Светлой памяти Н. С. Бахвалова посвящается

УДК 519.832.2

МНОГОУРОВНЕВЫЙ МЕТОД РЕШЕНИЯ БОЛЬШИХ МАТРИЧНЫХ ИГР

Е. В. Чижонков[1]

Для численного решения специального класса матричных игр предложен многоуровневый метод. Содержанием работы является адаптация идей метода Федоренко–Бахвалова, хорошо известного как многосеточный метод для решения эллиптических дифференциальных задач, к итерационному решению матричных игр. Работа выполнена при частичной финансовой поддержке РФФИ (код проекта 09–01–00625а).

Ключевые слова: матричные игры, итерационные методы, прямой решатель, базовый итерационный метод, процедура сужения, процедура продолжения, многоуровневый метод.

Введение. В настоящее время наиболее распространенным подходом к решению больших матричных игр является сведение игры к задаче линейного программирования (к LP-проблеме). В частности, это объясняется бурным развитием численных алгоритмов в линейном программировании в течение примерно 25 последних лет начиная с работы [1], фактически инициировавшей развитие метода внутренней точки [2, 3]. При этом известный симплекс-метод, хорошее изложение математической теории которого имеется в [4], также не потерял своего значения. В данном случае эти названия больше отражают идейную сущность, нежели конкретные версии алгоритмов. Отметим, что указанные методы имеют различную теоретическую сложность (полиномиальную в первом случае и экспоненциальную — во втором [5]), однако в вычислительной практике их использование соизмеримо.

С точки зрения решения матричных игр при таком подходе теряется специфика исходной задачи (в первую очередь — игровая интерпретация). Кроме того, как симплекс-метод, так и метод внутренней точки являются прямыми, т.е. приводят к точному решению LP-проблемы за конечное число итераций. Это означает, что практическая размерность решаемой игры существенно ограничена, в первую очередь, возможностями используемого компьютера. Итерационные методы решения больших задач имеют менее обременительные вычислительные ограничения, однако разработка их эффективных версий должна учитывать какие-то особенности решений, т.е. должна быть ориентирована на специальные классы задач.

В настоящей статье в качестве области применения предлагаемого метода имеются в виду большие задачи, допускающие формулировку симметричной матричной игры. Термин “большие”, в первую очередь, имеет отношение к вычислительной мощности современных компьютеров; для определенности будем считать, что представляют интерес задачи с количеством неизвестных $n \geqslant 10^5$ [6]. Важной при этом является структура оптимального решения. Предполагается, что из каких-то априорных соображений известно, что оптимальная стратегия существенным образом смешивается из относительно небольшого, по сравнению с n, количества s чистых стратегий. Однако какие именно чистые стратегии в этом задействованы и в каких пропорциях — неизвестно. “Идеальным” является вариант с $s = O(\ln n)$ и $\sum\limits_{i=1}^{s} p_i = 1$, где p_i — относительные частоты смешиваемых стратегий. Однако вполне допустимо расширение до ситуации: $s = O\left(n^{1/k}\right)$, где $k > 1$ — некоторый параметр, и $\sum\limits_{i=1}^{s} p_i \approx 1$. Имеется в виду, что после упорядочивания по невозрастанию компонент любой оптимальной стратегии значения p_i с номерами $i > s$ быстро убывают. Это имеет аналогию с поведением коэффициентов Фурье достаточно гладкой функции. Например, для 2π-периодических функций вещественной переменной x, допускающих аналитическое продолжение на полосу $\{z : z = x + iy,\ |y| < h,\ -\infty < x < \infty\}$ и ограниченных в ней константой M, справедлива оценка $|a_k| \leqslant M \exp\{-|k|h\}$ [7] для коэффициентов ряда Фурье $f(x) = \sum\limits_{k=-\infty}^{\infty} a_k \exp\{ikx\}$. Поэтому

[1] Московский государственный университет им. М. В. Ломоносова, механико-математический факультет, Ленинские горы, 119899, Москва; профессор, e-mail:chizhonk@mech.math.msu.su

образной характеристикой целевого класса задач может служить некоторая "достаточная гладкость" оптимальных стратегий.

Следует отметить, что формальных ограничений на размерности n и s в предлагаемом методе не имеется. Просто основным фактором является сложность разработки соответствующего программного обеспечения. Другими словами, если программа, реализующая симплекс-метод или метод внутренней точки, успешно справляется с LP-проблемой, равносильной исходной игровой задаче, то обсуждать применение многоуровневого метода в такой ситуации представляется нецелесообразным.

Целью работы в широком смысле является оптимизация итерационных алгоритмов для решения, в первую очередь, матричных игр. По форме это выглядит как перенесение некоторых идей из теории итерационных методов решения линейных алгебраических уравнений, существенно повышающих вычислительную эффективность, в область итерационного решения матричных игр. Наиболее интересной представляется соответствующая адаптация метода Федоренко–Бахвалова [8–10], хорошо известного как многосеточный метод для решения эллиптических и близких к ним задач. По сути — делается попытка на каждой итерации скомбинировать эффективные методы решения LP-проблем, сводя исходную игровую задачу к последовательности вспомогательных игр меньшей размерности; причем как декомпозиция игр, так и обратная композиция стратегий осуществляются из игровых соображений.

Статья организована следующим образом. Сначала приводятся предварительные сведения о постановке матричной игры и методически полезных итерационных методах ее решения. В разделе 2 описывается блочный метод, являющийся естественным (матричным) обобщением метода фиктивной игры, но уже использующий "черный ящик" для прямого (точного) решения вспомогательной игры. С целью подключения в алгоритм прямого решателя дополнительно определены процедура сужения (проектирования) исходной игры на игру фиксированной меньшей размерности и процедура продолжения (смешивания вспомогательной стратегии с приближением на предыдущей итерации). В разделе 3 статьи многоуровневый метод решения матричных игр излагается сначала в абстрактной (рекурсивной) форме, а затем — на содержательных примерах. Его составными частями на каждой итерации являются: базовый итерационный метод (фиксированный, но применяется к играм разной размерности), прямой решатель игр заранее фиксированной размерности, процедуры сужения и продолжения стратегий для двух соседних уровней. Далее формулируются близкие по структуре алгоритмы, как более простые, так и более сложные с вычислительной точки зрения. Эта часть завершается обсуждением проблем сложности, оптимальности и сходимости предлагаемого метода. В заключительных замечаниях определяются направления дальнейших исследований.

Учитывая специфику конструкции многоуровневого метода, в работе по мере необходимости (т.е. достаточно часто) будет проводиться аналогия с методом Федоренко–Бахвалова. Более того, необходимая по ходу изложения терминология будет заимствована из теории этого метода. В отечественной литературе имеется замечательное учебное издание [11] (возможно, одно из лучших в мире), в котором в доступной и одновременно содержательной форме отражены не только все основные технические элементы многосеточного метода, но и многие современные концепции его дальнейшего развития. Регулярные ссылки при изложении многоуровневого метода на [11] преследуют двоякую цель: подчеркнуть преемственность идей и сохранить формальную аналогию с методом Федоренко–Бахвалова в качестве удобной "точки опоры".

1. Предварительные сведения.

1.1. Постановка задачи. Рассмотрим смешанное расширение матричной игры $G = (\boldsymbol{X}, \boldsymbol{Y}, K)$, где $\boldsymbol{X}$ и $\boldsymbol{Y}$ — соответственно $(m-1)$-мерный и $(n-1)$-мерный симплексы, функция K определена на декартовом произведении $\boldsymbol{X} \times \boldsymbol{Y}$: $K(\boldsymbol{x}, \boldsymbol{y}) = \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} x_i a_{ij} y_j$, где $\boldsymbol{x} \in \mathbb{R}^m$, $\boldsymbol{y} \in \mathbb{R}^n$ и $\sum\limits_{i=1}^{m} x_i = \sum\limits_{j=1}^{n} y_j = 1$, $x_i,\ y_j \geqslant 0$. Матрица $A = \|a_{ij}\|$, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, называется матрицей выигрышей. Решением игры G называется пара оптимальных стратегий $(\boldsymbol{x}_*, \boldsymbol{y}_*)$: $\max\limits_{\boldsymbol{x}} \min\limits_{\boldsymbol{y}} K(\boldsymbol{x}, \boldsymbol{y}) = \min\limits_{\boldsymbol{y}} \max\limits_{\boldsymbol{x}} K(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{x}_*, \boldsymbol{y}_*) = V$; величину V называют ценой игры.

Хорошо известно, что решение игры G всегда существует [12], но не обязано быть единственным. Поэтому, как правило, ставится задача нахождения какой-либо пары $(\boldsymbol{x}_*, \boldsymbol{y}_*)$.

