СООБЩЕНИЯ
ОБЪЕДИНЕННОГО
ИНСТИТУТА
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ

Р11 - 87 - 332

И.В. Пузынин, Т.П. Пузынина, Т.А. Стриж

SLIPH4 – ПРОГРАММА ДЛЯ ЧИСЛЕННОГО РЕШЕНИЯ ЗАДАЧИ ШТУРМА–ЛИУВИЛЛЯ

1987

I. Введение

SLIPH4 – комплекс программ, являющийся развитием программы SLIP1 [1], предназначенный для численного решения задачи Штурма–Лиувилля в виде

$$\varphi^{(1)}(\lambda, y) \equiv [\frac{d^2}{dx^2} + 2p(x)\frac{d}{dx} + q(x) - \lambda r(x)]y(x) = 0;\ a < x < b; \qquad (1a)$$

$$\varphi^{(j)}(\lambda, y) \equiv [d_{j-1}(\lambda, x)\frac{d}{dx} + f_{j-1}(\lambda, x)]y(x)|_{x=l_j} = 0;\ j = 2, 3;\ l_2 = a;\ l_3 = b. \qquad (1b, c)$$

Здесь $p(x), q(x), r(x), d_{j-1}(\lambda, x), f_{j-1}(\lambda, x)$ – заданные функции, обеспечивающие существование нетривиальных собственных решений $y = y_n(x)$ граничной задачи (1), которым соответствуют собственные значения параметра $\lambda = \lambda_n$. Функции d_{j-1}, f_{j-1} непрерывно дифференцируемы по λ и $d^2_{j-1} + f^2_{j-1} > 0$.

Программа SLIPH4 по заданному начальному приближению $\{\lambda^0_n, y^0_n\}$ вычисляет частичное решение $\{\lambda^*_n, y^*_n(x)\}$ задачи (1), где n - фиксировано, $x \in \omega_h$, ω_h – равномерная разностная сетка на отрезке $a \leq x \leq b$ с шагом h . Точность вычисления является величиной $O(h^4)$ и совпадает с точностью используемой разностной аппроксимации задачи (1)[2].

Основные особенности комплекса SLIPH4 по сравнению с программой SLIP1 заключаются в следующем.

1. В комплекс включена программа вычисления начальных приближений $\{\lambda^0_n, y^0_n(x)\}$ с помощью модифицированного алгоритма [3], основанного на методе Ньютона для нахождения корней полинома с удалением вычисленных корней [4]. Это позволяет последовательно вычислять собственные значения λ^*_n и соответствующие им собственные функции $y^*_n(x)$ в достаточно широкой части спектра задачи (1).

2. Используется разностная аппроксимация на сетке ω_h для граничной задачи (1) с точностью $O(h^4)$. Она построена таким образом, что почти во всех внутренних узлах сетки является трехточечной. Это дает возможность решать задачу (1) с точностью $O(h^4)$ без существенного изменения структуры программы SLIP1 .

3. В комплекс включена программа, реализующая модифицированную ньютоновскую схему с фиксацией сдвига по собственному значению и с дополнительной

ортогонализацией найденного приближенного собственного решения. Эта программа позволяет одновременно решить задачи нахождения начального приближения и его уточнения в рамках единой вычислительной схемы.

Работа комплекса SLIPH4 проверена на достаточно широком наборе тестовых задач /5/. Кроме того, комплекс успешно использовался при решении разнообразных физических задач /6/.

Ниже дается краткое описание алгоритмов, используемых в программах комплекса SLIPH4, и обсуждаются вопросы анализа точности результатов. Приведено описание программ и их параметров, рассмотрены варианты использования комплекса.

2. Алгоритмы

2.1. Алгоритм вычисления начального приближения

Алгоритм вычисления начального приближения основан на решении двух задач Коши. Для первой задачи Коши ставятся начальные условия в точке $x = a$, где условие (1b) дополняется условием на производную. Аналогичное условие для второй задачи ставится в точке $x = b$.

Для приближенного решения этих задач применяется трехточечная разностная схема второго порядка. Рекуррентные соотношения, полученные с помощью разностных схем, позволяют последовательно находить значения функции y в узлах сетки ω_h при заданном значении λ.

Если граничные условия (1b,c) полиномиально зависят от λ , что выполняется для достаточно широкого круга задач, то упомянутые выше рекуррентные соотношения также определяют полиномы от λ , степени которых зависят от числа узлов сетки. Задача об отыскании собственного значения сводится тогда к отысканию корней полинома, появляющегося из условия равенства логарифмических производных для решений двух задач Коши в некоторой внутренней точке x_m сетки ω_h

$$\frac{y_n'(x_m)_{\text{лев}}}{y_n(x_m)_{\text{лев}}} = \frac{y_n'(x_m)_{\text{прав}}}{y_n(x_m)_{\text{прав}}}, \qquad (2.1.1)$$

$$T(\lambda) \equiv y_n'(x_m)_{\text{лев}} * y_n(x_m)_{\text{прав}} - y_n(x_m)_{\text{лев}} * y_n'(x_m)_{\text{прав}} = 0. \tag{2.1.2}$$

Корни полинома $T(\lambda)$ находятся по методу Ньютона с исключением уже найденных корней (k– номер итерации)

$$\lambda_n^{k+1} = \lambda_n^k - \frac{T(\lambda_n^k)}{T_\lambda'(\lambda_n^k) - T(\lambda_n^k)\sum\limits_{j=1}^{n-1}\frac{1}{\lambda_n^k - \lambda_j}}. \tag{2.1.3}$$

Итерационный процесс (2.1.3) прекращается при одновременном выполнении двух условий:

$$|\lambda_n^{k+1} - \lambda_n^k| \le \varepsilon_T \qquad \text{и} \qquad |T(\lambda_n^{k+1})| \le \varepsilon_T, \tag{2.1.4}$$

где $\varepsilon_T > 0$ – заданное малое число. После определения λ_n с требуемой точностью мы одновременно получаем и приближенное значение соответствующей собственной функции $y_n(x_i)$.

