УДК 004.932.4

Л.П. Фельдман, д.т.н., проф.,
И.А. Назарова, к.т.н., доц.
Донецкий национальный технический университет, г. Донецк, Украина
feldman@pmi.dgtu.donetsk.ua, ianazard@gmail.com

Применение графовых моделей при разработке параллельных алгоритмов решения нелинейной задачи Коши

Предложена иерархическая декомпозиционная методика распараллеливания с использованием математического аппарата графов влияния. Продемонстрировано применение разработанной методики для получения параллельных алгоритмов решения нелинейной задачи Коши для систем обыкновенных дифференциальных уравнений.

Ключевые слова: параллельные алгоритмы и методы, методика распараллеливания, задача Коши для систем обыкновенных дифференциальных уравнений, графы влияния, макрооперация

Введение

Современные исследования и вычислительные эксперименты в области высокопроизводительных вычислений показывают, что практически все новые параллельные методы, даже те из них, которые эффективны в теоретическом отношении, на практике не конкурентно способны. Поэтому на текущий момент надежным источником создания параллельных методов является подходящая реструктуризация проверенных временем последовательных алгоритмов [1]. Формально реструктуризация сводится к построению и анализу информационных графов алгоритмов с целью явно указать обнаруженный в алгоритмах потенциальный, скрытый параллелизм.

Целью исследования является разработка и исследование эффективности методики распараллеливания последовательных методов, основанной на применении формальных моделей математического аппарата теории графов.

Задачей исследования является разработка параллельных методов решения задачи Коши для систем обыкновенных дифференциальных уравнений большой размерности с оценкой локальной погрешности.

Графы влияния и иерархическая декомпозиционная методика

Развитым и общепризнанным математическим аппаратом разработки параллельных алгоритмов являются графовые модели: графы влияния, зависимостей и так далее, то есть некоторые информационные графы алгоритмов [1].

Каждая исполняемая операция алгоритма представляет собой вершину графа. На множество вершин естественным образом переносится лексикографический порядок. Пару вершин можно связывать друг с другом в том случае, если между ними имеется некоторая зависимость.

По определению [1], операция x влияет на операцию y, если изменением значения переменной, которую вычисляет операция x можно изменить значение переменной, которую вычисляет операция y. По указанному правилу строится ориентированный граф, причем операции алгоритма есть вершины орграфа. Дугами соединяют каждую из пар вершин, в которой одна из соответствующих им операций влияет на другую, направление дуги совпадает с направлением влияния. Полученный орграф называется графом влияния алгоритма. Граф влияния содержит много избыточной информации, и с ним неудобно работать. Поэтому по исходному графу строится минимальный остовный подграф, обладающий следующим свойством: если какая-либо операция x оказывает влияние на операцию y, то в подграфе должен существовать хотя бы один путь, связывающий x с y. Такой граф называют минимальным графом влияния. В силу транзитивности отношения влияния граф влияния оказывается транзитивным замыканием минимального графа.

Характерно, что для операций, которые можно выполнять независимо друг от друга, никакие две соответствующие этим операциям вершины не должны быть связаны дугами графа влияния. Тогда задача распараллеливания сводится к решению задачи отыскания максимального независимого множества вершин.

Однако для реальных задач применение описанной модели сопряжено с большими трудностями. Для прямого описания всех вершин и дуг соответствующего орграфа необходимы большие затраты компьютерной памяти, а для обработки или анализа используются графовые алгоритмы, которые практически все имеют

высокую вычислительную сложность, чаще всего экспоненциальную. Поэтому для разработки параллельных алгоритмов задач практического уровня сложности предлагается использовать многоэтапный технологический процесс, начинающийся анализом задачи, выбором модели вычислений, декомпозицией задачи на независимые параллельные процессы и заканчивающийся вопросами исследования эффективности и организации вычислительного эксперимента. Такая технология распараллеливания получила название – иерархической декомпозиционной методики [2-3].

Общая схема поэтапной иерархической декомпозиционной методики построения параллельного алгоритма представлена на рисунке 1 и реализует следующие этапы:

– выделение независимых вычислительных действий;

– определение информационных зависимостей;

– оценка возможности эффективного применения получаемой вычислительной схемы для целевых компьютерных систем.

Первоначально вычислительная задача разбивается на подзадачи, анализируются информационные взаимосвязи между ними, определяется множество макроопераций, оценивается эффективность потенциального и реального параллелизма, масштабируемость параллельной системы. В случае неудовлетворительного результата на любом из этапов схема разбиения изменяется, и весь процесс повторяется сначала 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 могут быть использованы для построения параллельного алгоритма, как на верхнем уровне для анализа взаимодействия подзадач, так и на нижнем уровне для распараллеливания отдельных макроопераций, соответствующих определенным подзадачам [1-2].

Введение макроопераций приводит к болем 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. Процесс введения макроопераций может осуществляться поэтапно с последовательно возрастающим уровнем детализации используемых операций.

Существует два принципиально разных способа разбиения процесса вычислений на независимые части: функциональное разбиение и разбиение по данным.

