ЖУРНАЛ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ФИЗИКИ, 2021, том 61, № 5, с. 787–799

ОБЩИЕ ЧИСЛЕННЫЕ МЕТОДЫ

УДК 519.614

НОВЫЕ АЛГОРИТМЫ ДЛЯ РЕШЕНИЯ НЕЛИНЕЙНОЙ ПРОБЛЕМЫ СОБСТВЕННЫХ ЗНАЧЕНИЙ

© 2021 г. В. Гандер[1,2]

[1] *8092 Zurich, Ramistrasse 1010, ETH, Switzerland*

[2] *Hong Kong Baptist University, 224 Waterloo Rd, Kowloon Tong, Hong Kong*

**e-mail: gander@inf.ethz.ch*

Поступила в редакцию 24.12.2020 г.
Переработанный вариант 24.12.2020 г.
Принята к публикации 14.01.2021 г.

Для решения нелинейной проблемы собственных значений предлагаются алгоритмы, использующие методы третьего порядка для вычисления нулей уравнения $\det A(\lambda) = 0$. Производные определителя вычисляются с помощью алгоритмического дифференцирования. Специальные алгоритмы представлены в случае ленточных матриц. Библ. 11. Фиг. 6. Табл. 2.

Ключевые слова: нелинейная проблема собственных значений, методы третьего порядка, алгоритмическое дифференцирование.

DOI: 10.31857/S0044466921050094

1. ВВЕДЕНИЕ

Пусть $A\colon \lambda \mapsto \mathbb{C}^{n\times n}$ – аналитическое отображение на открытой области $\{\lambda\} \subset \mathbb{C}$. Наша задача: найти λ такое, что $f(\lambda) = \det A(\lambda) = 0$.

Для вычисления нуля функции f можно рассмотреть метод Ньютона:

$$\lambda_{k+1} = \lambda_k - \frac{f(\lambda_k)}{f'(\lambda_k)},$$

использующий производные определителя. *Формула Якоби*, широко известная и обсуждаемая в учебниках линейной алгебры, дает для них явное выражение

$$f'(\lambda) = \det A(\lambda)\,\mathrm{tr}(A^{-1}(\lambda)A'(\lambda)).$$

Ньютоновская коррекция получает вид

$$\frac{f(\lambda_k)}{f'(\lambda_k)} = \frac{1}{\mathrm{tr}(A^{-1}(\lambda_k)A'(\lambda_k))}.$$

Альтернативным подходом к вычислению производных, а значит, и ньютоновской коррекции, является алгоритмическое дифференцирование (см. [1]–[3]).

2. ВЫЧИСЛЕНИЕ ОПРЕДЕЛИТЕЛЕЙ И НЬЮТОНОВСКОЙ КОРРЕКЦИИ

При получении определителя обычно строится LU-разложение по методу Гаусса. Пусть $PA = LU$, где P включает перестановки, связанные с частичным выбором ведущего элемента, L – нижняя унитреугольная матрица и U – верхняя треугольная матрица. Тогда

$$\det(A) = \pm u_{11}u_{22}\cdots u_{nn}.$$

Когда LU-разложение записывается на месте матрицы A, текущее значение определителя на k-м шаге исключения умножается на ведущий элемент: $f := f \times a_{kk}$. Это довольно быстро ведет к появлению очень больших или очень малых чисел. Поэтому лучше перейти к логарифмам: $\log f := \log f + \log(a_{kk})$.

Заметим, что производная логарифма

$$\frac{d}{d\lambda}\log f(\lambda) = \frac{f'(\lambda)}{f(\lambda)}$$

является обратной величиной к ньютоновской коррекции. Таким образом, если производная логарифма вычисляется алгоритмическим дифференцированием

$$\text{logf} := \text{logf} + \log(a_{kk}), \quad \Rightarrow \quad \text{logfp} := \text{logfp} + \frac{a'_k}{a_{kk}},$$

то обратная величина ffp = 1/logfp = $f(\lambda)/f'(\lambda)$ – в точности нужная нам ньютоновская коррекция. Это наблюдение используется в следующей программе (см. [2]):

```
function ffp=deta(A,Ap)
% DETA compute determinant of A and derivative
% Given A=A(lambda) and Ap=A'(lambda), DETA(A,Ap)
% computes Newton correction ffp=f/f' where f=det(A).
n=length(A); logfp=0;
for j=1:n
    [amax,kmax]= max(abs(A(j:n,j)));        % partial pivoting
    if amax == 0,ffp=0; return, end
    kmax=kmax+j-1;
    if kmax ~= j                            % interchange rows
       h=Ap(kmax,:); Ap(kmax,:)=Ap(j,:); Ap(j,:)=h;
       h=A(j,:); A(j,:)=A(kmax,:); A(kmax,:)=h;
    end
    logfp=logfp + Ap(j,j)/A(j,j);
    Ap(j+1:n,j)=(Ap(j+1:n,j)*A(j,j)-A(j+1:n,j)*Ap(j,j))/A(j,j)^2;
    A(j+1:n,j)=A(j+1:n,j)/A(j,j);
    Ap(j+1:n,j+1:n)=Ap(j+1:n,j+1:n) - Ap(j+1:n,j)*A(j,j+1:n)- …
                      A(j+1:n,j)*Ap(j,j+1:n);
    A(j+1:n,j+1:n)=A(j+1:n,j+1:n) - A(j+1:n,j)*A(j,j+1:n);
end
ffp=1/logfp;
```

3. ПОДАВЛЕНИЕ ВМЕСТО ДЕФЛЯЦИИ

С помощью функции `deta` мы можем вычислить решение уравнения $\det A(\lambda) = 0$ по методу Ньютона. Чтобы избежать перевычисления уже найденных нулей $\lambda_1, \ldots, \lambda_k$, мы подавим их, работая с функцией

