УДК 519.622.2

А.Б. Корчак[1,2], *А.А. Евдокимов*[2]

[1] Институт проблем проектирования в микроэлектронике РАН

[2] Московский физико-технический институт (Национальный исследовательский университет)

Метод параллельного расчёта расщеплённых систем дифференциальных уравнений с кратными шагами

Разрабатывается метод расчёта гетерогенных моделей с разными шагами по времени, основанный на семействе алгоритмов параллельного решения систем обыкновенных дифференциальных уравнений большой размерности. Применительно к слабосвязанным системам обыкновенных дифференциальных уравнений исследуется погрешность таких алгоритмов и их преимущество в затратах времени по сравнению с точными методами решения.

Ключевые слова: декомпозиция систем обыкновенных дифференциальных уравнений, параллельные вычислительные процессы, алгоритм ускоренного расчёта систем ОДУ.

В данной работе предлагается и исследуется подход к алгоритмам численного решения больших систем дифференциальных уравнений. При решении многих прикладных задач приходится сталкиваться с системами обыкновенных дифференциальных уравнений (ОДУ) большой размерности, например, при моделировании микросхем или иных многокомпонентных систем. Число уравнений может достигать 10^9. Численное решение таких систем напрямую представляется затруднительным даже для современных вычислительных мощностей.

Основной сложностью решения больших систем дифференциальных уравнений является существенное различие скорости протекания описываемых ими процессов. Рассматриваемые в работе системы ОДУ с большим разбросом скоростей могут быть как жёсткими системами (с существенным различием вещественных частей собственных чисел матрицы Якоби), так и колебательными системами с существенным разбросом частот (мнимых частей собственных чисел). Учёт всех быстрых процессов на численном уровне заставляет использовать либо очень малый шаг интегрирования, либо применять специальные классы численных методов, дающие устойчивые разностные задачи при умеренном шаге. В частности, для жёстких систем применяются жёстко-устойчивые и другие неявные методы, которые сложнее явных в реализации, а также могут приводить к неверному (хотя устойчивому) решению. Уменьшение же шага интегрирования приводит к возрастанию числа требуемых шагов и, как следствие, к неприемлемому росту временных затрат на решение больших систем. Обычно число арифметических операций, требуемых для выполнения одной итерации численного решения, нелинейно зависит от размерности задачи.

Существующие подходы к решению больших систем можно разделить на два основных класса. Одни подходы сосредотачиваются на алгоритмах распараллеливания программного кода численных методов (причём безо всякого изменения самих методов) [1]. Однако результаты решения систем ОДУ, получаемые при таком распараллеливании на кластерных системах, не являются впечатляющими (по крайней мере, по сравнению с параллельным решением уравнений в частных производных). Это обусловлено тем, что в типичной системе ОДУ число обменов значениями переменных между процессорами пропорционально числу переменных (в то время как, например, при решении двумерных задач с частными производными число обменов пропорционально квадратному корню из числа переменных). При этом не учитываются какие-либо особенности задачи (например, слабая связанность подсистем, см. ниже), за счёт которых зачастую в вычислительном смысле большие системы ОДУ могут быть сближены с системами уравнений в частных производных.

Второй класс подходов, характерный для некоторых прикладных областей (например, для схемотехники [2]), делает акцент на сокращение вычислительных затрат за счёт специальных эвристических приемов (обычно вытекающих из «физики задачи»), а точности решения уделяется меньше внимания. Это оправдано тем, что при решении прикладных задач погрешность входных данных (обусловленная, в частности, неточностью измерений тех или иных параметров физической системы) обычно на 1–2 порядка превышает погрешность методов и погрешности округления. Увеличение скорости работы алгоритмов расчёта может быть осуществлено за счёт снижения точности численных методов без понижения точности полученных результатов. Данная работа также использует этот факт (причём без привязки к определённой прикладной области), однако делает акцент на исследование точности решения.

Следует также отметить современные исследования по многоскоростному (multi-rate) решению систем ОДУ [3, 4], которые занимают промежуточное положение между двумя вышеописанными классами подходов. В них предлагают-

ся различные формулы для расчёта нескольких групп переменных, соответствующих различным шагам интегрирования (кратным друг другу), и экономия вычислительных затрат достигается за счёт меньшего числа операций, требуемых для расчёта «медленных» переменных (соответствующих большим шагам) в промежуточные моменты времени. Как правило, многоскоростные методы используют идею интерполяции (экстраполяции) значений «медленных» переменных в те моменты времени, когда вычисляются значения «быстрых» переменных. Однако здесь речь не идёт о настоящем расщеплении на подсистемы (при котором медленная переменная просто не существует в тот момент времени, когда вычисляется быстрая). Как преимущество следует отметить достаточно подробное рассмотрение исследователями подобных методов вопроса о точности результатов (оказывается, что интерполяция с нужным порядком точности позволяет даже сохранить высокий порядок исходного численного метода) [5, 6]. Как недостаток экономия вычислительных ресурсов у них получается не слишком существенной (в то время как для решения больших систем нужна экономия в десятки и сотни раз). При этом вопрос о распараллеливании многоскоростных методов вообще не рассматривается, поскольку с алгоритмической точки зрения они не отличаются от обычных методов расчёта ОДУ (а они, как было указано выше, плохо поддаются распараллеливанию).

