Alchemia

*Piotr KRZYŻANOWSKI**

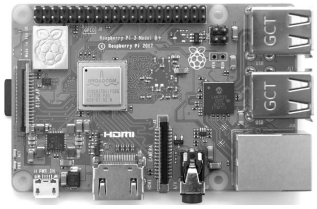
* Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

Witajcie, Młodzi Alchemicy! Na dzisiejszym spotkaniu nie będziemy się zajmować tak przebrzmiałym tematem, jak zamiana zwykłego metalu w złoto… (Kto jeszcze nie wie, jak to należy prawidłowo robić, może przeczytać przepis na marginesie niniejszych notatek.) Przed nami znacznie ciekawszy problem:

Przepis, mający mniej niż 40 lat: *Transmutacja bizmutu w (jakieś) złoto.* Po prostu weź folię z bizmutu i mocno uderzaj w nią bardzo rozpędzonymi jądrami węgla i neonu. *Źródło: Phys. Rev. C 23, 1044; 1 March 1981.* Ewentualnie, można mocno zderzyć ze sobą dwie gwiazdy neutronowe, wtedy też powstanie trochę złota: pisaliśmy o tym w *Delcie* 7/2018.

Jak z maliny wydobyć wolfram?

Okazuje się to całkiem proste, pod warunkiem, że wydobędziemy nie „zwykły” wolfram, tylko Wolfram Language. A i malina nie będzie zwykła: to popularny komputerek wielkości karty kredytowej, o apetycznej nazwie Raspberry Pi.



Raspberry Pi 3+ w skali 1:2
Źródło: Wikipedia

W rzeczywistości osiągniemy jeszcze więcej: mając ten komputerek, będziemy mogli uruchomić na nim pełną (no, *prawie* pełną) wersję najnowszej Mathematiki, potężnego systemu obliczeń symbolicznych. Na pozór nie wydaje się to szczególnym osiągnięciem, gdyż każdy może to samo zrobić na dowolnym laptopie – pod warunkiem wszakże, że wcześniej wyasygnuje pokaźną sumkę na zakup licencji (uczciwie dodajmy, że jeśli jest uczniem lub studentem, wystarczy skromne £30 za półroczne użytkowanie). Jednak na Raspberry Pi możemy mieć ten software natychmiast, legalnie (do użytku niekomercyjnego) i… całkowicie za darmo!

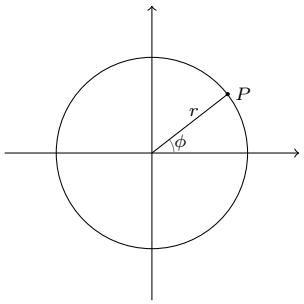
Oczywiście, jest pewien haczyk: możliwości nawet najmocniejszego Raspberry Pi są daleko poniżej laptopa ze średniej półki, więc zestaw zadań, jakie można *wygodnie* rozwiązywać w Mathematice na Raspberry Pi jest tym samym mocno ograniczony. Z drugiej strony, ten mały komputerek pozwala na prosty i tani kontakt ze światem realnym (przez możliwość podłączenia rozmaitych czujników: światła, temperatury itp., a także mikrofonu i kamery). Co więcej, jego Mathematica – dokładniej Wolfram Language, w którym ją programujemy – ma rozszerzenia pozwalające w prosty sposób to wykorzystać. No i – *last but not least* – zabawa z Malinką jest przednia (moim zdaniem, dla początkujących lepsza niż z Arduino), działa też na nim *programowalny w paru językach, w tym: Pythonie i Mathematice…* Minecraft! (W wersji *pocket edition*.)

Można też użyć Pythona i bogatego zestawu modułów obsługujących czujniki.

Zaczynamy od instalacji systemu operacyjnego Raspbian (specjalna wersja Linuxa) na karcie microSD – która zastępuje dysk w Raspberry Pi – a potem całość uruchamiamy i aktualizujemy. Oprogramowanie Wolframa już tam będzie.

Więcej o tym, jak zacząć przygodę z Raspberry Pi, można przeczytać np. na stronie `www.raspberrypi.org/documentation`. Jest też kilka polskich serwisów poświęconych Malince.

