КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2009 Т. 1 № 2 С. 129−138

Ки&М

МАТЕМАТИЧЕСКИЕ ОСНОВЫ И ЧИСЛЕННЫЕ МЕТОДЫ МОДЕЛИРОВАНИЯ

Система интеграции гетерогенных моделей и ее применение к расчету слабосвязанных систем дифференциальных уравнений

А. Б. Корчак[1,a], А. В. Евдокимов[1]

[1] Московский физико-технический институт (государственный университет), 141700, г. Долгопрудный Московской обл., Институтский переулок, д. 9

E-mail: [a] korchak_anton@mail.ru

Получено 26 марта 2008 г.

Разрабатывается программная система интеграции динамических моделей, неоднородных по своим математическим свойствам и/или по требованиям к шагу по времени. Предлагается семейство алгоритмов параллельного расчета гетерогенных моделей с разными шагами по времени. 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: гетерогенные модели, решение дифференциальных уравнений, параллельный расчет

Tool for integration of heterogeneous models and its application to loosely coupled sets of differential equations

A. B. Korchak[1], A. V. Evdokimov[1]

[1] *Moscow Institute of Physics and Technology (State University), Institutskii pereulok 9, 141700, Dolgoprudny, Moscow Region, Russia*

Abstract. — We develop the software tool for integration of dynamics models, which are inhomogeneous over mathematical properties and/or over requirements to the time step. The family of algorithms for the parallel computation of heterogeneous models with different time steps is offered. Analytical estimates and direct measurements of the error of these algorithms are made with reference to weakly coupled ODE sets. The advantage of the algorithms in the time cost as compared to accurate methods is shown.

Key words: heterogeneous models, differential equations solving, parallel computation

Citation: *Computer Research and Modeling*, 2009, vol. 1, no. 2, pp. 129−138 (Russian).

Введение

В области автоматизации научных исследований по численному моделированию актуальной является проблема построения гетерогенных моделей — моделей, неоднородных как математически, так и структурно. Многие научные направления развиваются сейчас за счет объединения воедино частных моделей и использования в одних моделях возможностей других (готовых). Однако чем более сложными и эффективными становятся модели и численные методы, тем больше усилий требуется для их реализации на конкретных задачах и тем больше вероятность того, что разработанная вычислительная модель будет применена лишь однократно (что мешает внедрению научных разработок в практику). Проблему построения гетерогенных моделей можно рассматривать как проблему повторного использования численных методов и моделей.

Существуют немногочисленные работы в области технологий моделирования, которые нацелены на преодоление проблемы гетерогенности (сложности) моделей. В результате получаются довольно громоздкие программные системы [1], область применения которых лишь ненамного шире, чем область применения объединяемых моделей. В качестве альтернативного подхода, существуют решения проблемы быстрой разработки моделей путем использования высокоуровневых программных библиотек [1]. Разработчику предлагается «собрать» программу для своей модели из имеющихся в библиотеке модулей (запрограммировав специфику). Однако такой подход может применяться лишь специалистами с глубокими знаниями в области программирования и проектирования программ.

Идея, развиваемая в данной работе, является компромиссом между изложенными выше подходами к проблемам гетерогенности и повторной используемости моделей. Реализуемую на основе этой идеи систему интеграции моделей предполагается применять специалистам по численному моделированию.

Отдельное внимание в работе уделяется созданию алгоритма совместного расчета нескольких моделей, гетерогенных с точки зрения математического шага по времени. Эта часть работы, в первую очередь, направлена на повышение эффективности численного решения систем дифференциальных уравнений высокой размерности — на снижение вычислительных затрат за счет использования разных временных шагов для подсистем и, как следствие, за счет сокращения числа арифметических операций. Но в отличие от существующих механизмов расчета систем, разделенных на подсистемы с разными шагами [2, 3], в предлагаемом подходе внутреннее состояние подсистемы скрыто от других подсистем на протяжении ее собственного шага по времени. Как следствие, этот подход к повышению скорости расчета применим также и к проблеме расчета гетерогенных моделей (необязательно имеющих высокую размерность, но обычно реализованных в виде отдельных программ или программных модулей); поэтому он положен в основу расчетного модуля системы интеграции моделей. И наоборот, большая система дифференциальных уравнений, даже если она задана как единое целое, подлежит расчету с помощью системы интеграции (особенно если в ней есть слабо связанные подсистемы, имеющие различные характерные времена).

