КЛАСТЕРНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА СОГЛАСОВАННОЙ ИДЕНТИФИКАЦИИ

К.Г. Пугачев [1], Е.В. Гошин [1,2], В.А.Фурсов [1,2]

[1] Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет) (СГАУ), Самара, Россия,
[2] Институт систем обработки изображений РАН, Самара, Россия

В статье рассматривается параллельная реализация вычислительного алгоритма согласованной идентификации. Идея метода состоит в том, что исходные данные разбиваются на множество подсистем небольшой размерности, на которых ищется множество оценок. Затем на этом множестве определяется подмножество наиболее согласованны на котором и строится искомая точечная оценка. Высокая точность и надежность метода обеспечивается за счет использования большого числа подсистем, формируемых путем различных сочетаний строк исходной системы. Однако это достигается ценой огромных вычислительных затрат. В работе рассматриваются проблемы организации параллельного вычислительного процесса и результаты исследований при различных размерностях задачи.

Ключевые слова: критерий согласованности, метод наименьших квадратов, ускорение, эффективность.

Введение

Задача идентификации [1] заключается в построении оптимальной в каком-то смысле модели объекта (системы) по результатам наблюдений входных и выходных данных. Бывают случаи, когда требуется проводить идентификацию по малому числу наблюдений. Это может быть связано с требованием по оперативности определения параметров, высокой стоимостью наблюдений, необходимость большого количества экспериментов и прочих факторов.

Одним из методов, позволяющих эффективно работать с малым числом измерений является метод согласованной идентификации. Большим преимуществом этого метода является то, что он позволяет справляться с шумом, возникающим при наблюдении сигнала. Высокая точность и надежность метода обеспечивается за счет использования большого числа подсистем, формируемых путем различных сочетаний строк исходной системы. За счет этого метод имеет высокую вычислительную сложность и требует больших затрат по памяти.

При увеличении размерности задачи идентификации на каком-то этапе вычисления на обычных персональных компьютерах не представляются возможными. В связи с этим возникает необходимость параллельных вычислений на суперкомпьютерах с целью уменьшения времени работы метода и решения задач большей размерности.

1. Постановка задачи согласованной идентификации

Рассматривается задача оценивания вектора параметров с линейной модели:

$$\mathbf{y} = \mathbf{X}\mathbf{c} + \xi, \qquad (1)$$

где $\mathbf{y}$ и $\mathbf{X}$ – наблюдаемые в эксперименте $N\times 1$-вектор и $N\times M$-матрица, составленная из столбцов $\mathbf{x}_i$, $i=\overline{1,N}$ соответственно, а $\xi=\left[\xi_1,\xi_2,...,\xi_N\right]^T$ – $N\times 1$ -вектор неизвестных ошибок. Задача идентификации заключается в определении по наблюдениям $\mathbf{y}$ и $\mathbf{X}$ $M\times 1$-

вектора оценок $\hat{\mathbf{c}}$. Если априорная информация об ошибках отсутствует, то обычно применяют метод наименьших квадратов (МНК):

$$\hat{\mathbf{c}} = \left[\mathbf{X}^T\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{y}. \qquad (2)$$

Известно, что МНК-оценки являются несмещёнными и эффективными при обычных предположениях [2]. Однако при малом числе наблюдений эти предположения оказываются ненадежными из-за недостаточной статистической устойчивости вероятностных характеристик. В работе [3] применительно к задаче идентификации управляемого объекта рассматривался подход, свободный от использования априорных предположений о распределении ошибок измерений, метод *согласованной идентификации*. Метод опирается на предположение, что решения, полученные на подсистемах, наиболее свободных от шума, будут более близкими (согласованными), а задача состоит в определении такой подсистемы. Приведем краткое описание алгоритма идентификации, основанного на принципе согласованности оценок.

Из исходной системы (2) можно «извлечь» некоторое множество подсистем малой размерности. Удобно представить это в виде умножения системы (2) слева на прямоугольную матрицу $\mathbf{G}_k$, в каждой строке которой содержится только одна единица, а остальные элементы нулевые:

$$\mathbf{y}_k = \mathbf{X}_k\mathbf{c}_k + \boldsymbol{\xi}_k, \quad k = 1,2,\ldots, \qquad (3)$$

