УДК 519.684.4

М.И. Горбийчук, В.М. Медведчук, А.Н. Лазорив

АНАЛИЗ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА СИНТЕЗА ЭМПИРИЧЕСКИХ МОДЕЛЕЙ НА ПРИНЦИПАХ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Введение

Построение эмпирических моделей занимает важное место как в научной, так и в практической деятельности. Такие модели используются в различных областях деятельности человека — прогноз, распознавание образов, оптимизация и т. п.

Суть задачи построения эмпирических моделей в том, что имеются некоторые входы и выходы, между которыми существует определенная функциональная зависимость, но она исследователю неизвестна. В лучшем случае можно указать некоторый класс функций, которому принадлежит такая функциональная зависимость.

Возможны два принципиально разных случая. Первый, когда входы наблюдаются без искажений, а на выходы накладывается аддитивная помеха. В этом случае для построения эмпирической модели применим метод наименьших квадратов (МНК) [1]. Классический МНК основан на максимизации функции правдоподобия при условии, что закон распределения помехи, накладываемой на выход, носит нормальный характер.

Во втором случае вход и выход наблюдаются на фоне помех. Учет помех, которые накладываются на переменные вход–выход, при построении эмпирических моделей возможен, если использовать байесовский подход [2, 3].

Важное место в современной теории построения эмпирических моделей занимает метод опорных векторов [4]. Его нелинейная версия носит название ядерный метод опорных векторов. Рассматривать его следует как один из методов математической статистики [4]: его линейная версия близка к методу гребневой регрессии [5], а нелинейная версия — к методу непараметрического ядерного оценивания [6].

Метод непараметрического ядерного оценивания успешно конкурирует с нейросетевым подходом [4], когда функциональная зависимость между парой вход–выход — реакция обобщенной регрессионной нейросети [7], вход которой представлен как множество векторов, компоненты которых — экспериментальные данные.

Во всех методах построения эмпирических моделей открытым остается вопрос о выборе структуры модели. В классическом МНК предполагается, что структура модели известна и задача сводится к отысканию функциональной зависимости вход–выход. Если модель линейна относительно своих параметров, задача МНК имеет аналитическое решение.

В общем случае синтез эмпирических моделей осуществляется, исходя из условия, что известен класс функций, и ставится задача выбрать такую функцию из заданного класса, которая в определенном смысле лучше всего описывает экспериментальные данные.

Для решения поставленной задачи акад. А.Г. Ивахненко предложил метод, основанный на теореме Геделя [8]. Полученные методы известны как метод группового учета аргументов (МГУА) и комбинаторный метод [9, 10].

МГУА в качестве аргументов содержит некоторые промежуточные величины, порожденные многорядной его природой, а комбинаторный метод для своей реализации требует значительных вычислительных затрат [11].

Цель настоящей работы — дальнейшее усовершенствование индуктивного метода самоорганизации эмпирических моделей на основе генетического подхода, а также исследование полученного алгоритма на параллелизм, что позволит создать эффективные алгоритмы построения эмпирических моделей.

Метод синтеза моделей оптимальной сложности на принципах генетических алгоритмов

В основе эмпирического моделирования многих явлений и процессов лежит широко известный МНК, содержание которого заключается в следующем. Пусть рассматривается некоторый объект или система, функционирование которых характеризуется вектором входных величин $\bar{x} = (x_1, x_2, \ldots x_n)^{\mathrm{T}}$ и выходной величиной y. В результате наблюдений за входными и выходными величинами получают множество значений $\bar{x}^{(1)}, \bar{x}^{(2)}, ..., \bar{x}^{(N)}$ и $Y^{(1)}$, $Y^{(2)}$, …, $Y^{(N)}$, где N — количество наблюдений. Совокупность векторов $\bar{x}^{(1)}, \bar{x}^{(2)}, ..., \bar{x}^{(N)}$ образует так называемую матрицу наблюдений:

$$X = [\bar{x}^{(1)}, \bar{x}^{(2)}, ..., \bar{x}^{(N)}]^{\mathrm{T}}.$$

В МНК допускают, что структура модели известна и чаще всего выбирается линейной относительно ее параметров:

$$y = \sum_{k=0}^{r} a_k f_k(\bar{x}), \tag{1}$$

где a_k, $k = \overline{0, r}$, — параметры модели.

Задача заключается в том, чтобы на основе наблюдений за входными и выходной величинами, определить параметры, a_k, $k = \overline{0, r}$, модели (1) таким образом, чтобы как можно точнее приблизить выход системы к выходу модели. Критерием такого приближения выбирают сумму квадратов отклонений расчетных $y^{(i)}$ от экспериментальных значений, $Y^{(i)}$, $i = \overline{1, N}$.

По известным $Y^{(i)}$ и вычисленным значениями $y^{(i)}$ согласно (1) можно получить критерий приближения (аппроксимации) $J(\bar{a}) = \sum_{i=1}^{N} (Y^{(i)} - y^{(i)})^2$, который в векторной форме приобретет следующий вид:

$$J(\bar{a}) = (\bar{Y} - F\,\bar{a})^{\mathrm{T}} (\bar{Y} - F\,\bar{a}), \tag{2}$$

где $\bar{Y}$ — вектор наблюдений с компонентами $Y^{(i)}$, $i = \overline{1, N}$; F — матрица, строки которой — значения функций $f_i(\bar{x}^{(j)})$, $i = \overline{0, r}$, $j = \overline{1, N}$, в точках наблюдений за величинами $\bar{x}^{(j)}$.

Значения параметров модели (1) вычисляют из условия, что критерий приближения (2) приобретет минимальное значение относительно вектора параметров $\bar{a}$. Минимизация (2) приводит к следующему результату:

$$M_F \bar{a} = F^{\mathrm{T}} \bar{Y}, \tag{3}$$

где $M_F = F^{\mathrm{T}} F$ — матрица Фишера.

Если размерность вектора $\overline{a}$ небольшая и матрица M хорошо обусловлена, из уравнения (3) можно непосредственно определить $\overline{a} = M_F^{-1} F^{\mathrm{T}} \overline{Y}$.

В подавляющем большинстве случаев структура модели (1) неизвестна, что приводит к необходимости выбора как структуры самой модели, так и ее параметров. Критерий приближения (2) является внутренним критерием [10], и его приложение приводит к ошибочному выводу: чем сложнее модель, тем она точнее. Поскольку на выход системы накладывается помеха (допускают, что она аддитивна), то чрезмерная точность модели может значительно исказить объективно существующую функциональную зависимость между выходом системы и ее входами.

Поэтому для выбора структуры модели (1) акад. А.Г. Ивахненко предложил индуктивный метод самоорганизации моделей [10], идейную сторону которого определяет теорема Геделя [8]. Относительно задачи синтеза модели геделевский подход означает применение внешнего критерия для однозначного выбора структуры модели из заданного класса моделей. Критерий называют внешним, если его вычисление основывается на данных, которые не использовались при решении задачи (2). Это значит, что все данные, полученные в результате эксперимента, разбиваются на две части: учебную N_A и проверочную N_B.

Преимущественно для выбора структуры модели (1) используют критерии регулярности

$$\Delta^2(B) = \frac{\sum_{i=1}^{N_B} (Y^{(i)}(B) - y^{(i)}(B))^2}{\sum_{i=1}^{N_B} Y^{(i)}(B)^2} \tag{4}$$

и смещения

$$\Delta^2(A,B) = \frac{\sum_{i=1}^{N} (y^{(i)}(A) - y^{(i)}(B))^2}{\sum_{i=1}^{N} (Y^{(i)})^2}, \tag{5}$$

где $y^{(i)}(A)$, $y^{(i)}(B)$ — значения выхода модели, которые вычислены соответственно на множествах экспериментальных значений N_A и N_B.

