УДК 519.852

СКЕЛЕТНЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ И ЕГО ПРИМЕНЕНИЕ ДЛЯ РЕШЕНИЯ ОПТИМАЛЬНЫХ ЗАДАЧ ОЦЕНИВАНИЯ

Борис Цолакович БАХШИЯН родился в 1944 г. в городе Москве. Ведущий научный сотрудник Института космических исследований РАН. Доктор физико-математических наук, старший научный сотрудник. Основные научные интересы — в области теории управления и математического программирования. Автор 55 научных работ.

Boris Ts. BAKHSHIYAN, D.Sci., was born in 1944, in Moscow. He is a Principal Research Associate at the Space Research Institute (IKI) of the Russian Academy of Sciences. His research interests are in control theory and mathematical programming. He has published 55 technical papers.

Александр Владимирович ГОРЯИНОВ родился в 1985 г. в городе Москве. Студент МАИ. Основные научные интересы — в области теории управления и математического программирования.

Alexander V. GORYAINOV, was born in 1985, in Moscow. He is a Student at the MAI. His research interests are in control theory and mathematical programming.

На основе идей работы [1] разработан новый алгоритм решения обычных и многопараметрических (обобщенных) задач линейного программирования, гарантирующий отсутствие почти вырожденных итераций и не требующий операций обращения матриц. Алгоритм опробован на оптимальных задачах минимаксного оценивания для полиномиальной и тригонометрической моделей.

1. Введение. Идея алгоритма

Задачи оптимальной линейной идеальной коррекции и ряд задач оптимального планирования эксперимента сводятся к *многопараметрическим* (*обобщенным*) задачам линейного программирования, в которых векторы условий выбираются из выпуклых множеств [2—7]. Решение этих задач методом *генерации столбцов* из указанных множеств часто приводит к вычислительным проблемам. Это обусловлено как наличием *почти вырожденных итераций*, при которых мало меняется целевая функция, так и появлением почти одинаковых векторов в базисе. Кроме того, сходимость метода генерации столбцов не доказана.

Предлагается новый, так называемый *скелетный алгоритм* (название возникло из структуры алгоритма), который позволяет избежать указанных вычислительных проблем. Изложим идею нового алгоритма для обычной задачи линейного программирования

$$L^* = \min_{x_i}\left\{L=\sum_{i=1}^{n} c_i x_i :\ \sum_{i=1}^{n} x_i a_i = b,\ x_i \ge 0,\ i=1,...,n\right\}. \quad (1.1)$$

Здесь x_i — скалярные переменные оптимизации; остальные параметры являются заданными: c_i — коэффициенты целевой функции, $a_i \in \mathbb{R}^m$ — столбцы матрицы уравнений относительно x_i, $b \in \mathbb{R}^m$ — вектор правой части, причем $b \ne 0$. Величину L^* будем называть *значением* задачи (1.1).

Пусть I_B — множество из m индексов столбцов допустимого базиса $\{a_i,\ i \in I_B\}$, т.е. $\sum_{i\in I_B} x_i a_i = b,\ x_i \ge 0$ и векторы базиса линейно независимы. Вектор, состоящий из компонентов x_i, $i \in I_B$, называют *базисным* вектором, а вектор $x = (x_1,...,x_n)'$, содержащий все компоненты базисного вектора (и имеющий, очевидно, равными нулю остальные компоненты), называется допустимым базисным решением. Обозначим

$$a_0 = \sum_{i\in I_B} x_i a_i = b,\ \ c_0 = \sum_{i\in I_B} c_i x_i.$$

Сопоставим задаче (1.1) задачу линейного программирования с одним дополнительным столбцом a_0 и дополнительной переменной x_0:

$$\tilde{L}^* = \min_{x_i} \left\{ \tilde{L} = c_0 x_0 + \sum_{i=1}^{n} c_i x_i : \right.$$
$$\left. x_0 a_0 + \sum_{i=1}^{n} x_i a_i = b,\ x_i \ge 0,\ i = 0,1,...,n \right\}. \quad (1.2)$$

Теорема 1.

1. Допустимое решение $x_0 = 1,\ x_1 = 0,...,\ x_n = 0$ задачи (1.2) дает то же значение целевой функции, что и базисное решение задачи (1.1), соответствующее базису $\{a_i,\ i \in I_B\}$, т.е. $\tilde{L} = L$.

2. Значения задач (1.1) и (1.2) совпадают, т.е. $\tilde{L}^* = L^*$.

Доказательство. Первое утверждение легко проверить. Второе утверждение вытекает из следующих соображений. Если оптимальный базис задачи (1.2) не содержит столбец a_0, то этот базис является также оптимальным базисом задачи (1.1) (так как значение задачи (1.2) не превосходит значения задачи (1.1)). В противном случае оптимальным для задачи (1.1) является начальный базис $\{a_i,\ i \in I_B\}$.

Отметим, что теорема 1 также следует из лемм 1—3 работы [2]. В связи с теоремой 1 будем называть задачу (1.2) *эквивалентной* задаче (1.1).

Текущее решение эквивалентной задачи (1.2), соответствующее текущему решению основной задачи, является строго вырожденным со степенью вырожденности $m-1$ (только одна компонента $x_0 = 1$ положительна в текущем базисном решении).

Изложим теперь теорию решения задачи (1.2) применительно к рассматриваемому случаю вырожденности порядка $m-1$. По терминологии [8, 9] вектор a_0 является *строгим базисом. Базисной матрицей* будем называть составную матрицу $B = (a_0,\ V)$, где V — такая произвольная $m \times (m-1)$-матрица (не обязательно составленная из векторов a_i), что матрица B невырожденная. В соответствии с представлением матрицы B в виде составной матрицы, представим каждый вектор a_i в виде суммы:

$$a_i = u_i + v_i;\ u_i = g_i a_0,\ v_i = V\tilde{a}_i;\ g_i \in \mathbb{R}^1,\ \tilde{a}_i \in \mathbb{R}^{m-1}. \quad (1.3)$$

Здесь скаляр g_i и вектор $\tilde{a}_i$ включают коэффициенты разложения столбца a_i соответственно по столбцу a_0 и столбцам матрицы V, т.е.

$$\begin{pmatrix} g_i \\ \tilde{a}_i \end{pmatrix} = B^{-1} a_i,\ i = 0,1,...,n. \quad (1.4)$$

Рассмотрим *вспомогательную* задачу линейного программирования с $m-1$ независимыми ограничениями-равенствами:

$$\min_{\tilde{x}_i} \left\{ \sum_{i=1}^{n} \tilde{c}_i \tilde{x}_i :\ \sum_{i=1}^{n} \tilde{x}_i \tilde{a}_i = \tilde{b},\ \tilde{x}_i \ge 0\ \ \forall\ i \right\}. \quad (1.5)$$

Здесь $\tilde{c}_i \doteq \Delta_i = c_i - \pi' a_i$ — так называемая *относительная оценка*; π — любое решение уравнения $c_0 - \pi' a_0 = 0$; $\tilde{b}$ — любой вектор, для которого совместны ограничения во вспомогательной задаче (1.5).