Снижение размерности задачи за счёт декомпозиции системы на несколько подсистем является одним из самых эффективных способов увеличения скорости расчёта. Расщепление системы на несколько независимых подсистем (точная декомпозиция) позволяет проводить параллельные расчёты с применением классических методов, но этот частный случай крайне редко встречается на практике. Неточная декомпозиция (расщепление системы на зависимые подсистемы) возможна всегда, однако с точки зрения распараллеливания она приводит к необходимости обмена значениями переменных между процессорами. Чем меньшим количеством переменных необходимо обмениваться, тем больше польза от распараллеливания. Однако декомпозиция на подсистемы, отталкивающаяся от этого очевидного критерия, на практике часто приводит к очень плохой балансировке загрузки процессоров (расчёт шага одной подсистемы может происходить на порядок быстрее, чем другой, и соответствующий ему процессор большую часть времени простаивает в ожидании завершения шага другим процессором).

Таким образом, распараллеливание систем ОДУ, расщепленных на подсистемы, наталкивается на проблему чрезмерного потока данных между процессорами и на проблему неравномерности загрузки процессоров, и они обе снижают эффект от распараллеливания. Эти две проблемы невозможно решить одновременно при условии сохранения алгоритмом точной постановки разностной задачи (результата применения некоторого численного метода к исходной дифференциальной задаче). Поэтому возникает идея решить эти проблемы за счёт некоторой модификации исходной задачи (в том числе за счёт применения в разных подсистемах разных шагов). Соответствующее снижение вычислительных затрат достигается ценой некоторого роста погрешности аппроксимации исходной дифференциальной задачи. Однако применение данной идеи к выделенному классу задач — слабосвязанным системам ОДУ (см. ниже) — вносит в итоговую погрешность решения лишь небольшой вклад. Более того, благодаря предлагаемой модификации существенно уменьшается вклад погрешности, связанной с конечной разрядностью машинного числа (что связано с уменьшением числа арифметических операций).

Кроме противоречий в выборе шага интегрирования при решении больших систем уравнений вычислитель сталкивается с проблемой выбора численного метода. Часто применение одного численного метода ко всей системе уравнений является малоэффективным или невозможным с точки зрения устойчивости. В то же время по отношению к подсистемам исходной системы оказывается эффективным применение разных методов (или одного метода с разным набором параметров).

В работе предлагается неклассический подход к вышеуказанным проблемам. Основное внимание уделяется алгоритму совместного расчёта нескольких подсистем с различным шагом интегрирования, а также соответствующему семейству алгоритмов синхронизации этих подсистем (определению моментов обмена данными). Благодаря этому алгоритму удаётся снизить вычислительные затраты, поскольку он позволяет в той или иной степени использовать многие способы оптимизации численного решения. Это снижение особенно проявляется при решении систем дифференциальных уравнений высокой размерности.

Подход, заложенный в основу разработанного алгоритма, имеет внешнее сходство с некоторыми существующими подходами (описанными выше), но обладает существенной новизной. В частности, в отличие от существующих многоскоростных методов в предлагаемом подходе внутреннее состояние подсистемы скрыто от других подсистем на протяжении её собственного шага по времени. Как следствие, этот подход к повышению скорости расчёта применим также и к проблеме расчёта гетерогенных моделей (необязательно имеющих высокую размерность, но обычно доступных лишь в виде набора независимых программ или программных модулей) [7].

В работе на примере нескольких задач проводятся аналитическое и численное исследования поведения погрешности, вводимое алгоритмом решения, и показывается первый порядок его ап-

проксимации, а также способы повышения порядка для конкретных методов. В работе не рассматриваются методы расщепления на подсистемы, а также алгоритмы определения степени связанности системы (см. ниже).

Дадим следующее определение. Слабосвязанными системами уравнений будем называть системы (как линейные, так и нелинейные), которые при помощи равносильных преобразований можно представить в виде совокупности подсистем, решение каждой из которых «слабо зависит» от решений других подсистем. В случае с линейными системами можно дать более наглядное определение. Линейные слабосвязанные системы уравнений — системы, матрица которых приводится к блочно-диагональному виду. Ранг матриц блочного пересечения (рис. 1) определяет количество переменных, по которым две подсистемы слабо связаны между собой. Координаты матриц пересечения указывают на эти переменные. Норма матриц определяет степени связанности подсистем.

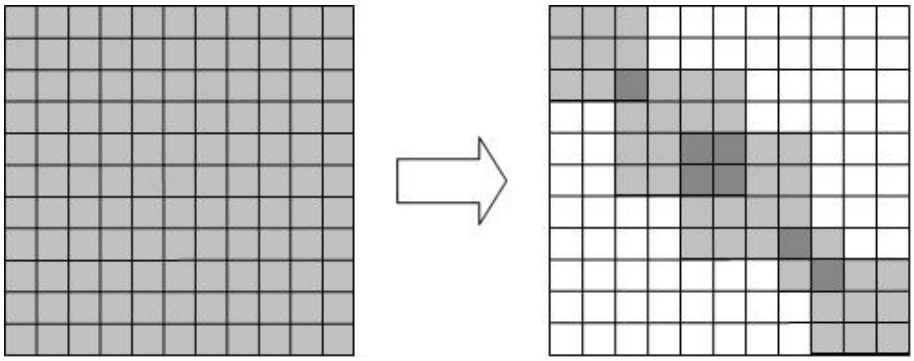

Рис. 1. Пример приведения заполненной матрицы к блочно-диагональному виду

Задачи, описываемые слабосвязанными системами уравнений, на практике в той или иной степени встречаются довольно часто, в основном это задачи совместного моделирования разнородных процессов. Для примера приведём несколько задач, описываемых слабосвязанными системами уравнений, из разных областей. Примером слабосвязанной жёсткой системы является система последовательных химических реакторов. В качестве примера слабосвязанной колебательной системы можно взять современную модель большой RLC интегральной схемы, в которой четко выделяются небольшие подсхемы. При относительно больших размерах элементов схемы получаемые таким образом подсистемы ОДУ можно рассматривать как несвязанные между собой. Однако появление в XXI в. субмикронных технологий привело к усилению взаимодействия между подобными подсхемами. Но поскольку сопротивления, ёмкости и индуктивности соединений между подсхемами все же во много раз отличаются от соответствующих параметров самих подсхем, такая система является слабосвязанной.