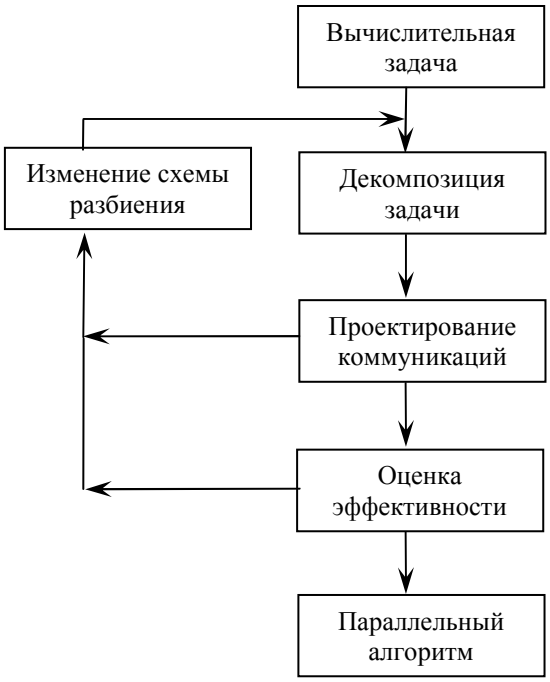


Рисунок 1 – Поэтапная иерархическая декомпозиционная методика построения параллельного алгоритма

Функциональное разбиение, в первую очередь, требует разделения процесса вычислений на отдельные процедуры и только после этого разбиения данных в зависимости от выделенных независимых процедур обработки. Второй способ разбиения данных подразумевает разделение обрабатываемых данных на равные блоки небольшого размера, чаще всего на основе геометрического разбиения области решения задачи, а затем, в зависимости от способа разбиения данных, декомпозицию процедур обработки распределенной информации. Вне зависимости от избранного способа разбиения процесса на части, следующим этапом построения параллельного алгоритма, является определение информационных потоков.

Различают следующие типы информационного взаимодействия параллельно выполняемых процессов:

– локальное и глобальное взаимодействие по количеству взаимодействующих процессов;

– структурное и взаимодействие "нетипового" вида по схемам коммуникационных топологий.

Организация взаимодействия процесса, приводящая к формированию вполне определенных схем коммуникации (кольцо, тор,

гиперкуб) – есть структурное взаимодействие. Неструктурное взаимодействие возникает, когда схема выполняемых операций передач данных имеет вид графа "нетипового" вида. В зависимости от способа реализации межпроцессорных обменов различают синхронный и асинхронный параллелизм (модель параллельной ВС типа SIMD и типа MIMD). Последними этапами построения параллельного алгоритма являются распределение вычислительной нагрузки по процессорам параллельной компьютерной системы и решение задачи балансировки, являющиеся NP-полной проблемой.

Существует и альтернативный подход для разбиения исходной вычислительной задачи на независимые части – агрегационный. Если декомпозиционный подход – это разбиение "сверху-вниз", то агрегационный – "снизу-вверх". 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. Данный подход чаще всего используется для вычислительных систем с распределенной памятью и моделью передачи сообщений. Однако нет никаких ограничений для применения описанной методики в системах с общей памятью, если для обеспечения информационного взаимодействия используется доступ к разделяемому адресному пространству.

Таким образом, в рамках данного исследования рассматриваются следующие этапы разработки параллельных алгоритмов на основе 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;

4) определение необходимых информационных взаимодействий для выделенных параллельных процессов обработки данных;

5) распределение сформированного набора процессов обработки данных по процессорам (отображение параллельной схемы решения задачи на архитектуру компьютерной вычислительной системы).

Продемонстрируем применение рассмотренной технологии распараллеливания для получения параллельных алгоритмов решения нелинейной задачи Коши для систем обыкновенных дифференциальных уравнений (СОДУ) с встроенными методами оценки локальной апостериорной погрешности: правило Рунге, вложенные формы и технология локальной экстраполяции Ричардсона (ЛЭР).

Распараллеливание решения задачи Коши для СОДУ на базе правила Рунге и вложенных форм

В данном разделе рассматривается построение и анализ эффективности параллельных алгоритмов определения локальной апостериорной погрешности на основе явных одношаговых схем типа Рунге-Кутты (ЯМРК) для численного решения систем обыкновенных дифференциальных уравнений общего вида. Предполагается, что набор процессоров однороден, известен до начала вычислений и не меняется в процессе счета. При этом каждый процессор может выполнить любую арифметическую операцию за один такт, временные затраты, связанные с обращением к запоминающему устройству, отсутствуют.

Распараллеливание явных одношаговых методов, к которым относятся формулы Рунге-Кутты, концентрируется на выполнении одного шага интегрирования. Системный параллелизм будет реализован через распределенные вычисления различных компонент системы, т.е. отдельных компонент шаговых векторов и векторов решений. Параллелизм метода связан с выделением независимых субвычислений внутри одного шага метода и значительно меньше системного.