Обозначим через $\boldsymbol{e}_i$ вектор, у которого i-я компонента равна единице, а все остальные — нулю, а через $\boldsymbol{l}$ — вектор с единичными компонентами, т.е. $\boldsymbol{l} = (1, \ldots, 1)^{\mathrm{T}}$. Будем считать, что каждый из них имеет размерность, требуемую по смыслу изложения.

1.2. Метод фиктивного розыгрыша игры. Изложим в удобной форме классический алгоритм фиктивного розыгрыша — FP-метод (Fictitious Play), известный как метод Брауна–Робинсон [13, 14]. В

данном случае удобство заключается в интерпретации вычислений с позиции первого игрока, хотя сам фиктивный розыгрыш безусловно является двусторонним.

Определим стартовые значения в методе следующим образом: $k = 1$ (первая итерация), $\boldsymbol{y}(0) = \mathbf{0}$, $\boldsymbol{x}(1) = \boldsymbol{e}_{i(1)} \in \mathbb{R}^m$ с некоторым $1 \leqslant i(1) \leqslant m$. Здесь и далее целочисленное значение аргумента k в скобках означает принадлежность величины к итерации с соответствующим номером, например, зафиксируем для метода $\alpha(k) = 1/k$.

Далее можно считать, что определены величины k, $\boldsymbol{y}(k-1)$, $\boldsymbol{x}(k)$, и переход к значениям $k+1$, $\boldsymbol{y}(k)$, $\boldsymbol{x}(k+1)$ реализуется следующим способом.

Шаг 1. Вычислим вспомогательный вектор-строку $\boldsymbol{v}^{\mathrm{T}}(k) = \boldsymbol{x}^{\mathrm{T}}(k)A \equiv \big(v_1(k), \ldots, v_n(k)\big)$ — накопленный проигрыш второго игрока. Этот вектор удобно интерпретировать как матрицу вспомогательной игры размерности $1 \times n$. Найти решение $\widetilde{\boldsymbol{y}}(k)$ такой игры достаточно просто, так как ее матрица имеет как минимум одну седловую точку. Поэтому процесс решения заключается в определении соответствующей чистой стратегии второго игрока и преобразовании последней в смешанную форму. Приведем формулы для вычислений. Сначала определим значение $j(k) = \arg\min\limits_q v_q(k)$. Если таких несколько, то из них выберем наименьшее. Затем положим $\widetilde{\boldsymbol{y}}(k) = \boldsymbol{e}_{j(k)}$ и одновременно определим приближение к нижней цене игры $w(k) = v_{j(k)}$. Далее для нахождения приближения $\boldsymbol{y}(k)$ проводим смешивание стратегий $\boldsymbol{y}(k-1)$ и $\widetilde{\boldsymbol{y}}(k)$ по формуле: $\boldsymbol{y}(k) = \big(1-\alpha(k)\big)\boldsymbol{y}(k-1) + \alpha(k)\widetilde{\boldsymbol{y}}(k)$.

Шаг 2. Вычислим вспомогательный вектор-столбец $\boldsymbol{u}(k+1) = A\boldsymbol{y}(k) \equiv \big(u_1(k), \ldots, u_m(k)\big)^{\mathrm{T}}$ — накопленный выигрыш первого игрока. Этот вектор удобно интерпретировать как матрицу вспомогательной игры размерности $m \times 1$. Процесс нахождения ее решения $\widetilde{\boldsymbol{x}}(k+1)$ аналогичен описанному на предыдущем шаге. Сначала определим значение $i(k+1) = \arg\max\limits_q u_q(k+1)$. Если таких несколько, то из них выбирается наименьшее. Затем положим $\widetilde{\boldsymbol{x}}(k+1) = \boldsymbol{e}_{i(k+1)}$ и одновременно определим приближение к верхней цене игры $W(k+1) = u_{i(k+1)}$. Наконец, для определения приближения $\boldsymbol{x}(k+1)$ проводим смешивание стратегий $\boldsymbol{x}(k)$ и $\widetilde{\boldsymbol{x}}(k+1)$ по формуле: $\boldsymbol{x}(k+1) = \big(1-\alpha(k+1)\big)\boldsymbol{x}(k) + \alpha(k+1)\widetilde{\boldsymbol{x}}(k+1)$.

Шаг 3. Проводим проверку на окончание итераций; если требуется продолжение вычислений, осуществляется переход к шагу 1.

Часто критерии останова базируются на малости абсолютной погрешности, когда $W(k+1) - w(k) \leqslant \varepsilon$, или на малости относительной погрешности $\dfrac{W(k+1) - w(k)}{\big|V(k)\big|} \leqslant \varepsilon$, где $V(k) = \dfrac{1}{2}\big(W(k+1) + w(k)\big) \neq 0$. При этом за приближенное значение цены игры принимают $V(k)$.

Достоинства и недостатки описанного алгоритма хорошо известны. К первым относятся ясная смысловая (игровая) интерпретация действий и простота вычислений. Платой за это является очень медленная скорость сходимости, как теоретическая (порядок сходимости оценивается как $O\big(k^{-1/(m+n-2)}\big)$ [15] с сильной зависимостью от размерности), так и практическая: $O\big(k^{-1/2}\big)$ [16]. Хотя следует отметить, что с момента установления фундаментального факта сходимости FP-метода [14] не прекращаются попытки его ускорить за счет использования различных модификаций [17–19].

1.3. Монотонный метод. С целью избавиться от немонотонности приближений к цене игры, присущих FP-методу, в работе [20] был предложен и обоснован принципиально другой алгоритм, который будем называть M-методом. Кроме строгого монотонного возрастания нижней границы для цены игры, его дополнительное удобство состоит в реализации действий только одного (первого) игрока. Приведем изложение алгоритма.

Стартовые значения в методе имеют вид: $k = 1$ (первая итерация), $\boldsymbol{x}(1) = \boldsymbol{e}_{i(1)} \in \mathbb{R}^m$ с некоторым $1 \leqslant i(1) \leqslant m$. Далее опишем переход от величин k, $\boldsymbol{x}(k)$ к значениям $k+1$, $\boldsymbol{x}(k+1)$.

Шаг 1. Построение вспомогательной игры $G(k)$. Вычислим $\boldsymbol{v}^{\mathrm{T}}(k) = \boldsymbol{x}^{\mathrm{T}}(k)A \equiv \big(v_1(k), \ldots, v_n(k)\big)$, который является вспомогательным вектором-строкой. По нему определим множество $J(k)$ индексов $j_1, \ldots, j_t$, таких, что

$$\min_q v_q(k) = v_{j_1} = \ldots = v_{j_t}, \quad t \leqslant n, \tag{1}$$

и одновременно определим приближение к нижней цене игры $w(k) = \min\limits_q v_q(k)$. Матрицу выигрышей $A(k)$ размерности $m \times t$, $t \leqslant n$ вспомогательной игры $G(k)$ определим так: $A(k) = \|a_{ij}\|$, $1 \leqslant i \leqslant m$, $j \in J(k)$.

Шаг 2. Находим в игре $G(k)$ любую из стратегий $\widetilde{\boldsymbol{x}}(k) = \big(\widetilde{x}_1(k), \ldots, \widetilde{x}_m(k)\big)^{\mathrm{T}}$, для которой выполнено неравенство $\min\limits_j \sum\limits_{i=1}^{m} \widetilde{x}_i(k)A_i(k) \geqslant V$, где $A_i(k)$ — i-я строка матрицы $A(k)$.

Заметим, что такая стратегия (например, оптимальная в игре $G(k)$) всегда существует, так как в

противном случае второй игрок в исходной игре G мог бы проиграть меньше, чем V, что противоречит определению цены игры.

Шаг 3. Определим параметр смешивания $0 \leqslant \alpha(k) \leqslant 1$ стратегий $\boldsymbol{x}(k)$ и $\widetilde{\boldsymbol{x}}(k)$ из соотношения

$$\max_{\alpha} \min_{q} \big[(1-\alpha)\boldsymbol{v}(k) + \alpha\widetilde{\boldsymbol{v}}(k)\big] = \min_{q}\Big[\big(1-\alpha(k)\big)\boldsymbol{v}(k) + \alpha(k)\widetilde{\boldsymbol{v}}(k)\Big],$$

где $\widetilde{\boldsymbol{v}}^{\,\mathrm{T}}(k) = \widetilde{\boldsymbol{x}}^{\,\mathrm{T}}(k)A \equiv \big(\widetilde{v}_1(k), \ldots, \widetilde{v}_n(k)\big)$. Эта задача эквивалентна решению вспомогательной игры $\widetilde{G}(k)$ с платежной матрицей размерности $2 \times n$ вида $\left\| \begin{matrix} v_1(k) & v_2(k) & \ldots & v_n(k) \\ \widetilde{v}_1(k) & \widetilde{v}_2(k) & \ldots & \widetilde{v}_n(k) \end{matrix} \right\|$.