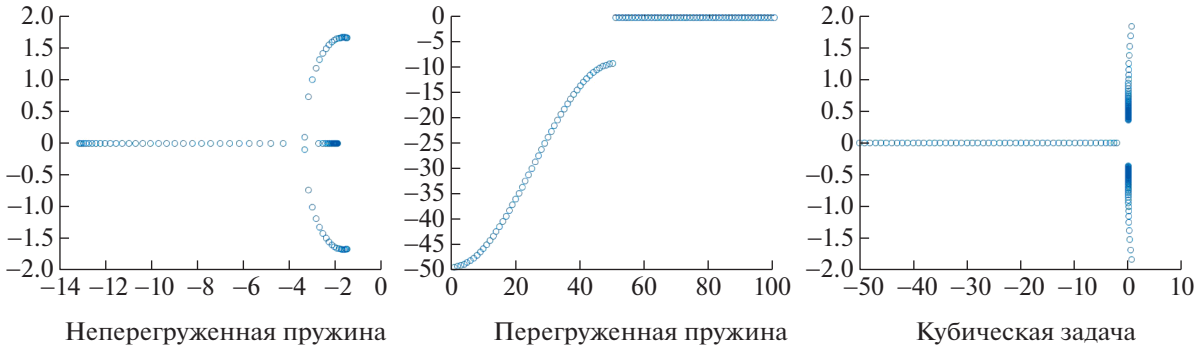
$$f_k(\lambda) := \frac{f(\lambda)}{p(\lambda)},$$

где $p(\lambda) = (\lambda - \lambda_1)\cdots(\lambda - \lambda_k)$. Тогда

$$p'(\lambda) = \sum_{j=1}^{k}\prod_{\substack{i=1\\ i\neq j}}^{k}(\lambda - \lambda_i) = p(\lambda)s(\lambda), \quad \text{где} \quad s(\lambda) = \sum_{j=1}^{k}\frac{1}{\lambda - \lambda_j}.$$

Производные для f_k имеют вид (мы опускаем аргумент λ)

$$f_k' = \frac{pf' - psf}{p^2} = \frac{f' - sf}{p}.$$



Фиг. 1. Примеры применения итераций Ньютона.

Ньютонова коррекция f_k/f_k', выраженная через f и f', приобретает вид

$$\frac{f_k}{f_k'} = \frac{f/p}{(f' - sf)/p} = \frac{f}{f' - sf} = \frac{f}{f'}\frac{1}{1 - \dfrac{f}{f'}s}. \tag{1}$$

В итоге получается итерация

$$\lambda_{j+1} = \lambda_j - \frac{f_k(\lambda_j)}{f_k'(\lambda_j)} = \lambda_j - \frac{f(\lambda_j)}{f'(\lambda_j)}\frac{1}{1 - \dfrac{f(\lambda_j)}{f'(\lambda_j)}\displaystyle\sum_{j=1}^{k}\frac{1}{\lambda - \lambda_j}},$$

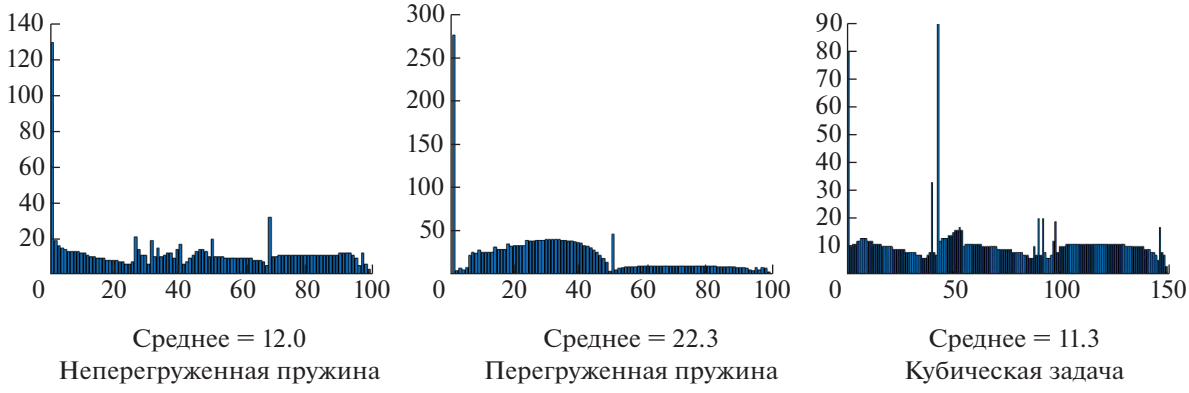
называемая итерацией *Ньютона–Мехли (Newton–Maehly)* (см. [4]).

В [2] мы привели расчеты для двух примеров масс с пружинами из [5] и для кубического примера из [1]. Вот MATLAB-программа для первого примера масс с пружинами:

```
n=50, tau=3, kappa=5,                    % nonoverdamped
e=-ones(n-1,1);
C=(diag(e,-1)+diag(e,1)+3*eye(n)); K=kappa*C; C=tau*C;
lam=-0.5+0.1*i; lamb=[];                 % start
for k=1:2*n
    ffp=1;
    while abs(ffp)>1e-14
        Qp=2*lam*eye(n)+C; Q=lam*(lam*eye(n)+ C)+K;
        ffp=deta(Q,Qp);
        s=sum(1./(lam-lamb(1:k-1)));
        lam=lam-ffp/(1-ffp*s);           % Newton step
    end
    lamb(k)=lam;
    lam=lam*(1+0.01*i);                  % start for next eigenvalue
end
plot(lamb,'o')
```

Итерация для первого собственного значения начинается с выбора случайного комплексного числа, здесь $\lambda_0 = -0.5 + 0.1i$. В качестве начального значения для следующего собственного значения мы выбираем последнее из найденных и подавляем значение λ_k с помощью малого возмущения: $\lambda_0 = \lambda_k(1 + i/100)$.