Численно решается в общем случае нелинейная задача Коши для системы обыкновенных дифференциальных уравнений:

$$\begin{cases} \dfrac{dy_1(x)}{dx} = f_1(x; y_1, y_2, ..., y_m) \\ \dfrac{dy_2(x)}{dx} = f_2(x; y_1, y_2, ..., y_m) \\ \quad ... \\ \dfrac{dy_m(x)}{dx} = f_m(x; y_1, y_2, ..., y_m) \end{cases} \quad (1)$$

с известными начальными условиями:

$$\begin{cases} y_1(x_0) = y_{10} \\ y_2(x_0) = y_{20} \\ \quad ... \\ y_m(x_0) = y_{m0} \end{cases}. \quad (2)$$

Явный s-стадийный метод Рунге-Кутты имеет следующий вид:

$$\begin{cases} \bar{y}_{n+1} = \bar{y}_n + h \cdot \sum_{i=1}^{s} b_i \cdot \bar{k}_i \\ \bar{k}_i = F(x_n + c_i h, \bar{y}_n + h \sum_{j=1}^{i-1} a_{ij} \bar{k}_j), \ i = \overline{1,s} \end{cases}. \quad (3)$$

Введем обозначение:

$$\bar{g}_i = \bar{y}_n + h \cdot \sum_{j=1}^{i-1} a_{ij} \bar{k}_j, i = \overline{1,s}.$$

Тогда, для ЯМРК, с учетом введенных обозначений, получим вычислительную схему:

$$\begin{cases} \bar{y}_{n+1} = \bar{y}_n + h \cdot \sum_{i=1}^{s} b_i \bar{k}_i; \\ \bar{k}_i = F(x_n + c_i h; \bar{g}_i); i = 1,...,s. \end{cases} \quad (4)$$

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 подзадачами, биективно к множеству подзадач определяется множество макроопераций, оценивается эффективность потенциального крупно или среднеблочного 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 [1].

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

$$\begin{cases} \bar{y}_{n+1}^{(1)} = \bar{y}_n + 2h \cdot \sum_{i=1}^{s} b_i \cdot \bar{k}_i^{(1)}(2h), \\ \bar{k}_i^{(1)} = F\left(x_n + c_i \cdot 2h, \bar{y}_n + 2h \sum_{j=1}^{i-1} a_{ij} \bar{k}_j^{(1)} \right), \\ \bar{y}_{n+\frac{1}{2}} = \bar{y}_n + h \cdot \sum_{i=1}^{s} b_i \cdot \bar{k}_i(h), \\ \bar{k}_i = F\left(x_n + c_i h, \bar{y}_n + h \sum_{j=1}^{i-1} a_{ij} \bar{k}_j \right), \\ \bar{y}_{n+1}^{(2)} = \bar{y}_{n+\frac{1}{2}} + h \cdot \sum_{i=1}^{s} b_i \cdot \bar{k}_i^{(2)}(h), \\ \bar{k}_i^{(2)} = F\left(x_{n+\frac{1}{2}} + c_i h, \bar{y}_{n+\frac{1}{2}} + h \sum_{j=1}^{i-1} a_{ij} \bar{k}_j^{(2)} \right), \\ d_n = \left\| \bar{y}^{(2)} - \bar{y}^{(1)} \right\|; \ i = \overline{1,s}. \end{cases} \quad (5)$$

Величина d_n определяет погрешность, вычисляемую на каждом шаге интегрирования, и применяется для выбора стратегии управления длиной шага интегрирования. Последовательный алгоритм, описываемый вычислительной схемой (5), можно представить, как решение трех подзадач:

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

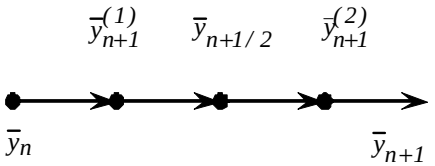
2) вычисление приближенного решения в промежуточной точке $x_{n+1/2} = x_n + h$ с шагом h: $\bar{y}_{n+1/2}(h)$;

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.

Рассмотрим три различные макрооперационные вычислительные схемы исследуемого алгоритма. Как видно из рисунка 2, первая подзадача может выполняться независимо от двух других, третья обязательно должна следовать за второй. Процесс решения всех трех подзадач может быть выполнен различными способами, причем каждый из них имеет свои преимущества.

Первый вариант макрооперационной вычислительной схемы конструируемого параллельного алгоритма представлен на рисунке 2. Здесь все три подзадачи выполняются последовательно, данные между процессорами распределены равномерно. К исходным данным относятся коэффициенты таблицы Батчера, определяющие явную схему, и решение на предыдущем шаге интегрирования $\bar{y}_n$.



(2.5)

Рисунок 2 – Граф влияния макрооперационной вычислительной схемы № 1 алгоритма ЯМРК с правилом Рунге

Второй вариант представлен на рисунке 3, здесь первая и вторая подзадачи выполняются параллельно, причем 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.

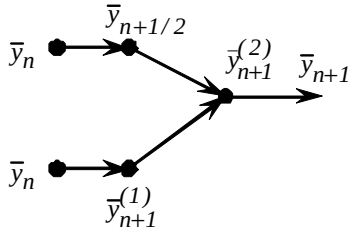


Рисунок 3 – Граф влияния макрооперационной вычислительной схемы № 2 алгоритма ЯМРК с правилом Рунге

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 макрооперационной вычислительной схемы (рис. 4) предполагает параллельное выполнение с одной стороны первой подзадачи на меньшем числе процессоров, а с другой стороны параллельно второй и третьей – на большем числе процессоров. При этом все множество процессоров делится на две пропорциональные части: $p_1 = 2p_2; p_1 + p_2 = p$, в соответствии с функциональным разбиением вычислительной схемы.

