INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Moderní přístup k aplikaci matematických dovedností v přírodovědných a ekonomických oborech
reg. č.: CZ.1.07/2.2.00/28.0168

Software Mathematica pro fyziky

František Látal, Lukáš Richterek, Ivo Vyšín, Marie Volná, Jan Říha

Olomouc 2015

Oponenti: doc. Ing. Luděk Bartoněk, Ph.D.
Mgr. Marek Honců, Ph.D.

Tato publikace neprošla redakční jazykovou úpravou.

Olomouc 2015
1. vydání

ISBN 978-80-244-4470-3

Obsah

1 Úvod

Publikace se věnuje využití programu *Mathematica* (verze 10.0) při řešení náročnějších fyzikálních problémů. V textu jsou vybrány některé zajímavé úlohy, které jsou zpracovány a popsány s využitím programu *Mathematica*. Matematika je dnes nezbytný obor pro studium v přírodovědných disciplínách a odpovídajících učitelských oborech. V době výkonných počítačů, notebooků či moderních tabletů je nezbytné zahrnout tyto progresivní prostředky do výuky a vést studenty k jejich smysluplnému a efektivnímu využívání. Díky moderním technologiím vzniká prostor k řešení náročnějších problémů. Implementace profesionálního matematického softwaru *Mathematica* přispívá ke zkvalitnění vyučovacího procesu dnes velmi náročných oborů, jako je fyzika, biologie i chemie.

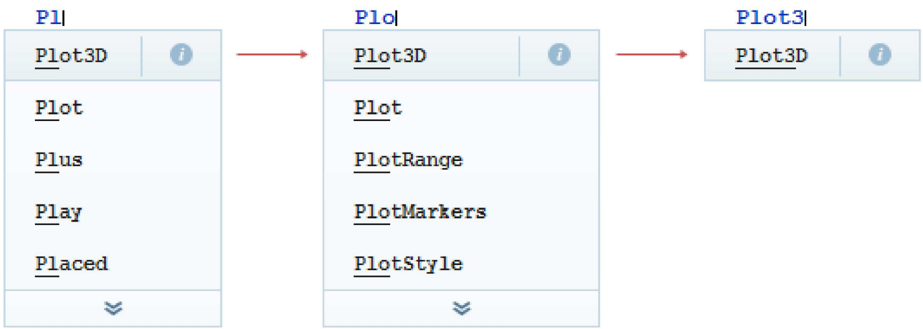
Publikace *Software Mathematica pro fyziky* je rozdělena do osmi kapitol včetně úvodu, kde jsou shrnuta základní pravidla pro práci se softwarem *Mathematica*. V druhé kapitole jsou uvedeny některé fyzikální úlohy, které lze v softwaru *Mathematica* jednoduše řešit pomocí hledání lokálních extrémů funkcí. Následující dvě kapitoly textu se zaměřují na zpracování dat jednoduchých měření, např. zkoumání závislosti elektrického odporu na teplotě, analýza závislosti tlaku sytých vodních par na teplotě či zpracování dat z měření tíhového zrychlení pomocí matematického a reverzního kyvadla. Tyto příklady a výpočty lze využívat v rámci praktických měření v laboratoři fyziky. V kapitole číslo pět je uvedeno několik příkladů a aplikací při řešení problémů z oblasti klasické mechaniky, např. Keplerova úloha, Buquoyova úloha a Van der Polův oscilátor. Další kapitola se zabývá interpretací výsledků řešených problémů z teorie elektromagnetického pole. Pozornost je věnována vybraným aplikacím v elektrostatice a v nestacionárním elektromagnetickém poli. Kromě klasické mechaniky jsou řešeny i otázky z moderní mechaniky, např. Planckův vyzařovací zákon, vývoj hustoty pravděpodobnosti částice v jednorozměrné nekonečně hluboké potenciálové jámě nebo vývoj hustoty pravděpodobnosti lineárního harmonického oscilátoru. Díky profesionálnímu softwaru je jeden oddíl textu věnován modelování fraktálů ve 2D i 3D rozměru, což by bez použití kvalitního programu nebylo nikdy možné realizovat. Poslední kapitola Obecná teorie relativity se zabývá výpočty Christoffelových symbolů, Ricciho a Einsteinova tenzoru pro daný metrický tenzor nebo pohybem částic v okolí Schwarzschildovy černé díry.

