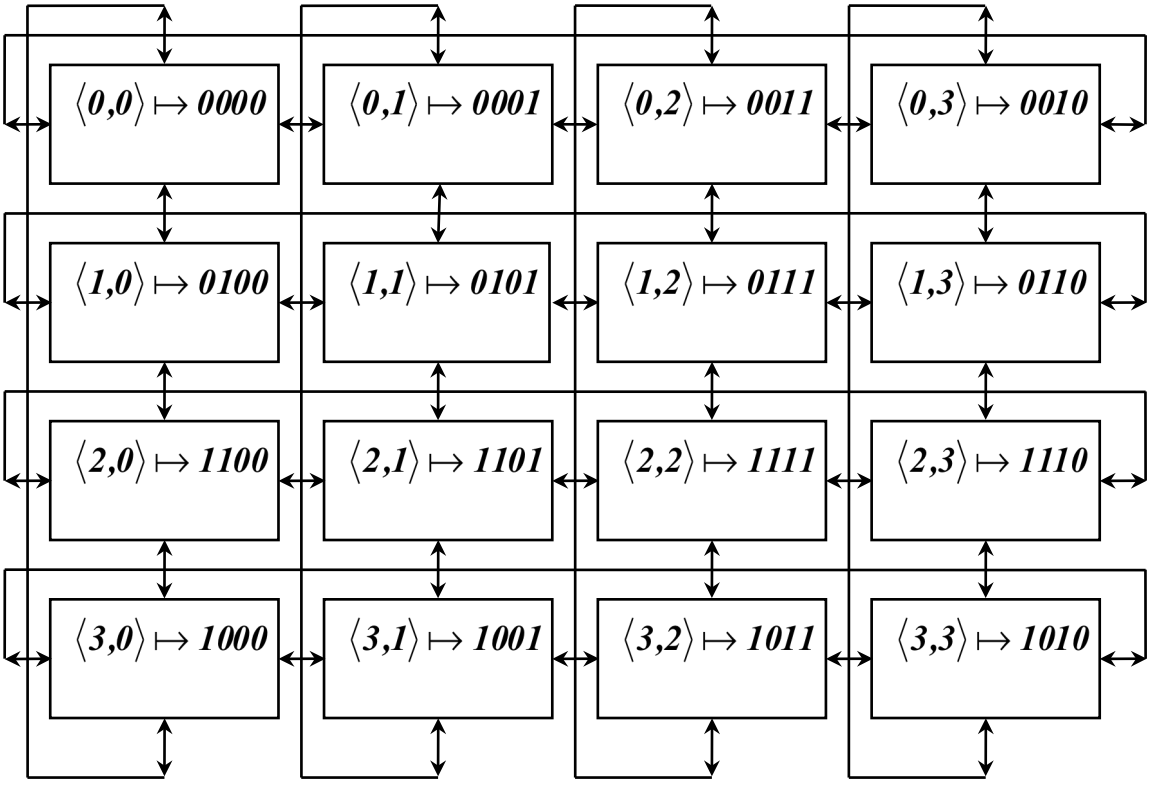


Рисунок 4.22 – Схема логического отображения топологии 2D-тор размерности 4×4 на топологию гиперкуб $H(4)$

Заметим, что в результате выполнения такого отображения процессоры, являющиеся “непосредственными соседями” в топологии тор, будут иметь те же свойства в топологии гиперкуб.

Таким образом, все параллельные алгоритмы линейной задачи Коши на основе матричной экспоненты, разработанные для топологии тор, только использованием программных средств языка *C++* и библиотеки *MPI*, могут быть перенесены без изменения алгоритма на коммуникационную сеть гиперкуб [41].

Построение аналогичного логического отображения описанного множества алгоритмов на параллельные архитектуры с физической топологией кольцо влечет за собой увеличение времени выполнения коммуникационной составляющей алгоритмов и имеет заведомо худшие характеристики параллелизма, чем в случае топологий тор и гиперкуб.

4. 6 Выводы

Разработаны параллельные методы численного решения однородной и неоднородной линейной задачи Коши с постоянными коэффициентами на базе аналитического представления решения в виде матричной экспоненты с тремя встроенными способами оценки локальной апостериорной погрешности:

1) экспоненциальный метод и правило Рунге;
2) экспоненциальный метод с технологией локальной экстраполяцией Ричардсона;
3) вложенные методы на основе матричной экспоненты.

Предложены вычислительные схемы параллельных алгоритмов определения локальной апостериорной погрешности на основе явных стандартных одношаговых методов решения линейной задачи Коши.

Проведен сравнительный анализ параллельных алгоритмов на основе стандартных схем и методов с использованием матричной экспоненты. Вне зависимости от способа вычисления локальной погрешности, параллельные алгоритмы, основанные на применении экспоненциального метода решения, имеют меньшую вычислительную сложность и лучшие показатели качества параллелизма по сравнению со своими стандартными аналогами:

- для правила Рунге - приблизительно в s раз;
- для вложенных методов - в $s/2$ раз;
- для локальной экстраполяции Ричардсона - в s_0 раз;

где s – число стадий сравниваемого стандартного ЯМРК,

s_0 – число стадий опорного метода для ЯМРК.

Значения s и s_0 могут быть поправлены на величину $c, c = 1,2...$, задающую разность между числом стадий и порядком точности явного метода, которая возрастает с ростом порядка метода: $r \uparrow \Rightarrow c \uparrow \Rightarrow s + c \uparrow$.

Исследована эффективность альтернативных способов определения локальной апостериорной погрешности при решении линейной задачи Коши для СОДУ, выявлены предпочтительные области применения различных алгоритмов вычисления шаговой погрешности:

– наименее трудоемким способом определения локальной апостериорной погрешности решения для управления шагом интегрирования являются методы вложенных форм на основе матричной экспоненты: $O[2 \cdot m^2 / p^2]$;

– наибольшей временной сложностью обладают схемы локальной экстраполяции и прежде всего на основе стандартной схемы: $O[sr^2 m^2 / p^2]$.

