ՀԱՅԱՍՏԱՆԻ ԳԻՏՈՒԹՅՈՒՆՆԵՐԻ ԱԿԱԴԵՄԻԱՅԻ ՏԵՂԵԿԱԳԻՐ
ИЗВЕСТИЯ АКАДЕМИИ НАУК АРМЕНИИ

Մաթեմատիկա XXVI, № 4, 1991 Математика

КРАТКИЕ СООБЩЕНИЯ

УДК 519.64

С. Г. КАРАДЖЯН

ВЕКТОРИЗАЦИЯ АЛГОРИТМОВ РЕШЕНИЯ НЕКОТОРЫХ ИНТЕГРАЛЬНЫХ УРАВНЕНИЙ

Введение

Как известно [1, 2], наиболее распространенным методом решения интегрального уравнения Фредгольма второго рода

$$y(x)=\int_0^T K(x,\ t)\ y(t)\ dt+f(x);\ 0\leqslant x\leqslant 1 \tag{1}$$

в случае гладкого ядра является способ приближенного сведения этого уравнения с помощью квадратурных формул к алгебраической системе с $N\times N$-матрицей (N — число точек дискретизации). При этом численная реализация этой схемы требует, по меньшей мере, памяти объемом $O(N^2)$ и числа операций порядка $O(N^3)$ $(N\to\infty)$.

В случае разностного ядра $K=K(x-t)$ эти показатели существенно снижаются: требуется память объемом $O(N)$ при числе операций порядка $O(N^2)$ (см. [3]). Этих же показателей можно достичь и в случае „почти разностных" ядер или ядер разностно-суммарных ([4, 5]).

В работе [6] был обнаружен более общий класс ядер, удовлетворяющих уравнению

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial t^2}\right)K(x,\ t)=0, \tag{2}$$

допускающий, в принципе, такие же показатели численной реализации решения уравнения [1]. На основе подхода, разработанного в [7], аналогичные показатели могут быть получены и для ядер K, удовлетворяющих уравнению четвертого порядка (см. [8])].

В настоящее время большое внимание уделяется (см. [3]) реализации алгоритмов на многопроцессорных системах. Для алгебраических систем с матрицей $N\times N$ метод Гаусса—Жордана позволяет реализовать параллельный счет на N^2 процессорах за N тактов. В теплицевом случае (т. е. при $K=K(x-t)$) могут быть задействованы уже $O(N)$ процессоров при тех же N тактах ([3]), при этом оказывается необходимым кардинально преобразовать алгоритмы.

Ниже аналогичные показатели получены, в принципе, для ядра [2]. Отметим, что здесь не затрагиваются вопросы максимальной эко-

номичности построенных для случая ядра (2) алгоритмов или (в случае нарушения устойчивости) способы повышения устойчивости.

1°. Будем исходить из формул, полученных в [6]. На плоскости (τ, x) введем сетку

$$\tau_j = jh;\ x_i = ih;\ 0 \leqslant i \leqslant j \leqslant N;\ N = [T/h].$$

В узлах сетки будут рекуррентно вычисляться функции

$$r_1^0(x_i;\ \tau_j) \overset{\text{def}}{=} r_2^0(i,\ j);\ r_1^0(x_i,\ \tau_j) = r_1^0(i,\ j),$$

$$r_1^1(x_i,\ \tau_j) = r_1^1(i,\ j);\ r_1^2(x_i,\ \tau_j) = r_1^2(i,\ j);\ y(x_i,\ \tau_j) = y(i,j).$$

Приведем схему реализации алгоритма А, приближенно решающего уравнение (1) с ядром (2).

Начальный этап: $0 \leqslant j \leqslant 7$. Для всех $0 \leqslant i \leqslant j \leqslant 7$, в узлах (i, j) вычисляются значения функций $r_2^0(i,\ j)$, $r_1^0(i,\ j)$, $y(i, j)$; $p = 0, 1, 2$ как решения систем уравнений, полученных посредством дискретизации соответствующих интегральных уравнений (см. [6]). На этом этапе структура ядра не учитывается.

