ЖУРНАЛ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ФИЗИКИ, 2021, том 61, № 7, с. 1059–1069

ОПТИМАЛЬНОЕ УПРАВЛЕНИЕ

УДК 517.977.5

ЧИСЛЕННОЕ ИССЛЕДОВАНИЕ ЗАДАЧ ОПТИМИЗАЦИИ БОЛЬШИХ РАЗМЕРНОСТЕЙ С ИСПОЛЬЗОВАНИЕМ МОДИФИКАЦИИ МЕТОДА Б.Т. ПОЛЯКА[1)]

© 2021 г. А. Н. Андрианов[2,*], А. С. Аникин[1,**], А. Ю. Горнов[1,*]**

[1] *664033 Иркутск, ул. Лермонтова, 134, Институт динамики систем и теории управления СО РАН, Россия*

[2] *125047 Москва, Миусская пл., 4, Институт прикладной математики им. М.В. Келдыша РАН, Россия*

**e-mail: gornov@icc.ru*

***e-mail: anikin@icc.ru*

****e-mail: and@a5.kiam.ru*

Поступила в редакцию 26.11.2020 г.
Переработанный вариант 26.11.2020 г.
Принята к публикации 11.03.2021 г.

Предложена модификация специального метода выпуклой оптимизации Б.Т. Поляка. Свойства соответствующего алгоритма исследованы путем вычислительных экспериментов для задач выпуклой сепарабельной и несепарабельной оптимизации, невыпуклых задач оптимизации потенциалов атомно-молекулярных кластеров и модельной задачи оптимального управления. Реализованы последовательные и параллельные версии алгоритма, позволившие решить задачи с размерностями до ста миллиардов переменных. Библ. 10. Фиг. 4. Табл. 6.

Ключевые слова: выпуклая оптимизация, метод Б.Т. Поляка, задачи больших размерностей.

DOI: 10.31857/S0044466921070036

1. ВВЕДЕНИЕ

В ранних работах Бориса Теодоровича Поляка (см., например, [1]) был предложен (суб)градиентный метод поиска экстремума в выпуклых задачах математического программирования. Большого внимания специалистов метод не привлек, что связано, очевидно, с некоторыми его “неудобными” особенностями. Во-первых, сходимость метода гарантировалась только при минимизации выпуклых функций, нарушение выпуклости могло приводить к нарушению релаксации на итерациях. Во-вторых, для работы метода требовалось априорное знание точного значения функции в экстремальных точках. Сформулированные требования и наличие развиваемых в те годы методов сопряженных градиентов Флетчера–Ривса и Полака–Поляка–Рибьера (см. [2]) привели к незаслуженному, на наш взгляд, снижению активности по исследованию и использованию предложенного алгоритма.

Новый всплеск интереса к методу появился лишь в последние годы, что связано, очевидно, с рассмотрением больших и сверхбольших задач оптимизации. В соответствии с современной классификацией (см., например, [3], [4]) задачи выпуклой оптимизации можно разделить по размерности на: “Small” – до 100 переменных, “Medium” – от 10^3 до 10^4, “Large” – от 10^5 до 10^7 и “Huge” – более 10^8 переменных. В соответствии с доминирующей в настоящее время точкой зрения решать задачи выпуклой оптимизации размерности 10^8 и более при отсутствии разреженности (при “плотных” векторах) с использованием современной вычислительной техники малореально.

Основные недостатки метода Поляка при рассмотрении задач, в которых указанные требования не выполняются, в том числе, невыпуклых, возможно преодолевать путем введения несложных модификаций. Например, шаг метода по направлению градиента в общем случае может приводить в точку, в которой значение функции больше, чем значение в текущей точке. В этой ситуации можно, не слишком утруждаясь, организовать кратное уменьшение шага, например, в

[1)]Работа выполнена при финансовой поддержке РФФИ (код проекта № 18-07-00587).

два раза, при этом свойство релаксации будет гарантировано, если, конечно, еще не найдено решение из множества оптимальных. Однако сильные стороны метода (“дешевая” итерация, малые требования к оперативной памяти) могут оказаться очень удобными при рассмотрении задач больших и сверхбольших размерностей.

В работе рассматриваются модификации метода Поляка, ориентированные на решение больших и свербольших задач оптимизации. На простейших примерах из семейств выпуклых функций сепарабельного и квазисепарабельного типа исследуются возможности численного решения гладких задач безусловной минимизации растущих размерностей. На примерах мультиэкстремальных задач оптимизации потенциалов атомно-молекулярных кластеров Морса (см. [5]) и Китинга (см. [6]) исследуются глобализующие свойства модификации метода Поляка. Для нелинейных задач оптимального управления оцениваются возможности решения аппроксимативных задач на сетках с уменьшающимся постоянным шагом. Рассматриваемые модификации метода реализованы на языке C (компиляторы BCC, GCC, Clang и ICC) под управлением операционных систем Linux, Mac OS X и Windows с использованием технологий распараллеливания OpenMP и MPI. Приводимые в работе вычислительные эксперименты выполнялись как на обычных персональных компьютерах, так и на кластерных системах МВС-100К Межведомственного Суперкомпьютерного Центра (МСЦ) РАН и Института прикладной математики им. М.В. Келдыша (ИПМ) РАН.

