Математические структуры и моделирование
2010, вып. 21, с. 24–31

УДК 519.688

МНОГОМЕРНЫЕ ЛИНЕЙНЫЕ МНОГООБРАЗИЯ КАК СПОСОБ ВОССТАНОВЛЕНИЯ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

И.Б. Ларионов

Рассматривается метод восстановления графической информации с применением многомерных линейных многообразий.

Введение

В данной работе рассматривается метод восстановления графической информации с использованием многомерных линейных многообразий. Ранее в работе [2] рассматривался метод сингулярного разложения одномерным линейным многообразием.

Решение будет приведено явными формулами для двумерного и трехмерного случаев, а также будут приведены формулы для n-мерного случая.

1. Многомерные линейные многообразия

Для построения линейных многообразий размерности больше единицы используются совершенно аналогичные формулы. Но так как в составе одного многообразия размерности больше единицы, задающие его векторы могут быть не ортогональными, то для облегчения интерпретации полученных факторов рекомендуется проводить соответствующую ортогонализацию.

Рассмотрим двумерные и трехмерные линейные многообразия, для которых опишем процедуры построения и ортогонализации, а также приведем формулы для размерности n.

Однако хочется заметить, что, как будет показано ниже, использование многомерных линейных многообразий не принесет никакого выигрыша ни в качестве, ни в скорости в применении к восстановлению графической информации.

1.1. Ортогонализация базисной системы векторов

Пусть дано подпространство размерности n, и в нем система линейно независимых векторов $\{y_k\}$, $k = 1, ..., n$. По определению, они образуют базис этого

Омский государственный университет им. Ф.М. Достоевского.
E-mail: g0gich@mail.ws

подпространства. Требуется провести процедуру ортогонализации этого базиса, т.е. построить такой новый базис этого подпространства $\{\overline{y}_k\}(k = 1, ..., n)$, чтобы $(\overline{y}_j, \overline{y}_j) = 0$ для любых $i \neq j$.

Для начала решим следующую задачу.

Пусть набор векторов $\{y_k\}(k = 1, ..., n-1)$ уже ортогонализирован, а y_n – еще нет. Тогда построим новый вектор $\overline{y}_n = \sum_{i=1}^{n-q} \alpha_i y_i + y_n$ и потребуем, чтобы $(\overline{y}_n, y_k) = 0$ для всех $k = 1, ..., n-1$. В результате имеем систему из $n-1$ уравнения с $n-1$ неизвестными:

$$\big(\sum_{i=1}^{n} \alpha_i y_i + y_n, y_k\big) = 0, k = 1, ..., n-1.$$

Учитывая, что $(y_i, y_k) = 0$ при $i \neq k$, то легко заметить, что полученная система будет диагональной. Отсюда неизвестные легко находятся по формуле

$$\alpha_k = -\frac{(y_n, y_k)}{y_k, y_k}.$$

Следовательно, если задана система из n векторов, в которой имеется ортогональная подсистема из $n-1$ вектора, то оставшийся вектор «ортогонализируется» по формуле

$$\overline{y}_n = y_n - \sum_{i=1}^{n-1} \frac{(y_n, y_i)}{(y_i, y_i)} y_i. \tag{1}$$

Тогда процедура ортогонализации системы из n базисных векторов выглядит как последовательная ортогонализация систем из $2, 3, \ldots, n$ векторов с использованием формулы 1.

1.2. Двумерные линейные модели

Для построения двумерного линейного многообразия минимизируем квадратичную форму:

$$\Phi = \sum_{\substack{i,j \\ a_{ij} \neq @}} (a_{ij} - x_{1i} y_{1j} - x_{2i} y_{2j} - b_j) \to \min, \tag{2}$$