Система интеграции моделей

Многие современные пакеты моделирования (такие как Simulink, ModelVision) позволяют производить процесс моделирования, собирая модели из готовых элементов «конструктора». Такой широко распространенный подход обладает очень низкими временными затратами, легкостью в использовании и не требует от вычислителя знания языков программирования. Но недостатком такого подхода является ограниченность наборов элементов «конструктора» наряду с тем фактом, что возможность подключения собственных элементов ограничена жестко заданным интерфейсом. Как следствие, пакеты визуального моделирования имеют весьма узкую область применения и пригодны для моделирования относительно несложных систем (моделей).

Построение (сборка) принципиально более сложной модели либо невозможна, либо требует большого количества времени.

В данной работе предлагается подход/инструмент, в котором модель может собираться также из блоков, но произвольной природы, не обязательно изначально заданных в качестве элементов «конструктора». Каждый блок является экземпляром некоторого «модуля» (со своими входными и выходными данными), а в качестве «модулей» рассматриваются программы (*.exe), представляющие собой отдельные процессы операционной системы, динамически загружаемые библиотеки (*.dll), Java-библиотеки (*.jar). В этом состоит отличие от стандартных пакетов, где пакет должен знать внутреннюю структуру всех модулей, чтобы из них сформировать общую систему дифференциально-алгебраических уравнений. Предлагаемая же система использует не исходный, а выполняемый код «модулей» и обеспечивает их взаимодействие путем обмена входными и выходными данными в процессе моделирования. Обмен данными может осуществляться через разные каналы передачи, например, через форматированные файлы, командную строку, вызов методов из jar/DLL. Принципиально, что формат/структура файлов и других каналов передачи не заданы жестко, а определяются пользователем в процессе интеграции.

Алгоритмы синхронизации при расчетах с разными шагами

На практике часто встречаются задачи, численное решение которых требует решения подзадач с разными шагами. Условно с точки зрения величины расчетного шага все подзадачи можно разделить на задачи с «медленными переменными» и задачи с «быстрыми переменными». Возможное улучшение характеристик (например, уменьшение временных затрат) численного решения таких задач диктуется учетом ее специфики, а именно различием в характерных шагах. Существуют разные подходы к оптимизации решения подобного типа задач [2, 3]. При этом существенной особенностью таких решений является сохранение понятия единого вектора системы. Фактически решается задача на каждом «быстром» шаге во времени — в том числе для «медленных» переменных. То есть качественные изменения затрагивают только способ получения вектора решения на каждой итерации. Примером тому является работа [3], в которой и «медленные» и «быстрые» переменные рассчитываются в одни и те же моменты времени, при этом для вычисления значений «медленных» переменных используется линейная интерполяция. Подобные походы лишь несущественно сокращают число арифметических операций, решая полную (но измененную) задачу.

Предлагаемый подход к решению подобных задач запрещает интерполяцию значений «медленных переменных», т. е. при расчете используются только известные значения соответствующих функций для подзадач. Другими словами, для вычисления правых частей подсистем «быстрых» переменных в моменты времени, соответствующие минимальному шагу интегрирования всей системы, используются последние вычисленные значения «медленных» переменных. Помимо достоинств такого подхода, которые будут рассмотрены ниже, следует отметить, что такие требования к алгоритму расчета позволяют решать каждую подзадачу отдельным решателем и рассматривать такой решатель как «черный ящик».