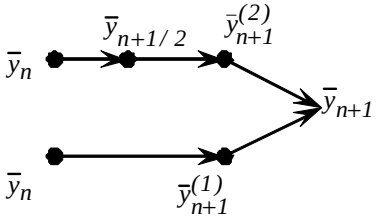


Рисунок 4 – Граф влияния макрооперационной вычислительной схемы № 3 алгоритма ЯМРК с правилом Рунге

На этом уровне детализации нет никаких оснований предпочесть одну из трех вычислительных схем алгоритма. Последовательность выполнения трех макроопераций в первой схеме компенсируется равномерной загрузкой и низкой интенсивностью взаимодействия процессов. С другой стороны, вторая и третья вычислительные схемы алгоритма имеют максимальную степень параллелизма равную – $Dop = p_{max} = const = 2m$, в то время как первая схема – $Dop = m$, то есть в два раза меньше. Перейдем 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 (5) состоит в вычислении множества стадийных коэффициентов $\bar{k}_i = (k_{i1}, k_{i2}, ..., k_{im})$, $i = \overline{1,s}$ и приближенного решения $\bar{y}_{n+1}(h)$.

Для явных схем вычисление величин $\bar{k}_i$ есть сугубо последовательный процесс. Воспользуемся так называемым системным параллелизмом, когда вычисление одного вектора коэффициентов схемы $\bar{k}_i$ распределяется на несколько независимых процессоров.

Тогда, последовательный алгоритм одного шага интегрирования по явной вычислительной схеме может быть представлен, как решение s подзадач вычисления стадийных коэффициентов и одной подзадачи вычисления приближенного решения. Поэтому в качестве первой макрооперации этого уровня детализации выделяется макрооперация – вычисление i-того стадийного коэффициента, а в качестве второй – вычисление численной аппроксимации решения в некоторой точке. Введение макроопераций, осуществляется поэтапно с последовательно возрастающим уровнем детализации используемых операций [4-5].

На рисунке 5 приведен граф влияния первой макрооперации: вычисления вектора значений i-того стадийного коэффициента при $m = p$. В этом случае каждый процессор вычисляет одну компоненту вспомогательного вектора $\bar{g}_i$, затем производится общий обмен между процессорами по типу "все-всем" для вычисления очередного коэффициента $\bar{k}_i$.

Вторая макрооперация определяет вычисление приближенного решения на некотором шаге интегрирования, и задает вычисления по формуле:

$$\bar{y}_{n+1} = \bar{y}_n + h_{n+1} \cdot \sum_{i=1}^{s} b_i \cdot \bar{k}_i(h_{n+1}),$$

где вне зависимости от конкретики: $\bar{y}_n$ – аппроксимация решения на предыдущем шаге интегрирования; $\bar{y}_{n+1}$ – значение на последующем шаге; $\bar{k}_i$ – вычисляемая матрица стадийных коэффициентов:

$$K = \left\| k_{ij} \right\| = \begin{matrix} \bar{k}_1: \\ \bar{k}_2: \\ \dots \\ \bar{k}_s: \end{matrix} \begin{pmatrix} k_{11} & k_{12} & \dots & k_{1m} \\ k_{21} & k_{22} & \dots & k_{2m} \\ \dots & \dots & \dots & \dots \\ k_{s1} & k_{s2} & \dots & k_{sm} \end{pmatrix},$$

$$i = \overline{1,s}; \quad j = \overline{1,m},$$

где i – номер вектора $\bar{k}_i = (k_{i1}, k_{i2}, ..., k_{im})$ схемы s-стадийного явного метода, j – номер компоненты вектора.

Вторая макрооперация определяет вычисление приближенного решения на некотором шаге интегрирования и при имеющемся разбиении данных представлена графом влияния, изображенным на рис. 6. Обе макрооперации являются базовыми для всех явных схем и будут использованы в дальнейшем при рассмотрении альтернативных способов определения локальной погрешности.