2.2. Алгоритм уточнения начального приближения

Заданное или вычисленное с помощью алгоритма п.2.1 начальное приближение $\lambda^0, y^0(x)$ уточняется с помощью итерационной ньютоновской процедуры для задачи Штурма–Лиувилля /1/.

При этом задача (1) дополняется условием нормировки

$$\varphi^{(4)}(\lambda, y) \equiv \int\limits_a^b y^2(x)\,dx + c^2\int\limits_b^\infty \psi^2(\lambda, x)\,dx - 1 = 0, \tag{2.2.1}$$

которое дает возможность учитывать особенности тех задач, в которых решение ищется на полубесконечном интервале $[a, \infty)$. Здесь $\psi(\lambda, x)$ – асимптотическое выражение для искомой собственной функции $y(x)$ при $x \to \infty$, c – константа сшивки решения в некоторой достаточно удаленной точке $x = b$,

$$y(b) = c\psi(\lambda, b). \tag{2.2.2}$$

Если задача рассматривается на конечном отрезке $[a, b]$, следует положить $c = 0.$ Один шаг с номером k итерационного процесса заключается в следующем:

1. Решить краевую задачу для функции ϑ_k

$$\varphi^{(1)}(\lambda_k, \vartheta_k) = s_{2k}(x), \tag{2.2.3}$$

$$\varphi^{(2)}(\lambda_k, \vartheta_k) = a_{2k},$$
$$\varphi^{(3)}(\lambda_k, \vartheta_k) = b_{2k},$$

где

$$s_{2k}(x) = r(x)y_k(x),$$

$$a_{2k} = -[d'_{1\lambda}y'_k(a) + f'_{1\lambda}y_k(a)], \tag{2.2.4}$$

$$b_{2k} = -[d'_{2\lambda}y'_k(b) + f'_{2\lambda}y_k(b)]$$

при известных с предыдущего шага $(k-1)$ значениях $\lambda_k, y_k(x)$.

2. Вычислить поправку μ_k для собственного значения λ_k с помощью выражения

$$\mu_k = \frac{\int\limits_a^b y_k^2(x)dx - c^2\int\limits_b^\infty \psi^2(\lambda_k, x)dx + 1}{2(\int\limits_a^b y_k(x)\vartheta_k(x)dx + c^2\int\limits_b^\infty \psi(\lambda_k, x)\psi'_\lambda(\lambda_k, x)dx)}. \tag{2.2.5}$$

3. Получить следующее приближение:

$$\lambda_{k+1} = \lambda_k + \tau_k\mu_k, \tag{2.2.6}$$

$$y_{k+1}(x) = (1-\tau_k)y_k(x) + \tau_k\mu_k\upsilon_k(x).$$

Итерационный параметр τ_k вычисляется с помощью алгоритмов, приведенных ниже. Итерации прекращаются при выполнении условия

$$\delta_k < \varepsilon, \tag{2.2.7}$$

где невязка δ_k может определяться как

$$\delta_k = \max_j \max_{x\in[a,b]} |\varphi^{(j)}(\lambda_k, y_k(x))|, \qquad j = 1, 2, 3, \tag{2.2.8}$$

или

$$\delta_k = [\int\limits_a^b \varphi^{(1)^2}(\lambda_k, y_k(x))dx]^{1/2}, \tag{2.2.9}$$

$\varepsilon > 0$ – заданное малое число.

2.3. Модифицированный алгоритм

Алгоритм, изложенный в п.2.2., может быть изменен таким образом, что будет одновременно и находить, и уточнять приближенное решение задачи (1), (2.2.1). За основу взята модифицированная ньютоновская схема, в которой на каждом шаге фиксируется сдвиг $\bar\lambda$ по собственному значению.

В этом случае на каждом шаге (с номером k) итерационного процесса для нахождения решения $\lambda_n^*, y_n^*(x)$ требуется выполнить следующее:

1. Решить две краевые задачи, отличающиеся правыми частями, относительно функций $\vartheta_k^{(1)}(x)$ и $\vartheta_k^{(2)}(x)$:

$$\begin{gathered} \varphi^{(1)}(\bar\lambda, \vartheta_k^{(i)}) = s_k^{(i)}(x), \\ \varphi^{(2)}(\bar\lambda, \vartheta_k^{(i)}) = a_{ik}, \\ \varphi^{(3)}(\bar\lambda, \vartheta_k^{(i)}) = b_{ik}, \qquad i = 1, 2, \end{gathered} \tag{2.3.1}$$

где

$$\begin{gathered} s_k^{(1)} = -\varphi^{(1)}(\lambda_k, y_k), \qquad s_k^{(2)} = r(x)y_k(x), \\ a_{1k} = -\varphi^{(2)}(\lambda_k, y_k(a)), \qquad a_{2k} = -[d'_{1\lambda}(\bar\lambda)y'_k(a) + f'_{1\lambda}(\bar\lambda)y_k(a)], \\ b_{1k}^{(1)} = -\varphi^{(3)}(\lambda_k, y_k(b)), \qquad b_{2k} = -[d'_{2\lambda}(\bar\lambda)y'_k(b) + f'_{2\lambda}(\bar\lambda)y_k(b)] \end{gathered} \tag{2.3.2}$$

при известных с предыдущего шага (с номером $k-1$) значениях λ_k, $y_k(x)$.

