ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ, ОПИСЫВАЕМЫХ ОБЫКНОВЕННЫМИ ДИФФЕРЕНЦИАЛЬНЫМИ УРАВНЕНИЯМИ

Фельдман Л.П.
Кафедра ПМиИ ДонГТУ
E-mail: feldman@.donetsk.ua

Abstract

Feldman L. Parallel algorithms of modeling of dynamic systems described by the ordinary differential equations. *Are offered and the parallel algorithms of modeling on SIMD computers of dynamic systems with the concentrated parameters are investigated. The estimations of labour input, acceleration and efficiency of the offered algorithms are received, in view of time of transfer of the information between processors in structure realizing model.*

Введение

Моделирование сложных динамических систем с сосредоточенными параметрами требует обычно решения систем обыкновенных дифференциальных уравнений высокого порядка. Сокращение времени решения и, следовательно времени моделирования, возможно при использовании мультипроцессорных систем, эффективность работы которых существенно зависит от характеристик применяемых параллельных алгоритмов. Численное решение задачи Коши для системы линейных обыкновенных дифференциальных уравнений с постоянными коэффициентами

$$\frac{d\bar{x}}{dt} = A\bar{x} + \bar{f}(t),\ \bar{x}(t_0) = \bar{x}^0 = (x_1^0, x_2^0, \ldots, x_m^0)^T \qquad (1),$$

здесь

$\bar{x} = \begin{pmatrix} x_1 \\ x_2 \\ . \\ x_m \end{pmatrix}$ – вектор неизвестных, $\bar{f}(t) = \begin{pmatrix} f_1(t) \\ f_2(t) \\ . \\ f_m(t) \end{pmatrix}$ – заданный вектор

$A = \begin{pmatrix} a_{11}\ a_{12}\ \ldots\ a_{1m} \\ a_{21}\ a_{22}\ \ldots\ a_{2m} \\ \ldots\ldots\ldots\ldots \\ a_{m1}\ a_{m2}\ \ldots a_{mm} \end{pmatrix}$ – матрица коэффициентов системы

дифференциальных уравнений.

можно получить последовательно по шагам с помощью, например, следующей формулы Рунге-Кутты

$$\bar{x}^{n+1} = \bar{x}^n + \frac{1}{6}(\bar{k}_1^n + 2\bar{k}_2^n + 2\bar{k}_3^n + \bar{k}_4^n),\quad n = 0,1,2,\ldots,$$

$$\text{где}\quad \bar{k}_1^n = \tau\,[A\bar{x}^n + \bar{f}^n],\ \bar{k}_2^n = \tau\,[A(\bar{x}^n + 0.5\bar{k}_1^n) + \bar{f}^{n+0.5}], \qquad (2)$$

$$\bar{k}_3^n = \tau\,[A(\bar{x}^n + 0.5\,\bar{k}_2^n) + \bar{f}^{n+0.5}],\ \bar{k}_4^n = \tau\,[A(\bar{x}^n + \bar{k}_3^n) + \bar{f}^{n+1}],$$

τ – *выбранный шаг интегрирования.*

Ниже рассматриваются алгоритмы распараллеливания вычислений, определяемых формулами (2), на процессорах SIMD структуры при использовании квадратной сетки,

содержащей m*m процессорных элементов (ПЭ). На такой сетке эффективно выполняются матричные операции.

1. Алгоритм параллельного моделирования систем однородных уравнений

На выбранной сетке сложение двух матриц выполняется за время $t_{сл}$ сложения двух чисел в каждом из ПЭ. Умножение двух матриц по систолическому алгоритму требует предварительного косого сдвига левого сомножителя по строкам и правого- по столбцам. На это требуется 2(m-1) одиночных сдвигов. Затем надо выполнить m умножений, m-1 сложений по элементно и m-1 одиночный сдвиг. Таким образом умножения двух матриц выполняется за время равное