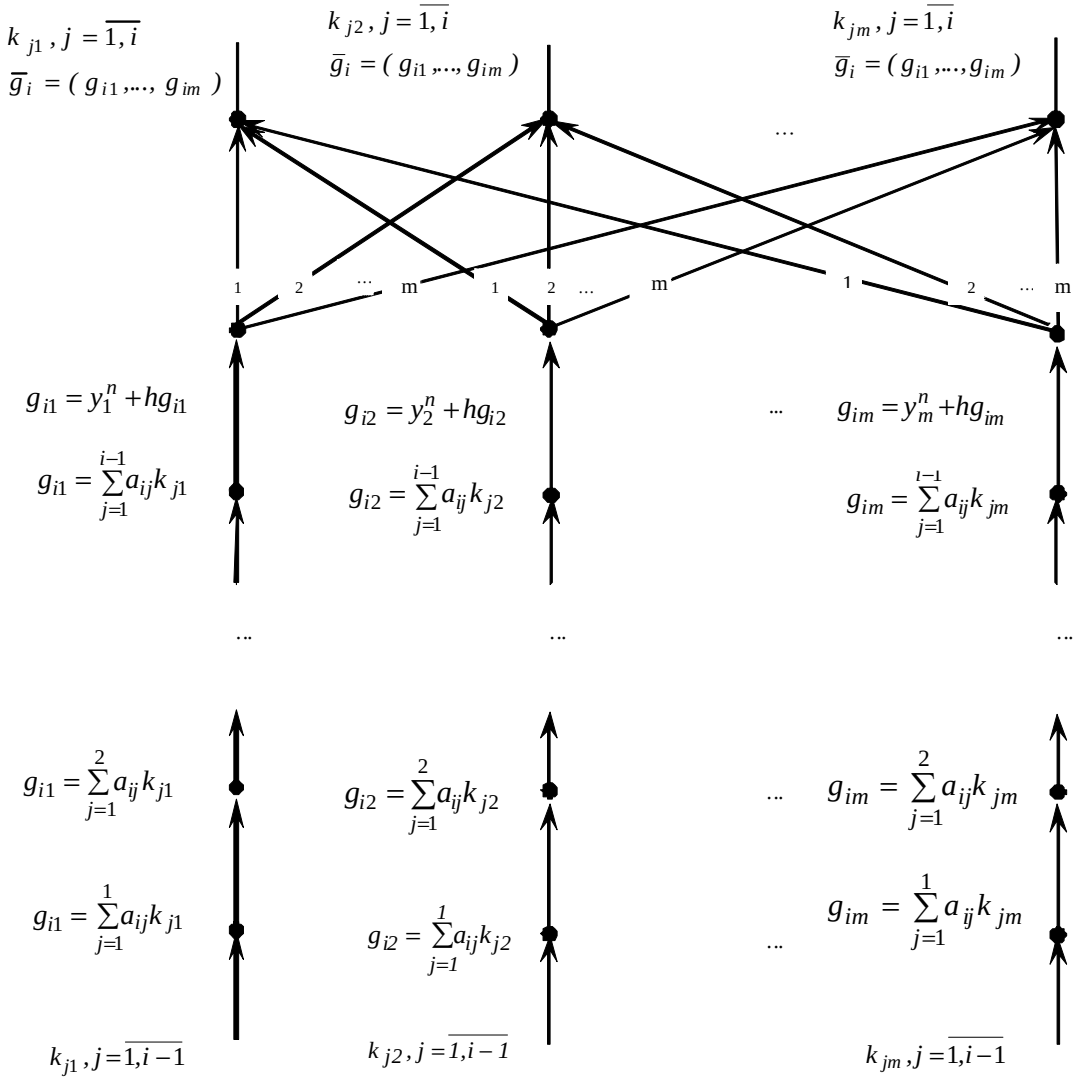


Рисунок 5 – Граф влияния для вычисления i-того стадийного коэффициента ЯМРК для СОДУ $\bar{k}_i = (k_{i1}, k_{i2}, ..., k_{im})$, $m = p$

После определения множества макроопераций и исследования их свойств, определяются оценки потенциального параллелизма схем. Чтобы задача построения параллельных алгоритмов стала математически корректной, необходимо сделать предположения относительно свойств параллельной вычислительной системы. Применим концепцию неограниченного параллелизма.

В качестве идеализированной модели высокопроизводительной параллельной вычислительной системы будем использовать модель мультикомпьютер с распределенной памятью и коммутационной схемой произвольной топологии.

Каждый компьютер в системе выполняет свою программу, программы могут иметь доступ к локальной памяти и посылать или принимать сообщения через сеть. В сети время передачи сообщения зависит от расстояния между узлами и характеристик сетевого трафика, а также от объема пересылаемых данных.

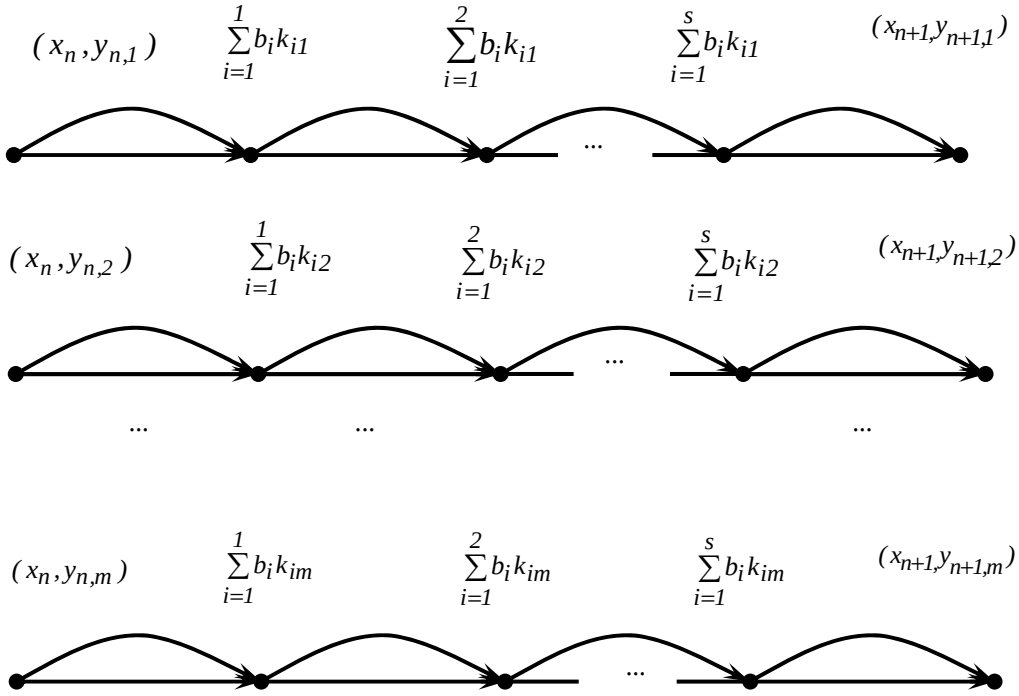


Рисунок 6 – Граф влияния второй макрооперации: вычисление численной аппроксимации решения в точке (x_{n+1}, y_{n+1})

Предложенная модель проста и позволяет исследовать возможности параллельного алгоритма до построения отображения на реальную вычислительную структуру определенной параллельной архитектуры, то есть до оценки реального параллелизма алгоритма.