Определение слабой связанности можно распространить на любые системы, описывающие слабовзаимодействующие процессы, то есть на системы, изначально имеющие естественное разделение по процессам. При этом математическое описание таких систем может быть произвольным. В частности, подход, представленный в работе, можно применять к более широкому классу дифференциальных задач — к уравнениям в частных производных. В работе [8] представлена комплексная модель ионосферно-плазмосферного взаимодействия, описывающая различные физические процессы (перенос ионов вдоль силовых линий, нагревание плазмы, переносы фотоэлектронов в магнито-сопряженных ионосферах, горизонтальный нейтральный ветер), имеющие слабое влияние друг на друга. Модель описывается системой дифференциальных уравнений в частных производных смешанного типа. Расчёт задачи на одной итерации по времени требует привлечения сложных стратегий расчёта, однако вычисления можно существенно упростить и ускорить, если рассматривать систему с точки зрения слабой связанности и предлагаемого подхода (алгоритма синхронизации).

Положим, что задача задается нестационарной системой (системой ОДУ), которую можно разделить на несколько нестационарных подсистем. Основная задача алгоритма — периодическая синхронизация множественных процессов. Каждый процесс занимается численным решением одной или нескольких подсистем, то есть определяет отдельный решатель. Таким образом, каждому процессу соответствует, вообще говоря, система дифференциальных уравнений:

$$\frac{d\vec{u}}{dt} = \vec{f}(\vec{u},t).$$

Такая система описывает некоторый (физический) процесс, протекающий с некоторой скоростью, определённой $\vec{f}'_{\vec{u}}(\vec{u})$ (матрицей Якоби). Эта скорость определяет оптимальный шаг τ интегрирования при решении подсистемы

$$\tau \left\| \vec{f}'_{\vec{u}}(\vec{u}) \right\| \ll 1.$$

Процессу с большой скоростью соответствует маленький шаг по времени, процессу с маленькой скоростью — большой. Характерные времена ис-

следуемых процессов, заключённых в общей системе, могут различаться на несколько порядков (более чем в 10^{12} раз для жёстких задач [9]). Максимальный шаг среди всех подсистем будем называть макрошагом всего расчёта; это определение можно распространить на подмножество подсистем. Ключевым в алгоритме является независимость вычислительной работы решателей на протяжении одного макрошага (см. существующие подходы выше). По окончании каждого макрошага происходит взаимная синхронизация решателей посредством обмена значениями собственных параметров (переменных).

Алгоритм формализуется следующим образом. Каждый решатель может находиться в любой момент времени только в одном из нескольких состояний: в состоянии выполнения расчётов или состоянии ожидания. Решатель может перейти в состояние выполнения расчётов только после того, как значения всех собственных параметров будут обновлены значениями параметров других решателей. Решатель может закончить макрошаг и перейти в состояние ожидания только после обновления параметров других решателей значениями своих параметров. Первый процесс будем называть процессом прямой синхронизации по отношению к решателю (рис. 2), второй — обратной синхронизацией (рис. 2).

На протяжении своего шага более медленному процессу не нужно знать о каких-либо изменениях, происходящих в более быстрых процессах; с точки зрения более быстрых процессов изменения, происходящие в более медленных процессах до их завершения шага, считаются незначительными. Благодаря такому подходу достигается независимость решателей по отношению друг к другу. Если процесс обратной синхронизации однозначно определен, процесс прямой синхронизации является более сложным. Как упоминалось выше, решатель не может начать свой шаг вычислений, пока ему извне не сообщат «новые» данные. «Новыми» по отношению к решателю считаются те данные, которые известны в конце временного шага этого решателя. После каждого шага решатель проверяет, не стали ли данные, полученные в результате численного решения, новыми для каких-либо других решателей. В случае положительного ответа решатель осуществляет обратную синхронизацию по отношению к себе и прямую по отношению к другим решателям. Наглядно выше приведённые слова проиллюстрированы на рис. 3.

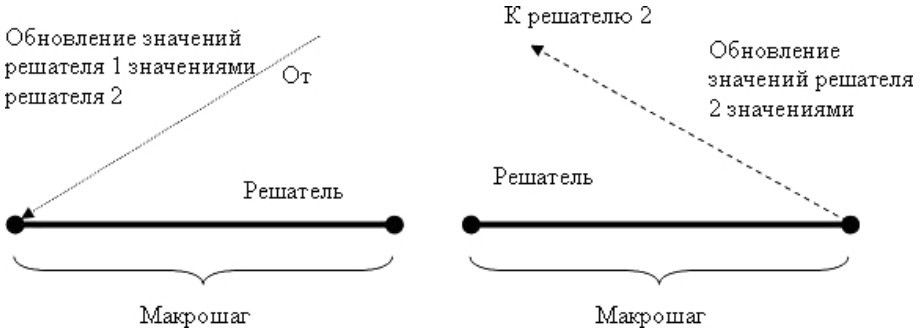

Рис. 2. Процесс прямой синхронизации для решателя 1 и обратной для решателя 2 (слева). Процесс обратной синхронизации для решателя 2 и прямой для решателя 1 (справа)

