ЖУРНАЛ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ФИЗИКИ, 2019, том 59, № 5, с. 775–791

УДК 519.62

ВЫЧИСЛЕНИЕ ОПТИМАЛЬНЫХ ВОЗМУЩЕНИЙ ДЛЯ СИСТЕМ С ЗАПАЗДЫВАНИЕМ[1)]

© 2019 г. Ю. М. Нечепуренко[1], М. Ю. Христиченко[2,*]

([1] 119333 Москва, ул. Губкина, 8, ИВМ им. Г.И. Марчука РАН, Россия;
[2] 125047 Москва, Миусская пл., 4, ИПМ им. М.В. Келдыша РАН, Россия)
**e-mail: misha.hrist@gmail.com*

Поступила в редакцию 25.05.2018 г.
Переработанный вариант 28.11.2018 г.
Принята к публикации 11.01.2019 г.

Предлагаются и обосновываются новые быстрые алгоритмы вычисления максимальной амплификации нормы решения и оптимальных возмущений для систем с запаздыванием. Предложенные алгоритмы опробованы на системе четырех нелинейных дифференциальных уравнений с запаздывающим временем, представляющей собой модель экспериментальной инфекции, вызванной вирусами лимфоцитарного хориоменингита. Приводятся и обсуждаются результаты численных экспериментов. Библ. 12. Фиг. 8. Табл. 4.

Ключевые слова: оптимальные возмущения, дифференциальные уравнения с запаздыванием, максимальная амплификация, метод Ланцоша, последовательная максимизация.

DOI: 10.1134/S0044466919050120

1. ВВЕДЕНИЕ

Понятие оптимального возмущения стационарного состояния динамической системы без запаздывания, обеспечивающего максимально возможную амплификацию возмущения в заданной норме (увеличение нормы возмущения по сравнению с ее первоначальным значением), широко используют для описания механизма докритического ламинарно-турбулентного перехода в теории аэродинамической устойчивости, смотри, например, работы [1]–[3] и их библиографии. Для динамической системы с запаздыванием понятие оптимального возмущения стационарного состояния было впервые введено в работах [4]–[7]. На примере модели динамики лимфоцитарного хориоменингита у мышей, описанной в [8], было показано, что оптимальные возмущения позволяют, в частности, находить минимально достаточные терапии, позволяющие переводить организм из состояний с высокой вирусной нагрузкой в состояния с низкой вирусной нагрузкой. Для вычисления оптимальных возмущений был предложен и обоснован достаточно простой алгоритм, эффективный для систем с запаздыванием небольшой размерности и основанный на сведении исходной задачи к вычислению матричных произведений. При этом алгебраический размер матричных произведений был обратно пропорционален шагу сетки по времени, используемому для численного интегрирования исходной системы.

Для эффективного вычисления оптимальных возмущений в случае систем с запаздыванием большой размерности и/или требующих для численного интегрирования очень мелких шагов по времени необходимы алгоритмы, использующие в своей схеме только умножение матрицы на вектор. Такого типа алгоритмы были предложены недавно в работе [9] для систем обыкновенных дифференциальных уравнений без запаздывания. Данная работа посвящена обобщению этих алгоритмов на системы с запаздыванием.

Работа состоит из восьми разделов. В разд. 2 даются определения максимальной амплификации нормы решения и оптимальных возмущений, аналогичные данным в [7]. В разд. 3 изложена модификация простейшего алгоритма вычисления оптимального возмущения для введенного класса норм. В разд. 4 предложена дальнейшая модификация этого алгоритма с использованием

[1)]Обоснование алгоритмов выполнено при финансовой поддержке программы Президиума РАН № 01 “Фундаментальная математика и ее приложения” (грант PRAS-18-01); разработка и реализация алгоритмов выполнены при финансовой поддержке РНФ (код проекта 17-71-21049).

метода Ланцоша, позволяющая эффективно вычислять оптимальные возмущения для задач большой размерности. В разд. 5 описан еще более эффективный алгоритм поиска оптимального возмущения, основанный на последовательной максимизации. В разд. 6 оцениваются вычислительные затраты алгоритмов, описанных в предыдущих разделах. В разд. 7 приводятся результаты численных экспериментов с предложенными алгоритмами на примере модели динамики заболевания, вызванного вирусами лимфоцитарного хориоменингита. Итог работы подводится в разд. 8.

2. ОПТИМАЛЬНЫЕ ВОЗМУЩЕНИЯ

Рассмотрим систему обыкновенных дифференциальных уравнений вида

$$\frac{d}{dt}U(t) = F(U(t), U(t-\tau_1), \ldots, U(t-\tau_p)), \tag{2.1}$$

где $0 < \tau_1 < \ldots < \tau_p$, $U(t)$ является n-компонентным вектором переменных, а $F(\xi_0, \xi_1, \ldots, \xi_p)$ – некоторое достаточно гладкое отображение. Будем предполагать, что решение задачи Коши для системы (2.1) существует и единственно для любого кусочно-непрерывного начального значения $U(t)$, $-\tau_p \le t \le 0$.

Нас будет интересовать поведение системы (2.1) вблизи устойчивого стационарного состояния $U(t) = \bar{U}$. Записывая решение вблизи стационарного состояния в виде $U(t) = \bar{U} + \varepsilon U'(t) + O(\varepsilon^2)$, где ε – малый по абсолютной величине параметр, подставляя это решение в (2.1) и требуя, чтобы полученные равенства выполнялись при всех сколь угодно малых значениях ε, для функции $U'(t)$ получим систему линейных дифференциальных уравнений вида

$$\frac{d}{dt}U'(t) = L_0U'(t) + \sum_{j=1}^{p} L_jU'(t-\tau_j), \tag{2.2}$$

где

$$L_j = \frac{\partial F}{\partial \xi_j}(\bar{U}, \bar{U}, \ldots, \bar{U}), \quad j = 0,1, \ldots, p,$$

и являются постоянными квадратными матрицами порядка n.

Для вектора переменных системы (2.2) введем в рассмотрение следующее семейство локальных норм в момент времени t:

$$\|U'\|_{D,\rho,t} = \left(\int_{t-\tau_p}^{t} \left(\|DU'(\xi)\|_2^2 + \rho \left\| D\frac{dU'}{d\xi}(\xi) \right\|_2^2 \right) d\xi \right)^{1/2}, \tag{2.3}$$

где D – заданная положительно определенная диагональная матрица порядка n, $\|\cdot\|_2$ – вторая (евклидова) векторная норма, ρ – неотрицательный параметр. В случае $\rho = 0$ мы будем для краткости называть эту норму L_2-нормой, а в случае $\rho = 1$ назовем W_2^1-нормой.

Под оптимальным возмущением $U'_{\text{opt}}(t)$ будем понимать решение системы (2.2), обеспечивающее максимальную амплификацию локальной нормы (2.3) возмущения по сравнению с ее первоначальным значением, то есть такое решение $U'(t)$, на котором достигается максимум величины

$$\max_{t\ge 0} \frac{\|U'\|_{D,\rho,t}}{\|U'\|_{D,\rho,0}}.$$

Оптимальное возмущение по определению является некоторым решением линейной системы (2.2) и, следовательно, полностью определяется своим значением в интервале $-\tau_p \le t \le 0$. Поэтому при построении оптимального возмущения наряду с выбором нормы, в которой проводится оптимизация, принципиальным является вопрос о том, из какого пространства функций, заданных в интервале $-\tau_p \le t \le 0$, мы берем начальные значения. Это пространство функций

$[-\tau_p, 0] \to \mathbb{R}^n$ мы далее будем обозначать через $\mathcal{Q}$. Для гарантированного существования максимума необходимо, чтобы $\mathcal{Q}$ было полным пространством в норме $\|\cdot\|_{D,\rho,0}$. Для этого достаточно выбирать в качестве $\mathcal{Q}$ линейную оболочку какого-либо конечного набора кусочно-непрерывных базисных функций.

Найденное оптимальное возмущение используют для возмущения стационарного состояния исходной нелинейной системы (2.1), выбирая в качестве начального значения

$$U(t) = \bar{U} + \varepsilon \tilde{U}'_{\text{opt}}(t) \tag{2.4}$$

при $-\tau_p \le t \le 0$, где $\tilde{U}'_{\text{opt}}(t)$ означает нормированное в L_2 норме оптимальное возмущение, а ε – вещественный параметр. Варьируя абсолютную величину этого параметра, можно увеличивать или уменьшать начальное возмущение. В зависимости же от его знака, заданная компонента решения при $t > 0$ либо начинает нарастать, либо убывать.

