ПРОГРАММИРОВАНИЕ И МАТЕМАТИКА*

В. Н. Малозёмов, А. Б. Певный
malv@gamma.math.spbu.ru pevnyi@syktsu.ru

17 мая 2005 г.

Введение

При разработке численного метода решения некоторой задачи нужно иметь в виду, что метод будет реализован в форме алгоритма, а затем программы. Конечным звеном этой цепочки является *программа*, и именно программа характеризует качество цепочки в целом.

Обычно существует несколько способов решения поставленной задачи. Каждый способ реализуется своей программой. Как конечный продукт, программа может быть примитивной или изящной, красивой. К сожалению, понятие «красивая программа» не формализуется. К необходимым условиям для составления такой программы следует отнести стремление к минимизации количества операций и используемой памяти. Этого можно достичь с помощью математических приёмов. В определённом смысле, *хорошая программа основана на хорошей математике.* Приведём три характерных примера.

Пример 1. Вычислить сумму

$$S_n = \sum_{k=1}^{n} \frac{(-1)^{k(k-1)/2}}{k!!},$$

где $k!!$ (двойной факториал) — это произведение натуральных чисел, не превосходящих k и имеющих одинаковую с k чётность, т. е.

$$(2m)!! = 2 \cdot 4 \ldots (2m), \quad (2m+1)!! = 1 \cdot 3 \cdot 5 \ldots (2m+1).$$

Обозначим $b_k = k(k-1)/2$, $c_k = k!!$, $a_k = (-1)^{b_k}/c_k$. Тогда $S_n = \sum_{k=1}^{n} a_k$. В принципе, можно независимо вычислить все a_k и, сложив их, получить S_n.

*Санкт-Петербургский городской семинар «Всплески (wavelets) и их приложения». Секция «Дискретный гармонический анализ»: http://www.math.spbu.ru/user/dmp/dha/

Программа, реализующая такой способ, будет примитивной. Чтобы написать красивую программу, будем действовать иначе.

Отметим, что $b_k - b_{k-2} = 2k - 3$. Это приводит к рекуррентному соотношению для последовательности $\{b_k\}$:

$$b_k = b_{k-2} + (2k-3),\ k = 2, 3, \ldots; \quad b_0 = b_1 = 0.$$

Менее тривиальным является рекуррентное соотношение для $\{c_k\}$:

$$c_k = kc_{k-2},\ k = 2, 3, \ldots; \quad c_0 = c_1 = 1.$$

Теперь имеем

$$a_k = \frac{(-1)^{b_k}}{c_k} = \frac{(-1)^{b_{k-2}+(2k-3)}}{kc_{k-2}} = -\frac{1}{k}a_{k-2},$$

так что

$$a_k = -a_{k-2}/k,\ k = 2, 3, \ldots; \quad a_0 = a_1 = 1.$$

Это соотношение положим в основу программы:

```
a := 1; a1 := 1; s := 1;
for k := 2 to n do begin
    a2 := a1; a1 := a;
    a := −a2/k; s := s + a
end
```

После работы программы $s = S_n$. В ячейке a находится текущее значение a_k, в ячейках $a1, a2$ — значения a_{k-1}, a_{k-2}. Табл. 1 дает представление об изменении содержимого ячеек $a, a1, a2$ в цикле по k.

Таблица 1

k	a	$a1$	$a2$
1	a_1	a_0	–
2	a_2	a_1	a_0
3	a_3	a_2	a_1
...	...	...	...
n	a_n	a_{n-1}	a_{n-2}

Пример 2. Вычислить значение производной n-го порядка функции $f(x) = \sin(x)/x$ в точке $x \neq 0$.

В отличие от предыдущего примера у нас нет явной формулы для $f^{(n)}(x)$. Однако без этого можно обойтись. Последовательно дифференцируя, получаем

$$xf(x) = \sin(x), \qquad f(x) + xf'(x) = [\sin(x)]',$$

$$2f'(x) + xf''(x) = [\sin(x)]'', \qquad 3f''(x) + xf'''(x) = [\sin(x)]''', \ldots,$$

$$nf^{(n-1)}(x) + xf^{(n)}(x) = [\sin(x)]^{(n)}.$$

Приходим к рекуррентному соотношению по номеру производной:

$$f^{(k)}(x) = \left([\sin(x)]^{(k)} - kf^{(k-1)}(x)\right)/x, \qquad k = 1, 2, \ldots, n;$$
$$f^{(0)}(x) = \sin(x)/x.$$

В свою очередь величина $p_k = [\sin(x)]^{(k)}$ удовлетворяет рекуррентному соотношению

$$p_k = -p_{k-2},\ k = 1, 2, \ldots, n; \qquad p_{-1} = -\cos(x),\ p_0 = \sin(x).$$

Указанные соображения приводят к следующей эффектной программе:

```
p := sin(x); p1 := − cos(x); fn := p/x;
for k := 1 to n do begin
    p2 := p1; p1 := p; p := −p2;
    fn := (p − k ∗ fn)/x
end
```

После работы программы $fn = f^{(n)}(x)$.