где $\mathbf{y}_k = \mathbf{G}_k\mathbf{y},\ \mathbf{X}_k = \mathbf{G}_k\mathbf{X},\ \boldsymbol{\xi}_k = \mathbf{G}_k\boldsymbol{\xi},$

$$rank\,\mathbf{G}_k = rank\,\mathbf{X}_k.$$

В каждую подсистему (3) войдут строки исходной системы (2), номера которых совпадают с номерами столбцов матрицы $\mathbf{G}_k$, содержащих единицы. Эти подсистемы далее будем называть *подсистемами нижнего уровня*. Для определенности далее будем полагать, что все матрицы $\mathbf{G}_k$ имеют размерность $M \times N$. При этом множество подсистем нижнего уровня будет содержать C_N^M подсистем с квадратными $M \times M$-матрицами $\mathbf{X}_k$. Вычисляя по доступным для каждой подсистемы (3)наблюдениям $\mathbf{X}_k, \mathbf{y}_k$ оценку $\hat{\mathbf{c}}_k$, можно получить соответственно C_N^M всех возможных оценок на подсистемах нижнего уровня.

Аналогичным образом (из нулей и единиц) строится множество прямоугольных $P \times N$-матриц $\mathbf{H}_l$ ($M < P < N$), с использованием которых формируются *подсистемы верхнего уровня*:

$$\tilde{\mathbf{y}}_l = \tilde{\mathbf{X}}_l\mathbf{c}_l + \tilde{\tilde{\boldsymbol{\xi}}}_l, \quad (4)$$

где $\tilde{\mathbf{X}}_l = \mathbf{H}_l\mathbf{X}$, $\tilde{\mathbf{y}}_l = \mathbf{H}_l\mathbf{y}$, $\tilde{\tilde{\boldsymbol{\xi}}}_l = \mathbf{H}_l\boldsymbol{\xi}$,

$$rank\,\mathbf{H}_l = P, \quad (M < P < N), \quad l = \overline{1,L}.$$

Каждой l-й подсистеме верхнего уровня (4) принадлежит некоторое множество подсистем нижнего уровня (3), на которых может быть получено соответствующее множество $\Theta(l)$ оценок $\hat{\mathbf{c}}_{l,k}$:

$$\Theta(l) = \left\{\hat{\mathbf{c}}_{l,k} \in \Theta(l)\right\} \quad l = \overline{1,L}, \quad k = \overline{1,K}.$$

Для характеристики множеств $\Theta(l)$ вводится функция взаимной близости оценок:

$$W(l) = \sum_{\substack{i,j=1,\\ i\neq j}}^{K} \left(\hat{\mathbf{c}}_{l,i} - \hat{\mathbf{c}}_{l,j}\right)^2, \qquad (5)$$

где $\hat{\mathbf{c}}_{l,i}$, $\hat{\mathbf{c}}_{l,j}$, $l=\overline{1,L}$, $i=\overline{1,K}$, $j=\overline{1,K}$ – оценки, полученные на подсистеме нижнего уровня, принадлежащей l-й подсистеме верхнего уровня. Индексы i,j в правой части (5) перебираются во всех возможных парных сочетаниях.

Множество $\Theta(l)$ оценок $\hat{\mathbf{c}}_{l,k}$, $k=\overline{1,K}$, $K=C_P^M$, для которого $W(l)$ принимает минимальное значение, называется наиболее согласованным. Гипотеза состоит в том, что наиболее согласованная подсистема является наименее зашумленной, поэтому задача сводится к отысканию номера подсистемы верхнего уровня $\hat{l}$:

(10)(6)

$$W(\hat{l}) = \min_{l} W(l), \quad l=1,L, \quad L=C_N^P.$$

На $\hat{l}$-й подсистеме верхнего уровня может быть построена либо точечная оценка, либо указано «облако» решений [2].

Нетрудно заметить, что реализация алгоритма согласованной идентификации требует огромных вычислительных затрат. Например, при $N=10,\ M=5,\ P=8$ число подсистем верхнего уровня – 45, а нижнего уровня – 42. При $N=20,\ M=10,\ P=16$ их уже 4845 и 184756 соответственно. Факт высокой вычислительной сложности метода согласованной идентификации – это неизбежная плата за недостаток априорной информации при малом числе наблюдений. Для применения относительно «дешевых» статистических схем обработки необходимы, возможно значительные, затраты на сбор данных. Однако в ряде случаев получение большого числа наблюдений принципиально невозможно. В этом случае наиболее целесообразный путь – построение эффективного параллельного алгоритма.

2. Общая схема параллельного алгоритма

Можно наметить, по крайней мере, два пути построения общей схемы параллельного алгоритма. Первый путь – сначала вычислить все оценки на подсистемах нижнего уровня, а затем реализовать обмен этими оценками для формирования множеств оценок, входящих в подсистемы верхнего уровня. Второй путь – сначала сформировать на различных процессорах все подсистемы верхнего уровня, а затем для каждой из них сформировать все подсистемы нижнего уровня и вычислить соответствующие множества оценок и функции взаимной близости.