Для повышения эффективности решения СЛОДУ разработаны параллельный метод на основе комбинации быстрого рекурсивного и систолического алгоритмов; модификация систолического алгоритма, снимающая ограничения на размерность перемножаемых матриц. Применение разработанного рекурсивно-систолического метода позволило в $(8/7)^d, d = 1,2...$ раз ускорить выполнение наиболее ресурсоемкой операции при решении СЛОДУ на основе экспоненты. Для предложенного алгоритма определены оптимальные значения глубины рекурсии и величины минимального блока перемножаемых матриц.

Разработаны вычислительные схемы отображения полученных параллельных алгоритмов на структуры параллельных ВС с распределенной памятью и топологией решетка/тор, построено логическое отображение топологии тор на гиперкуб. Для каждого из приведенных алгоритмов рассчитаны трудоемкости получения численного решения, получены характеристики параллелизма, которые свидетельствуют о высокой эффективности экспоненциальных методов.

ЗАКЛЮЧЕНИЕ

Исследования, приведенные в монографии, представляют собой решение актуальной научной задачи, заключающейся в повышении эффективности функционирования параллельных компьютерных систем за счет разработки и обоснования численных методов решения широкого класса научно-технических задач большой размерности.

1. Получены эффективные параллельные вычислительные схемы методов оценки локальной погрешности для технологии локальной экстраполяции, дублирования шага и вложенных форм при численном решении нелинейной задачи Коши на основе явных одношаговых схем.

Установлено, что наименее трудоемкими являются схемы с симметричными опорными методами малых порядков точности в сочетании с четными последовательностями. Показано, что параллельные методы на основе пропорционального и комбинационного распределения данных наиболее эффективны, поскольку позволяют увеличить балансировку загрузки ВС.

2. Предложены и теоретически обоснованы параллельные методы оценки локальной апостериорной погрешности численного решения задачи Коши для одного дифференциального уравнения на основе блочных неявных одношаговых разностных схем:

– блочный k-точечный метод с правилом дублирования шага;

– вложенные блочные методы на основе k и $(k+1)$-точечных методов и с использованием метода последовательного повышения порядка точности;

– локальная экстраполяция с блочным одношаговым опорным методом.

Проведен сравнительный анализ вычислительных качеств и эффективности неявных одношаговых методов решения общей задачи Коши на основе блочных k-точечных методов и s-стадийных полностью неявных методов типа Рунге-Кутты (ПНМРК) одного и того же порядка точности.

Установлено, что динамические характеристики блочных многоточечных методов практически в s раз превосходят соответствующие характеристики многостадийных методов для последовательной реализации и в $2s$ для параллельной, где s - число стадий ПНМРК.

Выполнено обобщение разработанных неявных параллельных методов решения нелинейной задачи Коши для систем обыкновенных дифференциальных уравнений.

3. Разработаны параллельные алгоритмы решения линейной задачи Коши на базе аналитического представления решения в виде матричной экспоненты:

– экспоненциальный метод и правило Рунге;

– экспоненциальный метод с локальной экстраполяцией;

– вложенные методы на основе матричной экспоненты.

Проведен сравнительный анализ параллельных алгоритмов на основе стандартных схем и алгоритмов с использованием матричной экспоненты. Вне зависимости от способа вычисления локальной погрешности, параллельные алгоритмы, основанные на применении экспоненциального метода решения, имеют меньшую вычислительную сложность и лучшие показатели качества параллелизма по сравнению со своими стандартными аналогами:

1) для правила Рунге в s раз;
2) вложенных методов в $s/2$ раз;
3) для локальной экстраполяции Ричардсона в s_0 раз,

здесь s – число стадий сравниваемого явного метода Рунге-Кутты (ЯМРК), s_0 – число стадий опорного метода для ЯМРК.

Разработан эффективный рекурсивно-систолический метод умножения плотнозаполненных матриц, ускоряющий выполнение этой ресурсоемкой операции экспоненциального метода.

4. Получены динамические характеристики потенциального параллелизма: ускорение, эффективность, степень параллелизма для всех разработанных методов.

Разработаны вычислительные схемы отображения полученных параллельных алгоритмов на структуры параллельных ВС с распределенной памятью и различными топологиями межпроцессорных связей:

- линейка/кольцо;
- решетка/тор;
- гиперкуб.

Исследована эффективность полученных вычислительных схем отображения параллельных алгоритмов на структуры ВС в зависимости

от размерности процессорных полей, модели вычислений, ориентированной на SIMD-, MIMD- и CLUSTER-структуры различных топологий, коммуникационных констант.

Для наиболее эффективного способа оценки локальной апостериорной погрешности методов вложенных форм оценена степень масштабируемости на основе изоэффективного анализа.

5. Определены приоритетные области применения параллельных алгоритмов в сочетании с параллельной архитектурой:

– для нелинейной задачи Коши наиболее эффективными с точки зрения вычислительных и коммуникационных затрат являются параллельные методы средних и малых порядков на основе явных вложенных форм при использовании MIMD-систем топологии гиперкуб или кластеры с высокоскоростными сетями;

– среди методов решения жестких СОДУ наиболее эффективными являются параллельные методы на основе неявных блочных вложенных форм для MIMD структур с топологией гиперкуб;

– для решения линейной задачи Коши меньшую временную сложность имеют вложенные экспоненциальные методы, наиболее эффективной топологией является решетка/тор при наличии как синхронных, так и асинхронных систем;

– преимущества численных схем, основанных на технологии локальной экстраполяции, проявляются при поиске высокоточных решений и доминировании обращения к правой части СОДУ над остальными вычислениями.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Grand Challenges: High performance computing and communications // A report by the Committee on Physical, Mathematical and Engineering Science, NSF/CISE, 1800 G. Street NW, Washington, DC 20550, 2001.