Аналогично проводятся вычисления для перегруженной (overdamped) пружины и кубической проблемы собственных значений для $n = 50$. Найденные собственные значения изображены на фиг. 1.



Фиг. 2. Число итераций.

Интересно посмотреть, сколько итераций нужно для каждого собственного значения. Столбцовые графики и средние числа итераций показаны на фиг. 2. Начальное значение для первого собственного значения, очевидно, выбрано не очень удачно, так как для сходимости требуется большое число итераций. Для кубической задачи большое число итераций возникает также для некоторых промежуточных собственных значений.

4. УЛУЧШЕНИЕ СХОДИМОСТИ

Причиной большого числа итерационных шагов при вычислении собственных значений в последних трех примерах является довольно плохая глобальная сходимость метода Ньютона. Локально метод Ньютона сходится к простому корню квадратично, обычно результат с машинной точностью получается за 3–4 итерации.

Пусть $f(z) = 0$ и λ_k – приближение к z. Ньютонова итерация заменяет f в окрестности λ_k на *линейную функцию* g такую, что $f(\lambda_k) = g(\lambda_k)$, $f'(\lambda_k) = g'(\lambda_k)$. Таким образом, g совпадает с отрезком ряда Тейлора $g(\lambda) = f(\lambda_k) + f'(\lambda_k)(\lambda - \lambda_k)$, а следующее приближение λ_{k+1} является нулем функции g.

Итерация Хэлли (Halley)

$$\lambda_{k+1} = \lambda_k - \frac{f(\lambda_k)}{f'(\lambda_k)} \frac{1}{1 - \frac{1}{2} \frac{f(\lambda_k) f''(\lambda_k)}{f'(\lambda_k)^2}} \tag{2}$$

заменяет f локально на *гиперболическую функцию*

$$g(\lambda) = \frac{a}{\lambda + b} + c$$

такую, что $f(\lambda_k) = g(\lambda_k)$, $f'(\lambda_k) = g'(\lambda_k)$ и $f''(\lambda_k) = g''(\lambda_k)$, следующее приближение λ_{k+1} является нулем функции g. Итерация Хэлли – метод третьего порядка, и значит, он сходится к простому корню кубически (см. [6]). Можно ожидать, что гиперболическая аппроксимация для f лучше с точки зрения глобальной сходимости.

5. РЕАЛИЗАЦИЯ ИТЕРАЦИИ ХЭЛЛИ

Нам нужна *вторая производная* определителя, более точно, нам нужно вычислять функцию

$$t(\lambda) = \frac{f(\lambda) f''(\lambda)}{f'^2(\lambda)}.$$

Заметим, что производная ньютоновской коррекции имеет вид

$$\frac{d}{dx}\left(\frac{f}{f'}\right) = \frac{f'^2 - ff''}{f'^2} = 1 - \frac{ff''}{f'^2}.$$

Отсюда

$$t = \frac{ff''}{f'^2} = 1 - \frac{d}{dx}\left(\frac{f}{f'}\right),$$

и чтобы получить $t(\lambda)$, нужно вычислять производную ньютоновской коррекции для нашей функции `deta`. Это делается алгоритмическим дифференцированием функции `deta`. Далее, функция `det2p` получает на вход матрицы A, A' и A'' и вычисляет ньютоновскую коррекцию f/f' и ее производную:

```
function [ffp,dffp] = det2p(A,Ap,App)
% DET2P computes Newton correction ffp = f/f'
%       and its derivative dffp = (f/f')'
n=length(A);
logfpp=0;                                % logfpp = log(f)''
logfp=0;                                 % log(f)'
for k=1:n
   [amax,kmax]=max(abs(A(k:n,k)));       % partial pivoting
   if amax==0                            % matrix singular
      ffp=0; dffp=0;return
   end
   kmax=kmax+k-1;
   if kmax~=k                            % interchange rows
     h=App(k,:); App(k,:)=App(kmax,:); App(kmax,:)=h;
     h=Ap(k,:); Ap(k,:)=Ap(kmax,:); Ap(kmax,:)=h;
     h=A(k,:); A(k,:)=A(kmax,:); A(kmax,:)=h;
   end
   logfpp=logfpp+(A(k,k)*App(k,k)-Ap(k,k)^2)/A(k,k)^2;
   logfp=logfp+Ap(k,k)/A(k,k);
   App(k+1:n,k)=(A(k,k)*App(k+1:n,k)-Ap(k+1:n,k)*Ap(k,k))/A(k,k)^2-…
        (Ap(k+1:n,k)*Ap(k,k)/A(k,k)^2+ …
        A(k+1:n,k)*App(k,k)/A(k,k)^2-…
        2* A(k+1:n,k)*Ap(k,k)^2/A(k,k)^3);

   Ap(k+1:n,k)=Ap(k+1:n,k)/A(k,k)-A(k+1:n,k)*Ap(k,k)/A(k,k)^2;
   A(k+1:n,k)=A(k+1:n,k)/A(k,k);         % elimination step

   App(k+1:n,k+1:n)=App(k+1:n,k+1:n) -…
        (App(k+1:n,k)*A(k,k+1:n) + Ap(k+1:n,k)*Ap(k,k+1:n) ) - …
        (Ap(k+1:n,k)*Ap(k,k+1:n) + A(k+1:n,k)*App(k,k+1:n));

   Ap(k+1:n,k+1:n)=Ap(k+1:n,k+1:n) - Ap(k+1:n,k)*A(k,k+1:n)-…
        A(k+1:n,k)*Ap(k,k+1:n);
   A(k+1:n,k+1:n)=A(k+1:n,k+1:n) - A(k+1:n,k)*A(k,k+1:n);
end
dffp=-logfpp/logfp^2; ffp=1/logfp;
```

6. ХЭЛЛИ–МЕХЛИ (HALLEY–MAEHLY)

