I. MINIMALIZACJA FUNKCJI

1. Wykorzystując metodę złotego podziału wykonaj kilka pierwszych kroków lokalizacji minimum funkcji $e^x - x$ w przedziale [-1,1]

Rozwiązanie.

Stosunek złotego podziału to $\varphi = \frac{\sqrt{5}-1}{2}$. Mając dane dwa punkty a i b wyznaczamy dwa punkty wewnątrz przedziału $[a,b]$:

$$x_1 = b - \varphi(b-a) \quad \text{i} \quad x_2 = a + \varphi(b-a)$$

W naszym przypadku otrzymujemy $x_1 = -0.23607$ i $x_2 = 0.23607$. W kroku 1 dostajemy tabelę wartości funkcji w 4 punktach:

x_i	-1	-0,236068	0,236068	1
$f(x_i)$	1,367879	1,025795	1,030192	1,718282

W kolejnych krokach odrzucamy ten ze skrajnych punktów, w którym wartość funkcji jest największa i dzielimy w złotym stosunku dłuższy z pozostałych dwu odcinków. W ten sposób znowu otrzymujemy 4 punkty i procedurę powtarzamy. Zauważmy, że w każdym kroku potrzebujemy obliczyć wartość funkcji tylko w jednym nowym punkcie.
Stosując ten schemat do naszych danych otrzymujemy (dane dla nowego punktu pogrubiono):

krok2

x_i	-1	**-0,52786**	-0,236068	0,236068
$f(x_i)$	1,367879	**1,117728**	1,025795	1,030192

krok 3

x_i	-0,52786	-0,236068	**0,055728**	0,236068
$f(x_i)$	1,117728	1,025795	**1,001582**	1,030192

krok 4

x_i	-0,236068	**-0,05573**	0,055728	0,236068
$f(x_i)$	1,025795	**1,001524**	1,001582	1,030192

krok 5

x_i	-0,236068	**-0,12461**	-0,05573	0,055728
$f(x_i)$	1,025795	**1,007451**	1,001524	1,001582

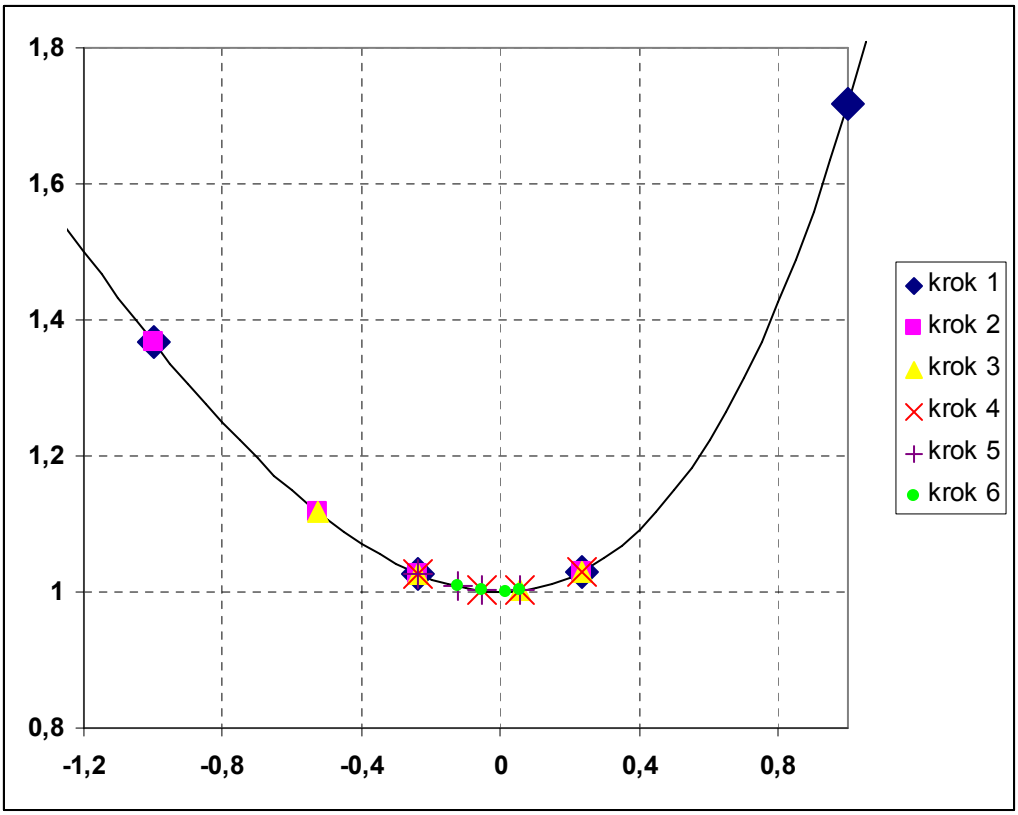
krok 6

x_i	-0,12461	-0,05573	**0,013156**	0,055728
$f(x_i)$	1,007451	1,001524	**1,000087**	1,001582

krok 7

x_i	-0,05573	**-0,013156**	0,013156	0,055728
$f(x_i)$	1,001524	**1,000086**	1,000087	1,001582

Widać, że coraz bardziej zbliżamy się do minimum funkcji (równego 1) w $x = 0$. Położenie punktów z kilku pierwszych kroków możemy obejrzeć na wykresie:



2. a) Znajdź punkty krytyczne funkcji

$$f(x,y) = x^4 + y^4 - 8x^2 - 2y^2 + 5$$

i określ ich rodzaj.

b) Pokaż, że funkcja Rosenbrocka

$$f(x,y) = 100(y - x^2)^2 + (1 - x)^2$$

ma minimum globalne w punkcie (1,1).

Rozwiązanie.

a) Pierwsze pochodne funkcji są równe

$$\frac{\partial f}{\partial x} = 4x^3 - 16x \qquad \frac{\partial f}{\partial y} = 4y^3 - 4y$$

Jak widać, ich miejsc zerowych można poszukiwać niezależnie. Pochodna po x zeruje się dla x = -2, 0 lub 2, pochodna po y dla y = -1, 0 lub 1.