2. Забродин А.В. Параллельные вычислительные технологии. Состояние и перспективы // Материалы первой молодежной школы ”Высокопроизводительные вычисления и их приложения”. – Режим доступа: ftp://parallel.ru/parallel/chg1999.

3. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608с.

4. Воеводин В.В. Математические проблемы параллельных вычислений // Тезисы докладов II Всероссийской научной конференции "Методы и средства обработки информации". 5-7 октября 2005. – Москва: МГУ, 2005. – С. 22-33.

5. Воеводин В.В. Супервычисления и структура алгоритмов // Материалы первой молодежной школы ”Высокопроизводительные вычисления и их приложения”. – Режим доступа: ftp://parallel.ru/parallel/chg1999.

6. Воеводин В.В. Информационная структура алгоритмов. – М.: Изд-во МГУ, 1997. – 139с.

7. Воеводин В.В. Математические основы параллельных вычислений. – М.: Изд-во МГУ, 1991. – 345с.

8. Воеводин В.В. Математические модели и методы в параллельных процессах. – М.: Наука, 1986. – 296с.

9. Voevodin V.V. Information structure of sequental programs // Rus. J. Num. An. and Math. Modeling. 1995. – Vol. 10, № 3. – P. 279-286.

10. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем.– Нижний Новгород: ННГУ, 2001. – 122с.

11. Гергель В.П. Теория и практика параллельных вычислений. – Москва: Бином. Лаборатория знаний, 2007. – 423с.

12. Grama A., Gupta A., Kumar V. Isoefficiency: Measuring the scalability of parallel algorithms and architectures // IEEE Parallel and Distributed technology, 1993. – P. 12-21.

13. Kumar V., Gupta A. Analyzing scalability of parallel algorithms and architectures // Journal of Parallel and Distributed Computing, 22(3), 1994. – P. 379-391(2nd edn., 2003).

14. Gupta A., Kumar V. Scalability of parallel algorithm for matrix multiplication // Technical report TR-91-54, Department of CSU of Minneapolis, 1994.

15. Забродин А.В. СуперЭВМ МВС-100, МВС-1000 и опыт их использования при решении задач механики и физики // Математическое моделирование, 2000, т. 12, № 5. – С. 61-66.

16. Забродин А.В., Левин В.К. Опыт разработки параллельных вычислительных технологий, создание и развитие семейства МВС // Тезисы докладов конференции "Высокопроизводительные вычисления и их приложения". – М.: МГУ, 2000. – С. 3-7.

17. Забродин А.В., Левин В.К., Каратанов В.В и др. Производительность 24-процессорного фрагмента многопроцессорной вычислительной машины МВС-1000М // Тезисы докладов конференции "Высокопроизводительные вычисления и их приложения". – М.:МГУ, 2000. – С. 9-11.

18. Самофалов К.Г., Луцкий Г.М. Структуры и организация функционирования ЭВМ и систем. – К.: Вища школа, 1978. – 392с.

19. Малиновский Б.Н., Боюн В.П., Козлов Л.Г. Вопросы построения высокопроизводительных средств обработки информации. // Управляющие системы и машины. – 1976. №6. – С. 19-22.

20. Малиновский Б.Н., Боюн В.П., Козлов Л.Г. Многопроцессорные вычислительные структуры. // Межведомственный тематический научный сборник. – Таганрог: Радиотехн. ин-т, 1987. – Вып. 9. – С. 19-21.

21. Малиновский Б.Н., Боюн В.П., Козлов Л.Г. Принципы повышения производительности проблемно-ориентированных процессоров для решения сложных научно-технических задач // Тезисы докл. IV Всесоюзн. школы-семинара "Распараллеливание обработки информации". – Львов: Физ.-мех. ин-т. АН УССР, 1985. – Ч. 3. – С. 3-6.

22. Боюн В.П., Козлов Л.Г. Высокопроизводительные средства обработки информации для систем реального времени. – Киев, 1976. – С. 31-38.

23. Боюн В.П., Козлов Л.Г., Терещенко В.И. Об одном подходе к построению мультипроцессорных систем для решения обыкновенных

дифференциальных уравнений // Управляющие машины и системы. – 1982. - №2. – С. 42-45.

24. Корнеев В.В. Параллельные вычислительные системы. – М.: Нолидж, 1999. – 320с.

25. Корнеев В.В. Параллельное программирование в MPI. Москва-Ижевск: Институт компьютерных исследований, 2003. – 304с.

26. Каляев И.А., Мельник Э.В. Интеллектуальные многопроцессорные вычислительные системы // Тезисы доклада третьей международной научной конференции "Высокопроизводительные вычислительные системы". – Таганрог: Изд-во ТРТУ, 2006. – С. 14-19.

27. Каляев А. В., Левин И.И. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. – М.: Янус-К, 2003. – 380с.

28. Отчет о научно-исследовательской работе ГТ-11-2000 "Научные основы оптимизации структур высокопроизводительных вычислительных систем и методы реализации параллельных алгоритмов" // Л.П. Фельдман, О.А. Дмитриева, И.А. Назарова, Т.В. Михайлова. – Донецк: ДонГТУ, 2002. – 173с., № гос. регистрации 0202U002619.

29. Отчет о научно-исследовательской работе Д-1-03 "Методы алгоритмизации, топологического отображения и оптимизации структур параллельных и распределенных вычислительных систем" // Л.П. Фельдман, О.А. Дмитриева, И.А. Назарова, Т.В. Михайлова. – Донецк: ДонГТУ, 2005. – 231с., № гос. регистрации 0103U001824.