Положим, что задача задается нестационарной системой, которую можно разделить на несколько нестационарных подсистем. Для определенности будем рассматривать систему обыкновенных дифференциальных уравнений. Основная задача алгоритма — периодическая синхронизация множественных процессов. Каждый процесс занимается численным решением одной или нескольких подсистем, т. е. определяет отдельный решатель. Таким образом, каждому процессу соответствует, вообще говоря, система дифференциальных уравнений

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

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

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

Процессу с большой скоростью соответствует маленький шаг по времени, процессу с маленькой скоростью — большой. Характерные времена исследуемых процессов, заключенных в общей системе, могут различаться на несколько порядков (более чем в 10^{12} раз для жестких задач). Максимальный шаг среди всех подсистем будем называть макрошагом всего расчета; это определение можно распространить на подмножество подсистем. Ключевым в алгоритме является независимость вычислительной работы процессов на протяжении одного макрошага. По окончанию каждого макрошага происходит взаимная синхронизация процессов посредством обмена значениями собственных параметров (переменных).

Алгоритм формализуется следующим образом. Далее каждый решатель, выполняющий расчетный процесс, будем называть процессором. Каждый процессор может находиться в любой момент времени только в одном из нескольких состояний: в состоянии выполнения расчетов или состоянии ожидания. Процессор может перейти в состояние выполнения расчетов только после того, как значения всех собственных параметров будут обновлены значениями параметров других процессоров. Процессор может закончить макрошаг и перейти в состояние ожидания только после обновления параметров других процессоров значениями своих параметров. Первый процесс будем называть процессом прямой синхронизации по отношению к процессору, второй — обратной синхронизацией (см. рис. 1).

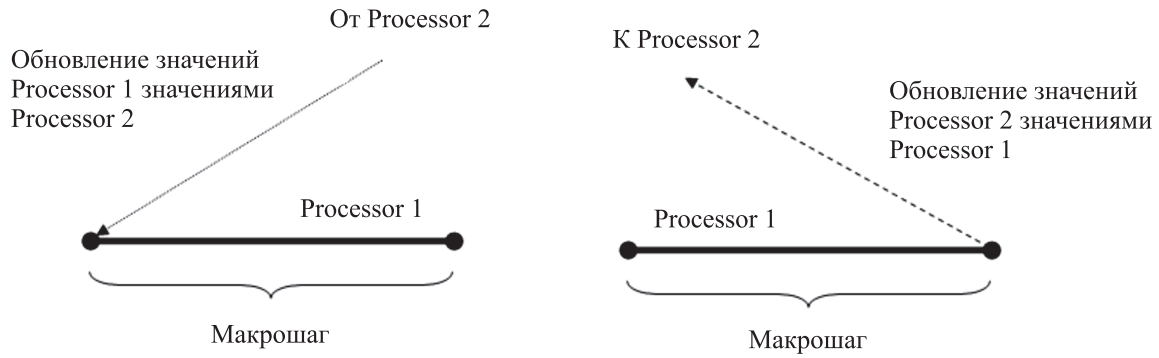

Рис. 1. Процесс прямой синхронизации для Processor 1 (первый рисунок). Процесс обратной синхронизации для Processor 1 (второй рисунок)

На протяжении своего шага более медленному процессу не нужно знать о каких-либо изменениях, происходящих в более быстрых процессах, этим достигается независимость процессоров по отношению к друг другу. С точки зрения более быстрых процессов изменения, происходящие в более медленных процессах до их завершения шага, считаются незначительными. Если процесс обратной синхронизации однозначно определен, процесс прямой синхронизации является более сложным. Как упоминалось выше, процессор не может начать свой шаг вычислений, пока ему извне не сообщат «новые» данные. «Новыми» по отношению к процессору считаются те данные, которые известны в конце временного шага этого процессора. После каждого шага процессор проверяет, не стали ли данные, полученные в результате численного решения, новыми для каких-либо других процессоров. В случае положительного ответа процессор осуществляет обратную синхронизацию по отношению к себе и прямую по отношению к другим процессорам. Наглядно вышеприведенные слова проиллюстрированы на рис. 2.