Punkty krytyczne naszej funkcji to (0,0), (±2,0), (0, ±1) i (±2, ±1). Ze względu na to, że funkcja jest parzysta w obu argumentach, wystarczy zbadać tylko współrzędne nieujemne, dla ujemnych wynik jest taki sam.

Drugie pochodne funkcji są równe

$$\frac{\partial^2 f}{\partial x^2} = 12x^2 - 16 \qquad \frac{\partial^2 f}{\partial x \partial y} = 0 \qquad \frac{\partial^2 f}{\partial y^2} = 12y^2 - 4$$

Macierz drugich pochodnych naszej funkcji wygląda więc następująco

$$H(x,y) = \begin{bmatrix} 12x^2 - 16 & 0 \\ 0 & 12y^2 - 4 \end{bmatrix}$$

W punkcie (0,0) jest równa

$H(0,0) = \begin{bmatrix} -16 & 0 \\ 0 & -4 \end{bmatrix}$, hesjan jest ujemnie określony – czyli w (0,0) funkcja ma maksimum

$H(0,1) = \begin{bmatrix} -16 & 0 \\ 0 & 8 \end{bmatrix}$, hesjan jest nieokreślony – w (0,1) funkcja ma punkt siodłowy

$H(2,0) = \begin{bmatrix} 32 & 0 \\ 0 & -4 \end{bmatrix}$, hesjan jest nieokreślony – w (2,0) funkcja ma punkt siodłowy

$H(2,1) = \begin{bmatrix} 32 & 0 \\ 0 & 8 \end{bmatrix}$, hesjan jest dodatnio określony – w (2,1) funkcja ma minimum równe $f(2,1) = -12$

b) Podstawiając sprawdzamy, że $f(1,1) = 0$. Ponieważ funkcja jest sumą kwadratów, więc przyjmuje jedynie wartości nieujemne, zatem rzeczywiście w punkcie (1,1) funkcja ma minimum globalne. Minimum to położone jest w wygiętej dolinie (stąd funkcja zwana jest czasem funkcją bananową) a sama funkcja jest wykorzystywana w testowaniu programów znajdujących minima.

3. Funkcja `grad.desc` z biblioteki `animation` w pakiecie R demonstruje minimalizację metodą najszybszego spadku. Wypróbuj jej działanie na funkcjach z zadania 2.

Rozwiązanie.

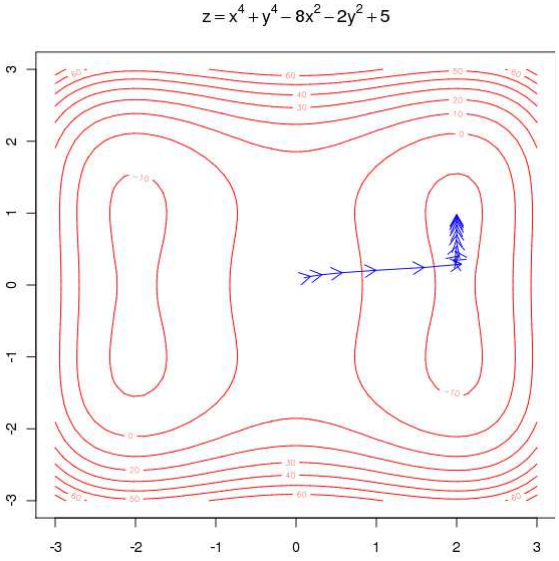
Ładujemy potrzebną bibliotekę i definiujemy funkcję

```
library(animation)
f = function(x,y) x^4 + y^4 -8*x^2 -2*y^2 + 5
```

Funkcji `grad.desc` podajemy nazwę funkcji dwu zmiennych *x* i *y*, wektor zadający naroża rysowanego obszaru i wektor określający punkt startowy. Warto też zwiększyć parametr `nmax` – liczbę dopuszczalnych kroków.

```
ani.options(nmax=50)
(grad.desc(f,c(-3,-3,3,3),c(0.1,0.1)))
```

W powyższym przykładzie wystartowaliśmy z punktu (0.1,0.1). Jak widać, procedura wykonuje w kierunku największego spadku krok o długości zależnej od wielkości gradientu:



Po zakończeniu pracy procedura wyświetla wyniki w rodzaju

```
$par
        x         y
1.9999183 0.9803762

$value
        x
-11.99849

$iter
[1] 19
```

w tym konkretnym przypadku mówiące, że po 19 krokach dotarła do punktu o współrzędnych (1.9999183 0.9803762), w którym funkcja ma wartość -11.99849.

Możemy sprawdzić, że po starcie z punktu (0.1,0)

```
ani.options(nmax=50)
(grad.desc(f,c(-3,-3,3,3),c(0.1,0)))
```

procedura ugrzęźnie w punkcie siodłowym (2,0)

```
$par
       x        y
2.008029 0.000000

$value
        x
-10.99896

$iter
[1] 10
```

z którego już się nie wydostanie ze względu na znikający gradient (z tego też powodu start z maksimum w (0,0) nic nam nie da).

Podobnie możemy spróbować znaleźć minimum funkcji Rosenbrocka

```
g = function(x,y) 100*(y-x^2)^2 + (1-x)^2
ani.options(nmax=50)
(grad.desc(g,c(-2,-2,2,2),c(0.999,0.999)))
```

stwierdzając, że procedura `grad.desc` zawodzi nawet dla punktu startowego tuż obok minimum, co wynika głównie z faktu, iż została ona stworzona do celów demonstracyjnych i nie nadaje się do prawdziwej pracy, ale też z wredności funkcji Rosenbrocka.

4. Znajdź minima funkcji z zadań 2 i 3 używając funkcji `optim` w pakiecie R.

Rozwiązanie.