2 . Вычислить поправку μ_k к собственному значению по формуле:

$$\mu_k = \frac{1 - \int\limits_a^b y_k^2(x)dx - c^2 \int\limits_b^\infty \psi^2(\lambda_k, x)dx - 2\int\limits_a^b y_k(x)\vartheta_k^{(1)}(x)dx}{2(\int\limits_a^b y_k(x)\vartheta_k^{(2)}(x)dx + c^2 \int\limits_b^\infty \psi(\bar{\lambda}, x)\psi'_\lambda(\bar{\lambda}, x)dx)}. \qquad (2.3.3)$$

3. Найти следующие приближения:

$$\lambda_{k+1} = \lambda_k + \tau_k \mu_k, \qquad (2.3.4)$$

$$y_{k+1}(x) = y_k(x) + \tau_k(\vartheta_k^{(1)}(x) + \mu_k \vartheta_k^{(2)}(x)).$$

4. Для приближения $y_{k+1}(x)$ к собственной функции $y_n(x)$ выполнить ортогонализацию по отношению ко всем найденным собственным функциям $y_m^*(x)$, $m = 0, 1, \ldots, n-1$ по формуле:

$$u_{k+1}(x) = y_{k+1}(x) - \sum_{m=0}^{n-1} y_m^*(x) \int\limits_a^b y_{k+1}(x) y_m^*(x)dx. \qquad (2.3.5)$$

5. Нормировать приближенную собственную функцию

$$y_{k+1}(x) = u_{k+1}(x)[\int\limits_a^b u_{k+1}^2(x)dx]^{-1/2}. \qquad (2.3.6)$$

Изложенная процедура является обобщением метода обратных итераций со сдвигом [/7/]. Она обеспечивает при довольно грубых приближениях к собственной функции сходимость к решению $\{\lambda_n^*, y_n^*(x)\}$, где λ_n^* – ближайшее к сдвигу $\bar{\lambda}$ собственное значение. Операции (2.3.5), (2.3.6) позволяют дополнительно подавлять вычислительные погрешности.

Изменение сдвига $\bar{\lambda}$ дает возможность сходиться к решению с нужным номером n. В качестве управления изменением сдвига можно принять следующую процедуру. Дня нахождения решения $\{\lambda_0^*, y_0^*(x)\}$ в качестве $\bar{\lambda}$ можно задать соответствующую границу спектра задачи (1). Далее в качестве сдвига задавать величину

$$\bar{\lambda} = \lambda_m^* + \varepsilon, \qquad m = 0, 1, \ldots, \qquad (2.3.7)$$

где λ_m^* – вычисленное собственное значение, $\varepsilon > 0$ – параметр регуляризации.

Отметим, что в данной процедуре большую роль играет задание итерационного параметра τ_k.

2.4. Дискретное представление

Решение краевых задач (2.2.3), (2.3.1) и (2.3.2) осуществляется с помощью метода конечных разностей на равномерной сетке ω_h. Эта задача аппроксимируется с помощью разностных формул с точностью аппроксимации $O(h^4)$. Как и в методе Нумерова /8/, при выводе разностных формул используется прием повышения порядка точности аппроксимации на решении уравнения.

Это дало возможность построить трехточечные разностные формулы с точностью аппроксимации $O(h^4)$ в узлах сетки с номерами $i = 3, \ldots, N-2$.

В приграничных ($i = 2, N-1$) и граничных ($i = 1, N$) узлах используются пятиточечные схемы того же порядка точности.

Формулы строятся так, чтобы коэффициенты уравнения (1) использовались только во внутренних узлах сетки ω_h.

Это важно в тех задачах, где коэффициенты уравнения в граничных точках имеют особенности.

В случае $p(x) \equiv 0$ выведенная разностная схема совпадает со схемой Нумерова.

Интегрирование в формулах (2.2.5), (2.2.9) и (2.3.3) выполняется с помощью квадратурной формулы Симпсона точности порядка h^4.

Полученные дискретные краевые задачи решаются методом прогонки.