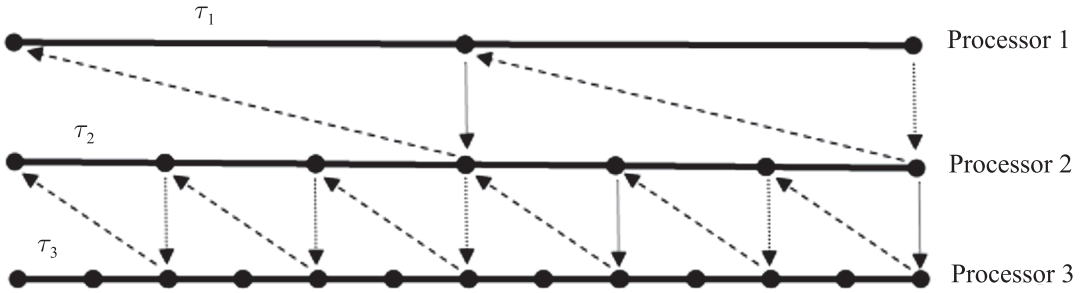

Рис. 2. Пример классического решения системы, последовательный вариант

Для удобства на рис. 2 представлены процессы синхронизации только между соседними процессорами. По горизонтальной оси откладывается математическое время, по вертикальной — процессоры с разными шагами, точкой изображается момент завершения численного шага решателя, стрелки определяют процессы прямой и обратной синхронизаций между процессорами по отношению к нижнему процессору. Следует обратить внимание на то, что при такой стратегии процессов синхронизации (прямой и обратной) макрорасчет представляет собой последовательный (или в некоторых случаях квазипоследовальный) процесс (см. рис. 2). Распараллеливание таких расчетов является бесполезным. Если теперь рассмотреть стратегию, в которой все процессоры могут начинать свои макрошаги вычислений со «старыми» данными и обмениваться «новыми» лишь в конце макрошага, то мы получим строго распараллеливающийся расчет (см. рис. 3). Отрицательной стороной такой стратегии является то, что все процессоры проводят вычисления, основываясь на устаревших значениях параметров других процессоров. Это хорошо видно, сравнивая рисунки.

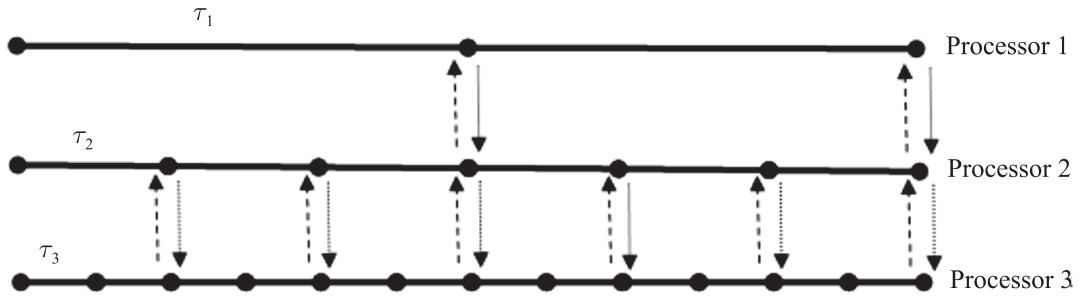

Рис. 3. Пример распараллеливающегося варианта решения системы

Безусловно, появляется погрешность, обусловленная запаздыванием данных, зато такая стратегия позволяет строго распараллеливать расчет. Наконец, существует компромисс между скоростью вычислений и погрешностью, обусловленной этой скоростью, который заключается в задании моментов обратной синхронизации внутри макрошага. На рис. 4 представлен пример обратной синхронизации на середине макрошага (или в моменты времени, наиболее близкие к середине макрошага).