Na użytek procedury `optim` funkcję musimy przedstawić jako zależną od składowych wektora **x**. Mamy zatem $x_1 = x$ i $x_2 = y$. Pierwszą funkcję definiujemy następująco:

```
f = function(x) x[1]^4 + x[2]^4 -8*x[1]^2 -2*x[2]^2 +5
```

i możemy startując np. z (0.001,0.001) poszukać minimum metodą sprzężonych gradientów:

```
optim(c(0.001,0.001),f,method="CG")
```

otrzymując w wyniku output:

```
$par
[1] 1.9999998 0.9999999

$value
[1] -12

$counts
function gradient
     153       35

$convergence
[1] 0
```

mówiący, że osiągnięto (`convergence =0`) punkt krytyczny (2,1) z wartością funkcji -12. Podobny wynik osiągniemy stosując metodę Broydena-Fletchera-Goldfarba-Shanno (aproksymującą hesjan przez obliczenia gradientu):

```
optim(c(0.001,0.001),f,method="BFGS")
```

metodę sympleksów

```
optim(c(0.001,0.001),f,method="Nelder-Mead")
```

lub symulowane wyżarzanie

```
optim(c(0.001,0.001),f,method="SANN")
```

choć ta ostatnia używa bardzo wielu obliczeń funkcji dając przy tym gorszy wynik.

Możemy sprawdzić, że dla startu z punktu (0.1,0) metody gradientowe skończą w punkcie siodłowym (2,0), np.

```
optim(c(0.1,0.),f,method="CG")
$par
[1] 2 0

$value
[1] -11
```

podczas gdy metoda sympleksów znajdzie minimum startując zarówno z (0.1,0) jak i (0,0)

```
optim(c(0.1,0.),f,method="Nelder-Mead")
optim(c(0.,0.),f,method="Nelder-Mead")
```

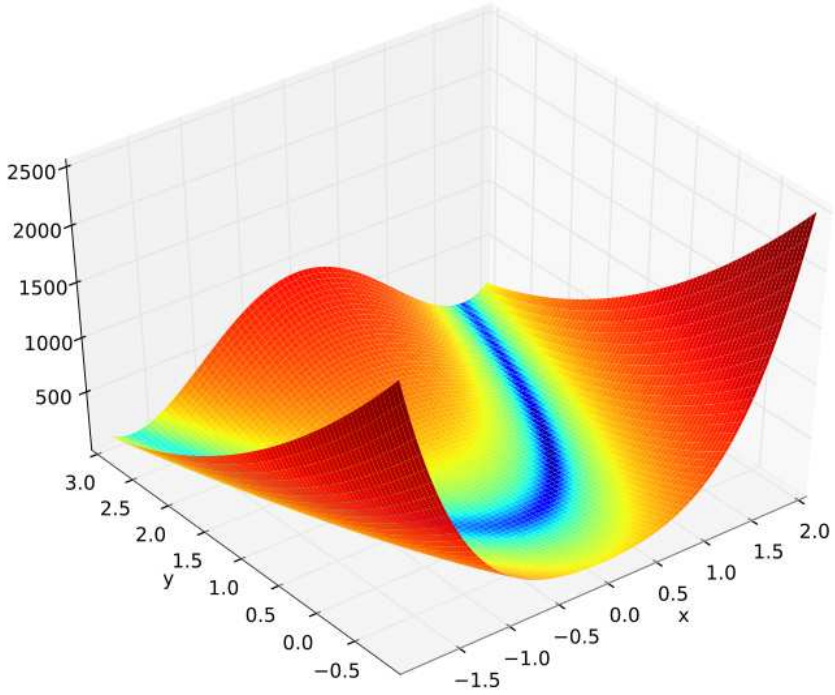
np. w tym drugim przypadku dostajemy

```
$par
[1] 2.0000775 0.9999574

$value
[1] -12
```

Tutaj metody gradientowe nic by nie zdziałały.

Spróbujmy wobec tego zmierzyć się z funkcją bananową Rosenbrocka. Jej kształt w okolicy minimum w (1,1) możemy obejrzeć na wykresie z Wikipedii:



Przy testowaniu programów za pomocą tej funkcji proponuje się startować z „kłopotliwego” punktu (-1.2,1)
Tak zrobimy i my – zdefiniujemy funkcję:

```
g = function(x) 100*(x[2]-x[1]^2)^2 + (1-x[1])^2
```

i możemy spróbować różnych metod:

Metoda sympleksów radzi sobie bez kłopotów:

```
optim(c(-1.2,1),g,method="Nelder-Mead")
$par
[1] 1.000260 1.000506

$value
[1] 8.825241e-08

$counts
function gradient
     195       NA

$convergence
[1] 0
```

podobnie jak metoda BFGS:

```
optim(c(-1.2,1),g,method="BFGS")
$par
[1] 0.9998044 0.9996084

$value
[1] 3.827383e-08
```

a nawet od biedy symulowane wyżarzanie:

```
optim(c(-1.2,1),g,method="SANN")
$par
[1] 1.002416 1.000767

$value
[1] 0.001662804
```

Natomiast metoda sprzężonych gradientów ma kłopoty:

```
optim(c(-1.2,1),g,method="CG")
$par
[1] -0.7648079  0.5927148

$value
[1] 3.106475

$counts
function gradient
     402      101

$convergence
[1] 1
```

Wartość parametru `convergence =1` wskazuje na przekroczenie limitu iteracji. Możemy temu zaradzić zwiększając ten limit:

```
optim(c(-1.2,1),g,method="CG",control=list(maxit=5000))
$par
[1] 0.9997152 0.9994295

$value
[1] 8.119731e-08

$counts
function gradient
   19938     4975

$convergence
[1] 0
```

Częściowo pomaga także zmiana sposobu aktualizacji hesjanu w trakcie obliczeń (ale to już są szczegóły techniczne):

```
optim(c(-1.2,1),g,method="CG",control=list(type=2,maxit=1000))
$par
[1] 0.9996933 0.9993856

$value
[1] 9.417974e-08

$counts
function gradient
    1038      249

$convergence
[1] 0
```

II. RÓWNANIA RÓŻNICZKOWE ZWYCZAJNE

Do znalezienia numerycznych rozwiązań następnych zadań będziemy wykorzystywali pakiet R. Ponieważ funkcje całkujące układy równań w R (jak i w wielu innych, np. Octave) pracują na układach postaci

$$x_i'(t) = f(t, x_1, \ldots, x_n)$$

będziemy starali się odpowiednio przekształcać nasze równania.