$$T_{sys}=m*t_{ум}+(m-1)*t_{сл}+3(m-1)*t_{сд}. \quad (3)$$

Вычисление компонент вектора $\bar{y} = A\bar{x}$, равного произведению матрицы и вектора, может быть выполнено по систолическому алгоритму. Вначале матрицу A сдвигают косо по строкам влево так, что она располагается в ПЭ таким образом, что элементы j строки сдвигаются циклически влево на (j-1) позицию. . На этом этапе требуется (m-1) одиночных сдвигов. Заменим вектор $\bar{x}$ матрицей X такой, что каждый последующий столбец, начиная со второго получается циклическим сдвигом предыдущего столбца на одну позицию вверх. На это потребуется (m-1) сдвиг на заполнение матрицы X вектором $\bar{x}$ и (m-1) одиночных сдвигов для косого сдвига этой же матрицы. Теперь, если перемножить поэлементно оба массива $A_L.*X_R$, то получим следующий массив A_LX_R,

$$A_L=\begin{pmatrix} a_{11}\ a_{12}\ \ldots a_{1m} \\ a_{22}\ a_{23}\ \ldots a_{21} \\ \ldots\ldots\ldots \\ a_{mm}\ a_{m1}\ \ldots a_{mm-1} \end{pmatrix} \qquad X_R=\begin{pmatrix} x_1\ x_2\ \ldots x_m \\ x_2\ x_3\ \ldots x_1 \\ \ldots\ldots\ldots \\ x_m\ x_1\ \ldots x_{m-1} \end{pmatrix} \qquad A_LX_R=\begin{pmatrix} a_{11}x_1\ a_{12}x_2\ \ldots\ a_{1m}x_m \\ a_{22}x_2\ a_{23}x_3\ \ldots\ a_{21}x_1 \\ \ldots\ldots\ldots\ldots\ldots \\ a_{mm}x_m\ a_{m1}x_1\ \ldots a_{mm-1}x_{m-1} \end{pmatrix}$$

в котором сумма элементов i строки равна i-ой компоненте y_i вычисляемого вектора. Все компоненты вектора $\bar{y}$ вычислим параллельно по алгоритму сдваивания. Сначала попарно сложим соседние элементы в каждой из строк, затем опять попарно сложим полученные значения, отстоящие друг от друга на две позиции, и так до тех пор, пока в каждой из строк все ее элементы станут равными вычисляемому значению соответствующей компоненты вектора $\bar{y}$. Этот процесс потребует $\log_2 m$ (берем ближайшее наибольшее целое к этому числу, равное l) сложений и столько же сдвигов. Если учесть, что сдвиги выполняются соответственно на $1, 2, 2^2, \ldots, 2^{l-1}$ позиций, то общее число одиночных сдвигов равно $N_{сд}= 2^l-1=\overline{m}$. Время, затрачиваемое на вычисление компонент вектора $\bar{y}$, может быть выражено следующей формулой

$$T_{\bar{y}} = t_{сл}*(m-1)+t_{ум}+t_{сл}*\log_2\overline{m}+t_{сд}*(\overline{m}-1) \quad (4)$$

В формуле (4) первое слагаемое определяет время косого сдвига матрицы X, а последнее- время сдвигов при сложении по алгоритму сдваивания по строкам. Времена косого сдвига матрицы A и заполнения матрицы X не вошли в (4), так как эти операции являются подготовительными и выполняются только один раз в начале решения основной задачи. Приведем другой алгоритм вычисления компонент вектора $\bar{y} = A\bar{x}$, в котором исходная матрица A транспонируется, а матрица X формируется из вектора $\bar{x}$ по столбцам без сдвигов. Если перемножить поэлементно оба массива $A^T.*X$, то получим

$$X = \begin{pmatrix} x_1\ x_1 \ldots x_1 \\ x_2\ x_2 \ldots x_2 \\ \ldots\ldots\ldots \\ x_m\ x_m \ldots x_m \end{pmatrix}. \qquad A^T.*X = \begin{pmatrix} a_{11}x_1 & a_{21}x_1 \ldots a_{m1}x_1 \\ a_{12}x_2 & a_{22}x_2 \ldots a_{m2}x_2 \\ \ldots\ldots\ldots\ldots\ldots \\ a_{1m}x_m & a_{2m}x_m \ldots a_{mm}x_m \end{pmatrix}.$$

