УДК 624.073.02

Тажмамат уулу К.- преп. ОшТУ
Tajmamat uulu K – teacher OshTU

АЛГОРИТМ БЫСТРОГО ОБРАЩЕНИЯ МАТРИЦ

В статье приводится реализация алгоритма быстрого обращения матриц, удобного для одновременного выполнения на нескольких компьютерах.

Ключевые слова: алгоритм, матрицы, компьютеры, параллельное программирование.

FAST MATRIX INVERSION ALGORITHM

The article provides a rapid implementation of the algorithm of matrix inversion, convenient for simultaneous execution on multiple computers.

Keywords: algorithm, matrix, computers, parallel programming.

Многие технические задачи промышленного моделирования при конструировании новых образцов продукции сводятся к математической задаче обращения матриц. Так, например, в строительных вычислительных комплексах построив и обратив главную матрицу жесткости один раз, многократно ее умножают на векторы – столбцы внешних усилий, приводимых к узловым для различных загружений и получают искомые перемещения, а затем и внутренние усилия для различных загружений и для каждого элемента комбинаторным способ ищут наиневыгоднейшее сочетание внутренних усилий от разных загружений, на которое рассчитывают уже прочность отдельных элементов по эмпирическим формулам. При этом больше всего времени уходит на обращение матрицы. Хоть задача обращения матриц на первый взгляд и кажется простой и давно решенной, больше всего ресурсов уходит в существующих промышленных программных комплексах именно на нее. Классические обращения с пошаговым понижением ранга матрицы или с применением исключений по Гауссу приводят к резкому росту потребления ресурсов и в конечном итоге невозможности обращения матрицы выше определенного размера. Кроме того, в прямом виде они неудобны для параллельного программирования, так как приводят к необходимости обмена большим количеством данных между параллельными процессами. Существующие клеточные и многофронтальные способы обращения матриц уменьшают количество вычислительных операций за счет резкого увеличения математической сложности задачи. Увеличение математической сложности приводит к увеличению вероятности допущения ошибки при реализации алгоритма, усложнению возможностей оптимизации и повышению трудозатрат на реализацию алгоритма. Так, первую приемлемую реализацию разложения больших матриц мультифронтальным методом смогли выполнить инженеры аэрокосмической корпорации Boieng с ее большими поддерживаемыми правительством США ресурсами. Те промышленные программы Советского Союза и СНГ, в которых основные ресурсы тратятся на обращение матриц, смогли освоить мультифронтальный способ только после 45 лет эксплуатации этих программ, а режим параллельных вычислений они до сих пор еще не освоили, заставляя пользователей покупать очень дорогие и мощные компьютеры. Зарубежные программы, позволяющие проводить подобные параллельные вычисления, продаются только определенным организациям и физическим лицам по очень высокой цене, то есть не каждый имеющий большие деньги может их купить, чтобы начать разрабатывать высокотехнологичную продукцию. Вопрос разработки собственных современных вычислительных комплексов, моделирующих физические процессы, не решила даже Китайская народная республика, поэтому ее инженеры вынуждены копировать некоторые важные узлы современной техники. Но оказывается, как будет видно далее, проблема обращения матриц, к которой сводятся многие современные технические проблемы, имеет неожиданно простое и

эффективное решение.

При моделировании промышленной продукции инженеры имеют стремление к увеличению числа учитываемых характеристик, что привело к необходимости обращать матрицы очень больших размеров. Общеизвестно, что ресурсы обычных компьютеров недостаточны для решения подобных задач, поэтому приходится использовать параллельные вычисления и суперкомпьютеры, состоящие из многих процессоров. Как известно, при использовании параллельного программирования разбивку задачи на одновременно и независимо выполняемые процессы лучше всего сделать на уровне алгоритма, а не реализации, так как это принесет наибольший экономический эффект. Такую попытку и произвел автор, для того, чтобы получить простой для реализации алгоритм.