3. ПРОСТЕЙШИЙ АЛГОРИТМ ВЫЧИСЛЕНИЯ ОПТИМАЛЬНЫХ ВОЗМУЩЕНИЙ

Искать оптимальное возмущение удобнее в два этапа. Сначала вычисляем максимальную амплификацию

$$\Gamma(t) = \max_{U'} \frac{\|U'\|_{D,\rho,t}}{\|U'\|_{D,\rho,0}} \tag{3.1}$$

локальной нормы решения системы (2.2), где максимум берется по всем решениям, начальные значения которых ненулевые и принадлежат $\mathcal{Q}$, и находим $t = t_{\text{opt}}$, при котором функция $\Gamma(t)$ достигает максимального значения. Если таких t несколько, то для определенности берем из них минимальное. Таким образом,

$$t_{\text{opt}} = \min \arg\max_{t \ge 0} \Gamma(t).$$

Затем находим

$$U'_{\text{opt}} \in \arg\max_{U'} \frac{\|U'\|_{D,\rho,t_{\text{opt}}}}{\|U'\|_{D,\rho,0}}. \tag{3.2}$$

Если D, ρ и $\mathcal{Q}$ фиксированы, то любое найденное оптимальное возмущение обеспечивает одну и ту же максимальную амплификацию локальной нормы решения. Обычно максимальная амплификация имеет только одну точку глобального максимума, а решение оптимизационной задачи (3.2) однозначно с точностью до ненулевой мультипликативной константы.

Оптимальные возмущения можно вычислять на основе любой разностной схемы, подходящей для решения задач Коши для систем обыкновенных дифференциальных уравнений с запаздывающим аргументом. Следуя работам [4]–[6], мы будем использовать неявную схему второго порядка BDF2 [10] на равномерной сетке

$$\{t_k = \delta k : k = -m_p + 1, -m_p + 2, \ldots\},$$

построенной в полуинтервале $(-\tau_p, \infty)$ с шагом $\delta > 0$. При этом величины $m_j = [\tau_j/\delta]$, где $[.]$ означает целую часть числа, будут дискретными аналогами задержек τ_j. После дискретизации система (2.2) примет вид

$$\frac{1.5U_k - 2U_{k-1} + 0.5U_{k-2}}{\delta} = L_0 U_k + \sum_{j=1}^{p} L_{\tau_j} U_{k-m_j}, \tag{3.3}$$

где U_k – сеточная функция, аппроксимирующая $U'(t_k)$. В качестве начальных данных для решения разностной задачи Коши нужно задать значения $U_{-m_p+1}, \ldots, U_0$.

Предполагая шаг δ достаточно малым, запишем уравнение (3.3) в виде

$$U_k = C_1 U_{k-1} + C_2 U_{k-2} + \sum_{j=1}^{p} C_{m_j} U_{k-m_j}, \tag{3.4}$$

где

$$C_1 = 2(1.5I - \delta L_0)^{-1}, \quad C_2 = -0.5(1.5I - \delta L_0)^{-1}, \quad C_{m_j} = (1.5I - \delta L_0)^{-1}\delta L_{\tau_j},$$

а I означает единичную матрицу порядка n, и дополним уравнение (3.4) тождествами $U_i = U_i$, $i = k-1, \ldots, k-m_p+1$. Полученную таким образом систему из m_p уравнений можно записать в виде

$$X_k = MX_{k-1}, \tag{3.5}$$

где

$$X_k = \begin{pmatrix} U_k \\ \vdots \\ U_{k-m_p+1} \end{pmatrix}, \quad M = \begin{pmatrix} M_{11} & \cdots & M_{1m_p} \\ \vdots & & \vdots \\ M_{m_p1} & \cdots & M_{m_pm_p} \end{pmatrix}. \tag{3.6}$$

Матрица M в (3.6) является блочной, блочного порядка m_p с блоками порядка n. Все блоки этой матрицы нулевые, кроме поддиагональных блоков $M_{i+1,i} = I$, $i = 1, \ldots, m_p - 1$ и $p+2$ блоков $M_{11} = C_1$, $M_{12} = C_2$, $M_{1m_j} = C_{m_j}$, $j = 1, \ldots, p$, стоящих в первой блочной строке.

Чтобы ввести сеточный аналог максимальной амплификации (3.1), необходимо определить сеточный аналог нормы (2.3). Для численного интегрирования первого слагаемого под интегралом в (2.3) будем использовать формулу трапеций, а для интегрирования второго – формулу прямоугольников. При этом производную $dU'/d\xi$, входящую во второе слагаемое, будем аппроксимировать в точках $t_{k+1/2}$ с помощью центральной разности:

$$\left(\frac{dU'}{d\xi}\right)_{k+1/2} = \frac{U_{k+1} - U_k}{\delta}.$$

Можно показать, что полученный таким образом сеточный аналог локальной нормы (2.3) в точке t_k может быть записан в следующем виде:

$$\|HX_k\|_2, \tag{3.7}$$

где $H = P \otimes D$ – квадратная матрица порядка nm_p, P – верхний треугольный фактор разложения Холецкого симметричной положительно определенной трехдиагональной матрицы

$$\frac{1}{\delta}\begin{pmatrix} \delta^2/2 + \rho & -\rho & & & & \\ -\rho & \delta^2 + 2\rho & -\rho & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\rho & \delta^2 + 2\rho & -\rho & \\ & & & -\rho & \delta^2/2 + \rho \end{pmatrix}$$

порядка m_p, а $\otimes$ – символ Кронекерова произведения.

В силу (3.5), (3.6), (3.7), сеточный аналог Γ_k максимальной амплификации (3.1) нормы решения можно записать следующим образом:

$$\Gamma_k = \max_{X_0 \in \mathrm{span}(Q)\setminus\{0\}} \frac{\|HM^k X_0\|_2}{\|HX_0\|_2},$$

где Q – матрица, столбцы которой образуют базис в сеточном аналоге подпространства $\mathcal{Q}$, а $\mathrm{span}(Q)$ означает линейную оболочку столбцов матрицы Q. Для определенности будем предполагать, что базисные функции для каждой переменной, входящей в вектор U', выбраны одинаковыми и их количество равно d, где $d \le m_p$. Обозначим через G матрицу размера $m_p \times d$, столб-

цы которой являются сеточными аналогами этих базисных функций. Тогда в качестве Q можно можно выбрать матрицу $G \otimes I$ размера $nm_p \times nd$.

Учитывая, что $X_0 = Q\xi$, где ξ – некоторый nd-компонентный вектор, имеем

$$HX_0 = HQ\xi = (P \otimes D)(G \otimes I)\xi = (PG \otimes D)\xi.$$

Пусть $PG = \hat{Q}\hat{R}$ – QR-разложение, где $\hat{Q}$ – ортогональная прямоугольная матрица размера $m_p \times d$, а $\hat{R}$ – верхняя треугольная квадратная матрица порядка d. Вводя обозначения $\tilde{Q} = \hat{Q} \otimes I$, $\tilde{R} = \hat{R} \otimes D$, имеем $HX_0 = \tilde{Q}\tilde{R}\xi = \tilde{Q}\tilde{\xi}$, где $\tilde{\xi} = \tilde{R}\xi$. Следовательно,

$$\frac{\left\|HM^k X_0\right\|_2}{\left\|HX_0\right\|_2} = \frac{\left\|HM^k H^{-1} HX_0\right\|_2}{\left\|HX_0\right\|_2} = \frac{\left\|HM^k H^{-1}\tilde{Q}\tilde{\xi}\right\|_2}{\left\|\tilde{Q}\tilde{\xi}\right\|_2} = \frac{\left\|HM^k H^{-1}\tilde{Q}\tilde{\xi}\right\|_2}{\left\|\tilde{\xi}\right\|_2},$$

и, таким образом, имеем

$$\Gamma_k = \max_{\tilde{\xi}\neq 0} \frac{\left\|HM^k H^{-1}\tilde{Q}\tilde{\xi}\right\|_2}{\left\|\tilde{\xi}\right\|_2} = \left\|HM^k H^{-1}\tilde{Q}\right\|_2.$$

Для повышения эффективности вычисления Γ_k вместо матрицы $H^{-1}\tilde{Q}$ будем использовать равную ей матрицу $Q\tilde{R}^{-1}$. Тогда для вычисления Γ_k окончательно получим следующую формулу:

$$\Gamma_k = \left\|A_k\right\|_2,$$

где

$$A_k = HM^k Q\tilde{R}^{-1}. \tag{3.8}$$

Таким образом, вычисление Γ_k сводится к формированию матрицы $Y_0 = Q\tilde{R}^{-1}$ и вычислению матриц Y_k по рекуррентной формуле $Y_k = MY_{k-1}$ с одновременным вычислением норм матриц HY_k для всех натуральных k, не превосходящих $N = [T/\delta]$, где T – произвольная априорная верхняя оценка t_{opt}. При этом, учитывая блочную структуру матрицы M, ее можно умножать на матрицу Y_{k-1} следующим образом:

$$Y_{k1} = C_1 Y_{k-1,1} + C_2 Y_{k-1,2} + \sum_{j=1}^{p} C_{m_j} Y_{k-1,m_j}, \quad Y_{ki} = Y_{k-1,i-1}, \quad i = 2,\ldots,m_p,$$

где Y_{li} – матрицы размера $n \times nd$, являющиеся блочными строками матрицы

$$Y_l = \begin{pmatrix} Y_{l1} \\ \vdots \\ Y_{lm_p} \end{pmatrix}.$$

Пусть k_{opt} означает значение k, при котором достигается максимум Γ_k. Вычислив нормированный правый сингулярный вектор η^{opt} матрицы $A_{k_{\text{opt}}}$, отвечающий ее максимальному сингулярному числу [11], начальное значение X_0^{opt} сеточного аналога оптимального возмущения X_k^{opt} можно найти по формуле $X_0^{\text{opt}} = Q\tilde{R}^{-1}\eta^{\text{opt}}$.