2. ОБЩАЯ СТРУКТУРА МЕТОДОВ ГРАДИЕНТНОГО ТИПА

Рассматривается стандартная задача конечномерной оптимизации:

$$f(x) \to \min, \quad f(x) \in \mathbb{R}, \quad x \in \mathbb{R}^n. \tag{1}$$

Итерация классического алгоритма градиентного типа записывается в следующем традиционном виде:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2}$$

где направление спуска d_k выбирается как антиградиент минимизируемой функции $f(x)$ в точке x_k:

$$d_k = -\nabla f(x_k). \tag{3}$$

Размер шага α_k может выбираться разными способами, опирающимися на учет свойств целевой функции. Например, для дифференцируемых функций с известной константой Липшица оптимальным является выбор шага $\alpha_k = 1/L$. Однако такой способ не всегда применим для задач рассматриваемого класса в силу отсутствия информации о значении константы Липшица. Величина шага в направлении спуска может в общем случае вычисляться по формуле

$$\alpha_k = \arg\min_{\alpha} f(x_k + \alpha d_k). \tag{4}$$

Данный вариант обеспечивает наилучшую функцию релаксации на одну итерацию, но требует значительных вычислительных затрат. Для сверхбольших задач оптимизации предлагается использовать вычислительные технологии, основанные на применении методов, не требующих постоянных вычислений оптимизируемой функции и позволяющих исследовать прикладные задачи оптимизации больших и сверхбольших размерностей.

3. МЕТОД Б.Т. ПОЛЯКА И ЕГО МОДИФИКАЦИЯ

Метод был предложен Б.Т. Поляком в [1] (и, по нашим сведениям, дальнейшего развития не получил):

$$x_{i+1} = x_i - \frac{f(x_i) - f^*}{\|\nabla f(x_i)\|^2} \nabla f(x_i), \tag{5}$$

где f^* – известное минимальное значение функции.

Для одномерного случая метод совпадает с методом Ньютона для решения уравнения $f(x) = f^*$.

Поскольку в большинстве реальных постановок задач оптимизации f^* заранее не известно, предлагается ввести некоторую величину $\delta_f > 0$, определяющую прогнозируемое значение оптимизируемой функции на каждом шаге:

$$f^* = f(x_i) - \delta_f,$$
$$x_{i+1} = x_i - \frac{\delta_f}{\|\nabla f(x_i)\|^2}\nabla f(x_i). \tag{6}$$

Сходимость метода следует из теоремы, доказанной в [1]:

Теорема 1. *Пусть $f(x)$ является выпуклой, непрерывной, Q выпукло и замкнуто, существует точка минимума $x^* \in Q$, $f(x^*) = f^*$ и $\|\nabla f(x)\| \le c$ на $S = \left\{x \in Q, \left\|x - x^0\right\| \le \left\|x^* - x^0\right\|\right\}$. Тогда в методе* (5) *$f(x^n) \to f^*$, а x^n слабо сходится к некоторой точке минимума.*

Здесь функция $f(x)$ определена на некотором множестве Q. В случае неактивных границ и использования множеств простой структуры (например, параллелепипедного типа), приведенные результаты применимы к рассматриваемой постановке (1).

Алгоритм предлагаемой модификации метода Б.Т. Поляка строится следующим образом.

В качестве входных данных используются: начальная точка x_0, точность по норме градиента $\varepsilon_\nabla > 0$, точность по функции $\varepsilon_f > 0$, прогнозная оценка $\delta_f > 0$, коэффициент корректировки шага $0 < k < 1$, минимально допустимый шаг $\varepsilon_\lambda > 0$, $i = 0$ – номер итерации.

Algorithm 1. Основной цикл алгоритма

1: Вычисляется $f(x_i)$, $\nabla f(x_i)$, $\|\nabla f(x_i)\|$
2: **if** $\|\nabla f(x_i)\| < \varepsilon_\nabla$ или $(f(x_{i-1}) - f(x_i)) < \varepsilon_f$ **then**
3: переходим на шаг 13
4: **end if**
5: Находится шаг $\lambda = \dfrac{\delta_f}{\|\nabla f(x_n)\|^2}$
6: Вычисляется $x_{i+1} = x_i - \lambda \cdot \nabla f(x_i)$
7: **if** $f(x_{i+1}) < f(x_i)$, **then**
8: полагается $i = i + 1$ и выполняется переход на шаг 1
9: **end if**
10: **if** $\lambda > \varepsilon_\lambda$ **then**
11: $\lambda = \lambda \cdot k$ и выполняется переход к шагу 6
12: **end if**
13: Возвращается x_i – как найденная точка минимума

Утверждение о сходимости предлагаемого метода для выпуклых задач с известным f^* и половинным делением шага ($k = 0.5$) является тривиальным следствием теоремы 1, поскольку последовательность модифицированного метода обязательно содержит в себе последовательность исходного, сходимость которого строго доказана.

Можно выделить следующие особенности предложенной модификации.

1. При соответствующей настройке алгоритмических параметров (δ_f) предложенный метод может делать “большие” шаги, что добавляет ему глобализующие свойства.

2. Метод имеет высокий потенциал параллелизма.

