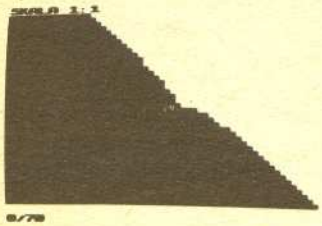
Powiększenia mikrokomputerowe

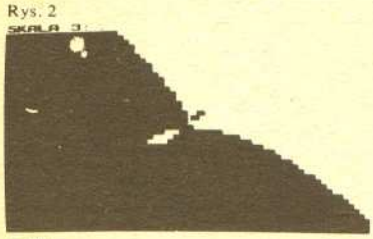
Dr hab. Jerzy JURKIEWICZ

```
 5 INPUT C
10 FOR X=-32 TO 31
20 FOR Y=-21 TO 20
30 IF 10*X*X*Y+(X-Y)*C+X*X*X*X+X*X+
     Y*Y*Y*Y*Y <= 0 THEN PLOT X+32,Y+21
40 NEXT Y
50 NEXT X
60 PRINT AT 0,0;"SKALA ";C;":1"
```

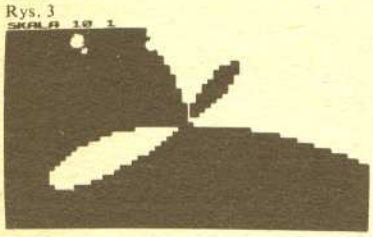
Rys. 1



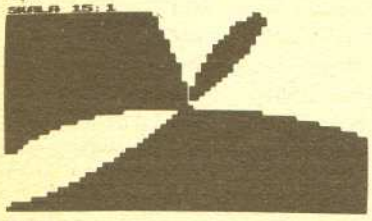
Rys. 2



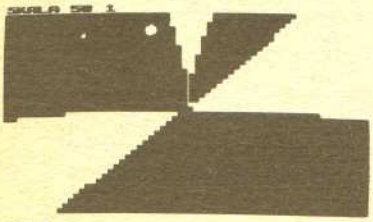
Rys. 3



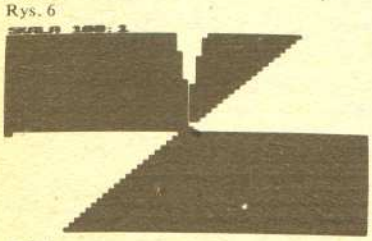
Rys. 4



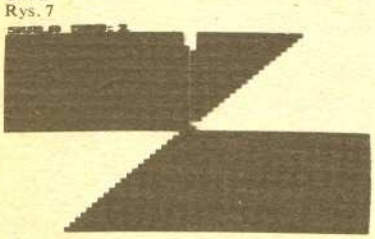
Rys. 5



Rys. 6



Rys. 7



Rys. 8

Znacie mikrokomputer ZX81 Sinclaira? Można go kupić za kilkanaście dolarów. Niektórzy twierdzą, że jego ,,ploty'' są zbyt grube, a więc obraz wytworzony przez to urządzenie na ekranie TV ma zbyt małą rozdzielczość dla subtelnych zastosowań graficznych. Otóż chcę Was przekonać, że to nieprawda. Jeżeli jakiś szczegół oglądanego na ekranie rysunku jest za mały i niewyraźny, wystarczy powiększyć skalę! Oto przykład.

Interesuje nas krzywa K o równaniu

$$(*) \qquad 10X^2Y(X-Y)+X^5+Y^5=0,$$

to znaczy zbiór punktów płaszczyzny, których współrzędne spełniają to równanie. Komputer będzie szukał punktów tej krzywej próbując różne pary liczbowe X, Y. Ze względu na błędy zaokrągleń jest jednak mało prawdopodobne, aby otrzymać dokładnie równość (*) nawet, gdyby znalezione X i Y stanowiły dokładne rozwiązanie. Praktyczniej będzie zastąpić równanie (*) nierównością

$$(**) \qquad 10X^2Y(X-Y)+X^5+Y^5\leqslant 0.$$

Jeżeli komputer zaczerni na ekranie ,,punkty'', których współrzędne spełniają tę nierówność, to krzywą K odnajdziemy jako *brzeg* zaczernionego obszaru.