Для удобства на рис. 3 представлены процессы синхронизации только между «соседними» решателями. По горизонтальной оси откладывается математическое время, по вертикальной — решатели с разными шагами, точкой изображается момент завершения численного шага решателя, стрелки определяют процессы прямой (сплошная линия) и обратной (пунктирная линия) синхронизаций между решателями по отношению к нижнему решателю. Следует обратить внимание на то, что при такой стратегии процессов синхронизации (прямой и обратной) макрорасчёт представляет собой последовательный (или в некоторых случаях квазипоследовальный) процесс (рис. 3). Распараллеливание таких расчётов является бесполезным (или невозможным). Если теперь рассмотреть стратегию, в которой все решатели могут начинать свои макрошаги вычислений со «старыми» данными, и обмениваться «новыми» лишь в конце макрошага, то мы получим строго распараллеливающийся расчёт (рис. 4). Отрицательной стороной такой стратегии является то, что все решатели проводят вычисления, основываясь на устаревших значениях параметров других решателей. Это хорошо видно сравнивая рисунки.

Безусловно, появляется погрешность, обусловленная запаздыванием данных, зато такая стратегия позволяет легко строго распараллелить расчёт. Полностью последовательный (рис. 3) и полностью параллельный (рис. 4) расчёты являются предельными стратегиями проведения расчётов. Существует компромисс между скоростью вычислений и погрешностью, обусловленной этой скоростью, который заключается в задании моментов обратной синхронизации внутри макрошага. На рис. 5 представлен пример прямой синхронизации на середине макрошага (или в моменты времени, наиболее близкие к середине макрошага). Момент синхронизации определяет степень распараллеливания расчётов. При необходимости она может быть задана любой от 0 до 1.

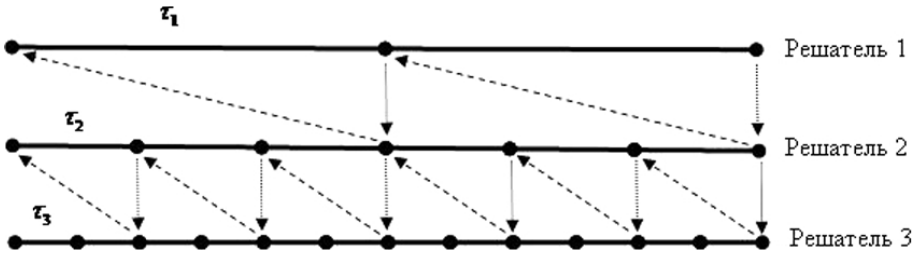

Рис. 3. Пример классического решения системы, последовательный вариант

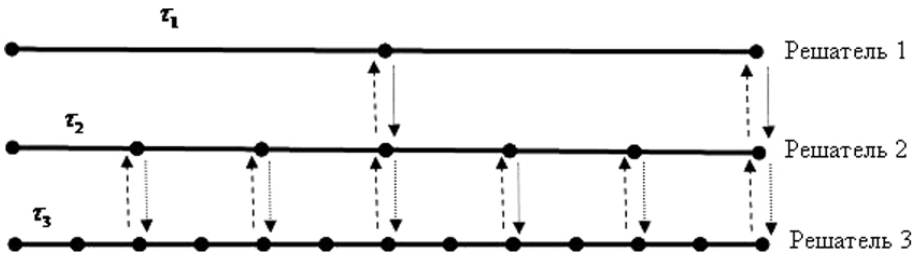

Рис. 4. Пример распараллеливающегося варианта решения системы

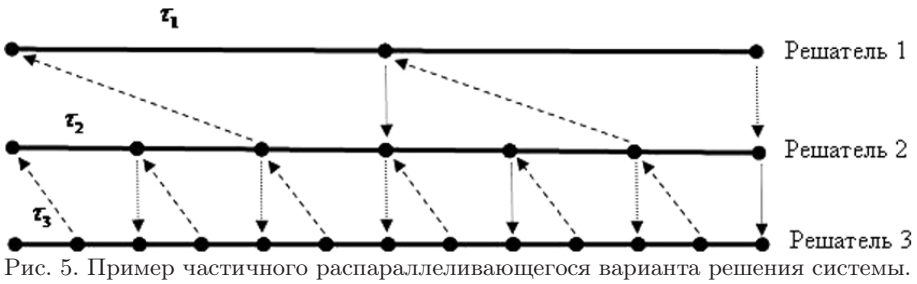

Рис. 5. Пример частичного распараллеливающегося варианта решения системы. Прямая синхронизация осуществляется на середине макрошага

Выбор той или иной стратегии должен опираться не только на точности и времени расчёта. Важно также учитывать такие параметры, как время, затрачиваемое на синхронизацию, и степень разбалансировки нагрузки между вычислительными ресурсами. Первый параметр определяется техническими особенностями вычислительной системы (например, значение параметра возрастает при синхронизации через сеть) и числом переменных, по которым осуществляется синхронизация. Разбалансировка определяется процессорными ресурсами, выделяемыми на выполнение одной итерации для подсистемы. Вычислительные эксперименты показывают, что при больших задержках синхронизации и большой разбалансировке параллельный вариант теряет преимущество в скорости по отношению к полупараллельному, при этом параллельная стратегия вносит максимальную погрешность алгоритма.

Случай, когда каждый решатель может начать серию шагов по времени точно в начале макрошага (как всей системы, так и любой подсистемы, «содержащей» этот решатель) и завершить её точно в конце макрошага, называется случаем кратных шагов. Все остальные случаи, когда серия шагов не умещается точно в макрошаг, называются случаем некратных шагов. Случай кратных шагов встречается только тогда, когда вычислитель самостоятельно задает (постоянные) шаги интегрирования, если шаги определяются автоматически из свойств системы, то следует ожидать только некратные шаги. Случаи синхронизации решателей с некратными шагами существенно отличаются от случаев с кратными. Разные соотношения между шагами требуют применения разных стратегий (прямой и обратной) синхронизаций. Случай некратных шагов в этой статье не рассматривается, поскольку он носит скорей прикладной характер.