В первом случае обмен оценками, полученными на подсистемах нижнего уровня, может потребовать значительных затрат на пересылку сообщений между процессорами. Недостаток второго подхода – это большой объем дублирующих вычислений на одних и тех же подсистемах нижнего уровня, принадлежащих различным подсистемам верхнего уровня.

В настоящей работе предлагается смешанная схема, в которой группы подсистем верхнего уровня формируются на разных процессорах. При этом эти группы подсистем на каждом узле кластера задаются таким образом, чтобы максимально сократить дублирующие вычисления на соответствующих им подсистемах нижнего уровня. При этом подсистемы верхнего уровня на каждом узле также обрабатываются параллельно в разных процессах.

На рис. 1 представлена общая блок-схема предлагаемого алгоритма, на рис. 2 – подробные блок-схемы этапов алгоритма: 2а – подготовительный этап, 2б – определение «наилучшей» подсистемы верхнего уровня для текущего процесса, 2в – сбор данных со всех процессов и расчет решения.

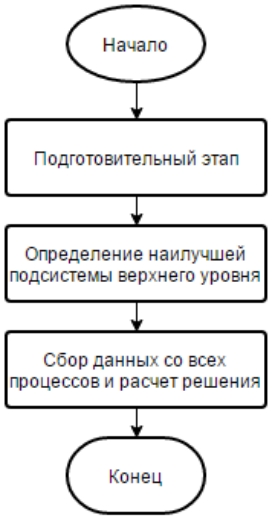


Рис.1. Общая схема алгоритма

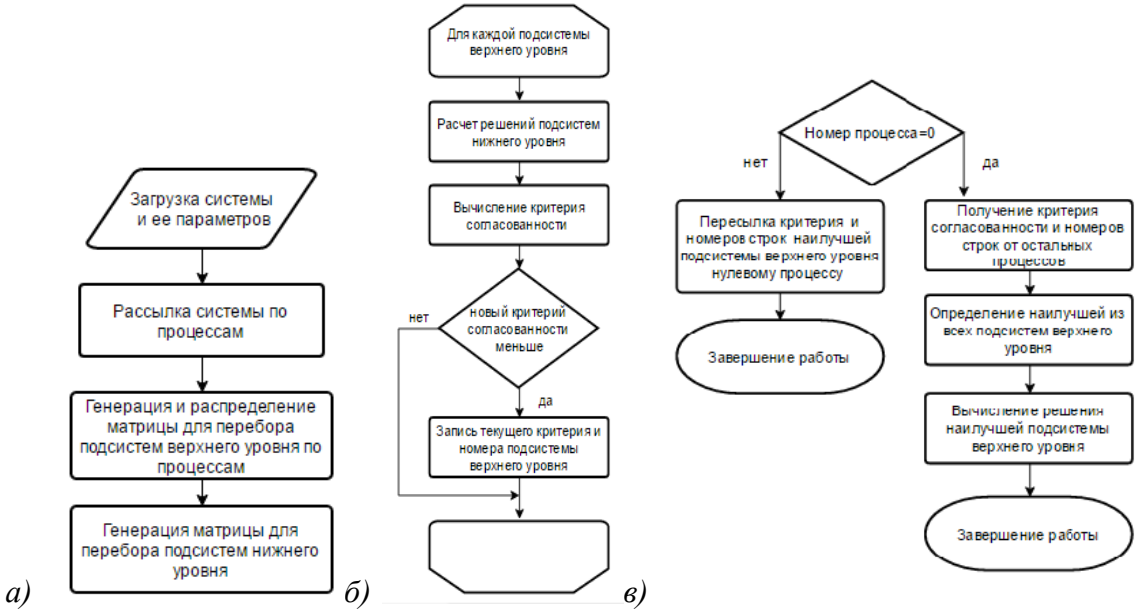


Рис. 2. Основные этапы алгоритма

3. Программная реализация алгоритма

Для реализации алгоритма фильтрации была написана программа на языке C++ в среде разработки Visual Studio 2010. Т.к. предполагалось, что программа должна поддерживать распределенные вычисления на нескольких узлах, для распараллеливания была использована технология Message Passing Interface (MPI).

За основные вычисления по методу согласованной идентификации отвечает созданный класс ConsistentIdentification.

Для обеспечения взаимодействия между процессами был написан специальный класс ClusterLinkCi, который содержит в себе экземпляр класса ConsistentIdentification.

Разработанная программа была запущена на кластере «Сергей Королев».