3. Алгоритм имеет простую конструкцию, что имеет большое значение при реализации метода на ряде высокопроизводительных архитектур, например, GPU.

Таблица 1. Объем памяти (ОЗУ), требуемой для хранения n-мерного вектора вещественных чисел (float – одинарная точность, double – двойная)

n	Float		Double	
10^2	0.39	Kb	0.78	Kb
10^3	3.91	Kb	7.81	Kb
10^4	39.06	Kb	78.13	Kb
10^5	390.63	Kb	781.25	Kb
10^6	3.81	Mb	7.63	Mb
10^7	38.15	Mb	76.29	Mb
10^8	381.47	Mb	762.94	Mb
10^9	3.73	Gb	7.45	Gb
10^{10}	37.25	Gb	74.51	Gb
10^{11}	372.53	Gb	745.06	Gb
10^{12}	3.63	Tb	7.28	Tb

Рассмотрим далее примеры использования предложенной модификации метода при решении экстремальных задач различных классов.

4. ВЫПУКЛАЯ ОПТИМИЗАЦИЯ. СЕПАРАБЕЛЬНАЯ ТЕСТОВАЯ ЗАДАЧА

Решение задач безусловной минимизации класса “Huge-Scale” естественным образом сопряжено с рядом сложностей, одна из которых – собственно, размерность. Увеличение числа переменных очевидным образом влечет за собой повышение требований к объему памяти доступной вычислительной системы. При рассмотрении задач с предельными размерностями возникают ситуации, когда требуемый объем памяти слишком велик для одного вычислительного узла, что влечет за собой необходимость использования распределенных суперкомпьютерных мощностей. Объем памяти, требуемый для задач различной размерности, приведен в табл. 1.

Можно увидеть, что с точки зрения потребления памяти задачи с размерностями до 10^9 вполне возможно решать на современных вычислительных системах с общей памятью, задачи с большими размерностями требуют использования памяти суперкомпьютеров.

Рассмотрим сепарабельную функцию, имеющую следующий вид:

$$f(x) = \sum_{i=1}^{n} (x_i^2 + x_i^6). \tag{7}$$

Очевидно, что данная функция имеет минимум в точке, где все компоненты вектора x равны 0 и значение функции в этой точке также равно 0.

Расчеты проводились на суперкомпьютерах ИПМ РАН и МСЦ РАН, имеющих ограничение в 1 ГБ ОЗУ на процессорное ядро. Данное ограничение необходимо учитывать при выборе числа требуемых процессоров, чтобы не допустить “перерасхода” памяти на вычислительных узлах. Результаты расчетов для задач с различной размерностью приведены в табл. 2 и на фиг. 1.

Максимальная размерность тестовой сепарабельной задачи, которую удалось решить – 10^{11} переменных. На фиг. 1 хорошо видна линейная зависимость времени расчета от числа переменных, что говорит о хорошей масштабируемости предложенной модификации алгоритма.

5. ВЫПУКЛАЯ ОПТИМИЗАЦИЯ. НЕСЕПАРАБЕЛЬНАЯ ТЕСТОВАЯ ЗАДАЧА

Рассмотрим тестовую несепарабельную функцию вида

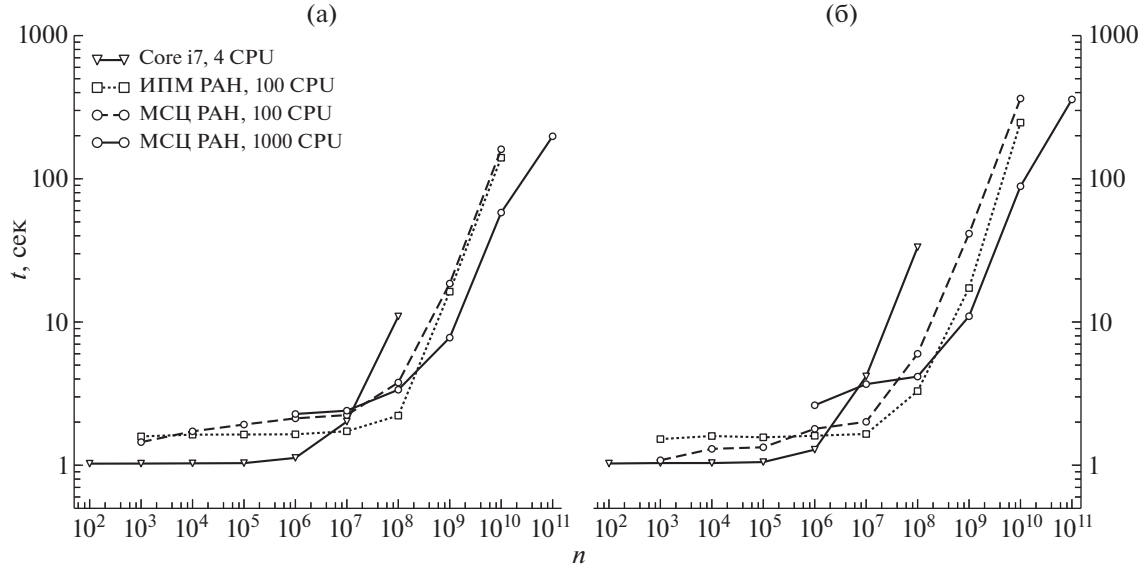
$$f(x) = \sum_{i=1}^{n} x_i^2 + \sum_{i=2}^{n} (x_i - x_{i-1})^2. \tag{8}$$

