В. Д. Ширяев, Масштабируемость эллипсоидального метода оценивания множеств достижимости линейных систем,
Программные системы: теория и приложения, 2011, том 2, выпуск 2, 53–60

ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ № 2(6), 2011, с. 53–60

УДК 519.688

В. Д. Ширяев

Масштабируемость эллипсоидального метода оценивания множеств достижимости линейных систем

Аннотация. Статья посвящена исследованию масштабируемости эллипсоидального метода оценивания множеств достижимости линейных систем. Приводятся измерения времени работы реализаций.

Ключевые слова и фразы: эллипсоидальный метод, множество достижимости, масштабируемость.

Введение

Рассматривается система дифференциальных уравнений

$$\begin{aligned}&\dot{x}(t) = A(t)x(t) + u(t),\\ &u(t) \in \mathcal{P}(t) = \mathcal{E}(P(t), p(t)) \text{ — управление},\\ &x(t_0) \in \mathcal{X}^0 = \mathcal{E}(X^0, x^0) \text{ — начальное множество.}\end{aligned}$$

$\mathcal{E}(P(t), p(t))$ обозначает эллипсоид с матрицей $P(t)$ и центром $p(t)$, строгое определение дано ниже. Все коэффициенты непрерывны. Необходимо найти множество достижимости системы, т. е. все точки, в которые можно привести фазовую точку с помощью управления при указанных ограничениях, варьируя как угодно начальное положение точки в начальном множестве $\mathcal{X}^0$ и выбирая какое угодно кусочно непрерывное управление, удовлетворяющее для всех t требованию $u(t) \in \mathcal{P}(t)$.

Задача решается с помощью эллипсоидального метода оценивания множества достижимости, а именно, будут строиться внешние оценки. Теория эллипсоидального метода приведена в [1]. Существует реализация для `MATLAB` ([2]).

Работа выполнена в рамках ФЦП «Научные и научно-педагогические кадры инновационной России на 2009–2013 годы» (контракт №16.740.11.0426 от 26 ноября 2010 года).

Для описания алгоритма необходимы определения:

Определение 1. *Пусть A — выпуклое компактное множество, а l — некоторый вектор пространства.* ***Опорной функцией множества A в направлении l*** *называется функция*

$$\rho\left(A \mid l\right) = \sup_{x \in A} \langle l, x \rangle .$$

Определение 2. *Пусть Q — неотрицательно определённая матрица.* ***Эллипсоидом $\mathcal{E}(Q, q)$ с центром q и матрицей Q*** *называется множество, такое что*

$$\rho\left(\mathcal{E}(Q,q) \mid l\right) \equiv \langle l, q \rangle + \langle l, Ql \rangle^{\frac{1}{2}} .$$

Строятся внешние оценки множества достижимости, каждая из них — это эллипсоид $\mathcal{E}(X_+(t,l_0), x_+(t))$, зависящий от времени и векторного параметра l_0, размерность которого равна размерности фазового пространства. Каждая внешняя оценка обладает важными свойствами:

- оценка включает в себя точное множество достижимости,
- опорная функция оценки $\mathcal{E}(X_+(t,l_0), x_+(t))$ совпадает с опорной функцией точного множества достижимости в момент времени t вдоль вектора $l(t,l_0)$, определяемого задачей Коши

$$\dot{l}(t,l_0) = -A^T(t) l(t,l_0),\ l(t_0,l_0) = l_0. \tag{1}$$

  Это значит, что оценка $\mathcal{E}(X_+(t,l_0), x_+(t))$ касается точного множества в направлении $l(t,l_0)$ в момент времени t.

Так что для получения точного множества достижимости нужно пересечь все внешние эллипсоидальные оценки, то есть оценки, построенные для всех значений l_0 на единичной сфере.

Матрица и центр оценки получаются как решения задачи Коши

$$\begin{cases} \dot{X}_+(t,l_0) = A(t)X_+(t,l_0) + (A(t)X_+(t,l_0))^T + \\ \qquad\qquad + \pi(t)X_+(t,l_0) + \dfrac{1}{\pi(t)}P(t),\ X_+(0,l_0) = X^0, \\ \dot{x}_+(t) = A(t)x_+(t) + p(t),\ x_+(0) = x^0, \\ \pi(t) = \sqrt{\dfrac{\langle l(t,l_0), P(t)l(t,l_0)\rangle}{\langle l(t,l_0), X_+(t,l_0)l(t,l_0)\rangle}}. \end{cases} \tag{2}$$

1. Анализ путей распараллеливания

Всякая эллипсоидальная оценка вычисляется независимо от других. Значит, каждый процессор, получив параметры задачи и значение $l(t_0, l_0)$, может вести вычисления совершенно независимо от других — это можно делать параллельно. Это — первая схема распараллеливания, и далее будут приведены результаты работы реализации, использующей этот подход.