5. Równanie różniczkowe opisujące kształt swobodnie zwisającej liny o końcach zaczepionych na równej wysokości ma postać

$$\frac{d^2y}{dx^2} = \alpha\left[1+\left(\frac{dy}{dx}\right)^2\right]^{1/2},$$

gdzie α jest pewną stałą. Załóżmy dla uproszczenia $\alpha = 1$; przyjmijmy warunki początkowe $y(0) = 1$, $y'(0) = 0$.
a) sprowadź powyższe równanie do układu równań rzędu pierwszego
b) narysuj krzywą będącą rozwiązaniem równania w przedziale [-1,1]

Rozwiązanie

a) Wprowadźmy nowe zmienne

$$t = x$$
$$x_1 = y$$
$$x_2 = \frac{dy}{dx}$$

wtedy oczywiście

$$x_2' = \frac{d^2y}{dx^2}$$

i otrzymujemy równoważny naszemu równaniu układ równań pierwszego rzędu

$$x_1' = x_2$$
$$x_2' = \left(1+x_2^2\right)^{1/2}$$

z warunkami początkowymi $x_1(0) = 1$, $x_2(0) = 0$.

b) Funkcje potrzebne nam do numerycznego rozwiązania równania znajdują się w bibliotece `deSolve` pakietu R. Przed pierwszym użyciem bibliotekę tę należy zainstalować poleceniem
`install.packages()`
Po instalacji polecenie
`library(deSolve)`
udostępnia do wykorzystania procedury z biblioteki.

Przystępując do rozwiązania układu równań różniczkowych w R musimy po pierwsze zdefiniować funkcję obliczającą prawe strony układu dla zadanego t, x_i i zestawu parametrów. Wobec tego wydajemy polecenia

```
lancuch <- function(t,x,parms){
dx1 <- x[2]
dx2 <- sqrt(1+x[2]^2)
list(c(dx1,dx2))}
```

Co prawda nasza funkcja nie zależy od żadnego parametru, ale procedurze rozwiązującej musimy podać jaką wartość, np.

```
parms=0
```

zadajemy warunki początkowe

```
x0=c(1,0)
```

Chcemy znaleźć rozwiązanie układu dla $t \in [-1,1]$. Procedura całkująca układ równań wymaga podania jako pierwszej wartości zmiennej t czasu odpowiadającego warunkom początkowym (u nas $t = 0$). Ponieważ jest to środek, a nie początek naszego przedziału, rozwiązanie podzielimy na dwa: od 0 do 1 i od 0 do -1.
Generujemy zatem dwa zestawy wartości czasu:

```
time1=seq(0,1,0.05)
time2=seq(0,-1,-0.05)
```

a potem wywołujemy procedurę rk4, rozwiązującą układ metodą Rungego-Kutty 4-go rzędu.

```
out1=as.data.frame(rk4(x0,time1,lancuch,parms))
out2=as.data.frame(rk4(x0,time2,lancuch,parms))
```

Poleceniem

```
out1
```

możemy obejrzeć tabelkę rezultatów pierwszego całkowania, w kolumnach oznaczonych `time`, `1`, `2` znajdują się wartości t, x_1, x_2.

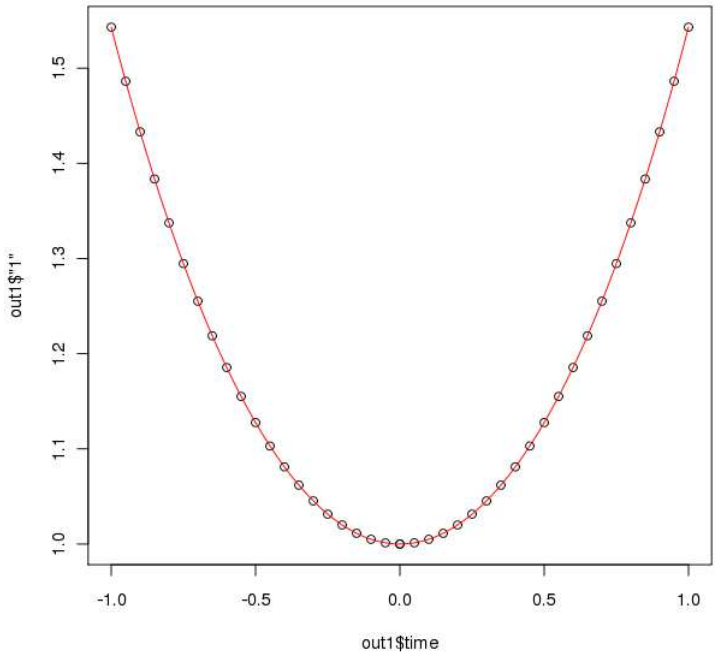
Rysujemy zatem wykres naszych rozwiązań

```
plot(out1$time,out1$"1",xlim=c(-1,1))
points(out2$time,out2$"1")
```

Możemy też nanieść znane rozwiązanie dokładne $y = \cosh x$ (czyli w naszych oznaczeniach $x_1 = \cosh t$)

```
curve(cosh(x),col="red",add=T)
```

Otrzymujemy ostatecznie wykres tzw. krzywej łańcuchowej:



6. Rozważ układ równań

$$x_1' = x_1 - x_1 x_2$$

$$x_2' = -x_2 + x_1 x_2$$

Jest to układ równań typu Lotki-Volterry.

a) pokaż, że przy pewnych założeniach układ ten opisuje schemat kinetyczny reakcji

$$\mathrm{A} + \mathrm{X} \xrightarrow{k_1} 2\mathrm{X}$$

$$\mathrm{X} + \mathrm{Y} \xrightarrow{k_2} 2\mathrm{Y}$$

$$\mathrm{Y} \xrightarrow{k_3} \mathrm{P}$$

b) dla warunków początkowych $x_1(0) = 2$, $x_2(0) = 0.5$ narysuj jego rozwiązania w przedziale [0,50]. Zbadaj metody Eulera i Rungego-Kutty 4-go rzędu.

Rozwiązanie

a) Zmiana stężenia substancji X wynosi

$$\frac{d[\mathrm{X}]}{dt} = k_1[\mathrm{A}][\mathrm{X}] - k_2[\mathrm{X}][\mathrm{Y}]$$

analogicznie

$$\frac{d[\mathrm{Y}]}{dt} = k_2[\mathrm{X}][\mathrm{Y}] - k_3[\mathrm{Y}]$$

Zauważmy jeszcze, że dodając równania reakcji dostajemy

$$\mathrm{A} \rightarrow \mathrm{P}$$

Zatem nasze równania opisują przekształcenie substratu A w produkt P w trójetapowym procesie z dwoma pośrednimi reagentami X i Y.