В соответствии с методом Рунге-Кутты (2) каждый следующий шаг состоит в повторении умножения исходной матрицы A на вектор, полученный на предыдущем шаге. Поэтому для продолжения решения полученную матрицу, каждая строка которой равна вектору $\bar{y}^T$, необходимо транспонировать. Матрицу можно транспонировать, если сделать вначале косой сдвиг влево по строкам, а затем косой сдвиг вниз по столбцам. В первом алгоритме требовался только один косой сдвиг матрицы по столбцам, откуда следует, что он менее трудоемок.

Вычисление значения вектора неизвестных $\bar{x}^{n+1}$ по формуле (2) требует предварительного определения значений $k_i, i=\overline{1,4}$ Эти вектора, как это следует из формул (2), должны вычисляться последовательно. Перепишем их для однородной системы в виде удобном для оценки времени выполнения вычислений

$$\bar{k}_1 = t\ A\bar{x}_{on},\quad \bar{k}_2 = \bar{k}_1 + 0.5t\ A\bar{k}_1,\quad \bar{k}_3 = \bar{k}_1 + 0.5t\ A\bar{k}_2,\quad \bar{k}_4 = \bar{k}_1 + t\,A\bar{k}_3 \qquad (5)$$

При вычислении каждого $\bar{k}_i$ необходимо умножить матрицу на вектор и сложить с другим вектором, поэтому используя (4), получим для времени вычисления всех $\bar{k}_i$ формулу

$$T_{\bar{k}} = 4T_{\bar{y}} + 4t_{ум} + 3t_{сл}. \qquad (6)$$

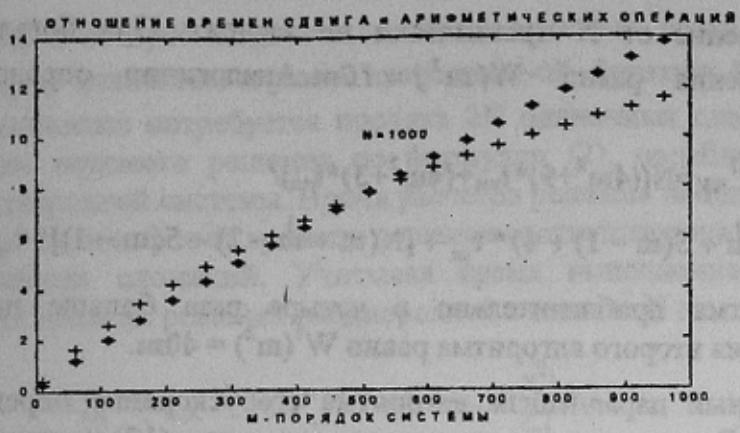
Теперь можно определить время необходимое для выполнения вычислений на одном шаге $T_n = T_{\bar{k}} + t_{ум} + 4t_{сл}$. Время решение задачи, содержащей N шагов, будет равно $T_{RK} = t_{сд} * (m-1) + NT_n$. Здесь первое слагаемое учитывает время подготовительных операций при начальном косом сдвиге матрицы A. Если подставить найденные выше значения, то последнюю формулу можно будет записать в виде

$$T_{RK}=9N*t_{ум}+N(4\log_2 m + 7) * t_{сл} + [4N(m + m - 2) + m - 1] * t_{сл}. \qquad (7)$$

В последней формуле явно выделены полные времена, затрачиваемые на каждый вид операций, за N тактов решения задачи. Значительная часть времени затрачивается на выполнение сдвигов. Если принять, для простоты оценки, что $t_{ум}=t_{сл}$ и обозначить через $z= t_{сд} /t_{ум}$ то приводимый ниже график (точки обозначены знаком *) зависимости отношения времени сдвигов ко времени арифметических операций при z=0.1 от порядка решаемой системы m, который показывает, что на выполнение сдвигов затрачивается значительно больше времени, чем на вычисления.