Рекуррентный шаг: Для $i = j-3,\ j-2,\ j-1,\ j$;

$$r_2^0(i,\ j) = 2r_2^0(i-1,\ j-1) + 10r_2^0(i-3,\ i-3) + r_2^0(-4,\ j-4) +$$
$$+12r_2^0(i-2, j-1) + 6r_2^0(i-3, j-2) - 18r_2^0(t-3, j-1) + 6r_2^0(i-4, j-1) +$$
$$+12r_2^0(i-4, j-2) - 6r_2^0(i-5, j-2) - 18r_2^0(i-4, j-3) + 6r_2^0(i-5, j-3) -$$
$$-12r_2^0(i-2, j-2) + 6hr_2^0(i-2, j-2)[r_2^0(j-3, j-3) + r_2^0(j-2, j-1) + \qquad (3)$$
$$+2r_2^0(j-3, j-2) - 3r_2^0(j-4, j-2) - 2r_2^0(j-2, j-2) + r_2^0(j-5, j-2)] +$$
$$+3/2\,h[r_2^0(i-1,\ j-1) - r_2^0(i-3, j-3) - 3r_2^0(i-2,\ j-2) +$$
$$+4r_2^0(i-3,\ j-2) - r_2^0(i-4,\ j-2) - 2hr_2^0(i-2, j-2)r_2^0(j-2, j-2)] \times$$
$$\times [r_2^0(j-1, j-1) - r_2^0(j-3, j-3)] + 2hr_1^0(i-2, j-2)\,[2r_2^0(0, j-2) -$$
$$-5r_2^0(1, j-2) + 4r_2^0(2, j-2) - r_2^0(3, j-2)] + h^2 r_1^1(i-2, j-2) \times$$
$$\times (3r_2^0(0, j-2) - 4r_2^0(1, j-2) + r_2^0(2, j-2)] + 2h^3 r_1^2(i-2, j-2) r_2^0(0, j-2).$$

Для $i = 2, 3, \cdots, j-4$ функция $r_2^0(i, j)$ и для $i = 0, 1, \cdots, j-2$ функции r_1^0, r_1^1, r_1^2, y вычисляются по формулам, приведенным в работе [6]. Не выписывая их здесь, обозначим их соответственно номерами (4) и (5). Для $i = 0, 1$ вычисление функции $r_2^0(i, j)$ производится следующим способом: вычисляются числа

$$D_l^j = [1 + h/2\, r_1^0(j, j)]K(l, j) + h\sum_{s=3}^{j-1} K(l, s) \cdot r_2^0(s, j);\ l = 0, 1, \qquad (4)$$

$$g_{2,0}^j = 1 - h/2 \cdot K(0, 0) - hK(1, 1) + h^2/2 \cdot K(0, 0)K(1, 1) - \qquad (5)$$
$$- h^2/2\, K(0, 1)\, K(1, 0).$$

После чего

$$r_2^0(0, j) = [D_0^j \cdot (1 - hK(1, 1)) + D_1^j h \cdot K(0, 1)]/g_{2,0}^j,$$
$$r_2^0(1, j) = [D_1^j (1 - h/2 \cdot K(0, 0)) + D_0^j \cdot h/2 \cdot K(1, 0)]/g_{2,0}^j. \qquad (6)$$

Для функций r_1^p, y в точках с $i=j-1$, j, для $p=0, 1, 2$ имеют место следующие формулы:

$$T^l_{l,p} = r_1^p(l, j) - h/2\,K(l, 0)\cdot r_1^p(0, j) - h\sum_{s=1}^{j-2} K(l, t)\,r_1^p(s, j) +$$

$$+ \frac{\partial^p K(l, t)}{\partial t^p}\Big/_{t=0}, \quad l = 0, 1,$$

$$g^l_{l,p} = h[K(0, j-1)\cdot K(1, j) - K(1, j-1)\cdot K(0, j)],$$

$$m^l_{l,p} = h/2\cdot[K(0, j)\cdot K(1, j-1) - K(1, j)\cdot K(0, j-1)], \qquad (7)$$

$$r_1^p(j-1, j) = [t^l_{0,p}\,K(1, j) - t^l_{1,p}\,K(0, j)]\cdot/g^l_{1,p},$$

$$r_1^p(j, j) = [t^l_{0,p}\,K(1, j-1) - t^l_{1,p}\,K(0, j-1)]/m^l_{1,p}.$$

Аналогичные формулы имеют место для $y(j-1, j)$ и $y(j, j)$, с той лишь разницей, что в формулах для $T^l_{l,p}$ вместо $\dfrac{\partial^p K(l, t)}{\partial t^p}\Big/_{t=0}$ нужно подставить $f(0)$ и $f(1)$.

Последний шаг: При $j=N-1$ закончить вычисления и выдать на печать значения $y(i, N-1)$, $i=0, 1, \cdots, N-1$.

Нетрудно подсчитать число операций (умножений и сложений) для алгоритма А. Оно составит величину $\Omega(N) = \frac{1}{2}\cdot(77N^2 + 429N - 6086)$.

2°. Сначала укажем на естественный параллелизм, позволяющий реализовать алгоритм А на 16 процессорах. Поясним это утверждение.