где @ – пропущенные элементы. Решение дается последовательными итерациями по явным формулам. При фиксированных y_{1j}, y_{2j} и b_j значения x_{1i} и x_{2i} однозначно находятся из системы равенств $\partial\Phi/\partial x_{1i} = 0$ и $\partial\Phi/\partial x_{2i} = 0$:

$$\begin{cases} \partial\Phi/\partial x_{1i} = -2 \sum_{j,\ a_{ij} \neq @} (a_{ij} - x_{1i} y_{1j} - x_{2i} y_{2j} - b_j) y_{1j} = 0, \\ \partial\Phi/\partial x_{2i} = -2 \sum_{j,\ a_{ij} \neq @} (a_{ij} - x_{1i} y_{1j} - x_{2i} y_{2j} - b_j) y_{2j} = 0. \end{cases}$$

То же самое в векторном виде:

$$\begin{cases} x_{1i}(y_1, y_1)_{a_j} + x_{2i}(y_2, y_1)_{a_i} = (a_i - b, y_1)_{a_i}, \\ x_{1i}(y_1, y_2)_{a_j} + x_{2i}(y_2, y_2)_{a_i} = (a_i - b, y_2)_{a_i}. \end{cases}$$

Аналогично при фиксированных x_{1i} и x_{2i} значения, доставляющие минимум квадратичной форме 2, однозначно находятся из равенств $\partial\Phi/\partial y_{1j} = 0$, $\partial\Phi/\partial y_{2j} = 0$ и $\partial\Phi/\partial b_j = 0$:

$$\begin{cases} \partial\Phi/\partial y_{1j} = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - b_j)x_{1i} = 0, \\ \partial\Phi/\partial y_{2j} = -2\sum_{i\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - b_j)x_{2i} = 0, \\ \partial\Phi/\partial b_j = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - b_j) = 0. \end{cases}$$

То же самое в векторной форме:

$$\begin{cases} y_{1j}(x_1, x_1)_{a_j} + y_{2j}(x_2, x_1)_{a_j} + b_j(x_1, 1)_{a_j} = (a_j, x_1)_{a_j}, \\ y_{1j}(x_1, x_2)_{a_j} + y_{2j}(x_2, x_2)_{a_j} + b_j(x_2, 1)_{a_j} = (a_j, x_2)_{a_j}, \\ y_{1j}(x_1, 1)_{a_j} + y_{2j}(x_2, 1)_{a_j} + b_j(1, 1)_{a_j} = (a_j, 1)_{a_j}. \end{cases}$$

Полученная система решается любым численным способом, например методом квадратного корня (т.к. полученная матрица является симметричной). Учитывая, что полученные векторы y_1 и y_2 могут быть не ортогональны, то в некоторых случаях их необходимо ортогонализировать. А так как они образуют базис подпространства размерности 2, то эта задача легко решается с использованием формулы 1.

1.3. Трехмерные линейные модели

Для построения трехмерного линейного многообразия минимизируем квадратичную форму:

$$\Phi = \sum_{\substack{i,j \\ a_{ij}\neq@}} (a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)^2 \to \min. \tag{3}$$

Решение дается последовательными итерациями по явным формулам. При фиксированных y_{1j}, y_{2j}, y_{3j} и b_j значения x_{1i}, x_{2i} и x_{3i} однозначно находятся из системы равенств $\partial\Phi/\partial x_{1i} = 0$, $\partial\Phi/\partial x_{2i} = 0$ и $\partial\Phi/\partial x_{3i} = 0$:

$$\begin{cases} \partial\Phi/\partial x_{1i} = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)y_{1j} = 0, \\ \partial\Phi/\partial x_{2i} = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)y_{2j} = 0, \\ \partial\Phi/\partial x_{3i} = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)y_{3j} = 0. \end{cases}$$

То же самое в векторной форме:

$$\begin{cases} x_{1i}(y_1, y_1)_{a_j} + x_{2i}(y_2, y_1)_{a_j} + x_{3i}(y_3, y_1)_{a_j} = (a_j - b, y_1)_{a_j}, \\ x_{1i}(y_1, y_2)_{a_j} + x_{2i}(y_2, y_2)_{a_j} + x_{3i}(y_3, y_2)_{a_j} = (a_j - b, y_2)_{a_j}, \\ x_{1i}(y_1, y_3)_{a_j} + x_{2i}(y_2, y_3)_{a_j} + x_{3i}(y_3, y_3)_{a_j} = (a_j - b, y_3)_{a_j}. \end{cases}$$