(Dociekliwym Czytelnikom wyjaśniam, że wielomian występujący w równaniu (*) nie ma czynników wielokrotnych, więc ma on różne znaki, po obu stronych każdej gałęzi krzywej K.)

I jeszcze jedno. Ponieważ, jak się przekonamy, interesujący jest punkt (0,0) naszej krzywej, więc aby lepiej zobaczyć jego otoczenie, przesuniemy rysunek na ekranie o wektor (32,21) tak, aby punkt (0,0) przeniósł się na środek ekranu. Tłumaczy to instrukcję ,,... THEN PLOT $X+32$, $Y+21$'' w linii 30 programu (rys. 1).

Teraz zlecamy komputerowi sprawdzenie wszystkich punktów o współrzędnych *całkowitych* X, Y, gdzie $-32\leqslant X\leqslant 31$, $-21\leqslant Y\leqslant 20$. Jeżeli dla pewnego punktu (X, Y) zachodzi nierówność (**), to odpowiedni kwadracik ma być zaczerniony. Trochę to potrwa, bo do przebadania jest $64\times 42 = 2688$ punktów. Program przedstawiony jest na rysunku 1. Występuje tam poza X i Y zmienna C, która oznaczać będzie skalę powiększenia, o czym dalej. Na razie nadajemy jej wartość 1, to znaczy po uruchomieniu programu wczytujemy **1**.

Po NEWLINE (czyli ENTER) na ekranie stopniowo ukazuje się zaczerniony obszar rysunku 2. Jak widać, w okolicy środka ekranu krzywa K, czyli brzeg zaczernionego obszaru ma jakąś osobliwość, prawie niewidoczną w tej skali. Po RUN dla uzyskania np. trzykrotnego powiększenia wczytujemy **3**. Powstanie obraz widoczny na rysunku 3. Oczywiście zwiększenie skali odbyło się kosztem zmniejszenia oglądanego fragmentu krzywej.

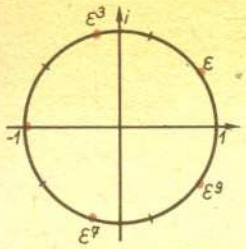
Powiększenie dziesięciokrotne jest ukazane na rysunku 4. Widać już wyraźnie, że krzywa ma dwa ,,liście'' wyrastające ze środka ekranu. Nadal niejasne jest jednak, co dzieje się bezpośrednio nad środkiem. Kolejne powiększenia widoczne na rysunku 5, rysunku 6 i rysunku 7 wyjaśniają sprawę. W górę wyrasta szpiczasta gałąź krzywej, tak zwany cusp (czyt. kasp). Przy skali 500:1 (rys. 8) ukazany fragment krzywej K sprowadza się, praktycznie biorąc, do trzech prostych. Dalsze powiększenia nic już nie wniosą. Rysunek 8 można interpretować następująco, posługując się równaniem (*). Dla X i Y bardzo bliskich zeru (w porównaniu z 1) składnik X^5+Y^5, który jest stopnia 5, jest bardzo mały w porównaniu ze składnikiem $10\cdot X^2Y(X-Y)$, mającym stopień 4. Zatem w pobliżu punktu (0,0) krzywa K opisana jest dość dokładnie równaniem

$$X^2Y(X-Y)=0,$$

a więc omalże pokrywa się z sumą prostych $X=0$, $Y=0$ i $X-Y=0$. Istotnie widzimy, że te właśnie proste stanowią brzeg czarnego obszaru w pobliżu punktu (0,0). Nie zapominajmy, że ten punkt umieściliśmy na środku ekranu! Opisane linie nazywamy prostymi stycznymi do gałęzi krzywej K w punkcie (0,0).