Если выбранный критерий регулярности (4), то выбирают такое распределение данных эксперимента [12]: $N_A = 0{,}7N$ и $N_B = 0{,}3N$, а при выборе критерия (5) — $N_A = 0{,}5N$ и $N_B = 0{,}5N$.

Реализация индуктивного метода самоорганизации моделей осуществляется поэтапно: первый этап генерирования моделей-претендентов (в определенном порядке повышения их сложности); второй — выбор наилучшей модели по минимуму критерия селекции (4) или (5).

Различают три способа генерирования моделей-претендентов. Первый из них комбинаторный [12]: выбор моделей-претендентов осуществляется приравниванием к нулю некоторых коэффициентов в выражении (1), которое дает возможность получить совокупность моделей. Выбор наилучшей из них осуществляется на основе одного из критериев селекции (4) или (5). Второй способ — МГУА [9], в котором генерирование моделей-претендентов осуществляется с помощью многорядной процедуры. Третий метод [10] подобен

второму. Разница лишь в том, что на каждом ряду селекции частичные модели образуются приравниванием к нулю определенного числа их коэффициентов.

Недостаток комбинаторного метода селекции моделей — необходимость большого перебора частичных моделей. Если начальной моделью выбран полный полином степени m, то общее число моделей-претендентов $2^M - 1$, где M — общее число членов начального полинома. Даже современные ЭВМ не способны реализовать комбинаторный алгоритм селекции моделей при значительном количестве входных величин и при высокой степени начального полинома. МГУА синтезирует модели, в которых фигурируют промежуточные переменные, что значительно усложняет процесс перехода к входным переменным системы. Сказанное относится и к третьему методу, поскольку он является модификацией МГУА.

МГУА имеет тот недостаток, что в соответствии с принятой процедурой синтеза эмпирических моделей выходные переменные достаточно сложно выразить в явном виде через входные переменные объекта.

Из всех трех методов наиболее привлекателен комбинаторный метод, поскольку он дает возможность получить оптимальную модель, где в качестве аргументов выступают входные переменные системы.

Для снятия проблемы большой размерности комбинаторного метода применим генетический подход. Как эмпирическую модель выберем полином степени m,

$$y = \sum_{i=0}^{M-1} a_i \prod_{j=1}^{n} x_j^{s_{ji}}, \qquad (6)$$

где a_i — коэффициенты полинома; s_{ji} — степени аргументов, которые должны удовлетворять ограничению $\sum_{j=1}^{n} s_{ji} \le m.$

Число членов M полинома (6) определяют согласно формуле [11]

$$M = \frac{(m+n)!}{m!\,n!}. \qquad (7)$$

Образуем упорядоченную последовательность, где на i-м месте будет находиться единица или нуль в зависимости от того, будет ли параметр a_i, $i = \overline{0, M-1}$, модели (6) отличаться от нуля или иметь значение нуль. В теории генетических алгоритмов такая упорядоченная последовательность носит название хромосомы, а атомарный элемент (единица или нуль) хромосомы — это ген; набор хромосом образует популяцию. Важным понятием теории генетических алгоритмов является функция приспособленности, которая определяет степень приспособленности отдельных особей в популяции. Она дает возможность из всей популяции выбрать особей, наиболее приспособленных, т.е. тех, которым отвечает наименьшее значение функции приспособленности. В задаче синтеза моделей оптимальной сложности как функцию приспособленности выберем критерий селекции (4) или (5).

Как и классический генетический алгоритм [13], алгоритм синтеза моделей оптимальной сложности состоит из следующих шагов.

Шаг 1. Формирование начальной популяции (инициализация). На первом шаге работы алгоритма случайным образом формулируется популяция из I особей, каждая из которых является хромосомой длиной M. Число генов в хромосоме определяется по формуле (7).

Шаг 2. Оценка приспособленности хромосомы в популяции. Для каждой хромосомы вычисляется значение критерия селекции (4) или (5) следующим образом. Если выбранный критерий селекции (4), то формируются матрицы F_A и F_B размерами $N_A \times M$ и $N_B \times M$. Из матрицы F_A изымается *i*-й столбец, если на *i*-й позиции хромосомы находится нуль; если единица, то соответствующий столбец остается без изменений. В итоге получим матрицу $\widetilde{F}_A$, из которой изъято c столбцов (по количеству нулей в хромосоме). Размер такой матрицы $N_A \times (M - c)$. Аналогично получим матрицу $\widetilde{F}_B$ размером $N_B \times (M - c)$. На множестве точек N_A вычисляются ненулевые коэффициенты a_{Aj}, $j = \overline{1, M - c}$, модели (8) путем решения нормального уравнения Гаусса, которое видоизменяется следующим образом:

$$\widetilde{M}_{F,\,A}\,\overline{a}_A = \widetilde{F}_A^{\mathrm{T}}\,\overline{Y}_A, \tag{8}$$

где $\overline{a}_A = (a_{A0}, a_{A1}, \ldots, a_{A,\,M-c-1})^{\mathrm{T}}$ — вектор ненулевых параметров модели, который ассоциируется с дежурной хромосомой; $\widetilde{M}_{F,\,A} = \widetilde{F}_A^{\mathrm{T}}\,\widetilde{F}_A$. $\overline{Y}_A = (Y^{(1)}, Y^{(2)}, \ldots, Y^{(N_A)})$ — вектор экспериментальных значений на множестве N_A.

По найденным коэффициентам $\overline{a}_A$ полиномиальной модели на множестве точек N_B вычисляют значение

$$\overline{y}(B) = \widetilde{F}_B\,\overline{a}_A. \tag{9}$$

Зная $\overline{y}(B)$, по формуле (4) вычисляют значение функции приспособленности $\Delta_j^2(B)$, $j = \overline{1, I}$, для каждой хромосомы из начальной популяции.

В том случае, когда критерий смещения (5) используют как функцию приспособленности, составляют уравнение (8), которое решают методом Гаусса относительно параметров $\overline{a}_A$. После этого вычисляют $\overline{y}(A) = \widetilde{F}_A\,\overline{a}_A$ по формуле (9). Полученные значения $\overline{y}(A)$ и $\overline{y}(B)$ дают возможность найти значение $\Delta_j^2(A, B)$, $j = \overline{1, I}$, для каждой хромосомы из популяции.

Шаг 3. Проверка условия остановки алгоритма. Определяют

$$\Delta_m^2(B) = \min_j \Delta_j^2(B), \tag{10}$$

$$\Delta_m^2(A, B) = \min_j \Delta_j^2(A, B). \tag{11}$$

Если минимальное значение (10) или (11) критерия селекции (4) или (5) не превышает некоторого заданного значения E, то происходит остановка вычислений. Она также может состояться и в случае, когда в результате выполнения алгоритма функция приспособленности уменьшается несущественно или когда выполнено заданное число итераций.

После выполнения одного из трех условий из текущей популяции выбирается та хромосома ch^*, для которой выполняется условие (10) или (11). Эта хромосома задает структуру модели оптимальной сложности и формирует матрицу F^* таким образом, что из начальной матрицы изымаются столбцы, ассоциируемые с ну-

левыми значениями соответствующих генов. Пересчет параметров модели (6) осуществляется на всем множестве точек N с помощью МНК.