Параллельный алгоритм первой макрооперационной схемы на мультикомпьютере из p процессоров использует выполнение одного шага интегрирования. Для системы ОДУ, состоящей из m уравнений, на одном временном шаге m_1 компонент коэффициентов $\bar{k}_i$ могут быть выполнены параллельно, причем: 1) $m_1 = m$ при $m = p$; 2) $m_1 = \lceil m/p \rceil$ при $m > p$. Затем вычисляются соответствующие компоненты вектора решения, при этом разбиение данных по процессорам остается без изменения. Определение векторов решений и переход к следующему шагу не требуют дополнительных передач данных.

При выполнении параллельного алгоритм решения СОДУ на мультикомпьютере из p процессоров для второй макрооперационной схемы поле процессоров разбивается на две одинаковые группы с числом процессоров равным $p_1 = \lceil p/2 \rceil$. Каждая группа вычисляет параллельно $\lceil m/p_1 \rceil$ компонент векторов решения для аппроксимаций в x_{n+1} точке с удвоенным шагом и в промежуточной точке, $\lceil m/p \rceil$ – для аппроксимации в x_{n+1} с одинарным шагом. Алгоритм решения той же задачи Коши для третьей макрооперационной схемы использует разбиение процессоров пропорционально объему вычислений. Базовая макрооперация описана графом влияния на рисунке 5. Первая группа процессоров вычисляет две аппроксимации решения: $\bar{y}_{n+1/2}$ и $\bar{y}^{(2)}_{n+1}(h)$ и имеет размер процессорного поля $p_1 = 2p_2$ в два раза больше, чем вторая группа.

Вторая группа процессоров вычисляет одну аппроксимацию решения с удвоенным шагом: $\bar{y}^{(1)}_{n+1}(2h)$. 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.

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:

$$\begin{cases} \bar{y}_{n+1} = \bar{y}_n + h \cdot \sum_{i=1}^{s} b_i \cdot \bar{k}_i, \\ \hat{\bar{y}}_{n+1} = \bar{y}_n + h \cdot \sum_{i=1}^{s} \hat{b}_i \cdot \bar{k}_i. \end{cases} \quad (6)$$

Если основу вложенных форм составляют явные разностные схемы, то стадийные коэффициенты должны определяются, как:

$$\begin{cases} \bar{k}_i = F(x_n + c_i h; \bar{g}_i), \\ \bar{g}_i = \bar{y}_n + h \sum_{j=1}^{i-1} a_{ij} \cdot \bar{k}_j, i = \overline{1,s'}, \end{cases} \quad (7)$$

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

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 (6)-(7) 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 практически совпадает с графом влияния для правила Рунге, максимальная степень параллелизма совпадает с размерностью СОДУ и равна $Dop = m$. В свою очередь макрооперации 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 рисунке 5-6.

Рассмотрим параллельный алгоритм решения задачи Коши по методу вложенных форм для мультикомпьютера из p процессоров. Пусть 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.

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.

Параллельная реализация технологии локальной экстраполяции 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 [6-7]:

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 параллельных алгоритмов решения нелинейной задачи Коши для СОДУ по технологии Ричардсона также базируется на декомпозиционной методике. После того, как определены все составляющие технологии экстраполяции, процесс решения можно разбить на две последовательно выполняемые подзадачи вычисление k аппроксимаций в точке x_{n+1} с разными шагами интегрирования и 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 8, все аппроксимации решения вычисляются последовательно одновременно для каждой размерности системы.

Максимальная степень параллелизма для этого варианта алгоритма технологии экстраполяции Ричардсона равна размерности системы обыкновенных дифференциальных уравнений и составляет: $Dop = m$.

По второй макрооперационной схеме параллельно вычисляется k независимых аппроксимаций решения в точке $x_n + H$, граф влияния для такой схемы приведен на рисунке 9.