30. Дьяконов В. Mathematica 4:учебный курс. – СПб.: Питер, 2001. – 656с.

31. Дьяконов В.П. Mathematica 4 с пакетами расширений. – М.: Нолидж, 2000. – 606с.

32. Антонов А.С. Введение в параллельные вычисления. – М.: Изд-во МГУ, 2002. – 69с.

33. Group W., Lusk E., Skjellum A. Using MPI. Portable parallel programming with Message Passing Interface. – MIT Press, 1999. – 303с.

34. MPI (Message Passing Interface): теория и примеры программ. – Режим доступа: http://www.iitk.ac.in/cc/param/index.html

35. Евсеев И. MPI для начинающих. Учебное пособие и примеры. – Режим http://rsusu1.rnd.runnet.ru/ доступа: ncube/koi8/mpibeg.html

36. MPI-FM пакет MPICH для высокопроизводительных кластеров. – Режим доступа: http://www-csag.ucsd.edu/projects/comm/mpi-fm.html

37. Арушунян О.Б., Залеткин С.Ф., Калиткин Н.Н. Тесты для вычислительного практикума по обыкновенным дифференциальным уравнениям // Вычислительные методы и программирование, 2002, т.3. – С. 11-19.

38. Фельдман Л.П., Назарова И.А. Эффективность параллельных алгоритмов оценки локальной апостериорной погрешности для численного решения задачи Коши // Электронное моделирование, т. 29, № 3, 2007. – С. 11-25.

39. Фельдман Л.П., Назарова И.А. Параллельные алгоритмы численного решения задачи Коши для систем обыкновенных дифференциальных уравнений // Математическое моделирование, т.18, № 9, 2006. – С. 17-31.

40. Фельдман Л.П., Назарова И.А., Хорошилов А.В. Параллельные блочные алгоритмы умножения матриц для мультикомпьютеров с распределенной памятью // Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка, випуск 8(120): – Донецьк, ДонНТУ, 2007. – С. 297-309.

41. Назарова И.А. Экспоненциальные методы решения линейной задачи Коши с альтернативными способами оценки локальной погрешности для массивно-параллельных компьютерных систем // Научно-теоретический журнал ИПИИ НАН Украины «Искусственный интеллект», №4, 2007. – Донецк: ИПИИ, 2007. – С. 474-482.

42. Назарова И.А. Эффективность применения технологии локальной экстраполяции в параллельных алгоритмах численного решения задачи Коши // Научно-теоретический журнал ИПИИ НАН Украины «Искусственный интеллект», №3, 2006. – Донецк: ИПИИ, 2006. – С. 192–202.

43. Назарова И.А. Параллельные полностью неявные методы численного решения жестких задач для СОДУ // Научно-теоретический

журнал ИПИИ МОН и НАН Украины «Искусственный интеллект», №3, 2005. – Донецк: ИПИИ, 2005. – С. 185-193.

44. Назарова И.А. Повышение эффективности параллельных вычислительных систем при решении задачи Коши неявными методами Рунге-Кутты // Научные труды Донецкого национального технического университета. Выпуск 93. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2005) – Донецк: ДонНТУ, 2005. – С. 58-67.

45. Назарова И.А. Эффективность численного решения нежестких СОДУ с контролем локальной погрешности для компьютеров с распределенной памятью // Научно-теоретический журнал ИПИИ НАН Украины «Искусственный интеллект», №3, 2004. – Донецк: ИПИИ, 2004. – С. 212-215.

46. Фельдман Л.П., Назарова И.А. Применение технологии локальной экстраполяции для высокоточного решения задачи Коши на SIMD-структурах // Научные труды Донецкого национального технического университета. Выпуск 70. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2003) – Донецк: ДонНТУ, 2003. – С. 98-107.

47. Фельдман Л.П., Назарова И.А. Особенности использования методов Рунге-Кутты при моделировании параллельных процессов // Научные труды Луганского отделения Международной Академии информатизации №2(7). – Луганск, 2003. – С. 73-77.

48. Фельдман Л.П., Назарова И.А. Параллельная реализация численного решения нежестких обыкновенных дифференциальных уравнений вложенным методом Кутты-Мерсона // Научные труды Донецкого национального технического университета. Серия: проблемы моделирования и автоматизации проектирования динамических систем, выпуск 52. – Донецк, 2002. – С. 106-112.

49. Фельдман Л.П., Назарова И.А. Эффективность параллельных алгоритмов вложенных методов Рунге-Кутты при моделировании сложных динамических систем // Материалы II Международного научно-практического семинара «Высокопроизводительные параллельные вычисления на кластерных системах». – Нижний Новгород: Изд-во НГУ, 2002. – С. 294-301.

50. Фельдман Л.П., Назарова И.А. Моделирование сложных динамических систем на базе вложенных методов Рунге-Кутты //

Сборник трудов международной научно-технической конференции. Компьютерные технологии в управлении, диагностике и образовании (КТУДО-2002). – Тверь, Тверской государственный технический университет, 2002. – С. 150-152.

51. Назарова И.А. Параллельные методы решения СОДУ большой размерности при моделировании сложных систем // Матеріали V Міжнародної науково-практичної конференції студентів, аспірантів та молодих вчених «Системний аналіз та інформаційні технології». – К.: НТУУ КПІ, 2003. – С. 87-88.

52. Фельдман Л.П., Назарова И.А. Методы контроля шаговой погрешности при параллельном решении систем обыкновенных дифференциальных уравнений // Материалы Двенадцатой Международной конференции по вычислительной механике и современным прикладным программным системам, Владимир.– М.: Изд-во МАИ, 2003. – т. 2. – С. 619-620.

53. Назарова И.А., Шаповалов В.А. Моделирование сложных динамических систем на высокопроизводительных компьютерах с распределенной памятью // Системний аналіз та інформаційні технології: Матеріали VI Міжнародної науково-практичної конференції.– К.: НТУУ «КПІ», 2004. – С. 189-191.