Движение алгоритма А по j — это „сугубо" последовательная часть алгоритма. Параллелизм наблюдается при реализации каждого j-ого шага

На первом этапе шага j вычисляется формула (3) и те части из формул (4) и (5), которые не зависят от i, т. е. те, которые на данном j-ом шаге вычисляются всего одии раз и помещаются в оперативиую память. Таких частей и формул 10. Для их вычислений задействуем все 16 процессоров, организовав их рациональное использование. Так как время, затрачиваемое на первом этапе, не будет зависеть от N, обозначим его через t_1.

На втором этапе j-ого шага для вычислений по формуле (4) предоставим 12 процессоров, а для вычислений по формуле (5) — 4 процессора. К следующему этапу перейдем тогда, когда будет вычислена самая длинная по времени формула, т. е. формула (4). Действительно, по формуле (4) время составит $t_2 = 5(j-5)t$, а по формуле (5) — $2(j-1)t$. Здесь t — время выполнения одной операции.

На третьем этапе организуем выполнения по формулам (6) и (7). Главной составляющей времени при вычислении по формулам (6) и (7) является время, затраченное на вычисление сумм типа D^l_l. Таких сумм 10. Для их вычислений воспользуемся схемой „сдваивания", а для

оценки времени воспользуемся теоремой Мунро—Патерсона (см. [8]). В итоге для всех сумм время составит $t_3 = \frac{5}{4}(j+25)t$.

Весь алгоритм займет время $T_{16} = \frac{25}{8} N^2 t$ (главная часть). Теперь нетрудно подсчитать к. п. б (коэффициент повышения быстродействия)

$$K_{16} = \frac{\Omega_-(N)\cdot t}{16\cdot T_{16}} = 0{,}77.$$

Как видим, эта характеристика достаточно благоприятна: при наличии 16-процессорной системы счет ускоряется более чем в 12 раз.

3°. Применение большого числа процессоров (количество $p = p(N)$ которых стремилось бы к бесконечности при $N \to \infty$ непосредственно к алгоритму А привело бы к очень малому к. п. б. ($K_p \to \infty$, $N \to \infty$) Причиной этого (см. [9]) являются выражения, содержащие скалярные произведения и обозначенные выше заглавными буквами.

Теперь осуществим абстрактное распараллеливание на $O(N)$ ($N \to \infty$ процессорах. В основе построения лежит понятие T и T'-алгоритмов (см. [3]), допускающих полную векторизацию. Основной идеей этих построений является замена скалярных произведений, т. е. N-членных сумм типа D_1^j, препятствующих непосредственному распараллеливанию,—k- членными суммами, где k не зависит от N.

Можно доказать, что алгоритм А является и T, и T'-алгоритмами (последний отличается экономичностью). Здесь приведем один из основных эпизодов доказательства этих утверждений.

Пусть $D^j = \sum_{s=2}^{j-1} K(0, s)\cdot r_1^0(s, j)$. Покажем, как векторизуется вычисление данной суммы.

Введем следующие обозначения: K_0— циркулянтная $N \times N$-матрица с первой строкой $[K(0,0)\ K(0,1), \cdots, K(0, N-1)]$; $\gamma_{s,j}$, $\beta_{s,j}$ — скалярные коэффициенты, вычисление которых осуществляется на каждом j-ом шаге за время, не зависящее от N. Через P_s, Q_s обозначим один из $N \times N$ матриц перестановок, умножение на вектор-столбец которого может быть осуществлено за время, не зависящее от N. Тогда векторизуя алгоритм А, получим следующее представление для N-мерных вектор-столбцов

$$[00\, r_2^0(2,j), \cdots, r_2^0(-j-4, j)\, 0\cdots 0]^T = \gamma_{1j} P_1 [00\, r_2^0(2, j-1)\cdots r_2^0(j-5, j-1)\, 0\cdots 0]^T + \gamma_{2j} P_2 [00\, r_2^0(2, j-2)\cdots r_2^0(j-6, j-2)\, 0\cdots 0]^T +$$
$$+ \gamma_{3j} P_3 [00\, r_2^0(2, j-3)\cdots r_2^0(j-7, j-3)\, 0\cdots 0]^T + \gamma_{4j} P_4 [00\, r_2^0(2, j-4)\cdots$$
$$\cdots r_2^0(j-8, j-4)\, 0\cdots 0]^T + \gamma_{5j} P_5 [00\, r_1^0(2, j-2)\cdots r_1^0(j-6, j-2) 0\cdots 0]^T +$$
$$+ \overline{\gamma_{6j}} P_6 [00\, r_1^1(2, j-2)\cdots r_1^1(j-6, j-2)\, 0\cdots 0]^T + \gamma_{7j} P_7 [00\, r_1^2(2, j-2)\cdots$$
$$\cdots r_1^2(j-6, j-2)\, 0\cdots 0] + \gamma_{8j} P_8 \cdot [1\ 00\cdots 0]^T;$$