4. Результаты экспериментов

В качестве исходных данных для эксперимента генерировались матрицы размером $N \times M$. Матрицы заполнялись случайными числами, равномерно распределенными на отрезке [-10, 10]. Искомый вектор для простоты задавался из одних единиц. По сгенерированной матрице и заданному вектору решения вычислялся вектор правой части. Вектор ошибок задавался числами, равномерно распределенными на отрезке [-0,5, 0,5]. Кроме того, с вероятностью 0,1 вносились грубые ошибки типа сбоев.

Табл. 1. Результаты эксперимента 1

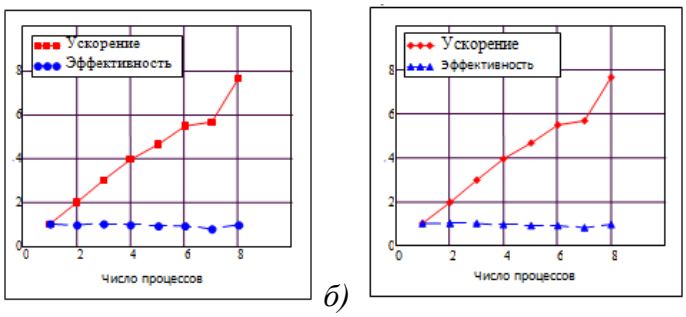
Число процессов	Время работы	Ускорение	Эффективность
1	793,530322	1,000000	1,000000
2	399,019544	1,988700	0,994350
3	264,601322	2,998966	0,999655
4	202,061476	3,927173	0,981793
5	170,967832	4,641401	0,928280
6	145,024123	5,471713	0,911952
7	139,921319	5,671261	0,810180
8	103,603291	7,659316	0,957414

Из таблицы 1 видно, что при увеличении числа процессов время работы программы уменьшается, а значения эффективности показывают высокую степень загруженности процессоров. Это связано с тем, что все вычисления равномерно распределяются по всем процессам, поэтому наблюдается практически линейный рост ускорения.

Для установления зависимости времени решения, ускорения и эффективности от размерности задачи, процедура сначала была запущена для системы с параметрами N=18, M=5, P=16, а затем для систем с параметрами N=18, M=5, P=9. Оказалось, что значения ускорения и эффективности при этом практически одинаковые. Таким образом, от размерности задачи (числа строк N) зависит только общее время, в частности, при меньшем значении N время работы программы сокращается. Результаты работы алгоритма при N=18, M=5, P=16 (эксперимент 2) показаны в таблице 2. На рис. 3 показаны графики ускорения и эффективности при параметрах: 3а – при N=18, M=5, P=16, 3б – при N=18, M=5, P=9.

Табл. 2. Результаты эксперимента 2

Число процессов	Время работы	Ускорение	Эффективность
1	78,381905	1,000000	1,000000
2	39,467237	1,985999	0,993000
3	26,104930	3,002571	1,000857
4	19,970079	3,924967	0,981242
5	16,878340	4,643934	0,928787
6	14,328394	5,470390	0,911732
7	13,812453	5,674727	0,810675
8	10,231733	7,660668	0,957583



а) *б)*

Рис. 3. Зависимость ускорения и эффективности от числа процессов

Было проведено также исследование зависимости времени работы программы при различных N и фиксированных значениях M=10 и P=12. Время работы программы при расчете на одном процессе (эксперимент 3) приведено в таблице 3. Нетрудно заметить, что при увеличении числа строк исходной системы (1) время работы значительно возрастает.

Табл. 3. Результаты эксперимента 3

Число строк	Время работы
14	0.074093
15	0.237907
16	0.895264
17	3.035027
18	9.341413
19	24.788283
20	63.145175

Заключение

Установлено, что разработанный алгоритм является хорошо масштабируемым, время затрачиваемое на работу уменьшается практически линейно. Эффективность также является достаточно высокой, т.к. нагрузка равномерно распределяется между процессорами. Ускорение и эффективность практически не зависят от размерности подсистем верхнего и нижнего уровня. При увеличении размерности исходной (числа строк N) время работы программы значительно возрастает.

Благодарности

Работа выполнена при поддержке Министерства образования и науки РФ и РФФИ (проект № 16-07-00729 А).

Литература

1. Эйкхофф П. Основы идентификации систем управления: Пер. с англ. / Под ред. Н.С. Райбмана. М.: Мир, 1975.
2. Лоусон. Ч., Хенсон Р. Численное решение задач метода наименьших квадратов: Пер. с англ. М.: Наука, Гл. ред. физ.- мат. лит., 1986. 232 с.
3. Vladimir A. Fursov, Andrey V. Gavrilov. Conforming Identification of the Controlled Object, Proceeding International Conference on Computing, Communications and Control Technologies: CCCT’2004, August 14-17, 2004 - Austin, Texas, USA. P. 326-330.