За основу был принят способ обращения матрицы при помощи разбиения на клетки. Пусть S – матрица порядка n, разбивается на 4 клетки, каждая из которой тоже представляет собой матрицу (см. рис.1).

$$S = \begin{array}{|c|c|}\hline A & B \\ \hline C & D \\ \hline \end{array} \qquad S^{-1} = \begin{array}{|c|c|}\hline K & L \\ \hline M & N \\ \hline \end{array}$$

Рис. 1. Схема разбивки матрицы на клетки.

Квадратные матрицы A и D в общем случае имеют порядки p и q, причем p+q=n. Матрицы B,C тогда в общем случае будут прямоугольными. S^{-1} – искомая обращенная матрица, в свою очередь состоящая из квадратных клеток K, N и прямоугольных клеток L, M. Если мы найдем матрицы K,L,M,N, то мы найдем и обращенную матрицу S^{-1}. Как известно, умножение прямой и обратной матриц должно дать единичную матрицу. При умножении клеточных матриц искомые клетки находятся на основе этого правила и вопрос обращения матрицы порядка n= p+q можно свести к обращению двух матриц порядка p и q. Выражения для вычисления искомых клеток [1].

$$\begin{aligned} N&=(D-CA^{-1}B)^{-1} \\ M&=-NCA^{-1} \\ L&=-A^{-1}BN \\ K&=A^{-1}-A^{-1}BM \end{aligned} \qquad (1)$$

Как видно, при этом обращенные матрицы A^{-1} и $(D-CA^{-1}B)^{-1}$. Первая матрица до обращения представляет собой первый квадрант исходной матрицы. Из физического смысла задач математической физики этот квадрант никогда не будет необращаемой матрицей. Если вторая матрица в ходе решения стала необращаемой, можно проверить еще один вариант формул.

$$\begin{aligned} K&=(A-BD^{-1}C)^{-1} \\ L&=-KBD^{-1} \\ M&=-D^{-1}CK \\ N&=D^{-1}-D^{-1}CL \end{aligned} \qquad (2)$$

Как видно, при этом обращенные матрицы D^{-1} и $(A-BD^{-1}C)^{-1}$. Первая матрица до обращения представляет собой четвертый квадрант исходной матрицы. Из физического смысла задач математической физики этот квадрант никогда не будет необращаемой матрицей. Если вторая матрица в ходе решения стала необращаемой, то необходимо проверить исходные данные на целостность. Причем это сообщение об ошибке в ряде случаев выходит уже на ранних этапах вычислений, что экономит ресурсы для обнаружения ошибки в исходных данных.

Размер матрицы зависит от задачи. Существующие клеточные методы обращения матриц применяют разбиение на большее число клеток с резким усложнением математического аппарата.

Наиболее эффективным было бы рекурсивное разбиение матрицы, то есть каждая клетка в свою очередь разбивается на 4 клетки до тех пор, пока размер клетки не станет настолько малым, что ее обращение не станет проблемой для одного вычислительного узла. Тогда математический алгоритм для обращения каждой клетки был бы подобен алгоритму обращения всей матрицы и самыми сложными математическими действиями стали бы всего лишь умножение матрицы на матрицу и вычитание матриц (см. рис.2).

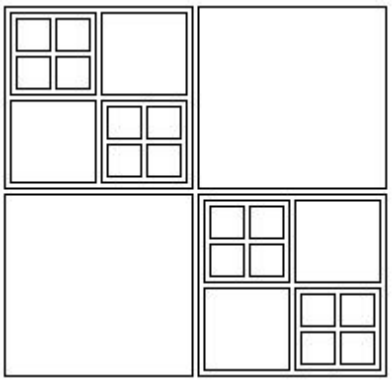

Рис. 2. Схема рекурсивной разбивки матрицы на клетки.

Можно даже в принципе размер конечной обращаемой матрицы довести до единицы, то есть матрица вырождается в ненулевой скаляр, обращение которого есть деление единицы на этот скаляр.

Существующие клеточные методы пытаются обратить матрицу заданного размера, хотя этот размер и не всегда бывает удобен для обработки, что приводит к нагромождению формул в реализации алгоритма и в свою очередь к высокой стоимости программ. Так, в многопроцессорных версиях зарубежных программных комплексов с увеличением числа процессоров, используемых при вычислениях, стоимость программного комплекса увеличивается линейно.