Аналогично при фиксированных x_{1i}, x_{2i} и x_{3i} значения, доставляющие минимум квадратичной форме 3, однозначно находятся из равенств $\partial\Phi/\partial y_{1j} = 0$, $\partial\Phi/\partial y_{2j} = 0$, $\partial\Phi/\partial y_{3j} = 0$ и $\partial\Phi/\partial b_j = 0$:

$$\begin{cases} \partial\Phi/\partial y_{1j} = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)x_{1i} = 0, \\ \partial\Phi/\partial y_{2j} = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)x_{2i} = 0, \\ \partial\Phi/\partial y_{2j} = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)x_{2i} = 0, \\ \partial\Phi/\partial b_j = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j) = 0. \end{cases}$$

То же самое в векторной форме:

$$\begin{cases} y_{1j}(x_1, x_1)_{a_j} + y_{2j}(x_2, x_1)_{a_j} + y_{3j}(x_3, x_1)_{a_j} + b_j(x_1, 1)_{a_j} = (a_j - b, x_1)_{a_j}, \\ y_{1j}(x_1, x_2)_{a_j} + y_{2j}(x_2, x_2)_{a_j} + y_{3j}(x_3, x_2)_{a_j} + b_j(x_2, 1)_{a_j} = (a_j - b, x_2)_{a_j}, \\ y_{1j}(x_1, x_3)_{a_j} + y_{2j}(x_2, x_3)_{a_j} + y_{3j}(x_3, x_3)_{a_j} + b_j(x_3, 1)_{a_j} = (a_j - b, x_3)_{a_j}, \\ y_{1j}(x_1, 1)_{a_j} + y_{2j}(x_2, 1)_{a_j} + y_{3j}(x_3, 1)_{a_j} + b_j(1, 1)_{a_j} = (a_j - b, 1)_{a_j}. \end{cases}$$

Полученная система решается любым численным способом, например методом квадратного корня (т.к. полученная матрица является симметричной).

Учитывая, что полученные векторы y_1, y_2 и y_3 могут быть не ортогональны, то в некоторых случаях их необходимо ортогонализировать. А так как они образуют базис подпространства размерности 3, то эта задача легко решается с использованием формулы 1.

1.4. Линейная модель произвольной размерности

Для построения линейного многообразия произвольной размерности минимизируем квадратичную форму:

$$\Phi = \sum_{\substack{i,j \\ a_{ij}\neq@}} (a_{ij} - \sum_n x_{ni}y_{nj} - b_j)^2 \to \min, \tag{4}$$

где n - размерность линейного многообразия.

Решение дается последовательными итерациями по явным формулам. При фиксированных y_{nj} и b_j значения x_{ni} однозначно находятся из системы равенств $\partial\Phi/\partial x_{ni} = 0$:

$$\begin{cases} \partial\Phi/\partial x_{1i} = -2\sum_{j,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j)y_{1j} = 0, \\ \partial\Phi/\partial x_{2i} = -2\sum_{j,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j)y_{2j} = 0, \\ \ldots \\ \partial\Phi/\partial x_{ni} = -2\sum_{j,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j)y_{nj} = 0. \end{cases}$$

То же самое в векторном виде:

$$\begin{cases} x_{1i}(y_1, y_1)_{a_j} + x_{2i}(y_2, y_1)_{a_i} + \cdots + x_{ni}(y_n, y_1)_{a_i} = (a_i - b, y_1)_{a_i}, \\ x_{1i}(y_1, y_2)_{a_j} + x_{2i}(y_2, y_2)_{a_i} + \cdots + x_{ni}(y_n, y_2)_{a_i} = (a_i - b, y_2)_{a_i}, \\ \ldots \\ x_{1i}(y_1, y_n)_{a_j} + x_{2i}(y_2, y_n)_{a_i} + \cdots + x_{ni}(y_n, y_n)_{a_i} = (a_i - b, y_n)_{a_i}. \end{cases}$$