Отметим, что величина шага сетки по времени, необходимая для достаточно точного численного интегрирования системы уравнений (2.2), а значит и вычисления максимальной амплификации $\Gamma(t)$, значительно меньше, чем это требуется для последующего вычисления с приемлемой точностью максимума $\Gamma(t)$ и оптимального возмущения, на котором достигается этот максимум. Поэтому для сокращения вычислительных затрат простейшего алгоритма имеет смысл вычислять $\left\|A_k\right\|_2$ не для всех натуральных k, не превосходящих $N = [T/\delta]$, а лишь для k кратных l, где l – некоторое заданное натуральное число.

Описанный алгоритм можно немного упростить, включив матрицу D в матрицу M. Действительно, правая часть равенства (3.8) не изменится, если в матрицах H и $\tilde{R}$ заменить матрицу D

единичной, а блоки C_i матрицы M заменить на DC_iD^{-1}. Мы будем считать, что такое предварительное преобразование матрицы M выполнено и матрица D, фигурирующая в матрицах H и $\tilde{R}$, – единичная.

4. МОДИФИКАЦИЯ ПРОСТЕЙШЕГО АЛГОРИТМА С ИСПОЛЬЗОВАНИЕМ МЕТОДА ЛАНЦОША

Учитывая, что M и остальные матрицы, образующие A_k в (3.8), являются разреженными матрицами большой размерности, для вычисления максимального сингулярного числа матрицы A_k и отвечающего ему правого сингулярного вектора можно использовать метод Ланцоша [12], применяя его к матрице $A_k^{\mathrm{T}}A_k$ для вычисления ее наибольшего собственного значения, корень из которого даст искомое сингулярное число, а собственный вектор матрицы $A_k^{\mathrm{T}}A_k$, отвечающий ее максимальному собственному значению, – искомый сингулярный вектор. Формальная схема алгоритма приведена ниже.

Алгоритм 1

Вычисление максимального сингулярного числа матрицы $A_k^{\mathrm{T}}A_k$ и отвечающего ему сингулярного вектора методом Ланцоша.

Input: матрицы H, M, Q, $\tilde{R}$ в разреженном формате, натуральное число k, начальный вектор v, минимальное допустимое относительное увеличение tol приближенного сингулярного числа за один шаг, максимальное число шагов $r_{\max}$ метода Ланцоша.

Output: приближенные значения наибольшего сингулярного числа s и соответствующего правого сингулярного вектора v матрицы A_k, определенной в (3.8).

if $r_{\max} > 1$ **then**

$$s_{-1} = s_0 = 0, \quad q_0 = 0, \quad \beta_0 = \|v\|_2, \quad r = 0$$

while $r < r_{\max}$ and $\beta_r > 0$ and $s_r \geq (1 + \mathrm{tol})s_{r-1}$ **do**

$$r := r + 1, \quad q_r = v/\beta_{r-1}$$

$$w = A_k^{\mathrm{T}}A_kq_r - \beta_{r-1}q_{r-1}, \quad \alpha_r = q_r^{\mathrm{T}}w,$$

$$v = w - \alpha_r q_r, \quad \beta_r = \|v\|_2$$

Сформировать $r \times r$ трехдиагональную матрицу $T_r = \mathrm{tridiag}(\beta_{i-1}, \alpha_i, \beta_i)$; $s_r = \sqrt{\lambda_{\max}(T_r)}$

end while

Вычислить собственный вектор y, отвечающий максимальному собственному значению матрицы T_r.

$$v = [q_1, \ldots, q_r]\, y$$

end if

Выполнить один шаг степенного метода:

$$w = A_kv, \quad v = w/\|w\|_2$$

$$w = A_k^{\mathrm{T}}v, \quad s = \|w\|_2, \quad v = w/s$$

Метод Ланцоша строит последовательность векторов $q_1, q_2, \ldots, q_r$, образующих ортонормированный базис в подпространстве Крылова

$$K_r(A_k^{\mathrm{T}}A_k, q_1) = \mathrm{span}(q_1, A_k^{\mathrm{T}}A_kq_1, \ldots, (A_k^{\mathrm{T}}A_k)^{r-1}q_1).$$

Пусть $T_r = \mathrm{tridiag}(\beta_{i-1}, \alpha_i, \beta_i)$ означает вещественную симметричную трехдиагональную матрицу с диагональными и внедиагональными элементами соответственно $\alpha_i = q_i^{\mathrm{T}}A_k^{\mathrm{T}}A_kq_i$ и $\beta_i = q_i^{\mathrm{T}}A_k^{\mathrm{T}}A_kq_{i+1}$. Тогда последовательность $\lambda_{\max}(T_r)$, где $\lambda_{\max}$ означает максимальное собственное значение, не

убывает (см., например, [12]) и стремится к наибольшему собственному значению матрицы $A_k^{\mathrm{T}}A_k$. Вместе с тем увеличение r ведет к увеличению объема данных, которые необходимо хранить, и времени вычислений. Чтобы избежать лишних вычислительных затрат, мы останавливаем алгоритм, когда рост $\sqrt{\lambda_{\max}(T_r)}$ становится слишком медленным или когда r достигает заданного максимально значения $r_{\max}$. Вектор $v = [q_1, q_2, \ldots, q_r]\, y_r$, где y_r – это собственный вектор, отвечающий $\lambda_{\max}(T_r)$, выбираем в качестве искомого приближенного правого сингулярного вектора, отвечающего максимальному сингулярному числу матрицы $A_k^{\mathrm{T}}A_k$. В отличие от простейшего алгоритма, в котором матрица A_k формируется явно, его модификация, использующая метод Ланцоша, требует лишь умножения сомножителей, образующих матрицы A_k и A_k^{T}, на вектор, что позволяет обойтись значительно меньшим объемом рабочей памяти.

Для сокращения вычислительных затрат, в качестве начального вектора при $k > 1$ можно брать сингулярный вектор, найденный при предыдущем значении k. Кроме того, как и в случае простейшего алгоритма, имеет смысл вычислять $\|A_k\|_2$ не при всех натуральных k, не превосходящих $N = [T/\delta]$, а лишь при k кратных l, где l – некоторое заданное натуральное число.

5. АЛГОРИТМ ПОСЛЕДОВАТЕЛЬНОЙ МАКСИМИЗАЦИИ

Алгоритм последовательной максимизации принципиально отличается от простейшего алгоритма и его модификации, описанной в предыдущем разделе, и состоит в следующем. Положим $k_1 = [N/2]$ и найдем максимальное сингулярное число матрицы A_{k_1} и отвечающий ему нормированный правый сингулярный вектор η с помощью метода Ланцоша, описанного выше. Вычислив последовательно нормы $\|A_1\eta\|_2, \ldots, \|A_N\eta\|_2$, положим

$$k_2 = \min \arg\max_{k\geq 1} \|A_k\eta\|_2.$$

Повторим вышеописанную процедуру с k_2 вместо k_1 и так далее до тех пор, пока k_i не совпадет с k_{i-1}. Это значение k_i и будет k_{opt}, причем по завершении процедуры мы одновременно получим максимальное сингулярное число матрицы $A_{k_{\mathrm{opt}}}$ и отвечающий ему нормированный правый сингулярный вектор η^{opt}. Также, как и в модификации простейшего алгоритма с помощью метода Ланцоша, в алгоритме последовательной максимизации матрицы A_k и A_k^{T} не формируется явно, а используется только умножение этих матриц на вектор. Формальная схема описанного алгоритма приведена ниже.

Алгоритм 2

Вычисление оптимального возмущения методом последовательной максимизации.

Input: матрицы H, M, Q, $\tilde{R}$ в разреженном формате, натуральное число N, параметры tol и $r_{\max}$ Алгоритма 1.