Пример 3. Заданы n вещественных чисел $a[1], a[2], \ldots, a[n]$. Вычислить

$$p_n = \max_{1 \leqslant i \leqslant j \leqslant n} \sum_{l=i}^{j} a[l]$$

при дополнительном условии: каждое $a[l]$ можно вызывать только один раз.

Если не обращать внимания на дополнительное условие, то задача решается просто. Действительно, обозначим

$$s_{ij} = \sum_{l=i}^{j} a[l].$$

Вычислив последовательно суммы

$$\begin{array}{cccccc}
s_{11}, & s_{12}, & s_{13}, & \ldots & s_{1,n-1}, & s_{1n}, \\
 & s_{22}, & s_{23}, & \ldots & s_{2,n-1}, & s_{2n}, \\
 & & \ldots & \ldots & \ldots & \ldots \\
 & & & & s_{n-1,n-1}, & s_{n-1,n}, \\
 & & & & & s_{nn}
\end{array}$$

и найдя среди них наибольшую, получим p_n. Дополнительное условие существенно усложняет задачу. Более того, не совсем понятно, как подступиться к её решению. На помощь приходят рекуррентные мотивы.

Положим

$$p_k = \max_{1 \leqslant i \leqslant j \leqslant k} s_{ij}, \quad q_k = \max_{1 \leqslant i \leqslant k} s_{ik}.$$

Очевидно, что

$$p_k = \max\{p_{k-1}, q_k\}, \ k = 2, 3, \ldots, n; \quad p_1 = a[1]. \tag{1.1}$$

Поскольку

$$\begin{aligned} q_k = \max\{s_{1k}, s_{2k}, \ldots, s_{k-1,k}, s_{kk}\} &= \\ = \max\{a[1] + a[2] + \ldots + a[k-1] + a[k], & \\ a[2] + \ldots + a[k-1] + a[k], & \\ \ldots\ldots\ldots\ldots\ldots\ldots & \\ a[k-1] + a[k], & \\ 0 + a[k]\} &= \\ = a[k] + \max\{q_{k-1}, 0\}, & \end{aligned}$$

то

$$q_k = a[k] + \max\{0, q_{k-1}\}, \ k = 2, 3, \ldots, n; \quad q_1 = a[1]. \tag{1.2}$$

Система рекуррентных соотношений (1.1), (1.2) позволяет записать редкую по красоте программу:

```
p := a[1]; q := p;
for k := 2 to n do begin
    if q < 0 then q := 0;
    q := a[k] + q;
    if p < q then p := q;
end
```

После работы программы $p = p_n$. Видно, что каждое $a[k]$ вызывается только один раз!

Во всех трёх примерах для составления продвинутой программы использовались рекуррентные соотношения. Такие соотношения не всегда лежат на поверхности, и для их обнаружения требуется определённый опыт. Вместе с тем, в области рекуррентных вычислений существуют вполне регулярные приёмы, которые применяются при решении существенно разных по содержанию задач. Одним из таких приёмов является схема Кленшоу. Эта замечательная схема, предложенная 50 лет назад, не относится однако к числу широко известных.

2. Схема Кленшоу

2.1. Рассмотрим алгебраический полином вида

$$P(x) = \frac{1}{2}a[0] + \sum_{k=1}^{n} a[k]T_k(x),$$

где $T_k(x)$ — полиномы Чебышёва, определяемые рекуррентным соотношением

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x),\ k = 2, 3, \ldots; \quad T_0(x) \equiv 1,\ T_1(x) = x. \tag{2.1}$$

Поставим задачу: вычислить значение $P(x)$ в фиксированной точке x.

Примитивную программу, решающую эту задачу, можно записать сразу.

```
p := 0.5 * a[0] + x * a[1];
t := x; t1 := 1; c := 2 * x;
for k := 2 to n do begin
    t2 := t1; t1 := t;
    t := c * t1 - t2;
    p := p + t * a[k]
end
```

После работы программы $p = P(x)$. В программе используется $2n+1$ умножений и $2n-1$ сложений.

Кленшоу в работе [1] предложил другой метод решения, который приводит к программе с $n+2$ умножениями и $2n+1$ сложениями. Идея Кленшоу заключается в следующем.