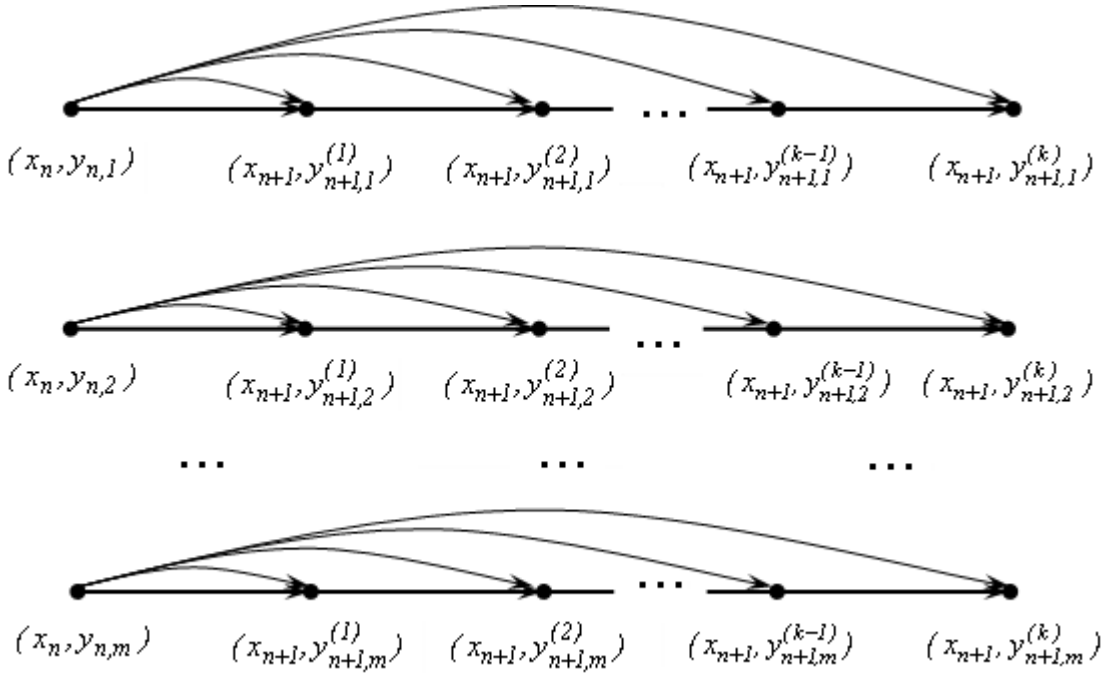


Рисунок 8 – Граф влияния макрооперационной схемы для вычисления аппроксимаций решения СОДУ (вариант №1)

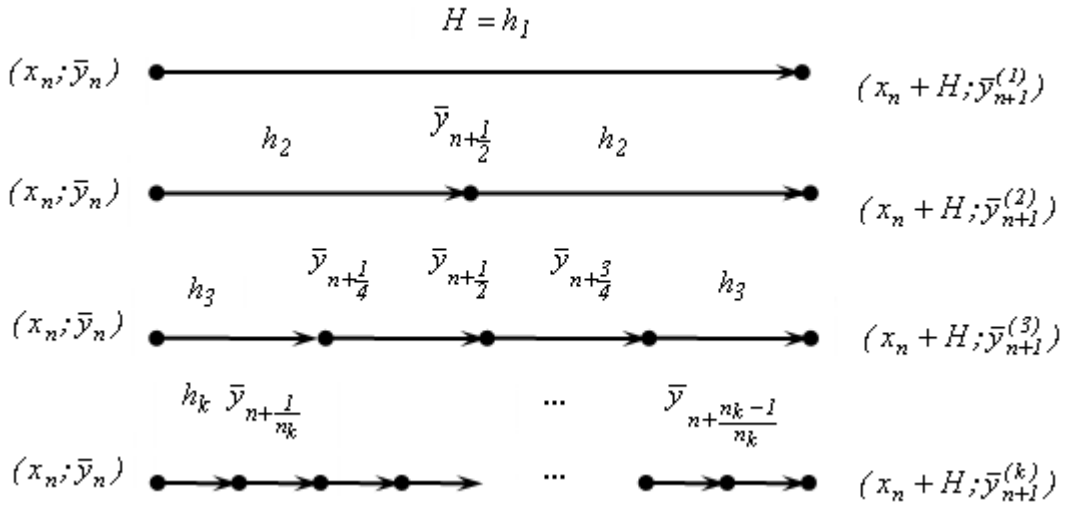


Рисунок 9 – Граф влияния макрооперационной схемы для вычисления аппроксимаций решения СОДУ (вариант № 2)

Второй вариант алгоритма имеет большую степень параллелизма, равную $Dop = m \cdot k$. 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. Распараллеливание может быть осуществлено за счет независимого выполнения вычислений по формуле полиномиальной экстраполяции для 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 10.