В этой игре найдем стратегию первого игрока, т.е. вектор вида $\big(1-\alpha(k), \alpha(k)\big)$, и затем вычислим следующее приближение $\boldsymbol{x}(k+1)$ по формуле $\boldsymbol{x}(k+1) = \big(1-\alpha(k)\big)\boldsymbol{x}(k) + \alpha(k)\widetilde{\boldsymbol{x}}(k)$.

Шаг 4. Проводим проверку на окончание итераций и, если требуется продолжение вычислений, осуществляем переход к шагу 1.

Как правило, итерации останавливаются, если $\alpha(k) = 0$ или достигнута заданная точность приближений к цене игры.

Если обозначить через V^* точную верхнюю грань последовательности $\big\{w(k)\big\}$, а через $\boldsymbol{x}^*$ — одну из предельных точек последовательности $\big\{\boldsymbol{x}(k)\big\}$, то результаты работы [20] гарантируют следующее. Во-первых: $w(k+1) > w(k) \quad \forall k \geqslant 1$, во-вторых: $V^* = V$, в-третьих: $\boldsymbol{x}^* = \boldsymbol{x}_*$, где $\boldsymbol{x}_*$ — одна из оптимальных стратегий первого игрока в исходной игре G.

Несмотря на замечательные теоретические характеристики и оригинальный подход к смешиванию стратегий на шаге 3, изложенный метод, к сожалению, носит практически неприемлемый характер. В частности, главной проблемой является реализация шага 2, на котором приходится решать вспомогательную игру с платежной матрицей размерности $m \times t$, $t \leqslant n$, и неизвестной ценой игры V, совпадающей с исходной. Дело в том, что при описанном подходе отсутствуют гарантии понижения размерности, т.е. вспомогательная игра $G(k)$ вполне может быть такого же размера, как и исходная игра G (на самом деле, совпадать с ней). Другой проблемой является неустойчивость (чувствительность к малым возмущениям) метода: целочисленный параметр t — вторая размерность вспомогательной игры — определяется на основе равенства вещественных (в общем случае) чисел (1). При наличии ошибок округлений это, как правило, будет приводить к значению $t = 1$, что делает монотонный алгоритм столь же примитивным, что и FP-метод.

2. Блочный метод. С целью более удобного восприятия метода сначала перечислим его основные элементы, а затем приведем саму формулировку в виде, удобном для реализации.

2.1. Симметричные игры. Предлагаемый в работе метод существенно опирается на тот факт, что матрица A исходной игры $G = (\boldsymbol{X}, \boldsymbol{Y}, K)$ является квадратной ($m = n$) и обладает свойством кососимметричности, т.е. $A = -A^{\mathrm{T}}$. Такие игры называют симметричными. Хорошо известны полезные свойства решений таких игр [12]: произвольное решение $(\boldsymbol{x}_*, \boldsymbol{y}_*)$ игры является симметричным, т.е. $\boldsymbol{x}_* = \boldsymbol{y}_*$, и цена игры равна нулю, т.е. $V = 0$. Кроме того, рассмотрение только симметричных игр позволяет упростить терминологию. Например, термин "кососимметричная матрица размера m" однозначно соответствует симметричной игре с указанной платежной матрицей. При этом слово "кососимметричная" можно опустить, так как встречающиеся далее игры (как исходная, так и вспомогательные), как правило, будут симметричными. Там же, где встретится игра общего вида, в явном виде будут указаны обе размерности матрицы.

Отметим, что рассмотрение процесса решения только симметричных игр не ограничивает общности подхода, так как для произвольной игры с $(m \times n)$-матрицей B существуют ее симметричные расширения за счет увеличения размерности. Видимо, впервые это было замечено в работе [21] (см. также [22]), где приведено два различных способа симметризации. При этом наиболее предпочтительным из них (с точки зрения размера получаемой симметричной игры) считается тот, в котором матрица $A = -A^{\mathrm{T}}$ имеет размер $m+n+1$ и следующий блочный вид: $A = \begin{pmatrix} 0 & B & -\boldsymbol{l} \\ -B^{\mathrm{T}} & 0 & \boldsymbol{l} \\ \boldsymbol{l}^{\mathrm{T}} & -\boldsymbol{l}^{\mathrm{T}} & 0 \end{pmatrix}$, где нулем обозначены состоящие только из нулей матрицы соответствующего размера, а обозначение $\boldsymbol{l}$ вводилось в разделе 1.1.

Скорее всего, можно предложить и другие способы симметризации игр; причем учет специфики получающихся при этом матриц безусловно приведет к увеличению эффективности вычислительных алгоритмов. Однако при изложении содержания данной работы никакие другие свойства матрицы игры (кроме кососимметричности) использоваться не будут.

Отметим, что так как основная процедура итерационных методов — умножение матрицы на вектор, то для вычислительной устойчивости матрицу исходной игры следует отнормировать, поделив ее на $\max\limits_{i,j}\left|a_{i,j}\right|$; тогда у полученной матрицы все элементы будут принадлежать отрезку $[-1,1]$. Именно для таких кососимметрических матриц будет проводиться все дальнейшее изложение алгоритмов.

2.2. Тестирование на седловые точки. Поскольку у матрицы произвольной игры (не обязательно симметричной) возможно существование одной или нескольких седловых точек, каждая из которых соответствует решению в чистых стратегиях, то такую ситуацию необходимо отслеживать с помощью SPS-процедуры (Saddle Point Solver). Ее смысл достаточно прост: для заданной матрицы требуется дать ответ о наличии по крайней мере одной седловой точки; если ответ положителен, то необходимо предъявить какое-либо решение соответствующей игры.

Например, в рассматриваемом случае симметричных игр можно предложить следующую процедуру. Обычным минимаксным перебором определяется множество седловых точек матрицы и соответствующий им набор I чистых стратегий $i_1,\ldots,i_t$. Если этот набор пуст, то формируется отрицательный ответ. В противном случае можно предъявить решение в виде произвольной выпуклой линейной комбинации векторов $\boldsymbol{e}_{i_1},\ldots,\boldsymbol{e}_{i_t}$. Разумным представляется выбор $\boldsymbol{y}_*=\sum\limits_{i\in I}\boldsymbol{e}_i/t$, который в некотором смысле несет информацию о всех седловых точках одновременно.

2.3. Прямой решатель. Ключевым элементом алгоритма является наличие в нем прямого (точного, если отсутствуют ошибки округлений) решателя симметричной игры с матрицей B порядка s; назовем его DS-процедурой (Direct Solver). Этот решатель может быть реализован в виде "черного ящика": на вход подаем кососимметричную матрицу B, на выходе имеем одно из возможных точных решений $\boldsymbol{y}_*$. В частности, в качестве одной из структурных частей DS-процедуры может выступать SPS-процедура. Хотя, в первую очередь, имеется в виду, что указанный решатель будет основан на сведении игры размера s к задаче линейного программирования, которая, в свою очередь, будет решаться симплекс-методом или методом внутренней точки.

При сведении вспомогательной игры к LP-проблеме рекомендуется сначала отнормировать элементы матрицы так, чтобы выполнялось неравенство $\left|a_{ij}\right|\leqslant 1$, а затем все элементы полученной матрицы увеличить на единицу; тогда абсолютная погрешность в цене игры будет совпадать с относительной.

Если n — размерность исходной игры, то формальное ограничение на s имеет вид $1\leqslant s<n$. Очень важно, что значение s фиксируется изначально и не меняется в процессе итераций. Заметим, что принципиальная возможность использования для s всего диапазона значений (от 1 до $(n-1)$) представляет интерес только для теоретических исследований. В практических же приложениях имеет смысл выбирать $3\leqslant s\ll n$, так как вычислительная стоимость DS-процедуры может расти экспоненциально (или, как минимум, полиномиально) в зависимости от s, а симметричная игра при $s\leqslant 2$ всегда имеет решение в чистых стратегиях.

Проведем аналогию с многосеточным методом решения эллиптических задач [11]. Для получения решения на самой мелкой сетке всегда имеется теоретическая возможность использовать метод Гаусса, однако при практической реализации его применяют только для решения вспомогательных задач на самой крупной сетке, учитывая кубический рост вычислительной сложности метода в зависимости от размерности системы.

Отметим, что главное содержание блочного метода — работа только со смешанными стратегиями, что принципиально отличает его от FP-метода, основанного на определении вспомогательных чистых стратегий с их последующим подмешиванием. Напомним, что в самом FP-методе это реализуется на шагах 1 и 2, где используются тривиальные прямые решатели (на самом деле, SPS-процедуры) для вспомогательных игр с матрицами $1\times n$ и $m\times 1$ соответственно.