Output: приближенное значение сеточного аналога оптимального возмущения X_0^{opt}.

$$k_0 = 0$$
$$k_1 = [N/2]$$
$$i = 1$$

while $k_i \neq k_{i-1}$ **do**

Найти максимальное сингулярное число матрицы A_{k_i} и отвечающий ему нормированный правый сингулярный вектор η с помощью алгоритма 1.

Вычислить нормы $\|A_1\eta\|_2, \ldots, \|A_N\eta\|_2$, и найти $k_{i+1} = \min \arg \max_{k\geq 1} \|A_k\eta\|_2$.

$$i := i + 1$$

end while

$$X_0^{\mathrm{opt}} = Q\tilde{R}^{-1}\eta$$

Для сокращения вычислительных затрат в алгоритме последовательной максимизации можно вычислять нормы $\|A_k\eta\|_2$ не при всех k, а лишь при k, кратных l, где l – некоторое заданное натуральное число, однако выигрыш от этого будет не столь значительным, как в двух предыдущих алгоритмах, поскольку в данном случае требуется вычислять нормы векторов, а не матриц.

6. ОЦЕНКА СЛОЖНОСТИ АЛГОРИТМОВ

Оценим главные члены числа арифметических операций и ячеек памяти, необходимые каждому из трех алгоритмов, описанных в предыдущих разделах. Под ячейкой памяти будем понимать 64-х битовое вещественное число с плавающей точкой. Учитывая, что основные вычислительные затраты простейшего алгоритма, его модификации, использующей метод Ланцоша, приходятся на вычисление максимальной амплификации, затраты на последующее вычисление оптимального возмущения мы для этих алгоритмов учитывать не будем.

В соответствии со сказанным в конце разд. 3, будем считать, что блоки матрицы M преобразованы преобразованием подобия с матрицей D, а матрицу D в матрицах H и $\tilde{R}$ будем считать единичной порядка n. Поскольку матрица P является верхним треугольным фактором разложения Холецкого симметричной положительно определенной трехдиагональной матрицы порядка m_p, количество ненулевых элементов в этой матрице есть $2m_p - 1$. Поэтому для хранения матрицы P достаточно $2m_p$ ячеек памяти, а для умножения на вектор матрицы $H = P \otimes I$ без ее формирования – $3nm_p$ арифметических операций. Для хранения ненулевых и не единичных блоков матрицы M достаточно $(p+2)n^2$ ячеек памяти, а для умножения этой матрицы на вектор – $2(p+2)n^2$ арифметических операций. Для хранения $m_p \times d$ матрицы G достаточно $m_p d$ ячеек памяти, а для умножения матрицы $Q = G \otimes I$ на вектор без ее формирования – $2dnm_p$ арифметических операций. Наконец, поскольку $\hat{R}$ – верхняя треугольная квадратная матрица порядка d, для ее хранения требуется $d^2/2$ ячеек памяти, а для решения системы с этой матрицей достаточно d^2 арифметических операций. Поэтому для умножения матрицы $\tilde{R}^{-1} = \hat{R}^{-1} \otimes I$ на вектор без ее формирования достаточно nd^2 арифметических операций.

Таким образом, для умножения на вектор матриц H, M, Q и $\tilde{R}^{-1}$ без их формирования достаточно соответственно $3nm_p$, $2(p+2)n^2$, $2nm_p d$ и nd^2 арифметических операций, а для хранения необходимой для этого информации об этих матрицах достаточно соответственно $2m_p$, $(p+2)n^2$, $m_p d$ и $d^2/2$ ячеек памяти.

Простейший алгоритм сначала формирует $nm_p \times nd$ матрицу $Y_0 = Q\tilde{R}^{-1} = G\hat{R}^{-1} \otimes I$. Для этого достаточно $m_p d^2$ арифметических операций. Далее, на каждом k-ом шаге он вычисляет матрицу $Y_k = MY_{k-1}$, для чего достаточно $2(p+2)n^3 d$ арифметических операций. Если k кратно l, то на k-м шаге вычисляется матрица $A_k = HY_k$, для чего достаточно $3n^2 m_p d$ арифметических операций, и ее норма. Для вычисления последней с помощью R-SVD разложения [11] достаточно $2n^3 m_p d^2 + 2n^3 d^3$ арифметических операций. Так как формирование матрицы $Q\tilde{R}^{-1}$ производится один раз и, следовательно, требует небольшого по сравнению с другими шагами числа операций, главный член суммарного числа арифметических операций простейшего алгоритма составляет

$$2Nn^3 d\left(p + 2 + \frac{m_p d}{l}\right). \tag{6.1}$$

Здесь и далее предполагается, что $1 \ll d \ll m_p$, $m_p \ll N$, p – небольшое натуральное число, $1 \le n \ll m_p$.

Для реализации простейшего алгоритма необходимо хранить текущую матрицу Y_k, которая является плотной и требует для хранения $n^2 m_p d$ ячеек памяти. Это значительно больше числа ячеек памяти, необходимых для хранения информации о матрицах H и M, необходимой для их

умножения на вектор. Поэтому главный член числа ячеек памяти, необходимых для реализации простейшего алгоритма, – $n^2 m_p d$.

Перейдем к оценке вычислительных затрат модификации простейшего алгоритма, использующей метод Ланцоша. Из выполненного выше анализа следует, что для умножения матрицы A_k на вектор q_r в главном члене достаточно $2nm_p d + 2k(p+2)n^2$ арифметических операций. Столько же арифметических операций достаточно и для умножения на вектор матрицы A_k^{T}. Остальные вычислительные затраты r-го шага метода Ланцоша пренебрежимо малы, кроме, может быть, вычисления максимального собственного значения симметричной трехдиагональной матрицы T_r, которое требует $O(r^2)$ арифметических операций с небольшой мультипликативной константой [11]. Однако, как показывает вычислительная практика, для нахождения максимального сингулярного числа матрицы A_k и соответствующего ему правого сингулярного вектора в среднем достаточно не более полутора десятков шагов метода Ланцоша. Это число шагов плюс один шаг степенного метода мы будем обозначать через r_{av}. Поэтому главный член числа арифметических операций рассматриваемого алгоритма составляет

$$2r_{\mathrm{av}}\left(2\frac{N}{l}nm_p d + \frac{N^2}{l}(p+2)n^2\right). \tag{6.2}$$

Для его реализации необходимо хранить матрицы H, M, $\tilde{R}^{-1}$, Q, а также векторы $q_1,\ldots,q_{r_{\mathrm{av}}}$, что требует ndr_{av} ячеек памяти. Так как в силу сделанных ранее предположений количество ячеек памяти, необходимых для хранения информации о матрицах Q и M, намного больше, чем количество ячеек памяти, необходимое для хранения аналогичной информации о матрицах H, $\tilde{R}^{-1}$, а также для хранения вышеупомянутых векторов, главный член числа ячеек памяти, достаточного для реализации модификации простейшего алгоритма, использующей метод Ланцоша, составляет $m_p d + (p+2)n^2$.

Оценим теперь число арифметических операций, необходимое алгоритму последовательной максимизации. Каждая итерация этого алгоритма включает в себя нахождение максимального сингулярного числа и соответствующего ему правого сингулярного вектора матрицы A_{k_i} с помощью метода Ланцоша, что, согласно полученным выше оценкам, можно выполнить за $4r_{\mathrm{av}}(nm_p d + k_i(p+2)n^2)$ арифметических операций. Также каждая итерация требует вычисления $\|A_l\eta\|_2, \|A_{2l}\eta\|_2, \ldots, \|A_N\eta\|_2$. Для этого, так как произведение $Q\tilde{R}^{-1}\eta$ вычисляется один раз, в главном члене достаточно $(2(p+2)n^2 + 5nm_p/l)N$ арифметических операций. Следовательно, главный член числа арифметических операций алгоритма последовательной максимизации составляет

$$q\left(4r_{\mathrm{av}}(nm_p d + (p+2)n^2 k_{\mathrm{av}}) + \left(2(p+2)n^2 + \frac{5nm_p}{l}\right)N\right). \tag{6.3}$$

Здесь

$$k_{\mathrm{av}} = \frac{1}{q}\sum_{i=1}^{q} k_i,$$

где q – число итераций этого алгоритма. Число ячеек памяти, необходимых для реализации метода последовательной максимизации, очевидно совпадает с числом ячеек памяти, необходимых для реализации модификации простейшего алгоритма, использующей метод Ланцоша, то есть в главном члене составляет $m_p d + (p+2)n^2$.