Таблица 2. Время [с], затраченное на работу алгоритма в зависимости от размерности тестовой сепарабельной функции

n	Core i7, 4 CPU	ИПМ, 100 CPU	МСЦ, 100 CPU	МСЦ, 1000 CPU
10^2	1.02559	–	–	–
10^3	1.02685	1.58932	1.44929	–
10^4	1.03031	1.63431	1.72392	–
10^5	1.03406	1.63837	1.92358	–
10^6	1.12513	1.64386	2.12392	2.27892
10^7	2.01723	1.72646	2.23966	2.40381
10^8	10.93844	2.22012	3.77897	3.36539
10^9	–	16.29881	18.55662	7.78179
10^{10}	–	140.30873	160.74543	58.09801
10^{11}	–	–	–	198.05384

При реализации варианта алгоритма для этого класса задач потребовалось применение технологии обмена сообщениями MPI для кластерных вычислительных систем. Произведенные оценки показали, что обмен данными между вычислительными узлами при нахождении градиента функции составляют незначительную величину от общего времени расчетов. Основным лимитирующим фактором для алгоритма все так же является объем доступной памяти.

Результаты расчетов для задач с различной размерностью приведены в табл. 3 и на фиг. 1. Произведенные вычислительные эксперименты показали, что для несепарабельных задач также имеет место линейная зависимость временных затрат от размерности. В частности, решение “максимальной” задачи с 10^{11} переменными с использованием 1000 процессоров потребовало всего 6 мин.



Фиг. 1. Время, затраченное на работу алгоритма в зависимости от размерности задачи: (а) – сепарабельная функция, (б) – несепарабельная функция.

Таблица 3. Время [с], затраченное на работу алгоритма в зависимости от размерности тестовой несепарабельной функции

n	Core i7, 4 CPU	ИПМ, 100 CPU	МСЦ, 100 CPU	МСЦ, 1000 CPU
10^2	1.02845	–	–	–
10^3	1.03536	1.52187	1.08275	–
10^4	1.03566	1.59923	1.30111	–
10^5	1.05166	1.56577	1.33428	–
10^6	1.28414	1.60642	1.79563	2.62236
10^7	4.17233	1.65246	2.00984	3.68216
10^8	33.29102	3.29295	5.99972	4.15825
10^9	–	17.26339	41.40902	10.97092
10^{10}	–	246.75377	363.31317	88.66335
10^{11}	–	–	–	358.11812

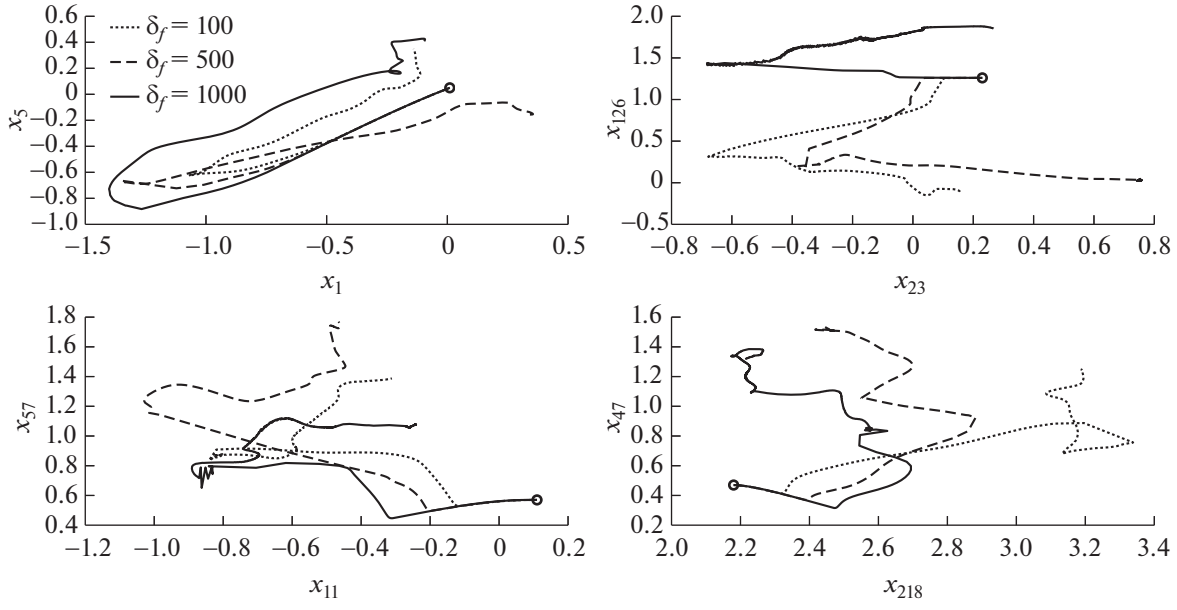
6. НЕВЫПУКЛАЯ ОПТИМИЗАЦИЯ. ПОИСК МОЛЕКУЛЯРНЫХ КЛАСТЕРОВ С МИНИМАЛЬНОЙ ЭНЕРГИЕЙ

Рассматривается потенциальная функция Морса (см. [5])

$$f(x) = V_M = \sum_{i=1}^{n}\sum_{j>i}^{n}[(e^{\rho_0(1-r_{ij})} - 1)^2 - 1]. \tag{9}$$

Задача минимизации данного потенциала является популярной “мультиэкстремальной” задачей глобальной оптимизации с астрономически растущим от размерности числом локальных экстремумов. Данная функция относится к классу несепарабельных, и оптимальное значение f^* неизвестно.