2.5. Алгоритмы вычисления τ_k

В итерационных процедурах, описанных в пунктах 2.2 и 2.3, важную роль играет итерационный параметр τ_k . Выбор этого параметра позволяет регулировать ход итерационного процесса. При этом изменение параметра τ_k связано с изменением невязки δ_k в ходе итераций. Ниже приводятся следующие алгоритмы выбора τ_k, в которых τ_0 – некоторое заданное значение, $0 \leq \tau_0 \leq 1$.

$$1. \qquad \tau_k \equiv \tau_0. \tag{2.5.1}$$

Этот алгоритм при достаточно малом τ_0 обычно применяется при плохих начальных приближениях, с целью проверить возможность сходимости от этих приближений. Сходимость при этом очень медленная. При $\tau_0 \equiv 1$ получается классическая схема Ньютона.

$$2. \qquad \tau_k = \begin{cases} \min(1, 2\tau_{k-1}), \text{если} & \delta_k < \delta_{k-1}, \\ \max(\tau_0, \tau_{k-1}/2), \text{если} & \delta_k \geq \delta_{k-1}, \end{cases} \tag{2.5.2}$$

где δ_k определяется по формуле (2.2.8). Этот алгоритм аналогичен широко распространенному способу выбора шага интегрирования в стандартных программах решения задачи Коши, вычисления интегралов и т.п.[/9/].

Алгоритм рекомендуется применять при хороших начальных приближениях. Он обеспечивает быструю сходимость, однако не всегда устойчив.

$$3. \qquad \tau_k = \begin{cases} \min(1, \tau_{k-1}\dfrac{\delta_{k-1}}{\delta_k}), \text{если} & \delta_k < \delta_{k-1}, \\ \max(\tau_0, \tau_{k-1}\dfrac{\delta_{k-1}}{\delta_k}), \text{если} & \delta_k \geq \delta_{k-1} \end{cases} \tag{2.5.3}$$

Здесь δ_k тоже вычисляется по формуле (2.2.8). Этот алгоритм более устойчив и обеспечивает сходимость в достаточно широкой области начальных приближений.

$$4. \qquad \tau_k = \frac{\delta_{k-1}}{\delta_{k-1} + \delta_k(1)}, \tag{2.5.4}$$

где $\delta_k(1)$ – невязка на k -той итерации для $\tau_k = 1$. Величина δ_k вычисляется по формуле (2.2.9). Это алгоритм оптимального выбора τ_k , предложенный в работе [/10/]. Он основан на квадратичной аппроксимации зависимости δ от τ . При итерациях он должен обеспечить минимум невязки на каждом шаге.

5. На равномерной сетке ω_τ отрезка $[0,1]$ с шагом $\triangle\tau$ вычисляется последовательность невязок δ^i по формуле (2.2.8) и выбирается такое значение τ_k , которому соответствует минимальная невязка. Этот алгоритм более общий, чем в п.4, хотя и требует большего объема вычислений. Точность нахождения оптимального шага τ_k, обеспечивающего минимум невязки на каждом шаге, зависит от выбора сетки ω_τ.

Эту сетку можно выбрать таким образом, чтобы точность нахождения τ и быстродействие алгоритма оптимально сочетались.

Алгоритмы 1, 2, 3 были реализованы в SLIP1 и прошли проверку на широком круге задач. Алгоритмы 4, 5 использовались при решении задач /6/.

2.6. Точность вычислительной схемы

Пусть $z_n^* = \{\lambda_n^*, y_n^*(x)\}$ – решение задачи (1), существование которого предполагается.

В результате вычислений по изложенной схеме за конечное число итераций k , т.е. при выполнении условия $\delta_k < \varepsilon$ (δ_k - невязка (2.2.8) в сеточной C - норме), получается приближенное решение $z_{nh}^k = \{\lambda_{nh}^k, y_{nh}^k(x)\}$.

Пусть z_{nh} - точное разностное решение. Тогда в сеточной C -норме можно написать следующую оценку:

$$\| z_n^* - z_{nh}^k \| \leq \| z_n^* - z_{nh} \| + \| z_{nh} - z_{nh}^k \| . \qquad (2.6.1)$$

Для первого слагаемого в правой части (2.6.1) имеется следующая оценка /11/:

$$\| z_n^* - z_{nh} \| \leq C_n h^4. \qquad (2.6.2)$$

Второе слагаемое оценивается как /12/

$$\| z_{nh} - z_{nh}^k \| \leq B_n \delta_k. \qquad (2.6.3)$$

Здесь $B_n > 0,\ C_n > 0$ – константы.

Когда $B_n\delta_k << C_n h^4$, что выполняется при задании малого ε , точность полученного разностного решения близка к теоретической оценке

$$\| z_n^* - z_{nh}^k \| \approx C_n h^4. \qquad (2.6.4)$$

Ее можно детально исследовать путем расчетов на последовательности сгущающихся сеток. При этом возможно уточнение разностного решения /11/.

В случае, если исходная задача является сингулярной, в оценку (2.6.1) добавляется слагаемое, характеризующее ошибку аппроксимации сингулярных граничных условий условиями (1b,c) и (2.2.1).

Как правило, оценить эту погрешность можно путем проведения последовательных расчетов на расширяющихся интервалах $[a, b]$.

Все эти факторы следует иметь в виду при задании параметров вычислительной схемы и при оценке полученных результатов.

3. Описание программ комплекса

Комплекс SLIPH4 состоит из набора подпрограмм, которые можно объединить в три группы.

К первой группе относятся программы общего назначения. В них определяются величины, необходимые для работы остальных подпрограмм. Некоторые из них составляются пользователем комплекса.

Вторая группа программ обеспечивает нахождение начального приближения по алгоритму, изложенному в п.2.1 .

В третью группу входят подпрограммы, осуществляющие уточнение начального приближения (см. п.2.2 и п.2.3).

Каждая из этих групп подпрограмм может быть использована самостоятельно.

На рисунке приведена схема комплекса программ SLIPH4 .