Из выполненного анализа в частности следует, что с ростом числа уравнений n число ячеек памяти, необходимых для реализации простейшего алгоритма, значительно больше, чем требуют два других алгоритма. Оценки числа операций (6.1), (6.2) и (6.3) позволяют заключить, что в этом случае простейший алгоритм проигрывает и по числу арифметических операций, так как с увеличением n оно растет как n^3, а для модификации простейшего алгоритма и алгоритма последовательной максимизации – как n^2. Кроме того, при достаточно большом числе d базисных

функций число операций простейшего алгоритма так же значительно больше, чем у остальных алгоритмов, так как с увеличением d оно растет как d^2, а для остальных алгоритмов – линейно. Наконец, поскольку $m_p \sim 1/\delta$ и $N \sim 1/\delta$, где δ – шаг сетки по времени, число операций каждого алгоритма пропорционально $1/\delta^2$. При этом константа пропорциональности простейшего алгоритма, его модификации, использующей метод Ланцоша, и алгоритма последовательной максимизации равна $2Tn^3d^2\tau_p/l$, $2r_{\text{av}}(2Tn\tau_p d + T^2(p+2)n^2)/l$ и $5Tn\tau_p/l$ соответственно. То есть она значительно меньше у алгоритма последовательной максимизации.

Таким образом, в случае систем с большим числом переменных и/или базисных функций и/или малых шагов по времени эффективнее модификация простейшего алгоритма, использующая метод Ланцоша, и еще более эффективен алгоритм последовательной максимизации.

7. РЕЗУЛЬТАТЫ ЧИСЛЕННЫХ ЭКСПЕРИМЕНТОВ

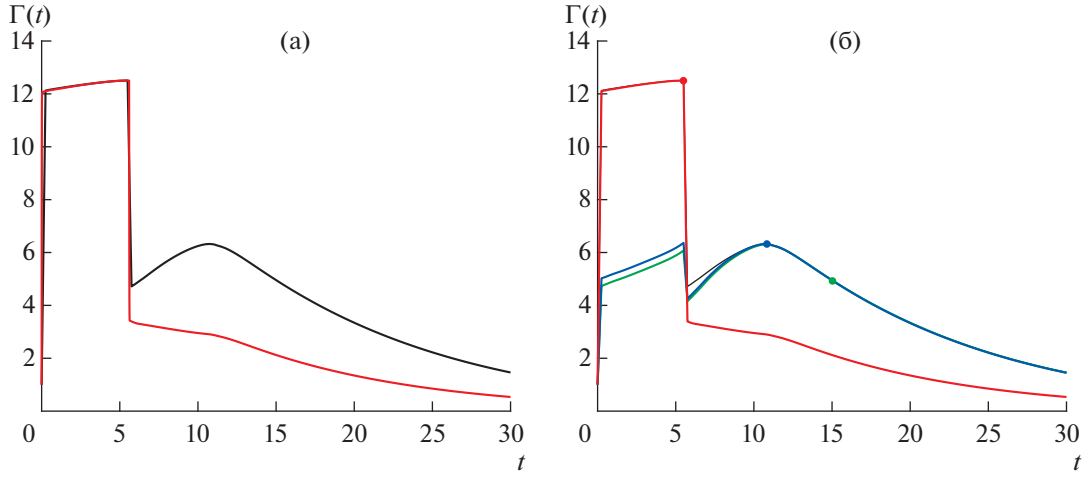
Проиллюстрируем работу простейшего алгоритма, его модификации, использующей метод Ланцоша, и алгоритма последовательной максимизации, которые далее для краткости будем называть соответственно алгоритмами I, II и III, на примере модели динамики инфекции, вызванной вирусами лимфоцитарного хориоменингита. Эта модель была сформулирована в работе [8] в виде системы нелинейных дифференциальных уравнений вида (2.1) с $p = 2$, $\tau_1 = 0.6$, $\tau_2 = 5.6$, и описывает динамику концентраций вирусных частиц V, клеток-прекурсоров E_p, клеток-эффекторов E_e и кумулятивной вирусной нагрузки W. При значениях параметров, указанных в табл. 5 из [8], кроме $\beta = 0.102$ и $\alpha_{E_p} = 0.006$, данная модель имеет следующее устойчивое стационарное состояние: $V \approx 4.8\times10^7$, $E_p \approx 1.0$, $E_e \approx 1.8\times10^{-4}$, $W \approx 4.4\times10^8$. В качестве базисных функций для вычисления оптимальных возмущений будем использовать следующие функции, качественно аппроксимирующие поведение препаратов в рамках однокамерных и двухкамерных фармакокинетических моделей:

$$\phi(t,t_0) = \begin{cases} 0, & -\tau_2 \le t < t_0, \\ \exp\{-3(t-t_0)\} - \exp\{-9(t-t_0)\}, & t_0 \le t \le 0. \end{cases} \tag{7.1}$$

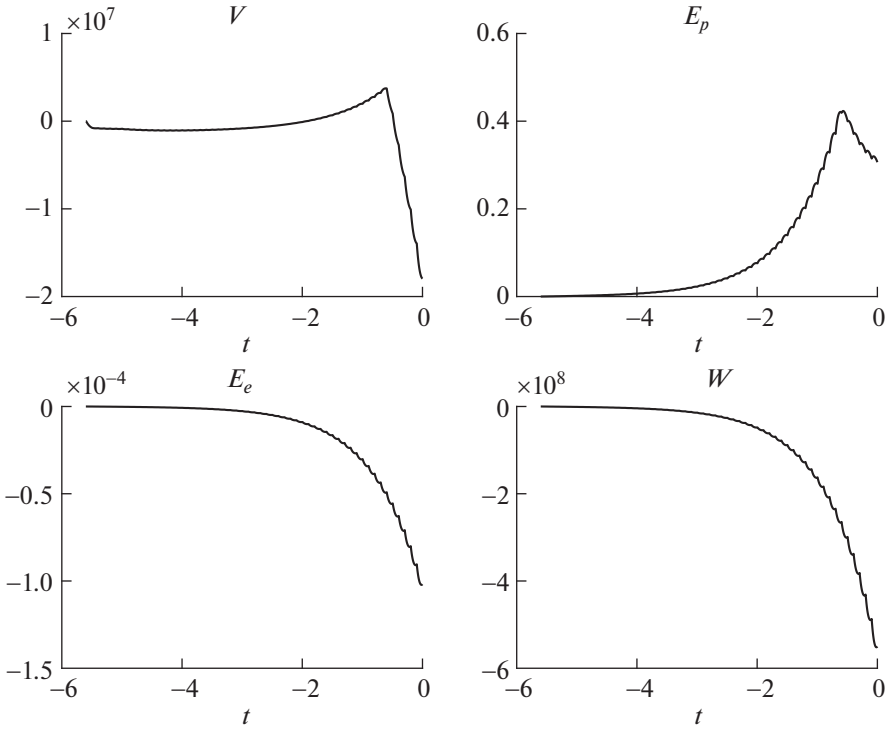
Для расчетов будем использовать равномерную сетку с шагом $\delta = 5\times10^{-3}$. В качестве априорной верхней оценки t_{opt} выберем $T = 30$. Таким образом, будем иметь $n = 4$, $p = 2$, $N = 6000$, $m_p = 1120$. Для каждой переменной будем использовать d базисных функций вида (7.1) с узлами равномерной сетки в интервале $-\tau_2 < t < 0$ в качестве t_0. Расчеты будем проводить с $l = 1$ и 50. Для нахождения максимального сингулярного числа и отвечающего ему правого сингулярного вектора методом Ланцоша выберем $\text{tol} = 10^{-9}$, а в качестве начального вектора «v всегда будем выбирать случайный вектор.

Рассмотрим четыре тестовые задачи, которые в дальнейшем будем называть тестами 1–4. Между собой эти тесты отличаются значением параметра d, видом локальной нормы, в которой вычисляется оптимальное возмущение, и весами локальной нормы (диагональные элементы матрицы D). А именно, $d = 14$ в тесте 2 и 56 во всех остальных тестах. В качестве локальной нормы, в тесте 4 выбрана L_2-норма, а во всех остальных тестах – W_2^1. В тесте 3 веса берутся единичными, а во всех остальных тестах они берутся равными величинам, обратным компонентам рассматриваемого стационарного состояния.

Для каждого из тестов с помощью алгоритмов I и II были рассчитаны максимальная амплификация, оптимальное возмущение и локальная норма оптимального возмущения. Максимальная амплификация для случая $l = 50$ изображена на фиг. 1а, 3а, 5а и 7а. Там же изображена локальная норма найденного оптимального возмущения, а компоненты его начального значения изображены на фиг. 2, 4, 6 и 8. Стоит отметить, что вид максимальной амплификации в случае тестов 1 и 2 (фиг. 1а и 3а соответственно) не является типичным. Более типичными являются максимальные амплификации, полученные для тестов 3 и 4 и изображенные на фиг. 5а и 7а соответственно.