Шаг 4. Селекция хромосом. По рассчитанным на втором шаге алгоритма значениям функции приспособленности осуществляется отбор хромосом, участвующих в создании потомков для новой популяции. Такой отбор проводится по принципу естественного отбора, когда наибольшие шансы в создании новой популяции имеют хромосомы с наилучшими значениями функции приспособленности (4) или (5). Самый распространенный метод селекции — метод рулетки и турнирный метод [13]. Метод рулетки можно применять тогда, когда функция приспособленности положительна, что делает его пригодным лишь для задач максимизации. Турнирный метод можно использовать как в задачах максимизации, так и в задачах минимизации.

При турнирной селекции все хромосомы разбиваются на подгруппы со следующим выбором из каждой подгруппы хромосомы с наилучшей приспособленностью. Подгруппы могут иметь произвольный размер, но чаще популяцию делят на подгруппы по 2–3 особи в каждой.

Шаг 5. Формирование новой популяции потомков. Над отобранными особями на четвертом шаге алгоритма осуществляют видоизменения с помощью двух основных операторов: скрещивание и мутации. Следует заметить, что оператор мутации применяется реже по сравнению с оператором скрещивания. Вероятность скрещивания достаточно высока ($0 \le P_c \le 1$), тогда как вероятность мутации составляет $0 \le P_m \le 0{,}1$. Вероятность мутации P_m можно задать путем выбора случайного числа из интервала $[0;\ 0{,}1]$ для каждого гена. Мутация может осуществляться как над пулом родственников, так и над пулом потомков.

Шаг 6. После выполнения оператора скрещивания возвращаемся к шагу 2.

Алгоритм распараллеливания процесса формирования эмпирической модели

Реализация генетического алгоритма синтеза эмпирических моделей оптимальной сложности показала, что значительная часть машинного времени тратится на решение системы линейных алгебраических уравнений (8).

На практике построение эмпирической модели порождает систему уравнений большой размерности [14].

Нормальное уравнение Гаусса (8) запишем в такой форме:

$$\widetilde{A}\overline{a}_A = \widetilde{b}, \tag{12}$$

где $\widetilde{A} = \widetilde{M}_{F,\,a};\ \widetilde{b} = \widetilde{F}_A^T\ \overline{Y}_A.$

Уравнение (14) будем решать методом гауссового исключения, где система (12) приводится к верхней диагональной форме по следующим формулам [15]:

$$\widetilde{a}_{i\bullet}^{(i)} = \frac{\widetilde{a}_{i\bullet}^{(i-1)}}{\widetilde{a}_{ii}^{(i-1)}},\ \ i = \overline{1,\, M-c}; \tag{13}$$

$$\widetilde{a}_{k\bullet}^{(i)} = \widetilde{a}_{k\bullet}^{(i-1)} - \widetilde{a}_{i\bullet}^{(i)}\widetilde{a}_{kj\bullet}^{(i-1)},\ \ k = \overline{i+1,\, M-c},\ \ j = \overline{1,\, M-c+1}, \tag{14}$$

где $\widetilde{a}_{i\bullet}^{(i)}, \widetilde{a}_{k\bullet}^{(i)}$ — i - и k -й строки расширенной матрицы, получаемой присоединением к матрице $\widetilde{A}$ векторного столбца $\widetilde{b}$; $\widetilde{a}_{i\bullet}^{(0)} = \widetilde{a}_{i\bullet}$, $i = \overline{1,\, M-c}$; $\widetilde{a}_{k\bullet}^{(0)} = \widetilde{a}_{k\bullet}$, $k = \overline{i+1,\, M-c}.$

В вычислительном процессе цикл по индексу k является внутренним по отношению к внешнему циклу по индексу i. Результат применения итерационных процедур (13) и (14) — верхняя диагональная матрица U с единицами на главной диагонали. Остальные элементы матрицы: $u_{ij} = a_{ij}^{(i)}$, $i = \overline{1, M-c}$, $j = \overline{i+1, M-c+1}$.

Таким образом, уравнение (12) превратится в эквивалентную систему линейных алгебраических уравнений:

$$\widetilde{U}\,\overline{a}_A = \overline{b}_A, \tag{15}$$

где $\widetilde{U}$ — квадратная матрица размером $M-c$, которая получена из матрицы U путем исключения последнего столбца: $b_{A,i} = u_{i,M-c+1}$, $i = \overline{1, M-c}$.

Уравнение (12) решается методом обратной прогонки, начиная с последнего уравнения. Очевидно, что

$$a_{A,M-c} = b_{A,M-c}. \tag{16}$$

Другие значения $a_{A,i}$ вычисляются по такой итерационной процедуре:

$$a_{A,i} = b_{A,i} - \sum_{j=i+1}^{M-c} u_{ij} a_{A,j}, \quad i = \overline{1, M-c-1}. \tag{17}$$

Вычисление параметров модели (4) по рекуррентным соотношениям (16) и (17) дает ощутимую экономию машинного времени по сравнению с классическим методом Гаусса. Сказанное справедливо и для LU — метода, в соответствии с которым матрица $\widetilde{A}$ подается в виде произведения двух матриц: L и U_R. Первая из них — нижняя диагональная с единицами на главной диагонали, а вторая — верхняя диагональная матрица. С помощью прямой и обратной прогонок решаются треугольные системы $L\overline{\gamma} = \overline{b}_A$ и $U_R\overline{a}_A = \overline{\gamma}$.

Другой способ уменьшения затрат машинного времени — распараллеливание алгоритма приведения матрицы $\widetilde{A}$ к верхней треугольной матрице U. На эту операцию тратится бо⊄льшая часть времени из расходуемого на решение системы уравнений (12).

Суть такого алгоритма в следующем. На стадии инициализации (начальной стадии) вся матрица $\widetilde{A}$ размещается в рабочем пространстве первого процессора-мастера (далее — мастера). Первый шаг — мастер отправляет, по возможности, ровные слои матрицы $\widetilde{A}$ другим процессором-рабочим (далее — рабочим) так, что каждый из процессоров, включая и первый, получает подматрицы $\widetilde{A}_i^{(1)}$, $i = \overline{0, q-1}$, где q — общее количество процессоров. После отправления мастер нормирует первую строку своей подматрицы $\widetilde{A}_0^{(1)}$ по формуле (13) и тут же отсылает значение $\widetilde{a}_{1\bullet}^{(1)}$ другим рабочим. Рабочие, получив строку $\widetilde{a}_{1\bullet}^{(1)}$, каждая из них согласно формуле (14), где $k = \overline{1, M-c}$, перечисляют элементы своих подматриц, включая и элемент $\widetilde{a}_{qi,M-c+1}$. Одновременно мастер вычисляет новые значения элементов по формуле (14), начиная со второй строки. В результате такого пересчета рабочие получают подматрицы $\widetilde{A}_i^{(1)}$, $i = \overline{1, q-1}$, в которых нулевыми будут первые столбцы. В матрице $\widetilde{A}_1^{(0)}$ первый столбец будет иметь нулевые элементы, кроме первого $\widetilde{a}_{11}^{(1)}$, равного единице.

Второй шаг — мастер нормирует вторую строку своей подматрицы и значение $\widetilde{a}_2^{(2)}$, которое вычислено по формуле (13), посылает всем рабочим. Одновременно с мастером они модифицируют свои подматрицы в соответствии с процедурой (14). Причем мастер изменяет индекс k от 3 до $M-c$, а рабочие соответственно — от 2 до $M-c$. В итоге длина второй строки с ненулевыми элементами сократилась на единицу (первый элемент равняется нулю, а второй — единице). Подматрицы, которые размещены в пространствах рабочих, будут иметь по два нулевых столбца.