Традиционно для решения задач такого типа применяются специализированные методы невыпуклой оптимизации. Предложенная модификация алгоритма, дополненная механизмом случайного мультстарта, позволила успешно найти глобальные решения для кластеров, содер-



Фиг. 2. Проекции траекторий спусков для метода Поляка (модиф.) при различных значениях δ_f; минимизация потенциала Морса, 80 атомов (240 переменных).

Таблица 4. Минимизация потенциала Китинга, 102900 переменных (343000 атомов)

Метод	Время, с	$f_{\min}$	$\|\nabla f_{\min}\|$
Сопряженных градиентов	1087.242	4.421883e+01	3.401397e–05
Коши	2819.445	4.421883e+01	1.734354e–05
Поляка (модиф.)	19615.083	4.421883e+01	2.191246e–05

жащих до 80 атомов. При этом проявились нестандартные “глобализующие” свойства изначально локального метода, аппроксимирующего случайное распределение проб в окрестности рекордного значения. Примеры проекций траекторий спусков приведены на фиг. 2.

7. МИНИМИЗАЦИЯ ПОТЕНЦИАЛА КИТИНГА

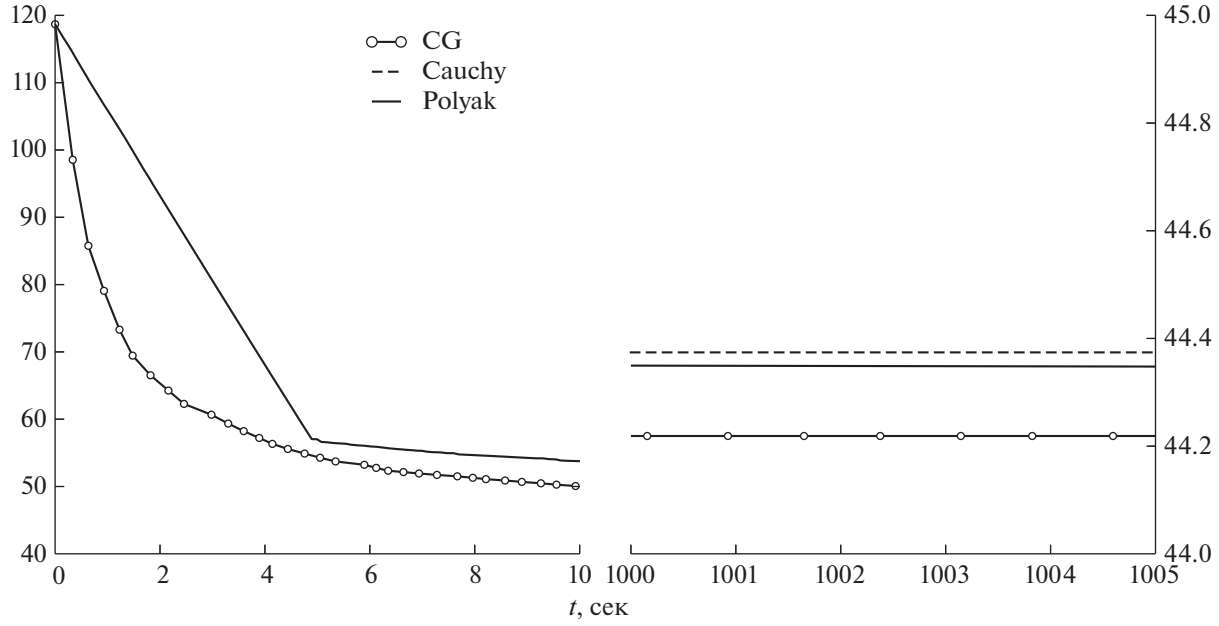
Рассмотрим задачу минимизации потенциала Китинга (см. [6]), моделирующего межатомное взаимодействие в кристаллических структурах “кремний–германий”:

$$E_k = \sum_{i=1}^{n}\left[\frac{3}{16}\sum_{j=1}^{4}\frac{\alpha_{ij}}{d_{ij}^2}\left\{\|r_i - r_j\|^2 - d_{ij}^2\right\}^2 + \frac{3}{8}\sum_{j=1}^{4}\sum_{k=j+1}^{4}\frac{\beta_{ijk}}{d_{ij}d_{ik}}\left\{\langle r_i - r_j, r_i - r_k\rangle + \frac{d_{ij}d_{ik}}{3}\right\}^2\right].$$