Изложим две теоремы, которые доказаны в [8, 9].

Теорема 2. *Если задача (1.5) имеет решение, то текущее вырожденное базисное решение эквивалентной задачи (1.2) оптимально. В противном случае (целевая функция задачи (1.5) не ограничена) целевую функцию задачи (1.2) можно уменьшить либо установить неразрешимость задачи (1.2).*

Подробнее остановимся на второй части теоремы 2. Пусть $I_{\tilde{B}}$ состоит из индексов столбцов, входящих в текущий базис $\tilde{B}$ вспомогательной задачи (1.5). Если целевая функция задачи (1.5) не ограничена на множестве допустимых решений, то согласно теории линейного программирования [10] на некотором шаге симплекс-метода находится небазисный вектор $\tilde{a}_p$, такой, что коэффициенты α_j его разложения по базису, определяемые однозначно из уравнений

$$\tilde{a}_p = \sum_{j \in I_{\tilde{B}}} \alpha_j \tilde{a}_j, \quad (1.6)$$

удовлетворяют неравенствам

$$\tilde{\Delta}_p \doteq \tilde{c}_p - \sum_{j \in I_{\tilde{B}}} \tilde{c}_j \alpha_j < 0;\quad \alpha_j \le 0,\quad j \in I_{\tilde{B}}. \quad (1.7)$$

Здесь $\tilde{\Delta}_p$ — относительная оценка вектора $\tilde{a}_p$ во вспомогательной задаче. Соотношение (1.6) озна-

чает, согласно (1.3), что вектор $a_p - \sum_{j\in I_{\tilde{B}}} \alpha_j a_j$ пропорционален вектору a_0 [8, 9]. Это эквивалентно условиям:

$$a_p - \sum_{j\in I_{\tilde{B}}} \alpha_j a_j = \alpha a_0 \Rightarrow \alpha = g_p - \sum_{j\in I_{\tilde{B}}} \alpha_j g_j. \qquad (1.8)$$

Обозначим

$$S = \{j : j \in I_{\tilde{B}}, \alpha_{pj} < 0\} \cup \{p\}.$$

Теорема 3. *Если* $\alpha \le 0$*, то целевая функция задачи (1.2) не ограничена на множестве своих допустимых решений. В противном случае при замене строгого базиса* a_0 *на векторы* a_i, $i \in S$ *целевая функция задачи (1.2) уменьшается на величину* $\left|\tilde{\Delta}_p\right| / \alpha$.

Замечание 1. Указанное в теореме 2 уменьшение будет мало по сравнению с модулем текущего значения L целевой функции (итерация является почти вырожденной), только если мала величина $\left|\tilde{\Delta}_p\right| / |L|$. Однако в этом случае, согласно [2], отличие текущего значения целевой функции от ее минимума есть $C\left(\left|\tilde{\Delta}_p\right| / |L|\right)$, где величина C сравнима с единицей. Это означает, что при малой величине $\left|\tilde{\Delta}_p\right| / |L|$ (т.е. при появлении почти вырожденной итерации) мы находимся вблизи оптимума. Если же $\left|\tilde{\Delta}_p\right| / |L|$ не малая величина, то уменьшение целевой функции в задаче (1.2) также не мало. В этом состоит основное преимущество предлагаемого ниже метода по сравнению с симплекс-методом, не гарантирующим отсутствие почти вырожденных итераций.

Дальнейшая идея состоит в том, что решение задачи (1.2) с размерностью столбцов $m-1$ аналогично можно свести к решению задачи размерности $m-2$ (а потом перейти к задаче размерности $m-3$ и т.д.). При этом ниже, в разделе 2 мы воспользуемся правом произвольно выбирать правую часть ограничений в задаче размерности $m-2$ из условия их совместности. Естественно, мы выберем правую часть равной одному из столбцов матрицы ограничений и сразу получим вырожденное базисное решение со строгим базисом из одного вектора.

Итак, схема алгоритма следующая. На основе изложенного выписывается серия задач линейного программирования размерности от m до 1 (размерностью задачи будем называть число ее ограничений-равенств). Задача размерности m эквивалентна исходной задаче и имеет строгий базис из одного вектора (т.е. степень вырожденности максимальна и равна $m-1$). Для нее опять строится эквивалентная задача со строгим базисом, состоящим из одного вектора. Далее задача размерности $m-2$ является вспомогательной к задаче размерности $m-1$ и т.д. Во всех этих задачах только одна компонента положительна в текущем базисном решении. В результате мы получаем относительно простую одномерную задачу. Если одномерная задача имеет решение, то текущее решение двумерной задачи оптимально, а значит, оптимально и решение исходной m-мерной задачи. В противном случае либо уменьшается целевая функция двумерной задачи и вновь строится одномерная задача, либо устанавливается неразрешимость двумерной задачи и производится аналогичное рассмотрение трехмерной задачи и т.д. В результате мы либо установим оптимальность исходной задачи, либо уменьшим значение ее целевой функции.

Достоинства скелетного алгоритма состоят в отсутствии почти вырожденных итераций, операций обращения матриц. Кроме того, мы можем гарантировать сходимость алгоритма в следующем смысле. Если указанное в теореме 2 значение $\left|\tilde{\Delta}_p\right| / |L|$ не мало по сравнению с единицей, то, согласно этой теореме, значение целевой функции уменьшается на относительно немалую величину. Если значение $\left|\tilde{\Delta}_p\right| / |L|$ мало, то текущее значение целевой функции близко к оптимальному, как это следует из замечания 2. Таким образом, после некоторого числа шагов мы получаем почти оптимальное решение с заданной точностью.

2. Процедура скелетного алгоритма

Используя теоремы 1—3 из раздела 1, сформулируем теорию скелетного алгоритма. Перепишем исходную задачу (1.1) в виде

$$\min_{\substack{m \\ x_i}} \left\{ \sum_{i=1}^{n} \overset{m}{c}_i \overset{m}{x}_i : \sum_{i=1}^{n} \overset{m}{x}_i \overset{m}{a}_i = \overset{m}{b};\ \overset{m}{x}_i \ge 0,\ i = 1,\ldots,n \right\}. \qquad (2.1)$$

Здесь индекс m вверху означает *размерность задачи,* под которой мы понимаем размерность векторов столбцов этой задачи.

