УДК 681.3

ИДЕНТИФИКАЦИЯ ЭКСТРЕМУМОВ ФУНКЦИЙ НА ОСНОВЕ СОРТИРОВКИ С ПРИЛОЖЕНИЕМ К ВЫЧИСЛИТЕЛЬНЫМ СХЕМАМ АЛГЕБРЫ, АНАЛИЗА И РАСПОЗНАВАНИЮ ИЗОБРАЖЕНИЙ

Я.Е. Ромм, И.В. Заика, И.А. Тюшнякова

Таганрогский государственный педагогический институт
347936, Таганрог, ул. Инициативная, 48
тел.: (863)(44) 60 1753; факс: (863)(44) 60 5397, Romm@list.ru

Показано, что сортировка может служить единой основой для автоматической идентификации нулей и экстремумов произвольной функции одной и более переменных в произвольно фиксированной части области определения. Нули полинома вычисляются с учетом кратности, включая случай характеристического полинома матрицы. Функция может задаваться значениями на равномерной сетке. В частности, идентифицируются нули и экстремумы разностных решений обыкновенных дифференциальных уравнений и уравнений в частных производных. Для оцифрованного изображения на плоскости как дискретной функции двух переменных на сетке пиксельных элементов на этой основе строится вектор распознавания. Процесс обработки использует лишь операции сравнения, что исключает накопление погрешности, влечет высокую точность локализации экстремумов и устойчивость идентификации изображений.

It is shown that sorting can form a unique basis for automatic identification of zeroes and extremums of any function of one and more variables in any way fixed part of definitional domain. Polynomials` zeroes are calculated with consideration of multiplicity, including a case of a characteristic polynomial of a matrix. Function can be set by values on a analytical grid. In particular, zeroes and extremums of difference solutions of the ordinary differential equations and the equations in partial derivatives are identified. For digitized image on a plane as discrete function of two variables on a pixel elements' grid on this basis the recognition vector is under construction. Manufacturing process uses only comparison operations, which excludes error accumulation and implyies split-hair accuracy of localization of extremums and stability of images identification.

Постановка вопроса

Цель сообщения – показать применимость сортировки в качестве единой основы для автоматической идентификации всех нулей и экстремумов произвольной алгебраической и трансцендентной функции одной и более переменных в произвольно фиксированной части области определения. Функция, в частности, может быть задана дискретно, например, последовательностью отсчетов, значениями на равномерной сетке. В последнем случае ставится задача идентифицировать все нули и экстремумы разностных решений обыкновенных дифференциальных уравнений и уравнений в частных производных. Оцифрованное изображение на плоскости представляет собой разновидность дискретной функции двух переменных на равномерной сетке с длиной шага, равной стороне пикселя. Для такого изображения требуется на рассматриваемой основе построить идентифицирующий вектор признаков.

Схема автоматической идентификации всех экстремумов и нулей функции на основе сортировки

Пусть вначале требуется локализовать и приближенно вычислить все минимумы действительной функции одной действительной переменной $y = f(x)$ на промежутке $[\, x^{(0)}, x^{(N)} \,]$ в области ее определения. На равномерной сетке $h = \left| x^{(N)} - x^{(0)} \right| / N$, $x_\ell = x^{(0)} + \ell h$, $\ell = 0, 1, ..., N$, считываются значения функции и запоминаются как элементы массива $c[\, i \,] = f(x_{i-1})$, $i = 1, 2, \ldots, N$. Элементы этого массива сортируются. Для простоты описания используется сортировка подсчетом в модификации из [1, 2]. Массив C из N элементов, располагается горизонтально сверху и вертикально слева от матрицы сравнений (МС). Например, для массива $C = (1, 3, 5, 2, 8)$ МС примет вид

	1	3	5	2	8
1	0	+	+	+	+
3	–	0	+	–	+
5	–	–	0	–	+
2	–	+	+	0	+
8	–	–	–	–	0

Чтобы произвести вставку в отсортированный массив j-го элемента входного массива, достаточно подсчитать число нулей и плюсов в j-м столбце над диагональю, включая диагональный элемент, и сложить это число с числом плюсов ниже диагонали. Сумма накапливается в счетчик по k, значение k становится

© K. Georgiev, E. Donev, 2006

адресом вставки: $c1[k] := c[j]$, при этом запоминается входной индекс вставленного элемента $e[k] := j$ (обратный адрес). На выходе сортировки $c1[k]$ и $e[k]$ имеют равный номер k. Оценки сложности и параллелизм обсуждаются в [1, 3], программа в примере описана далее в процедуре SORT.

В качестве сортировки можно использовать любую сортировку со свойствами адресности и обратной адресности, например, быструю сортировку ($T = O(N \log_2 N)$) на основе слияния по МС [4]. Последняя используется на практике во всех описываемых далее алгоритмах [1–4, 5, 6] для ускорения работы компьютера.

К выходу процедуры сортировки подсоединяется условный оператор, локализующий после выполнения сортировки все минимумы среди этих значений [1, 2], – k:=1; while k<= n do begin FOR l := 1 TO k-1 do if abs(e[k]-e[k-l]) <= eps0/h then goto 22; xk:= x0+e[k]*h;
22: k:=k+1; end.