Процесс модификации элементов подматриц осуществляется циклически по приведенной схеме до тех пор, пока мастер не приведет свою подматрицу к верхнему диагональному виду. В итоге после окончания первого цикла в рабочем пространстве мастера будет храниться верхняя прямоугольная подматрица с единицами на главной диагонали, а в рабочих пространствах рабочих получим подматрицы, число нулевых столбцов которых определяется числом строк матрицы мастера.

В начале второго цикла рабочие посылают свои подматрицы мастеру, где происходит их объединение. Затем мастер объединенную подматрицу разделяет на q частей и рассылает их всем рабочим, в том числе и себе. Мастер нормирует первую строку своей подматрицы. При этом ведущим элементом будет первый ненулевой элемент первой строки подматрицы мастера. Модифицированную первую строку в соответствии с формулой (13) мастер рассылает всем рабочим, которые модифицируют свои строки по формуле (14). В дальнейшем вычисления происходят так, как это было в первом цикле. Алгоритм продолжает работать до тех пор, пока размер дежурного слоя, подлежащего отправлению, не будет превышать количества процессоров. Тогда мастер приводит оставшуюся подматрицу к верхней прямоугольной матрице с единицами на главной диагонали. Завершающий этап — объединение всех матриц, сохраненных в рабочем пространстве мастера.

Вычислим количество операций на каждом шаге вычислений, которые выполняются мастером и каждым рабочим при их параллельной работе. Выполнение мастером операции нормирования первой строки подматрицы $\widetilde{A}_0^{(1)}$ в соответствии с формулой (13) требует $z = M - c$ операций деления (диагональным элементам подматрицы $\widetilde{A}_0^{(1)}$ присваивается значение единицы). На перечисление элементов остальных $s_0^{(1)}$ строк подматрицы $\widetilde{A}_0^{(1)}$ по формуле (14) будут тратить $(s_0^{(1)} - 1)z$ операций умножения и $(s_0^{(1)} - 1)z$ операций вычитания. Общее количество операций, которое выполнено мастером на первом шаге, $N_{10}^{(1)} = (2s_0^{(1)} - 1)z$.

Каждый рабочий, получив $\widetilde{A}_i^{(1)}$ подматрицу с $s_i^{(1)}$ строками, выполняет $N_i^{(1)}$ операций на пересчет элементов по формуле (14). Всего будет $s_i^{(1)}z$ операций умножения и $s_i^{(1)}z$ операций вычитания. Общее количество операций, которое выполняет i-й рабочий, $N_{1i}^{(1)} = 2s_i^{(1)}z$, $i = \overline{1, q-1}$.

На втором шаге вычислений мастер нормирует вторую строку матрицы, $\widetilde{A}_0^{(1)}$, затратив $z-1$ операций деления. Другие строки матрицы $\widetilde{A}_0^{(1)}$ перечисляются по формуле (14). При этом будет выполнено $(s_0^{(1)} - 2)(z-1)$ операций умножения и такое же количество операций вычитания. Следовательно, на втором шаге вычислений мастер выполнит $N_{10}^{(2)} = (2s_0^{(1)} - 3)(z-1)$ арифметических операций.

Поскольку в рабочем пространстве каждого i -го рабочего имеем матрицы, в которых на один столбец ненулевых элементов стало меньше, то общее количество операций деления, умножения и вычитания, выполняемых i-м рабочим, $N_{1i}^{(2)} = 2s_i^{(1)}(z-1)$, $i = \overline{1, q-1}$.

На третьем шаге вычислений мастер нормирует третью строку, выполнив $z-2$ операций деления. Начиная с четвертой строки, мастер пересчитывает все элементы подматрицы $\widetilde{A}_0^{(1)}$ по формуле (14), тратя такое количество операций деления, умножения и вычитания: $N_{10}^{(3)} = (2s_0^{(1)} - 5)(z-2)$.

Другие $q-1$ рабочие пересчитывают элементы своих подматриц $\widetilde{A}_i^{(1)}$ по формуле (14), выполнив при этом $N_{1i}^{(3)} = 2s_i^{(1)}(z-2)$, $i = \overline{1, q-1}$, операций умножения и вычитания.

Обобщая полученные результаты, можно утверждать, что на k -м шаге вычислений мастер и i-й рабочий выполняют соответственно такое количество операций деления и вычитания:

$$N_{10}^{(k)} = (2(s_0^{(1)} - k) + 1)(z - k + 1), \tag{18}$$

$$N_{1i}^{(k)} = 2s_i^{(1)}(z - k + 1), \quad i = \overline{1, q-1}. \tag{19}$$

Первый цикл параллельных вычислений заканчивается тогда, когда выполнится условие $k = s_0^{(1)}$, т.е. в формулах (18), (19) $k = \overline{1, s_0^{(1)}}$.

Таким образом, после первого цикла вычислений получаем верхнюю прямоугольную матрицу, в которой на главной диагонали будут помещены единицы. Размер такой матрицы $s_0^{(1)}(z+1)$.

После объединения мастером $q-1$ матриц в его памяти будет храниться матрица размером $(z - s_0^{(1)}) \times (z - s_0^{(1)} + 1)$, в которой изъяты все нулевые элементы.

Полученную матрицу мастер делит на q слоев, мощность каждого из них составляет $s_i^{(2)}$, $i = \overline{1, q-1}$, строк. В рабочем пространстве мастера будет находиться матрица $\widetilde{A}_0^{(2)}$ с $s_0^{(2)}$ строками.

Количество операций деления, умножения и вычитания, которые выполняет мастер, и количество операций умножения и вычитания, которые выполняет i -й рабочий, вычислим по формулам (18), (19), учитывая, что количество столбцов подматриц, которые размещены в рабочих пространствах мастера и рабочих, составляет $z - s_0^{(1)} + 1$. Это значит, что в формулах (18), (19) необходимо z заменить на $z - s_0^{(1)}$, а $s_0^{(1)}$ и $s_i^{(1)}$ — соответственно на $s_0^{(2)}$ и $s_i^{(2)}$. В результате такой замены получим:

$$N_{20}^{(k)} = (2(s_0^{(2)} - k) + 1)(z - s_0^{(1)} - k + 1),$$

$$N_{2i}^{(k)} = 2s_i^{(2)}(z - s_0^{(1)} - k + 1), \quad i = \overline{1, q-1}.$$

Второй цикл вычислений закончится выполнением условия $k = s_0^{(2)}$.

После окончания второго цикла вычислений в памяти мастера будет храниться прямоугольная матрица размером $(s_0^{(1)} + s_0^{(2)}) \times (z+1)$, на главной диагонали, которой будут помещены единицы.

В начале третьего цикла вычислений мастер объединяет $q-1$ подматрицу в одну матрицу размером $(z - s_0^{(1)} - s_0^{(2)}) \times (z - s_0^{(1)} - s_0^{(2)} + 1)$, которая не содержит нулевых элементов, и делит ее на q подматриц. Первая подматрица, которая будет иметь $s_0^{(3)}$ строк, остается в рабочем пространстве мастера, а другие отправляются $q-1$ рабочим.

Как и во втором цикле, количество операций деления и вычитания, которые выполняют в третьем цикле соответственно мастер и рабочие, вычислим по формулам (18), (19), заменив z на $z - s_0^{(1)} - s_0^{(2)}$.