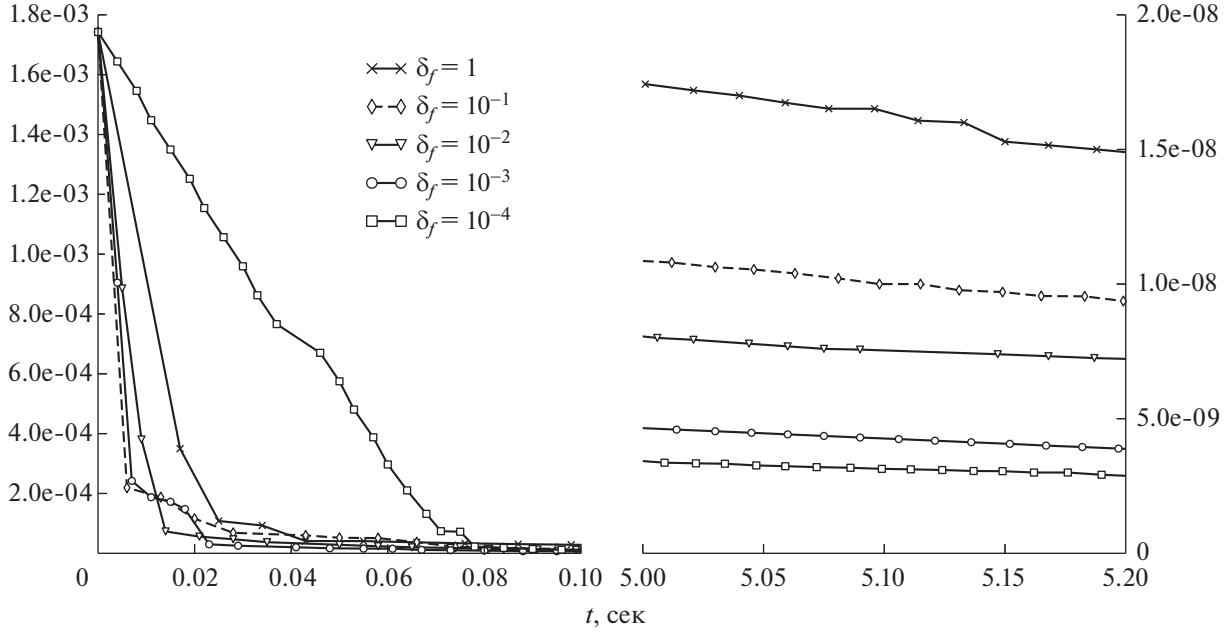
С применением предложенной модификации метода авторами производилась минимизация потенциала Китинга для кристалла с 1029000 переменными (343000 атомов в кристаллической решетке). Проводилось сравнение результатов расчетов с результатами вычислений, полученных с помощью метода сопряженных градиентов (вариант Полак–Поляк–Рибьер) и “метода Коши” (модификация метода наискорейшего спуска). Результаты расчетов приведены в табл. 4. На фиг. 3 показано поведение (сходимость) сравниваемых алгоритмов оптимизации.

Проводилось исследование поведения модифицированного алгоритма при различных значениях параметра δ_f на задаче минимизации потенциала Китинга для кристалла с 24000 переменными (8000 атомов), результаты показаны на фиг. 4.

Проведенные эксперименты показали, что и в этих задачах, в общем случае невыпуклых, с применением предложенной модификации алгоритма удалось получить верные решения за приемлемое время, хотя и немного проиграв в эффективности.



Фиг. 3. Минимизация потенциала Китинга, 1029000 переменных (343000 атомов).



Фиг. 4. Минимизация потенциала Китинга, 24000 переменных (8000 атомов).

8. ЗАДАЧА ОПТИМАЛЬНОГО УПРАВЛЕНИЯ

Традиционная постановка простейшей задачи оптимального управления (так называемая задача со свободным правым концом) заключается в поиске минимума терминального функционала

$$I(u) = \phi(x(t_1)), \tag{10}$$

определенного на траекториях управляемой системы дифференциальных уравнений (дифференциального включения)

$$\dot{x} = f(x(t), u(t), t), \tag{11}$$

заданной на интервале времени $t \in [t_0, t_1]$. Начальный фазовый вектор $x_0 = x(t_0)$ предполагается фиксированным, на управления наложены ограничения $\underline{u} \le u(t) \le \overline{u}$, где $\underline{u}$, $\overline{u}$ – фиксированные векторы. Рассматриваемая задача занимает центральное место в теории оптимального управления, к такой постановке сводится большое количество задач более сложных классов (с интегральными функционалами, с терминальными и фазовыми ограничениями, с нефиксированным временем и др.). Для задачи в такой постановке получен значительный объем глубоких теоретических результатов и предложено множество вычислительных методов. Однако практика численного решения задач такого класса, особенно прикладных, несмотря на усилия множества специалистов по теории управления, позволяет сделать вывод о доминировании методов, основанных на редукции к задачам конечномерной оптимизации. После введения сетки разбиения интервала времени, например, с равномерным шагом h, оказывается возможным сформулировать задачу конечномерной оптимизации, аппроксимирующую задачу в исходной постановке. В аппроксимативной задаче динамика системы описывается рекуррентными соотношениями $x(t+h) = f_h(x(t), u(t), t, h)$, все остальные условия сохраняются. Аппроксимативные задачи, очевидно, представляют собой специальный класс задач математического программирования с условиями типа равенств и неравенств. Если зафиксировать разностную схему, по которой выполняется дискретизация (Эйлера, Рунге–Кутты или другую), то можно воспользоваться предложенным в теории оптимального управления методом оценки градиента функционала, основанным на применении формализма сопряженных переменных. Получаемые по такой схеме алгоритмы оказываются весьма близкими к алгоритмам, которые генерируются с помощью методик быстрого автоматического дифференцирования (см., например, [7]). Вычисление градиента функционала по любой из указанных схем требует решения всего двух задач Коши. Таким образом, задача оптимального управления со свободным правым концом может быть рас-