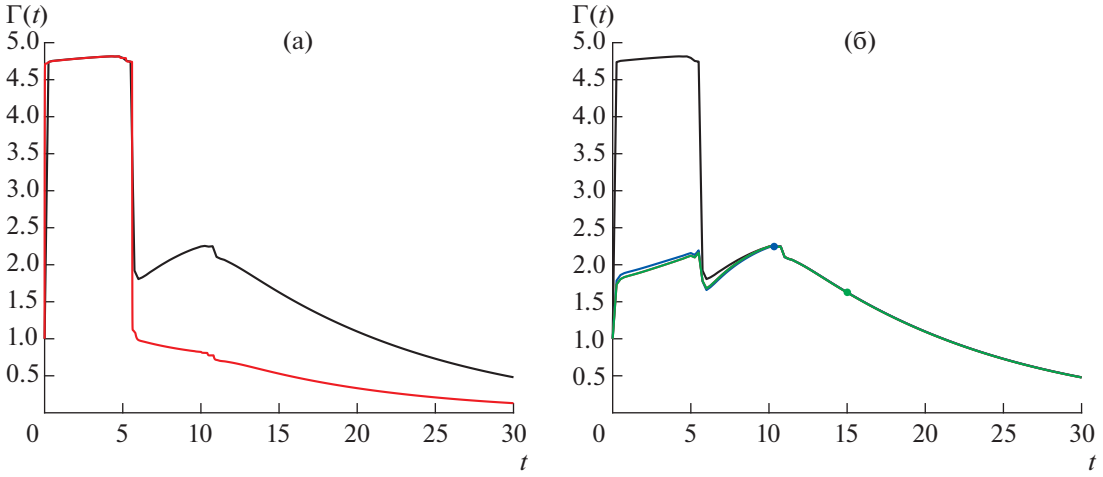
Фиг. 1. Максимальная амплификация (черная линия) для теста 1 и локальная норма оптимального возмущения, полученные алгоритмами I и II (красная линия) (а) и локальные нормы приближений к оптимальному возмущению, найденных на итерациях алгоритма III: первой – зеленая, второй – синяя, третьей – красная линии (б).



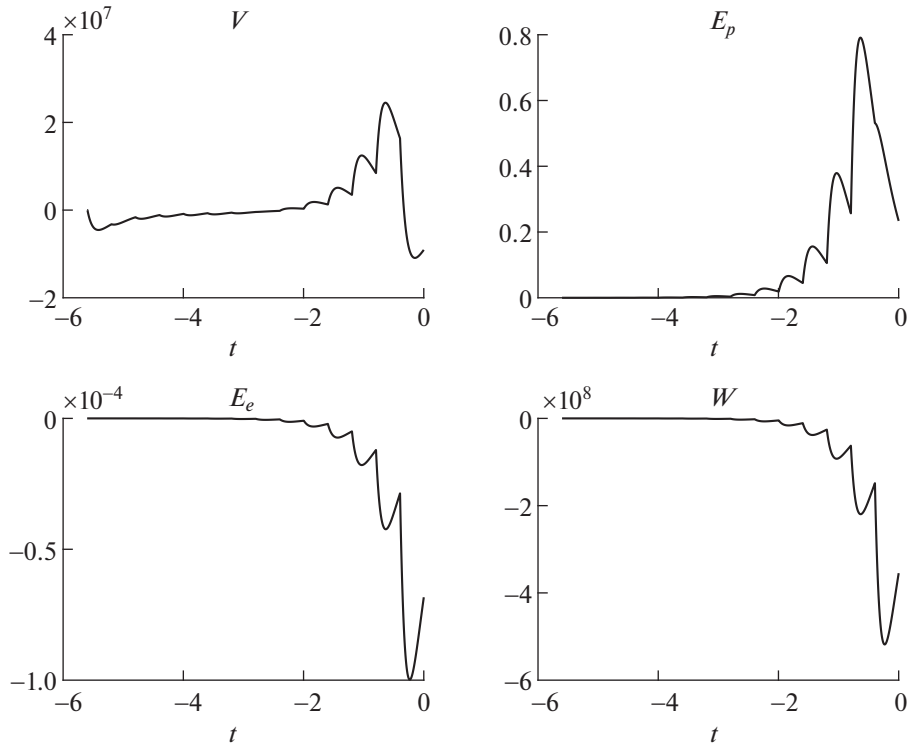
Фиг. 2. Компоненты начального значения оптимального возмущения для теста 1.

Фигуры 1б, 3б, 5б и 7б демонстрируют работу алгоритма III при $l = 50$. Крупные точки соответствуют моментам времени $k_1\delta$ (зеленая), $k_2\delta$ (синяя), $k_3\delta$ (красная), то есть первому, второму и третьему приближениям к t_{opt}.

В случае теста 1 все рассматриваемые алгоритмы дали одинаковые результаты $t_{\text{opt}} \approx 5.50$, $\Gamma(t_{\text{opt}}) \approx 12.5$ с точностью до 16 значащих десятичных цифр вне зависимости от значения l. В случае тестов 3 и 4 все алгоритмы с точностью до 16 значащих десятичных цифр дали одинаковые



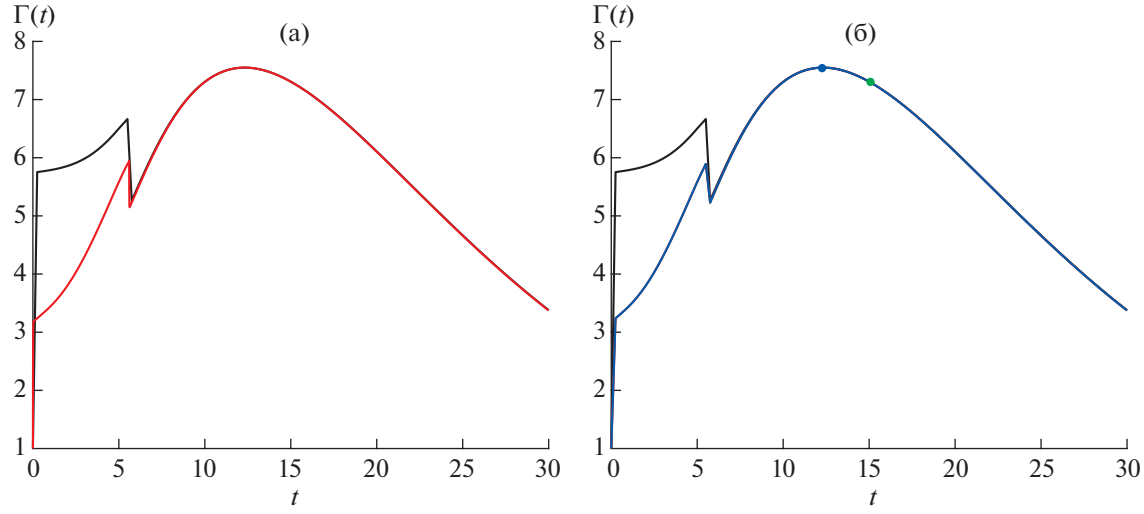
Фиг. 3. Максимальная амплификация (черная линия) для теста 2 и локальная норма оптимального возмущения, полученные алгоритмами I и II (красная линия) (а) и локальные нормы приближений к оптимальному возмущению, найденных на итерациях алгоритма III: первой – зеленая, второй – синяя линии (б).



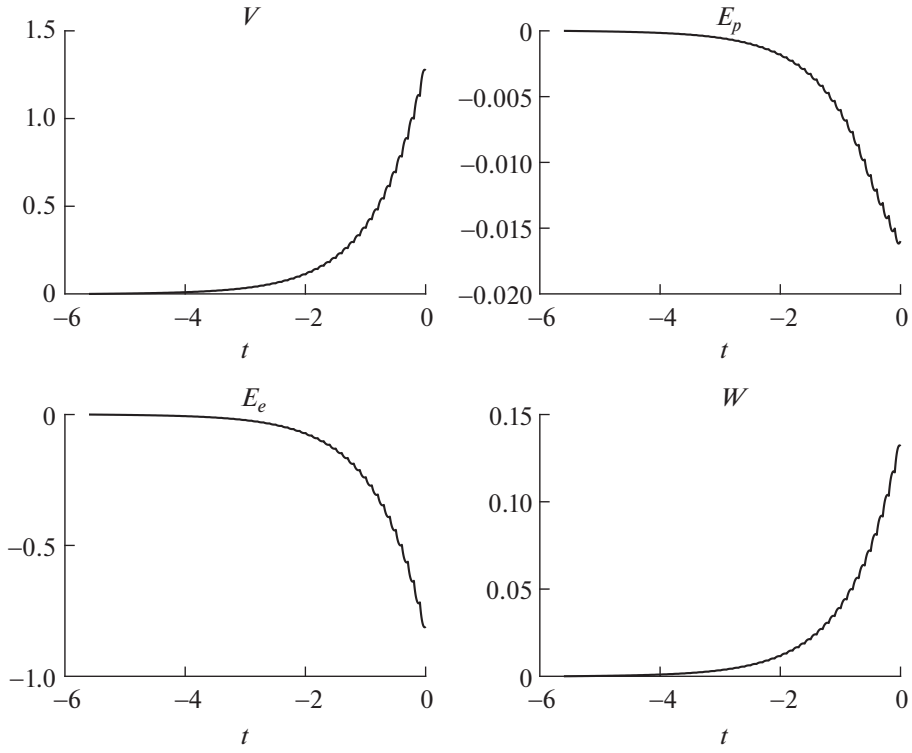
Фиг. 4. Компоненты начального значения оптимального возмущения для теста 2.