При этом необходимо учесть, что мощность подматрицы $\widetilde{A}_0^{(3)}$ составляет $s_0^{(3)}$ строк, а мощность подматриц $\widetilde{A}_i^{(3)}$ имеет $s_i^{(3)}$, $i = \overline{1, q-1}$, строк.

Следовательно,

$$N_{30}^{(k)} = (2(s_0^{(3)} - k) + 1)(z - s_0^{(1)} - s_0^{(2)} - k + 1), \tag{20}$$

$$N_{3i}^{(k)} = 2s_i^{(3)}(z - s_0^{(1)} - s_0^{(2)} - k + 1), \; i = \overline{1, q-1}. \tag{21}$$

Третий цикл заканчивается при условии, что $k = s_0^{(3)}$.

Пусть выполнено r циклов вычислений. В результате в памяти мастера будет размещена прямоугольная матрица размером $\left(\sum_{j=1}^{r-1} s_0^{(j)} \right) \times (z+1)$, на главной диагонали которой размещены единицы. В рабочем пространстве мастера будет находиться матрица размером $\left(z - \sum_{j=1}^{r-1} s_0^{(j)} \right) \times \left(z - \sum_{j=1}^{r-1} s_0^{(j)} + 1 \right)$ с ненулевыми элементами, которую мастер делит на q слоев. В результате получим q подматриц, каждая из которых имеет мощность $s_i^{(r)}$, $i = \overline{0, q-1}$.

Опираясь на результаты формул (20), (21), можем утверждать, что по завершении r-го цикла вычислений мастер и каждый i-й рабочий выполнят такое количество операций деления и вычитания:

$$N_{r0}^{(k)} = (2(s_0^{(r)} - k) + 1)\left(z - \sum_{j=1}^{r-1} s_0^{(j)} - k + 1 \right), \tag{22}$$

$$N_{ri}^{(k)} = 2s_i^{(r)}\left(z - \sum_{j=1}^{r-1} s_0^{(j)} - k + 1 \right), \; r = \overline{1, N_z - 1}, \; i = \overline{1, q-1}. \tag{23}$$

Окончанием r-го цикла является выполнение условия $k = s_0^{(r)}$.

Алгоритм приведения матрицы $\widetilde{A}$ к верхнему треугольному виду с единицами на главной диагонали заканчивает свою работу тогда, когда выполнится условие

$$z - \sum_{j=1}^{N_z - 1} s_0^{(j)} \le q, \tag{24}$$

где N_z — общее количество циклов вычислений.

Теперь вычислим количество операций деления, умножения и вычитания, которые выполняются в течение r-го цикла. Мастер выполняет такое количество указанных операций:

$$N_{M,r} = \sum_{k=1}^{s_0^{(r)}} N_{r0}^{(k)}, \quad r = \overline{1, N_z - 1}, \tag{25}$$

а на долю каждого i-го рабочего приходится следующее количество операций умножения и вычитания:

$$N_{w,r}^{(i)} = \sum_{k=1}^{s_0^{(r)}} N_{ri}^{(k)}, \quad r = \overline{1, N_z - 1}, \; i = \overline{1, q-1}, \tag{26}$$

где $N_{r0}^{(k)}$ и $N_{ri}^{(k)}$ — соответственно определяются по формулам (22) и (23).

Поскольку в общем случае размер матрицы $\widetilde{A}$ не кратен количеству параллельных процессов, то неодинаковое количество строк приводит к неравномерной вычислительной нагрузке процессоров и окончание вычислений будет определяться временем выполнения наиболее загруженного процессора.

Если учесть операции на пересылки $\widetilde{a}_{ij}^{(i)}$ элемента в каждом k-м шаге вычислений, то общее количество операций деления и вычитания, которые выполняются в r-м цикле, будет таким:

$$N_r = \max_{1 \le i \le q-1} \{N_{M,r}, N_{w,r}^{(i)}\} + s_0^{(r)}, \; r = \overline{1, N_z - 1}. \tag{27}$$

Все N_r операций в r-м цикле вычислений выполняются параллельно. Допустим, что на выполнение всех операций деления и вычитания в r-м цикле вычислений тратится τ_r единиц времени, а на операции пересылки — $\tau_{t,r}$ единиц времени. Тогда общие затраты времени на реализацию параллельного алгоритма приведения матрицы $\widetilde{A}$ к верхнему диагональному виду следует вычислять по следующей формуле:

$$T = \sum_{r=1}^{N_z - 1} \left(\tau_r \max_{1 \le i \le q-1} \{N_{M,r}, N_{w,r}^{(i)}\} + \tau_{t,r} s_0^{(r)} \right) + T_f, \tag{28}$$

где T_f — затраты времени на приведение подматрицы на последнем шаге вычислений к верхней диагональной форме.

На последнем этапе вычислений, когда выполнено условие (24), мастер подматрицу $\widetilde{A}_z$ размером $\left(z - \sum_{j=1}^{r-1} s_0^{(j)} \right) \times \left(z - \sum_{j=1}^{r-1} s_0^{(j)} + 1 \right)$ приводит к верхнему диагональному виду в соответствии с формулами (13) и (14).

Вычислим количество операций деления и вычитания, которые выполняются на последнем цикле вычислений. Введем такие обозначения: $z_\alpha = z - \sum_{j=1}^{r-1} s_0^{(j)}$.

Матрицу $\widetilde{A}_z$ можно рассматривать как квадратную матрицу размером z_α, к которой присоединен столбец свободных членов системы линейных алгебраических уравнений. Приведение такой матрицы к верхнему диагональному виду с единицами на главной диагонали потребует [17]

$$N_f = \frac{4z_\alpha^3 + 3z_\alpha^2 - z_\alpha}{6} \tag{29}$$

операций деления, умножения и вычитания.

С учетом значения N_f формулу (28) запишем

$$T = \sum_{r=1}^{N_z - 1} \left(\tau_r \max_{1 \le i \le q-1} \{N_{M,r}, N_{w,r}^{(i)}\} + \tau_{t,r} s_0^{(r)} \right) + \tau_f N_f,$$

где τ_f — затраты времени на выполнение операций на завершающем этапе вычислений.

Найдем верхнюю оценку для величины T, считая, что если начальная матрица $\widetilde{A}$, подлежащая преобразованию к верхнему треугольному виду по формулам (13) и (14), разбивается на q одинаковых слоев, то $s_i^{(r)}$, $i = \overline{0, q-1}$, $r = \overline{1, N_z}$. Поскольку операция пересылки элемента $\widetilde{a}_{ij}^{(i)}$ выполняется значительно быстрее, чем арифметические операции умножения, деления и вычитания, то в последнем выражении можно пренебречь величиной $s_0^{(r)}$. Также допустимо, что $\tau = \max\limits_{1 \le r \le N_z - 1} \{\tau_r, \tau_{t,r}\}$. Тогда

$$T = \tau \left(\sum_{r=1}^{N_z - 1} \max_{1 \le i \le q-1} \{N_{M,r}, N_{w,r}^{(i)}\} + N_f \right). \tag{30}$$

Из формулы (30) следует, что оценка значения T сводится к определению общего числа операций, которые необходимо выполнить для приведения матрицы $\widetilde{A}$ к верхнему треугольному виду.

На первом цикле вычислений мощность каждого из $s_i^{(1)}$ слоев $s_i^{(1)} = \frac{z}{q}$, $i = \overline{0, q-1}$, а матрица $\widetilde{A}_1 = \widetilde{A}$ имеет $n_1 = n$ строк.