MAIN
- CALL PQRX
- CALL MINMAX
- CALL NEWTON
  - CALL TTP
  - CALL SIGMA
- CALL ZEROS
- CALL SLIPH4
  - CALL AKBK
    - CALL D1F1
    - CALL D2F2
  - CALL SKI
    - CALL DYH4
    - CALL D2YH4
  - CALL DELTA
  - CALL PROGRV
  - CALL SIMPYV
  - CALL ASIM
  - CALL TAUKA
    - CALL DELTA
- CALL DOPORT
- CALL UNORM
- CALL ZEROS

Главная программа MAIN и подпрограммы PQRX, D1F1, D2F2 и ASIM составляются пользователем комплекса с учетом особенностей конкретной задачи. Часть подпрограмм, таких как TTP, AKBK, SIGMA, DELTA, PROGRV, SIMPYV, SKI и TAUKA , являются внутренними программами комплекса и пользователю не доступны. Ниже дается их краткое описание. Подробно описаны обращения и параметры подпрограмм, доступных пользователю.

3.1. Подпрограммы общего назначения PQRX, MINMAX, ZEROS, UNORM

Подпрограмма PQRX предназначена для заполнения массивов P , Q , R таблицами значений коэффициентов $p(x), q(x), r(x)$ уравнения (1) в узлах x_i сетки ω_h . Значения узлов x_i помещаются в массив X .

Обращение:

CALL PQRX (A,B,N,P,Q,R,X,XMIN,XMAX,XW,M) ,

где A, B – левая и правая границы отрезка изменения x ,

N – количество узлов равномерной сетки ω_h ,

P, Q, R, X – массивы размерности N , которые должны быть описаны в вызывающей программе,

XMIN, XMAX – левая и правая границы интервала, на котором ищутся начальные приближения собственных значений,

XW и M – точка сшивки (2.1.1) и ее номер в массиве X(N).

В подпрограмме используется ключ LQ из общего блока

COMMON/SWITCH/LINFUN, LOUT, LQ, LEVZ .

Если пользователь вычисляет в подпрограмме значения XMIN , XMAX , XW и M сам, то следует присвоить LQ значение 0 (LQ=0).

Подпрограмма MINMAX по ключу LQ$\neq$ 0 из общего блока /SWITCH/ вычисляет значения XMIN=$minQ(x)$ и XMAX=$maxQ(x)$ для $x \in \omega_h$, определяет точку сшивки XW как Q(XW)=XMAX и M – ее номер в массиве X . M должно удовлетворять условию

$$4 \leq M \leq N-3. \tag{3.1}$$

Обращение:

CALL MINMAX (N,X,Q,XMIN,XMAX,XW,M).

Если LQ $\neq 0$, то после нахождения XW и M проверяется условие (3.1). Выполнение его ведет к возврату в вызывающую программу. При невыполнении условия (3.1) величине M присваивается ближайшее из двух значений 4 или N-3, определяется новое значение XW=X(M), печатается сообщение о старом и новом значениях точки сшивки и происходит выход из подпрограммы.

Подпрограмма UNORM нормирует функцию Y .

Вычисляется новое значение $y = y/\sqrt{\int\limits_a^b y^2(x)dx}$.

Обращение:

CALL UNORM(N,H,Y) ,

где H – шаг равномерной сетки ω_h,

Y – массив размерности N , описанный в главной программе.

Подпрограмма ZEROS определяет число изменений знака в таблице значений функции Y . Оно засылается в NZ и отождествляется с числом нулей функции Y.

Обращение:

CALL ZEROS(N,Y,NZ),

где Y – массив размерности N , описанный в главной программе.

3.2. Подпрограммы вычисления начального приближения NEWTON, TTP и SIGMA

Подпрограмма NEWTON определяет начальные приближения задачи (1) по алгоритму, изложенному в п.2.1.

Обращение:

CALL NEWTON(A,B,P,Q,R,N,XMIN,XMAX,XW,M,EV,X,Y,DX,EPS,
ITMIN,ITMAX,LSTEP,MMAX,NIS,NNEC,NEVMAX) ,

где параметры A, B, P, Q, R, N, XMIN, XMAX, XW, M, X описаны в п.3.1; параметры DX, EPS, ITMIN, ITMAX, LSTEP, NIS, NNEC, NEVMAX задаются пользователем. Результат работы программы записывается в массивы Y и EV , где

EV – массив корней полинома (2.1.2) размерности NEVMAX,

Y – массив размерности N , содержащий начальное приближение к решению задачи (1) для последнего найденного корня;

EPS – значение малой величины ε_T из формулы (2.1.4);

ITMIN – число итераций процесса (2.1.3) без проверки условия (2.1.4);

ITMAX – максимально допустимое число итераций. Если точность после этого не достигнута, выдается сообщение, значение найденного приближения, его точность и значение полинома T (2.1.2);

LSTEP – ключ, задающий направление поиска корня на отрезке [XMIN , XMAX] . При LSTEP = 0 поиск ведется от XMIN , при LSTEP=1 – от XMAX.

DX – заданная величина. Поиск корней ведется на отрезке [XMIN-DX, XMAX+DX].

MMAX – после завершения работы программы NEWTON этому параметру присваивается значение, равное количеству найденных корней полинома T.