Мы попробуем использовать хитрость – подкорректировать размер матрицы так, чтобы ее можно было легко разбивать на клетки. Такая же хитрость и используется в алгоритме быстрых преобразований Фурье – количество точек дискретизации равняется 2^n. Дополнительные точки вводят искусственно, увеличение количества обрабатываемых точек с лихвой компенсируется общим улучшением алгоритма. В нашем случае сторону матрицы примем за 2^n. Всегда матрицу можно дополнить, получив большую матрицу, в которой диагональные члены равны 1, остальные ровны 0 (см. рис.3).

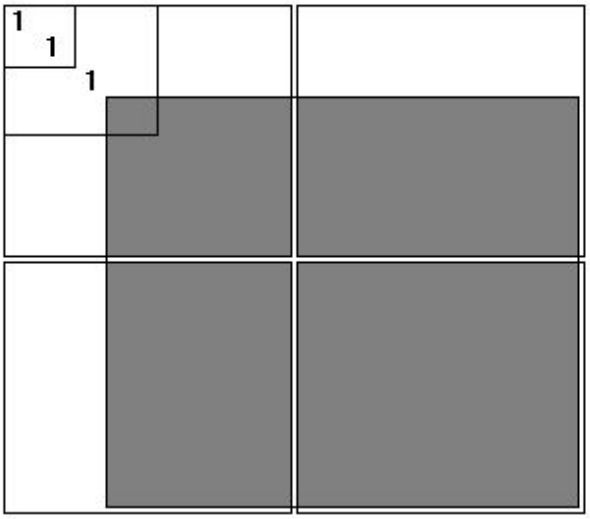

Рис. 3. Схема дополнения матрицы.

На рисунке серым цветом указана матрица размером, не равным 2^n. Ее всегда можно вписать в матрицу размером, равным 2^n, которая может делится на 4 клетки многократно вплоть до клетки размером 1, то есть до скаляра. Обращение скалярного значения есть единица, деленная на это скалярное значение. Первые несколько этапов рекурсии можно пройти, вписав матрицу размером 2^m , ($m<n/2$), в левый верхний угол увеличенной матрицы. Эта матрица размером 2^m всегда будет единичной матрицей, а обращение единичной матрицы нам всегда дает единичную матрицу.

Так, например, для матрицы

1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
		3	4	4	2	1	6
0	0	4	5	3	2	1	7
		4	5	5	3	3	4
0	0	3	4	6	3	4	3
		8	7	6	4	2	4
0	0	6	6	5	5	3	3
		7	6	3	4	8	7
0	0	8	5	2	7	7	7
		9	8	7	6	5	4
0	0	9	8	7	6	5	4
		6	2	4	3	2	1
0	0	6	2	4	3	2	1

обратная матрица будет

1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	-0.01385	-0.0025	0.035391	0.010771	-0.03388	0.015883
0	0	-0.01988	0.009122	0.032067	0.001759	-0.00507	-0.03196
0	0	-0.01445	0.054098	0.005987	-0.00568	-0.0313	0.018054
0	0	0.070627	-0.09493	-0.10074	-0.02517	0.129787	-0.00819
0	0	-0.02867	0.040077	0.00657	0.013518	-0.0266	0.004015
0	0	0.026092	-0.01498	-0.01134	0.003042	0.002437	0.003846

в то время, как для матрицы

34	45	43	22	11	67
43	54	56	33	34	43
86	76	65	45	23	43
78	65	32	47	87	77
99	88	77	66	55	44
66	22	44	33	22	11

обратная матрица будет

		0.03539	0.01077	-0.0338	0.01588
-0.01385	-0.0025	1	1	8	3
	0.00912	0.03206	0.00175	-0.0050	-0.0319
-0.01988	2	7	9	7	6

	0.05409	0.00598	-0.0056		0.01805
-0.01445	8	7	8	-0.0313	4
	-0.0949	-0.1007	-0.0251	0.12978	-0.0081
0.070627	3	4	7	7	9
	0.04007		0.01351		0.00401
-0.02867	7	0.00657	8	-0.0266	5
	-0.0149	-0.0113	0.00304	0.00243	0.00384
0.026092	8	4	2	7	6

Можно легко проверить, например, в программе Excel (tm).