2.1. Построение серии вспомогательных задач

Выбираем начальный допустимый базис $\left\{\overset{m}{a}_i,\ i\in I_B,\ |I_B|=m\right\}$. Он удовлетворяет условию,

$$\sum_{i\in I_B} z_i \overset{m}{a}_i = \overset{m}{b},\ \ z_i \ge 0.$$

Обозначим

$$\overset{m}{c}_0 \doteq \sum_{i\in I_B} \overset{m}{c}_i z_i,\ \ \overset{m}{a}_0 \doteq \overset{m}{b}.$$

Для задачи (2.1) эквивалентная задача (1.2) имеет вид

$$\min_{x_i}\left\{\overset{m}{c}_0\overset{m}{x}_0+\sum_{i=1}^{n}\overset{m}{c}_i\overset{m}{x}_i : \overset{m}{x}_0\overset{m}{a}_0+\sum_{i=1}^{n}\overset{m}{x}_i\overset{m}{a}_i=\overset{m}{b};\ \overset{m}{x}_i\ge 0,\ i=0,1,...,n\right\}. \qquad (2.2)$$

Согласно теореме 1 допустимым решением этой задачи будет вырожденное решение

$$\left(\overset{m}{x}_0,\ \overset{m}{x}_1,...,\ \overset{m}{x}_n\right)' = (1,\ 0,...,\ 0)'.$$

Иначе можно сказать, что $\overset{m}{a}_0$ есть строгий базис, соответствующий вырожденному базисному решению со степенью вырожденности $m-1$. Используя теорию из раздела 1, построим вспомогательные задачи размерности $j=m-1,...,1$.

2.1.1. Найдём любой вектор $\overset{m}{\pi}$ из условия $\overset{m}{\Delta}_0 \doteq \overset{m}{c}_0 - \overset{m}{\pi}'\overset{m}{a}_0 = 0$. Выберем решение с минимальной нормой: $\overset{m}{\pi} = c\overset{m}{a}_0$, где $c = \overset{m}{c}_0 / \left|\overset{m}{a}_0\right|^2$. Вычислим относительные оценки (обозначаемые в (1.5) через $\tilde{c}_i$)

$$\overset{m-1}{c}_i \doteq \overset{m}{\Delta}_i = \overset{m}{c}_i - \overset{m}{\pi}'\overset{m}{a}_i,\ \ i=1,...,n. \qquad (2.3)$$

Если $\overset{m}{\Delta}_i \ge 0\ \forall\ i\ge 1$, то решение

$$\left(\overset{m}{x}_0,\overset{m}{x}_1,...,\overset{m}{x}_n\right)' = (1,0,...,0)'$$

оптимально, и базисный вектор $(z_i,...,z_m)'$ дает решение исходной задачи. В противном случае (если достаточное условие оптимальности не выполнено и найдется хотя бы одна $\overset{m}{\Delta}_i < 0$) переходим к построению следующей задачи.

2.1.2. Пусть $a_0 = (a_{01},...,a_{0n})'$. Без ограничения общности будем считать, что $\overset{m}{a}_{01}$ — элемент, модуль которого не мал. Возьмем базисную матрицу

$$\overset{m}{B} = \begin{pmatrix} \overset{m}{a}_{01} & 0 \\ \overset{m}{a}_{02} & \\ \vdots & I_{m-1} \\ \overset{m}{a}_{0m} & \end{pmatrix}$$

(I_{m-1} — единичная матрица порядка $m-1$).

Пусть

$$\overset{m}{s}'_i = \begin{pmatrix} \overset{m}{g}_i,\ \overset{m-1}{a}'_i \end{pmatrix}$$

— вектор координат столбца $\overset{m}{a}_i$ в базисе $\overset{m}{B}$. Здесь $\overset{m-1}{a}_i \in \mathbb{R}^{m-1}$, $\overset{m}{g}_i \in \mathbb{R}^1$.

Таким образом, $\overset{m}{B}\overset{m}{s}_i = \overset{m}{a}_i$. Решая это уравнение относительно $\overset{m-1}{a}_i$, получаем:

$$\overset{m}{g}_i = \frac{\overset{m}{a}_{i1}}{\overset{m}{a}_{01}},\qquad \overset{m-1}{a}_i = \begin{pmatrix} \overset{m}{a}_{i2} \\ \vdots \\ \overset{m}{a}_{im} \end{pmatrix} - \begin{pmatrix} \overset{m}{a}_{02} \\ \vdots \\ \overset{m}{a}_{0m} \end{pmatrix}\overset{m}{g}_i.$$

Это есть преобразования (1.4) в новых обозначениях $\overset{m}{g}_i = g_i,\ \ \overset{m-1}{a}_i = \tilde{a}_i$.

Рассмотрим вспомогательную задачу вида (1.4)

$$\min_{x_i}\left\{\sum_{i=1}^{n}\overset{m-1}{c}_i\ \overset{m-1}{x}_i : \sum_{i=1}^{n}\overset{m-1}{x}_i\ \overset{m-1}{a}_i = \overset{m-1}{b},\ \overset{m-1}{x}_i \ge 0,\ i=1,...,n\right\}. \qquad (2.4)$$

Коэффициенты $\overset{m-1}{c}_i$ определены в (2.3). Примем что $\overset{m-1}{b} = \overset{m-1}{a}_{f(m-1)}$, где $f(m-1)$ — произвольный

индекс. Тогда вектор $\overset{m-1}{a}_0 \doteq \overset{m-1}{a}_{f(m-1)}$ является строгим базисом вырожденного решения

$$\overset{m-1}{x}_0 \doteq \overset{m-1}{x}_{f(m-1)} = 1, \quad \overset{m-1}{x}_i = 0 \ \forall\ i \geq 1.$$

Так как $\overset{m-1}{a}_0$ есть строгий базис, то аналогично мы строим вспомогательную задачу размерности $m-2$, потом вспомогательные задачи меньшей размерности вплоть до задачи размерности 1.

Замечание 2. При построении вспомогательных задач мы выбирали вектор правых частей условий задачи равным одному из векторов условий $\overset{j}{a}_{f(j)},\ j = m-1,...,1$. Это позволило сразу найти вырожденное решение со строгим базисом, состоящим из одного вектора $\overset{j}{a}_0 \doteq \overset{j}{a}_{f(j)},\ j = m-1,...,1$. Таким образом, отпадает необходимость в конструировании эквивалентной задачи для каждой вспомогательной задачи размерности меньшей *m*. В то же время для удобства мы вводим обозначения $\overset{j}{a}_0 \doteq \overset{j}{a}_{f(j)}$ и $\overset{j}{x}_0 = \overset{j}{x}_{f(j)}$ для строгого базиса и соответствующей переменной и переписываем задачу (2.4) в эквивалентном виде:

$$\min_{x_i} \left\{ \overset{m-1}{c}_0\ \overset{m-1}{x}_0 + \sum_{i=1}^{n} \overset{m-1}{c}_i\ \overset{m-1}{x}_i : \right.$$
$$\left. \overset{m-1}{x}_0\ \overset{m-1}{a}_0 + \sum_{i=1}^{n} \overset{m-1}{x}_i\ \overset{m-1}{a}_i = \overset{m-1}{b},\quad \overset{m-1}{x}_i \geq 0,\ i = 1,...,n \right\}. \quad (2.5)$$

Замечание 3. При построении вспомогательных задач может выявиться оптимальность решения вспомогательной задачи размерности *j*. Это означает, согласно теореме 2, что текущее решение исходной задачи размерности *m* оптимально.