Wróćmy do rysunku 2, który przedstawia krzywą w skali 1:1 (oczywiście dla danego ekranu). Dla dalekich od zera (w porównaniu z 1) liczb X i Y składnik X^5+Y^5 jest duży w porównaniu ze składnikiem $10X^2Y(X-Y)$, więc równanie (*) jest dobrze przybliżane równaniem $X^5+Y^5=0$. W dziedzinie liczb rzeczywistych opisuje ono prostą $X+Y=0$, bo $X^5=-Y^5=(-Y)^5\Leftrightarrow X=-Y$.

Rys. 9



Spróbuj, Czytelniku, obejrzeć krzywą o równaniu

$$X^2 - Y^2 + Y^4 = 0.$$

Inne ciekawe rysunki można otrzymać rozpatrując równania postaci

$$(X + A_1 Y)^{n_1} \cdot (X + A_2 Y)^{n_2} \cdot \ldots \cdot (X + A_k Y)^{n_k} + w(X, Y) = 0,$$

gdzie $w(X, Y)$ jest wielomianem jednorodnym stopnia większego niż $n_1 + n_2 + \ldots + n_k$, tzn.

$$w(X,Y) = B_0 X^n + B_1 X^{n-1} Y + \ldots + B_{n-1} X Y^{n-1} + B_n X^n,$$

gdzie $n > n_1 + \ldots + n_k$.

Tę właśnie prostą widzimy (w przybliżeniu) na rysunku. Wyznacza ona tak zwany punkt w nieskończoności albo punkt niewłaściwy krzywej K. Gdyby ekran ukazywał rozwiązania nierzeczywiste (zespolone), zobaczylibyśmy jeszcze cztery pozostałe punkty w nieskończoności, odpowiadające „prostym" $X = \varepsilon^k \cdot Y$, gdzie $\varepsilon = \cos\frac{2\pi}{10} + i\sin\frac{2\pi}{10}$, a $k = 1, 3, 7$ lub 9 (rys. 9).

Widzimy więc, że oglądając krzywą K w różnej skali możemy zbadać ją z lokalnego i z globalnego punktu widzenia.

Wróćmy do parametru C. Niech C będzie liczbą rzeczywistą różną od zera i niech K' oznacza krzywą powstałą z K przez C-krotne powiększenie. Mamy

$$(X, Y) \in K' \Leftrightarrow (X/C, Y/C) \in K,$$

a zatem krzywa K' opisana jest równaniem

$$10(X/C)^2(Y/C)(X/C - Y/C) + (X/C)^5 + (Y/C)^5 = 0.$$

Mnożąc obie strony przez C^5 mamy

$$10X^2Y(X - Y) \cdot C + X^5 + Y^5 = 0.$$

Język BASIC ma co prawda operację potęgowania, ale jest ona poprawna tylko dla dodatnich argumentów. Komplikowałoby to program na tyle, że praktyczniej będzie w naszym przypadku napisać $X*X*X*X*X$ zamiast $X**5$ itd. Wyjaśnia to postać najważniejszej, 30 linii programu na rysunku 1.

Czego nie potrafi komputer

Mgr Jarosław DEMINET

Czy komputer potrafi wszystko?
Wiadomo, że obecnie wiele problemów przekracza możliwości współczesnych komputerów, np. gra w szachy (na poziomie mistrza świata) albo niezawodna prognoza pogody. Nie ma jednak podstaw do przypuszczeń, że tych akurat problemów nie da się rozwiązać kiedyś w przyszłości. Czy jednak istnieją problemy, których żaden komputer nigdy nie rozwiąże? Okazuje się, że tak. Niektóre z nich są zresztą bardzo proste.
Oto przykład.