Аналогично при фиксированных x_{ni} значения, доставляющие минимум квадратичной форме 4, однозначно находятся из равенств $\partial\Phi/\partial y_{nj} = 0$ и $\partial\Phi/\partial b_j = 0$:

$$\begin{cases} \partial\Phi/\partial y_{1j} = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j)x_{1i} = 0, \\ \partial\Phi/\partial y_{2j} = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j)x_{2i} = 0, \\ \cdots \\ \partial\Phi/\partial y_{nj} = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j)x_{ni} = 0, \\ \partial\Phi/\partial b_j = -2\sum_{i,\ a_{ij}\neq@}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j) = 0. \end{cases}$$

То же самое в векторной форме:

$$\begin{cases} y_{1j}(x_1, x_1)_{a_j} + y_{2j}(x_2, x_1)_{a_j} + \cdots + y_{nj}(x_n, x_1)_{a_j} + b_j(x_1, 1)_{a_j} = (a_j, x_1)_{a_j}, \\ y_{1j}(x_1, x_2)_{a_j} + y_{2j}(x_2, x_2)_{a_j} + \cdots + y_{nj}(x_n, x_2)_{a_j} + b_j(x_2, 1)_{a_j} = (a_j, x_2)_{a_j}, \\ \cdots \\ y_{1j}(x_1, x_n)_{a_j} + y_{2j}(x_2, x_n)_{a_j} + \cdots + y_{nj}(x_n, x_n)_{a_j} + b_j(x_n, 1)_{a_j} = (a_j, x_n)_{a_j}, \\ y_{1j}(x_1, 1)_{a_j} + y_{2j}(x_2, 1)_{a_j} + \cdots + y_{nj}(x_n, 1)_{a_j} + b_j(1, 1)_{a_j} = (a_j, 1)_{a_j}. \end{cases}$$

Полученная система решается любым численным способом, например методом квадратного корня (т.к. полученная матрица является симметричной). Учитывая, что полученные векторы y_n могут быть не ортогональны, то в некоторых случаях их необходимо ортогонализировать. А так как они образуют базис подпространства размерности n, то эта задача легко решается с использованием формулы 1.

2. Применение многомерных линейных многообразий для восстановления графической информации

В ходе работы был реализован класс MatrixSolver, который при заданной размерности многообразия строил вектора x_n, y_n и b и производил итерационное исчерпание матрицы данных, как это описано в работе [2].

Итерационные вычисления проводились согласно приведенным формулам многомерных линейных многообразий, приведенных в векторной форме. Данный подход позволил строить СЛАУ, решение которых не вызывало серьезных технических и временных затрат. Все вычисления на каждой итерации сводились к построению и решению i СЛАУ порядка n и j СЛАУ порядка $n + 1$.

Были проведены испытания для размерностей 2, 3, ..., 10. Исходя из полученных на тот момент результатов были проведены испытания для размерности 100. Все испытания проводились с одним и тем же изображением, имеющем размеры 1024x768 пикселей, что позволяет судить о количестве решенных СЛАУ и их размерности.

Оценка ошибки производилось методом среднего квадратичного [4]. Данная метрика была выбрана как наиболее наглядная и простая для вычисления. Кроме того, в рамках данной работы ключевым моментом является выявление зависимости размерности линейного многообразия и таких его свойств, как время, затраченное на достижение определенной, наперед заданной максимальной ошибки.

Для начала в таблице 1 приведем зависимости ошибки восстановления на первых 10-ти шагах восстановления для разных размерностей.