Здесь n – число узлов, совпадающее с числом сортируемых элементов; h – шаг сетки; $eps0$ – константа, равная радиусу $eps0$-окрестности текущего узла, радиус выбирается априори таким образом, чтобы не превысить половины числа узлов, отсчитываемых вдоль оси между ближайшими друг к другу минимумами функции $y = f(x)$. Оператор для текущего узла $e[k]$ находит каждый узел $x0 + e[k] * h$, такой что значение функции в этом узле не больше значений в остальных узлах $eps0$-окрестности. Значение локализованной абсциссы точки минимума $xk := x0 + e[k] * h$ дает привязку к локализуемой точке минимума, она фиксируется и выполняется спуск к наименьшему значению в окрестности локализованной точки. Он осуществляется с помощью выбора наименьшего значения в сужаемой окрестности с фиксированным числом элементов в сетке, пока не будет достигнута требуемая точность. Более детально смысл этого условия обсуждается в [4–6]. Там достаточно детально описан его перенос на случай функций двух и трех переменных. Условие локализации максимумов в рамках данного метода задается операторами k:=1; while k<= n do begin FOR l := 1 TO n-k do if abs(e[k]-e[k+l]) <= eps0/h then goto 222; xk:= x0+e[k]*h; 222: k:=k+1; end.

Инвариантность метода относительно длинны промежутка достигается путем смещения базового промежутка с числом узлов N на расстояние, равное его длинне. Во избежание ложных экстремумов отсекаются обе границы текущего промежутка, проверка границ на экстремумы производится с помощью сортировки. Эта сортировка локализует экстремумы по ранее описанной схеме для элементов массива, образуемых значениями исследуемой функции слева от правой границы и справа от левой границы. В результате достигается инвариантность схемы относительно размеров промежутка, а в общем случае, – области поиска экстремумов. По построению схема инвариантна относительно вида функции, на которую не накладывается математических ограничений. Функция, в частности, может быть задана таблично, при этом спуск окажется излишним и локализация экстремумов равносильна его идентификации. При этом удается достигнуть высокой точности идентификации экстремумов. Например, все минимумы функции $y = \sin(x)$ на $[-30, 30]$ в результате работы программы, листинг которой представлен в [3], примут значения:

значения аргументов	значения минимумов
-2.67035375557E+0001	-1.00000000000000E+0000
-2.04203522486E+0001	-1.00000000000000E+0000
……..	……
2.98451302089E+0001	-1.00000000000000E+0000

Для локализации и вычисления нулей функции достаточно на вход сортировки подать абсолютные величины ее значений на равномерной сетке $c[i] = \left| f(x_{i-1}) \right|$, $i = 1, 2, \ldots, n$, и искать минимумы выше описанным способом. Та же программа [3] локализует и вычислит, например, все нули полинома $P = P_1 \times P_2 \times P_3$, где

$$P_1 = (x-1)(x-2.99)(x-3)(x-4)(x-5)(x-6.001)(x-6.002)(x-7),$$

$$P_2 = (x-8.23)(x-9)(x-10.976)(x-11)(x-12)(x-13)(x-14)(x-15),$$

$$P_3 = (x-16)(x-17)(x-18)(x-19)(20.098)(x-21)(x-22.12345)(x-23)$$

в виде

значения аргументов	значения нулей
1.00000000000000000E+0000	0.00000000000000E+0000
3.00000000000000000E+0000	0.00000000000000E+0000
……..	……
2.21234500000000000E+0001	0.00000000000000E+0000
2.30000000000000000E+0001	0.00000000000000E+0000

Следует отметить, что изложенная схема вычисления нулей полинома достраивается до схемы вычисления его нулей с учетом кратности [4–6].

Предложенная схема идентифицирует все нули и экстремумы функций общего вида, включая трансцендентные.

Без принципиальных затруднений схема переносится на случай вычисления нулей и экстремумов функций двух и более переменных [3–6].

Идентификация нулей и экстремумов разностных решений дифференциальных уравнений

По данной схеме можно идентифицировать все экстремумы разностного решения системы обыкновенных дифференциальных уравнений. Для простоты рассматривается задача Коши для одного уравнения

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0 .$$

На промежутке $[x_0, x_n]$ вычисляется разностное приближение, для определенности выбран метод Эйлера:

$$y_{i+1} = y_i + f(x_i, y_i)h, \; x_i = x_0 + ih, \; i = 0, 1, ..., n, \; h = \frac{x_n - x_0}{n} .$$

Разностные значения принимаются за элементы массива

$$c[i] = y_{i-1}, \; i = 1, 2, ..., n,$$

который сортируется. Присоединение условия локализации минимумов (максимумов) к программе сортировки влечет устойчивую идентификацию экстремумов приближенного решения рассматриваемой задачи. Инвариантность схемы относительно длины промежутка достигается вышеописанным способом (с использованием сортировки и повторной локализации на границах смещаемого промежутка).

Для случая системы дифференциальных уравнений схема применяется к каждому уравнению в отдельности. Детальное описание и другие примеры на случай системы приводятся в [3–5]. Для идентификации нулей достаточно на вход сортировки подать абсолютные величины разностных значений решения.