Так для системы сотого порядка это отношение равно 2, а для системы, порядок которой равен 1000, это отношение равно 14. При решении систем высокого порядка, несмотря на массовый обмен данными, реализованный в системах SIMD, он определяет время решения задачи, ограничивая вычислительные возможности системы.

Рассмотрим алгоритм, который имеет меньшее число сдвигов. Для этого выразим на очередном шаге полный оператор, позволяющий определить значение вектора неизвестных на следующем шаге. Назовем его оператором (матрицей) перехода. Используя формулы (5),



выразим k_i для $i=2,3,4$ через k_1 , подставим найденные выражения в (2), получим для однородной системы

$$\bar{x}_{од}^{\,n+1} = (E + \tau A(E + \frac{\tau A}{2}(E + \frac{\tau A}{3}(E + \frac{\tau A}{4})))) \bar{x}_{од}^{\,n},\ E - \textit{единичная матрица}. \qquad (8)$$

Матрицу перехода следует определить один раз до выполнения вычислений по формуле (8). На это потребуется три умножения матриц на скаляр, четыре последовательных сложения полученных матриц с единичной матрицей и три матричных умножения. Умножение матриц будем выполнять по систолическому алгоритму, учитывая, что в каждом из трех матричных произведений используется один и тот же левый сомножитель равный τA. Время, затраченное на выполнение предварительных операций вычисления матрицы перехода, представляется следующей формулой $T_{под}=3T_{sys}+4t_{сл}+3t_{ум}$.

Теперь для вычисления искомого вектора $\bar{x}^{n+1}$ надо умножить матрицу перехода на вектор $\bar{x}^{n}$, что потребует, в соответствии с (4), времени равного T_y . Общее время решения задачи по рассмотренному алгоритму составит $T^{1}_{RK} = T_{под} + N*T_y$.

Подставив выражения для входящих сюда $T_{под}$ и T_y получим окончательно

$$T^{1}_{RK} = (N+3m+3)*t_{ум} + (N\log_2 m + 3(m-1)+4)*t_{сл} + [N(m+m-2)+5(m-1)]*t_{сл} \qquad (9)$$

Сравнивая с (7) можно заключить, что хотя время выполнения подготовительных операций этого алгоритма больше времени алгоритма, рассмотренного выше, однако при больших m и N это не имеет существенного значения, так как доля их в общем объеме вычислений пренебрежимо мала. Существенным здесь является то, что в общем объеме вычислений в последнем алгоритме число сдвигов в 4 раза меньше в сравнении с первым алгоритмом. Потому он будет решать задачу быстрее не менее чем в четыре раза. Для второго алгоритма также характерным является то, что на выполнение сдвигов затрачивается значительно больше времени, чем на вычисления (точки обозначены на графике знаком +).

Время выполнения всех N тактов решения задачи по методу Рунге-Кутты на однопроцессорном компьютере может быть определено по формуле

$$T^{1}_{пос}=N((4m^2+9)*t_{ум}+(4m^2+3)*t_{сл}). \qquad (10)$$

Ускорение первого, из рассмотренных выше алгоритмов использующего m^2 процессоров, выражается следующей формулой

$$W(m^2)=T^{1}_{вос}/T_{RK}=N((4m^2+9)*t_{ум}+(4m^2+3)*t_{сл})/$$
$$(9N*t_{ум}+N(4\log_2 m+7)*t_{сл}+[4N(m+m-2)+m-1]*t_{сд}). \qquad (11)$$

Из нее следует, что ускорение от N практически не зависит. При больших m асимптотическое значение ускорения равно $W(m^2) = 10m$. Аналогично определяется ускорение для второго алгоритма