Таблица 1. Ошибка восстановления при разных размерностях

i/n	2	3	4	5	6
1	250691595,68	113356199,79	64534470,18	41384009,45	27736091,44
2	125345797,84	56678099,89	32267235,09	20692004,72	13868045,72
3	83563865,23	37785399,93	21511490,06	13794669,82	9245363,81
4	62672898,92	28339049,95	16133617,54	10346002,36	6934022,86
5	50138319,14	22671239,96	12906894,04	8276801,89	5547218,29
6	41781932,61	18892699,96	10755745,03	6897334,91	4622681,91
7	35813085,10	16193742,83	9219210,03	5912001,35	3962298,78
8	31336449,46	14169524,97	8066808,77	5173001,18	3467011,43
9	27854621,74	12595133,31	7170496,69	4598223,27	3081787,94
10	25069159,57	11335619,98	6453447,02	4138400,94	2773609,14
i/n	7	8	9	10	100
1	20368692,15	15482141,30	10264537,78	3636251,88	51516,38
2	10184346,07	7741070,65	5132268,89	1818125,94	25758,19
3	6789564,05	5160713,77	3421512,59	1212083,96	17172,13
4	5092173,04	3870535,33	2566134,44	909062,97	12879,10
5	4073738,43	3096428,26	2052907,56	727250,38	10303,28
6	3394782,02	2580356,88	1710756,30	606041,98	8586,06
7	2909813,16	2211734,47	1466362,54	519464,55	7359,48
8	2546086,52	1935267,66	1283067,22	454531,48	6439,55
9	2263188,02	1720237,92	1140504,20	404027,99	5724,04
10	2036869,21	1548214,13	1026453,78	363625,19	5151,64

Далее для наглядности на рисунке 1 приведем график среднеквадратичной ошибки на первых 10-ти итерациях для $n = 2, \ldots, 10, 100$.

В ходе испытаний было выявлено минимальное значение ошибки, которого удалось достичь при восстановлении изображений - 4760. После достижения данного значения при всех рассматриваемых n алгоритм останавливался, так как межшаговое падение ошибки было достаточно малым [2].

Приведем таблицу длительности одной итерации.

Таблица 2. Длительность первой итерации

n	2	3	4	5	6	7	8	9	10	100
Длительность, мс	52	115	202	315	470	640	842	1270	3585	153045

Как видно из этой таблицы, длительность одной итерации пропорциональна квадрату размерности многообразия, что говорит о малом количестве накладных расходов на построение матриц и их решение.

Для оценки эффективности применения линейных многообразий больших порядков рассмотрим изменение ошибки восстановления за единицу времени.