Идентификация всех нулей и экстремумов разностных решений уравнений в частных производных

Описанная схема переносится на случай идентификации всех нулей и экстремумов разностных решений уравнений в частных производных. Пусть, например, рассматривается задача о свободных колебаниях струны:

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2}, \tag{1}$$

где краевые и начальные условия заданы соответственно в виде

$$u\big|_{x=0} = 0, \; u\big|_{x=l} = 0, \tag{2}$$

$$\frac{\partial u}{\partial t}\Big|_{t=0} = \psi(x), \tag{3}$$

$$u\big|_{t=0} = \phi(x). \tag{4}$$

Пусть задача определена в области $(0, l)\times(0, T)$, для которой строится равномерная прямоугольная сетка:

$$x_i = ih, \; h = l/n, \quad i = 0, 1, ..., n; \; t_j = j\tau, \; \tau = T/m, \quad j = 0, 1, ..., m .$$

Обозначив u_i^j значения сеточной функции, приходим к разностной схеме

$\frac{u_i^{j-1} - 2u_i^j + u_i^{j+1}}{\tau^2} = a^2 \frac{u_{i-1}^j - 2u_i^j + u_{i+1}^j}{h^2}$. Пусть $\tilde{y} = a^2 \frac{\tau^2}{h^2}$, тогда

$$u_i^{j+1} = -u_i^{j-1} + \tilde{y}u_{i-1}^j + (2 - 2\tilde{y})u_i^j + \tilde{y}u_{i+1}^j, \tag{5}$$

где $i = 1, 2, \ldots, n-1$; $j = 1, 2, \ldots, m-1$. Дополнив (5) значениями на нулевом слое

$$u_0^0 = 0, \; u_1^0 = \phi_1, ..., \; u_{n-1}^0 = \phi_{n-1}, \; u_n^0 = 0$$

и значениями $u_0^j = 0, \; u_n^j = 0$ на границе в соответствии с условиями (2), (4) и аппроксимируя производную в (3) для подсчета u_i^1 на первом слое, получаем явную формулу второго порядка точности для вычисления

недостающих значений на первом слое: $u_i^1 = \tau\varphi_i + (1-\tilde{y})\phi_i + \frac{1}{2}\tilde{y}(\phi_{i-1}+\phi_{i+1})$, $i = 1,2,...,n-1$. Пусть каким-либо способом осуществлен процесс вычислений (5) при фиксированных n и m (или h и τ), т.е. получена таблица значений u_i^j – приближенный каркас решения $u(x_i, t_j)$. Известно [7], что условие $\tilde{y} < 1$ (или $\tau < \frac{h}{a}$) обеспечивает устойчивость схемы и сходимость решения $\left\| u_i^j - u(x_i, t_j) \right\| \to 0$ при $h \to 0, \tau \to 0$ в условиях (2) – (4).

Как только сформирована таблица значений u_i^j, она интерпретируется как дискретная функция двух переменных x_i, t_j, которая стандартно поступает на вход метода, и для нее без принципиальных затруднений идентифицируются все искомые нули и экстремумы. Более детально схема описывается непосредственно далее, затем приводится программный пример.

Для нахождения минимумов полученной сеточной функции выполняется проход в направлении оси Ot вдоль i-го столбца прямоугольной сетки, во время которого находится минимальное по строкам значение $c[i] = \min\limits_{\substack{i=\text{const} \\ 1 \le j \le m}} u_i^j$, этот минимум заносится на вход сортировки как i-й элемент сортируемого одномерного массива. К выходу процедуры подсоединяется оператор локализации минимума. Оператор идентифицирует каждый узел $e[k]*h$, в проекции $eps0$ – окрестности которого на OX нет узлов, доставляющих значения элементов, предшествующие $c[e[k]]$ в отсортированном массиве.

Значение локализованной абсциссы точки минимума $xk := e[k]*h$ дает привязку к локализуемой точке двумерного минимума, оно фиксируется и аналогичным образом локализуется ордината tk, в которой идентифицируется минимум значения сеточной функции (5).

Пример 1. Пусть дано уравнение (1) при $a = 0.009$ с краевыми и начальными условиями (2) – (4) вида:

$$U\big|_{x=0} = 0,\ U\big|_{x=l} = 0,\ U\big|_{t=0} = \sin(x),\ \frac{\partial U}{\partial t}\Big|_{t=0} = 0.$$