результаты, но различные при $l = 1$ и 50: $t_{\text{opt}} \approx 12.32$, $\Gamma(t_{\text{opt}}) \approx 7.5498$ при $l = 1$ и $t_{\text{opt}} \approx 12.25$, $\Gamma(t_{\text{opt}}) \approx 7.5496$ при $l = 50$ в случае теста 3, и $t_{\text{opt}} \approx 11.27$, $\Gamma(t_{\text{opt}}) \approx 39.7884$ при $l = 1$ и $t_{\text{opt}} \approx 11.25$, $\Gamma(t_{\text{opt}}) \approx 39.7883$ при $l = 50$ в случае теста 4.

В случае теста 2 алгоритмы I и II с точностью до 16 значащих десятичных цифр дали одинаковые результаты $t_{\text{opt}} \approx 4.40$, $\Gamma(t_{\text{opt}}) \approx 4.817$ при $l = 1$ и $t_{\text{opt}} \approx 4.25$, $\Gamma(t_{\text{opt}}) \approx 4.815$ при $l = 50$. Алгоритм III остановился после $q = 2$ итераций. При этом были получены следующие результаты:



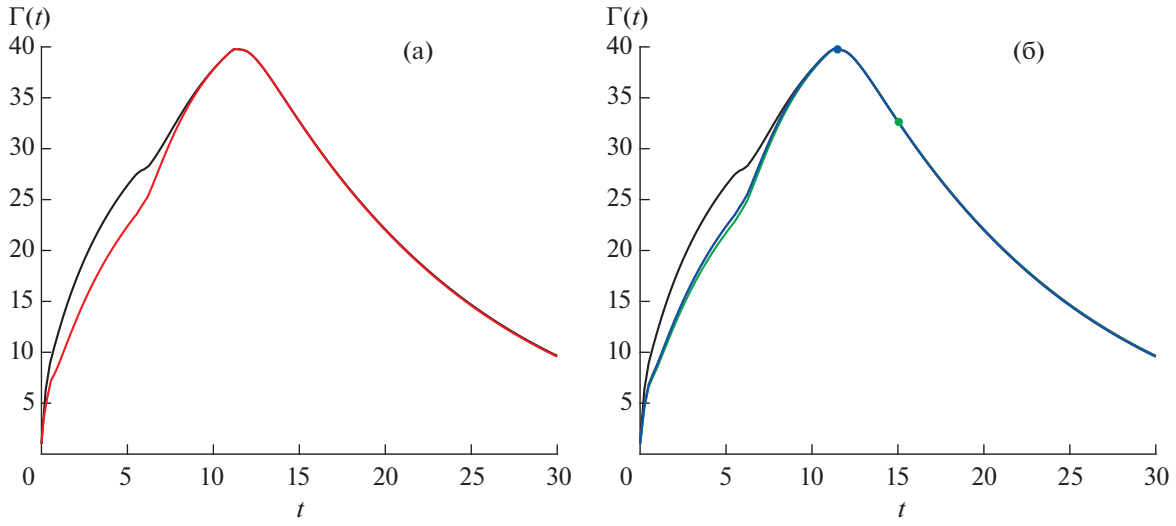
Фиг. 5. Максимальная амплификация (черная линия) для теста 3 и локальная норма оптимального возмущения, полученные алгоритмами I и II (красная линия) (а) и локальные нормы приближений к оптимальному возмущению, найденных на итерациях алгоритма III: первой – зеленая, второй – синяя (б).



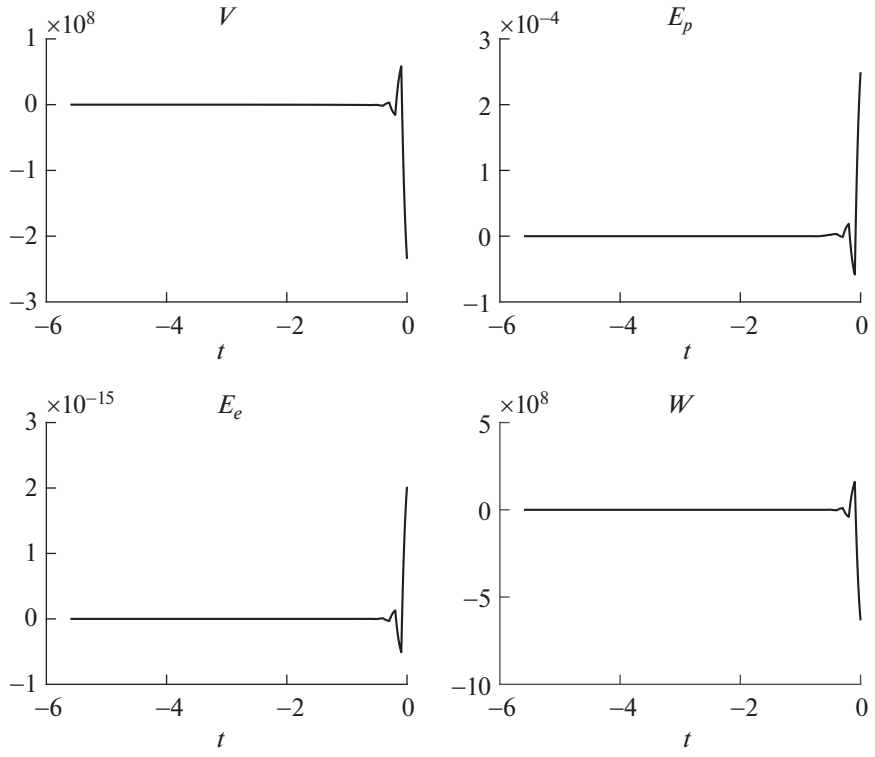
Фиг. 6. Компоненты начального значения оптимального возмущения для теста 3.

$t_{opt} \approx 10.41$, $\Gamma(t_{opt}) \approx 2.26$ при $l = 1$ и $t_{opt} \approx 10.25$, $\Gamma(t_{opt}) \approx 2.25$ при $l = 50$. То есть в данном случае последовательная максимизация сходилась не к оптимальному возмущению, а к возмущению, на котором достигается локальный максимум.

В табл. 1 приведены среднее число шагов метода Ланцоша r_{av} (оно оказалось одинаковым для алгоритмов II и III во всех тестах, кроме теста 4), число итераций q алгоритма III и k_{av}. Подставляя эти величины и указанные ранее значения параметров расчета в формулы (6.1), (6.2), (6.3),



Фиг. 7. Максимальная амплификация (черная линия) для теста 4 и локальная норма оптимального возмущения, полученные алгоритмами I и II (красная линия) (а) и локальные нормы приближений к оптимальному возмущению, найденных на итерациях алгоритма III: первой – зеленая, второй – синяя (б).



Фиг. 8. Компоненты начального значения оптимального возмущения для теста 4.

получим приведенные в табл. 2 числа операций, необходимых каждому из рассматриваемых алгоритмов. Для двух первых алгоритмов учитывалось только число операций, необходимых для вычисления максимальной амплификации. В табл. 3 указаны времена работы всех трех алгоритмов в среде MATLAB на компьютере с 2-х ядерным процессором Intel Core i7 с частотой 2.4 GHz. Для двух первых алгоритмов учитывалось только время вычисления максимальной амплификации. Вычисление самого оптимального возмущения требовало еще 4.2 и 0.3 секунд соответ-

Таблица 1. Результаты работы алгоритмов II и III

	Тест 1		Тест 2		Тест 3		Тест 4	
	$l = 1$	$l = 50$	$l = 1$	$l = 50$	$l = 1$	$l = 50$	$l = 1$	$l = 50$
r_{av}	6	6	6	6	5	5	6/5	6/5
q	4	3	2	2	3	2	4	2
k_{av}	1845	2083	2541	2525	2642	2725	2444	2625

Таблица 2. Число операций

Алгоритм	Тест 1		Тест 2		Тест 3		Тест 4	
	$l = 1$	$l = 50$	$l = 1$	$l = 50$	$l = 1$	$l = 50$	$l = 1$	$l = 50$
I	2.7×10^{12}	5.4×10^{10}	1.7×10^{11}	3.4×10^{9}	2.7×10^{12}	5.4×10^{10}	2.7×10^{12}	5.4×10^{10}
II	6.4×10^{10}	1.3×10^{9}	3.7×10^{10}	7.3×10^{8}	5.3×10^{10}	1.1×10^{9}	6.4×10^{10}	1.3×10^{9}
III	5.8×10^{8}	3.8×10^{7}	2.8×10^{8}	1.8×10^{7}	4.3×10^{8}	2.4×10^{7}	2.5×10^{8}	2.0×10^{7}