Введём рекуррентную последовательность $\{b_k\}$, исходя из условий

$$a[k] = b_k - 2xb_{k+1} + b_{k+2},\ k = n, n-1, \ldots, 0; \quad b_{n+2} = b_{n+1} = 0. \tag{2.2}$$

Такую последовательность построить можно:

$$b_{n+1} = 0,\ b_n = a[n]; \quad b_k = a[k] + 2xb_{k+1} - b_{k+2},\ k = n-1, n-2, \ldots, 0. \tag{2.3}$$

Обозначим $\tilde{P}(x) = P(x) + \frac{1}{2}a[0]$, $t_k = T_k(x)$ и запишем

$$\begin{aligned}
\tilde{P}(x) &= \sum_{k=0}^{n} a[k]t_k = \sum_{k=0}^{n}(b_k - 2xb_{k+1} + b_{k+2})t_k = \\
&= \sum_{k=0}^{n} b_k t_k - 2x\sum_{k=1}^{n+1} b_k t_{k-1} + \sum_{k=2}^{n+2} b_k t_{k-2} = \\
&= b_0t_0 + b_1t_1 - 2xb_1t_0 + \sum_{k=2}^{n} b_k(t_k - 2xt_{k-1} + t_{k-2}).
\end{aligned}$$

Согласно (2.1) последняя сумма равна нулю, поэтому

$$\tilde{P}(x) = b_0 t_0 + b_1(t_1 - 2xt_0) = b_0 - xb_1.$$

Теперь имеем

$$P(x) = \tilde{P}(x) - \frac{1}{2}a[0] = b_0 - xb_1 - \frac{1}{2}(b_0 - 2xb_1 + b_2) = \frac{1}{2}(b_0 - b_2).$$

Мы пришли к замечательной формуле

$$P(x) = \frac{1}{2}(b_0 - b_2).$$

Она показывает, что в основу вычисления значения $P(x)$ можно положить рекуррентное соотношение (2.3). Приведём соответствующую программу:

$b := a[n];\ b1 := 0;\ c := 2 * x;$
for $k := n - 1$ **downto** 0 **do begin**
    $b2 := b1;\ b1 := b;$
    $b := a[k] + c * b1 - b2$
end;
$p := 0.5 * (b - b2)$

После работы программы $p = P(x)$. В ячейке b находится текущее значение b_k, в ячейках $b1, b2$ — значения b_{k+1}, b_{k+2}. Как и было обещано, в программе используется $n + 2$ умножения и $2n + 1$ сложения.

Приём Кленшоу заметно уменьшает количество умножений. Суть этого приёма составляет разложение (2.2) коэффициентов $a[k]$, в котором правая часть соответствует однородной форме рекуррентного соотношения (2.1).

2.2. Встанем на более общую точку зрения. Она поможет нам выявить дополнительные возможности схемы Кленшоу.

Пусть имеется скалярный динамический процесс

$$v_k = c_k v_{k-1} + d_k v_{k-2}, \quad k = 1, 2, \ldots, n, \tag{2.4}$$

с заданными начальными значениями v_0, v_{-1}. Определим стоимость траектории линейной функцией

$$P = \sum_{k=1}^{n} a_k v_k.$$

Требуется вычислить P как функцию начальных значений v_0, v_{-1}.

Воспользуемся идеей Кленшоу. Введём числовую последовательность $\{b_k\}$, исходя из условий

$$a_k = b_k - c_{k+1}b_{k+1} - d_{k+2}b_{k+2},\ k = n, n-1, \ldots, 1; \quad b_{n+2} = b_{n+1} = 0. \tag{2.5}$$

Такую последовательность построить можно:

$$b_n = a_n,\ b_{n-1} = a_{n-1} + c_n b_n,$$
$$b_k = a_k + c_{k+1} b_{k+1} + d_{k+2} b_{k+2}, \quad k = n-2, n-3, \ldots, 1. \tag{2.6}$$

На основании (2.5) запишем

$$\begin{aligned} P &= \sum_{k=1}^{n} (b_k - c_{k+1} b_{k+1} - d_{k+2} b_{k+2}) v_k = \\ &= \sum_{k=1}^{n} b_k v_k - \sum_{k=2}^{n+1} b_k c_k v_{k-1} - \sum_{k=3}^{n+2} b_k d_k v_{k-2} = \\ &= b_1 v_1 + b_2 v_2 - b_2 c_2 v_1 + \sum_{k=3}^{n} b_k (v_k - c_k v_{k-1} - d_k v_{k-2}). \end{aligned}$$