С другой стороны, имеется также принципиальное отличие и от M-метода, в котором предполагается использование прямых решателей для игр как с матрицами постоянных размерностей $2\times n$ (на шаге 3), так и переменных — $m\times t$, $t\leqslant n$ (на шаге 2 от итерации к итерации может меняться значение t). Подчеркнем, что в описываемом методе размерность s прямого решателя симметричной игры фиксирована и строго меньше n.

2.4. Сужение (проектирование). Чтобы применить DS-процедуру на каждой итерации, требуется сформировать вспомогательную игру размера $s<n$ в зависимости от имеющегося приближения $\boldsymbol{x}(k)$ к решению исходной игры с матрицей A; назовем это R-процедурой (restriction).

Вычислим вспомогательный вектор-строку $\boldsymbol{v}^{\mathrm{T}}(k)=\boldsymbol{x}^{\mathrm{T}}(k)A\equiv\big(v_1(k),\ldots,v_n(k)\big)$ и упорядочим его компоненты неубывающим образом, запомнив при этом их исходные номера из структуры вектора $\boldsymbol{v}$. Если несколько компонент окажутся одинаковыми, то проведем их дополнительное упорядочивание каким-

либо фиксированным образом (например, по возрастанию их исходных номеров). Пусть после упорядочивания компоненты вектора $\boldsymbol{v}$ расположились следующим образом: $v_{j_1} \geqslant v_{j_2} \geqslant \ldots \geqslant v_{j_n}$, тогда определим множество J как набор первых s индексов $j_1, \ldots, j_s$. Вычислим приближение к цене игры — $w(k) = v_{j_1}$ и построим по J матрицу проектирования $D(k)$ размерности $s \times n$ с ненулевыми элементами $d_{1,j_1} = d_{2,j_2} = \ldots = d_{s,j_s} = 1$ и остальными — $d_{i,j} = 0$. Теперь матрицу выигрышей $A(k)$ размерности s вспомогательной игры определим так: $A(k) = \|a_{ij}\|$, $i \in J$, $j \in J$, или, что то же самое, $A(k) = D(k)AD^{\mathrm{T}}(k)$.

Таким образом, R-процедура на итерации с номером k по некоторому приближению $\boldsymbol{x}(k)$ к решению исходной игры с матрицей A определяет: $w(k)$ — приближение к цене игры, $\boldsymbol{v}(k)$ — вектор выигрыша–проигрыша, $D(k)$ — матрицу проектирования, $A(k)$ — матрицу вспомогательной игры, готовую для применения DS-процедуры.

2.5. Продолжение (смешивание). Чтобы использовать результат DS-процедуры на итерации с номером k, требуется преобразовать решение игры размера $s < n$ в некоторую смешанную стратегию исходной игры, а затем каким-то образом смешать ее с предыдущим приближением $\boldsymbol{x}(k)$; назовем этот процесс P-процедурой (prolongation).

Это удобно реализовать, например, как в M-методе на шаге 3. Обозначим результат DS-процедуры через $\boldsymbol{y}(k)$ и определим вспомогательную стратегию $\widetilde{\boldsymbol{x}}(k) = D^{\mathrm{T}}(k)\boldsymbol{y}(k)$, где $D(k)$ — матрица проектирования из R-процедуры. Далее параметр смешивания $0 \leqslant \alpha(k+1) \leqslant 1$ предыдущей и вспомогательной стратегий определяется из соотношения

$$\max_{\alpha} \min_{q} \big[(1-\alpha)\boldsymbol{v}(k) + \alpha\widetilde{\boldsymbol{v}}(k)\big] = \min_{q} \Big[\big(1-\alpha(k+1)\big)\boldsymbol{v}(k) + \alpha(k+1)\widetilde{\boldsymbol{v}}(k)\Big],$$

где $\widetilde{\boldsymbol{v}}^{\,\mathrm{T}}(k) = \widetilde{\boldsymbol{x}}^{\,\mathrm{T}}(k)A \equiv \big(\widetilde{v}_1(k), \ldots, \widetilde{v}_n(k)\big)$, вектор $\boldsymbol{v}(k)$ берется из R-процедуры, а сам результат смешивания получается по формуле $\boldsymbol{x}(k+1) = \big(1-\alpha(k+1)\big)\boldsymbol{x}(k) + \alpha(k+1)\widetilde{\boldsymbol{x}}(k)$. Недостаточность детализации изложения можно восполнить из пункта 1.2 (см. шаг 3). Напомним, что определение $\alpha(k)$ эквивалентно решению дополнительной (в общем случае несимметричной) игры с матрицей размерности $2 \times n$, что может быть эффективно реализовано различными способами, например, с помощью симплекс-метода для равносильной LP-проблемы. При этом, если $\widetilde{\boldsymbol{x}}(k)$ совпадет с оптимальной стратегией, то параметр $\alpha(k+1)$ автоматически станет равным единице. В случае продолжения итераций значения приближений уже меняться не будут, так как все последующие значения $\alpha(q) = 0$ при $q > k+1$.

Таким образом, P-процедура на итерации с номером k по результату $\boldsymbol{y}(k)$ DS-процедуры и $D(k)$ — матрицы проектирования устанавливает следующее приближение $\boldsymbol{x}(k+1)$ к решению исходной игры.

2.6. Формулировка метода. Пусть $k = 1$ (первая итерация) и имеется некоторое приближение к решению $\boldsymbol{x}(1)$.

При отсутствии априорной информации о решении представляется разумным сначала применить SPS-процедуру, а в случае отрицательного ответа — смешать все стратегии в одинаковой пропорции, т.е. $\boldsymbol{x}(1) = \sum_{i=1}^{n} \boldsymbol{e}_i/n$.

Одну итерацию блочного метода можно описать следующей последовательностью шагов. Итерации останавливаются, если достигнута заданная точность приближений к цене игры или $\alpha(k) = 0$.

Шаг 1. R-процедура. Вход: $\boldsymbol{x}(k)$. Выход: $w(k)$, $\boldsymbol{v}(k)$, $D(k)$, $A(k)$.

Шаг 2. DS-процедура (с обязательным применением SPS-процедуры внутри). Вход: $A(k)$. Выход: $\boldsymbol{y}(k)$.

Шаг 3. P-процедура. Вход: $\boldsymbol{y}(k)$, $D(k)$. Выход: $\boldsymbol{x}(k+1)$, $\alpha(k+1)$.

Шаг 4. Проверка на окончание итераций; если требуется продолжение вычислений, осуществляем переход к шагу 1.

Легко видеть, что блочный метод является промежуточным (между FP-методом и M-методом) алгоритмом с точки зрения вычислительной сложности. При этом он конструктивно и устойчиво реализуем в отличие от M-метода, а также способен эффективно (при небольших s) генерировать смешанные вспомогательные стратегии, что качественно его отличает от FP-метода.

Здесь уместно провести аналогию с ситуацией из теории итерационных методов решения систем линейных уравнений. Например, блочный метод Якоби (см., например, [23]) более привлекателен для расчетов по сравнению со своим точечным вариантом, так как допускает применение к матрицам, содержащим нулевые элементы на диагонали, и, кроме того, всегда сходится не медленнее, чем точечный. Однако за это приходится расплачиваться большей трудоемкостью каждой итерации — прямым решением вспомогательных подсистем с квадратными матрицами меньшего размера. Поэтому для сравнения

реальной вычислительной эффективности необходимо учитывать не только количество итераций методов, но и затраты на одну итерацию.

При использовании блочного метода для решения симметричных матричных игр, видимо, следует ожидать сходную тенденцию: увеличение параметра s — размерности прямого решателя — будет сокращать общее число итераций для достижения заданной точности, одновременно увеличивая вычислительную сложность (трудоемкость) каждой итерации. Поэтому можно предложить следующее правило выбора s в зависимости от размерности n решаемой задачи: вычислительные затраты DS-процедуры должны быть соизмеримы с остальными затратами (P- и R-процедурами) на итерации. Например, трудоемкость симплекс-метода для решения LP-проблемы размерности $s \times s$ можно оценить как $O\left(2^{s/2}\right)$ [5]; при этом количество остальных вычислений составляет $O\left(n^2\right)$ действий, пропорциональное умножению матрицы на вектор; отсюда заключаем, что $s = O\left(\log_2 n\right)$. При таком соотношении вычислительные сложности итераций блочного метода и FP-метода сопоставимы, но следует ожидать существенного снижения общего числа итераций в блочном варианте.