Как и раньше, мы хотим подавить уже вычисленные собственные значения и снова рассматриваем

$$f_k(\lambda) : \frac{f(\lambda)}{p(\lambda)}, \quad p(\lambda) = (\lambda - \lambda_1) \cdots (\lambda - \lambda_k).$$

Теперь мы применяем итерации Хэлли к f_k:

$$\lambda_{\text{new}} = \lambda - \frac{f_k}{f_k'} \frac{1}{1 - \frac{1}{2} \frac{f_k f_k}{f_k'^2}},$$

и записываем итерацию в терминах f, f' и f''. Для ньютоновской коррекции f_k/f_k' используем уравнение (1). Для f_k''/f_k' выражаем сначала

$$f_k'' = \frac{d}{d\lambda}\left(\frac{f' - sf}{p}\right) = \frac{p(f'' - s'f - sf') - p'(f' - sf)}{p^2} =$$
$$= \frac{f'' - s'f - sf' - sf' + s^2 f}{p}, \quad p' = ps, \quad s = \sum_{j=1}^{k} \frac{1}{\lambda - \lambda_j}.$$

Затем, деля на f_k', получаем

$$\frac{f_k''}{f_k'} = \frac{f'' - s'f - 2sf' + s^2 f}{f' - sf} = \frac{\frac{f''}{f'} - s'\frac{f}{f'} - 2s + s^2 \frac{f}{f'}}{1 - s\frac{f}{f'}},$$

и после умножения на f_k/f_k' находим

$$t = \frac{f_k f_k''}{f_k'^2} = \frac{\frac{ff''}{f'^2} + (s^2 - s')\left(\frac{f}{f'}\right)^2 - 2s\frac{f}{f'}}{\left(1 - s\frac{f}{f'}\right)^2}. \tag{3}$$

Суммируя, реализуем итерацию Хэлли–Мехли, для этого необходимо следующее.

1. Вычислить ньютоновскую коррекцию для f_k:

$$\frac{f_k}{f_k'} = \frac{f}{f'} \frac{1}{1 - \frac{f}{f'}s}.$$

2. Вычислить $t(\lambda)$ для f_k в соответствии с уравнением (3).
3. Итерировать

$$\lambda_{\text{new}} = \lambda - \frac{f_k}{f_k'} \frac{1}{1 - \frac{1}{2}t}.$$

Мы решаем три нелинейные проблемы собственных значений с помощью метода Хэлли и сравниваем результаты с итерациями Ньютона (см. табл. 1). Глобальная сходимость и в самом деле улучшилась, число итераций уменьшилось.

7. ЛАГЕР И ОСТРОВСКИЙ

Еще один метод третьего порядка для вычисления нулей многочлена – это *метод Лагера*. В качестве аппроксимации многочлена f степени n он использует многочлен $g(\lambda) = a(\lambda - \lambda_1)(\lambda - \lambda_2)^{n-1}$. Параметры a, λ_1 и λ_2 определяются таким образом, что g интерполирует f вместе с производны-

Таблица 1. Сравнение методов Ньютона и Хэлли

Число итераций	Неперегруженная пружина	Перегруженная пружина	Кубическая задача
Ньютон:			
максимальное	128	275	90
среднее	11.4	20.9	11.3
Хэлли:			
максимальное	67	140	46
среднее	7	12.1	7.1

Таблица 2. Сравнение методов Хэлли, Лагера и Островского

Число итераций	Неперегруженная пружина	Перегруженная пружина	Кубическая задача
Хэлли:			
максимальное	67	140	46
среднее	7	12.1	7.1
Лагер:			
максимальное	18	36	16
среднее	5.3	6.6	5.2
Островский:			
максимальное	23	43	18
среднее	5.5	7.1	5.2

ми $f(\lambda_k) = g(\lambda_k)$, $f'(\lambda_k) = g'(\lambda_k)$, $f''(\lambda_k) = g''(\lambda_k)$. Следующее приближение – это нуль для g, ближайший к λ_k:

$$\lambda_{k+1} = \lambda_k - \frac{f(\lambda_k)}{f'(\lambda_k)} \frac{n}{1 + \sqrt{(n-1)^2 - n(n-1)\dfrac{f(\lambda_k)f''(\lambda_k)}{f'(\lambda_k)^2}}}. \tag{4}$$

Степень n является параметром метода Лагера. Пусть в (4) $n \to \infty$. Тогда мы получаем итерацию

$$\lambda_{k+1} = \lambda_k - \frac{f(\lambda_k)}{f'(\lambda_k)} \frac{1}{\sqrt{1 - \dfrac{f(\lambda_k)f''(\lambda_k)}{f'(\lambda_k)^2}}}, \tag{5}$$

которая называется *итерацией Островского с квадратным корнем*.

Заметим, что в итерационных методах Лагера и Островского так же, как и в методе Хэлли, нам нужно всего лишь два выражения:

$$\frac{f}{f'} \quad \text{и} \quad t = \frac{ff''}{f'^2}.$$

Поскольку метод Лагера задумывался как метод вычисления нулей многочлена, можно ожидать, что он будет хорошо работать для трех наших примеров. Сравнение трех методов в табл. 2 показывает, что это действительно так.

8. МЕТОДЫ ТРЕТЬЕГО ПОРЯДКА

Методы Хэлли, Лагера и Островского реализуют специальные случаи следующей теоремы.

Теорема 1 (методы третьего порядка, см. [6]). *Пусть s – простой нуль функции f, а G – любая функция такая, что*

$$G(0) = 1, \quad G'(0) = \frac{1}{2}, \quad |G''(0)| < \infty.$$

Тогда итерационный метод

$$x_{\text{new}} = x - \frac{f(x)}{f'(x)} G(t(x)), \quad t(x) = \frac{f(x)f''(x)}{f'(x)^2},$$

сходится к s по крайней мере кубически.