В силу (2.4) последняя сумма равна нулю. Значит,

$$P = b_1 v_1 + b_2 (v_2 - c_2 v_1) = b_1 v_1 + b_2 d_2 v_0 =$$
$$= b_1 (c_1 v_0 + d_1 v_{-1}) + b_2 d_2 v_0 = (b_1 c_1 + b_2 d_2) v_0 + b_1 d_1 v_{-1}.$$

Если положить $a_0 = 0$ и расширить рекуррентное соотношение (2.6) до $k = 0$, то получим

$$P = b_0 v_0 + b_1 d_1 v_{-1}.$$

Приведём на псевдоязыке программу вычисления P как функции параметров v_0, v_{-1}:

$a_0 := 0;\ b1 := a_n;$
$b := a_{n-1} + c_n * b1;$
for $k := n - 2$ **downto** 0 **do begin**
  $b2 := b1;\ b1 := b;$
  $b := a_k + c_{k+1} * b1 + d_{k+2} * b2$
end;
$p := b * v_0 + b1 * d_1 * v_{-1}$

Видим, что изменение начальных значений v_0, v_{-1} повлияет только на последнюю строку программы.

Коэффициенты a_k, c_k, d_k могут быть заданы явно в виде простой формулы или, в свою очередь, удовлетворять некоторым рекуррентным соотношениям. Это следует учитывать при составлении окончательного варианта программы.

3. Алгоритм Гёрцеля

Заметка Кленшоу [1] была опубликована в 1955 году. В 1958 году Гёрцель предложил быстрый алгоритм вычисления одной компоненты дискретного преобразования Фурье [2]. В этом разделе мы покажем, что алгоритм Гёрцеля является одной из реализаций схемы Кленшоу.

Напомним, что дискретное преобразование Фурье определяется формулой

$$X_k = \sum_{j=0}^{n-1} x_j \omega_n^{-kj}, \quad k = 0, 1, \ldots, n-1. \tag{3.1}$$

Здесь $\omega_n = \exp(2\pi i/n)$, так что

$$\omega_n^{-kj} = \cos\left(\frac{2\pi kj}{n}\right) - i \sin\left(\frac{2\pi kj}{n}\right).$$

Рассмотрим задачу вычисления X_k *при фиксированнном* k по заданным $x_0, x_1, \ldots, x_{n-1}$ (вообще говоря, комплексным).

Обозначим $\varphi = 2\pi k/n$, $c_j = \cos(j\varphi)$, $s_j = \sin(j\varphi)$. Тогда формулу (3.1) можно переписать так:

$$X_k = x_0 + \sum_{j=1}^{n-1} x_j c_j - i \sum_{j=1}^{n-1} x_j s_j. \tag{3.2}$$

Для одновременного вычисления сумм

$$A_k = \sum_{j=1}^{n-1} x_j c_j, \quad B_k = \sum_{j=1}^{n-1} x_j s_j$$

воспользуемся идеей Кленшоу. Для этого заметим, что при $j = 1, 2, \ldots$

$$\begin{aligned} c_j + c_{j-2} &= 2\cos(\varphi) c_{j-1}, & s_j + s_{j-2} &= 2\cos(\varphi) s_{j-1}, \\ c_0 = 1,\ c_{-1} &= \cos(\varphi); & s_0 = 0,\ s_{-1} &= -\sin(\varphi). \end{aligned}$$

Таким образом, последовательности $\{c_j\}$ и $\{s_j\}$ удовлетворяют одному и тому же рекуррентному соотношению

$$v_j = 2\cos(\varphi) v_{j-1} - v_{j-2}, \tag{3.3}$$

но разным начальным условиям. Формула (3.3) соответствует (2.4) при $c_j = 2\cos(\varphi)$, $d_j = -1$. Для вычисления суммы

$$P = \sum_{j=1}^{n-1} x_j v_j$$

введем последовательность $\{b_j\}$, исходя из условий

$$x_j = b_j - 2\cos(\varphi)b_{j+1} + b_{j+2},\ j = n-1, n-2, \ldots, 1; \qquad b_{n+1} = b_n = 0.$$

Здесь

$$\begin{gathered} b_n = 0,\ b_{n-1} = x_{n-1}, \\ b_j = x_j + 2\cos(\varphi)b_{j+1} - b_{j+2},\ j = n-2, n-3, \ldots, 1. \end{gathered} \tag{3.4}$$

Тогда, как показано в п. 2.2,

$$P = (b_1c_1 + b_2d_2)v_0 + b_1d_1v_{-1} = (2\cos(\varphi)b_1 - b_2)v_0 - b_1v_{-1}.$$

В частности,

$$A_k = (2\cos(\varphi)b_1 - b_2) - b_1\cos(\varphi) = b_1\cos(\varphi) - b_2, \qquad B_k = b_1\sin(\varphi).$$

Вспоминая (3.2), приходим к окончательному представлению

$$X_k = x_0 + A_k - iB_k = x_0 - b_2 + b_1(\cos(\varphi) - i\sin(\varphi)). \tag{3.5}$$

Вычисление X_k по формуле (3.5) называется *алгоритмом Гёрцеля*. Мы получили этот алгоритм с помощью схемы Кленшоу.