Отметим, что существуют различные модификации FP-метода, учитывающие как просто кососимметричность матрицы игры, так и ее конкретную структуру [18, 19], которые приводят к значительному сокращению числа итераций. Мы не будем останавливаться на их адаптации применительно к блочному методу, так как, с одной стороны, теоретическое преимущество модификаций не установлено (имеются данные только численных экспериментов), а с другой — блочный метод представляет интерес, в первую очередь, методический: он исключительно удобен для изложения структуры многоуровнего метода. Однако напомним, что учет специфики решаемой задачи всегда приводит к повышению эффективности алгоритма, и отложим изучение модификаций блочного метода для симметричных игр на будущее.

3. Многоуровневый метод. Зафиксируем, что многоуровневый метод строится для симметричной матричной игры, т.е. $A = -A^{\mathrm{T}}$.

Блочный метод, описанный в предыдущем разделе, имеет ясную игровую интерпретацию. Изначально предполагается, что оптимальная стратегия смешивается не более чем из s чистых стратегий. Затем на каждой итерации, на основании имеющегося приближения выбирается s наиболее перспективных чистых стратегий, исходная задача "проектируется" на это подмножество и задача меньшего размера решается точно. Чтобы получить новое приближение, имеющееся приближение смешивается с полученным вспомогательным так, чтобы цена игры не уменьшилась.

Идея блочного метода не является новой, она, в частности, встречалась в [17]. Предложенная здесь версия отличается способом смешивания (в P-процедуре) и типом прямого решателя. В [17] предполагалось, что DS-процедура имеет игровую структуру, а здесь отдается предпочтение сведению к LP-проблеме.

Главным недостатком блочного метода является так называемое "заедание", т.е. ситуация, когда число примерно одинаковых наибольших компонент вектора $\boldsymbol{v}$ в R-процедуре больше s или когда оптимальная стратегия смешивается более, чем из s стратегий, тогда получение удовлетворительного решения может потребовать значительного числа итераций. Для преодоления этой трудности нужны некоторые дополнительные усилия; в частности, справляющимся с проблемой примером может служить двухуровневый метод.

3.1. Двухуровневый метод. Одна итерация двухуровневого метода является результатом семи последовательных шагов начиная с некоторого приближения $\boldsymbol{x}(k)$.

Здесь предполагается, что кроме элементов блочного метода в нашем распоряжении дополнительно имеется базовый итерационный алгоритм несложной структуры (например, FP-метод или какая-то его модификация для симметричных игр; см. также методы из[17]), требующий вычислительных затрат на итерации не более $O\left(n^2\right)$; назовем его T-методом. Приведем описание одной итерации метода, детализируя параметры шагов там, где это необходимо.

Шаг 1. Стартуя с начального приближения $\boldsymbol{x}(k) \equiv \boldsymbol{z}^0$, сделаем ν итераций T-метода; в результате получим приближение $\boldsymbol{z}^\nu$.

Шаг 2. R-процедура блочного метода. Вход: $\boldsymbol{z}^\nu$.

Шаг 3. DS-процедура (с обязательным применением SPS-процедуры внутри). Выход: $\boldsymbol{y}(k)$.

Шаг 4. Продолжение (т.е. дополнение нулями до требуемой размерности) $\boldsymbol{b}^0 = D^{\mathrm{T}}(k)\boldsymbol{y}(k)$, где $D(k)$ — матрица проектирования из R-процедуры.

Шаг 5. Стартуя с начального приближения $\boldsymbol{b}^0$, сделаем δ итераций T-метода; в результате получим приближение $\boldsymbol{b}^\delta$.

Шаг 6. Смешивание стратегий $\widetilde{\boldsymbol{x}}(k) \equiv \boldsymbol{b}^\delta$ и $\boldsymbol{z}^\nu$, как в P-процедуре блочного метода. Выход: $\boldsymbol{x}(k+1)$, $\alpha(k+1)$.

Шаг 7. Проводим проверку на окончание итераций и, если требуется продолжение вычислений, осуществляем переход к шагу 1.

В двухуровневом методе, в отличие от блочного, P-процедура конструктивно состоит из шагов 4–6, т.е. между продолжением и смешиванием стратегий встроено $\delta \geqslant 0$ итераций T-метода.

Название метода "двухуровневый" отражает тот факт, что содержательные вычисления проводятся на двух уровнях, характеризуемых различными размерностями решаемых задач: на первом уровне в задаче имеется n неизвестных, на втором — s. На самом деле, удобнее нумеровать уровни с нулевого и по возрастанию числа неизвестных, т.е. в данном случае считать уровень с s неизвестными нулевым, а с n неизвестными — первым.

3.2. Формальное (рекурсивное) описание многоуровнего метода. Напомним, что самый трудоемкий в вычислительном аспекте шаг блочного метода (и, соответственно, двухуровневого) — это DS-процедура на нулевом уровне. Однако ее следствием (после продолжения и смешивания) является лишь очередное приближение к оптимальной стратегии. Поэтому совершенно необязательно находить точное решение вспомогательной задачи, достаточно ограничиться некоторым разумным приближением к результату DS-процедуры. Для получения этого "разумного" приближения можно, в свою очередь, сделать несколько (общим числом γ) итераций двухуровнего метода, переходя в нем на уровни с все меньшими количествами переменных. *Многоуровневый метод* заключается в рекурсивном повторении этой процедуры, пока не будет достигнут уровень с малым количеством неизвестных s, на котором применение DS-процедуры уже не будет обременительным с точки зрения объема вычислений.

Формализуем идею многоуровневого метода; будем называть его ML-методом (Multi-Level). Пусть задана иерархия симплексов

$$\boldsymbol{X}_0 \subset \boldsymbol{X}_1 \subset \ldots \subset \boldsymbol{X}_m \subset \ldots \subset \boldsymbol{X}_l = \boldsymbol{X}, \tag{2}$$

характеризующих смешанные стратегии вспомогательных игр. Отметим, что исходной игре соответствует $(n-1)$-мерный симплекс $\boldsymbol{X}$ и что указанные симплексы определяются только своими размерностями; в частности, размерность $\boldsymbol{X}_0$ совпадает с размерностью прямого решателя s. Сами симплексы связаны друг с другом последовательно через процедуры сужения и продолжения:

$$\boldsymbol{X}_0 \underset{R}{\overset{P}{\rightleftarrows}} \boldsymbol{X}_1 \underset{R}{\overset{P}{\rightleftarrows}} \cdots \underset{R}{\overset{P}{\rightleftarrows}} \boldsymbol{X}_m \underset{R}{\overset{P}{\rightleftarrows}} \cdots \underset{R}{\overset{P}{\rightleftarrows}} \boldsymbol{X}_l.$$

Будем считать, что матрица исходной игры $A_l = A$ и матрицы A_m, $m = 0, 1, \ldots, l-1$, определены (например, вычисляются про правилам R-процедуры). Номер уровня будем отмечать нижним индексом. Из уровней с номерами $(m+1)$ и m уровень с меньшим числом неизвестных (равным n_m) будем называть *грубым*.

Зафиксируем натуральное ν, число итераций T-метода на каждом уровне, и натуральное γ, число рекурсивных вызовов метода на каждом уровне в иерархии (2). Одна итерация многоуровнего метода решения исходной симметричной игры с матрицей A_l определяется следующей рекурсивной процедурой: $\boldsymbol{x}(k+1) = \mathrm{ML}\big(l, \boldsymbol{x}(k)\big)$, где $\boldsymbol{x}(k)$ и $\boldsymbol{x}(k+1)$ соответствуют последовательным приближениям к оптимальной стратегии на самом высоком уровне, т.е. в исходной игре. Приведем рекурсивное описание процедуры вычисления $\overline{\boldsymbol{z}} = \mathrm{ML}(m, \boldsymbol{z})$

{

Если $m = 0$, то

0. Полагаем $\overline{\boldsymbol{z}}$ равным результату DS-процедуры (точное решение на уровне с минимальным числом неизвестных s), далее выход из процедуры.

Иначе $(m > 0)$

1. Полагаем $\boldsymbol{z}^0 = \boldsymbol{z}$ и делаем ν итераций T-метода с этим начальным приближением, в результате имеем $\boldsymbol{z}^\nu$.

2. R-процедура (сужение на грубый уровень), SPS-процедура (тестирование на седловые точки соответствующей подматрицы). Если результат положителен, то обозначаем соответствующую стратегию через $\boldsymbol{y}^\gamma$ и переходим к шагу 4, т.е. фактически сразу "отрезаем" все низшие уровни.

3. Выбираем начальное приближение $\boldsymbol{y}^0$ к игре на грубом уровне и делаем γ итераций многоуровнего метода по формуле $\boldsymbol{y}^{i+1} = \mathrm{ML}(m-1, \boldsymbol{y}^i)$ для $i = 0, 1, \ldots, \gamma-1$.