Таблица 5. Время решения задачи оптимального управления программным комплексом OPTCON-A (N – число узлов сетки дискретизации)

N	$I(u^*)$	t, с	Задачи Коши	Норма градиента	Итерации
101	1.191521713372e+01	0	135	5.7e–3	15
201	1.190999877810e+01	0	317	2.4e–4	23
401	1.190841888965e+01	0	276	8.0e–5	24
801	1.190817286834e+01	0	182	2.2e–4	21
1601	1.190804296061e+01	0	241	1.2e–5	21
3201	1.190804923806e+01	0	167	6.8e–5	17
6401	1.190801439789e+01	1	109	1.8e–5	16
12801	1.190805357054e+01	3	206	3.5e–5	19
25601	1.190806771202e+01	8	237	4.7e–5	21
51201	1.190812331446e+01	13	203	4.4e–5	19
102401	1.190811071137e+01	22	167	3.8e–5	17
204801	1.190815091482e+01	44	166	4.0e–5	17
409601	1.190814768992e+01	88	166	2.9e–5	17
819201	1.190847626971e+01	144	131	2.9e–5	15

смотрена как задача безусловной минимизации с “дешевым”, алгоритмически оцениваемым градиентом, но вычисляемым с некоторой погрешностью, зависящей от размеров шага дискретизации h. Размер шага дискретизации непрерывной задачи задает число переменных в аппроксимативной задаче, равное, грубо говоря, $N = (t_1 - t_0)/h$.

Эффективность предложенной модификации метода Поляка попробуем оценить с помощью численных экспериментов для одной из самых популярных тестовых задач – задачи успокоения нелинейного маятника. Будем последовательно решать аппроксимативные задачи с растущим числом переменных N (и, соответственно, числом прямых ограничений на переменные), фиксируя процессорное время расчета, число итераций, решенных задач Коши и достигнутое значение функционала.

Динамика модели описывается системой (см., например, [8])

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = u - \sin x_1, \tag{12}$$

определенной на интервале времени $t \in [0, 5]$ с начальными условиями $x_1(0) = 5$, $x_2(0) = 0$. Требуется минимизировать функционал $I(u) = x_1^2(5) + x_2^2(5)$ при ограничениях на управления $|u(t)| \le 1$. В задаче имеет место два локальных экстремума, глобально оптимальное значение функционала $I(u^*) = 11.908013767$, полученное с применением метода интегрирования DOPRI8 (8-й порядок точности), достигается на релейном управлении:

$$u^*(t) = \begin{cases} +1, & t \in [0.0,\ 0.98244286] \cup [4.55036911,\ 5.0], \\ -1, & t \in [0.98244286,\ 4.55036911]. \end{cases}$$

Для решения задач Коши использовался метод интегрирования из семейства Рунге–Кутты 2-го порядка (так называемый метод “Эйлера с итерациями”). Расчет прерывался при выполнении условия $I(u^{k-1}) - I(u^k) < 10^{-6}$. Все задачи решались, начиная с одинаковых начальных приближений $u^0(t) \equiv -0.5$. В этой задаче, очевидно, известна гарантированная, но неоптимальная, нижняя оценка значения функционала, равная 0.

Эксперимент выполнялся с использованием стандартного программного комплекса (ПК) OPTCON-A (см. [9], [10]) в состав которого включена предложенная модификация алгоритма. Результаты эксперимента приведены в табл. 5.

Максимальной размерностью задач, решенных при использовании ПК OPTCON" A, оказалась размерность 819201, что связано, очевидно, с излишним объемом внутренней памяти комплекса, используемой для работы других алгоритмов. Процессорное время решения при размер-

Таблица 6. Время [с] решения задачи оптимального управления специализированной программой (N – число узлов сетки дискретизации)