По окончании первого цикла вычислений количество строк матрицы $\widetilde{A}$ уменьшилось на величину $s_0^{(1)}$. Новая матрица $\widetilde{A}_2$ будет иметь следующее количество строк: $z_2 = z - s_0^{(1)}$.

Учитывая значения $s_0^{(1)}$, находим $z_2 = \frac{z}{q}(q-1)$.

В начале второго этапа вычислений мастер делит матрицу $\widetilde{A}_2$ на q ровных слоев, что в итоге дает $s_i^{(2)} = \frac{z}{q^2}(q-1)$, $i = \overline{0, q-1}$.

Выполнение второго этапа вычислений приводит к сокращению матрицы $\widetilde{A}_2$ на $s_0^{(2)}$ строк. В итоге получаем матрицу $\widetilde{A}_3$, имеющую следующее количество строк: $z_3 = z - s_0^{(1)} - s_0^{(2)}$.

Если принять к сведению значения $s_0^{(1)}$ и $s_0^{(2)}$, то получим $z_3 = \frac{z}{q^2}(q-1)^2$.

После раздела матрицы $\widetilde{A}_3$ на q ровных слоев получим $s_i^{(3)} = \frac{z}{q^3}(q-1)^2$, $i = \overline{0, q-1}$.

В результате выполнения третьего этапа вычислений получим матрицу $\widetilde{A}_4$ с таким количеством слоев: $z_4 = z - s_0^{(1)} - s_0^{(2)} - s_0^{(3)}$. С учетом значений $s_0^{(1)}$, $s_0^{(2)}$ и $s_0^{(3)}$ будем иметь $z_4 = \frac{z}{q^3}(q-1)^3$.

Если теперь матрицу $\widetilde{A}_4$ разделить на q одинаковых слоев, то $s_i^{(4)} = \frac{z}{q^4}(q-1)^3$, $i = \overline{0, q-1}$.

Следовательно, по завершении r-го цикла вычислений получим матрицу $\widetilde{A}_r$ с количеством строк $z_r = \frac{z}{q^r}(q-1)^r$.

Для продолжения работы алгоритма на $r+1$ цикле мастер разделяет матрицу $\widetilde{A}_r$ на q ровных слоев, так что

$$s_i^{(r)} = \frac{z}{q^r}(q-1)^{r-1}, \ i = \overline{0, q-1}, \ r = \overline{1, N_z - 1}. \tag{31}$$

Предположим, что на последнем цикле вычислений условие (24) будет иметь следующий вид: $z - \sum_{j=1}^{N_z-1} s_0^{(j)} = q$. В соответствии с формулой (27) $s_0^{(j)} = \frac{z}{q^j}(q-1)^{j-1}$, поэтому $1 - \frac{1}{q}\sum_{j=1}^{N_z-1}\left(\frac{q-1}{q}\right)^{j-1} = \frac{q}{z}$.

Выражение $\sum_{j=1}^{N_z-1}\left(\frac{q-1}{q}\right)^{j-1}$ является суммой геометрической прогрессии, вычисление которой, с учетом последней формулы, приводит к следующему результату: $\left(\frac{q-1}{q}\right)^{N_z-1} = \frac{q}{z}$. Последнее уравнение дает возможность найти количество циклов вычислений, необходимых для приведения матрицы $\widetilde{A}$ к верхнему треугольному виду $N_z = \ln\frac{q-1}{z} \cdot \left(\ln\frac{q-1}{q}\right)^{-1}$.

В общем случае величина N_z не принадлежит множеству натуральных чисел, поэтому его следует округлить до целого числа.

Вычисление общего количества операций

$$N_p = \sum_{r=1}^{N_z-1}\left(\max_{1 \le i \le q-1}\{N_{M,r}, N_{w,r}^{(i)}\} + s_0^{(r)}\right) + N_f. \tag{32}$$

происходит согласно следующему алгоритму. По формулам (25) и (26) находим значение $N_{M,r}$ и $N_{w,r}^{(i)}$. Поскольку в каждом цикле вычислений дежурная матрица $\widetilde{A}_i$ разбивается на q ровных слоев, то $N_{w,r}^{(i)}$ зависит только от индекса r, т.е. $N_{w,r} = N_{w,r}^{(i)}$.

С учетом формулы (31) соотношения (22) и (23), которые входят в формулы (25) и (26), приобретут следующий вид:

$$N_{r0}^{(k)} = \left(2\left(\frac{z}{q}L(q) - k\right) + 1\right)(zL(q) - k + 1),$$

$$N_r^{(k)} = 2\frac{z}{q}L(q)(zL(q) - k + 1), \quad r = \overline{1, N_z - 1},$$

где $L(q) = \left(1 - \frac{1}{q}\right)^{r-1}$.

Зная $N_{M,r}$ и $N_{w,r}$, по формуле (32) вычислим N, $s_0^{(r)}$ определим согласно соотношению (31), а N_f — по формуле (29).

Обратный ход, в котором в обратном порядке определяются неизвестные, осуществляется по формулам

$$a_{A,z} = u_{z,z+1},\ a_{A,i} = u_{i,z+1} - \sum_{j=i+1}^{z} u_{ij}a_{A,j},\ i = \overline{z-1,1}. \tag{33}$$

Проанализируем алгоритм (33) решения системы линейных алгебраических уравнений с треугольной матрицей U методом обратной подстановки. Физическое содержание задачи синтеза оптимальной структуры эмпирической модели таково, что матрица U является невырожденной.

Формула (33) не определяет алгоритм однозначно, так как не определен порядок нахождения суммы. Рассмотрим последовательную операцию суммирования, которая выполняется в правой части соотношения (33). Соответствующую рекуррентную процедуру запишем:

$$a_{A,z}^{(0)} = a_{A,z} = u_{z,z+1},\ a_{A,z-i}^{(j)} = a_{A,z-i}^{(j-1)} - u_{z-i,z-j+1}a_{A,z-j+1}^{(j-1)},$$

$$a_{A,z-i}^{(0)} = u_{z-i,z+1},\ a_{A,z-i} = a_{A,z-i}^{(i)},\ i = \overline{1,z-1},\ j = \overline{1,i}. \tag{34}$$

Процедуре нахождения решения уравнения (15) по формулам (33) свойственен внутренний параллелизм. Поэтому для экономии машинного времени синтезируем соответствующий параллельный алгоритм.

На первом шаге вычислений предпоследний z столбец матрицы U, который имеет $z-1$ отличающихся от нуля элементов (не учитывая элемента $u_{zz} = 1$), разделим между q процессорами так, что мастер будет иметь $s_0^{(1)}$ элементов, а рабочие — по $s_k^{(1)}$, $k = \overline{1, q-1}$, элементов. По формуле (16) вычислим значение $a_{A,z-i}^{(1)}$, где $i = \overline{1, s_0^{(1)}}$ для мастера; $i = \overline{s_{k-1}^{(1)} + 1, s_k^{(1)}}$, $k = \overline{1, q-1}$, для рабочих. Рабочие пересылают свои значения $a_{A,z-i}^{(1)}$ мастеру, при этом формируется множество значений $a_{A,z-1}$ и $a_{A,z-i}^{(1)}$, $i = \overline{2, z-1}$, хранящихся в рабочей области мастера.