Основным элементом алгоритма Гёрцеля является вычисление коэффициентов b_1, b_2 по формуле (3.4). Запишем программу вычисления указанных коэффициентов:

```
b := x[n-1]; b1 := 0;
a := 2 * cos(2 * π * k/n);
for j := n - 2 downto 1 do begin
    b2 := b1; b1 := b;
    b := x[j] + a * b1 - b2
end
```

После работы программы $b = b_1$, $b1 = b_2$. В программе используется $n-2$ умножения на вещественное число a, $2(n-2)$ сложения и лишь один раз вычисляется тригонометрическая функция!

4. Решение системы линейных уравнений с трёхдиагональной матрицей

Рассмотрим систему линейных уравнений

$$\begin{gathered} q_kx_{k-1} + p_kx_k + x_{k+1} = f_k,\ k = 1, 2, \ldots, n, \\ x_0 = c,\ x_{n+1} = d. \end{gathered} \tag{4.1}$$

В отличие от (2.4) соотношения (4.1) являются неоднородными, а вместо начальных условий заданы краевые условия. Тем не менее, для решения системы (4.1) попытаемся применить схему Кленшоу. Введём «стоимость траектории»

$$P = \sum_{k=1}^{n+1} a_k x_k,$$

где $a_1 = a_2 = \ldots = a_n = 0,\ a_{n+1} = 1$. Собственно, величина P нам известна, $P = d$. Приём Кленшоу побуждает нас ввести специальное представление для известных коэффициентов a_k.

Построим последовательность $\{b_k\}$, исходя из условий

$$a_k = q_{k+1}b_{k+1} + p_k b_k + b_{k-1},\ k = n+1, n, \ldots, 1; \qquad b_{n+2} = b_{n+1} = 0. \tag{4.2}$$

Такую последовательность построить можно:

$$b_n = 1,\ b_{n-1} = -p_n; \qquad b_{k-1} = -p_k b_k - q_{k+1}b_{k+1},\ k = n-1, n-2, \ldots, 1.$$

Согласно (4.2) и (4.1)

$$\begin{aligned} P &= \sum_{k=1}^{n+1}(q_{k+1}b_{k+1} + p_k b_k + b_{k-1})x_k = \\ &= \sum_{k=2}^{n+2} b_k q_k x_{k-1} + \sum_{k=1}^{n+1} b_k p_k x_k + \sum_{k=0}^{n} b_k x_{k+1} = \\ &= b_1 p_1 x_1 + b_0 x_1 + b_1 x_2 + \sum_{k=2}^{n} b_k f_k. \end{aligned}$$

Поскольку $p_1 x_1 + x_2 = f_1 - q_1 x_0$, то

$$P = b_0 x_1 + b_1(f_1 - q_1 x_0) + \sum_{k=2}^{n} b_k f_k = b_0 x_1 - b_1 q_1 x_0 + \sum_{k=1}^{n} b_k f_k.$$

Значит,

$$b_0 x_1 = d + b_1 q_1 c - \sum_{k=1}^{n} b_k f_k. \tag{4.3}$$

Эта формула является следствием (4.1). В частности, если $b_0 = 0$, а правая часть (4.3) отлична от нуля, то система (4.1) не имеет решения.

Предположим, что $b_0 \neq 0$. Тогда

$$x_1 = \Big(d + b_1 q_1 c - \sum_{k=1}^{n} b_k f_k\Big)\Big/ b_0.$$

На основании (4.1) по известным x_0, x_1 последовательно найдём $x_2, x_3, \ldots, x_n$.

Запишем программу, реализующую описанный метод решения системы (4.1):

$b := -p_n;\ b1 := 1;\ s := f_n;$
for $k := n-1$ **downto** 1 **do begin**
  $s := s + b * f_k;$
  $b2 := b1;\ b1 := b;$
  $b := -b1 * p_k - b2 * q_{k+1}$
end;
$x[0] := c;\ x[1] := (d + c * b1 * q_1 - s)/b;$
for $k := 1$ **to** $n-1$ **do**
  $x[k+1] := f_k - p_k * x[k] - q_k * x[k-1];$
$x[n+1] := d$

После работы программы в массиве $x[0..n+1]$ будет находиться решение системы (4.1).