Примеры:

- формула Хэлли (Halley)

$$G(t) = \frac{1}{1 - \frac{1}{2}t} = 1 + \frac{1}{2}t + \frac{1}{4}t^2 + \frac{1}{8}t^3 + \ldots;$$

- формула Эйлера (Euler)

$$G(t) = \frac{2}{1 + \sqrt{1 - 2t}} = 1 + \frac{1}{2}t + \frac{1}{2}t^2 + \frac{5}{8}t^3 + \ldots;$$

- квадратичная обратная интерполяция

$$G(t) = 1 + \frac{1}{2}t;$$

- итерация Островского с квадратным корнем

$$G(t) = \frac{1}{\sqrt{1 - t}} = 1 + \frac{1}{2}t + \frac{3}{8}t^2 + \ldots;$$

- Лагер (Laguerre)

$$G(t) = \frac{n}{1 + \sqrt{(n-1)^2 - n(n-1)t}} 1 + \frac{1}{2}t + \frac{1}{8}\frac{3n-2}{n-1}t^2 + \ldots;$$

- семейство формул Хансен–Патрик (Hansen–Patrick) (см. [7])

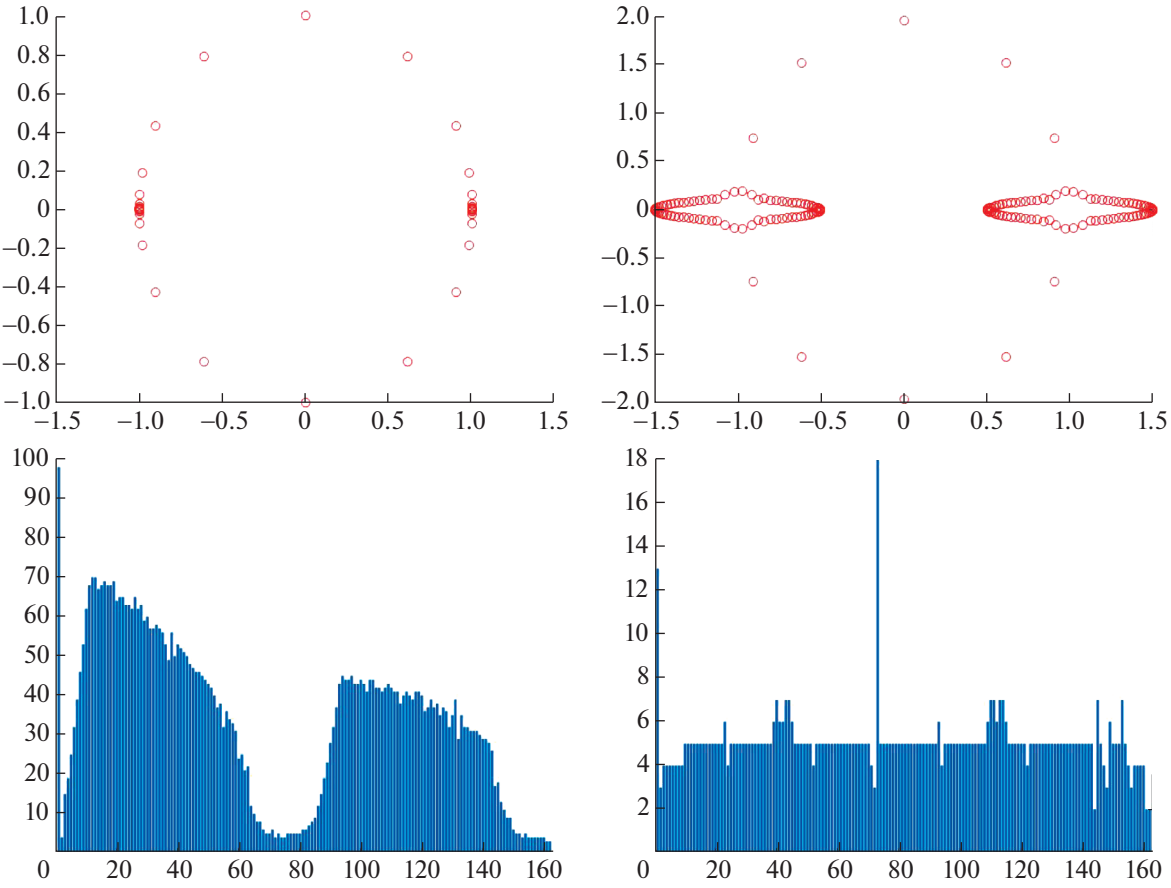
$$G(t) = \frac{\alpha + 1}{\alpha + \sqrt{1 - (\alpha + 1)t}} = 1 + \frac{1}{2}t + \frac{\alpha + 3}{8}t^2 + \ldots.$$

Заметим, что для применения этих итераций нам нужно вычислять лишь ньютоновскую коррекцию и $t = ff''/f'^2$.

9. NLEVP – КОЛЛЕКЦИЯ НЕЛИНЕЙНЫХ ПРОБЛЕМ СОБСТВЕННЫХ ЗНАЧЕНИЙ

В замечательной коллекции NLEVP нелинейных проблем собственных значений есть примеры всех типов матриц (см. [8] и [9]). (http://www.maths.manchester.ac.uk/our-research/research-groups/numerical-analysis-and-scientific-computing/numerical-analysis/software/nlevp/)

Используя метод Лагера, мы решили две квадратичные задачи `sign1` и `sign2` (плотные матрицы, $n = 81$). Результаты приведены на фиг. 3. Задача `sign1` имеет $2n = 162$ собственных значений на единичной окружности с двумя кластерами в точках ± 1. Из графиков следует, что сходимость к собственным значениям из этих двух кластеров медленная. Сходимость в задаче `sign2` намного лучше, так как ее собственные значения лучше разделены. MATLAB-инструменты для замера времени `tic`, `toc` показали, что на использованном автором ноутбуке задача `sign1` решалась 63.92 с, а задача `sign2` – 10.82 с.