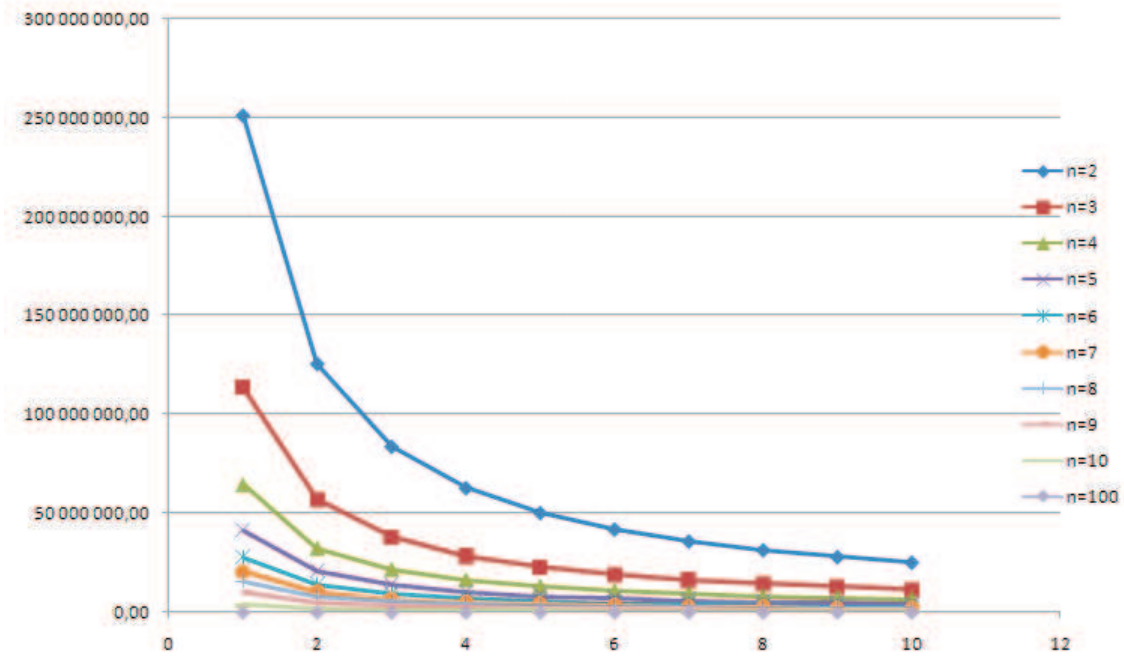

Рис. 1. График среднеквадратичной ошибки на первых 10-ти итерациях

Вычисления производятся по формуле

$$\partial\Phi_i = \frac{\Phi_{i-1} - \Phi_i}{\Delta t},$$

где Φ_i - ошибка восстановления на i-й итерации, а Δt - время выполнения итерации. Сведем данные в таблицу.

Как видно из таблицы 3, эффективность падает не только от итерации к итерации, но и с ростом размерности n. В таблице 4 приведено время, требуемое для остановки алгоритма согласно критериям остановки для каждой размерности.

Выводы

В ходе работы были выведены формулы построения линейного многообразия произвольной размерности и произведены испытания последовательного исчерпания матрицы с пропусками.

Результаты испытаний наглядно показывают, что данный метод может применяться для восстановления графической информации, однако использование больших размерностей не приводит ни к ускорению восстановления, ни к улучшению качества восстановления.

Кроме того, увеличение размерности линейного многообразия влечет за собой только увеличение длительности выполнения алгоритма, а также уменьшение количества итераций, требуемых для остановки алгоритма. Однако во всех рассмотренных случаях алгоритм останавливался из-за малого падения ошибки за итерацию, и во всех случаях значение конечной ошибки было не менее 5000.

Таблица 3. Временная эффективность алгоритма

i/n	2	3	4	5	6
2	2410496,11	492853,04	159738,79	65688,90	29506,48
3	803498,70	164284,35	53246,26	21896,30	9835,49
4	401749,35	82142,17	26623,13	10948,15	4917,75
5	241049,61	49285,30	15973,88	6568,89	2950,65
6	160699,74	32856,87	10649,25	4379,26	1967,10
7	114785,53	23469,19	7606,61	3128,04	1405,07
8	86089,15	17601,89	5704,96	2346,03	1053,80
9	66958,23	13690,36	4437,19	1824,69	819,62
10	53566,58	10952,29	3549,75	1459,75	655,70
i/n	7	8	9	10	100
2	15913,04	9193,67	4041,16	507,15	0,10
3	5304,35	3064,56	1347,05	169,05	0,03
4	2652,17	1532,28	673,53	84,52	0,02
5	1591,30	919,37	404,12	50,71	0,01
6	1060,87	612,91	269,41	33,81	0,01
7	757,76	437,79	192,44	24,15	0,00
8	568,32	328,35	144,33	18,11	0,00
9	442,03	255,38	112,25	14,09	0,00
10	353,62	204,30	89,80	11,27	0,00

Таблица 4. Время работы алгоритма, t MM:CC

n	2	3	4	5	6	7	8	9	10	100
t	11:53	11:54	11:54	11:54	11:55	11:55	11:56	11:59	12:05	15:23

Из всего вышесказанного можно сделать вывод, что восстановление изображений при помощи итерационного исчерпания матрицы с пропусками с использованием линейных многомерных многообразий следует проводить только в двумерном случае.

Литература

1. Горбань А.Н., Макаров С.В., Россиев А.А. Итерационный метод главных компонент для таблиц с пробелами // Третий сибирский конгресс по прикладной и индустриальной математике (ИНПРИМ-98), 22–27 июня 1998. Тезисы докладов. Ч. 5. Новосибирск: Изд-во Института математики СО РАН, 1998.
2. Ларионов И.Б. Кластеризация матриц с пропусками как метод восстановления графической информации // Математические структуры и моделирование. 2009. Вып. 20. С. 97–106.
3. Самарский А.А. Введение в численные методы М.: Наука, 1982.
4. Ismail Avcibas, Ismail Avcibas, Bulent Sankur, Lale Akarun, Emin Anarim, Nasir Memon, Yucel Yemez. Image Quality Statistics and Their Use in Steganalysis and Compression. 2001.