Требуется найти все локальные минимумы в окрестностях радиуса $eps0 = \pi$ разностного решения данного волнового уравнения в области $(0,l)\times(0,T)$, где $l = 40, T = 40$ и $n = 1100, m = 2200$. Минимумы идентифицирует следующая программа (Object Pascal, Delphi):

```
  PROGRAM extrem;
{$APPTYPE CONSOLE}
uses  SysUtils;
LABEL 22,23;
Const eps0=pi; n=1100; m=2200;  n00=2500; h=40/(n); tau=40/(m); a=0.009;
TYPE vect=ARRAY [0..n00] OF extended;
vect1=ARRAY [0..n00] OF longint;
type matr=array[0..m,0..m] of extended;
VAR     i,j,k,k1,r,nn0,xk,tk : longint; c1,c: vect; e, e1: vect1; x0,min: extended; u: matr;
Procedure uuu(Var u:matr);
var i,j:longint; fi,psi:vect; x,yy:extended;
begin
yy:=sqr(a)*(sqr(tau) / sqr(h)); x0:=0; u[0,0]:=0;u[n,0]:=0;
for i:=1 to n-1 do begin x:=(x0+i*h); u[i,0]:=sin(x);end;
fi[0]:=0;fi[n]:=0;
for i:=1 to n-1 do    begin x:=x0+i*h; psi[i]:=0; fi[i]:=sin(x);
u[i,1]:=tau*psi[i]+(1-yy)*fi[i]+0.5*yy*(fi[i-1]+fi[i+1]);end;
for j:=1 to m do begin u[0,j]:=0;u[n,j]:=0;end;
for i:=1 to n-1 do for j:=1 to m-1 do begin
u[i,j+1]:=-u[i,j-1]+yy*u[i-1,j]+(2-2*yy)*u[i,j]+yy*u[i+1,j]; end;end;
function func( var i,j:integer):extended;
begin  func:= {abs}(u[i,j]); end;
PROCEDURE mint {maxt} (VAR i,nn0:integer;var min {max}:extended);
var j:integer;
BEGIN  nn0:=m;j:=1; min{max}:=func(i,j);  FOR j:=1 TO nn0 DO
IF min > func(i,j) {max< func(i,j)}THEN BEGIN min:=func(i,j) {max:=func(i,j)};  END; END;
procedure SORT (var nn0: integer; var c: vect; var e: vect1);
var i,j,k:integer;
begin
FOR j := 1 TO nn0 do begin k:=0;
```

```
FOR i := 1 TO j do
IF c[j] - c[i] >= 0 THEN k:=k+1;
FOR i := j+1 TO nn0 do
IF c[j] - c[i] > 0 THEN k:=k+1; c1[k]:=c[j]; e[k] := j; end; end;
BEGIN
nn0:=n;  uuu(u);
FOR i:=1 TO nn0 DO  BEGIN mint (i,nn0,min);c[i]:=min; END;
SORT ( nn0, c, e);
k:=1;  WHILE k<= nn0 DO  BEGIN
FOR r := 1 TO k-1 DO IF abs(e[k]-e[k-r]) <= eps0/h THEN GOTO 23;
 xk:= e[K];  nn0:=m;
FOR j:=1 TO nn0 DO c[j]:=func(xk,j);
SORT ( nn0, c, e1);
k1:=1; WHILE k1<= nn0 DO BEGIN
FOR r := 1 TO k1-1 DO IF abs(e1[k1]-e1[k1-r]) <=eps0/h THEN GOTO 22;
tk:= e1[K1];
writeln ('    ', xk,'   '); writeln ('    ', tk,'    ', func(xk,tk));  writeln;
22: k1:=k1+1;  END;
23: k:=k+1;  END; writeln; readln;
END.
```

Результат работы программы:

```
216                                          ……….
2095   -1.01E+0000                             562
216                                          2096   -1.01E+0000
699   -1.00E+0000                            562
                                             699   -1.00E+0000
```

Для идентификации всех нулей достаточно в той же программе на вход метода подать $f(i,j) = |u_i^j|$ (достаточно снять комментарий в подпрограмме *func*). Получится:

```
1100                                         ……..
1    0.00E+0000                              257
994                                          445    1.98E-0006
350    1.54E-0007                            816
994                                          1740    4.04E-0006
1746    7.37E-0006                           354
                                             331    5.15E-0006
```

Для идентификации локальных максимумов в окрестностях того же радиуса производится замена оператора локализации минимума на оператор локализации максимума и замена процедуры поиска минимума *mint* новой процедурой поиска максимума *maxt*:

```
PROCEDURE maxt (VAR i,nn0:integer;var max:extended);var j:integer;
begin nn0:=m; j:=1; max:=func(i,j); FOR j:=1 TO nn0 DO IF max< func(i,j) then begin max:=func(i,j); end;
end;
```

Видоизмененная таким образом программа *extrem* идентифицирует все максимумы разностного решения рассматриваемого уравнения.

Результаты работы программы:

```
2200                                         ……
2200    0.00E+0000                           475
994                                          2095    1.01E+0000
699    1.00E+0000                            648
994                                          699    1.00E+0000
2096    1.01E+0000                           648
                                             2095    1.01E+0000
```

В общем случае, предложенная схема идентифицирует все локальные и глобальные экстремумы разностных решений уравнений в частных производных второго порядка на прямоугольной сетке произвольной размерности. Схема может быть перенесена на уравнения более высокого порядка.

Локализация и вычисление собственных значений матриц на основе сортировки

Собственные значения матрицы

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

вычисляются по описанной схеме как нули характеристического полинома $|A-\lambda E|=(-1)^n(\lambda^n+p_1\lambda^{n-1}+...+p_n)$. Особенности этой схемы для случая комплексных нулей полиномов детально описаны в [1-4], полный листинг программ с учетом случая кратных нулей дан в [6]. Его коэффициенты предварительно определяются по методу Леверье [8]: решается система уравнений Ньютона: $S_k+S_{k-1}p_1+S_{k-2}p_2+...+S_1p_{k-1}=-kp_k$, где $k=\overline{1..n}$, $S_k=Sp(A^k)$ – след матрицы A^k. Полином $P_n(\lambda)=(-1)^n(\lambda^n+p_1\lambda^{n-1}+...+p_n)$ ($\lambda=x+Iy$) задан своими коэффициентами, для формирования функции $f(x,y)$, которая подается на вход метода, необходимо выделить действительную и мнимую части рассматриваемого полинома. Для этого используются специальные процедуры, реализуюшие комплексную арифметику. В результате $f(x,y)=(\mathrm{Re}\,(P_n(z)))^2+(\mathrm{Im}(P_n(z)))^2$. В случае отсутствия процедур комплексной арифметики для формирования входной функции можно использовать бином Ньютона степени $z^l=(x+y\cdot I)^l$, коэффициенты бинома определяются по треугольнику Паскаля и приводятся подобные при действительной и мнимой части [2, 4].

Если коэффициенты характеристического полинома вычислены достаточно точно, то предложенный метод вычисления собственных значений матриц оказывается устойчивым в виду фактически верного (с точностью до формата представления данных в языке Object Pascal) вычисления нулей полиномов.