Столбец $\overset{j-1}{a}_i$ будем называть *образом* столбца $\overset{j}{a}_i$. Последний будем называть *прообразом* столбца $\overset{j-1}{a}_i$ и использовать обозначение $\overset{j}{a}_i = pt\left[\overset{j-1}{a}_i\right]$ (от анг. «prototype»). Прообраз линейной комбинации столбцов вида $\overset{j}{a}_i$ будет равен линейной комбинации прообразов этих столбцов.

Столбцы и правые части вспомогательных задач строятся на первом этапе алгоритма. Перейдем к итерациям алгоритма, на которых будут изменяться только коэффициенты целевых функций вспомогательных задач.

2.2. Описание итераций

Замечание 4. При выполнении итераций всегда $\overset{j}{b} = \overset{j}{a}_{f(j)}$, кроме случая $j = 1$. Нам представляется логичным выбирать индекс

$$f(j) = \arg\min_{i=1,...,n} \overset{j+1}{\Delta}_i, \quad j = m-1,...,2.$$

Случай $j = 1$ описан ниже. Так как столбец $\overset{j}{a}_0$ равен вектору $\overset{j}{b}$ (см. раздел 1), то столбцы вспомогательных задач не пересчитываются при выполнении итераций. Это следует из формул (1.3) при $a_i = \overset{j}{a}_i,\ \tilde{a}_i = \overset{j-1}{a}_i$.

2.2.1. Решение одномерной задачи. Покажем, что обычная одномерная задача линейного программирования

$$\min\left\{ \overset{1}{c}_0\ \overset{1}{x}_0 + \sum_{i=1}^{n} \overset{1}{c}_i\ \overset{1}{x}_i :\ \sum_{i=1}^{n} \overset{1}{x}_i\ \overset{1}{a}_i = \overset{1}{b};\ \ \overset{1}{x}_i \geq 0,\ \ i=0,1,..,n \right\}$$

может быть решена аналитически. Вычисляем

$$\overset{1}{\pi} = \frac{\overset{1}{c}_0}{\left|\overset{1}{a}_0\right|^2}\overset{1}{a}_0 = \frac{\overset{1}{c}_0}{\overset{1}{a}_0},\quad \overset{1}{\Delta}_i = \overset{1}{c}_i - \overset{1}{\pi}\ \overset{1}{a}_i = \overset{1}{c}_i - \frac{\overset{1}{c}_0}{\left|\overset{1}{a}_0\right|^2}\overset{1}{a}_i.$$

Возьмем текущий базис $\overset{1}{a}_0 \doteq \overset{1}{a}_{f(1)} \neq 0$ из условия $f(1) = \arg\min_i \left\{ \overset{1}{c}_i / \overset{1}{a}_i :\ \overset{1}{a}_i > 0 \right\}$ и примем, подобно тому как это делалось при рассмотрении *m*-мерной задачи, $\overset{1}{b} \doteq \overset{1}{a}_{f(1)}$. Тогда текущее значение целевой функции равно $\overset{1}{c}_{f(1)}$. При определенном выше выборе индекса $f(1)$ имеются только две возможности.

1. Выполняется условие

$$\min\left\{ \overset{1}{\Delta}_i : i = 0,1,...,n \right\} = 0,$$

на практике проверяется условие

$$\min\left\{\overset{1}{\Delta}_i : i = 0,1,...,n\right\} > -\varepsilon$$

где ε — малое положительное число. Тогда базис $\overset{1}{a}_0$ в одномерной задаче оптимален. Согласно теореме 2 в этом случае будет оптимален текущий базис во всех задачах большей размерности. Поэтому текущий базис $\overset{m}{a}_1, \overset{m}{a}_2, ..., \overset{m}{a}_n$ в эквивалентной задаче также оптимален и исходная задача решена.

2. Находится индекс p, такой, что $\overset{1}{\Delta}_p < 0$ и верно соотношение (1.5):

$$\overset{1}{a}_p = \alpha \overset{1}{a}_0, \quad \alpha \le 0.$$

Согласно теореме 3, это означает, что у одномерной задачи нет решения. В этом случае мы переходим к п. 2.2.2.

2.2.2. Переход к двумерной задаче при $\overset{1}{\alpha} \le 0$ («подъём»). Перепишем соотношение между $\overset{1}{a}_p$ и $\overset{1}{a}_0$ в виде (1.6)

$$\overset{1}{a}_p - \overset{1}{\alpha}\overset{1}{a}_0 = 0 \;\Leftrightarrow\; \overset{2}{a}_p - \overset{1}{\alpha}\, pt\left[\overset{1}{a}_0\right] = \overset{2}{\alpha}\overset{2}{a}_0.$$

Здесь $pt\left[\overset{1}{a}_0\right]$ — прообраз вектора $\overset{1}{a}_0$ в пространстве размерности 2. Всегда $\overset{1}{a}_0 = \overset{1}{a}_{f(1)}$, поэтому $pt\left[\overset{1}{a}_0\right] = \overset{2}{a}_{f(1)}$ и окончательно получаем

$$\overset{2}{a}_p - \overset{1}{\alpha}\overset{2}{a}_{f(1)} = \overset{2}{\alpha}\overset{2}{a}_0.$$

В зависимости от знака $\overset{2}{a}_0$, согласно теореме 3 имеем два случая (п. 2.2.3 и 2.2.4 соответственно).

2.2.3. Решение двумерной задачи («спуск от двумерной задачи»). Если $\overset{2}{\alpha} > 0$ (на практике $\overset{2}{\alpha} > \varepsilon$, где ε — малое положительное число, см. замечание 1), то согласно теореме 2 целевую функцию будем уменьшать до тех пор, пока либо не получим оптимального решения двумерной задачи (в этом случае доказывается оптимум текущего решения m-мерной задачи), либо не установим неразрешимость этой задачи (в этом случае переходим к рассмотрению соотношений для трехмерной задачи — переход к п. 2.2.4). Для этого опять рассмотрим вспомогательную задачу размерности 1. Согласно теореме 3 вводим в базис вместо вектора $\overset{2}{a}_0$ векторы $\overset{2}{a}_0$ и $pt\left[\overset{1}{a}_0\right]$. Далее, используя теорему 1, переходим к эквивалентной задаче, вводя новый вектор $\overset{2}{a}_{0new}$ как линейную комбинацию двух векторов и соответствующий коэффициент $\overset{2}{c}_{0new}$ в целевой функции:

$$\overset{2}{c}_{0new} = \frac{1}{\overset{2}{\alpha}}\left(\overset{2}{c}_p - \overset{1}{\alpha}\overset{2}{c}_{f(1)}\right), \quad \overset{2}{a}_{0new} = \frac{1}{\overset{2}{\alpha}}\left(\overset{2}{a}_p - \overset{1}{\alpha}\, pt\left[\overset{1}{a}_0\right]\right).$$

При этом численно $\overset{2}{a}_{0new} = \overset{2}{a}_0$. Целевая функция в задаче размерности 2 уменьшается и становится равной $\overset{2}{c}_{0new}$. После этого мы переходим к *спуску*, под которым мы понимаем построение новой одномерной задачи (см. разд. 2.2.6).