Jeśli założymy, że stężenie A jest stałe i wynosi 1 (np. substrat jest cały czas wprowadzany do reaktora) oraz $k_1 = k_2 = k_3 = 1$, to oznaczając $[\mathrm{X}] = x_1$ i $[\mathrm{Y}] = x_2$ dostajemy nasz układ równań. (Przy innym stężeniu A i innych stałych dostalibyśmy analogiczny układ, tylko ze zmienionymi współczynnikami).

b) Zaczynamy od zdefiniowania funkcji obliczającej prawe strony układu

```
lotvol <- function(t,x,parms){
dx1 <- x[1] - x[1]*x[2]
dx2 <- -x[2] + x[1]*x[2]
list(c(dx1,dx2))}
```

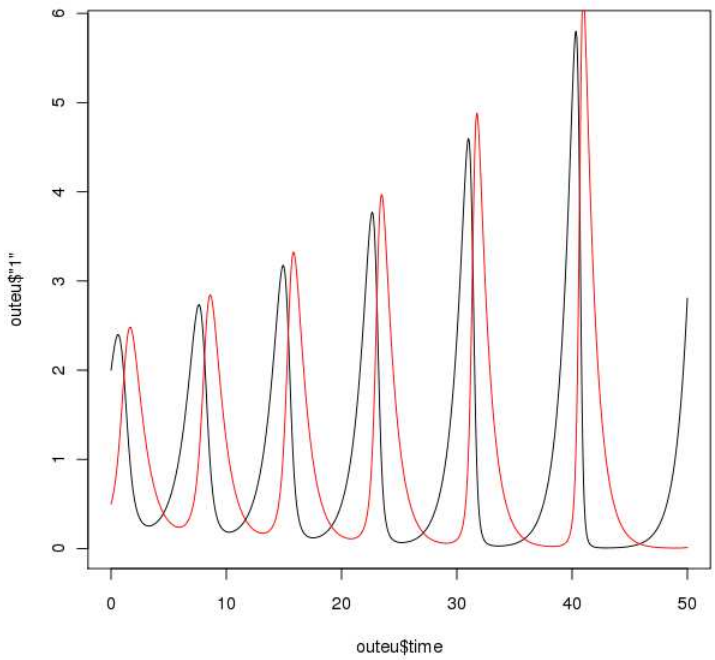
Określamy parametry (nie mamy żadnego), warunki początkowe i zakres *t*

```
parms=0
x0 = c(2,0.5)
time = seq(0,50,0.05)
```

Następnie rozwiązujemy układ korzystając z metody Eulera i tworzymy wykres

```
outeu=as.data.frame(euler(x0,time,lotvol,parms))
plot(outeu$time,outeu$"1",type="l")
lines(outeu$time,outeu$"2",col="red")
```

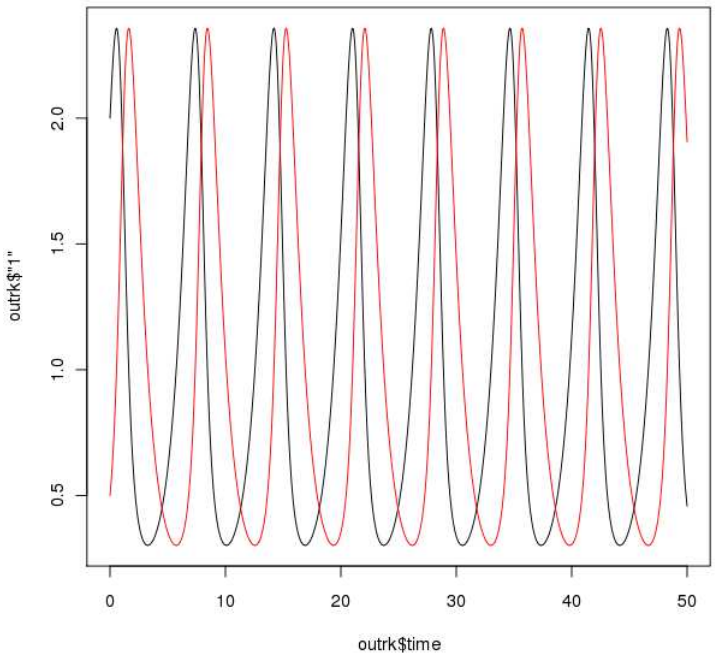
Otrzymujemy następujący wykres



Możemy sprawdzić dla mniejszego kroku czasowego, że maksima na krzywych narastają wolniej. Oznacza to problemy z dokładnością rozwiązania (zależy od kroku).
Sprawdźmy efekt działania metody Rungego-Kutty

```
outrk=as.data.frame(rk4(x0,time,lotvol,parms))
plot(outrk$time,outrk$"1",type="l")
lines(outrk$time,outrk$"2",col="red")
```

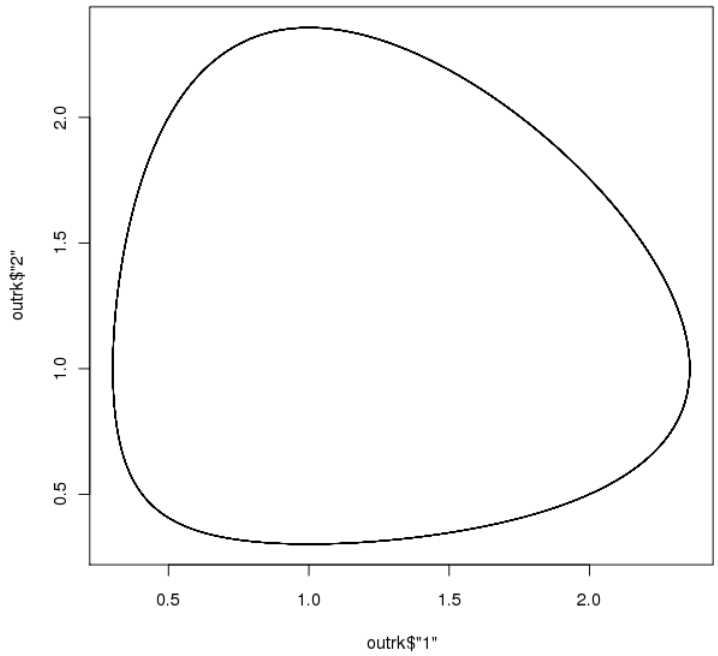
otrzymując tym razem wykres



Jak widać, stężenia substancji X i Y zmieniają się periodycznie w czasie – jest to przykład reakcji oscylacyjnej. Równania Lotki-Volterry stosowane są do opisu zmian liczebności populacji w układzie drapieżnik-ofiara.