Также для $p = 0, 1, 2$

$$[00\, r_2^p(2,j)\cdots r_1^p(j-4,j)\,0\cdots 0]^T = \beta_{1j}\,Q_1[00\, r_1^p(2,j-1)\cdots r_1^p(j-5),$$
$$j-1)\,0\cdots 0]^T + \beta_{2j}\,Q_2[00\, r_1^p(2,j-2)\cdots r_1^p(j-6,j-2)\,0\cdots 0]^T +$$
$$+\beta_{3j}\,Q_3\cdot[1\,00\cdots 0]^T.$$

Введем векторы

$$\varphi^j \overset{\text{def}}{=} [\varphi_0^j\,\varphi_1^j\cdots\varphi_{N-1}^j]^T = K_0\cdot[00\, r_2^0(2,j)\cdots r_2^0(j-4,j)\,0\cdots 0]^T,$$
$$\psi^{j,p} \overset{\text{def}}{=} K_0\cdot[00\, r_1^p(2,j)\cdots r_1^p(j-4,j)\,0\cdots 0]^T;\ p=0,\ 1,\ 2.$$

Так как циркулянтная матрица перестановочна с любой из матриц перестановок, то легко прийти к следующим рекуррентным соотношениям для $j=8,\ 9,\cdots,\ N-1$:

$$\varphi^j = \sum_{k=1}^{4}\gamma_{kj}\cdot P_k\cdot\varphi^{j-k} + \sum_{k=6}^{7}\gamma_{kj}\cdot P_k\cdot\psi^{j-2,\,7-k} + \gamma_{8j}\,P_8\cdot$$
$$\cdot[K(0,0)\,K(0,\ N-1)\,K(0,N)\cdots K(0,\ 1)]^T,$$

$$\psi^{j,p} = \beta_{1j}\,Q_1\,\psi^{j-1,p} + \beta_{2j}\,Q_2\,\psi^{j-2,p} + \beta_{3j}\,Q_3\cdot[K(0,0)\,K(0,N-1)\,K(0,N-2)\cdots$$
$$\cdots K(0,\ 1)]^T,\ p=0,\ 1,\ 2.$$

Легко видеть, что искомая сумма $D_N^j = \varphi_0^j$. В некотором смысле „потерей“ в процессе векторизации можно принять увеличение размера оперативной памяти, т. к. нам пришлось ввести и запомнить $4N$-мерных вектора (это ψ^j, $\psi^{j,p}$, $p=0,\ 1,\ 2$). Соответственно несколько увеличивается общее число операций, что вполне естественно (см. [3]). В итоге оказывается, что при наличии $56N$ процессоров векторизованная форма алгоритма А позволяет вести параллельный счет с линейным ускорением (с к. п. б., равным 0,22), что сопоставимо, например, с параллельной формой алгоритмов типа Левинсона—Тренча (см. [3]).

Кировоканский филиал
Ереванского политехнического
института

Поступила 17.IX 1990

ЛИТЕРАТУРА

1. *А. Ф. Верлань, В. С. Сизиков.* Интегральные уравнения: методы, алгоритмы, программы. Справочное пособие, Киев. «Наукова Думка», 1986.
2. *L. M. Delves. J. L. Mohamed.* Compatational methods for integral equations, Cambridge Univ. Press. 1985.
3. *В. В. Воеводин, Е. Е. Тыртышников.* Вычислительные процессы с теплицевыми матрицами, М., Наука, 1987.
4. *T. Kailath, L. Ljung, M. Morf.* Generalized Krein—Levinson equations for efficient calculation of Fredholm Resolvents of nondisplacement kernels, Topics in Funct. Anal, Adv. M. Suppl. St. 3, 1978.
5. *А. Б. Нерсесян, А. А. Папоян.* Построение матрицы, обратной сумме Тёплица и Ганкеля, Изв. АН АрмССР, Математика, XVIII, № 2, 1983, 950—960.

6. *С. Г. Караджян, А. Б. Нерсесян.* Быстрое решение некоторых интегральных уравнений, Деп. рук. Арм. ВНИИТИ, 1988.
7. *А. Б. Нерсесян.* О структуре резольвент некоторых интегральных операторов с ядрами, определяемыми дифференциальными уравнениями, ДАН СССР, Математика, 279, № 4, 1984, 805—809.
8. *С. Г. Караджян.* Экономичный алгоритм обращения интегрального оператора с разностно-суммарно-гармоническим ядром, Изв. АН Арм.ССР, Математика, XXIV, № 3, 1989, 300—305.
9. *Е. Валях.* Последовательно-параллельные вычисления, Мир, 1985.