Вычисление каждой эллипсоидальной оценки — это решение задачи Коши (для матриц). При решении задачи Коши может понадобиться многократное вычисление правой части дифференциальных уравнений системы (2). Несложно обнаружить, что в правых частях уравнений системы (2) наиболее тяжёлая операция — это перемножение матриц. Остальные операции либо гораздо проще по числу операций, либо распараллеливаются очевидным образом, и их масштабируемость не вызывает сомнений. Так что интерес представляет только масштабируемость операции перемножения двух матриц. Оказывается, эта операция тоже отлично масштабируется, и ниже будет описана соответствующая реализация и даны результаты её работы.

2. Параллелизм на уровне разных эллипсоидальных оценок

Таблица 1 содержит измерения времени работы реализации первой схемы распараллеливания, таблица 2 — затраты процессорного времени (по вертикали отложено число используемых процессоров, по горизонтали — количество вычисляемых эллипсоидальных оценок), а 3 — ускорение (число процессоров отложено по горизонтали).

Реализация выполнена для машины BlueGene/P, установленной на факультете ВМиК МГУ имени М. В. Ломоносова, на языке `C` с использованием библиотеки `MPI` для массового параллелизма и `ESSL` для матричных операций на каждом вычислительном узле. Схема работы следующая: сначала по всем вычислительным узлам рассылаются параметры задачи и начальное значение $l(t_0, l_0) = l_0$, после чего каждый узел решает задачу Коши (2). Затем матрицы-ответы $X_+(t, l_0)$ собираются со всех узлов на одном.

Таблицы демонстрируют линейное ускорение решения задачи, что свидетельствует об отличной масштабируемости алгоритма на уровне построения разных эллипсоидальных оценок.

Таблица 1. Время работы программы

	512	1024	2048	4096	8192	16384	32768
1	382.1	763.0					
4	95.28	181.0	381.8	764.4			
8	48.15	91.36	182.7	365.3	766.8		
16	23.91	45.47	90.89	181.8	363.5	765.7	
32	11.97	22.78	45.51	91.02	182.0	364.0	765.2
64	05.99	11.41	22.75	45.50	90.99	182.0	363.9
128	02.99	05.96	11.38	22.75	45.48	90.97	181.9
256	01.50	02.86	05.70	11.38	22.76	45.50	91.02
512	00.75	01.49	02.86	05.70	11.39	22.77	45.52

Таблица 2. Затраты машинного времени в процессоро-секундах

	512	1024	2048	4096	8192	16384	32768
1	382.1	763.0					
4	381.1	723.9	1527.0	3057			
8	385.2	730.8	1461.4	2922	6134		
16	382.6	727.6	1454	2908	5817	12251	
32	383.0	728.9	1456	2913	5824	11647	24484
64	383.1	730.0	1456	2912	5823	11646	23291
128	383.2	763.4	1456	2911	5822	11644	23285
256	383.4	733.2	1460	2914	5825	11649	23300
512	383.7	761.8	1462	2918	5830	11656	23307

Таблица 3. Ускорение

	1	4	8	16	32	64	128	256	512
512	1.00	4.01	7.94	16.0	31.9	63.8	128	255	510
1024	1.00	4.27	8.35	16.8	33.5	66.9	128	266	513

3. Параллелизм на уровне одной эллипсоидальной оценки

Масштабируемость алгоритма на уровне одной оценки определяется масштабируемостью операции перемножения двух матриц. В условиях рассматриваемой задачи эти матрицы всегда квадратные, что не ограничивает общности описываемых далее методов, но упрощает их описание.