Obejrzyjmy jeszcze zależność x_2 od x_1

```
plot(outrk$"1",outrk$"2",type="l")
```



Krzywa jest zamknięta, co potwierdza periodyczność zmian w układzie.

Układy co najmniej 3 nieliniowych równań różniczkowych pierwszego rzędu mogą wykazywać zachowania chaotyczne, tzn. mała zmiana warunków początkowych może skutkować dramatyczną zmianą rozwiązania. Pokazują to następne przykłady.

7. Rozważmy układ następujących równań różniczkowych

$$x_1' = \alpha(x_2 - x_1)$$
$$x_2' = \beta x_1 - x_2 - x_1 x_3$$
$$x_3' = x_1 x_2 - \gamma x_3$$

z parametrami $\alpha = 10$, $\gamma = 8/3$

a) znajdź rozwiązanie układu w [0,20] dla $\beta = 15$ i następujących warunków początkowych

$x_1(0) = 15$, $x_2(0) = 10$, $x_3(0) = 15$

porównaj to rozwiązanie z rozwiązaniem dla

$x_1(0) = 15$, $x_2(0) = 10.2$, $x_3(0) = 15$

b) powtórz czynności z punktu a) dla $\beta = 30$.

(Porównuj zmiany x_1)

Rozwiązanie

a) Definiujemy funkcję obliczającą prawe strony układu (tym razem zależną od parametrów)

```
lorenz <- function(t,x,parms) {
with(as.list(parms), {
dx1 <- alfa*(x[2]-x[1])
dx2 <- beta*x[1]-x[2]-x[1]*x[3]
dx3 <- x[1]*x[2]-gamma*x[3]
list(c(dx1,dx2,dx3))})}
```

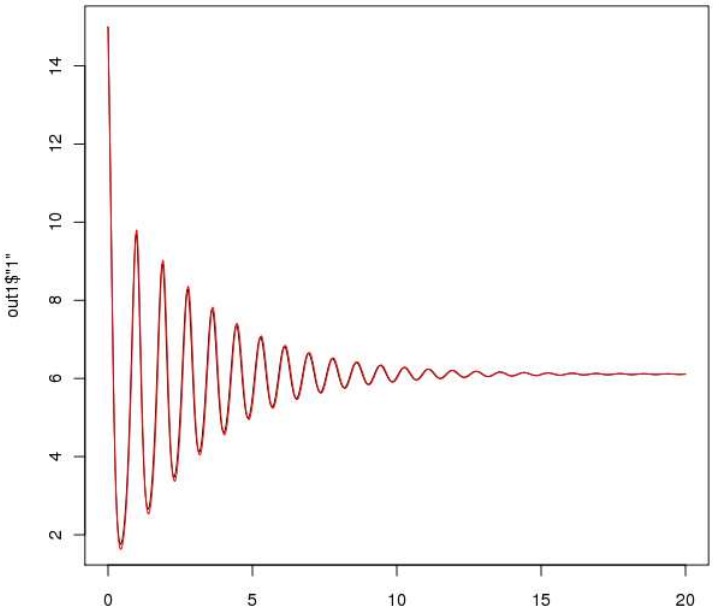
Określamy parametry, warunki początkowe i zakres czasu

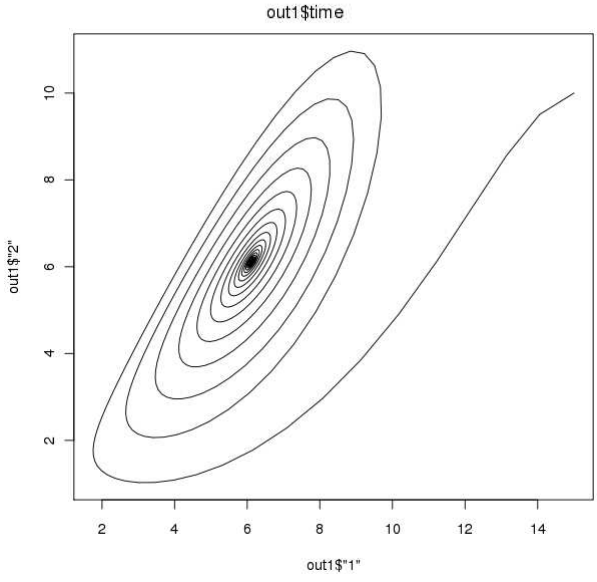
```
parms=c(alfa=10,beta=15,gamma=8/3)
x01=c(15,10,15)
x02=c(15,10.2,15)
time=seq(0,20,0.02)
```