На втором шаге $z-2$ элементы $z-1$ столбца разделяем между q процессорами. В результате такого разделения мастер получит $s_0^{(2)}$ элементов, а рабочие — по $s_k^{(2)}$, $k = \overline{1, q-1}$, элементов. Затем мастер, включая и себя, рассылает соответствующее

количество элементов $a_{A,\,z-1}$ и $a^{(1)}_{A,\,z-i}$, $i=\overline{2,\,z-1},$ каждому процессору, включая и самого себя. Затем по формуле (34) вычисляют значение $a^{(2)}_{A,\,z-i-1}$, где $i=\overline{1,\,s^{(1)}_0}$ для мастера; $i=\overline{s^{(2)}_{k-1}+1,\,s^{(2)}_k}$, $k=\overline{1,\,q-1},$ для рабочих. При этом $a_{A,\,z-2}=a^{(2)}_{A,\,z-2}$. Завершается второй шаг формированием в рабочей области мастера множества значений $a_{A,\,z-2}$ и $a^{(2)}_{A,\,z-i}$, $i=\overline{3,\,z-1}$.

Дальнейшие шаги вычислений происходят по приведенной схеме и на r-м шаге мастер разделяет $z-r$ элементы $z-r+1$ столбца между q процессорами таким образом: мастер будет иметь $s^{(r)}_0$ элементов, а рабочие — по $s^{(r)}_k$, $k=\overline{1,\,q-1},$ элементов. По формуле (34) вычисляются значения $a^{(r)}_{A,\,z-i-r+1}$, где $i=\overline{1,\,s^{(1)}_0}$ для мастера; $i=\overline{s^{(r)}_{k-1}+1,s^{(r)}_k}$, $k=\overline{1,\,q-1},$ для рабочих. Как результат вычислений в рабочей области мастера будет содержаться множество значений $a_{A,\,z-r}$ и $a^{(r)}_{A,\,z-i}$, $i=\overline{r+1,\,z-1}$.

Процесс решения алгебраического уравнения (15) методом обратного хода с использованием параллельного алгоритма завершается последним шагом вычислений при выполнении условия $z-r\le q.$ Тогда мастер вычисляет все значения $a_{A,\,z-i},$ где $i=\overline{r,\,z-1},$ по формуле (34). Вычислим количество операций умножения и вычитания, которые осуществляются в результате реализации параллельного алгоритма вычислений по формуле (16). На r-м шаге вычислений мастер осуществит

$$N^{(r)}_0=2s^{(r)}_0, \tag{35}$$

а каждый рабочий

$$N^{(r)}_i=2s^{(r)}_i \tag{36}$$

операций умножения и вычитания.

В общем случае размер каждого столбца матрицы U может оказаться не кратным количеству процессоров, что приводит к разной вычислительной нагрузке процессоров и окончание вычислений на каждом шаге будет определяться временем работы наиболее загруженного процессора, поэтому на $r-1$-м шаге вычислений $N^{(r)}=\max\limits_{1\le i\le q-1}(N^{(r)}_0,\,N^{(r)}_i)$.

Поскольку параллельные вычисления ведутся по столбцам матрицы $U,$ то $r=\overline{1,\,z}$ и общее число операций при реализации параллельного алгоритма будет следующим:

$$N=\sum_{r=1}^{z-s^{(f)}_0}\max_{1\le i\le q-1}(N^{(r)}_0,\,N^{(r)}_i)+N_f,$$

где N_f — число операций умножения и вычитания на последней стадии вычислений.

При переходе от одного шага вычислений к другому число элементов текущего столбца уменьшается на единицу. Поэтому на определенном шаге вычислений

будет иметь место условие $s_0^{(f)} = q$, которое и определит завершение процесса вычислений. Учитывая условие завершения процесса вычислений, а также формулы (35), (36), приходим к выводу, что

$$N = 2\sum_{r=1}^{z-q} \max_{1 \le i \le q-1} (s_0^{(r)}, s_i^{(r)}) + N_f. \tag{37}$$

Вычислим величину N_f. Последний шаг вычислений начинается с условия $s_0^{(f)} = q$. Дальнейшие действия производит мастер, вычисляя параметры модели согласно описанной процедуре. Поэтому общее количество операций умножения и вычитания на последнем шаге вычислений определяется по формуле $N_f = 2\sum_{i=1}^{q} (q-i) = q(q-1)$. С учетом значения N_f формула (38) будет следующей:

$$N = 2\sum_{r=1}^{z-q} \max_{1 \le i \le q-1} (s_0^{(r)}, s_i^{(r)}) + q(q-1). \tag{38}$$

При равномерной загрузке процессоров $s_0^{(r)} = s_i^{(r)} = \frac{z-r}{q}$, $\forall i$. В таком случае значение N согласно формуле (38) приобретет такой вид: $N = \frac{2}{q}\sum_{r=1}^{z-q}(z-r) + q(q-1)$. Вычислив значение $\sum_{r=1}^{z-q}(z-r)$, приходим к выводу, что

$$N = \frac{z-q}{q}(z+q-1) + q(q-1). \tag{39}$$

Реализация последовательного алгоритма вычисления коэффициентов математической модели (6), после того как полученная матрица U, требует $\widetilde{N} = z(z-1)$ операций умножения и вычитания [17].

Формулу (9) запишем в несколько ином виде:

$$\bar{y}(B) = \sum_{i=1}^{z} \bar{f}_{B,\bullet i} a_{A,i}, \tag{40}$$

где $\bar{f}_{B,\bullet i}$ — i-й столбец матрицы $\widetilde{F}_B$.

Анализ формулы (40) показывает, что каждое слагаемое суммы может вычисляться независимо, а это свидетельствует о внутреннем параллелизме алгоритма вычисления $\bar{y}(B)$ по формуле (9).

Разделим столбцы матрицы $\widetilde{F}_B$ между q процессорами так, что в памяти каждого процессора будет храниться по s_i, $i = \overline{0, q-1}$, столбцов. Теперь каждый из q параллельно работающих процессоров будет вычислять сумму $\bar{y}_i(B) = \sum_{j=1}^{s_i} \bar{f}_{B,\bullet j} a_{A,j}$, $i = \overline{0, q-1}$.

Очевидно, что каждый из процессоров выполнит $N_B s_i$ операций умножения и $N_B(s_i - 1)$ операций суммирования. Поэтому общее количество операций умножения и суммирования, которые выполняет каждый процессор на первом шаге вычислений, определится выражением $N_i = N_B(2s_i - 1),\ i = \overline{0, q-1}$.

По окончании первого шага вычислений каждый рабочий передает значение $\bar{y}_i(B),\ i = \overline{1, q-1},$ мастеру, который и вычисляет $\bar{y}(B) = \sum_{i=0}^{q-1} \bar{y}_i(B),$ тратя на эту процедуру qN_B операций суммирования. Поскольку машинное время работы системы параллельных процессоров определяется временем работы процессора, который имеет наибольшее количество столбцов матрицы $\widetilde{F}_B$, то общее количество операций умножения и суммирования при реализации параллельного алгоритма вычисления $\bar{y}(B)$, будет определяться формулой $N = \max\limits_{0 \le i \le q-1} N_i + qN_B$.

Для случая, когда столбцы матрицы $\widetilde{F}_B$ равномерно распределены между всеми процессорами, $s_i = \frac{z}{q},\ i = \overline{0, q-1},$ и $N = N_B(2\frac{z}{q} + q - 1)$.