54. Назарова И.А., Фельдман Л.П. Эффективность параллельного численного решения нежестких систем обыкновенных дифференциальных уравнений с контролем локальной погрешности // Материалы XX Международного семинара по струйным, отрывным и нестационарным течениям. – Санкт-Петербург. – СПб.: ИПЦ СПбГУТД, 2004. – С. 201-202.

55. Назарова И.А., Фельдман Л.П. Масштабируемый параллельный алгоритм численного решения линейных СОДУ для компьютеров с распределенной памятью // Материалы V Международной конференции по неравновесным процессам в соплах и струях (NPNJ-2004). – М.: Вузовская книга, 2004. – С. 153-155.

56. Фельдман Л.П., Назарова И.А. Эффективность способов оценки апостериорной локальной погрешности при параллельном решении систем линейных однородных ОДУ // Высокопроизводительные параллельные вычисления на кластерных системах. Материалы четвертого Международного научно-

практического семинара и Всероссийской молодежной школы. – Самара, 2004. – С. 255-263.

57. Назарова И.А. Параллельные алгоритмы численного решения задачи Коши для СОДУ / Донбас-2020: наука і техніка – виробництву: Матеріали III науково-практичної конференції. м. Донецьк, 30-31 травня 2004. – Донецьк, ДонНТУ Міністерства освіти і науки, 2004. – С. 522 – 527.

58. Назарова И.А., Фельдман Л.П. Разработка и анализ эффективности параллельных алгоритмов итерационных методов решения СОДУ для мультипроцессоров с распределенной памятью // Материалы международной конференции по вычислительной механике и современным прикладным программным системам (ВМСППС-2005). – М.: Вузовская книга, 2005. – С. 343-345.

59. Назарова И.А. Масштабируемые параллельные алгоритмы численного решения жестких систем обыкновенных дифференциальных уравнений для многопроцессорных вычислительных систем с распределенной памятью // Материалы международной научно-технической конференции “Интеллектуальные и многопроцессорные системы: ИМС‘2005”.– Таганрог-Донецк-Минск: Изд-во ТРТУ, т.1, 2005. – С. 218-220.

60. Назарова И.А. Масштабируемый параллельный алгоритм численного решения задачи Коши для ВС с распределенной памятью // Тези доповідей Міжнародної науково-практичної конференції “Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій”. – Запоріжжя, 2006. – С. 173-175.

61. Назарова И.А. Масштабируемые параллельные алгоритмы численного решения задачи Коши для СОДУ / Сборник трудов международной конференции “Моделирование – 2006”. – Киев: Институт проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины, 2006. – С. 335-340.

62. Фельдман Л.П., Назарова И.А. Эффективность отображения экстраполяционных параллельных алгоритмов численного решения СОДУ на вычислительные структуры различных топологий. // Материалы седьмой международной научно-технической конференции “Интеллектуальные и многопроцессорные системы: ИМС‘2006”.– Таганрог: Изд-во ТРТУ, т.1, 2006. – С. 269–272.

63. Фельдман Л.П., Назарова И.А. Параллельные алгоритмы численного решения задачи Коши для мультипроцессоров с распределенной памятью // Труды III международной конференции «Параллельные вычисления и задачи управления» PACO 2006 памяти И.В. Прангишвили. – М.: Институт проблем управления им. В.А. Трапезникова РАН, 2006. – С. 184-196.

64. Назарова И.А., Фельдман Л.П. Параллельные экстраполяционные схемы высокоточного решения задачи Коши для мультикомпьютеров с распределенной памятью // Материалы VI Международной конференции по неравновесным процессам в соплах и струях (NPNJ-2006), 26 июня-1июля 2006 г. Санкт-Петербург. – М.: Вузовская книга, 2006. – С. 255-257.

65. Фельдман Л.П., Назарова И.А. Параллельная реализация технологии локальной экстраполяции симметричных методов решения задачи Коши для кластерных систем // Материалы XV международной конференции по вычислительной механике и современным прикладным программным системам (ВМСППС-2007), Алушта, Крым, 25-31 мая 2007. – М.: Вузовская книга, 2007. – С. 485-487.

66. Назарова И.А. Алгоритмические методы повышения эффективности параллельных ВС при численном решении СОДУ с контролем погрешности на шаге / И. А. Назарова // Материалы II международной конференции “Моделирование и компьютерная графика”, Донецк: ДонНТУ, 2007. – С. 202-205.

67. Flynn M. Very high-speed computing systems. Proceeding of the IEEE №54 (12), 1966. – P.1901-1909.

68. Flynn M. Some Computer Organisations and Their Effectiveness // IEEE Trans. Computers. 1972. . – –V.21. N 9. – P.948-960.

69. Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем. – СПб.: Питер, 2004. – 668с.

70. Бройнль Т. Параллельне програмування: Початковий курс: Навч. посібник / Вступ. слово А. Ройтера; Пер. з нім. В.А.Святного. – К.: Вища шк., 1997. – 358с.

71. Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ-Петербург, 2002. – 396с.

72. Барский А.Б. Параллельные информационные технологии. – Москва: ИУИТ. Бином. Лаборатория знаний, 2007. – 503с.

73. Хокни Р., Джессхоуп К. Параллельные ЭВМ: Архитектура, программирование и алгоритмы. – М.: Радио и связь, 1986. – 392с.

74. Жуков И.А. Классификация архитектур вычислительных систем // Управляющие системы и машины , 1995, №6. – С.52-56.

75. Букатов А.А., Дацюк В.Н., Жегуло А.И. Программирование многопроцессорных ВС. – Ростов-на Дону: Изд-во ООО ЦВВР, 2003. – 208с.