Mając w jednej garści i Malinkę, i Wolframa, aż korci, aby dokładając stosowne czujniki, szybko *transmutować* tę parę – np. w parapetową stację pogodową. Dziś jednak skupimy się tylko na zabawie z samym Wolframem (czyli Mathematiką) i dokonamy „alchemicznego” przekształcenia garści liczb w obiekty o naturalnym kształcie, takim jak kwiaty, nasiona, rozgwiazdy.



P ma współrzędne biegunowe (r, ϕ)

Wybierzemy sześć liczb: $a, b > 0$, $m, n_1, n_2, n_3 \geqslant 0$, a następnie zdefiniujemy na płaszczyźnie krzywą we współrzędnych biegunowych (r, ϕ), zadając promień wodzący wzorem

$$r = S(\phi),$$

gdzie

$$S(\phi) = \left(\left| a^{-1} \cos\left(\frac{\phi\, m}{4}\right) \right|^{n_2} + \left| b^{-1} \sin\left(\frac{\phi\, m}{4}\right) \right|^{n_3} \right)^{-1/n_1}.$$

Powyższy wzór podał w 1999 roku belgijski botanik Johan Gielis i – zachwycony tym, jak różnorodne kształty można z niego uzyskać – ochrzcił bez zbędnej skromności *superformułą*. A następnie, w co trudno uwierzyć, swój pomysł… opatentował, zob. `patents.google.com/patent/EP1177529B1`. My jednak zauważmy (Gielis oczywiście też to wiedział), że jest to uogólnienie wzoru na elipsę poprzez umożliwienie wyboru potęg: faktycznie, dla $m = 4$ i $n_1 = n_2 = n_3 = 2$ dostajemy zwyczajne równanie elipsy. Gielis poszedł jeszcze dalej i w swój wzór radośnie wplótł możliwość dalszego skalowania promienia:

$$r = f(\phi) \cdot S(\phi),$$

gdzie f jest, hmm…, *dowolną* funkcją…

Idea Gielisa była inspirująca. Na przykład gra komputerowa *No Man's Sky* generuje swoje sztuczne niezliczone światy w sposób proceduralny, opierając się na pomyśle podobnym do superformuły Gielisa – co parę lat temu zdradził Sean Murray, jeden z twórców gry.

Jednak nawet gdy $f \equiv 1$, efekty wyboru 6 parametrów potrafią być intrygujące, co nie powinno nas dziwić: po prostu mamy całkiem dużo stopni swobody.

Czy potrafisz znaleźć parametry funkcji S, które generują kształty podobne do tych na ikonie Mathematiki powyżej?

Jak trzeźwo zauważył Eric Weisstein, redaktor strony `mathworld.wolfram.com/Superellipse.html`, już John von Neumann miał sarkastycznie (choć może z pewną przesadą) powiedzieć:

> *Przy czterech parametrach* [krzywej] *dopasuję ją do* [kształtu] *słonia; przy pięciu, będzie machał trąbą.*

No to do roboty! Wykorzystajmy Mathematikę do rysowania krzywych zadanych superformułą. Dzięki możliwości wygodnego definiowania wykresów zależnych od parametrów, a także umiejętności rysowania krzywych zadanych wprost we współrzędnych biegunowych, całość zamkniemy w dwu liniach kodu, na który głównie składa się seria zagnieżdżonych wywołań funkcji:

```
1 S[t_, a_, b_, m_, n1_, n2_, n3_] := ( Abs[ Cos[(m*t)/4]/a ]^n2 + Abs[ Sin[(m*t)/4]/b
      ]^n3 )^(-1/n1);
2
3 Manipulate[ PolarPlot[ S[t, a, b, m, n1, n2, n3], {t, 0, 2*Pi}, PlotLabel -> {m, n1, n2,
      n3}, Axes -> False ], {{a, 1}, 0, 5}, {{b, 1}, 0, 5}, {{m, 4}, 0, 20, 1}, {{n1, 2},
      0, 20}, {{n2, 2}, 0, 20}, {{n3, 2}, 0, 20}]
```

Wprowadzenie do Mathematiki na Raspberry Pi: `projects.raspberrypi.org/en/projects/getting-started-with-mathematica/`

Należy pamiętać o dziwnym zwyczaju zatwierdzania każdej instrukcji nie przez zwykły *Enter*, tylko przez kombinację klawiszy *Shift+Enter*.