NIS – количество уже найденных каким-то образом корней полинома (собственных значений). Их значения до обращения к этой подпрограмме должны быть присвоены первым NIS элементам массива EV.

NNEC – номер корня, до которого включительно надо найти все корни полинома, начиная с (NIS+1)-го, и определить функцию Y.

NEVMAX – максимально допустимое число корней NEVMAX $\geq$NNEC.

В подпрограмме используется ключ LOUT из общего блока /SWITCH/ .

Если LOUT = 0 , происходит проверка найденного значения корня на нахождение его в интервале (XMIN-DX , XMAX+DX). Если LOUT$\neq$ 0 , проверка не происходит.

Общий блок:

COMMON/CMINMA/CMIN,CMAX ,

заполняемый пользователем в главной программе, задает диапазон чисел для конкретной ЭВМ (на CDC-6500 CMIN = 10^{-100} , CMAX=10^{+100}).

Подпрограмма NEWTON обращается к внутренней подпрограмме TTP и функции SIGMA .

Подпрограмма TTP вычисляет значение полинома $T(\lambda)$ из (2.1.2), его производной $T^{'}(\lambda)$ и функции Y .

Функция SIGMA вычисляет значение суммы из выражения (2.1.3):

$$\sum_{j=1}^{n-1} \frac{1}{\lambda_n^k - \lambda_j} .$$

3.3. Подпрограммы уточнения начального приближения SLIPH4 , AKBK, DELTA, PROGRV, SIMPYV, SKI, TAUKA, D1F1, D2F2

Подпрограмма SLIPH4 организует итерационный процесс, описанный в п.2.2 и п.2.3.

Обращение:

CALL SLIPH4(A,B,N,NMAX,P,Q,R,EV0,Y0,LAS,YS,EPS,NT,LST,

LPR,LX,T0,X,LDEL,AN, BN,V,S) ,

где A, B, N, P, Q, R, X – описаны в п.3.1,

параметры NMAX, LAS, EPS, NT, LST, LPR, LX, T0, LDEL задаются пользователем.

NMAX – размерность массивов P, Q, R, Y0, X, AN, BN, V, S в главной программе, $NMAX \geq N$.

EV0 и Y0 – начальное приближение к собственному значению и собственной функции, задаваемое пользователем. Оно может быть получено способом, изложенным в п.3.1 с помощью подпрограммы NEWTON или каким-то другим образом. После завершения работы подпрограммы SLIPH4 в EV0 и Y0 находится уточненное решение задачи (1).

LAS – параметр, определяющий наличие асимптотики для собственной функции на $[b, \infty)$. Если LAS= 0 , задача считается определенной на конечном интервале $[a, b]$. При LAS$\neq$ 0 происходит обращение к подпрограмме пользователя ASIM для вычисления интегралов, входящих в выражения (2.2.5) и (2.3.3).

YS – рабочий массив размерности NMAX в главной программе.

EPS – малое число ε (2.2.7).

NT – максимально допустимое количество итераций по t.

LST – параметр, определяющий способ выбора шага τ_k (см. п. 2.5).

τ_k определяется в соответствии со значением LST:

при LST=1 – по алгоритму 2.5.1,

при LST=2 – по алгоритму 2.5.2,

при LST=3 – по алгоритму 2.5.3,

при LST=4 – по алгоритму 2.5.4,

при LST=5 – по алгоритму 2.5.5.

LPR – параметр, определяющий шаг выдачи на печать результатов промежуточных итераций (2.2.6).

LX – шаг таблицы функции $y_k(x)$ при печати $h_{LPT} = h * LX$.

T0 – заданное начальное значение τ_0 . Рекомендуемые значения $\tau_0 = 0.01; 0.05; 0.1$.

LDEL – параметр, задающий способ вычисления невязки δ_k :

при LDEL=1 – по формуле (2.2.8),

при LDEL=2 – по формуле (2.2.9).

AN, BN, V, S – массивы размерности NMAX для прогоночных коэффициентов и значений функции ϑ_{ki} (2.2.3).

В подпрограмме используются общие блоки:

/CMINMA/ , в котором в главной программе пользователем определен диапазон чисел на данной ЭВМ;

/MOD/AK,BK,MOD , в котором значение MOD= 0 определяет реализацию итерационного процесса п.2.2, а значение MOD= 2 – процесса, описанного в п.2.3.

AK, BK – значения правых частей в граничных условиях (2.2.4) или (2.3.2).

В SLIPH4 происходит обращение к подпрограммам AKBK, DELTA, PROGRV, SKI, SIMPYV, TAUKA. Обращение к подпрограмме ASIM происходит при LAS $\neq 0$.

Подпрограмма ASIM составляется пользователем, если это необходимо. В ней должны быть вычислены значения интегралов на полубесконечном интервале $[b, \infty)$, входящие в выражения (2.2.5) и (2.3.3).

Обращение:

CALL ASIM(A,B,EV,Y(N),S1,S2) ,

где A, B – границы отрезка [a, b] ,

EV – собственное значение λ_k,

Y(N) – значение собственной функции $y_k(b)$,

S1 – значение интеграла $\int\limits_b^\infty \psi^2(\lambda_k, x)dx$, S2 – значение интеграла $\int\limits_b^\infty \psi(\lambda_k,x)\psi'_\lambda(\lambda_k,x)dx$.