4. P-процедура (продолжение, итерирование и смешивание стратегий для возвращения на предыдущий уровень): $\boldsymbol{b}^0$ — результат продолжения $\boldsymbol{y}^\gamma$, $\boldsymbol{b}^\delta$ — результат $\delta \geqslant 0$ итераций T-метода, $\overline{\boldsymbol{z}}$ — результат смешивания стратегий $\boldsymbol{b}^\delta$ и $\boldsymbol{z}^\nu$.

}

Одно выполнение процедуры $\boldsymbol{x}(k+1) = \mathrm{ML}\big(l, \boldsymbol{x}(k)\big)$ является одной итерацией $(l+1)$-уровнего метода. Всего будем выполнять заданное число итераций или пока не будет достигнуто приближение к цене игры с заданной точностью.

Представляется полезным после завершения каждой итерации многоуровневого метода проводить перенумерацию неизвестных в соответствии с упорядочиванием по невозрастанию компонент вектора $\boldsymbol{x}(k+1)$. В частности, это приведет к постепенному перемещению подматрицы, определяющей смешивающиеся в решении чистые стратегии, в левый верхний угол общей матрицы игры. Результаты такого процесса могут использоваться, во-первых, для сокращения вычислительных затрат при определении матриц вспомогательных игр при переходе с уровня на уровень и, во-вторых, для апостериорного анализа полученного решения.

3.3. Примеры вариантов многоуровнего метода. Рассмотрим характерные варианты ML-метода, соответствующие выбору $\gamma = 1$ и $\gamma = 2$. В первом случае одну итерацию метода удобно называть V-циклом, а второго — W-циклом. Названия обоих вариантов иллюстрируются рис. 1 и 2 с использованием четырех уровней.

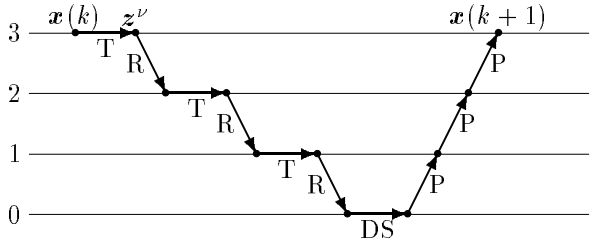

Рис. 1. Полный V-цикл

Рассмотрим рис. 1. На нем стрелками указано направление передачи данных в соответствии с иерархией уровней: стрелка вниз с буквой R означает применение R-процедуры, стрелка вверх с буквой P — применение P-процедуры, горизонтальная стрелка с буквой T — применение базового итерационного метода на уровнях $0 < m \leqslant l$, горизонтальная стрелка с буквами DS — применение прямого (точного) решателя на уровне $m = 0$. На рисунке не отражено только применение SPS-процедуры: места ее использования совпадают с точками, в которых заканчиваются стрелки вниз. Если результат тестирования на наличие седловых точек положителен, то происходит движение по горизонтали до пересечения с точкой начала стрелки вверх, т.е. происходит "отрезание" всех уровней с меньшим числом переменных. На рисунке такая ситуация не отображена, поэтому его название — "полный" V-цикл.

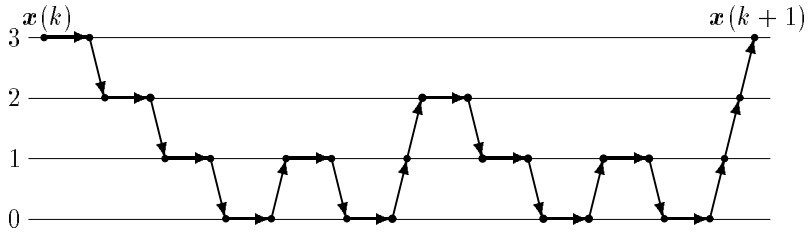

Рис. 2. Полный W-цикл

На рис. 2 все буквы, обозначающие применение процедур, убраны с целью большей наглядности. Смысл всех стрелок точно такой же, как на рис. 1. Не следует только забывать, что горизонтальные стрелки на нулевом (нижнем) уровне обозначают применение DS-процедуры (в данном случае четыре раза), а на остальных уровнях — базовые итерации T-метода (семь раз).

Укажем две близкие модификации многоуровнего метода. Первая из них, F-цикл, является некоторым компромиссным вариантом между V-циклом и W-циклом с точки зрения вычислительной сложности. В нем для решения игры на уровне с меньшим числом неизвестных процедура вызывает один раз себя, а потом один раз V-цикл. Другая модификация ML-метода, *постсглаживание*, связана с возможностью проведения базовых итераций на каждом уровне после P-процедуры. Их аналоги в многосеточном ме-

тоде [11] хорошо себя зарекомендовали: первый — с практической, а второй — с теоретической точки зрения.

Легко заметить, что для полноты описания ML-метода необходимо соответствующим образом модифицировать R- и P-процедуры и определить способ выбора начального приближения при переходе к игре с меньшим количеством неизвестных.

3.4. Необходимые модификации процедур блочного метода. Рассмотрим два соседних уровня с номерами $m+1$ и m, размерности соответствующих пространств стратегий обозначим через n_{m+1} и n_m. На уровне $m+1$ известны: матрица вспомогательной игры A_{m+1} и некоторое приближение к оптимальной стратегии $\boldsymbol{x}_{m+1}$. Напомним, что $\boldsymbol{x}_m$ возникает после применения T-метода на m-м уровне.

Начнем с модификации R-процедуры (сужения). Как и в пункте 2.4, вычислим вспомогательный вектор $\boldsymbol{v}^{\mathrm{T}} = \boldsymbol{x}_{m+1}^{\mathrm{T}} A_{m+1} \equiv \bigl(v_1, \ldots, v_{n_{m+1}}\bigr)$ и упорядочим его компоненты неубывающим образом, запомнив при этом их исходные номера из структуры вектора $\boldsymbol{v}$. Если несколько компонент окажутся одинаковыми, то проведем их дополнительное упорядочивание каким-либо фиксированным образом (например, по возрастанию их исходных номеров). Пусть после упорядочивания компоненты вектора $\boldsymbol{v}$ расположились следующим образом: $v_{j_1} \geqslant v_{j_2} \geqslant \ldots \geqslant v_{j_{n_{m+1}}}$, тогда определим множество J как набор первых n_m индексов $j_1, \ldots, j_{n_m}$ и построим по J матрицу проектирования D_l размерности $n_m \times n_{m+1}$ с ненулевыми элементами $d_{1,j_1} = d_{2,j_2} = \ldots = d_{s,j_{n_m}} = 1$ и остальными — $d_{i,j} = 0$. Теперь матрицу A_m порядка n_m вспомогательной игры на уровне m определим так: $A_m = \bigl\|a_{ij}\bigr\|$, $i \in J$, $j \in J$, или, что то же самое, $A_m = D_m A_{m+1} D_m^{\mathrm{T}}$.

Дополнительно требуется определить начальное приближение $\boldsymbol{y}^0$ для итераций многоуровнего метода при старте с уровня m (см. шаг 3 ML-метода.) Для этого, например, построим вектор $\widetilde{\boldsymbol{y}} = D_m \boldsymbol{x}_{m+1}$ длины n_m и промасштабируем его по формуле $\boldsymbol{y}^0 = \widetilde{\boldsymbol{y}} / \sum\limits_{i=1}^{n_m} \widetilde{y}_i$. Геометрически это соответствует продолжению вектора $\widetilde{\boldsymbol{y}}$ до пересечения с симплексом $\boldsymbol{X}_{n_m}$. Если вектор $\widetilde{\boldsymbol{y}}$ окажется состоящим только из нулей, то воспользуемся в качестве начального приближения равносмешанной стратегией: $\boldsymbol{y}^0 = \sum\limits_{i=1}^{n_m} \dfrac{\boldsymbol{e}_i}{n_m}$. С теоретической точки зрения более предпочтительным представляется другой вариант — в качестве $\boldsymbol{y}^0$ использовать проекцию $\widetilde{\boldsymbol{y}}$ на $\boldsymbol{X}_{n_m}$.