3.1. Алгоритм Кэннона перемножения матриц

Для перемножения двух матриц используется метод Кэннона перемножения матриц. Он заключается в следующем. Пусть имеется k^2 вычислительных узлов, k — натуральное число, и пусть узлы соединены коммуникационной сетью, образующей топологию двумерного тора. То есть, если k^2 узлов обозначать символом $N(i,j)$, $0 \leqslant i,j \leqslant k-1$, то узел $N(i,j)$ может быстро обмениваться данными с четырьмя узлами $N(i \pm 1 \mod k, j \pm 1 \mod k)$.

Пусть теперь матрица A умножается на матрицу B, и размеры матриц делятся на k. Матрицы разбиваются на подматрицы соответственно топологии вычислительных узлов, то есть, на k^2 одинаковых по размеру подматриц:

$$A = \begin{pmatrix} A(0,0) & A(0,1) & \cdots & A(0,k-1) \\ A(1,0) & A(1,1) & \cdots & A(1,k-1) \\ \vdots & \vdots & \ddots & \vdots \\ A(k-1,0) & A(k-1,1) & \cdots & A(k-1,k-1) \end{pmatrix},$$

$$B = \begin{pmatrix} B(0,0) & B(0,1) & \cdots & B(0,k-1) \\ B(1,0) & B(1,1) & \cdots & B(1,k-1) \\ \vdots & \vdots & \ddots & \vdots \\ B(k-1,0) & B(k-1,1) & \cdots & B(k-1,k-1) \end{pmatrix}.$$

Затем происходит построение матриц $A[0]$ и $B[0]$: блок $A[0](i,j)$ матрицы $A[0]$ — это блок $A(i, j+i \mod k)$, а блок $B[0](i,j)$ матрицы $B[0]$ — это блок $B(i+j \mod k, j \mod k)$. Наглядно это означает следующее: первый столбец $A[0]$ — это главная диагональ A, а первая строка $B[0]$ — это главная диагональ B. Заранее определим матрицы $A[s]$ и $B[s]$ следующим образом: блок $A[s](i,j)$ матрицы $A[s]$ — это блок $A(i, j+i+s \mod k)$, а блок $B[s](i,j)$ матрицы $B[s]$ — это блок $B(i+j+s \mod k, j \mod k)$. Иными словами, блоки матрицы $A[s]$ —

это сдвинутые на s позиций влево циклическим образом блоки матрицы $A[0]$, а блоки $B[s]$ — это сдвинутые на s позиций вверх циклическим образом блоки матрицы $B[0]$.

На узле (i, j) будет получаться блок $C(i, j)$ матрицы-произведения A и B. Начальное значение блока $C(i, j)$ — нулевая квадратная матрица со стороной k. Кроме того, на узле (i, j) будет всегда храниться по одному блоку матриц A и B. Начальное положение таково: узел (i, j) хранит блоки $A[0](i, j)$ и $B[0](i, j)$.

Собственно алгоритм Кэннона состоит из k шагов: на n-том шаге, $0 \leqslant n \leqslant k - 1$, происходит следующее:

(1) узел (i, j) получает блоки $A[n](i, j)$ и $B[n](i, j)$ (от своих соседей по сети в топологии двумерного тора),
(2) узел (i, j) перемножает хранящиеся на нём блоки и добавляет это произведение к хранящемуся на нём блоку $C(i, j)$.

После выполнения последнего шага $n = k-1$ узел (i, j) будет хранить блок $C(i, j)$. Матрица $C = AB$ из этих блоков получается так же, как на них разбивались A и B:

$$C = \begin{pmatrix} C(0,0) & C(0,1) & \cdots & C(0,k-1) \\ C(1,0) & C(1,1) & \cdots & C(1,k-1) \\ \vdots & \vdots & \ddots & \vdots \\ C(k-1,0) & C(k-1,1) & \cdots & C(k-1,k-1) \end{pmatrix}.$$

Обратим внимание на то, что для выполнения этапа, на котором узел (i, j) получает блоки множителей, достаточно получения узлом данных от пары своих соседей и отправки данных другой паре своих соседей по сети в топологии двумерного тора. Последнее обстоятельство говорит о том, что алгоритм хорошо приспособлен к машинам с соответствующими возможностями коммуникационной сети — время коммуникационного шага алгоритма состоит из времени максимум четырёх пересылок одного блока матрицы, хотя через сеть на каждом шаге оба множителя проходят целиком.