Схему нахождения нулей характеристического полинома с учетом кратности можно дополнить до построения полной нормальной жордановой формы матрицы [6].

Приложение метода к распознаванию плоских изображений

Описанная схема вычисления собственных значений матриц применяется для распознавания как двуцветных, так и цветных плоских многосвязных изображений в каноническом расположении в декартовых координатах. Излагаемый метод, в отличие от известных [9], исключает этап выделения ключевых признаков, этап предобработки изображения включает в себя лишь укрупнение исходной матрицы (масштабирование), после чего в качестве механизма классификации выступает схема вычисления собственных значений оцифрованной матрицы пикселей, описанная выше.

Первоначально будем рассматривать входные изображения небольшого размера: 16x16, 32x32 или 64x64 пиксела. Модификация метода для изображений большего размера излагается в дальнейшем.

Распознавание двуцветного изображения. Рассмотрим класс изображений, каждый пиксел который после оцифровки может принимать значение 0 или 1, т. е. изображение является черно-белым или любым двуцветным. В данном классе изображений одним из типов объектов, распознавание которых представляет практическую ценность, являются, в частности, лингвистические символы. Их распознавание может использоваться для ускорения ввода в компьютер рукописного текста, иероглифов и любых символов. Суть метода заключается в следующем.

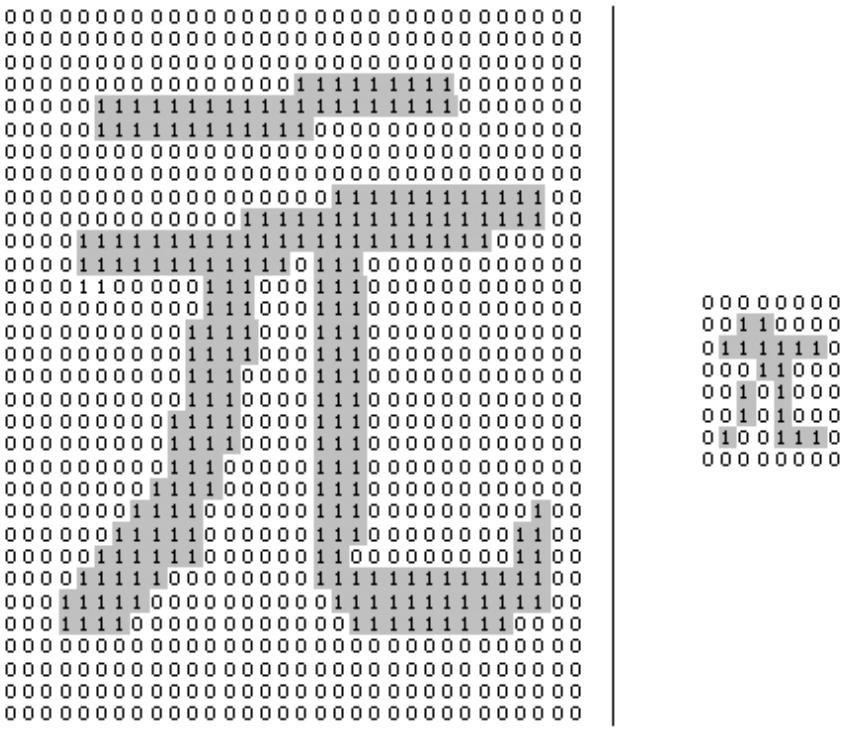
Для оцифрованной квадратной матрицы пикселей плоского двухцветного изображения вычисляется набор собственных значений с учетом кратности. В случае прямоугольного изображения матрица достраивается до квадратной путем дописывания нулевых строк и столбцов. Вектор распознавания состоит из диагональных клеток жордановой формы матрицы. Это влечет возможность квадратичного сжатия изображения и определяет выбор эталонного вектора для его идентификации. Численный эксперимент показал, что в качестве вектора распознавания на практике достаточно использовать вектор собственных значений матрицы с учетом кратности, без построения клеток Жордана. На этом основании далее не учитываются элементарные делители и инвариантные множители оцифрованной матрицы изображения.

Оговорок требует представление входной матрицы. Клетки изображения целесообразно укрупнить для идентификации существенных деталей (рис. 1).

Клетка полагается закрашенной при условии, что число закрашенных в ней пикселей превосходит некоторое априори заданное значение. Такой вариант укрупнения позволяет избавиться от случайных помех, но не дает возможность однозначно идентифицировать конкретный фрагмент изображения с учетом каждого пиксела. Возможен другой вариант укрупнения, учитывающий все детали изображения, он описан далее для цветного изображения.

Укрупненная матрица B, нормируется для уменьшения значений следов ее степеней. Целесообразно применить умножение траспонированной матрицы на исходную матрицу: $B^T\times B=C$. Это ускорит работу программы, так как результатом произведения является положительно-определенная матрица C, имеющая только действительные неотрицательные собственные значения [8], вследствие чего нахождение нулей характеристического полинома матрицы C производится на основе программы вычисления действительных нулей полиномов. Вектор распознавания строится из собственных значений матрицы C, упорядоченных по неубыванию.

Видоизменение вектора распознавания, состоящего из собственных значений матрицы C, может быть основано на взаимно однозначном соответствии нулей характеристического полинома его коэффициентам. Коэффициенты, в свою очередь, однозначно определяются следами всех степеней матрицы в силу единственности решений треугольной системы уравнений Ньютона. Отсюда вектор распознавания может строиться по числовым значениям следов степеней этих матриц.