Фиг. 3. Слева – `sign1`, справа – `sign2`.

10. НЕПОЛИНОМИАЛЬНАЯ ПРОБЛЕМА СОБСТВЕННЫХ ЗНАЧЕНИЙ

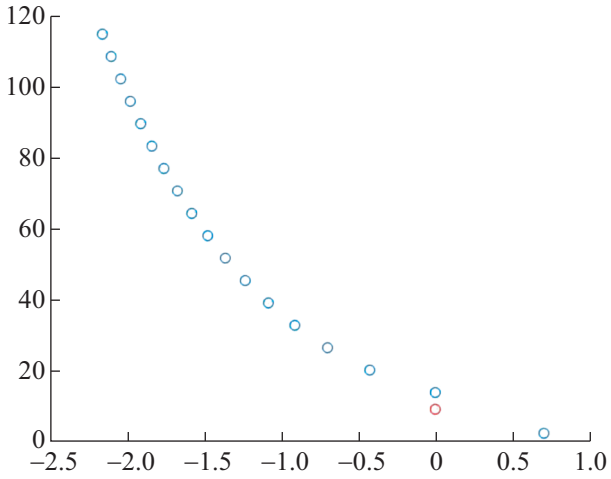
`TimeDelay` представляет собой неполиномиальную нелинейную проблему собственных значений из NLEVP-коллекции с 3×3-матрицей $A(\lambda)$:

$$A(\lambda) = -\lambda I + A_0 + A_1 e^{-\lambda}.$$

Об этой задаче пишут (см. [8]): "...характеристическое уравнение системы с запаздыванием по времени с единственной задержкой и постоянными коэффициентами. Задача имеет двойное непростое собственное значение $\lambda = 3\pi i$".

Нелинейное уравнение $\det A(\lambda) = 0$ имеет бесконечно много решений. Используя итерации Островского, мы можем, например, вычислить первые 20 решений и получить график фиг. 4. На мнимой оси получаем упомянутые выше двойные собственные значения (λ_2 и λ_3, фиг. 4) и также простое собственное значение $\lambda_4 = 4.5\pi i$. Собственное значение $\lambda_1 = 0.705244109106679 + 2.741466762205487i$ имеет положительную вещественную часть, остальные собственные значения имеют отрицательные вещественные части. Двойное собственное значение вычисляется с точностью, характерной для стандарта IEEE арифметики с плавающей точкой:

$$\lambda_1 = 0.705244109106679 + 2.741466762205487i,$$
$$\lambda_2 = 0.000000005149180 + 9.424777943433675i,$$
$$\lambda_3 = -0.000000007679198 + 9.424777969836999i,$$
$$\lambda_4 = -0.000000000000001 + 14.137166941154069i,$$
$$\lambda_5 = -0.422996397305027 + 20.485362607960255i,$$
$$\lambda_6 = -0.693701244038287 + 26.758000106609209i.$$



Фиг. 4. Нелинейная задача: пример с запаздыванием по времени.

11. ИСКЛЮЧЕНИЕ ГАУССА ДЛЯ ЛЕНТОЧНЫХ МАТРИЦ

Многие задачи из NLEVP-коллекции имеют ленточные матрицы (например, `beamsensitivity` с 7-ю или `pdde-stabiity` с 32-мя диагоналями). Определители таких матриц разумно вычислять по специальному алгоритму. MATLAB-функция для исключения Гаусса для ленточных матриц с частичным выбором описана в [3]. Если A имеет q нижних и p верхних диагоналей, то мы храним их как столбцы матрицы B. Для реализации частичного выбора добавляем q нулевых столбцов к матрице B:

$$A = \begin{bmatrix} x & x & 0 & 0 & 0 & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 & 0 & 0 \\ x & x & x & x & 0 & 0 & 0 & 0 \\ 0 & x & x & x & x & 0 & 0 & 0 \\ 0 & 0 & x & x & x & x & 0 & 0 \\ 0 & 0 & 0 & x & x & x & x & 0 \\ 0 & 0 & 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & 0 & 0 & x & x & x \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & x & x & 0 & 0 \\ 0 & x & x & x & 0 & 0 \\ x & x & x & x & 0 & 0 \\ x & x & x & x & 0 & 0 \\ x & x & x & x & 0 & 0 \\ x & x & x & x & 0 & 0 \\ x & x & x & x & 0 & 0 \\ x & x & x & 0 & 0 & 0 \end{bmatrix}.$$