Таблица 4 содержит измерения времени работы реализации этого метода умножения матриц, таблица 5 — затраты процессорного времени, а 6 — ускорение. По горизонтали откладывается число используемых процессоров, а по вертикали — размер стороны матрицы-сомножителя. Реализация сделана для BlueGene/P, на языке `C` с применением библиотеки `MPI`. В этот раз не использовалась библиотека

ESSL, использование которой позволяет сократить время работы программы в несколько раз (не меньше, чем в 6). Это обстоятельство, однако, влияет на масштабируемость не принципиально.

Таблица 4. Время работы программы

	1	4	16	36	64	100	144	196	256	324	361	400	441	484	1024
512	1.355	0.348	0.088	0.047	0.024	0.014	0.011	0.008	0.006	0.006	0.005	0.005	0.005	0.004	0.003
1024	10.84	2.745	0.691	0.324	0.178	0.122	0.099	0.069	0.049	0.034	0.031	0.028	0.025	0.024	0.013
2048	86.75	21.82	5.485	2.608	1.383	0.925	0.664	0.500	0.356	0.310	0.277	0.246	0.221	0.230	0.104
4096	694.4	174.1	43.63	19.99	11.00	7.408	5.277	3.770	2.745	2.234	2.104	1.866	1.653	1.604	0.741
8192			348.1	159.0	87.38	57.95	40.21	29.63	22.02	17.91	16.24	14.82	13.23	12.36	5.688

Таблица 5. Затраты машинного времени в процессоро-секундах

	1	4	16	36	64	100	144	196	256	324	361	400	441	484	1024
512	1.355	1.390	1.410	1.690	1.570	1.390	1.550	1.530	1.450	1.810	1.920	1.930	2.040	2.140	3.240
1024	10.84	10.98	11.06	11.66	11.39	12.25	14.19	13.56	12.54	11.04	11.21	11.24	11.21	11.38	13.04
2048	86.75	87.29	87.76	93.88	88.51	92.45	95.62	98.09	91.08	100.6	100.2	98.49	97.44	111.5	106.9
4096	694.4	696.3	698.0	719.5	704.2	740.8	759.9	738.9	702.7	723.8	759.4	746.2	729.2	776.3	758.8

Таблица 6. Ускорение

	1	4	16	36	64	100	144	196	256	324	361	400	441	484	1024
512	1.000	3.900	15.35	28.86	55.39	97.58	125.6	173.1	240.0	242.8	254.8	280.6	292.4	307.1	428.0
1024	1.000	3.950	15.69	33.49	60.94	88.52	110.0	156.7	221.4	318.2	349.2	385.8	426.5	461.3	851.7
2048	1.000	3.980	15.82	33.27	62.73	93.84	130.7	173.4	243.8	279.8	312.7	352.3	392.6	376.8	830.7
4096	1.000	3.990	15.92	34.74	63.11	93.74	131.6	184.2	253.0	310.8	330.1	372.2	420.0	433.0	937.1

По данным таблиц видно, что пусть ускорение всегда при увеличении числа используемых вычислительных узлов хорошо заметно и на некоторых размерах задач линейно, для меньших размеров задач использование очень большого количества процессоров приводит к падению эффективности.

Заключение

Приведённые факты свидетельствуют о хорошей масштабируемости эллипсоидального метода оценивания множеств достижимости линейных систем как на уровне построения разных оценок, так и на уровне построения одной оценки.

Список литературы

[1] Kurzhanski A. B., Valyi I. Ellipsoidal Calculus for Estimation and Control. Boston : Birkhauser, 1997. ↑[]

[2] Kurzhanski A. A., Varaiya P. Ellipsoidal Toolbox, 2006, http://code.google.com/p/ellipsoids. ↑[]

V. D. Shiryaev. *A study on scalability of the ellipsoidal method of linear systems' reachability estimation.*

Abstract. The article is devoted to reseach of scalability of the ellipsoidal method of linear systems' reachability estimation. The article contains benchmarking results.

Key Words and Phrases: the ellipsoidal method, reach set, reachability, scalability.

Образец ссылки на статью:

В. Д. Ширяев. *Масштабируемость эллипсоидального метода оценивания множеств достижимости линейных систем* // Программные системы: теория и приложения : электрон. научн. журн. 2011. № 2(6), с. 53–60. URL: http://psta.psiras.ru/read/psta2011_2_53-60.pdf