На практике при решении систем большой размерности применяются численные методы, требующие небольших вычислительных ресурсов. К таким методам относятся явные и неявные методы невысокого порядка — первого или второго. Поэтому дальнейшие аналитические оценки будем проводить для двух методов, различных по своим свойствам: явного метода Эйлера первого порядка и метода трапеций (неявного метода второго порядка). Как упоминалось выше, применение вышеописанных алгоритмов синхронизации неизбежно приводит к появлению дополнительной погрешности. Для квазипоследовательного алгоритма (рис. 3) оценки погрешности можно проделать аналитически. Проведение подобных оценок для частично распараллеливающихся и строго распараллеливающихся вариантов алгоритма не имеют большого практического значения, поскольку поведение этих вариантов зависит от случайных факторов (передача данных по сети, работа пла-

нировщика задач в операционной системе и т.п.). Эти погрешности могут определяться не аналитически, а экспериментально.

Алгоритмическую особенность и некоторые математические выводы удобно проделать на примере механизма оценки погрешности численного метода на одном шаге (на одной итерации) по времени. Для простоты будем рассматривать линейную систему из двух уравнений (с «медленной» переменной x и «быстрой» y):

$$\begin{cases} \frac{dx}{dt} = f_x(x,y), \\ \frac{dy}{dt} = f_y(x,y), \end{cases}$$

где правая часть

$$\begin{cases} f_x(x,y) = ax + by, \\ f_y(x,y) = cx + dy. \end{cases}$$

Анализ погрешности будем проводить покомпонентно — как разность между решением с общим шагом τ и решением с шагом для x, равным $\tau_x = k\tau$ (k — целое число, определяющее степень кратности шагов для «быстрых» и «медленных» переменных; шаг для y равен τ). В нижеприведённых выкладках приводится лишь главный член суммы, то есть отбрасываются члены, имеющие больший порядок малости по τ.

Классический явный метод Эйлера имеет следующий вид (случай $k = 1$):

$$\begin{cases} x_{n+1} = x_n + \tau f_x(x_n,y_n), \\ y_{n+1} = y_n + \tau f_y(x_n,y_n). \end{cases}$$

В случае проведения расчётов с использованием алгоритма синхронизации для произвольной кратности k имеем следующую (модифицированную) разностную задачу:

$$\begin{cases} x_{n+k} = x_n + k\tau f_x(x_n,y_n), \\ y_{n+1} = y_n + \tau f_y(x_n,y_n), \\ y_{n+2} = y_{n+1} + \tau f_y(x_n,y_{n+1}), \\ \dots \\ y_{n+k} = y_{n+k-1} + \tau f_y(x_n,y_{n+k-1}), \end{cases}$$

то есть многократное использование метода Эйлера при условии, что для «медленной» переменной $x_n = x_{n+1} = x_{n+2} = \dots = x_{n+k-1}$. Погрешности по разным компонентам на шаге τ решения, полученного при помощи модифицированного метода Эйлера, по отношению к решению классическим метод соответственно равны

$$E_x = g(k)\tau^2(af_x + bf_y) + O(\tau^3),$$

$$E_y = g(k)\tau^2 cf_x + O(\tau^3),$$

где $g(k) = C_k^2 = k(k-1)/2$. При этом погрешность самого метода Эйлера составляет

$$\varepsilon_x = \frac{\tau_x^2}{2}(af_x + bf_y) + O(\tau_x^3) =$$

$$= \frac{k^2\tau^2}{2}(af_x + bf_y) + O(\tau^3),$$

$$\varepsilon_y = \frac{\tau^2}{2}(cf_x + df_y) + O(\tau^3),$$

(заметим, что это погрешности при разных шагах — при тех, которые считаются приемлемыми для численного решения соответствующих уравнений). Отсюда следует, что главный член погрешности алгоритма E_x (с точки зрения естественного для переменной x шага τ_x) при всех k меняется от $\varepsilon_x/2$ до ε_x, то есть её отношение к погрешности метода (Эйлера) не зависит от степени влияния второго уравнения на первое ($\alpha_{yx} = |b/a|$). Для интерпретации погрешности по y следует ввести другую степень связанности переменных — $\alpha_{xy} = |c/d|$ — и предположить слабую связанность: $\alpha_{xy} \ll 1$. Можно определить предельную кратность шага k, при которой погрешность алгоритма не превышает погрешность метода:

$$\frac{k^2}{2}\tau^2|cf_x| \approx g(k)\tau^2|cf_x| \leqslant \frac{\tau^2}{2}|cf_x + df_y| \approx \frac{\tau^2}{2}|df_y|.$$

Отсюда кратность шага $k \leqslant k^* = \sqrt{\left|\frac{df_y}{cf_x}\right|}$.

Заметим, что для неколебательных систем введённое выше понятие «быстрых» и «медленных» подсистем означает, что отношение $|f_y/f_x|$ больше единицы (для колебательных это подразумевает другое — различие периодов). Обычно кратность шагов делают пропорциональной или просто равной отношению производных $|f_y/f_x|$ (обратному отношению характерных времен). В этом случае получаем: $k^* = |d/c|$. Но независимо о того, связан ли выбор параметра алгоритма k с правой частью системы $|f_y/f_x|$, слабая связанность подсистем ($|c/d| \ll 1$) даёт возможность полагать k достаточно большим.