2.2.4. Переход к трехмерной задаче («подъем» от двумерной задачи). Если $\overset{2}{\alpha} \le 0$, то согласно теореме 3 двумерная задача не имеет решения. Подъем к трехмерной задаче осуществляется аналогично подъему к двумерной задаче. Имеем в этом случае соотношение вида (1.6):

$$\overset{2}{a}_p - \overset{1}{\alpha}\, pt\left[\overset{1}{a}_0\right] - \overset{2}{\alpha}\overset{2}{a}_0 = 0 \;\Leftrightarrow$$

$$\Leftrightarrow\; \overset{3}{a}_p - \overset{1}{\alpha}\, pt^2\left[\overset{1}{a}_0\right] - \overset{2}{\alpha}\, pt\left[\overset{2}{a}_0\right] = \overset{3}{\alpha}\overset{3}{a}_0,$$

где $pt^2\left[\overset{1}{a}_0\right] = pt\left[pt\left[\overset{1}{a}_0\right]\right] = \overset{3}{a}_{f(1)}$ — прообраз вектора $\overset{1}{a}_0$ в пространстве $\mathbb{R}^3$.

2.2.5. Спуск от трехмерной задачи. Если $\overset{3}{\alpha} > 0$, то вычисляем

$$\overset{3}{c}_{0new} = \frac{1}{\overset{3}{\alpha}}\left(\overset{3}{c}_p - \overset{1}{\alpha}\overset{3}{c}_{f(1)} - \overset{2}{\alpha}\overset{3}{c}_{f(2)}\right)$$

и рассматриваем двумерную вспомогательную задачу с новыми коэффициентами целевой функции.

2.2.6. Процедура подъема и спуска в общем случае. При переходе от $(j-1)$-мерной задачи к j-мерной задаче мы получаем уравнение вида (1.8):

$$\overset{j}{a}_p - \overset{1}{\alpha}\, pt^{j-1}\left[\overset{1}{a}_0\right] - \ldots - \overset{j-1}{\alpha}\, pt\left[\overset{j-1}{a}_0\right] = \overset{j}{\alpha}\overset{j}{a}_0. \qquad (2.6)$$

Возможны два случая.

1. Величина $\overset{j}{\alpha} \le 0$. Тогда j-мерная задача не имеет решения и мы переходим к аналогичному соотношению для $(j+1)$-мерной задачи:

$$\overset{j}{a}_p - \overset{1}{\alpha}\, pt^{j-1}\left[\overset{1}{a}_0\right] - \ldots - \overset{j}{\alpha}\, pt\left[\overset{j}{a}_0\right] = \overset{j+1}{\alpha}\overset{j+1}{a}_0.$$

Тем самым осуществляется подъем к $(j+1)$-мерной задаче.

2. Величина $\overset{j}{\alpha} > 0$. Тогда вычисляем обновленное значение целевой функции, равное коэффициенту при переменной $\overset{j}{x}_0$:

$$\overset{j}{c}_{0new} = \frac{1}{\overset{j}{\alpha}}\left(\overset{j}{c}_p - \overset{1}{\alpha}\overset{j}{c}_{f(1)} - \ldots - \overset{j-1}{\alpha}\overset{j}{c}_{f(j-1)}\right) \qquad (2.7)$$

и запоминаем выражение для $\overset{j}{a}_{0new}$ (численно $\overset{j}{a}_{0new} = \overset{j}{a}_0$):

$$\overset{j}{a}_{0new} = \frac{1}{\overset{j}{\alpha}}\left(\overset{j}{a}_p - \overset{1}{\alpha}\overset{j}{a}_{f(1)} - \ldots - \overset{j-1}{\alpha}\overset{j}{a}_{f(j-1)}\right). \qquad (2.8)$$

Далее рассматриваем вспомогательную задачу размерности j:

$$\min_{x_i}\left\{\overset{j}{c}_{0new}\overset{j}{x}_0 + \sum_{i=1}^{n}\overset{j}{c}_i\overset{j}{x}_i : \right.$$
$$\left. \overset{j}{x}_0\overset{j}{a}_{0new} + \sum_{i=1}^{n}\overset{j}{x}_i\overset{j}{a}_i = \overset{j}{b};\ \ \overset{j}{x}_i \ge 0,\ \ i = 0,1,\ldots,n\right\}.$$

Для нее строгий базис есть $\overset{j}{a}_{0new}$. Коэффициенты в целевых функциях вспомогательных задач размерностей $(j-1),\ldots,1$ пересчитываются аналогично (2.3):

$$\overset{j}{\pi} = \frac{\overset{j}{c}_0}{\left|\overset{j}{a}_0\right|^2}\overset{j}{a}_0,\ \ \overset{j-1}{c}_i = \overset{j}{\Delta}_i = \overset{j}{c}_i - \overset{j}{\pi}'\overset{j}{a}_i,\ \ i = 1,\ldots,n. \qquad (2.9)$$

Таким образом осуществляется «спуск» к одномерной задаче. Далее решаем одномерную задачу согласно п. 2.2.1.

Замечание 5. Рассмотрим подробнее вид прообраза. Если при переходе от размерности $j-1$ к размерности j нужно вычислять прообраз вектора, то он будет иметь различный вид в зависимости от того, на каком шаге закончился подъём на предыдущей итерации. Возможны два случая.

1. Предыдущий подъём закончился на задаче размерности больше j. Тогда

$$pt\left[\overset{j-1}{a}_0\right] = \overset{j}{a}_{f(j-1)}.$$

2. Предыдущий подъём закончился на задаче размерности j или меньше. Тогда

$$pt\left[\overset{j-1}{a}_0\right] = \frac{1}{\overset{j}{\tilde{\alpha}}}\left(\overset{j}{a}_{\tilde{p}} - \sum_{i=1}^{j-1}\overset{i}{\tilde{\alpha}}\overset{j}{a}_{\tilde{f}(i)}\right),$$

где $\overset{j}{\tilde{\alpha}}$, $\tilde{p}$ и $\tilde{f}(i)$ — коэффициенты и индексы, полученные на предыдущей итерации.