Rozwiązujemy układ dla dwu różnych zestawów warunków początkowych, rysujemy zależność x_1 od czasu oraz x_2 od x_1 dla pierwszego przypadku ($x_2(0) = 10$).

```
out1=as.data.frame(rk4(x01,time,lorenz,parms))
out2=as.data.frame(rk4(x02,time,lorenz,parms))
plot(out1$time,out1$"1",type="l")
lines(out2$time,out2$"1",col="red")
plot(out1$"1",out1$"2",type="l")
```

otrzymując wykresy



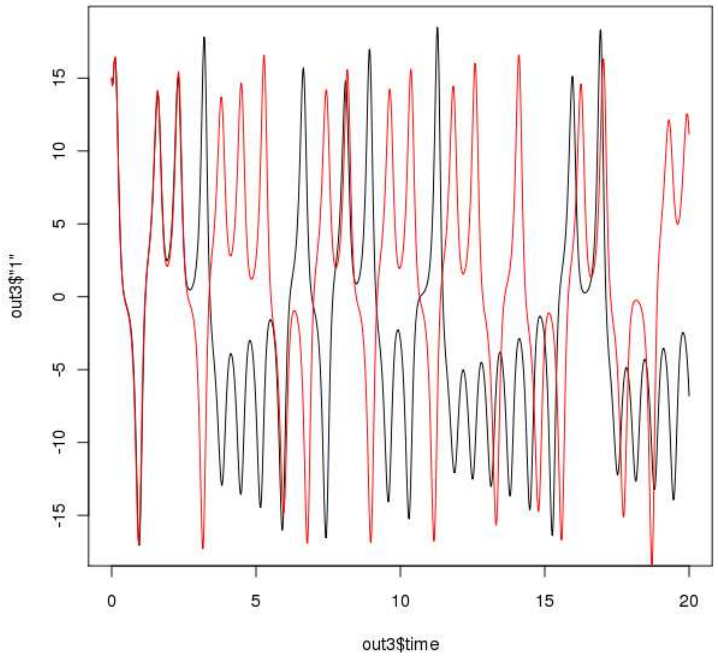


Zmienna x_1 oscylując dąży do stałej wartości, podobnie zachowuje się x_2, co widać z trajektorii fazowej (krzywa spiralnie zbiega się do punktu - atraktora). Zależności otrzymane przy małej zmianie warunków początkowych różnią się niewiele.

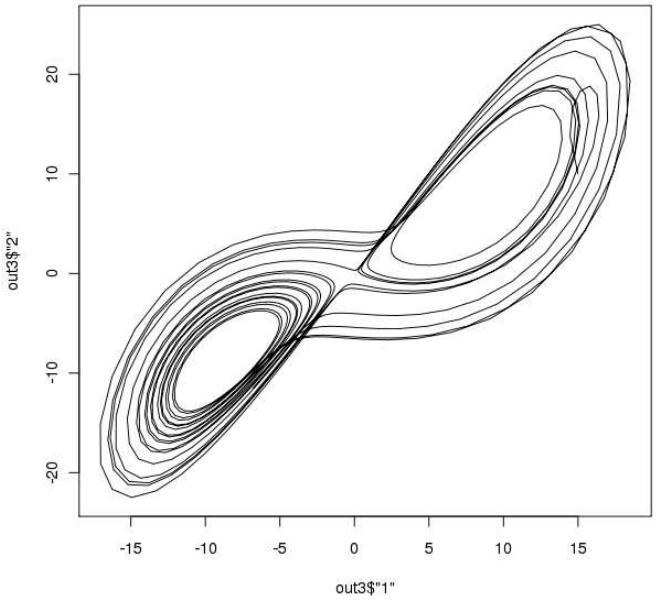
b) Zmieniamy wartość parametru β i powtarzamy operacje

```
parms=c(alfa=10,beta=30,gamma=8/3)
out3=as.data.frame(rk4(x01,time,lorenz,parms))
out4=as.data.frame(rk4(x02,time,lorenz,parms))
plot(out3$time,out3$"1",type="l")
lines(out4$time,out4$"1",col="red")
plot(out3$"1",out3$"2",type="l")
```

Jak widać, dla $\beta = 30$ mała zmiana w warunkach początkowych prowadzi do zupełnie innego rozwiązania



Jest to przykład chaotycznego zachowania układu. Charakterystyczny jest wykres zależności x_2 od x_1



Przedstawiony twór to tzw. atraktor Lorenza (właściwie jego rzut na płaszczyznę, bo ze względu na trzy zmienne atraktor ten jest trójwymiarowy).

8. Rozważmy równanie różniczkowe wymuszonego oscylatora z tłumieniem

$$\ddot{x} + 0.05\dot{x} + x^3 = 7.5\cos t$$

a) pokaż, że równanie to jest równoważne układowi trzech równań pierwszego rzędu

b) obejrzyj jego rozwiązanie dla $t \in [0,100]$ i warunków początkowych $x(0) = 2,\ \dot{x}(0) = 0$. Porównaj z rozwiązaniem dla $x(0) = 2.002$. Narysuj portret fazowy: zależność $\dot{x}$ od x.

Rozwiązanie.

a) Przyjmując

$$x_1 = x \text{ oraz } x_2 = x_1'$$

dostajemy układ

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= 7.5\cos t - 0.05x_2 - x_1^3 \end{aligned} \qquad (*)$$

Przekształcając go do układu autonomicznego (tzn. takiego, w którym po prawej stronie czas nie występuje jawnie) wprowadzamy trzecią zmienną

$$x_3 = t$$

i otrzymujemy układ trzech równań rzędu pierwszego

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= 7.5\cos x_3 - 0.05x_2 - x_1^3 \\ x_3' &= 1 \end{aligned} \qquad (**)$$

b) W zasadzie moglibyśmy rozwiązywać układ (**) (z warunkiem $x_3(0) = 0$), ale zauważmy, że w R i tak musimy zdefiniować prawe strony zależne od czasu. Wobec tego wygodniejsza będzie postać (*). Warunki początkowe to $x_1(0) = 2$ (lub 2.002) i $x_2(0) = 0$.

Definiujemy funkcję

```
osc <- function(t,x,parms) {
dx1 <- x[2]
dx2 <- 7.5*cos(t)-0.05*x[2]-x[1]^3
list(c(dx1,dx2))}
```