$$W^1(m^{2)}=T^1_{вос}/T^1_{RK}=N((4m^2+9)*t_{ум}+(4m^2+3)*t_{сл})/$$

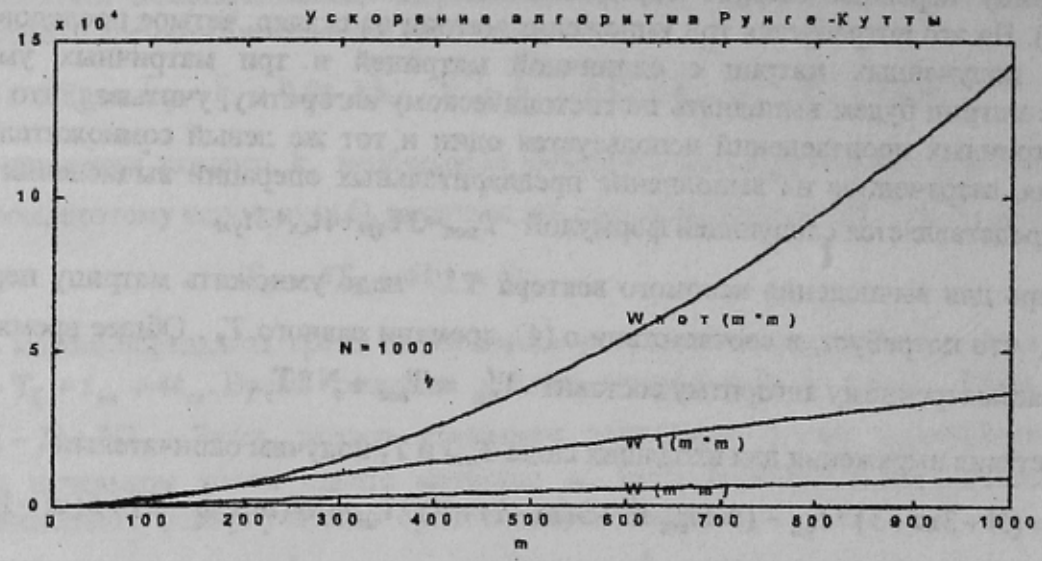
$$((N+3m+3)*t_{ум}+(N\log_2 m+3(m-1)+4)*t_{сл}+[N(m+m-2)+5(m-1)]*t_{со}) .(12)$$

Ускорение второго алгоритма приблизительно в четыре раза больше первого. Асимптотическое значение ускорения второго алгоритма равно $W^1(m^2) = 40m$.

Чтобы оценить потенциальный параллелизм алгоритма его ускорение определяют обычно не учитывая время сдвигов. В нашем случае для второго алгоритма (12) получим

$$W^1_{пот}(m^{2)}=T^1_{пос}/T^1_{RK}=N((4m^2+9)*t_{ум}+(4m^2+3)*t_{сл})/((4+N)*t_{ум}+(N\log_2 m+3)*t_{сл}). \quad (13)$$

Ниже приведены для сравнения графики ускорений для параллельного алгоритма без учета времени на сдвиги $W_{пот}$ первого и второго параллельных алгоритмов формулы (10 и 11). Асимптотическое значение для потенциального алгоритма равно $W_{пот}=8m^2/\log_2 m$.



2. Алгоритм параллельного моделирования систем неоднородных уравнений

При решении неоднородной системы уравнений по формуле Рунге-Кутты (2) необходимо, дополнительно к рассмотренным выше расчетам по формулам (5), вычислить на каждом шаге n также и значения всех функций $f_i(t)$ в трех точках. В общем случае все эти функции могут быть различными, поэтому одновременное параллельное их вычисление на SIMD процессоре невозможно. Однако расчет значений одной, любой из них, может быть выполнен параллельно для всех значений аргумента t_n, $t_{n+0.5}$, $n = \overline{0,N}$. Поскольку для произвольной функции $f_i(t)$ нельзя определить возможность распараллеливания вычислений ее значения для заданного значения аргумента, то обозначим через σ_i время последовательного расчета ее значения для одного значения аргумента на ПЭ. Вычисления таблицы значений всех функций $f_i(t)$, $i = \overline{1,m}$, для всех 2N значений аргумента t потребует на линейке из 2N ПЭ времени равного сумме всех σ_i, т.к. функции вычисляются последовательно. Эти вычисления выполняются до начала решения задачи по формулам (2).Если ПЭ достаточно, то для каждой функции можно выделить свою линейку. Затем на сетке m*m ПЭ сформируем из полученной таблицы значений функций f_i^n матрицу F(n),