76. Богачев К.Ю. Основы параллельного программирования. – М.: БИНОМ. Лаборатория знаний, 2003. – 342с.

77. Информационно-аналитические материалы по параллельным вычислениям научно-исследовательского вычислительного центра (НИВЦ) МГУ. – Режим доступа: http://parallel.ru.

78. Основные классы современных параллельных компьютеров. – Режим доступа: http://parallel.ru/computers/classes.html

79. Т-Платформы. – Режим доступа: http://www.t - platforms.ru/clusters/communications.html

80. Наиболее используемые коммуникационные технологии. – Режим доступа: http://parallel.ru/computers/interconnects.html

81. Руководство по коммутаторам Myrinet. Официальное описание. – Режим доступа:http://www.myri.com/myrinet/m3switch/guide/myrinet-2000_switch_guide.pdf

82. Официальный портал разработчика SCI. – Режим доступа: http://www.dolphinics.com

83. Официальный портал Intel, раздел коммуникаций Infiniband – Режим доступа: http://www.intel.com/technology/infiniband

84. Список 500 наиболее мощных суперкомпьютеров мира. – Режим доступа: http://www.top500.org

85. Список 50 наиболее мощных суперкомпьютеров СНГ. – Режим доступа: http://www.supercomputers.ru/page=rating

86. Foster I. Designing and Building Parallel Programs. – Addison-Wesley, 1999. – 302с.

87. Берзигияров П.К., Султанов В.Г. Технология разработки масштабируемых параллельных вычислений для SMP систем на базе MPI // Материалы первой молодежной школы

«Высокопроизводительные вычисления и их приложения». – Режим доступа: ftp://parallel.ru/parallel/chg1999.

88. Amdahl G. Validity of the single processor approach to achieving large scale computing capabilities // In AFIPS Conference proceeding, Washington, D.C.: Thompson Books, Vol. 30. – P.483-485.

89. Gustavson J.L. Reevaluating Amdahl's law // Communications of the ACM. 31(5). – P.532-533.

90. Pastor L., Bosque J. L. An efficiency and scalability model for heterogeneous clusters // Proceedings of Cluster 2001, 8-11 October 2001, Newport Beach, CA, USA. IEEE Computer Society. – P.427-434.

91. Dongarra J., R. van de Geun, Walker D. Scalability Issues Affecting the Design of a Dense Linear Algebra Library // Journal of Parallel and Distributed Computing, 22, 1994. – P.523-537.

92. Kalinov A. Scalability Analysis of Matrix-Matrix Multiplication on Heterogeneous Clusters, Proceedings of 3rd ISPDC/HeteroPar'04, Cork, Ireland, July 05 - 07, 2004, IEEE CS Press. – P.303-309.

93. Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. – М.: Мир, 1990. – 512с.

94. Хайрер Э., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Жесткие и дифференциально-алгебраические задачи. – М.: Мир,1999. – 685с.

95. Холл Дж., Уатт Дж. Современные численные методы решения обыкновенных дифференциальных уравнений. – М.: Мир, 1999. – 311с.

96. Miranker W.L. A survey of parallelism in numerical analysis. SIAM Review, 1971, vol. 13. – P.524-547.

97. Houwen P.J., Sommeijer B.P. Parallel ODE solvers // Proceedings of the International Conference on Supercomputing, 1990, ACM Press. – P. 71-81.

98. Parallel predictor-corrector methods of Runge-Kutta type / Houwen P.J. van der, Nguyen hun Cong. // Rept/ Cent. Math. And Comput. Sci. – 1992. – Nm R9220. – P. 1-10.

99. Houwen, P.J. van der & Sommeijer, B.P. Runge-Kutta methods on parallel computers // J. Z. Angen Math. Mech. – 1992. - №68. – P. 3-10.

100. Jackson K.R., Norsett S.P. The potential for parallelism in Runge-Kutta methods. Part I: RK formulas in standart form // Rep. Depart. of Comp. Sc. 1990 – № 239/90. – P. 41-60.

101. Houwen, P.J. van der & Sommeijer, B.P. Parallel iteration of high-order Runge-Kutta methods with stepsize control // J. Comp. Appl. Math. – 1990. №29. – P. 111-127.

102. Г.Ю. Куликов. Численное решение задачи Коши для системы дифференциально-алгебраических уравнений с помощью неявных методов Рунге-Кутты с нетривиальным предиктором // Журнал вычислительной математики и математической физики, 1998, том 38, № 1. – С.68-84.

103. Shampine L.F.&Watts H.A. Block implicit one-step methods // Math. Comp. 23, 108. – P. 252-266.

104. Houwen P.J., Sommeijer B.P. Parallel ODE solver // Proceedings of the International Conference on Supercomputing. – ACM Press, 2000. – P.71-81.

105. Houwen P.J., Sommeijer B.P. CWI Contribution to the development of parallel Runge-Kutta methods / Preprint NM-R9506, CWI, Amsterdam, 2001. –23p.

106. Houwen P.J., Swart J.J. Triangularly implicit methods for the numerical solution of ordinary differential equations / Preprint NM-R9510, CWI, Amsterdam, 2003. –23p.

107. Worland P.B. Parallel methods for the numerical solution of ordinary differential equations // IEEE Trans. Comp. C. – 25, 10(1976). – P.1045-1048.

108. Молчанов И.Н. Введение в алгоритмы параллельных вычислений. – Киев: Наукова думка, 1990. – 128с.

109. Фельдман Л.П. Сходимость и оценка погрешности параллельных одношаговых блочных методов моделирования динамических систем с сосредоточенными параметрами // Наукові праці ДонДТУ. Серія: Інформатика, кібернетика та обчислювальна техніка, випуск 15, Донецьк: ДонДТУ, 2000. – С.34-39.