а б

Рис. 1. Матрица пикселей символа «начало»: а – исходная; б – укрупненная

Таким образом, имеется возможность построения трех типов вектора распознавания (набор собственных значений, набор коэффициентов характеристического полинома, набор следов степеней матриц) и они эквивалентны. С точки зрения быстродействия, наиболее целесообразно использовать набор следов n степеней исходной матрицы. Например, вектор распознавания, состоящий из следов, для символа, изображенного на рис. 1.

(0.5, 0.12037037037037, 0.0366512345679012, 0.0118312757201646, 0.00387469516841945, 0.00127375344523193, 0.000419162717792286, 0.00013797674134256).

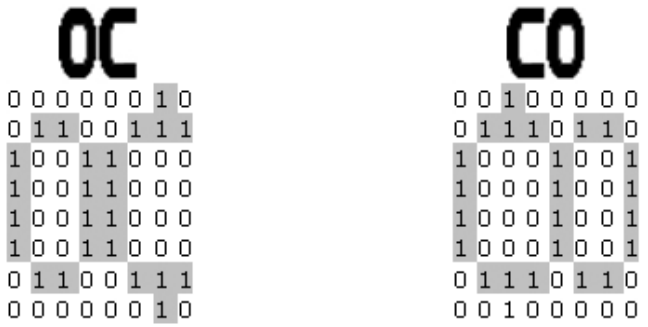
Длительность работы в случае матриц большого порядка может быть компенсирована параллелизмом метода, основанного на алгоритме Ксанки [10] параллельного вычисления n степеней матрицы.

На основе описанного метода возможно распознавание произвольного геометрического места точек, в том числе сочетаний букв и слов целиком. Как в данном, так и в ранее описанном классе задач возникает следующая трудность. Так как подобные матрицы имеют одну и ту же жорданову форму [8], а, следовательно, одинаковый набор собственных значений и следов степеней матриц, то различные изображения могут быть идентифицированы с помощью описанной схемы как тождественные. Например, изображение сочетания букв «ОС» и «СО» размерностью 32 х 32 пикселя после укрупнения представляется подобными матрицами (рис. 2).

Векторы распознавания для изображений совпадают: (0.96, 0.4096, 0.18432, 0.08388608, 0.0383778816, 0.017632854016, 0.0081335943168, 0.0037658273251328).

Идентификация изображений, представимых подобными матрицами. Для однозначной идентификации изображений, матрицы которых подобны, предлагается следующая модификация схемы распознавания.

В дополнении к алгебраическому методу применяется вспомогательная схема идентификации экстремумов (например, максимумов) дискретной функции двух переменных. Задается промежуток с границами $[1, np] \times [1, np]$ (где np – размерность укрупненной матрицы B) и строится равномерная прямоугольная сетка с шагом $h = 1$. Функция задается в узлах этой сетки, причем $f(i, j) = B_{ij}$. Максимумы $f(x, y)$ находятся по вышеописанной (для уравнений в частных производных) схеме нахождения экстремумов дискретно заданной функции двух переменных (с той оговоркой, что вычислять значения входной сеточной функции не требуется).



а б

Рис. 2. Укрупненная матрица пикселей изображения: а – ОС; б – СО

Следует принять во внимание, что схема идентификации экстремумов разностного решения уравнения в частных производных, по существу, не отличается от рассматриваемой схемы идентификации экстремумов плоского изображения. Более того, можно было бы ограничиться только этой схемой распознавания данных изображений. Однако при этом идентифицирующие векторы имели бы различное число компонент, что повлекло бы трудности при сравнении с эталонами. Напротив, кроме равенства числа компонент идентифицирующих векторов, предложенная ранее схема идентификации на основе собственных значений входной матрицы изображения с очевидностью обладает свойством сжатия N^2 исходных элементов до значения $O(N)$.

В результате вектор распознавания в общем случае строится как набор следов всех *n* степеней матрицы и он дополняется отсортированным набором длин проекций радиус-векторов локализованных максимумов функции $f(x,y)$, описывающей матрицу изображения. В частности, изображениям сочетания букв «ОС» и «СО» соответствует различное число максимумов, что указывает на их различие при совпадении следов степеней матрицы. Окрестность локализации при этом выбрана в виде $eps0 = np\ div\ 4$.

В качестве набора дополнительных идентификаторов изображения не обязательно использовать длинны проекций радиус-векторов максимумов функции $f(x,y)$, в их роли могут выступать индексы положения этих максимумов внутри матрицы, можно также рассматривать подстановки из индексов [1].

Следует отметить, что имеет место сходимость описанного метода, основанная на следующих соображениях. При итеративном сужении окрестности локализации $eps0$ найдется определенная окрестность радиуса $\overline{eps0}$, начиная с которой поиск экстремумов будет давать один и тот же результат. Это обуславливается тем, что $\overline{eps0}$ не превышает половины расстояния между проекциями соседних экстремумов.

Поскольку на вход метода может поступать любое геометрическое место точек, то описанный метод применим для распознавания отпечатков пальцев. В отличие от известных [1], он не требует высокого качества исходного изображения и выделения особых точек, минимально использует математический аппарат. С помощью данного метода возможно гарантированное распознавание отпечатков пальцев в каноническом расположении, т. е. без учета смещения и поворота пальца, изменения давления, изменения качества поверхности кожи и т.д. Следует отметить, что все известные методы распознавания отпечатков пальцев также зависят от этих факторов, т.е. качество изображения папиллярного узора пальца является одним из основных критериев, от которого зависит распознавание и в конечном итоге идентификация человека. Хранение вектора распознавания занимает небольшой объем памяти, поэтому для идентификации конкретного человека возможно сравнение его отпечатка с набором нескольких эталонных, снятых под разными углами и с различным давлением.