Случай, когда каждый процессор может начать серию шагов по времени точно в начале макрошага и завершить ее точно в конце макрошага, называется случаем кратных шагов. Все остальные случаи, когда серия шагов не умещается точно в макрошаг, называется случаем некратных шагов. Разные соотношения между шагами требуют применения разных стратегий (прямой и обратной) синхронизаций. Например, для некратных шагов характерным становится размывание момента синхронизации.

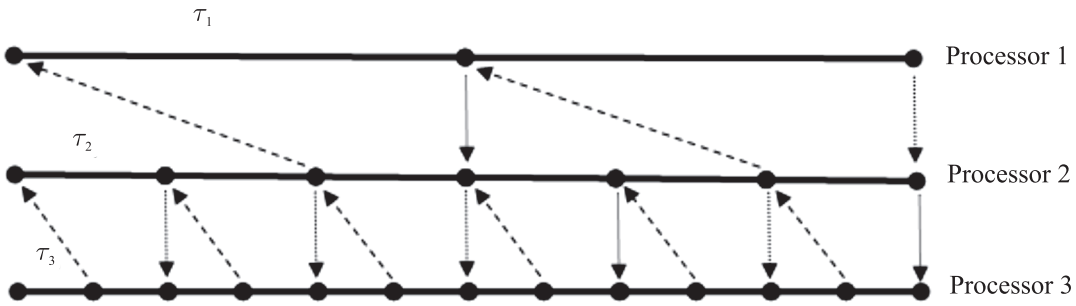

Рис. 4. Пример частичного распараллеливающегося варианта решения системы. Прямая синхронизация осуществляется на середине макрошага

Слабосвязанные системы обыкновенных дифференциальных уравнений

Описанное выше разбиение на подсистемы искажает исходную задачу, внося неустранимую погрешность численного решения. Но величина этой погрешности может быть существенно сокращена в случае применения алгоритма к выделенному классу задач. В случае задач с обыкновенными дифференциальными уравнениями — это системы слабосвязанных дифференциальных уравнений. Слабосвязанные системы уравнений — 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, причем максимальная норма и ранг матриц блочного пересечения определяют степень связанности системы.

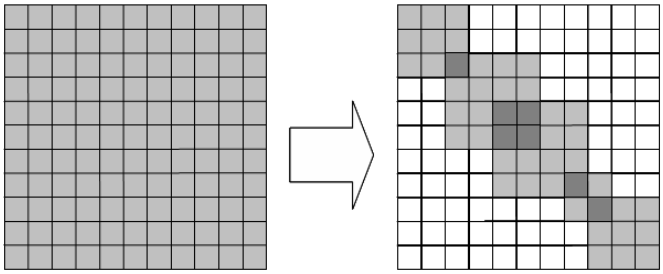

Рис. 5. Пример приведения заполненной матрицы к блочно-диагональному виду

Чаще всего на практике при моделировании физических процессов (в частности, физиологических) встречаются именно такие типы задач.

Отдельной особенностью подхода является декомпозиция задачи на подзадачи. В некоторых случаях, например, для систем линейных обыкновенных дифференциальных уравнений декомпозиция может быть выполнена автоматически программно-математическими средствами [4]; случае с нелинейными системами декомпозиция может опираться на линеаризованную систему.

Аналитические оценки погрешности

Применение вышеописанных алгоритмов синхронизации приводит к появлению дополнительной погрешности. Для квазипоследовательного алгоритма (см. рис. 2) и для системы из двух уравнений (с «медленной» переменной x и «быстрой» y)

$$\begin{cases} \dot{x} = f_x \equiv Ax + By, \\ \dot{y} = f_y \equiv Cx + Dy \end{cases}$$

погрешность ниже оценивается 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). Оценки погрешности 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 планировщика задач в операционной системе и т. п.). Эти погрешности определяются не аналитически, а экспериментально.