В окончательный вариант программы нужно включить проверку условия $b_0 \neq 0$ в виде $|b_0| > eps$, где eps — малое положительное число.

Другой подход к решению системы (4.1) предложен в [3].

5. Свёртывание конечной цепной дроби

Рассмотрим последовательность цепных дробей

$$R_0(x) \equiv b_0, \quad R_1(x) = b_0 + \frac{a_1 x}{b_1}, \quad R_2(x) = b_0 + \cfrac{a_1 x}{b_1 + \cfrac{a_2 x}{b_2}}, \ldots,$$

$$R_n(x) = b_0 + \cfrac{a_1 x}{b_1 + \cfrac{a_2 x}{b_2 + \begin{array}{c}\vdots\\ + \cfrac{a_n x}{b_n}\end{array}}}.$$

Каждую такую дробь можно представить в виде отношения алгебраических полиномов

$$R_k(x) = \frac{P_k(x)}{Q_k(x)}.$$

Например,

$$R_0(x) = \frac{b_0}{1}, \quad R_1(x) = \frac{b_0 b_1 + a_1 x}{b_1}, \quad R_2(x) = \frac{b_0(b_1 b_2 + a_2 x) + b_2 a_1 x}{b_1 b_2 + a_2 x}.$$

Поставим задачу: вычислить коэффициенты полиномов $P_n(x)$ и $Q_n(x)$.

Как известно [4, 5], последовательности $\{P_k(x)\}$ и $\{Q_k(x)\}$ удовлетворяют рекуррентным соотношениям: при $k = 1, 2, \ldots, n$

$$P_k(x) = b_k P_{k-1}(x) + a_k x P_{k-2}(x),$$
$$Q_k(x) = b_k Q_{k-1}(x) + a_k x Q_{k-2}(x),$$
$$P_0(x) \equiv b_0, \quad Q_0(x) \equiv 1, \quad P_{-1}(x) \equiv 1, \quad Q_{-1}(x) \equiv 0.$$

Проследим за степенями полиномов $P_k(x)$ и $Q_k(x)$ (табл. 2).

Таблица 2

k	0	1	2	3	4	5	…	n
степень P_k	0	1	1	2	2	3	…	$\lfloor (n+1)/2 \rfloor$
степень Q_k	0	0	1	1	2	2	…	$\lfloor n/2 \rfloor$

В программе коэффициенты полиномов P_k, Q_k будут размещены в массивах $p, q[0..\lfloor (n+1)/2 \rfloor]$.

Отметим, что полиномы P_k, Q_k удовлетворяют одному и тому же рекуррентному соотношению

$$v_k = b_k v_{k-1} + a_k x v_{k-2}, \qquad k = 1, 2, \ldots, n,$$

при разных начальных значениях v_0, v_{-1}. Выразим v_n через v_0 и v_{-1}. Для этого воспользуемся схемой Кленшоу.

Имеем

$$v_n = \sum_{k=0}^{n} c_k v_k,$$

где $c_0 = c_1 = \ldots = c_{n-1} = 0,\ c_n = 1$. Представим c_k в виде

$$c_k = g_k - b_{k+1} g_{k+1} - a_{k+2} x g_{k+2},\ k = n, n-1, \ldots, 0; \quad g_{n+2} = g_{n+1} = 0.$$

Последовательность $\{g_k\}$ построить можно:

$$g_n = 1,\ g_{n-1} = b_n; \qquad g_k = b_{k+1} g_{k+1} + a_{k+2} x g_{k+2},\ k = n-2, n-3, \ldots, 0. \quad (5.1)$$

Здесь $g_k = g_k(x)$ — алгебраический полином. Теперь запишем

$$\begin{aligned} v_n &= \sum_{k=0}^{n} (g_k - b_{k+1} g_{k+1} - a_{k+2} x g_{k+2}) v_k = \\ &= \sum_{k=0}^{n} g_k v_k - \sum_{k=1}^{n+1} g_k b_k v_{k-1} - \sum_{k=2}^{n+2} g_k a_k x v_{k-2} = \\ &= g_0 v_0 + g_1 v_1 - g_1 b_1 v_0 = g_0 v_0 + g_1 a_1 x v_{-1}. \end{aligned}$$

Если положить $v_0 = b_0,\ v_{-1} = 1$, то получим

$$P_n(x) = b_0 g_0(x) + a_1 x g_1(x). \tag{5.2}$$

Если положить $v_0 = 1,\ v_{-1} = 0$, то получим $Q_n(x) = g_0(x)$. Правая часть (5.2) соответствует правой части (5.1) при $k = -1$. Перепишем (5.1) в виде

$$g_n = 1,\ g_{n-1} = b_n; \qquad g_{k-1} = b_k g_k + a_{k+1} x g_{k+1},\ k = n-1, n-2, \ldots, 0. \tag{5.3}$$

Тогда $P_n(x) = g_{-1}(x)$, $Q_n(x) = g_0(x)$. Таким образом, задача вычисления коэффициентов полиномов $P_n,\ Q_n$ сведена к задаче вычисления коэффициентов полиномов g_{-1}, g_0 по формуле (5.3).