Требуется также модифицировать P-процедуру для возвращения с уровня m на уровень $m+1$; теперь она будет состоять из трех частей. Перед ее применением мы имеем матрицу проектирования D_{m+1}, полученную из R-процедуры на основе приближения $\boldsymbol{x}_{m+1}$, и некоторое приближение к оптимальной стратегии $\boldsymbol{y}_m$, вычисленное на уровне m. Сначала, как и на шаге 4 двухуровнего метода, построим вспомогательную стратегию $\boldsymbol{b}^0 = D_{m+1}^{\mathrm{T}} \boldsymbol{y}_m$, т.е. дополним нулями вектор $\boldsymbol{y}_m$ до требуемой размерности. Затем, как на шаге 5 двухуровнего метода, с начальным приближением $\boldsymbol{b}^0$ и матрицей A_{m+1} проведем $\delta \geqslant 0$ итераций T-метода; получим $\boldsymbol{b}^\delta \equiv \widetilde{\boldsymbol{x}}_{m+1}$. Наконец, как на шаге 6 двухуровнего метода, определим параметр смешивания $0 \leqslant \alpha_{m+1} \leqslant 1$ стратегий $\boldsymbol{x}_{m+1}$ и $\widetilde{\boldsymbol{x}}_{m+1}$ из соотношения

$$\max_{\alpha} \min_{q} \bigl[(1-\alpha)\boldsymbol{v} + \alpha \widetilde{\boldsymbol{v}}\,\bigr] = \min_{q} \bigl[(1-\alpha_{m+1})\boldsymbol{v}(k) + \alpha_{m+1} \widetilde{\boldsymbol{v}}\,\bigr],$$

где $\widetilde{\boldsymbol{v}}^{\mathrm{T}} = \widetilde{\boldsymbol{x}}_{m+1}^{\mathrm{T}} A_{m+1} \equiv \bigl(\widetilde{v}_1, \ldots, \widetilde{v}_{n_{m+1}}\bigr)$. При этом сам результат смешивания получается по формуле $\boldsymbol{y}_{m+1} = (1-\alpha_{m+1})\boldsymbol{x}_{m+1} + \alpha_{m+1} \widetilde{\boldsymbol{x}}_{m+1}$.

3.5. Аддитивный многоуровневый метод. Сформулируем важное замечание: ничто не мешает интерпретировать θ итераций ML-метода как процесс *точного* решения некоторой *вспомогательной* игры, т.е. в качестве предобусловливающего оператора, используя терминологию итерационных методов решения линейных уравнений. Поясним это подробнее. Пусть для игры с матрицей A и начальным приближением $\boldsymbol{y}^0$ используется какой-либо вариант многоуровневого метода, тогда после проведения θ итераций (циклов) получим приближение $\boldsymbol{y}^\theta$. Будем считать, что $\boldsymbol{y}^\theta$ является точным решением вспомогательной игры с некоторой матрицей $A(\theta)$; в этом случае про начальное приближение уже можно забыть. Когда $\theta = \infty$, без ограничения общности можно считать, что $A(\theta) = A$, $\boldsymbol{y}^\theta = \boldsymbol{x}_*$.

Предположим, что для решения исходной игры имеется некоторый итерационный метод (назовем его *внешним*). Если он является стандартным процессом [17], соседние приближения к решению связаны соотношением (смешиванием) вида

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \alpha \boldsymbol{r}(k), \tag{3}$$

где $\alpha \equiv \alpha(k+1)$ — итерационный параметр, а $\boldsymbol{r}(k)$ каким-либо образом порожден вектором $\boldsymbol{x}(k)$, например $\boldsymbol{r}(k) = \widetilde{\boldsymbol{x}}(k) - \boldsymbol{x}(k)$, $\widetilde{\boldsymbol{x}}(k)$ — результат вспомогательных действий. Пусть для рассматриваемого

примера в соотношении (3) вектор $\widetilde{\boldsymbol{x}}(k)$ совпал с решением $\boldsymbol{x}_*$, тогда при $\alpha = 1$ метод сойдется за одну итерацию, независимо от $\boldsymbol{x}(k)$. Однако процесс вычисления такого "точного приближения" $\widetilde{\boldsymbol{x}}(k)$ имеет большую вычислительную сложность (равную трудоемкости исходной задачи). Поэтому хочется найти компромисс между большим объемом расчетов на одной итерации и небольшим количеством существенно менее затратных итераций.

Идея метода состоит в том, что предлагается использовать на каждой итерации внешнего итерационного метода θ итераций многоуровнего метода: $\boldsymbol{y}^0 = \boldsymbol{x}(k)$, $\widetilde{\boldsymbol{x}}(k) = \boldsymbol{y}^\theta$. В этом случае естественно называть многоуровневый метод *внутренними* итерациями. Представляется правдоподобным, что такого рода комбинация (назовем ее *аддитивным* многоуровневым, или AML-методом) может быть значительно эффективнее исходного ML-метода. Приведем формальное описание одной итерации алгоритма вычисления $\overline{\boldsymbol{z}} = \mathrm{AML}(m, \boldsymbol{z})$

{

Если $m = 0$, то

0. Полагаем $\overline{\boldsymbol{z}}$ равным результату DS-процедуры (точное решение на уровне с минимальным числом неизвестных s), далее выход из процедуры.

Иначе $(m > 0)$

1. Полагаем $\boldsymbol{z}^0 = \boldsymbol{z}$ и делаем ν итераций T-метода с этим начальным приближением, в результате имеем $\boldsymbol{z}^\nu$.

2. R-процедура (сужение на грубый уровень), SPS-процедура (тестирование на седловые точки соответствующей подматрицы). Если результат положителен, то обозначаем соответствующую стратегию через $\boldsymbol{y}$ и переходим к шагу 4, т.е. фактически сразу "отрезаем" все низшие уровни.

3. Выбираем начальное приближение $\boldsymbol{y}^0$ к игре на грубом уровне и выполняем многоуровневый метод по формуле $\boldsymbol{y} = \mathrm{AML}(m-1, \boldsymbol{y}^0)$.

4. P-процедура (продолжение, итерирование и смешивание стратегий для возвращения на предыдущий уровень): $\boldsymbol{b}^0$ — результат продолжения $\boldsymbol{y}$, $\boldsymbol{b}^\delta$ — результат $\delta \geqslant 0$ итераций T-метода, $\overline{\boldsymbol{z}}$ — результат смешивания стратегий $\boldsymbol{b}^\delta$ и $\boldsymbol{z}^\nu$.

}

Легко видеть, что шаг 3 не зависит от шага 1 и может выполняться параллельно, что увеличивает привлекательность AML-метода.

Заманчивой представляется идея использовать в качестве внешнего метода хороший полиномиальный алгоритм для решения равносильной LP-проблемы (типа метода внутренней точки). Отметим, что методы внутренней точки по сути сводят точное решение задачи линейного программирования к приближенной минимизации некоторой нелинейной функции. Поэтому они допускают двойственную интерпретацию: одновременно как точную, так и итерационную (аналогично методу сопряженных градиентов для линейных уравнений с матрицей $A = A^{\mathrm{T}} > 0$ [24]). Отметим также, что возможны и другие конструкции близких в идейном плане алгоритмов, например, методы параллельных и последовательных коррекций на подпространствах и иерархических базисов (см., например, [11]).

3.6. Про вычислительную сложность, оптимальность и сходимость. Пусть n_m — число неизвестных игры на m-м уровне. Обозначим $C_n = \min\limits_{m=1,\ldots,l} \dfrac{n_m}{n_{m-1}}$. Предположим, что одна базовая итерация, а также выполнение R- и P-процедур требуют на m-м уровне для разреженных матриц $O(n_m)$ операций, для плотных матриц — $O(n_m^2)$ операций. Обозначим через N_{ML} вычислительную сложность одного цикла многоуровнего метода. Несложно проверить, что в разреженном случае справедливо следующее приближенное равенство с некоторой постоянной c: $N_{\mathrm{ML}} \approx c\, n_l \Big(1 + \kappa\Big(1 + \kappa\big(1 + \kappa(\ldots)\big)\Big)\Big)$, всего l вложений, где $\kappa = \gamma C_n^{-1}$.

Пусть сходимость ML-метода не зависит от числа уровней. Тогда из оптимальности вычислительной сложности одной итерации (одного цикла) следует оптимальная сложность для решения исходной игры с любой наперед заданной точностью.

Очень интересным является вопрос об оптимальном выборе уровней. Всегда имеется возможность зафиксировать последовательность уровней заранее, например, по правилу двойного сокращения неизвестных: $n_{m+1} = 2\, n_m$. С другой стороны, является заманчивым автоматическое разбиение на уровни на основе приближенного равенства старших компонент вектора $\boldsymbol{v}$ в R-процедуре. Представляется правдоподобной оптимальность комбинированного алгоритма разбиения, когда фиксируется множитель сокращения неизвестных $\sigma > 1$, но выполнение жесткого равенства необязательно: $n_{m+1} \approx \sigma\, n_m$.