Zadajemy warunki początkowe i czas całkowania

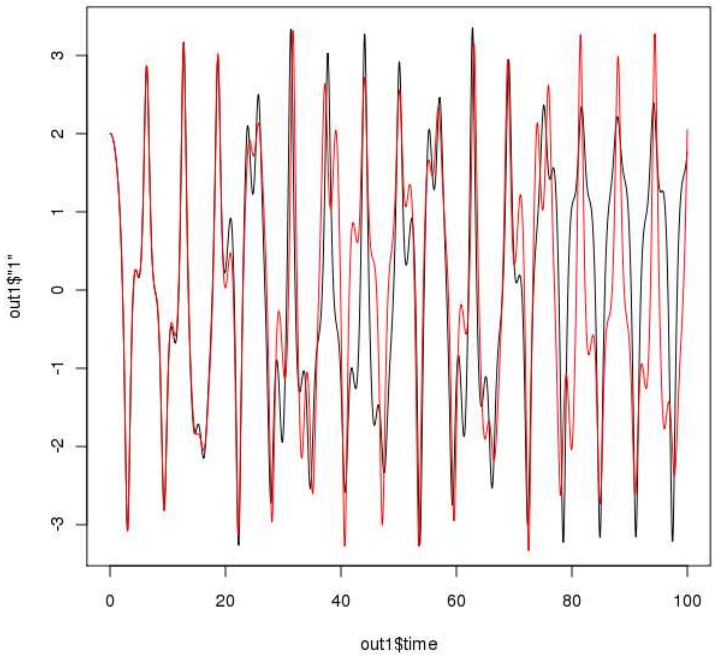
```
parms=0
time=seq(0,100,0.02)
x0a=c(2,0)
x0b=c(2.002,0)
```

i znajdujemy rozwiązania

```
out1=as.data.frame(rk4(x0a,time,osc,parms))
out2=as.data.frame(rk4(x0b,time,osc,parms))
```

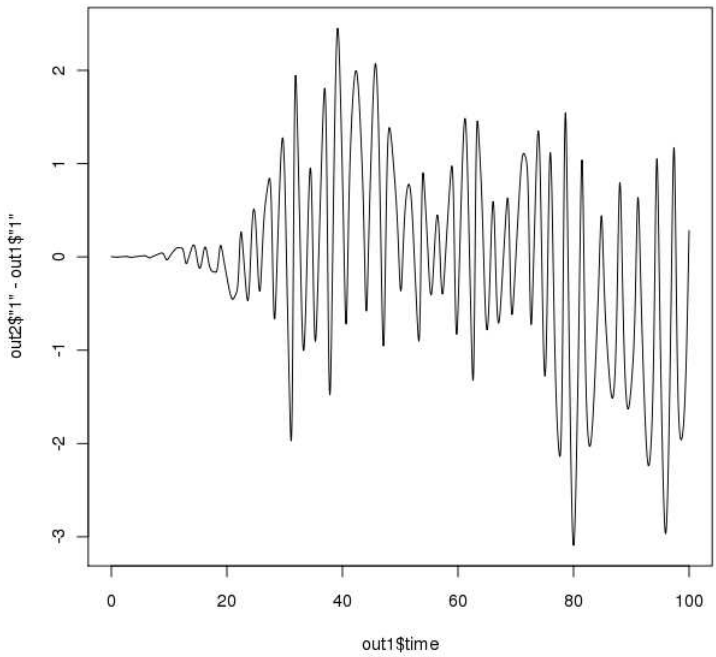
Ponieważ nasze wyjściowe x to x_1 a $\dot{x}$ to x_2, więc wykres zmian x w czasie otrzymujemy poleceniami

```
plot(out1$time,out1$"1",type="l")
lines(out2$time,out2$"1",col="red")
```



Z wykresu widać, że przy drobnej zmianie warunków początkowych rozwiązania początkowo są zbliżone, lecz w miarę upływu czasu zaczynają się rozjeżdżać, co możemy stwierdzić wykreślając ich różnicę

```
plot(out1$time,out2$"1"-out1$"1",type="l")
```

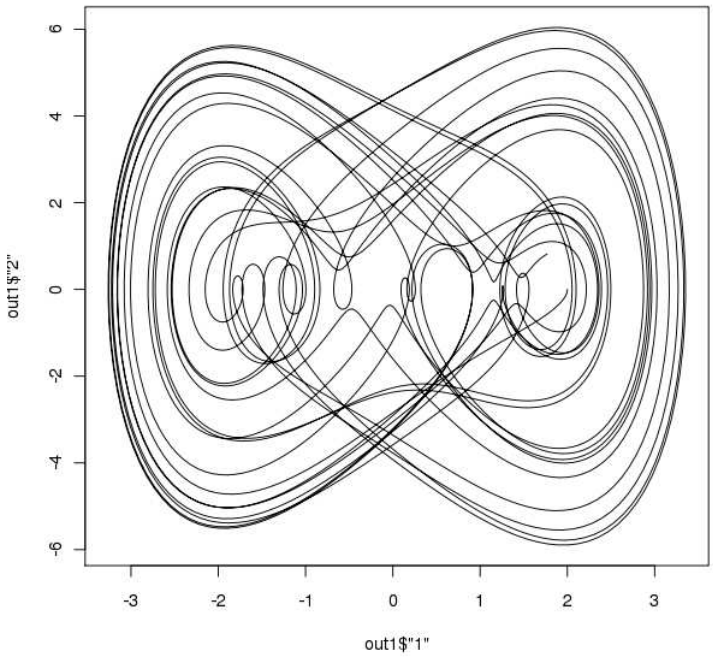


Jest to zachowanie typowe dla układów chaotycznych.

Wykreślmy jeszcze zależność $\dot{x}$ od x poleceniem

```
plot(out1$"1",out1$"2",type="l")
```

otrzymując charakterystyczny obraz trajektorii fazowej



III. RÓWNANIA RÓŻNICZKOWE CZĄSTKOWE

9. Określ, jakiego typu są następujące równania różniczkowe cząstkowe:
a) równanie przewodnictwa cieplnego

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{\alpha^2}\frac{\partial T}{\partial t}$$

b) równanie Laplace'a

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

c) równanie falowe

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{\upsilon^2}\frac{\partial^2 u}{\partial t^2}$$

Rozwiązanie.

Typ równania cząstkowego dwu zmiennych

$$a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} + d = 0$$

gdzie d nie zależy od drugich pochodnych cząstkowych określamy na podstawie znaku wyrażenia $b^2 - 4ac$

a) $a = 1, b = c = 0$, $b^2 - 4ac = 0$ równanie paraboliczne
b) $a = c = 1, b = 0$, $b^2 - 4ac = -4 < 0$ równanie eliptyczne
c) $a = 1, b = 0, c = -1/\upsilon^2$ $b^2 - 4ac = 4/\upsilon^2 > 0$ równanie hiperboliczne