Отметим, что степень полинома g_k не превосходит $r_k = \lfloor (n-k)/2 \rfloor$. Очевидно, что

$$r_{k-1} = r_{k+1} + 1,\ k = n-1, n-2, \ldots, 0; \quad r_n = r_{n-1} = 0.$$

В табл. 3 приведены значения r_k.

Таблица 3

k	n	$n-1$	$n-2$	$n-3$	$\ldots$	0	-1
r_k	0	0	1	1	$\ldots$	$\lfloor n/2 \rfloor$	$\lfloor (n+1)/2 \rfloor$

Пусть

$$g_k(x) = \sum_{j=0}^{r_k} \xi_{kj} x^j.$$

Из (5.3) выведем рекуррентное соотношение для коэффициентов ξ_{kj}. Имеем

$$\sum_{j=0}^{r_{k-1}} \xi_{k-1,j} x^j = b_k \sum_{j=0}^{r_k} \xi_{kj} x^j + a_{k+1} \sum_{j=1}^{r_{k-1}} \xi_{k+1,j-1} x^j.$$

Отсюда следует, что при $k = n-1, n-2, \ldots, 0$

$$\begin{gathered} \xi_{k-1,0} = b_k \xi_{k0}; \quad \xi_{k-1,j} = b_k \xi_{kj} + a_{k+1} \xi_{k+1,j-1},\ j = 1, 2, \ldots, r_k; \\ \xi_{k-1,r_{k-1}} = a_{k+1} \xi_{k+1,r_{k-1}-1},\ \text{если } r_k < r_{k-1}. \end{gathered} \tag{5.4}$$

При $r_k < r_{k-1}$ можно положить $\xi_{k,r_{k-1}} = 0$ и использовать рекуррентное соотношение для $\xi_{k-1,j}$ при $j = 1, 2, \ldots, r_{k-1}$. К (5.4) нужно добавить начальные условия $\xi_{n0} = 1$, $\xi_{n-1,0} = b_n$.

Матрица коэффициентов $\{\xi_{kj}\}$ заполняется строка за строкой снизу вверх, как показано в табл. 4 при $n = 4$, $r_{-1} = 2$.

Таблица 4

	$k \backslash j$	0	1	2
	-1	$\xi_{-1,0}$	$\xi_{-1,1}$	$\xi_{-1,2}$
	0	ξ_{00}	ξ_{01}	ξ_{02}
	1	ξ_{10}	ξ_{11}	0
$p_{\text{нов}}$	2	ξ_{20}	ξ_{21}	
p	3	ξ_{30}	0	
q	4	ξ_{40}		

Для формирования строк будут использоваться два массива $p, q\Big[0..\lfloor (n+1)/2 \rfloor\Big]$.

Приведём соответствующую программу:

```
p[0] := b_n; q[0] := 1;
r := 0; r1 := 0;
for k := n − 1 downto 0 do begin
    r2 := r1; r1 := r; r := r2 + 1;
    if r1 < r then p[r] := 0;
    u := b_k; v := a_{k+1};
    for j := r downto 1 do begin
        d := p[j];
        p[j] := u ∗ d + v ∗ q[j − 1];
        q[j] := d
    end;
    d := p[0]; p[0] := u ∗ d; q[0] := d
end
```

После работы программы $r = \lfloor (n+1)/2 \rfloor$ и

$$P_n(x) = \sum_{j=0}^{r} p[j]x^j, \quad Q_n(x) = \sum_{j=0}^{r} q[j]x^j.$$

6. Заключительные замечания

Мы рассмотрели схему Кленшоу применительно к линейным рекуррентным соотношениям 2-го порядка. Нетрудно понять, как она будет выглядеть в случае линейных рекуррентных соотношений 3-го, 4-го и более высоких порядков.

Наряду со схемой Кленшоу существуют и другие регулярные приёмы, используемые при составлении эффектных программ. Эти приёмы основаны на идеях из различных разделов дискретной математики, таких как комбинаторика, теория графов, динамическое программирование и пр. [6, 7, 8].

В заключение приведём 12 задач, допускающих красивые решения. Много других задач подобного рода имеются в [9, 10].