При анализе методов, имеющих рекурсивную структуру, принципиально важным является устано-

вление сходимости двухуровневого метода. Как правило, при этом накладывается ограничение снизу на число базовых итераций $\overline{\nu} > 0$ и соответствующее утверждение предполагает ограниченность нормы оператора сокращения ошибки двухуровневого метода $M_2(\nu)$ вида $\|M_2(\nu)\| \leqslant C_A\eta(\nu)$ при всех $\nu \geqslant \overline{\nu}$. Здесь $M_2(\nu)$ определен как $\boldsymbol{x}(k+1) - \boldsymbol{x}_* = M_2(\nu)\big(\boldsymbol{x}(k) - \boldsymbol{x}_*\big)$, C_A — постоянная, а функция $\eta(\nu) \to 0$ при $\nu \to \infty$, причем C_A и $\eta(\nu)$ не зависят от s — размерности нулевого уровня. Конечно, утверждения такого рода представляются близкими к идеальным; более правдоподобны оценки для оператора сокращения ошибки ML-метода с некоторой постоянной $c > 0$ вида $\|M_l(\nu)\| \leqslant 1 - (cl)^{-1}$, т.е. зависящие от количества уровней в методе.

Типичными видами функции $\eta(\nu)$ при решении многосеточными методами эллиптических задач являются $\eta(\nu) = O\big(\nu^{-1}\big)$ или $\eta(\nu) = O\big(\nu^{-1/2}\big)$. Отметим, что предполагаемая (но пока не доказанная!) [16] сходимость классического FP-метода соответствует $\eta(\nu)$ второго вида, а численные эксперименты для различных его модификаций (в том числе ориентированных на симметричные игры) демонстрируют сходимость с оценкой первого вида [19]. Если двухуровневый метод использует в качестве базового FP-метод, то его сходимость практически гарантирована.

Тем не менее, следует зафиксировать, что технический аппарат для теории сходимости многоуровневого метода фактически отсутствует. Приведенные замечания, в первую очередь, базировались на аналогии с многосеточным методом [11] решения уравнений. В случае матричных игр имеет место важный аспект — возможная неединственность решения $\boldsymbol{x}_*$, которая, безусловно, осложнит анализ сходимости. Кроме того, оператор сокращения ошибки ML-метода нетривиальным образом зависит от предыдущего приближения $\boldsymbol{x}(k)$, что также не упрощает ситуацию. При этом, конечно, произвольный итерационный метод решения симметричной матричной игры нужно понимать как процесс проектирования некоторого начального приближения $\boldsymbol{x}(0) \in \boldsymbol{X}$ на ядро одного из главных миноров матрицы A с сохранением для итераций принадлежности к симплексу $\boldsymbol{X}$, т.е. нормы $\big\|\boldsymbol{x}(0)\big\|_1 = \sum\limits_{i=1}^{n}\big|x_i(0)\big| = 1 = \big\|\boldsymbol{x}(k)\big\|_1$ и неотрицательности компонент $x_i(k) \geqslant 0$, $1 \leqslant i \leqslant n$. Матрицы, сохраняющие $\|\cdot\|_1$-норму, к сожалению, не являются ортогональными, поэтому подходящий выбор операторной нормы и построение соответствующих матричных разложений для исследования сходимости также могут представлять известную проблему.

Заключительные замечания. Следующим естественным шагом в исследовании многоуровневого метода является проведение численных экспериментов для получения информации внутреннего и внешнего рода. К первой относится: вычислительная устойчивость различных элементов (метод излагался в предположении отсутствия ошибок округлений), оптимальное разбиение на уровни, анализ соотношения практически приемлемых вычислительных затрат DS-процедуры и остальных действий, экспериментальное сравнение V- и W-циклов и их ближайших модификаций при одинаковых условиях, экспериментальное определение количеств итераций базового метода ν и δ и т.п. Внешняя информация носит другой характер — сравнение с другими методами (в первую очередь, решением на основе сведения к LP-проблемам) для выяснения конкурентоспособности на больших задачах, т.е. для определения нижней границы размерности "большой" задачи. Хотя на первый взгляд существование n, при котором ML-метод будет иметь преимущество, неочевидно, следует заметить, что лучшие из имеющихся в настоящее время для решения LP-проблем алгоритмы имеют полиномиальную сложность (т.е. являются прямыми), а у итерационных методов при сопоставлении с прямыми всегда выявляется своя область применения вследствие учета специфики постановки (например, разреженности матрицы).

В случае если эти надежды оправдаются, возникают естественные постановки формального обоснования сходимости и оптимизации различных вариантов метода. Какого вида теоремы удастся доказать — вопрос открытый. Конечно, есть ориентир — теория многосеточного метода решения эллиптических и близких к ним уравнений. Однако она не допускает формального переноса в силу значительного увеличения сложности постановки задачи.

Имеет ли смысл адаптировать многоуровневый метод к большим "соседним" проблемам, допускающим естественную игровую формулировку: задачам линейного программирования и линейного приближения в метриках l_1 и l_∞ [25]? В настоящий момент ответ неясен, но, если потребуется, для этого в статье имеются хорошие заготовки. В любом случае поле для будущей деятельности в направлении развития многоуровнего метода представляется достаточно обширным.

Автор выражает искреннюю благодарность Е. А. Ивину и С. Султанходжаеву, чье активное влияние значительно ускорило появление на свет результатов статьи.

СПИСОК ЛИТЕРАТУРЫ

1. *Karmarkar N.* A new polonomial-time algorithm for linear programming // Combinatorica. 1984. **4**, N 4. 373–395.

2. *Wright M.H.* The interior-point revolution in optimization: history, recent developments, and lasting consequences // Bull. Amer. Math. Soc. 2004. **42**, N 1. 39–56.
3. *Nemirovski A.S., Todd M.J.* Interior-point methods for optimization // Acta Numerica. 2008. **17**. 191–234.
4. *Васильев Ф.П., Иваницкий А.Ю.* Линейное программирование. М.: Факториал, 1998.
5. *Хачиян Л.Г.* Сложность задач линейного программирования. М.: Знание, 1987.
6. *Colombo M.* Advances in interior point methods for large-scale linear programming. PhD Thesis in Optimization. School of Mathematics. University of Edinburgh. Edinburgh, 2007.
7. *Бабенко К.И.* Основы численного анализа. М.: Наука, 1986.
8. *Федоренко Р.П.* Релаксационный метод решения разностных эллиптических уравнений // Журн. вычисл. матем. и матем. физ. 1961. **1**, № 5. 922–927.
9. *Федоренко Р.П.* О скорости сходимости одного итерационного процесса // Журн. вычисл. матем. и матем. физ. 1964. **4**, № 3. 227–235.
10. *Бахвалов Н.С.* О сходимости одного релаксационного метода для эллиптического оператора с естественными ограничениями // Журн. вычисл. матем. и матем. физ. 1966. **6**, № 5. 101–135.
11. *Ольшанский М.А.* Лекции и упражнения по многосеточным методам. М.: Физматлит, 2005.
12. *Нейман Дж., Моргенштерн О.* Теория игр и экономическое поведение. М.: Наука, 1970.
13. *Brown G.W.* Iterative solution of games by fictitious play // Activity analysis of production and allocation. / Edited by T.C. Koopmans. New York: Wiley, 1951. 374–376.
14. *Robinson J.* An iterative method of solving a game // Ann. Math. 1951. **54**. 296–301.
15. *Shapiro H.N.* Note on a computation method in the theory of games // Commun. Pure Appl. Math. 1958. **11**. 587–593.
16. *Szép J., Forgó F.* Introduction to the theory of games. Dordrecht: Reidel, 1985.
17. *Беленький В.З., Волконский В.А., Иванков С.А., Поманский А.Б., Шапиро А.Д.* Итеративные методы в теории игр и программировании. М.: Наука, 1974.
18. *Gaas S.I, Zafra P.M., Qui Z.* Modified fictitious play for solving matrix games and linear programming problems // Comp. Oper. Res. 1995. **22**. 893–903.
19. *Washburn A.* A new kind of fictitious play // Naval Research Logistics. 2001. **48**. 269–280.
20. *Садовский А.Л.* Монотонный итеративный алгоритм решения матричных игр // Доклады АН СССР. 1978. **238**, № 3. 538–540.
21. *Gale D., Kuhn H.W., Tucker A.W.* On symmetric games // Contributions to the theory of games / Edited by H.W. Kuhn and A.W. Tucker. Annals of Mathematics Study N 24. Princeton: Princeton University Press. 1950. **1**. 81–87.
22. Матричные игры / Под ред. Н.Н. Воробьева. М.: ГИФМЛ, 1961.
23. *Воеводин В.В., Кузнецов Ю.А.* Матрицы и вычисления. М.: Наука, 1984.
24. *Бахвалов Н.С.* Численные методы. М.: Наука, 1973.
25. *Vaserstein L.N.* Matrix games, linear programming, and linear approximation // arXiv.org, math.cs/0609056v1[cs.GT], 27 Jan 2006.

Поступила в редакцию
20.07.2009