Подпрограмма AKBK вычисляет величины a_{ik} и b_{ik} по формулам (2.2.4) или (2.3.2). Она, в свою очередь, обращается к подпрограммам пользователя D1F1 и D2F2 , которые вычисляют значения $d_1, f_1, d'_{1\lambda}, f'_{1\lambda}, d_2, f_2, d'_{2\lambda}, f'_{2\lambda}$ в выражениях (2.2.4) и (2.3.2).

Обращение:

CALL D1F1(EV,A,D1,DD1,F1,DF1) ,

где EV – заданное значение λ_k ,

A – левая граница отрезка [a, b],

D1 – $d_1(\lambda_k,a)$,

DD1 – $d'_{1\lambda}(\lambda_k,a)$,

F1 – $f_1(\lambda_k,a)$,

DF1 – $f'_{1\lambda}(\lambda_k,a)$.

CALL D2F2(EV,B,D2,DD2,F2,DF2) ,

где EV – заданное значение λ_k ,

B – правая граница отрезка [a,b] ,

D2 – $d_2(\lambda_k,b)$,

DD2 – $d'_{2\lambda}(\lambda_k,b)$,

F2 – $f_2(\lambda_k,b)$,

DF2 – $f'_{2\lambda}(\lambda_k,b)$.

Подпрограмма DELTA вычисляет значение невязки δ_k по формулам (2.2.8) или (2.2.9).

В PROGRV методом прогонки определяются функции $\vartheta_k^{(i)}$ в задачах (2.2.3), (2.2.4) и (2.3.1), (2.3.2).

В подпрограмме SIMPYV находятся значения интегралов

$\int\limits_a^b y_k^2(x)dx$ и $\int\limits_a^b y_k(x)\vartheta_k^{(i)}(x)dx$, используемые в выражениях (2.2.5) и (2.3.3).

Подпрограмма TAUKA реализует описанные в п.2.5 алгоритмы вычисления параметра τ_k.

Подпрограмма SKI вычисляет значения выражений $s_{2k}, s_k^{(i)}, i = 1, 2$ по формулам (2.2.4) и (2.3.2).

Подпрограмма DOPORT осуществляет ортогонализацию и нормировку по формулам (2.3.5) и (2.3.6).

Результаты вычислений при $MOD = 0$ (п.2.1) записываются в выходной файл SLIPH4_2.DAT.

Если используется модифицированный алгоритм, описанный в п.2.3, (MOD=2), то во второй выходной файл SLIPH4_1.DAT записываются все найденные собственные функции, к которым применена процедура ортогонализации (2.3.5).

4. Примеры использования комплекса SLIPH4

1. В качестве примера использования программы SLIPH4 было рассмотрено решение задачи (1) для случая потенциала Морзе /1/

$p(x) \equiv 0, \qquad q(x) = -2MD(e^{-2a_1(x-x_0)} - 2e^{-a_1(x-x_0)})$

на отрезке [a,b] с граничными условиями

$d_1 = d_2 = 1;\ f_1 = \sqrt{\lambda} - \sqrt{2MD}e^{a_1 x_0},\ f_2 = \sqrt{\lambda} - \sqrt{2MD}e^{-a_1(b-x_0)}$,

где $a = 0,\ b = 20$, константы $a_1 = 0.67;\ x_0 = 2.15;\ M = 4.69; D = 0.1055$, шаг дискретной сетки $\omega_h \qquad h = 0.0125\ (N = 1601)$.

В таблице 1 приведены результаты работы комплексов программ SLIPH4, SLIP1 и аналитическое решение. LDEL=1 (невязка δ считается по формуле (2.2.8)), LST=3

(параметр τ_k определяется по формуле (2.5.3)), k – количество итераций.

Таблица 1

	Аналит.	SLIPH4	SLIP1
λ	4.353116E-1	4.35311593E-1	4.35317356E-1
x	$YANAL$	$y^{(k)}, k = 11$	$y^{(k)}, k = 100$
0	1.88093417E-2	1.88093441E-2	1.880620E-2
2	5.14335066E-1	5.14335081E-1	5.143384E-1
4	4.61811049E-1	4.61811042E-1	4.618091E-1
6	1.69509499E-1	1.69509480E-1	1.695076E-1
8	4.92265882E-2	4.92265749E-2	4.922594E-2
10	1.34452673E-2	1.34452604E-2	1.344509E-2
12	3.61380423E-3	3.61380121E-3	3.613756E-3
14	9.67238461E-4	9.67237232E-4	9.672260E-4
16	2.58597485E-4	2.58597015E-4	2.585942E-4
18	6.91177870E-5	6.91176144E-5	6.911688E-5
20	1.84723677E-5	1.84723066E-5	1.847214E-5
		$\delta = 0.23E-7$	$\delta = 9.35E-11$

Таблица 1 наглядно демонстрирует преимущества комплекса SLIPH4 по сравнению со SLIP1 в смысле точности получаемого решения и скорости сходимости итерационного процесса.

В таблице 2 приведены результаты работы комплекса SLIPH4 на последовательности вдвое сгущающихся сеток (h = 0.05). Видно, что сходимость разностного решения задачи (1) имеет порядок h^4.