Задачи

1. Для объема k-мерного шара радиуса r справедлива формула

$$v_k = \frac{(2r)^k}{k!!}\left(\frac{\pi}{2}\right)^{\lfloor k/2\rfloor},\ k = 1, 2, \ldots$$

Вычислить v_n.

2. Вычислить значение производной n-го порядка функции

$$f(x) = \exp(-x^2/2)$$

в точке x.

3. Вычислить интеграл

$$c_n = \int_0^{\varphi} \sin^n x\, dx$$

при фиксированном φ и $n \geqslant 1$.

4. Алгебраический полином

$$P(x) = (1 + x + x^2 + \ldots + x^n)^r$$

представить в виде

$$P(x) = \sum_{k=0}^{rn} a_k x^k.$$

Коэффициенты $a_0, a_1, \ldots, a_{rn}$ записать в массив $a[0..rn]$.

5. Рассмотрим алгебраический полином в форме Бернштейна

$$B(x) = \sum_{k=0}^{n} y[k] C_n^k x^k (1-x)^{n-k}. \tag{6.1}$$

Вычислить $B(x)$ в точке x.

6. Полином (6.1) представить в виде

$$B(x) = \sum_{k=0}^{n} b_k x^k.$$

Коэффициенты $b_0, b_1, \ldots, b_n$ записать в исходный массив $y[0..n]$. (Такие преобразования называются *преобразованиями на месте.*)

7. Алгебраический полином

$$B(x) = \sum_{k=0}^{n} y[k] x^k$$

представить в форме Бернштейна (6.1).

8. Полиномы Бернулли $Q_k(x)$ определяются с помощью рекуррентного соотношения

$$Q_k'(x) = kQ_{k-1}(x), \ \int_0^1 Q_k(x)\,dx = 0, \ k = 1, 2, \ldots; \quad Q_0(x) \equiv 1.$$

Вычислить коэффициенты полинома $Q_n(x)$.

9. Рекуррентное соотношение

$$P_k(x) = \frac{d}{dx}\left[x(1-x)P_{k-1}(x)\right], \ k = 1, 2, \ldots; \quad P_0(x) \equiv 1$$

определяет так называемые *логистические полиномы*. Вычислить коэффициенты $P_n(x)$.

10. Числа Стирлинга первого рода вводятся как коэффициенты разложения

$$x(x-1)\ldots(x-n+1) = \sum_{k=1}^{n} (-1)^{n-k} a_{nk} x^k.$$

Вычислить $a_{n1}, a_{n2}, \ldots, a_{nn}$.

11. Числа Стирлинга второго рода вводятся как коэффициенты разложения

$$x^n = \sum_{k=1}^{n} b_{nk} x(x-1)\ldots(x-n+1).$$

Вычислить $b_{n1}, b_{n2}, \ldots, b_{nn}$.

12. Вычислить сумму

$$S = \sum_{j=1}^{n} (j^3 + 3j^2 + 2j)\, a[j]\,,$$

используя только сложения (в количестве $4n - 1$ операций).

Литература

[1] *Clenshaw C.* A note on the summation of Chebyshev series // Math. Tabl. Aids. Comput. 1955. V. 9. P. 118–120.

[2] *Goertzel G.* An algorithm for the evaluation of finite trigonometric series // Amer. Math. Monthly. 1958. V. 65. P. 34–35.

[3] *Mikloško J.* The numerical computation of three-term recurrence relations and the tridiagonal system of linear equations by the method of shooting // Журн. вычисл. мат. и матем. физ. 1974. Т. 14. №6. С. 1371–1377.

[4] *Хинчин А.Я.* Цепные дроби. Изд. 4-е. М.: Наука, 1978. 112 с.

[5] *Скоробогатько В.Я.* Теория ветвящихся цепных дробей и ее применение в вычислительной математике. М.: Наука, 1983. 312 с.

[6] *Романовский И.В.* Дискретный анализ. Изд. 3-е. СПб: Невский диалект, 2004. 320 с.

[7] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. Пер. с англ.. М.: МЦНМО, 2002. 960 с.

[8] *Кнут Д.Э.* Искусство программирования. Т. 2. Пер. с англ. Изд. 3-е. М.-СПб-Киев: Вильямс, 2000. 828 с.

[9] *Малоземов В.Н., Певный А.Б.* Рекуррентные вычисления. Л.: Изд-во ЛГУ, 1976. 56 с.

[10] *Воронин А.В., Кузнецов В.А., Корзун Д.Ж.* Командные чемпионаты по программированию: организация, задачи и решения. Петрозаводск: Изд-во ПетрГУ, 2001. 268 с.