элементы $F_{i,j}(n)$ которой являются линейным массивом длины 2N. Все элементы i-ой строки матрицы $F_{i,j}(n)$ одинаковы и равны значению i-ой функции f_i^n для $t=t_n$, т.е. $f_i^n = F_{i,j}(n)$. На эти преобразования потребуется порядка 2N одиночных сдвигов. Теперь можно перейти к вычислениям искомого решения по формулам (2), подобно тому, как это делалось при решении однородной системы. Время расчетов решения неоднородной системы на очередном шаге n будет отличаться от времени решения соответствующей однородной системы на время трех матричных сложений. Учитывая время выполнения подготовительных операций, получим для времени решения неоднородной задачи за N шагов следующую формулу

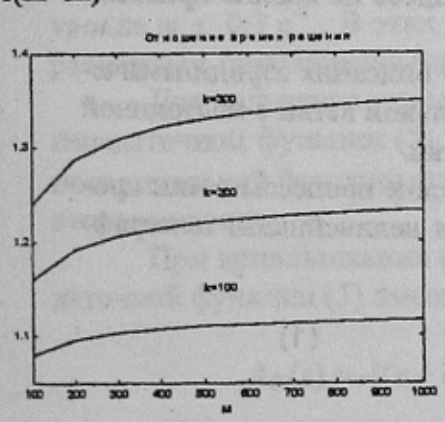
$$T_{RK}^{н} = T_{RK} + 3t_{сл} * N + 2N * t_{сл} + \sum_{i=1}^{m} \sigma_i, \qquad (14)$$

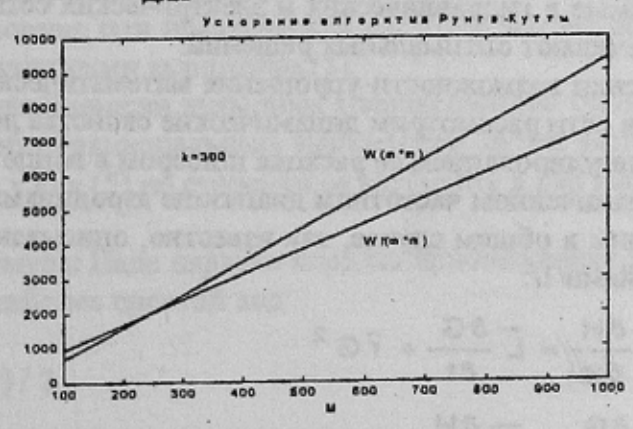
где T_{RK} определяется формулой (7). Выполнение алгоритма, определяемого формулами (2), на однопроцессорном компьютере потребует времени равного

$$T_{пос}^{н} = N(4m(m+8) * t_{ум} + 4m(m+7) * t_{сл} + 2\sum_{i=1}^{m} \sigma_i). \qquad (15)$$

Заключение

Оценим влияние времени вычисления значений функций $f_i(t)$ в зависимости от их сложности на время решения задачи. Ниже приведен график отношения времен решения неоднородной и однородной систем дифференциальных уравнений в зависимости от сложности вычисления функций $f_i(t)$. Для расчетов было принято, что все $f_i(t)$ имеют одинаковую сложность, которая выражается числом операций равным k для каждой функции. Из графиков видно, что рассмотренный выше алгоритм предварительного параллельного расчета функций $f_i(t)$ незначительно увеличивает время решения неоднородной системы по сравнению со временем решения однородной системы. На следующем рисунке приведены графики ускорения решения однородной W(m*m) и соответствующей неоднородной систем Wf(m*m) для k=300.





Таким образом предложенные алгоритмы позволяют существенно ускорить процесс моделирования.

Литература

1. .Самарский А.А., Гулин А.В. Численные методы.-М.:Наука,1989.
2. .Фельдман Л.П.,Труб И.И. Эффективность параллельных алгоритмов численного решения краевых задач с частными производными при их реализации на SIMD компьютерах. Информатика, кибернетика и вычислительная техника. Сб. Научн. Тр. Донецкого государственного технического университета. Вып. 1.Донецк: ДонГТУ, 1997, с 26-34.