В случае применения явного метода Эйлера погрешности алгоритма на одном шаге расчета по компонентам x и y, соответственно, равны

$$\mathrm{E}_x = g(k)\tau^2\left(Af_x + Bf_y\right),\ \ \mathrm{E}_y = g(k)\tau Cf_x,\ \text{где}\ \ g(k) = k(k-1)/2,$$

а погрешности самого метода Эйлера равны

$$\varepsilon_x = \frac{\tau_x^2}{2}\left(Af_x + Bf_y\right) = \frac{k^2\tau^2}{2}\left(Af_x + Bf_y\right),\ \ \varepsilon_y = \frac{\tau^2}{2}\left(Cf_x + Df_y\right)$$

(заметим, что это погрешности при разных шагах — при тех, которые считаются приемлемыми для численного решения соответствующих уравнений).

Отсюда следует, что погрешность алгоритма E_x (с точки зрения естественного для переменной x шага τ_x) при всех k меняется от $\varepsilon_x/2$ до ε_x, т. е. ее 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: $\alpha_{xy} \ll 1$. Возьмем модуль вышеприведенных разностей и определим предельную кратность шага k, при которой погрешность алгоритма не превышает погрешность метода,

$$\frac{k^2}{2}\tau^2\left|Cf_x\right| \approx g(k)\tau^2\left|Cf_x\right|_x \le \frac{\tau^2}{2}\left|Cf_x + Df_y\right| \approx \frac{\tau^2}{2}\left|Df_y\right| \ \Rightarrow\ k \le k^* = \sqrt{\left|\frac{Df_y}{Cf_x}\right|}.$$

Заметим, что для не-колебательных систем введенное выше понятие «быстрых» и «медленных» подсистем означает, что отношение $\left|f_y/f_x\right|$ больше единицы (для колебательных это подразумевает другое — различие периодов). Обычно кратность шагов делают пропорциональным или просто равным отношению производных $\left|f_y/f_x\right|$ (обратному отношению характерных времен). В этом случае получаем $k^* = |D/C|$. Но независимо от того, связан ли выбор параметра алгоритма k с правой частью системы $\left|f_y/f_x\right|$, слабая связанность подсистем ($|C/D| \ll 1$) дает возможность полагать k достаточно большим (что экономит вычислительные ресурсы).

Применительно к достаточно широкому классу численных методов несложно сделать аналогичные оценки и для некоторого (не-колебательного) класса слабосвязанных систем показать, что погрешность «быстрой» переменной, обусловленная предлагаемым в работе подходом, для не слишком больших k имеет тот же порядок, что и погрешность численного метода (а погрешность «медленной» переменной всегда имеет тот же порядок).

Однако даже если погрешность численного метода существенно меньше погрешности алгоритма синхронизации (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 неопределенность исходных данных. Поэтому именно с этой неопределенностью в каждом случае имеет смысл сопоставлять погрешность предлагаемого подхода.

Результаты вычислительных экспериментов

С использованием системы интеграции были изучены особенности подхода, основанного на использовании алгоритма синхронизации, с точки зрения его технических и вычислительных свойств.

Технический анализ алгоритма проводился на модельных задачах. В первую очередь уделялось внимание скоростным характеристикам. В частности, было установлено, что при некоторых постановках задач и особенностях операционной среды эффективность решения для частично распараллеливающегося варианта (см. рис. 4) не уступает решению для целиком распараллеливающегося варианта (см. рис. 3). При этом погрешность численного решения должна убывать с уменьшением степени распараллеливания. Ниже приведен пример для случая исследования влияния временных задержек при синхронизации процессоров. Данный вопрос является актуальным в случае применения сетевых технологий при решении задач больших размерностей.