Wiadomo, że każdy niebanalny program komputerowy zawiera pętle, tzn. ciągi instrukcji wykonywane wielokrotnie. Kontynuowanie obliczeń w pętli przebiega do momentu spełnienia określonego warunku, np. w algorytmie szybkiego sortowania (*Delta* 9/1985) zewnętrzna pętla wykonywała się, dopóki $i < j$, a zatem kończyła się, gdy $i \geqslant j$. Zarówno i, jak i j były zmieniane wewnątrz pętli, przy czym i było zwiększane, a j zmniejszane. Można dowieść, że po skończonej liczbie wykonań pętli rzeczywiście obie zmienne przyjmą takie wartości, że $i \geqslant j$. Co jednak byłoby, gdybyśmy pisząc algorytm zrobili jakiś błąd i gdyby dla pewnych danych wejściowych warunek $i \geqslant j$ nigdy nie był spełniony? Powiedzielibyśmy wówczas, że program się zapętlił — pewien jego fragment wykonywałby się nieskończenie długo i wymagałby przerwania z zewnątrz. Oczywiście sytuacja taka jest przykra i każdy chce jej uniknąć. Własność programu polegająca na tym, że program zawsze zakończy swoją pracę, nazywamy własnością stopu. Na ogół potrafimy udowodnić „ręcznie" własność stopu dla swojego programu. Może wobec tego można napisać program komputerowy, który potrafiłby automatycznie dowodzić własność stopu lub jej brak dla dowolnego programu? Okazuje się, że nie, a najprościej tego dowieść przez sprowadzenie do sprzeczności.

Przypuśćmy, że mamy procedurę WŁASNOŚĆ—STOPU (X), dla której parametrem X jest tekst dowolnej procedury, zapisanej w jakimś języku programowania. Procedura daje w wyniku wartość logiczną *prawda*, gdy procedura X ma własność stopu, zaś *fałsz*, gdy tej własności nie posiada. Oczywiście dysponujemy tekstem procedury WŁASNOŚĆ—STOPU. Teraz możemy napisać bezparametrową procedurę:

```
ZAGADKA:
    dopóki WŁASNOŚĆ—STOPU (ZAGADKA) wykonuj
        początek
            nic nie rób
        koniec
```

Tekst tej prostej procedury przekazujemy jako parametr procedurze WŁASNOŚĆ—STOPU. Jaki będzie wynik?

Załóżmy, że WŁASNOŚĆ—STOPU (ZAGADKA) jest prawdą. Ale to znaczy, że warunek pętli w procedurze ZAGADKA będzie zawsze spełniony, a zatem pętla będzie się wykonywać nieskończenie i ZAGADKA nie ma własności stopu.

Załóżmy, że WŁASNOŚĆ—STOPU (ZAGADKA) jest fałszem. To znaczy, że pętla w procedurze ZAGADKA nie wykona się ani razu i procedura od razu zakończy się, a zatem ma ona własność stopu.

W obu przypadkach dochodzimy do sprzeczności. Oznacza to, że procedura WŁASNOŚĆ—STOPU o szukanych własnościach nie istnieje. Sprzeczność zniknie, gdy założymy, że procedura WŁASNOŚĆ—STOPU sama nie ma własności stopu, tzn. że dla niektórych wartości swojego parametru (dla niektórych procedur) nie daje w ogóle odpowiedzi w skończonym czasie. Ale to już nie to...

A więc nigdy nie da się zaprogramować komputera tak, aby określał, czy dowolny program zakończy swoją pracę. Mówimy, że własność stopu *nie jest rozstrzygalna*. Okazuje się zresztą, że wiele innych własności programów nie jest rozstrzygalnych. Nie istnieje np. algorytm znajdujący zbędne instrukcje i zmienne (zbędne — tzn. takie, które zawsze będą ominięte przez program). Można bowiem dowieść, że gdyby taki algorytm istniał, to pozwoliłby on również na dowodzenie własności stopu.

A dla programistów wynika z tego morał: żaden komputer nie sprawdzi naszych programów całkowicie i do końca. Musimy błysnąć intuicją i sami znajdować dowody własności stopu naszych programów.