N	AMD 6220 Linux	i7-2640M Linux			i7-2640M Mac OS X		i5 4260U Mac OS X	
	icc	icc	gcc	clang	gcc	clang	gcc	clang
	14.0.0	15.0.1	4.8.2	3.4.2	4.8.3	3.4.2	4.8.3	3.5SVN
101	0.006	0.001	0.006	0.009	0.001	0.001	0.002	0.002
201	0.014	0.003	0.012	0.005	0.003	0.003	0.004	0.004
401	0.018	0.007	0.014	0.014	0.006	0.006	0.009	0.008
801	0.033	0.010	0.023	0.028	0.011	0.010	0.015	0.015
1601	0.019	0.010	0.019	0.014	0.006	0.007	0.008	0.008
3201	0.085	0.031	0.065	0.070	0.044	0.034	0.045	0.046
6401	0.078	0.041	0.099	0.100	0.046	0.051	0.066	0.069
12801	0.289	0.159	0.271	0.311	0.156	0.154	0.220	0.255
25601	0.674	0.298	0.620	0.687	0.366	0.380	0.500	0.495
51201	1.030	0.521	1.060	1.234	0.627	0.613	0.870	0.942
102401	1.890	0.887	1.799	2.023	1.142	1.160	1.505	2.100
204801	3.546	1.765	4.415	4.015	2.352	2.319	2.866	3.044
409601	7.251	3.576	7.133	8.034	4.588	4.511	5.509	5.938
819201	11.662	5.714	11.608	12.883	7.447	7.315	8.752	8.682
1638401	18.197	8.929	18.203	20.443	11.588	11.381	13.547	13.313
3276801	35.353	17.975	35.947	40.622	22.972	25.862	27.110	26.504
6553601	50.279	26.991	52.610	59.321	33.680	33.090	39.626	39.432
13107201	103.237	–	–	–	–	–	–	–
26214401	205.603	–	–	–	–	–	–	–
52428801	346.577	–	–	–	–	–	–	–
104857601	692.295	–	–	–	–	–	–	–

ностях задач от 101 до 3201 переменной было менее 0.5 с и оценено в 0 с, поскольку используемый датчик времени позволяет учитывать его с точностью до 1. Начиная с размерности 401 переменных, во всех задачах получены значения функционала, с точностью до четырех знаков совпадающие с известным оптимальным значением. Быстрый рост процессорного времени с ростом размерности задач связан с неэффективностью работы процедур аппроксимации управления общего назначения, не позволяющих учитывать специфические особенности алгоритма.

Второй эксперимент проводился со специальной программной реализацией алгоритма, проверенной с использованием нескольких популярных компиляторов на нескольких различных компьютерах. В эксперименте участвовали следующие пары “процессор–компилятор”.

1. Intel Core i7-2640M; Linux; icc-15.0.1, gcc-4.8.2, gcc-4.9.1, clang-3.4.2, clang-3.5.0.

2. Intel Core i7-2640M; Mac OS X; gcc-4.8.3, clang-3.4.2.

3. Intel Core i5 4260U; Mac OS X; gcc-4.8.3, clang-600.0.56 (based on LLVM 3.5svn).

4. AMD Opteron 6220; Linux; icc-14.0.0.

Результаты выполненных расчетов приведены в табл. 6. Наилучшую производительность показала версия алгоритма, реализованная с использованием компилятора icc.

Проведенные эксперименты показали принципиальную возможность решения аппроксимативных задач оптимального управления с размерностями, превышающими 10^8 даже без применения параллельных вычислительных технологий.

9. ЗАКЛЮЧЕНИЕ

Рассмотренная в работе модификация метода Поляка позволила создать вычислительные технологии, обладающие способностью эффективно решать экстремальные задачи размерностей “Huge-Scale” из различных классов. На основе численных экспериментов выявлены ограничения по размерности решаемых “плотных” задач, которые оказались значительно большими, чем предполагалось теоретически (см. [3]). Метод Б.Т. Поляка 1969 г. с применением простейших модификаций может рассматриваться в качестве удобного и высокомасштабируемого инструмента для численного решения “больших” задач конечномерной и бесконечномерной оптимизации.

СПИСОК ЛИТЕРАТУРЫ

1. *Поляк Б.Т.* Минимизация негладких функционалов // Ж. вычисл. матем. и матем. физ. 1969. Т. 9. № 3. С. 509–521.
2. *Поляк Б.Т.* Введение в оптимизацию. М.: Наука, 1983. 384 с.
3. *Нестеров Ю.Е.* Введение в выпуклую оптимизацию. М.: МЦНМО, 2010. 280 с.
4. Yurii Nesterov Subgradient Methods for Huge-Scale Optimization Problems // Math. Program. 2014. V. 146. Iss. 1-2. P. 275–297.
5. *Marques J.M.C., Pais A.A.C.C., Abreu P.E.* On the use of big-bang method to generate low-energy structures of atomic clusters modeled with pair potentials of different ranges // J. Comput. Chemistry. 2012. V. 33. № 4. P. 442–452.
6. *Keating P.N.* Effect of Invariance Requirements on the Elastic Strain Energy of Crystals with Application to the Diamond Structure // Phys. Rev. 1966. V. 145. P. 637–645.
7. *Евтушенко Ю.Г.* Оптимизация и быстрое автоматическое дифференцирование. Препринт ВЦ им. А.А. Дородницына РАН, 2013. 144 с.
8. *Горнов А.Ю.* Вычислительные технологии решения задач оптимального управления. Новосибирск: Наука, 2009. 278 с.
9. *Gornov A.Yu., Zarodnyuk T.S., Anikin A.S.* The computational technique for nonlinear nonconvex optimal control problems based on modification gully method // DEStech Transact. Comput. Sci. Eng. 2018. P. 152–162.
10. *Gornov A.Yu., Tyatyushkin A.I., Finkelstein E.A.* Numerical methods for solving terminal optimal control problems // Comput. Math. and Math. Phys. 2016. V. 56. № 2. P. 221–234.