Адаптируя функцию `det2p` к этой ленточной стуктуре, получаем функцию `det2pband`:

```
function [ffp,dffp]=det2pband(p,q,B,Bp,Bpp);
% DET2PBAND computes Newton-correction and derivative for a banded matrix
n=length(B); logfpp=0; logfp=0;
Bpp=[Bpp,zeros(n,q)];
Bp=[Bp,zeros(n,q)]; B=[B,zeros(n,q)];    % augment B with q columns
normb=norm(B,1);
for j=1:n
   maximum=0; kmax=j;                    % search pivot
   for k=j:min(j+q,n)
      if abs(B(k,j-k+q+1))>maximum,
         kmax=k; maximum=abs(B(k,j-k+q+1));
      end
   end
   if maximum<1e-14*normb;               % only small pivots
      ffp=0; dffp=0; return              % consider det=0
   end
```

```
   if j~=kmax                                  % interchange rows
      ind1=j-kmax+q+1:min(n,j+2*q+p-kmax+1);
      ind2=q+1:min(n,2*q+p+1);
      h=Bpp(kmax,ind1); Bpp(kmax,ind1)=Bpp(j,ind2); Bpp(j,ind2)=h;
      h=Bp(kmax,ind1); Bp(kmax,ind1)=Bp(j,ind2); Bp(j,ind2)=h;
      h=B(kmax,ind1); B(kmax,ind1)=B(j,ind2); B(j,ind2)=h;
   end

   logfpp=logfpp+(B(j,q+1)*Bpp(j,q+1)-Bp(j,q+1)^2)/B(j,q+1)^2;
   logfp=logfp+Bp(j,q+1)/B(j,q+1);
   for k=j+1:min(n,j+q)                        % elimination step
      ind3=j-k+q+1;
      Bpp(k,ind3)=(Bpp(k,ind3)*B(j,q+1)-Bp(j,q+1)*Bp(k,ind3))/B(j,q+1)^2 …
                 -(Bp(k,ind3)*Bp(j,q+1)+B(k,ind3)*Bpp(j,q+1))/B(j,q+1)^2 …
                 +2*Bp(j,q+1)^2*B(k,ind3)/B(j,q+1)^3;
      Bp(k,ind3)=(B(j,q+1)*Bp(k,ind3)-B(k,ind3)*Bp(j,q+1))/B(j,q+1)^2;;
      B(k,ind3)=B(k,ind3)/B(j,q+1);
   end
   for k=j+1:min(n,j+q)
      for l=j+1:min(n,j+p+q)
         ind4=l-k+q+1; ind5=j-k+q+1; ind6=l-j+q+1;
         Bpp(k,ind4)=Bpp(k,ind4)-Bpp(k,ind5)*B(j,ind6)
                     -Bp(k,ind5)*Bp(j,ind6)…
                    -Bp(k,ind5)*Bp(j,ind6)-B(k,ind5)*Bpp(j,ind6);
         Bp(k,ind4)=Bp(k,ind4)-Bp(k,ind5)*B(j,ind6)-B(k,ind5)*Bp(j,ind6);
         B(k,ind4)=B(k,ind4)-B(k,ind5)*B(j,ind6);
      end
   end
end
dffp=-logfpp/logfp^2; ffp=1/logfp;
```

Пример `beamsensitivity` дает квадратичную проблему собственных значений с 7-диагональной матрицей. Для $n = 200$ вычисляются 400 собственных значений по методу Лагера. Время измеряется MATLAB-функциями `tic`, `toc`. Если применить метод для плотной матрицы, то потребуется 83.85 с. Алгоритм с использованием ленточной структуры снижает время до 10.39 с.

12. ПРИМЕР СТРУНЫ В ВЯЗКОЙ СРЕДЕ

Хайям и др. пишут (см. [10]): “Стандартный подход к численному решению квадратичной задачи переводит $Q(\lambda) = \lambda^2 M + \lambda D + K$ в линейный полином $L(\lambda) = \lambda X + Y$ с матрицей удвоенного размера с сохранением спектра. Уравнение $L(\lambda)z = 0$ обычно решается QZ-алгоритмом для задач умеренного размера и крыловскими методами для больших разреженных задач. Обычный выбор L на практике – это первая сопровождающая форма вида

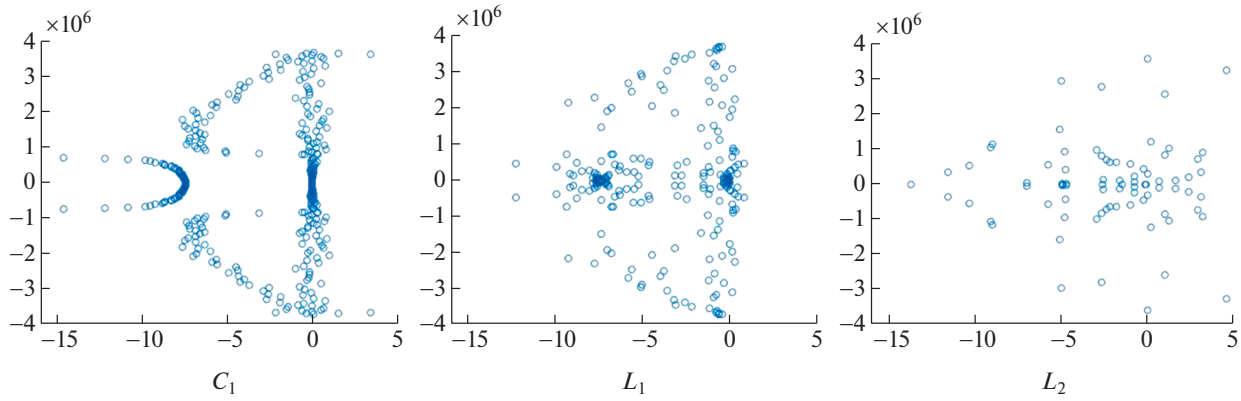
$$C_1(\lambda) = \lambda\begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} D & K \\ -I & 0 \end{bmatrix}.$$

Когда K и M невырожденные, соответственно, два пучка:

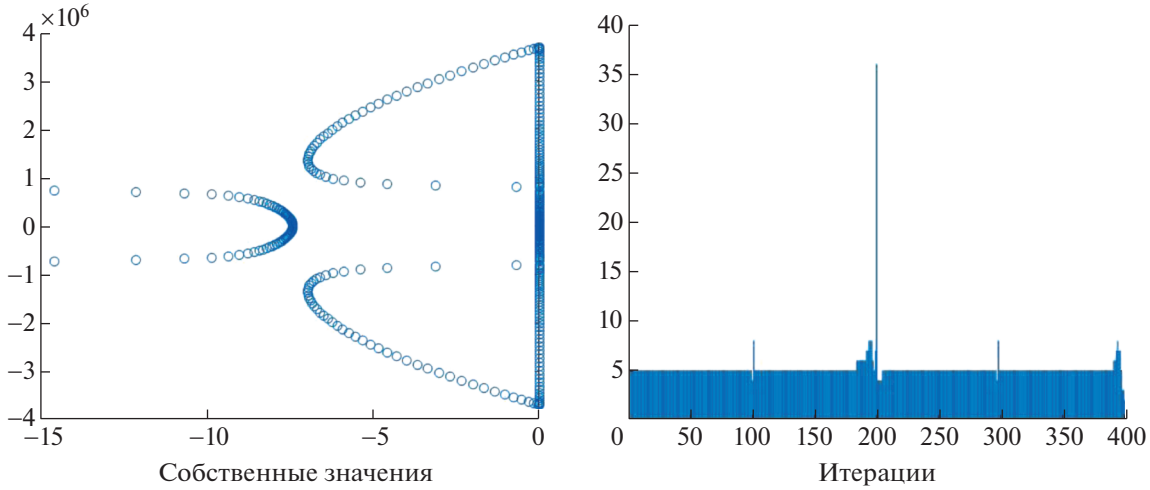
$$L_1(\lambda) = \lambda\begin{bmatrix} M & 0 \\ 0 & -K \end{bmatrix} + \begin{bmatrix} D & K \\ K & 0 \end{bmatrix}, \quad L_2(\lambda) = \lambda\begin{bmatrix} 0 & M \\ M & D \end{bmatrix} + \begin{bmatrix} -M & 0 \\ 0 & K \end{bmatrix},$$

являются другими возможными линеаризациями”.

При использовании этих линеаризаций (см. [10]) результаты решения обобщенной проблемы собственных значений с помощью MATLAB-функции `eig` довольно удручающие (фиг. 5).



Фиг. 5. Линеаризация без масштабирования.



Фиг. 6. Итерации Лагера для струны в вязкой среде.

Другие авторы (Fan, Lin, Van Dooren) показали в [11], как плохая обусловленность линеаризованных задач может быть исправлена масштабированием. В [10] объясняется, почему преобразованные системы без масштабирования являются настолько плохо обусловленными.

Ситуация напоминает мне старый пример Джима Уилкинсона, показывающий, что сведение проблемы собственных значений к задаче вычисления корней характеристического многочлена является не лучшим подходом к решению задачи, потому что изменяет ее обусловленность радикальным образом.

Однако при прямом решении уравнения $\det A(\lambda) = 0$ с помощью одного из наших методов, например, итераций Лагера, получаем корректные результаты без необходимости применять масштабирование (фиг. 6).

13. ВЫВОДЫ

Мы показали, как реализуются итерационные методы третьего порядка для решения $f(\lambda) = \det A(\lambda) = 0$ с использованием автоматического (алгоритмического) дифференцирования. Эта техника дает точные производные, так как при вычислении определителей методом Гаусса используются лишь четыре арифметические операции.

Поскольку мы работаем непосредственно с исходной задачей, нет преобразований, которые могли бы изменить обусловленность задачи.

Кубическую сходимость можно получить с помощью многошаговых итераций, которые обходятся без вторых производных. Однако, используя только точечные итерации, мы получаем алгоритм `det2p`, в котором вычисляются ньютоновская коррекция и величина $t = ff''/f'^2$, содержащая нужные нам производные.

Вычисление вторых производных дорого. Для полной $n \times n$-матрицы нужно $\sim n^3$ операций на одну итерацию. Тем не менее, благодаря мощным процессорам наших компьютеров, мы можем решать нелинейные проблемы собственных значений умеренных размеров. Ситуация намного более благоприятна в случае ленточных матриц, для которых одна итерация требует лишь $\sim n$ операций.

Автор выражает благодарность Zhong-Zhi Bai и Yu-Mei Huang – организаторам Мемориального семинара в Ланжоу, посвященного Джину Голубу (Lanzhou, 2019). Приглашение участвовать в этой конференции дало особый стимул для разработки алгоритмов, обсуждаемых в этой статье.

СПИСОК ЛИТЕРАТУРЫ

1. *Arbenz P., Gander W.* Solving nonlinear eigenvalue problems by algorithmic differentiation // Computing. 1986. V. 36. P. 205–215.
2. *Gander W.* Zeros of determinants of λ-matrices // Matrix Methods: Theory, Algorithms and Applications. Dedicated to the Memory of Gene Golub. 2010. P. 238–246.
3. *Gander W., Gander Martin J., Kwok F.* Scientific Computing, an Introduction Using MAPLE and MATLAB. Switzerland: Springer, 2014.
4. *Maehly H.J.* Zur iterativen Auflösung algebraischer Gleichunge // Zeitschrift für angewandte Mathematik und Physik. 1954. P. 260–263.
5. *Tisseur F., Meerbergen K.* The quadratic eigenvalue problem // SIAM Rev. 2001. V. 43. P. 234–286.
6. *Gander W.* On Halley's iteration method // Am. Math. Month. 1985. V. 92. № 2. P. 131–134.
7. *Hansen E., Patrick M.* A family of root finding methods // Numer. Math. 1977. V. 27. P. 257–269.
8. *Betcke T., Higham N.J., Mehrmann V., Schröder C., Tisseur F.* NLEVP: A collection of nonlinear eigenvalue problems // ACM Trans. Math. Softw. 2013. V. 39. № 2. P. 1–28.
9. *Betcke T., Higham N.J., Mehrmann V., Schröder C., Tisseur F.* A collection of nonlinear eigenvalue problems. Users' guide // MIMS EPrint 2011.117, Manchester Inst. Math. Sci., Univer. of Manchester, UK, 2011.
10. *Higham N.J., Mackey D.S., Tisseur F., Garvey S.D.* Scaling, sensitivity and stability in the numerical solution of quadratic eigenvalue problems // Int. J. Numer. Meth. Engng. 2008. V. 73. P. 344–360.
11. *Fan H.-Y., Lin W.-W., Van Dooren P.* Normwise scaling of second order polynomial matrices // SIAM J. Matrix Anal. Appl. 2004. V. 26. P. 252–256.