Ten krótki kod daje nam przedsmak charakteru języka Mathematiki, czyli Wolfram Language. Kilka rzeczy, które od razu rzucają się w oczy, to:

- argumenty funkcji otacza się nawiasami kwadratowymi,
- nazwy funkcji wbudowanych (np. sinus) pisze się wielką literą.

W pierwszej linii programu definiujemy funkcję S (dla uproszczenia, zmienną ϕ w kodzie zamieniamy na t; ponieważ parametry S też traktujemy jako następne jej argumenty, w kodzie powyżej definiujemy w efekcie funkcję siedmiu argumentów). Definiując w Mathematice własną funkcję, korzystamy z operatora „:=", a argumenty *po lewej stronie definicji* funkcji musimy – co użytkownikom innych języków może wydawać się dziwaczne – wyróżniać przyrostkiem „_". Dlatego funkcję, która wyznacza oba pierwiastki zadanego trójmianu kwadratowego $ax^2 + bx + c$, zapisalibyśmy Pierwiastki[a_,b_,c_] := **Roots**[a*x^2+b*x+c==0, x] (wbudowana funkcja **Roots** rozwiązuje równania wielomianowe), a następnie użyli na przykład tak: Pierwiastki[2,0,−m]. W linii numer 3 najważniejsza jest funkcja PolarPlot, rysująca wykres we współrzędnych biegunowych $[a, b] \ni t \mapsto (r(t), t)$, gdzie r jest długością promienia wodzącego, a t jest kątem, jaki tworzy z osią OX. Najprostsze jej wywołanie w tym kontekście miałoby postać PolarPlot[r[t], {t, a, b}]. W naszym kodzie dodatkowo zmieniamy pewne opcje tej funkcji, korzystając z operatora nadawania wartości „−>":

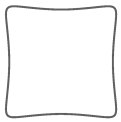
```
PolarPlot[ S[t, a, b, m, n1, n2, n3], {t, 0, 2*Pi}, PlotLabel -> {m, n1, n2, n3}, Axes
    -> False ]
```

powodując, że w tytule wykresu są wypisywane wartości parametrów m, n_1, n_2, n_3, a osie współrzędnych nie będą rysowane.

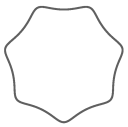
Aby mieć możliwość interaktywnego zmieniania wartości argumentów funkcji S innych niż t, zanurzamy wywołanie PolarPlot w funkcji Manipulate, która automatycznie stworzy okno z suwakami odpowiadającymi wartościom podanym jako dalsze jej argumenty. Przykładowo, podając (zob. w trzecim wierszu trzeciej linii kodu) {{a, 1}, 0, 5}, instruujemy Manipulate, że parametr a może przyjmować wartości z przedziału $[0, 5]$, z domyślnie ustawioną wartością równą 1. Nieco bardziej złożony argument {{m, 4}, 0, 20, 1} oznacza, że m będzie domyślnie równe 4, będzie miało wartości z zakresu od 0 do 20, ale zmiana m będzie odbywać się ze skokiem co 1 – dzięki czemu m będzie przyjmować tylko *całkowite* wartości.

Teraz możemy zacząć zabawę z suwakami i śledzić na bieżąco, jak zmiany parametrów wpływają na zmiany kształtu krzywej. Na marginesie powyżej pokazano te efekty działania superformuły, które są zbliżone do naturalnych kształtów i które tak zachwyciły Gielisa. Parametry w postaci $(m, n_1, n_2 = n_3)$, z „biologicznymi" interpretacjami, pochodzą z artykułu Gielisa w *American Journal of Botany* 90(3) z 2003 roku. Czy obrazki są ładne? Niewątpliwie. Ale czy jest w nich coś głębszego? Na to pytanie nie podam odpowiedzi: przecież alchemia to wiedza tajemna.

Nuphar luteum petiole $(3, 4.5, 10)$



Scrophularia nodosa stem $(4, 12, 15)$



Equisetum stem $(7, 10, 6)$

Starfish 1 $(5, 2, 7)$

Starfish 2 $(5, 2, 13)$

Modified Rose curve $(10, 1/1.3, 2.7)$, $r(\phi) = |\cos(2.5\phi)|$

Jedna z nieskończenie wielu form, których nie było w artykule Gielisa: $(40, 8, 8)$, $r(\phi) = |\cos(3.5\phi)|$