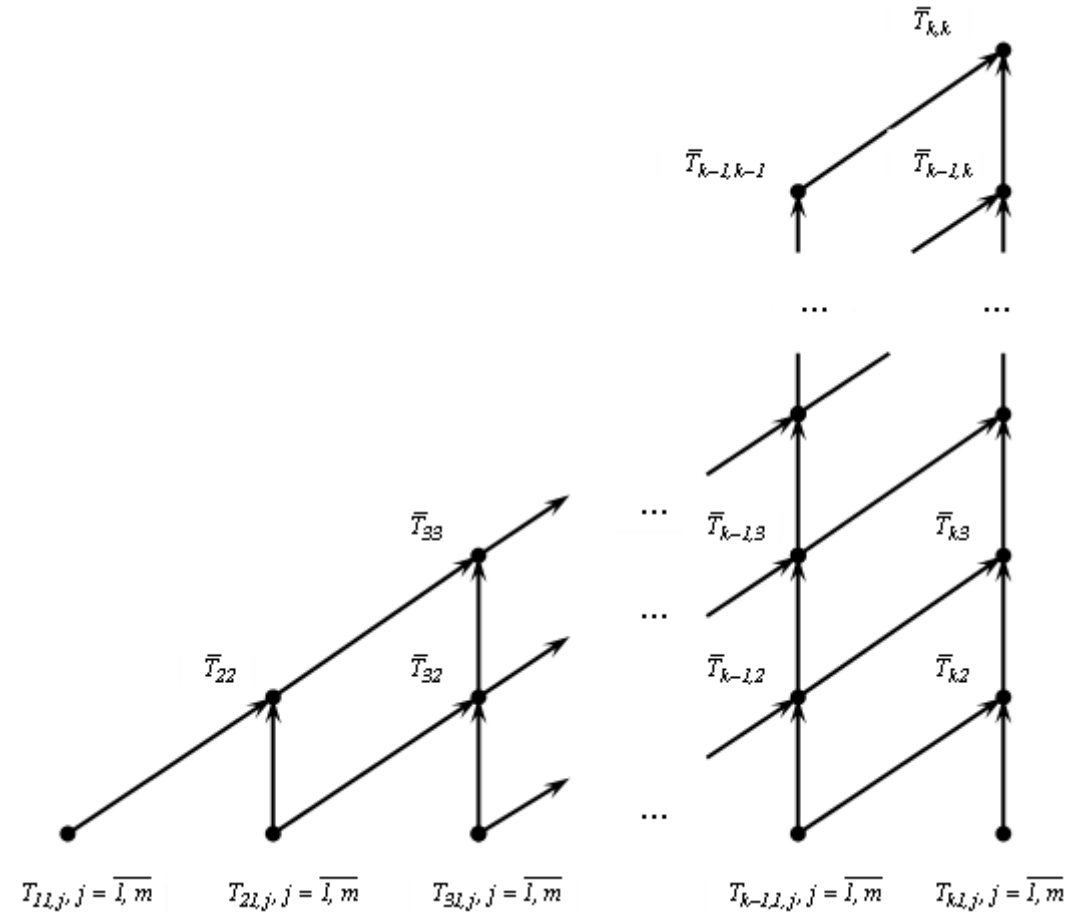


Рисунок 10 – Граф влияния метода построения экстраполяционной таблицы

На рассмотренном этапе детализации дан подробный анализ эффективности каждой из вычислительных схем. Анализ внутреннего параллелизма метода повторяет рассуждения, приведенные в разделе для правила Рунге, с тем различием, что теперь у нас k последовательно вычисляемых аппроксимаций по ЯМРК.

Таким образом, в статье приведены результаты разработки и анализа эффективности параллельных методов решения задачи Коши для СОДУ на основе явных численных схем с использованием поэтапной декомпозиции и аппарата графов влияния.

Заключение

В статье приведена поэтапная иерархическая методика распараллеливания, базирующаяся на сочетании трех составляющих: графов влияния, аппарата макроопераций и декомпозиционной технологии. Продемонстрировано применение данной методики для разработки параллельных алгоритмов решения нелинейной задачи Коши для СОДУ с встроенными методами оценки локальной апостериорной погрешности. Предложенная иерархическая методика

распараллеливания, позволяет сократить время разработки эффективного параллельного алгоритма решения поставленной задачи и провести анализ его внутренней структуры до реального отображения на параллельную архитектуру [4-6].

Научная новизна работы заключается в разработке новой модифицированной методики распараллеливания, позволяющей сократить время разработки эффективного параллельного алгоритма решения поставленной задачи и провести анализ его внутренней структуры до реального отображения на параллельную архитектуру [4-7].

Практическая значимость предложенной работы заключается в разработке эффективных параллельных численных методов решения нелинейной задачи Коши, сокращающих время моделирования динамических задач большой размерности и допускающих отображение на параллельные архитектуры различных топологий.

К перспективным направлениям дальнейших исследований следует отнести разработку новых эффективных и масштабируемых параллельных методов решения реальных динамических задач, а также наработку теоретических основ для создания автоматизированного и/или автоматического метода распараллеливания.

Список использованной литературы

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608с.

2. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем.– Н. Новгород: ННГУ, 2001. – 122с.

3. Гергель В.П. Теория и практика параллельных вычислений. – Москва: Бином. Лаборатория знаний, 2007. – 423с.

4. 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.

5. 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.

6. Фельдман Л.П., Назарова І.А. Паралельні однокрокові методи чисельного розв'язання задачі Коші: монографія / Л.П. Фельдман, І.А. Назарова. – Донецьк: «ДВНЗ» ДонНТУ, 2011. – 185 с.: іл.

7. 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.

Надійшла до редколегії 30.01.2012

Л.П. ФЕЛЬДМАН, І.А. НАЗАРОВА
Донецький національний технічний університет

Застосування графових моделей при розробці паралельних алгоритмів розв'язання нелінійної задачі Коші

Запропоновано ієрархічну декомпозиційну методику розпаралелювання з використанням математичного апарату графів впливу. Продемонстровано застосування розробленої методики для отримання паралельних алгоритмів рішення нелінійної задачі Коші для систем звичайних диференціальних рівнянь.

Ключові слова: паралельні алгоритми та методи, задача Коші для систем звичайних диференційних рівнянь, графи впливу, макрооперація

L.P. FELDMAN, I.A. NAZAROVA
Donetsk National Technical University

The use of graph models in the development of parallel algorithms for solving the nonlinear Cauchy's problem

A hierarchical decompositional technique of parallelization using the mathematical apparatus of graph of influence is proposed. Application of the developed technique for parallel algorithms for solving the Cauchy's problem for nonlinear systems of ordinary differential equations is demonstrated.

Key words: parallel algorithms and methods, systems of ordinary differential equations, graph of influence, macrooperations