Таблица 3. Время работы (в секундах)

Алгоритм	Тест 1		Тест 2		Тест 3		Тест 4	
	$l = 1$	$l = 50$	$l = 1$	$l = 50$	$l = 1$	$l = 50$	$l = 1$	$l = 50$
I	184.9	23.2	28.1	5.7	184.9	23.2	178.9	22.1
II	3155	63.2	3064	61.4	3154	63.2	3229	64.7
III	2.4	1.7	1.5	1.3	2.1	1.5	2.5	1.3

Таблица 4. Производительность (число операций в секунду)

Алгоритм	Тест 1		Тест 2		Тест 3		Тест 4	
	$l = 1$	$l = 50$	$l = 1$	$l = 50$	$l = 1$	$l = 50$	$l = 1$	$l = 50$
I	1.5×10^{10}	2.3×10^{9}	6.0×10^{9}	6.0×10^{8}	1.5×10^{10}	2.3×10^{9}	1.5×10^{10}	2.4×10^{9}
II	2.0×10^{7}	2.1×10^{7}	1.2×10^{7}	1.2×10^{7}	1.7×10^{7}	1.7×10^{7}	2.0×10^{7}	2.0×10^{7}
III	2.4×10^{8}	2.2×10^{7}	1.9×10^{8}	1.4×10^{7}	2.0×10^{8}	1.6×10^{7}	1.0×10^{8}	1.5×10^{7}

ственно. Алгоритм III для нахождения оптимального возмущения не требует дополнительных вычислений. Отношение числа операций, необходимых для реализации каждого алгоритма на время его работы, дает производительность, которая приведена в табл. 4.

Из табл. 4 видно, что производительность на алгоритме I при всех l и на алгоритме III при $l = 1$ существенно выше производительности на алгоритме II при всех l и на алгоритме III при $l = 50$. Это связано с тем, что некоторые стандартные функции среды MATLAB, реализующие матричные алгоритмы, особенно эффективно используют вычислительные возможности компьютера. На этих функциях достигается существенно более высокая производительность, чем на других стандартных функциях MATLAB и тем более на пользовательских реализациях тех же и других алгоритмов на языке среды MATLAB.

Поясним сказанное на примере теста 1. Вычисления алгоритма I при $l = 1$ можно разбить на три части: вычисление $Y_k = M^k Y_0$ при $0 \le k \le N$, вычисление матриц $A_k = HY_k$ для всех найденных матриц Y_k, и вычисление вторых норм всех найденных матриц A_k. Для первой части достаточно 2×10^8, для второй – 1.8×10^{10}, для третьей – 2.7×10^{12} арифметических операций. Первая часть была реализована нами на языке MATLAB без существенного использования стандартных

функций, во второй части использовалось стандартное умножение на разреженную матрицу в разреженном формате. В третьей части использовалась очень эффективная стандартная функция norm (время ее работы совпадает с временем вычисления сингулярных чисел с помощью стандартной функции svd на матрице того же размера). Первая часть потребовала 20.6, вторая – 26, третья – 158 секунд. Таким образом, производительности на этих трех частях алгоритма I оказались существенно различными и составляли соответственно 9.7×10^6, 6.9×10^8 и 1.7×10^{10} операций в секунду. При $l = 50$ алгоритм I выполняет в 50 раз меньше умножений на матрицу H и вычислений матричных норм по сравнению с числом умножений на матрицу M. Как следствие, производительность на нем уменьшается на порядок.

В алгоритме II число умножений на матрицу H пренебрежимо мало по сравнению с числом умножений на матрицу M, а нормы матриц стандартными функциями вообще не вычисляются. Отсюда и сравнительно низкая производительность.

На алгоритме III более высокая производительность при $l = 1$, чем при $l = 50$, достигается благодаря большему вкладу в общее число операций вычисления $\|A_1\eta\|_2, \|A_{21}\eta\|_2, \ldots, \|A_N\eta\|_2$, для которого мы использовали стандартную функцию norm и стандартное умножение на разреженную матрицу.

8. ЗАКЛЮЧЕНИЕ

В данной работе алгоритмы, разработанные ранее для вычисления оптимальных возмущений стационарных состояний систем без запаздывания, были модифицированы для поиска оптимальных возмущений стационарных состояний систем с запаздыванием. Был выполнен сравнительный анализ вычислительных затрат этих алгоритмов. Их работа была проиллюстрирована на примере модели динамики заболевания, вызванной вирусами лимфоцитарного хориоменингита.

Алгоритмом, наиболее точно вычисляющим оптимальные возмущения, очевидно, является простейший алгоритм, поскольку в нем вычисление максимального сингулярного числа и отвечающего ему правого сингулярного вектора выполняются с машинной точностью. Однако для систем с достаточно большим количеством переменных он требует больше памяти и числа арифметических операций. Менее точной, но более эффективной является модификация этого алгоритма с использованием метода Ланцоша, хотя по времени работы наша реализация этой модификации в среде MATLAB в рассмотренных случаях уступает нашей реализации простейшего алгоритма в силу используемых в коде стандартных процедур умножения на разреженную матрицу и вычисления второй нормы матрицы, очень эффективно реализованных в среде MATLAB.

Наиболее эффективным алгоритмом вычисления оптимального возмущения оказался алгоритм последовательной максимизации. Он предпочтительнее с точки зрения объема требуемой памяти и количества операций, более того, время работы его реализации в среде MATLAB значительно меньше времен работы простейшего алгоритма и его модификации, использующей метод Ланцоша. Хотя алгоритм последовательной максимизации не гарантирует, вообще говоря, получения правильного результата во всех случаях, поскольку этот метод может сойтись к локальному максимуму, тем не менее его имеет смысл использовать при больших параметрических расчетах, выборочно контролируя результат с помощью одного из двух других алгоритмов.

СПИСОК ЛИТЕРАТУРЫ

1. *Boiko A.V., Nechepurenko Y.M., Sadkane M.* Fast computation of optimal disturbances for duct flows with a given accuracy // Ж. вычисл. матем. и матем. физ. 2010. Т. 50. № 11. С. 2017–2027.
2. *Boiko A.V., Nechepurenko Y.M., Sadkane M.* Computing the maximum amplification of the solution norm of differential-algebraic systems // Comput. Math. Model. 2012. V. 23. № 2. P. 216–227.
3. *Nechepurenko Y.M., Sadkane M.* A low-rank approximation for computing the matrix exponential norm // SIAM J. Matrix. Anal. Appl. 2011. V. 32. № 2. P. 349–363.
4. *Бочаров Г.А., Нечепуренко Ю.М., Христиченко М.Ю., Гребенников Д.С.* Оптимальные возмущения систем с запаздывающим аргументом для управления динамикой инфекционных заболеваний на основе многокомпонентных воздействий // Современная математика. Фундаментальные направления. 2017. Т. 63. № 3. С. 392–417.

5. *Бочаров Г.А., Нечепуренко Ю.М., Христиченко М.Ю., Гребенников Д.С.* Управление моделями вирусных инфекций с запаздывающими переменными на основе оптимальных возмущений // Препринты ИПМ им. Келдыша. 2017. № 52. 28 с.
6. *Bocharov G.A., Nechepurenko Yu.M., Khristichenko M.Yu., Grebennikov D.S.* Maximum response perturbation-based control of virus infection model with time-delays // Russian J. Num. Anal. Math. Model. 2017. V. 32. № 5. P. 275–291.
7. *Бочаров Г.А., Нечепуренко Ю.М., Христиченко М.Ю., Гребенников Д.С.* Оптимальные возмущения бистабильных систем с запаздыванием, моделирующих вирусные инфекции // Докл. АН. 2018. Т. 481. № 2. С. 123–126.
8. *Bocharov G.A.* Modelling the dynamics of LCMV infection in mice: conventional and exhaustive CTL responses // J. Theor. Biol. 1998. V. 192. № 3. P. 283–308.
9. *Nechepurenko Y.M., Sadkane M.* Computing humps of the matrix exponential // J. Comput. Appl. Math. 2017. V. 319. P. 87–96.
10. *Hairier E., Norsett S., Wanner G.* Solving ordinary differential equations. Berlin: Springer, 1987.
11. *Golub G.H., Van Loan C.F.* Matrix computations. London: Jhon Hopkins University press, 1989.
12. *Parlett B.N.* The symmetric eigenvalue problem. Berkeley: SIAM, 1998.