Замечание 6. Если на некотором подъеме уменьшение целевой функции m-мерной задачи мало , то, согласно замечанию 1, мы получили почти оптимальное решение исходной задачи.

Замечание 7. Следует отметить, что в результате решения задачи линейного программирования при помощи скелетного алгоритма оптимальное решение может быть не базисным, т.е. количество ненулевых переменных в решении может превышать размерность задачи. Это связано с тем, что на некоторых итерациях при «подъёме» к задаче большей размерности прообразом одного вектора может быть линейная комбинация векторов.

3. Скелетный алгоритм

Опишем шаги алгоритма в соответствии с изложенной пролцедурой.

Шаг 1. Построение вспомогательных задач.

Конструируем серию вспомогательных задач в соответствии с разд. 2.1. Если на каком-либо этапе оказывается, что все коэффициенты целевой функции любой вспомогательной задачи неотрицательны, то решение вспомогательной задачи является оптимально и мы переходим к шагу 4. Если ни в одной из вспомогательных задач решение не является оптимальным, то после построения одномерной задачи переходим к шагу 2.

Шаг 2. Подъём.

Решаем одномерную задачу. Если она разрешима, то переходим к шагу 4. Если эта задача нераз-

решима (величина $\overset{1}{\alpha} \le 0$), то из соотношения (2.6) последовательно вычисляем $\overset{j}{\alpha}$ для $j = 2,3,...$ до того минимального индекса j, при котором $\overset{j}{\alpha} > 0$ (иначе говоря, значения $\overset{j}{\alpha}$ вычисляются в цикле, условием выхода из которого является $\overset{j}{\alpha} > 0$). После этого переходим к шагу 3.

Шаг 3. Спуск.

В результате выполнения шага 2 найден индекс j, такой, что $\overset{i}{\alpha} \le 0,\ i = 1,...,j-1$, но $\overset{j}{\alpha} > 0$.

Вычисляем $\overset{j}{c}_{0new}$ по формуле (2.7) и запоминаем выражение (2.8) для нового вспомогательного вектора $\overset{j}{a}_{0new}$ (согласно замечанию 7, при программной реализации алгоритма значения коэффициентов в выражении (2.8) должны храниться в памяти вплоть до окончания следующего «подъема»).

После этого пересчитываем коэффициенты $\overset{k}{c}_i$ для $k = j, j-1,...,1$ по формуле (2.9). Далее переходим к шагу 2.

Шаг 4. Окончание решения задачи. Решение задачи дает текущий строгий базис задачи размерности m.

4. Минимаксная задача оценивания

Рассмотрим задачу оценивания, при условии, что модель измерений имеет вид

$$y_i = H_i\theta + \xi_i,\ \ i = 1,...,n, \qquad (4.1)$$

где θ — m-вектор неизвестных параметров; y_i, ξ_i — результаты измерений и их ошибки; H_i — известные векторы.

Пусть $y = (y_1,...,y_n)'$ — соответствующий вектор измерений, $l = b'\theta$ — контролируемый параметр (b — известный вектор), $\hat{l} = \sum_{i=1}^{n} x_i y_i$ — линейная оценка параметра l, где коэффициенты x_i удовлетворяют так называемому условию несмещенности алгоритма

$$\sum_{i=1}^{n} x_i H_i = b.$$

Это условие означает, что при нулевых ошибках измерений оценка является истинной: $\hat{l} = b'\theta$, если $\xi_i = 0,\ \ i = 1,...,n$.

Предполагая, что $|\xi_i| \le 1$, поставим минимаксную задачу:

$$L^* = \min_{x_i} \max_{|\xi_i| \le 1} \left\{ |\hat{l} - l| :\ \ \sum_{i=1}^{n} x_i H_i = b \right\}.$$

Она сводится к задаче минимизации [4]:

$$L^* = \min_{x_i} \left\{ L = \sum_{i=1}^{n} |x_i| :\ \ \sum_{i=1}^{n} x_i H_i = b \right\}. \qquad (4.2)$$

Последняя задача представляет собой задачу линейного программирования. В явном виде она становится таковой, если сделать следующую замену переменных [4]:

$$x_i = x_{1i} - x_{2i},\ \ |x_i| = x_{1i} + x_{2i},\ \ x_{1i}x_{2i} = 0,\ \ x_{1i},\ x_{2i} \ge 0.$$

При этом она примет вид

$$L^* = \min_{x_{i1}, x_{i2}} \left\{ \sum_{i=1}^{n} (x_{1i} + x_{2i}) : \right.$$
$$\left. \sum_{i=1}^{n} (x_{1i} - x_{2i}) H_i = b;\ x_{1i}, x_{2i} \ge 0,\ i = 1,...,n \right\}, \qquad (4.3)$$

так как оптимальное решение удовлетворяет условию $x_{1i}x_{2i} = 0$.

5. Численные эксперименты

5.1. Решение минимаксной задачи для тригонометрической модели

Рассмотрим тригонометрическую модель измерений третьего порядка вида (4.1):

$$y(t) = \theta_1 + \theta_2 \cdot \sin t + \theta_3 \cdot \cos t + \xi(t),\ \ t \in [0,1].$$

Здесь t — время.

Замечание 8. Будем считать, что измерения могут производиться в любой момент времени на $[0,1]$, и называть в этом случае модель измерений *непрерывной*. Это формально соответствует бесконечному числу столбцов в задаче (4.3), но не приводит к трудностям решения получившейся задачи симплекс-методом. В этом случае в базис на каждой итерации вводится столбец $\pm(1,\ \sin t,\ \cos t)'$, для которого относительная оценка (определение ее дано в начале разд. 1) минимальна на $[0,1]$ и отрицатель-

на. Формально мы решаем при этом обобщенную задачу линейного программирования методом генерации столбцов, упомянутым в разд. 1. Однако это не приводит к дополнительным вычислениям, так как модификация симплекс-метода состоит в том, что минимальная относительная оценка находится не перебором, а численным методом или аналитически. Более того, для непрерывной модели измерений в памяти хранится только аналитическое представление столбцов $\pm(1,\ \sin t,\ \cos t)'$.

В качестве контролируемого параметра l возьмем θ_2. Начальный базис образуем из векторов, соответствующих моментам времени {0,88; 0,888; 0,8888}. Этот случай неблагоприятен для симплекс-метода, так как базисная матрица плохо обусловлена.

Результаты решения задачи (4.2) представлены в табл. 1. Под итерацией скелетного алгоритма будем понимать последовательность «подъёмов» и «спусков», приводящих к улучшению значения целевой функции исходной задачи.

Схема спусков и подъемов для представлена на рис. 1.

Спуск от трёхмерной задачи к двумерной выполнялся один раз, от двумерной к одномерной — шесть раз, подъём от одномерной задачи к двумерной — пять раз, от двумерной к трёхмерной — один раз, пересчёт c_0^2 — пять раз, пересчёт c_0^3 — один раз. Таким образом, согласно приведённым в приложении формулам, количество элементарных операций составило 995.