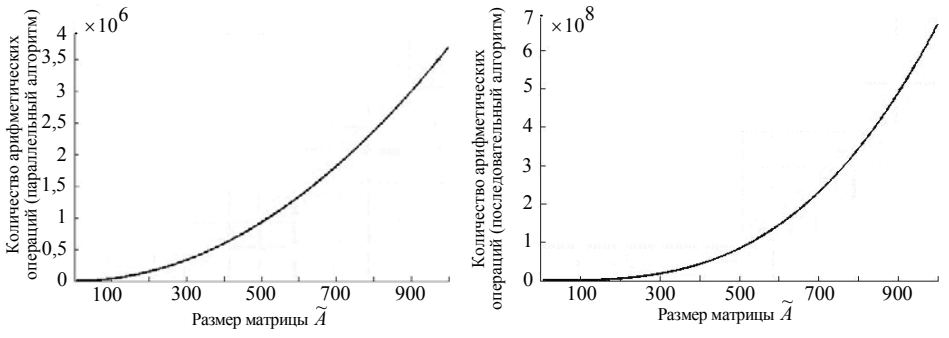
Таким образом, для реализации последовательного алгоритма вычисления $\bar{y}(B)$ по формуле (9) необходимо выполнить $\widetilde{N} = N_B(2z - 1)$ операций умножения и суммирования.

Суммарное количество арифметических операций N_P, выполняемых при реализации параллельного алгоритма, будет определяться количеством операций на приведение матрицы $\widetilde{A}$ к верхнему треугольному виду, количеством операций на решение уравнения (15) и количеством операций на вычисления значений $\bar{y}(B)$. Соответствующим образом суммарное количество арифметических операций N_S определяется и для последовательного алгоритма [17].

Результаты численных экспериментов

На рис. 1 показан график суммарного количества операций, необходимых при решении уравнения (12), для параллельного (*а*) и последовательного (*б*) алгоритмов в зависимости от размера матрицы $\widetilde{A}$, а рис. 2 — это зависимость N_S / N_P от размера матрицы $\widetilde{A}$.

Анализ полученных результатов показывает, что при использовании параллельного алгоритма решения уравнения (12) эффективность параллельного алгоритма растет с увеличением размера матрицы $\widetilde{A}$.



а *б*

Рис. 1

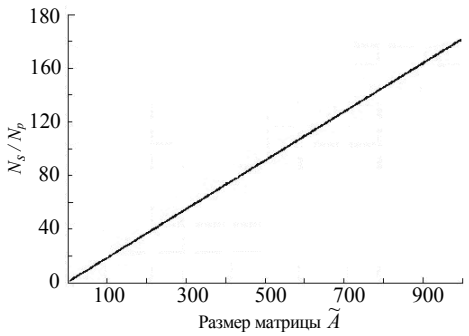


Рис. 2

Заключение

Синтез эмпирических моделей оптимальной сложности на принципах генетических алгоритмов требует многократного решения системы линейных алгебраических уравнений и нахождения значения $\overline{y}(B)$ по формуле (9), на что тратится большая часть машинного времени. Применение параллельного алгоритма для решения поставленных задач дает значительный выигрыш во времени по сравнению с последовательным алгоритмом.

Следует отметить, что приведенные расчеты не учитывают потери времени на обмен данными, синхронизацию и конфликты памяти. Учет таких потерь несколько увеличит затраты времени на реализацию параллельного алгоритма, но они будут значительно меньше затрат времени на реализацию последовательного алгоритма решения уравнения (12) и на отыскание значения $\overline{y}(B)$ по формуле (9).

М.І. Горбійчук, В.М. Медведчук, А.М. Лазорів

АНАЛІЗ ПАРАЛЕЛЬНОГО АЛГОРИТМУ СИНТЕЗУ ЕМПІРИЧНИХ МОДЕЛЕЙ НА ПРИНЦИПАХ ГЕНЕТИЧНИХ АЛГОРИТМІВ

Розроблено метод синтезу емпіричних моделей з використанням генетичних алгоритмів, який порівняно з індуктивним методом самоорганізації моделей значно скорочує затрати машинного часу на їх реалізацію. Для підвищення ефективності обчислювального процесу розроблено паралельний алгоритм та здійснено його аналіз, що дало змогу оцінити зменшення затрат машинного часу порівняно з послідовним алгоритмом.

M.I. Gorbiychuk, V.M. Medvedchuk, A.N. Lazoriv

ANALYSIS OF PARALLEL ALGORITHM OF EMPIRICAL MODELS SYNTHESIS ON THE PRINCIPLES OF GENETIC ALGORITHMS

The developed method of synthesis of empirical models using the genetic algorithms reduces the computing time for the implementing of empirical models, comparing to the inductive method of self-organizing models. To improve the effectiveness of computing process the parallel algorithm was developed and analyzed, which allows reducing the computing time comparing to the sequential or / FIFO / algorithm.

1. *Ермаков С.М., Жиглявский А.А.* Математическая теория оптимального эксперимента. — М.: Наука, 1987. — 320 с.
2. *Химмельблау Д.* Анализ процессов статистическими методами. — М.: Мир, 1973. — 957 с.
3. *Грешилов А.А.* Математические методы принятия решений. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2006. — 584 с.
4. *Норкин В.И., Кайзер М.А.* Об асимптотической эффективности ядерного метода опорных векторов (SVM) // Кибернетика и системный анализ. — 2009. — № 4. — С. 81–97.
5. *Draper N. R., Smith H.* Applied regression analysis. — Chichester: Wiley, 1998. — 736 p.
6. *A distribution* free theory of nonparametric regression / L. Gyorfi, M., Kohler, A. Krzyzak, H. Walk. — New York; Berlin; Heidelberg: Springer, 2002. — 647 p.
7. *Хайкин С.* Нейронные сети: полный курс. — М.: Вильямс, 2006. — 1104 с.
8. *Нагель Э., Ньюмен Д.* Теорема Геделя. — М.: Знание, 1970. — 62 с.
9. *Ивахненко А.Г., Лапа В.Г.* Предвидение случайных процессов. — Киев: Наук. думка, 1969. — 420 с.
10. *Ивахненко А.Г.* Индуктивный метод самоорганизации моделей сложных систем. — Киев: Наук. думка, 1981. — 286 с.
11. *Горбийчук М.И., Когутяк М.И., Заячук Я.И.* Индуктивный метод построения математических моделей газоперекачивающих агрегатов природного газа // Нефтяная и газовая промышленность. — 2008. — № 5. — С. 32–35.
12. *Справочник* по типовым программам моделирования / А.Г. Ивахненко, Ю.В. Коппа, В.С. Степашко и др. — Киев: Техніка, 1980. — 180 с.
13. *Рутковская Д., Пилиньский М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы — М.: Горячая линия — Телеком, 2004. — 452 с.
14. *Метод* синтеза математических моделей на принципах генетических алгоритмов / М.И. Горбийчук, М.И. Когутяк, О.Б. Василенко, И.В. Щупак // Разведка и разработка нефтяных и газовых месторождений. — 2009. — № 4 (33). — С. 72–79.
15. *Оленев Н.Н., Печенкин Р.В., Чернецов А.М.* Параллельное программирование в MatLab и его приложение. — М.: ВЦ РАН, 2007. — 120 с.
16. *Вержбицкий В.М.* Основы численных методов: учебник для вузов. — М.: Высш. школа, 2002. — 840 с.
17. *Волков Е.А.* Численные методы: учебное пособие для вузов. — 2-е изд., исп. — М.: Наука, 1987. — 248 с.

Получено 20.04.2015
После доработки 23.07.2015