110. Фельдман Л.П., Дмитриева. О.А. Разработка и обоснование параллельных блочных методов решения обыкновенных дифференциальных уравнений на SIMD-структурах // Наукові праці ДонДТУ. Серія: Проблеми моделювання та автоматизації проектування динамічних систем, випуск 29:- Донецьк: ДонДТУ, 2001. – С.70-79.

111. Feldman L., Dmitriewa O.A., Gerber S. Abbildung der blockartigen Algoritmen auf die Parallelrechnerarchitekture / Simulationstechnik, 17. Symposium in Magdeburg, Sept. 2003: SCS-Europe BVBA (ISBN 3-936150-27-3), Magdeburg, Germany, 2003. – P.359-364.

112. Feldman L., Svjatnyj V., Dmitriewa O.A. Stabilitat von parallelen Simulationsverfahren fur dynamische Systeme mit konzentrierten Parametern / Simulationstechnik, 17. Symposium in Magdeburg, Sept. 2003: SCS-Europe BVBA (ISBN 3-936150-27-3), Magdeburg, Germany, 2003. – P.105-110.

113. Фельдман Л.П. Общие линейные блочные многошаговые методы решения // Наукові праці ДонНТУ. Серія: Інформатика, кібернетика і обчислювальна техніка, випуск 8(120): – Донецьк: ДонНТУ, 2007. – С. 282-297.

114. Вержбицкий В.М. Основы численных методов. – М.: Высшая школа, 2002. – 840с.

115. Арушанян О.Б., Залеткин С.Ф. Численное решение обыкновенных дифференциальных уравнений на Фортране.– М.: МГУ,1990.–336с.

116. Крылов В.И., В.В. Бобков В.В., Монастырский П.И. Вычислительные методы, том I. – М.: Наука, 1976. – 303с.

117. Крылов В.И., В.В. Бобков В.В., Монастырский П.И. Вычислительные методы, том II. – М.: Наука, 1977. – 399с.

118. Иванов В.В. Методы вычислений на ЭВМ. – К.: Наукова думка, 1986. – 584с.

119. Голуб Д., Ван Лоун Ч. Матричные вычисления: Пер. с англ. – М.: Мир, 1999. – 548с.

120. Деммель Дж. Вычислительная линейная алгебра. Теория и приложения. Пер. с англ. – М.: Мир, 2001. – 430с.

121. Demmel J., Higham N.J. Stability of block algorithms with fast Level 3 BLAS // ACM Trans. Math. Software, 18, 1992. − P. 274-291.

122. Choi J., Dongarra J., Walker D. PUMMA: Parallel universal matrix multiplication algorithms on distributed memory concurrent computers. – Режим доступа: http://citeseer.ist.psu.edu/choi93pumma.html

123. Strassen V. Gaussian elimination is not optimal // Numer. Math. 13. − P. 354-356.

124. Pan V. How can we speed up matrix multiplication // SIAM Rev., 26, 1984. − P. 393-416.

125. Winograd S. A new algorithm for inner product // IEEE Trans. Comp. C-17. − P. 693-694.

126. Bailey D.H. Extra high speed matrix multiplication on CRAY-2 // SIAM J. Sci. and Stat. Comp. 9. − P. 603-607.

127. Bailey D.H., Lee K., Simon H.D. Using Strassen's algorithm to accelerate the solution of linear systems. J. Supercomputing, 4: 97-371, 1991.

128. Андерсон Дж. Дискретная математика и комбинаторика. – М.: Мир, 2001. – 960с.

129. Назарова И.А. Повышение эффективности решения жестких динамических задач для мультикомпьютеров с распределенной памятью / Сборник трудов международной конференции "Моделирование – 2008". – Киев: Институт проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины, 2008. – С. 471-476.

130. Назарова И.А. Повышение эффективности параллельного численного решения жестких задач на основе неявных блочных одношаговых методов // Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка, випуск 8(120): – Донецьк, ДонНТУ, 2008. - С. 41-46.

131. Назарова І.А. Підвищення ефективності паралельного розв'язання лінійної задачі Коші на основі методу рекурсивного множення матриць // Научно-теоретический журнал ИПИИ НАН Украины «Искусственный интеллект», №3, 2008. – Донецк: ИПИИ, 2008. – С.706-714.

132. Фельдман Л.П., Назарова І.А., Шматько А.Е. Паралельний блоковий рекурсивно-систоличний метод реалізації матричного добутку // Материалы международной научно-технической конференции "Искусственный интеллект. Интеллектуальные системы: ИИ-2008.– Таганрог-Донецк-Минск: Изд-во ТРТУ, т.2, 2008. – С. 380–382.

133. Фельдман Л.П., Назарова И.А. Многопроцессорная реализация неявных многоточечных методов решения жестких задач // Материалы VII Международной конференции по неравновесным процессам в соплах и струях (NPNJ'2008) .– М.: Вузовская книга, 2008. – с. 255–257.

134. Назарова И.А. Анализ масштабируемости параллельных алгоритмов численного решения задачи Коши. // Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка, випуск 10(153): – Донецьк, ДонНТУ, 2009. - С. 21-26.

135. Фельдман Л.П., Назарова И.А. Эффективность и масштабируемость параллельных методов численного решения задачи Коши // Материалы XVI международной конференции по вычислительной механике и современным прикладным программным системам (ВМСППС-2009), Алушта, Крым, 25-31 мая 2009г. – М.: Вузовская книга, 2009. – С. 723–724.

136. Кейс О.С., Назарова І.А. Біліотека параллельного чисельного аналізу: підсистема інтегрування лінійної задачі Коші // Матеріали V всеукраїнської науково-технічної конференції студентів, аспірантів та молодих вчених (КМІТ-2009), 12-15 травня 2009р. – Донецьк: ДонНТУ, 2009. – С. 344–345.