Таблица 2

	$\lambda_h = 0.435311628$	$\lambda_{h/2} = 0.435311571$	$\lambda_{h/4} = 0.435311568$	$\sigma = 16.3$
x	y_h	$y_{h/2}$	$y_{h/4}$	σ
0	$0.188097822 \cdot 10^{-1}$	$0.188093685 \cdot 10^{-1}$	$0.188093434 \cdot 10^{-1}$	16.4
3	0.592725832	0.592725848	0.592725849	16.5
6	0.169509462	0.169509497	0.169509499	16.3
9	$0.258172752 \cdot 10^{-1}$	$0.258172844 \cdot 10^{-1}$	$0.258172849 \cdot 10^{-1}$	16.2
12	$0.361380251 \cdot 10^{-2}$	$0.361380418 \cdot 10^{-2}$	$0.361380429 \cdot 10^{-2}$	16.2
15	$0.500156669 \cdot 10^{-3}$	$0.500156968 \cdot 10^{-3}$	$0.500156987 \cdot 10^{-3}$	16.2
18	$0.691177382 \cdot 10^{-4}$	$0.691177892 \cdot 10^{-4}$	$0.691177924 \cdot 10^{-4}$	16.2
δ	$0.34 \cdot 10^{-9}$	$0.36 \cdot 10^{-9}$	$0.44 \cdot 10^{-9}$	

Здесь

$$\sigma = \frac{z_h - z_{h/2}}{z_{h/2} - z_{h/4}}.$$

В этом примере $p(x) \equiv 0$.

2. Работа комплекса для случая $p(x) \neq 0$ демонстрируется на уравнении Лежандра

$$y^{''} - \frac{2x}{1-x^2}y^{'} - \frac{\lambda}{1-x^2}y = 0,$$

$-1 < x < 1$,

с граничными условиями

$$y^{'} - \frac{\lambda}{2}y|_{x=-1} = 0,$$

$$y^{'} + \frac{\lambda}{2}y|_{x=+1} = 0.$$

В таблице 3 приведены результаты, полученные на последовательности вдвое сгущающихся сеток (h = 0.04) для решения с двумя нулями.

Дальнейшим развитием пакета является добавление к нему блока автоматического нахождения параметра N дискретной сетки ω_h (а также параметров A и B в случае сингулярной граничной задачи) в зависимости от требуемой точности результата.

Таблица 3

	$\lambda_h = -6.00000000$	$\lambda_{h/2} = -6.000000000$	$\lambda_{h/4} = -6.00000000$	16.0
x	y_h	$y_{h/2}$	$y_{h/4}$	σ
-1.0	$0.158113581 \cdot 10$	$0.158113864 \cdot 10$	$0.158113882 \cdot 10$	16.0
-0.8	0.727322472	0.727323775	0.727323856	16.0
-0.6	$0.632454326 \cdot 10^{-1}$	$0.632455456 \cdot 10^{-1}$	$0.632455527 \cdot 10^{-1}$	16.0
-0.4	-0.411095310	-0.411096046	-0.411096093	16.0
-0.2	-0.695699755	-0.695701002	-0.695701080	16.0
0.0	-0.790567904	-0.790569320	-0.790569409	16.0
0.2	-0.695699755	-0.695701002	-0.695701080	16.0
0.4	-0.411095310	-0.411096046	-0.411096093	16.0
0.6	$0.632454326 \cdot 10^{-1}$	$0.632455456 \cdot 10^{-1}$	$0.632455527 \cdot 10^{-1}$	16.0
0.8	0.727322472	0.727323775	0.727323856	16.0
1.0	$0.158113581 \cdot 10$	$0.158113864 \cdot 10$	$0.158113882 \cdot 10$	16.0
δ	$0.28 \cdot 10^{-10}$	$0.33 \cdot 10^{-10}$	$0.12 \cdot 10^{-9}$	

Литература

1. *Пузынин И.В., Пузынина Т.П.* В сб.: Алгоритмы и программы для решения некоторых задач физики. KFKI -74-34, Будапешт, 1974, с.93-111.

2. *Марчук Г.И., Шайдуров В.В.* Повышение точности решений разностных схем. "Наука", М., 1979.

3. *Акишин П.Г., Пузынин И.В.* ОИЯИ, 5-10992, Дубна, 1977.

4. *Уилкинсон Дж.Х.* Алгебраическая проблема собственных значений. "Наука", М., 1970.

5. *Bailey P.B.* SLEIGN An Eigenvalue–Eigenfunction Code for Sturm–Liouville Problems. SAND77-2044, Sandia Laboratories, 1978.

6. *Faifman M.P. et al.* Z. Phys. D, 1986, 2, p.79-85;

7. *Виницкий С.И. и др.* Акустический журнал, 1985, 31, с.787-790.
Касчиев М.С. и др. ОИЯИ, Р11-84-832, Дубна, 1984.

8. *Калиткин Н.Н.* Численные методы. "Наука", М., 1978.

9. Современные численные методы решения обыкновенных дифференциальных уравнений. Ред. Дж.Холл, Дж.Уатт. "Мир", М., 1979.

10. *Дымарский Я.С. и др.* Справочник программиста, т.1, Судпром ГИЗ, Л., 1963.

11. *Ермаков В.В., Калиткин Н.Н.* ЖВМиМФ, 1981, 21, с.491.

12. *Марчук Г.И., Шайдуров В.В.* Повышение точности решений разностных схем. "Наука", М., 1979.

13. *Гавурин М.К.,* Изв. Вузов, Матем., 1958. 5(6), с.18.