Таблица 1

№ итерации	Симплекс-метод		Скелетный алгоритм	
	Целевая функция	Базис	Целевая функция	Базис
1	483039,19	{0,88; 0,888; 0,8888}	483039,19	{0,88; 0,888; 0,8888}
2	2502,10	{0; 0,888; 0,8888}	**7,83**	{0; 05; 1}
3	8,85	{0; 0,444; 0,8888}		
4	7,93	{0; 0,444; 1}		
5	**7,83**	{0; 05; 1}		

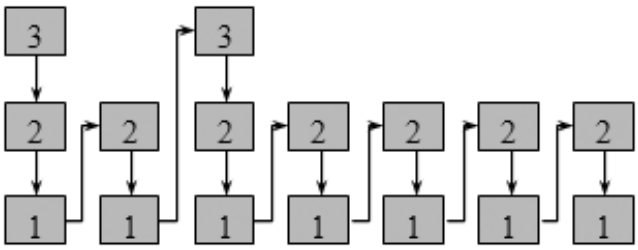

Рис. 1. Схема спусков и подъёмов для начального базиса, соответствующего моментам времени {0,88; 0,888; 0,8888}

При решении задачи симплекс-методом, согласно приложению, число операций составляет 1245.

В табл. 2 представлены результаты решения задачи (4.2) для начального базиса, соответствующего моментам времени {0,23; 0,33; 0,43}. Эта ситуация более благоприятна для симплекс-метода.

Схема спусков и подъемов для представлена на рис. 2.

Спуск от трёхмерной задачи к двумерной выполнялся один раз, от двумерной к одномерной — пять раз, подъём от одномерной задачи к двумерной — пять раз, от двумерной к трёхмерной — один раз, пересчёт c_0^2 — четыре раза, пересчёт c_0^3 — один раз. Количество элементарных операций равно 902.

Таблица 2

№ итерации	Симплекс-метод		Скелетный алгоритм	
	Целевая функция	Базис	Целевая функция	Базис
1	129,73	{0,23; 0,33; 0,43}	129,73	{0,23; 0,33; 0,43}
2	35,11	{0,23; 0,33; 1}	**7,83**	{1; 0,5; 0}
3	8,88	{0; 0,33; 1}		
4	**7,83**	{0; 0,5; 1}		

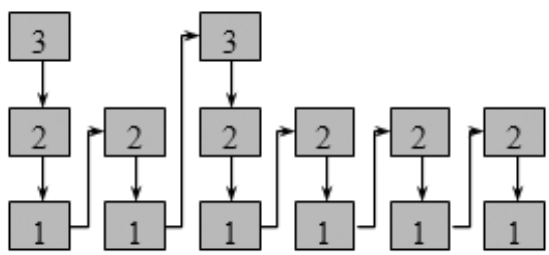

Рис. 2. Схема спусков и подъёмов для начального базиса, соответствующего моментам времени {0,23; 0,33; 0,43}

При решении задачи симплекс-методом число элементарных операций составляет для четырёх итераций 996.

5.2. Решение минимаксной задачи для полиномиальной модели

Рассмотрим полиномиальную модель четвёртого порядка:

$$y(t) = \theta_1 + \theta_2 \cdot t + \theta_3 \cdot t^2 + \theta_4 \cdot t^3 + \xi(t),\ \ t \in [0,1].$$

Вычисления будем производить в соответствии с замечанием 8. В качестве контролируемого параметра l возьмем θ_2. Начальный базис образуем из векторов, соответствующих моментам времени {0,01; 0,02; 0,03; 0,04}. Результаты решения задачи (4.2) даны в табл. 3. Схема спусков и подъемов представлена на рис. 3.

Таблица 3

№ итерации	Симплекс-метод		Скелетный алгоритм	
	Целевая функция	Базис	Целевая функция	Базис
1	2251,667	{0,01; 0,02; 0,03; 0,04}	2251,667	{0,01; 0,02; 0,03; 0,04}
2	815,329	{0,01; 0,02; 0,03; 1}	18,232	{0; 0,33; 0,74; 1}
3	212,580	{0,01; 0,02; 0,67; 1}	17,008	{0; 0,26; 0,75; 1}
4	18,444	{0,01; 0,27; 0,67; 1}	**17,000**	{0; 0,25; 0,75; 1}
5	17,447	{0; 0,27; 0,67; 1}		
6	17,062	{0; 0,27; 0,74; 1}		
7	17,001	{0; 0,25; 0,74; 1}		
8	**17,000**	{0; 0,25; 0,75; 1}		

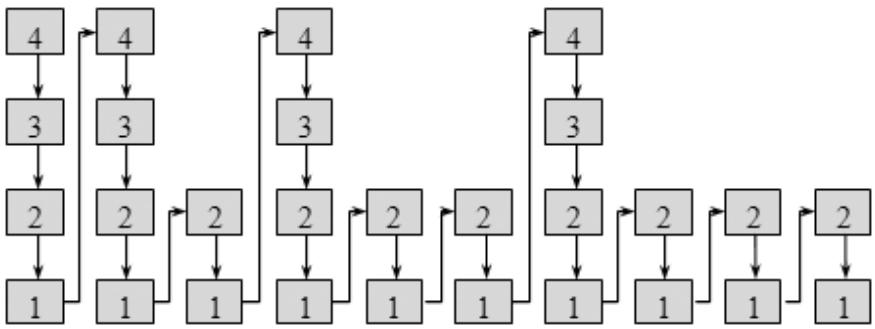

Рис. 3. Схема спусков и подъёмов для полиномиальной модели с начальными базисом, соответствующим моментам {0,01; 0,02; 0,03; 0,04}

Посчитанная аналогично пункту 5.1 вычислительная сложность решения задачи обоими способами составляет для скелетного алгоритма 2643, а для симлекс-метода 3000 операций.

Замечание 9. Численные экспериметны показали интересную особенность скелетного алгоритма: на первой итерации происходит очень существенное (часто на порядок и более) улучшение значения целевой функции, а на последующих итерациях её значение приближается к оптимальному довольно медленно. Тем не менее вычислительные затраты на этих итерациях весьма существенны. В свете этих наблюдений представляется перспективным вариант комбинированного использования скелетного алгоритма и симплекс-метода: после одной итерации скелетного алгоритма, существенно улучшающей значение целевой функции, решение продолжается симплекс-методом.

6. Заключение

Данная работа посвящена разработке и тестированию нового скелетного алгоритма на примерах решения минимаксной задачи оценивания различных порядков. При возрастании вычислительной сложности решаемых задач скелетный алгоритм позволяет избежать, в отличие от стандартного симплекс-метода, наличия почти вырожденных итераций и накопления больших вычислительных ошибок. Алгоритм достаточно прост и не требует операций обращения матриц.