Метод трапеций в классическом виде выглядит следующим образом:

$$\begin{cases} x_{n+1} = x_n + \frac{\tau}{2}(f_x(x_n,y_n) + f_x(x_{n+1},y_{n+1})), \\ y_{n+1} = y_n + \frac{\tau}{2}(f_y(x_n,y_n) + f_y(x_{n+1},y_{n+1})). \end{cases}$$

Модифицированная разностная задача для произвольной кратности k имеет следующий вид:

$$\begin{cases} x_{n+k} = x_n + \frac{k\tau}{2}(f_x(x_n,y_n) + f_x(x_{n+k},y_{n+k})), \\ y_{n+1} = y_n + \frac{\tau}{2}(f_y(x_n,y_n) + f_y(x_n,y_{n+1})), \\ y_{n+2} = y_{n+1} + \frac{\tau}{2}(f_y(x_n,y_{n+1}) + f_y(x_n,y_{n+2})), \\ \dots \\ y_{n+k-1} = y_{n+k-2} + \frac{\tau}{2}(f_y(x_n,y_{n+k-2}) + \\ \qquad + f_y(x_n,y_{n+k-1})), \\ y_{n+k} = y_{n+k-1} + \frac{\tau}{2}(f_y(x_n,y_{n+k-1}) + \\ \qquad + f_y(x_{n+k},y_{n+k})). \end{cases}$$

Анализ погрешности для этой задачи (аналогичный вышеприведённому анализу для метода Эйлера) показывает, что порядок метода трапеций (неявного метода второго порядка) в случае применения алгоритма синхронизации снижается до первого. При этом погрешность модифицированного метода на шаге не превышает погрешности метода Эйлера, а неявность метода не приводит к

дополнительным погрешностям. Однако даже если погрешность алгоритма синхронизации существенно больше погрешности численного метода (например, в сильно связанных подсистемах или в системах с переменным во времени отношением скоростей — например, в колебательных системах), это отнюдь не означает неприменимость алгоритма. Дело в том, что на практике в большинстве случаев наибольший вклад в итоговую погрешность результата вносит не численный метод, а неопределённость исходных данных. Поэтому именно с этой неопределённостью в каждом случае имеет смысл сопоставлять погрешность предлагаемого подхода.

С другой стороны, для квазипоследовательной стратегии синхронизации (рис. 3) порядок метода может быть повышен искусственно за счёт вычитания главных членов невязки из правых частей разностной задачи. Рассмотрим этот вариант на примере описанного выше метода трапеций. Модифицированная разностная задача, аппроксимирующая исходную дифференциальную задачу со вторым порядком, имеет следующий вид:

$$\begin{cases} x_{n+k} = x_n + \frac{k\tau}{2}(f_x(x_n,y_n) + f_x(x_{n+k},y_{n+k})), \\ y_{n+1} = y_n + \frac{\tau}{2}(f_y(x_n,y_n) + f_y(x_n,y_{n+1})) + F, \\ y_{n+2} = y_{n+1} + \frac{\tau}{2}(f_y(x_n,y_{n+1}) + f_y(x_n,y_{n+2})) + \\ \qquad +3F, \\ \ldots \\ y_{n+k-1} = y_{n+k-2} + \frac{\tau}{2}(f_y(x_n,y_{n+k-2}) + \\ \qquad +f_y(x_n,y_{n+k-1})) + (2k-3)F, \\ y_{n+k} = y_{n+k-1} + \frac{\tau}{2}(f_y(x_n,y_{n+k-1}) + \\ \qquad +f_y(x_{n+k},y_{n+k})) + kF, \end{cases}$$

где $F = \frac{\tau^2}{2}c[ax_n + by_n] = \frac{\tau^2}{2}cf_x(x_n,y_n)$. Таким образом, порядок метода может контролироваться во время расчёта. Однако этот контроль потребует дополнительного (хотя и не значительного) числа арифметических операций.

Для оценки погрешности численного решения на практике больший интерес представляет оценка погрешности решения от времени. Для расчёта расщепленной задачи методом Эйлера решение можно записать в следующем явном виде:

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \Omega \begin{pmatrix} w_n \\ v_n \end{pmatrix},$$

где Ω — матрица из собственных векторов матрицы системы — матрица перехода к диагональному виду. Пропуская промежуточные математические выкладки, явный вид численного решения системы $\{w_n,v_n\}$ в базисе из собственных векторов может быть представлен в виде

$$\begin{cases} w_n = C_1 e^{t_n\lambda_x} \cdot \left(1 - \frac{k}{2}\tau t_n \lambda_x^2\right) + O(\tau^2), \\ v_n = C_2 e^{t_n\lambda_y} \cdot \left(1 + \tau t_n \left(\frac{k-1}{2}d^2 - \frac{k}{2}\lambda_y^2\right)\right) + O(\tau^2), \end{cases}$$

где λ_x — скорость медленного процесса (собственное число матрицы по «медленной» переменной x), где λ_y — скорость быстрого процесса (собственное число матрицы по «быстрой» переменной y). Полученное выражение справедливо при условии $bc \ll ad$ (условия слабой связанности подсистем, см. также постановку задачи). Таким образом, можно сделать следующую оценку абсолютной погрешности численного решения:

$$\Delta_n = \Omega \begin{pmatrix} C_1 e^{t_n\lambda_x} \cdot \frac{k}{2}\tau t_n \lambda_x^2 \\ C_2 e^{t_n\lambda_y} \cdot \tau t_n \left(\frac{k-1}{2}d^2 - \frac{k}{2}\lambda_y^2\right) \end{pmatrix} + O(\tau^2).$$

Помимо практической составляющей полученного результата важно подчеркнуть линейный характер зависимости погрешности от степени кратности, что подтверждается на практике.