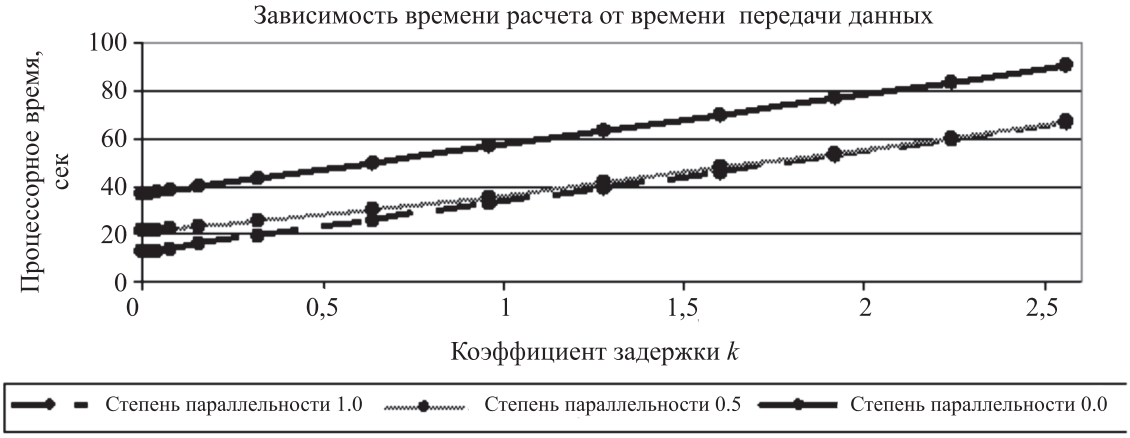

Рис. 6. Влияние временных задержек передачи данных между процессорами на итоговое время расчета для различных вариантов распараллеливания, $T_{задержка} = T_{выполнение\_шага} \cdot k$

Некоторые вычислительные особенности подхода продемонстрируем на примере жесткой задачи Коши для слабосвязанной системы обыкновенных дифференциальных уравнений

$$\begin{cases} \dfrac{dx}{dt} = -10x + 10\sin\left(40t\right)^2, \\ \dfrac{dy}{dt} = 0.01x - 100y, \end{cases} \quad \begin{cases} x(0) = 1, \\ y(0) = 1. \end{cases}$$

Такая система уравнений может описывать систему из двух химических реакторов с малой скоростью перетока вещества и периодической функцией источника массы в первом реакторе. Переменная x является «медленно переменной», y — «быстрой».

Расчет системы проводился методами Эйлера и Рунге–Кутты 4-го порядка при разных степенях кратности шагов для подсистем. Особый интерес представляет изучение влияния «медленной переменной» на «быструю».

На графиках (рис. 7) представлен рост погрешности предлагаемого метода при увеличении степени кратности. «Периодичность» погрешности — влияние характера функции источника для «медленной» переменной. Следует обратить внимание на неэкспоненциальный характер роста погрешности, обусловленной степенью кратности шагов, а также несущественное влияние порядка численного метода на погрешности решения. Из последнего можно заключить, что результаты аналитических выкладок для методов первого порядка можно распространить на методы более высоких порядков.