Скелетный алгоритм оказывается особенно эффективным для обобщенных задач линейного программирования, ввиду того что обычно можно воспользоваться аналитическим представлением столбцов и не хранить в памяти все массивы столбцов вспомогательных задач размерностей от $m-1$ до 1. Например, это имеет место для рассмотренной нами минимаксной задачи оценивания (4.1), для которой возможные моменты измерений заполняют отрезок.

Для более полных выводов об области эффективного применения скелетного алгоритма необходимы дополнительные массовые расчеты.

Приложение.

Вычислительная сложность алгоритма

Оценим вычислительную сложность скелетного алгоритма и стандартного симплекс-метода. Для этого подсчитаем количество элементарных арифметических операций, выполняемых при решении задачи. Будем считать, что операция умножения занимает в два раза больше машинного времени, чем операция сложения (для чисел с плавающей точкой это допущение вполне приемлемо). Операциями сравнения и присваивания будем пренебрегать. Приняв, в силу сделанного выше предположения, умножение за два сложения, подсчитаем сложность одной итерации симплекс-метода. На каждой итерации выполняются следующие действия.

Пересчёт элементов d_{ij} матрицы B^{-1} по формулам [10]

$$\tilde{d}_{ij} = \begin{cases} d_{ij} - d_{rj}\dfrac{\alpha_i}{\alpha_r}, & i \neq r \\ d_{rj}\dfrac{1}{\alpha_r}, & i = r. \end{cases} :$$

$5m(m-1)+2m = 5m^2+3m$ операций.

Пересчёт значения целевой функции:

$3(m+1)$ операций.

Вычисление вектора $\pi' = c'_B B^{-1}$ (см. (2.3)):

$(2m+m-1)m = 3m^2 - m$ операций.

Вычисление вектора α координат столбца $\alpha = B^{-1}A_s$, вводимого в базис:

$(2m+m-1)m = 3m^2 - m$ операций.

Вычисление относительных оценок $\Delta_i = c_i - \pi' a_i$:

$(1+2m+m-1)n = 3mn$ операций.

Определение вектора, выводимого из базиса:

$2m$ операций.

Таким образом, совокупное количество операций на одной итерации симплекс-метода составляет

$$11m^2 + 7m + 3 + 3nm.$$

Для скелетного алгоритма сложность этапов следующая.

Построение вспомогательных задач:

$$\sum_{j=2}^{m} (2j + 2n + 6jn + 3) \text{ операций.}$$

Спуск от задачи размерности j к задаче размерности $j-1$:

$2j + 3jn + 3$ операций.

Подъём от задачи размерности $j-1$ к задаче размерности j:

$3j - 4$ операции.

Пересчёт величины $\overset{j}{c}_0$:

$3j - 4$ операции.

Сложность процедуры нахождения начального базиса в сравниваемых методах учитывать не будем.

Множитель n в приведённых выражениях (кроме члена $2n$) соответствует вычислению n относительных оценок Δ_i. В наших вычислительных экспериментах, согласно разд. 4, Δ зависит от непрерывно изменяющегося параметра t и минимизация по t проводится численно. Сходимость стандартных методов численной безусловной оптимизации описывается формулой $N \sim \log_2 \frac{1}{\varepsilon}$, где N — число итераций, ε — требуемая точность. Если принять желаемую точность $\varepsilon = 10^{-5}$, то необходимое число итераций будет равно 14.

Член $2n$ в выражении, описывающем сложность построения вспомогательных задач, соответствует нахождению n чисел $\overset{j}{g}_i$ (см. раздел 2.1.2). В случае непрерывно изменяющегося параметра t формула для $\overset{j}{g}(t)$ вычисляется один раз и n в этом слагаемом можно положить равным единице.

Выводы

В статье разработан новый алгоритм решения обычных и многопараметрических (обобщенных) задач линейного программирования, гарантирующий отсутствие почти вырожденных итераций и не требующий операций обращения матриц. Алгоритм опробован на оптимальных задачах минимаксного оценивания для полиномиальной и тригонометрической моделей. Приведённые примеры показывают эффективность алгоритма в сравнении со стандартным симплекс-методом.

Summary

A new algorithm is proposed to solve ordinary and multiparameter (generalized) linear programming problems. The algorithm guarantees absence of almost degenerate iterations and does not require inversion of matrices. The algorithm was tested to solve optimal minimax estimation problems for polynomial and trigonometric models. Some examples are presented to demonstrate an effectiveness of the algorithm in comparison with the standard simplex method.

Библиографический список

1. *Бахшиян Б.Ц.* Алгоритм для многопараметрических задач линейного программирования, возникающих при решении оптимальных задач коррекции и планирования эксперимента // Устойчивость, управление и моделирование динамических систем: Сб. науч. трудов / Под науч. ред. Г. А. Тимофеевой, д.ф.м.н.; Материалы Международн. науч. конференц., посв. 75-летию со дня рождения И.Я. Каца. 2006. Изд-во УрГУПС. С. 28 -29.

2. *Бахшиян Б.Ц., Федяев К.С.* Об эффективном решении почти вырожденных и плохо обусловленных задач линейного программирования, возникающих при управлении системой // Изв.РАН. ТиСУ. 2005. №4. С.77 — 88.

3. *Лидов М.Л.* Математическая аналогия между некоторыми оптимальными задачами коррекции траекторий и выбора состава измерений и алгоритмы их решения // Космические исследования. 1971. Т.9. № 5.

4. *Бахшиян Б.Ц., Назиров Р.Р., Эльясберг П.Е.* Определение и коррекция движения. — М.: Наука, 1980.

5. *Бахшиян Б.Ц., Соловьев В.Н.* Теория и алгоритмы решения задач L- и MV-оптимального планирования эксперимента // Автоматика и телемеханика. 1998. №8. С.80 — 96.

6. *Войсковский М.И.* Симплексный алгоритм поиска E-оптимальных планов//Изв. РАН. ТиСУ. 2001. №2.

7. *Бахшиян Б.Ц.*, *Войсковский М.И.* О возможности эффективного решения задачи оценивания при линейных ограничениях на оцениваемый вектор//Изв. РАН. ТиСУ. 2003. №4.

8. *Бахшиян Б.Ц.* Критерии оптимальности и алгоритмы решения вырожденной и обобщенной задач линейного программирования // Экономика и мат. методы. 1989. Т.28. № 2. С. 314 — 324.

9. *Бахшиян Б.Ц.*, *Матасов А.И., Федяев К.С.* О решении вырожденных задач линейного программирования // Автоматика и телемеханика. 2000. № 1. С.105 — 117.

10. *Муртаф Б.* Современное линейное программирование. — М.: Мир, 1984.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект № 06-08- 00882-а).

Московский авиационный институт
Статья поступила в редакцию 15.12.2007