Для того, чтобы было удобно реализовывать этот алгоритм, клетки матрицы должны записываться на носитель данных в одной строго определенной последовательности, например, ABCD. И каждая клетка в свою очередь тоже должна записываться в виде ABCD вплоть до вырождения в скаляр. Так, матрица

$$\begin{vmatrix} & 0.0040 \\ -0.0266 & 15 \\ 0.0024 & 0.0038 \\ 37 & 46 \end{vmatrix}$$

должна записываться в последовательности

-0.0266; 0.004015; 0.002437; 0.003846.

А матрица

$$\begin{vmatrix} 0.00598 & -0.0056 & & 0.01805 \\ 7 & 8 & -0.0313 & 4 \\ -0.1007 & -0.0251 & 0.12978 & -0.0081 \\ 4 & 7 & 7 & 9 \\ & 0.01351 & & 0.00401 \\ 0.00657 & 8 & -0.0266 & 5 \\ -0.0113 & 0.00304 & 0.00243 & 0.00384 \\ 4 & 2 & 7 & 6 \end{vmatrix}$$

должна записываться в последовательности

0.005987; -0.00568; -0.10074; -0.02517; -0.0313; 0.018054; 0.129787; -0.00819; 0.00657; 0.013518;

-0.01134; 0.003042; -0.0266; 0.004015; 0.002437; 0.003846.

Для вычисления индекса элемента матрицы в указанном порядке можно вывести однозначную зависимость, а для чтения и записи клетки любого размера можно однозначно вычислить позицию в последовательном потоке данных.

В качестве примеров мы привели маленькие матрицы, чтобы была понятна суть. В обычных задачах уже идут матрицы размером миллионы на миллионы, а в усложненных задачах миллиарды на миллиарды.

Очень часто в задачах математической физики возникают разреженные матрицы. В нашем случае для разреженной матрицы можно простым способом экономить действия процессора и память, так как при этом у нас будут только действия по умножению и вычитанию матриц. Умножение числа на ноль и добавление к другому числу можно не выполнять. Хотя на листе бумаги это действие выглядит простым, для процессора оно занимает столько же времени, сколько

и для полностью ненулевых чисел.

Для оптимизации хранения разреженной матрицы в нашем случае ячейки, заполненные нулями, можно не хранить, поставив вместо них флаг. Такой способ хранения разреженной матрицы намного выгодней существующих.

Таким образом, мы получили легко распараллеливаемый алгоритм обращения матриц произвольного размера. При резком увеличении размера матрицы достаточно будет увеличить число вычислительных узлов с сохранением той же программы. Как мы видим, с ростом размера матрицы имеется линейный, а не квадратичный или кубический рост затрат ресурсов. Может быть для маленьких матриц этот способ и не даст особого преимущества, поэтому его, наверное, не нашли Гаусс и Крамер, но для очень больших матриц наступит такой момент, когда для существующих методов обращения матриц с квадратичным и даже кубическим ростом расхода ресурсов, указанных способ будет оптимальней существующих. Для оптимальной нагрузки вычислительной системы количество вычислительных узлов одного ранга также должно равняться 2^n.

Как Чарльз Хоар назвал найденный им самый быстрый из известных человечеству алгоритмов сортировки просто алгоритмом быстрой сортировки, назовем и этот алгоритм просто алгоритмом быстрого обращения матриц.

Реализация этого алгоритма уже не представляет сложности, что позволяет своими силами разрабатывать крупные промышленные вычислительные комплексы, позволяющие моделировать и разрабатывать высокотехнологичную промышленную продукцию, собирая простым способом суперкомпьютеры из множества обычных дешевых компьютеров, и таким образом догнать и перегнать развитые страны.

Литература:

1. Фаддеев Д.К., Фаддеева В.Н. Вычислительные методы линейной алгебры.- Изд. 2-е.- М.: «Наука», 1963. стр. 184.