Разработанный подход был применен при проведении вычислительного эксперимента для модели электродвигателя [4].

$$\frac{1}{\omega_s}\frac{d\Psi_d}{dt} = -\frac{R_a}{L'_d}\Psi_d + \frac{R_a}{L'_d}E'_q + \frac{\omega}{\omega_s}\Psi_q + V\sin\delta,$$

$$\frac{1}{\omega_s}\frac{d\Psi_q}{dt} = -\frac{R_a}{L'_d}\Psi_q - \frac{R_a}{L'_d}E'_d - \frac{\omega}{\omega_s}\Psi_d + V\cos\delta,$$

$$T'_{q0}\frac{dE'_d}{dt} = -\frac{L_q}{L'_q}E'_d - \frac{(L_q - L'_q)}{L'_q}\Psi_q,$$

$$T'_{d0}\frac{dE'_q}{dt} = -\frac{L_d}{L'_d}E'_q - \frac{(L_q - L'_q)}{L'_q}\Psi_d + E_{xc},$$

$$\frac{d\delta}{dt} = \omega - \omega_s,$$

$$\frac{2H}{\omega_s}\frac{d\omega}{dt} =$$

$$= T_m + \left(\frac{1}{L'_q} - \frac{1}{L'_d}\right)\Psi_d\Psi_q + \frac{1}{L'_q}\Psi_d E'_d + \frac{1}{L'_d}\Psi_q E'_q.$$

Модель описывает два статора с потокосцеплениями Ψ_d и Ψ_q, напряжениями, пропорциональными потокосцеплениям поля E'_q и демпферной обмотки E'_q, и частотой ω. Численное решение представленной системы показано на рис. 6 (а именно: решение для двух переменных Ψ_d и δ).

Расчёт методом Эйлера для разных степеней кратности шагов интегрирования подсистем представлен на рис. 7. На графиках виден равномерный рост алгоритмической погрешности от степени кратности. Увеличение кратности в 2 раза приводит к возрастанию погрешности в среднем в 1.5 раза.

Исследования более медленных составляющих процессов, протекающих в системе, проделано на переменных Ψ_d и δ с использованием метода Рунге–Кутты 4-го порядка (рис. 8).

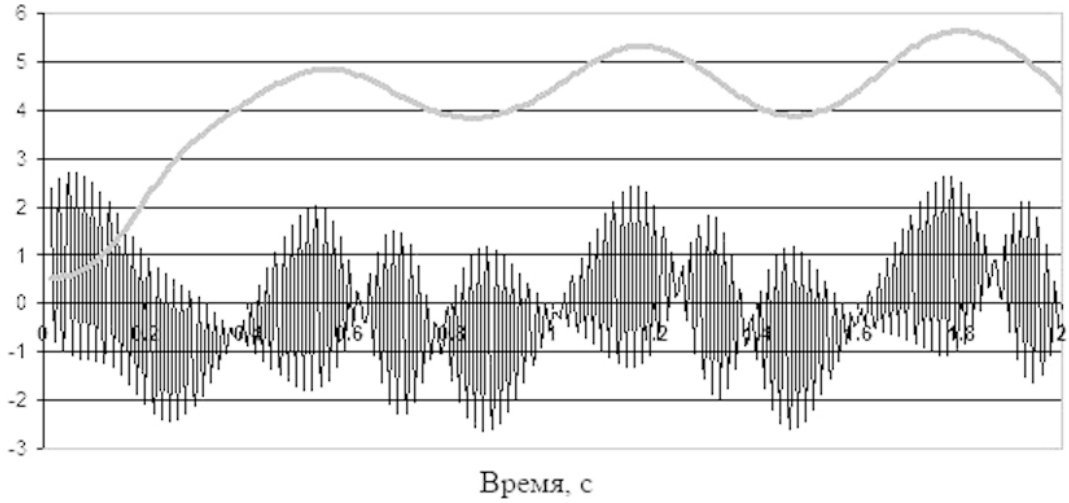

Рис. 6. Численное «точное» решение «быстрой» Ψ_d и «медленной» δ переменных

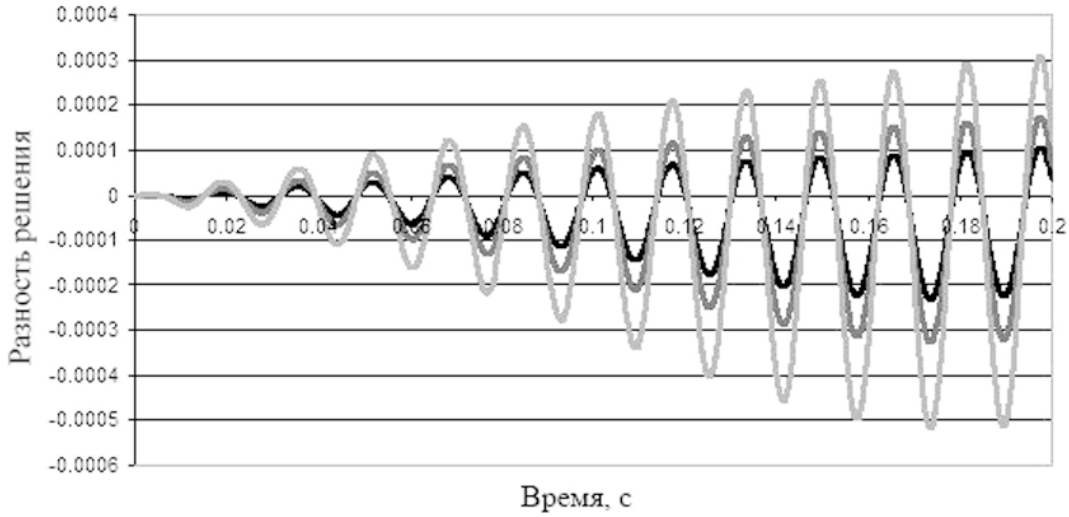

Рис. 7. Зависимость алгоритмической погрешности решения для переменной Ψ_d от степени кратности шагов для метода Эйлера. Темный график — кратность 2, серый — кратность 4, светлый график — кратность 8