Для применения данного метода изображение отпечатка пальца, введенное в память какого-либо считывающего устройства, сохраняется в виде цифрового эталона, размер которого совпадает с количеством элементов вектора распознавания (дополненного при необходимости набором соответственных максимумов). Поскольку эталоны отпечатков хранятся в памяти в виде цифровых векторов, это полностью исключает возможность восстановления из них реального изображения отпечатков. К тому же фактическое изображение отпечатка пальца при контакте со считывателем необязательно сохранять и запоминать, что немаловажно для инфомационной безопасности.

Распознавание цветного изображения. Метод распознавания двуцветного изображения, вышеописанный, можно распространить на цветное изображение. Рассмотрим класс изображений, состоящих из ограниченного количества цветов, например восьмицветные изображения.

Оцифровка матрицы пикселей происходит следующим образом: каждому пикселю изображения $am[i,j]$ присваивается в зависимости от его окраски определенное число от 0 до 7. Один из возможных вариантов укрупнения ячеек входной матрицы заключается в следующем. Будем подсчитывать сумму

произведений элементов текущей ячейки (слева направо сверху вниз) и степеней $8^0, 8^1, ...8^{hm^2}$ (hm – числовой параметр, кратный n, и определяющий размер укрупнения ячеек матрицы). Таким образом, часть исходного изображения до укрупнения, т.е. конкретная ячейка матрицы AM, представляется числом в восьмерочной системе счисления; после укрупнения значение элемента матрицы B, соответствующее данной ячейке, представляется тем же числом в десятичной системе счисления. Для фрагмента исходной матрицы и $hm = 4$ получаем

$$\begin{pmatrix} am_{11} & am_{12} & am_{13} & am_{14} & am_{15} & \dots \\ am_{21} & am_{22} & am_{23} & am_{24} & am_{25} & \dots \\ am_{31} & am_{32} & am_{33} & am_{34} & am_{35} & \dots \\ am_{41} & am_{42} & am_{43} & am_{44} & am_{45} & \dots \\ am_{51} & am_{52} & am_{53} & am_{54} & am_{55} & \dots \\ \dots & \dots & \dots & \dots & \dots & \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & \dots & \dots \\ b_{21} & b_{22} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix},$$

$$b_{11} = am_{11} \cdot 8^0 + am_{12} \cdot 8^1 + am_{13} \cdot 8^2 + am_{14} \cdot 8^3 + am_{21} \cdot 8^4 + am_{22} \cdot 8^5 + ... + am_{44} \cdot 8^{16},$$

аналогично b_{11} определяются остальные элементы b_{ij} в правой части равенства.

Вследствии единственности представления одного и того же числа в различных системах счисления, можно утверждать, что только абсолютно идентичные части изображений будут представимы одним и тем же числом. Остальные этапы распознавания изображения, включая нормирование укрупненной матрицы и вычисление следов, аналогичны случаю двуцветного изображения. Вектором распознавания также служит вектор, состоящий из числовых значений следов матриц.

Описанный метод возможно перенести на любое количество цветов, используя процесс сегментации. Под сегментацией, в широком смысле, понимают преобразование полутоновых или цветных изображений в изображение, имеющее меньшее число цветов или тонов, чем исходное [9]. Применение известных методов сегментации позволит использовать описываемый метод для произвольных цветных изображений.

Следует отметить, что укрупнение ячеек оцифрованной матрицы изображения описанным способом приведет к распознаванию изображений, идентифицирующихся до точного значения каждого пикселя.

Для избавления от случайных помех можно использовать второй вариант укрупнения ячеек исходной матрицы, который заключается в определении доминирующего цвета путем подсчета числа пикселей каждого цвета в текущей ячейке. Числовая характеристика получившегося цвета является элементом укрупненной матрицы B, соответствующим текущей ячейке матрицы.

Распознавание изображений произвольного размера. В большинстве прикладных задач распознаваемые объекты имеют большие общепринятые размеры, например, 512 x 512 пикселей. Матрицу произвольного размера можно свести к матрице необходимого размера путем разбиения на клетки. Размер клетки устанавливается на основании численного эксперимента в конкретной предметной области (как правило, это 4 x 4 или 2 x 2, – при необходимости недостающее количество строк или столбцов можно заполнить нулями). Вся исходная клетка разбивается на вложенные друг в друга клетки такого же размера. Число вложений неограниченно до момента точной идентификации всех элементов такой клетки. Схематически этот итеративный процесс можно изобразить следующим образом (рис. 3).

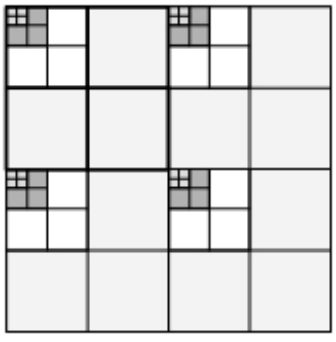


Рис. 3. Схема процесса укрупнения клеток матрицы изображения

Пример 2. Рассмотрим двуцветное изображение (64 x 64 пикселя) (рис. 4).