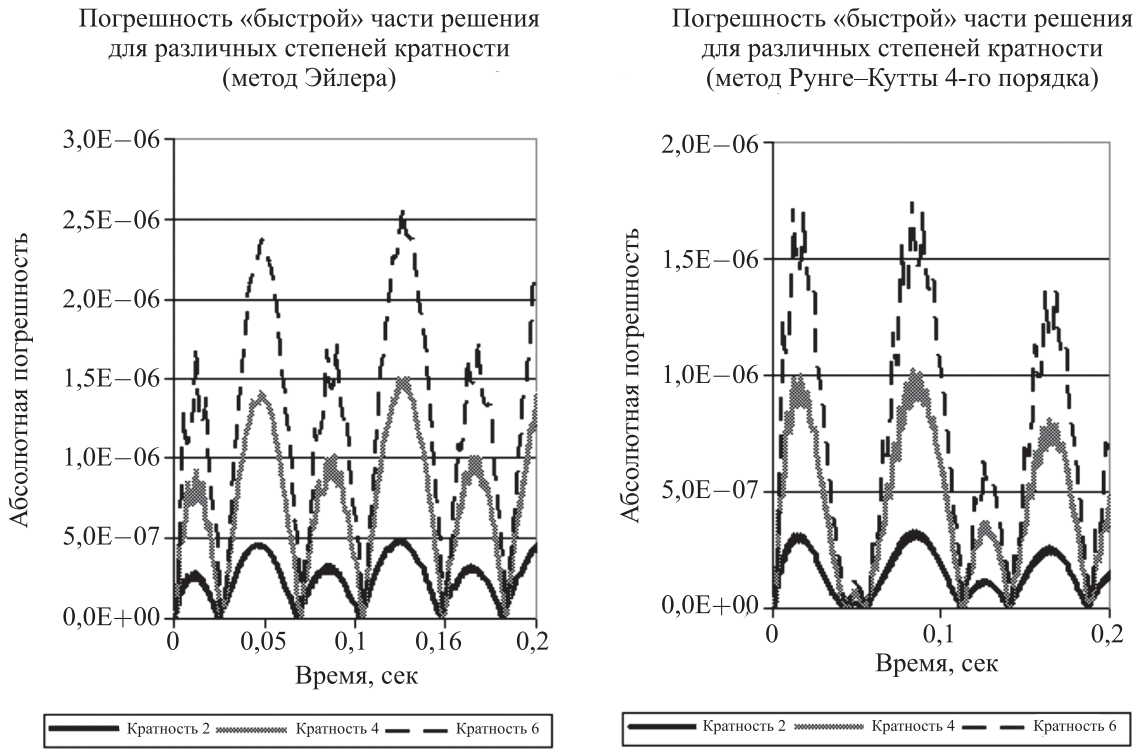

Рис. 7. Влияние кратности шага на погрешность численного решения для «быстрой» части с использованием методов Эйлера и Рунге–Кутты 4-го порядка

Заключение

В работе проанализирована вычислительная система для динамических задач, поддерживающая гетерогенные составные части — программные модули на разных языках; различные по своему математическому классу модели; а для однородных систем обыкновенных дифференциальных уравнений — подсистемы с разными характерными временами. Предложено решение проблемы интеграции гетерогенных моделей путем их независимого расчета на некотором интервале времени, причем применение специальных (в той или иной степени распараллеливающихся) алгоритмов синхронизации моделей ценой некоторой потери точности позволит повысить скорость вычислений.

Аналитические выкладки в частном случае (метод Эйлера, неколебательная система из двух уравнений) показали корректность применения предложенного подхода (по крайней мере, квазипоследовательного алгоритма синхронизации) с точки зрения погрешности вычислений. Как и следовало ожидать, полученная максимально допустимая (без потери точности) кратность шагов падает как с ростом степени связанности подсистем (коэффициента влияния «медленной» подсистемы

на «быструю»), так и с уменьшением отношения характерных скоростей «быстрой» и «медленной» подсистем.

Результаты численных экспериментов с системой химических реакторов подтверждают вывод о корректности предложенного подхода, в том числе для более сложных численных методов.

Список литературы

1. *Бенькович Е. С., Колесов Ю. Б., Сениченков Ю. Б.* Практическое моделирование динамических систем. СПб.: БХВ-Петербург, 2002. 464 с.
2. *Andrus J. F.* Automatic Integration of systems of second-order ODE's separated into subsystems // SIAM Journal on Numerical Analysis. 1983. Vol. 20, № 4. P. 815–827.
3. *Siddharth S. S., Edward J. H., Laurent O. J.* Dual-Rate Integration Using Partitioned Runge–Kutta Methods for Mechanical Systems with Interacting Subsystems // Mechanics Based Design of Structures and Machines. 2004. Vol. 32, № 3. P. 253–282.
4. *Базилевич Ю. Н., Коротенко Л. М., Швец И. В.* Численное решение задач иерархической декомпозиции линейных математических моделей // Комп'ютерне моделювання: Міждержавна науково-методична конференція. Дніпродзержинськ: ДДТУ, 2001. С. 45–46.