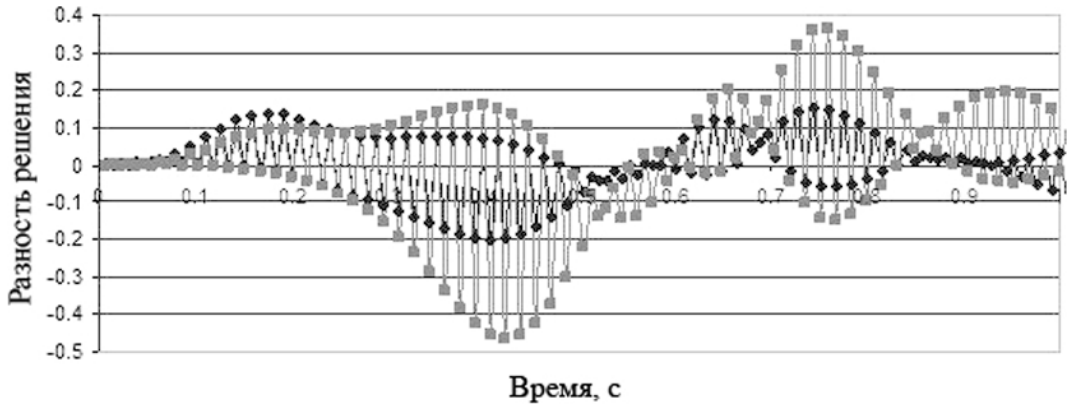

Рис. 8. Зависимость алгоритмической погрешности решения для «быстрой» переменной Ψ_d от степени кратности шагов для метода Рунге–Кутты 4-го порядка. Темный график — кратность 2, светлый — кратность 4

При расчёте задач с применением синхронизации подсистем наблюдается линейный рост (абсолютной) погрешности решения от кратности шага. Это объясняется снижением порядка методов до 1-го. Величина погрешности, вносимой за счёт асинхронного расчёта подсистем, является малой для задач, описываемых слабосвязанными системами уравнений, в большинстве случаев она не превышает погрешности обычного расчёта методом первого порядка. Уже на системах размерности порядка 10 наблюдается приращение скорости счета при малых кратностях шагов; при этом

есть все основания полагать, что ускорение расчёта будет существенно большим на системах высокой размерности (подход ориентирован на системы порядка более, чем 10^3). Более того, время расчёта может быть существенно сокращено за счёт применения адаптированных под вычислительную систему алгоритмов синхронизации, в частности, алгоритма параллельного расчёта. Результаты расчётов с применением последовательного алгоритма синхронизации подтверждаются аналитическими оценками.

Таким образом, в работе обосновано применение подхода для решения ОДУ большой и умеренной размерности. Возможность гибкого контроля точности и скорости расчёта позволяет подстраивать алгоритм для ускоренного моделирования в тех случаях, когда неопределённость входных данных играет принципиальную роль. Подход хорошо применим в областях, где требуется или возможно распараллеливание вычислительной нагрузки. Существует большой спектр областей, где остро стоит потребность в ускоренном моделировании. Дальнейшее развитие работы планируется проводить в схемотехническом направлении.

Данная работа выполняется в рамках проекта № 09-07-00077 при поддержке РФФИ.

Литература

1. *Эндрюс Г.* Основы многопоточного параллельного и распределенного программирования. — М.; СПб.; Киев, 2003.

2. *Денисенко В.* Проблемы схемотехнического моделирования КМОП СБИС // Компоненты и технологии. — 2002. — № 3–4.

3. *Haug E.J., Negrut D., Serban R., Solis D.* Numerical methods for high speed vehicle dynamics simulation // Mech. of Structures and Machines. — 1999. — V. 27, N. 4. — P. 507–533.

4. *Crow M.L., Chem James G.* The multirate methods for simulations of power system dynamics // IEEE Transactions on Power System. — 1994. V. 9, N. 3. — P. 1684–1690.

5. *Shome S.S., Haug E.J., Jay L.O.* Dual-rate integration using partitioned Runge–Kutta methods for mechanical systems with interacting subsystems // Mechanics based design of structures and machines. — 2004, V. 32, N. 3. — P. 253–282.

6. *Andrus J.F.* Automatic Integration of systems of second-order ODE's separated into subsystems // SIAM Journal on Numerical Analysis. — 1983. — V. 20, N. 4. — P. 815–827.

7. *Корчак А.Б., Евдокимов А.В.* Система интеграции гетерогенных моделей динамических систем // Моделирование и обработка информации. — М., 2008.

8. *Кринберг И.А., Тащилин А.В.* Ионосфера и плазмосфера. — М.: Наука, 1984.

9. *Петров И.Б., Лобанов А.И.* Лекции по вычислительной математике. — М.: Интернет–Университет информационных технологий; Бином. Лаборатория знаний, 2006. — 523 с.

Поступила в редакцию 25.12.2009.