Рис. 4. Иероглиф «Сотня»

Разобъем оцифрованную матрицу изображения (64x64) на клетки размерностью 4x4. Процесс последовательного укрупнения клеток матрицы приведен далее.

(16x16)

0	0	0	0	0	0	0	0	0	57344	61440	61440	61440	61440	64512	0
0	34944	65520	65520	65520	65528	65535	65535	65535	65535	65535	65535	65535	65535	65535	0
0	204	255	255	255	255	255	51455	65535	65535	511	255	255	255	255	0
0	0	0	0	0	0	49152	65534	65535	4919	0	0	0	0	0	0
0	0	0	0	0	61120	65535	65535	65535	65281	65280	65280	65528	65535	4369	0
0	0	65535	65535	65535	65535	65535	65535	65535	65535	65535	65535	65535	65535	4369	0
0	0	65535	65535	0	0	0	0	0	0	0	0	61166	65535	4369	0
0	32768	65535	65535	0	0	0	0	0	0	0	0	61166	65535	4369	0
0	34952	65535	65535	0	57344	61440	61440	64512	65280	65280	65280	65518	65535	4369	0
0	34952	65535	65535	65532	65535	65535	4607	255	255	255	255	61183	65535	4369	0
0	34952	65535	65535	0	0	0	0	0	0	0	0	61166	65535	4369	0
0	34952	65535	65535	0	0	0	0	0	0	0	0	61166	65535	13073	0
0	34952	65535	65535	0	0	0	0	0	0	0	0	61166	65535	13107	0
0	34952	65535	65535	64512	65280	65280	65535	65535	65535	65535	65535	65535	14207	16	0
0	52424	65535	65535	65535	65535	65535	65535	65535	65535	65535	65535	32767	273	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(4x4)

1.459E+7	3.141E+9	3.767E+8	8.239E+6
3.704E+9	1.664E+7	1.671E+7	9.163E+8
3.741E+9	8.782E+6	1.04E+6	1.059E+9
2.427E+8	2.673E+8	2.674E+8	1.028E+7

Вектор распознавания, состоящий из следов степеней последней матрицы, имеет вид (1.72411927354642, 1.84155825913441, 2.20477622051669, 2.76258275551884).

Заключение

Таким образом, сортировка может служить единой основой для автоматической идентификации нулей и экстремумов произвольной функции одной и более переменных в произвольно фиксированной части области определения. В частности, с использованием метода Леверье на этой основе вычисляются нули характеристического полинома матрицы. Для нулей полиномов в общем случае учитывается их кратность. Функция может быть задана значениями на равномерной сетке. В этом случае идентифицируются нули и экстремумы разностных решений обыкновенных дифференциальных уравнений и уравнений в частных производных. Для оцифрованного изображения на плоскости, рассматриваемого как разновидность дискретной функции двух переменных на сетке пиксельных элементов, на той же основе строится вектор распознавания. При этом входное изображение классифицируется как произвольное геометрическое место точек. Для подхода в целом существенно, что процесс обработки использует только операции сравнения. Это исключает накопление погрешности, влечет высокую точность локализации экстремумов и устойчивость идентификации изображений.

1. *Ромм Я.Е.* Метод вычисления нулей и экстремумов функций на основе сортировки с приложением к поиску и распознаванию. I // Кибернетика и системный анализ. – 2001. – № 4. – С. 142 – 159.
2. *Ромм Я.Е.* Метод вычисления нулей и экстремумов функций на основе сортировки с приложением к поиску и распознаванию. II // Там же. – 2001. – № 5. – С. 81 – 101.
3. *Ромм Я.Е., Заика И.В.* Программная локализация экстремумов функций и разностных приближений решений дифференциальных уравнений // Известия вузов. Северо-Кавказский регион. Технические науки. Спец. выпуск. «Математическое моделирование и компьютерные технологии». – 2005. – С. 47 – 52.
4. *Ромм Я.Е., Гуревич М.Ю., Белоконова С.С., Соловьева И.А.* Вычисление нулей и полюсов функций на основе устойчивой адресной сортировки с приложением к поиску и распознаванию // Проблемы программирования. – 2004. – № 2-3.– С. 462 – 472.
5. *Ромм Я.Е., Заика И.В., Соловьева И.А.* Метод программной оптимизации в приложении к математическим моделям экономики // В сб.: «Проблемы регионального управления, экономики, права и инновационных процессов в образовании». Таганрог: 2005, Т. 2. – С. 17–26.
6. *Ромм Я.Е., Соловьёва И.А.* Вычисление собственных значений матриц на основе сортировки с приложением к построению нормальной жордановой формы ТГПИ. – Таганрог, 2005. – 32 с. Деп. В ВИНИТИ 17. 05. 2005, № 710-В2005.
7. *Крылов В.И., Бобков В.В., Монастырный П.И.* Вычислительные методы. – М.: Наука, 1977. – Т. .2. – 400 с.
8. *Фаддеев Д.К., Фаддеева В.Н.* Вычислительные методы линейной алгебры. – СПб: Лань, 2002. – 736 с.
9. *Путятин Е.П., Аверин С.И.* Обработка изображений в робототехнике. – М: Машиностроение, 1990. – 320 с.
10. *Солодовников В.И.* Верхние оценки сложности решения систем линейных уравнений. – В кн.: Теория сложности вычислений. I: Записки научных семинаров ЛОМИ АН СССР. – Л., 1982. – 118. – С. 159 – 187.