137. Душинська Н.О., Назарова І.А., Фельдман Л.П. Паралельна реалізація неявних блокових однокрокових методів чисельного розв'язання жорсткої задачі Коші // Матеріали V всеукраїнської науково-технічної конференції студентів, аспірантів та молодих вчених (КМІТ-2009), 12-15 травня 2009р. – Донецьк: ДонНТУ, 2009. – с. 346–347.

138. Шматько О.Є., Назарова І.А., Фельдман Л.П. Бібліотека параллельного чисельного аналізу: масштабовані алгоритми блокового матричного добутку // Матеріали V всеукраїнської науково-технічної конференції студентів, аспірантів та молодих вчених (КМІТ-2009), 12-15 травня 2009р., – Донецьк: ДонНТУ, 2009. – с. 368–370.

139. Назарова І.А. Паралельні неявні блокові методи чисельного розв'язання жорстких динамічних задач із зосередженими параметрами // Научно-теоретический журнал ИПИИ НАН Украины «Искусственный интеллект», №3, 2009. – Донецк: ИПИИ, 2009. – С.404-412.

140. Назарова І.А. Чисельне моделювання жорстких динамічних задач на базі неявних блокових методів із контролем локальної похибки // Материалы третей международной научно-технической конференции «Моделирование и компьютерная графика». – Донецк: ДонНТУ. – 2009. – С. 209-213. (7-8 октября 2009г.)

141. Иванов А.В., Назарова И.А., Фельдман Л.П. Экстраполяционные одношаговые параллельные методы решения ОДУ (ІУС та КМ-2010) // Матеріали 1 всеукраїнської науково-технічної конференції студентів, аспірантів та молодих вчених.- 19-21 травня 2010р., Донецьк, ДонНТу.-2010.- С. 202-206.

142. Назарова И.А. Оценка масштабируемости параллельного решения СОДУ // Збірник наукових праць. Матеріали міжнародної наукової конференції «Моделювання -2010». Київ. 2010. т.2.-С. 282-290.

143. Фельдман Л.П., Назарова И.А. Параллельные алгоритмы экстраполяционных методов решения задачи Коши для компьютеров с распределенной памятью // Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка, випуск 11 (164): – Донецьк, ДонНТУ, 2010. - С. 7-13.

144. Назарова І.А., Фельдман Л.П. Масштабованість паралельного розв'язання систем звичайних диференційних рівнянь для мультикомп'ютерів із розподіленою пам'яттю // Материалы международной научно-технической конференции «Искусственный интеллект. Интеллектуальные системы: ИИ-2010».– Таганрог-Донецк-Минск: Изд-во ИПИИ, т.1, 2010. – С. 159-163.

145. Назарова И.А. Экстраполяционные блочные одношаговые численные методы решения жестких задач Коши // Научно-теоретический журнал ИПИИ НАН Украины «Искусственный интеллект», №3 , 2010. – Донецк: ИПИИ, 2010. – С.116-126 .

146. Feldman L.P., Nazarova I.A., Dmitrieva O.A., Mikhaylova T.V. Embedded block parallel methods for initial Cauchy problem numerical solution // Proceedings of Donetsk National Technical University. №1, 2010, pp. 12-17.

147. Паралельні однокрокові методи чисельного розв’язання задачі Коші: монографія / Л.П. Фельдман, І.А. Назарова. – Донецьк: «ДВНЗ» ДонНТУ, 2011. – 185 с.: іл.

148. Фельдман Л.П., Назарова И.А. Параллельные экспоненциальные методы решения задачи Коши с оценкой локальной апостериорной погрешности // Наукові праці Донецького національного технічного університету. Серія: Інформатика,

кібернетика та обчислювальна техніка, випуск 13 (185): – Донецьк, ДонНТУ, 2011. - С. 7-12.

149. Фельдман Л.П., Назарова И.А. Оценка масштабируемости параллельных вычислений для одношаговых многоточечных методов решения задачи Коши // Материалы XVII международной конференции по вычислительной механике и современным прикладным программным системам (ВМСППС-2011), Алушта, Крым, 25-31 мая 2011г. – М.: Вузовская книга, 2011. – С. 261–263.

150. Фельдман Л.П., Назарова И.А., Михайлова Т.В. Параллельные вложенные блочные неявные методы решения задачи Коши // Материалы международной научно-технической конференции “Искусственный интеллект. Интеллектуальные системы: ИИ-2011. – Донецк: ІПШІ «Наука і освіта», т.1, 2011. – С. 57–61.

151. Фельдман Л.П., Назарова И.А., Кожухов А.Е. Особенности реализации блочных алгоритмов матричного умножения для кластерных систем MIMD-архитектуры // Материалы VII Международной конференции по неравновесным процессам в соплах и струях (NPNJ'2012) .– М.: Вузовская книга, 2012. – С. 521–523.

152. Фельдман Л.П., Назарова И.А. Применение графовых моделей при разработке параллельных алгоритмов решения нелинейной задачи Коши // Наукові праці ДонНТУ. Серія: ІКОТ, вип. 15(203):–Донецьк, ДонНТУ, 2012. - С. 216-227.

153. Кулаков В.В., Назарова И.А., Фельдман Л.П. Экстраполяционные блочные одношаговые методы численного высокоточного решения задачи Коши // Сборник трудов международной научно-технической конференции студентов, аспирантов и молодых ученых –18-19 сентября 2012 года Донецк, ДонНТУ. – 2012. в 2 томах , Т. 2. – С. 68-73.

154. Никишин Р.Ю., Назарова И.А., Фельдман Л.П. Параллельные неявные методы решения жестких задач Коши для систем обыкновенных дифференциальных уравнений // Сборник трудов международной научно-технической конференции студентов, аспирантов и молодых ученых –18-19 сентября 2012 года Донецк, ДонНТУ. – 2012. в 2 томах , Т. 2. – С. 91-96.