1.1 Základní pravidla práce v programu *Mathematica*

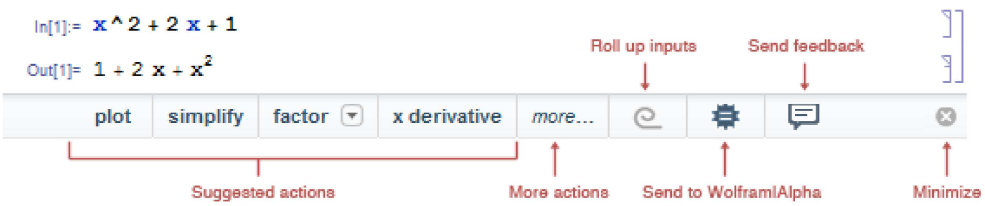
Veškerá práce v programu *Mathematica* se provádí v Notebooku. Pro správné zobrazení zadaných úloh, je nutné dodržovat několik základních pravidel:

- V první řadě je důležité stanovit formát v jakém bude buňka napsána, zda se bude jednat o text, příkaz, nadpis, kód, číslovaný matematický vzorec apod. Toto rozdělení najdeme v horním řádku Format → Style.
- Pro matematické výpočty, tvorbu grafů nebo simulaci použijeme styl buňky Input. Příkaz bude označen In[n]:= a výsledek se zobrazí v buňce s označením Out[n]=.
- Provedení příkazu v buňce se provede po stisknutí kombinace kláves SHIFT+ENTER nebo stisknutím klávesy ENTER na numerické klávesnici.
- Pokud se na konci příkazu napíše středník, tak se příkaz vykoná, ale výsledek se nezobrazí v Notebooku.
- Názvy funkcí se píší vždy velkým písmenem na začátku a argumenty funkcí se uvádí v hranatých závorkách.
- Pokud si nenadefinujeme jinak, tak platí, že k zápisu desetinného čísla se používá tečka.
- Násobení symbolů může být reprezentováno mezerou.
- Důležitým pomocníkem při práci v Notebooku jsou Palety, které umožňují jednoduchým způsobem využívat funkce programu. Palety najdeme v horním řádku pod názvem Palettes.
- Pokud neznáme přesnou syntaxi, jak zapsat některý příkaz, lze využít tzv. free-form input. Zápisem znaku = na začátku buňky Input (např. = sin x from pi to 2pi), dojde s využitím serveru WolframAlpha k převedení příkazu do správné syntaxe (v tomto případě Plot[Sin[x], {x, Pi, 2Pi}]) a poté k provedení příkazu. Při zápisu znaku == na začátku buňky Input dostaneme plný výstup, který vytvoří server Wolfram Alpha.
- V rovnicích znamená symbol = přiřazení (definování) nové proměnné, pro rovnost je potřeba napsat symbol ==.

- Vytvořený program v softwaru *Mathematica* lze uložit ve formátu *.nb. Novou možností je ovšem uložení programu ve formátu Computable Document *.cdf. Programy v tomto formátu lze vkládat např. na webové stránky, kde si jej mohou spustit i uživatelé bez softwaru *Mathematica*, stačí, když si zdarma nainstalují *Wolfram CDF Player*.
- Od verze *Mathematica* 9 došlo k výraznému zjednodušení práce s programem. *Input Assistant* automaticky dokončuje příkazy, které chce uživatel do programu napsat.



- Od verze *Mathematica* 9 lze také využívat *Suggestion Bar*, který umožňuje jednoduše doplnit a vylepšit výsledek matematického příkazu.



- Verze *Mathematica* 10 přinesla možnost vrátit libovolný počet kroků zpět při práci v programu. Dřívější verze umožňovaly pouze jeden krok zpět.
- Software *Mathematica* obsahuje podrobnou anglicky psanou nápovědu, kterou lze nalézt v horním řádku softwaru Help → Documentation Center. Rozsáhlou podporu pro práci v programu *Mathematica* nabízí webová stránka www.wolfram.com.

2 Fyzikální úlohy vedoucí k zjišťování lokálních extrémů funkcí

2.1 Obecný postup hledání lokálních extrémů

Při hledání lokálních extrémů funkcí lze použít tento obecný postup řešení:

1. Nejprve vypočítáme první derivaci funkce.
2. První derivaci položíme rovnu nule a vypočítáme hodnoty x_0, pro které je první derivace rovna nule.
3. Vypočítáme druhou derivaci funkce.
4. Dosadíme hodnoty x_0 do druhé derivace.
5. Je-li $y''(x_0) > 0$, nastává v bodě x_0 minimum, je-li $y''(x_0) < 0$, nastává v bodě x_0 maximum. Je-li $y''(x_0) = 0$, můžeme v bodě x_0 dostat inflexní bod.

Poznámka 1:
Výpočet druhé derivace není nutné vždy provádět, někdy stačí pozorovat funkční hodnoty bodů v okolí bodu, v němž je první derivace rovna nule.

Poznámka 2:
Při vyšetřování průběhu funkce je třeba si uvědomit, že jinak se chová např. mocninná funkce se sudým exponentem a jinak mocninná funkce s lichým exponentem, viz následující dva příklady.

Příklad:

Pomocí obecného postupu hledání extrémů funkce nejděte extrémy u funkce $f : y = x^4$.

Řešení:

Nejprve vypočítáme 1. derivaci funkce f.

```
In[1]:= D[x^4, x]
Out[1]= 4 x^3
```

Vypočítáme hodnoty x_0, pro které je první derivace rovna nule.

```
In[2]:= Solve[4 x^3 == 0, x]
Out[2]= {{x → 0}, {x → 0}, {x → 0}}
```

Vypočítáme druhou derivaci funkce f.

```
In[3]:= D[4 x^3, x]
Out[3]= 12 x^2
```

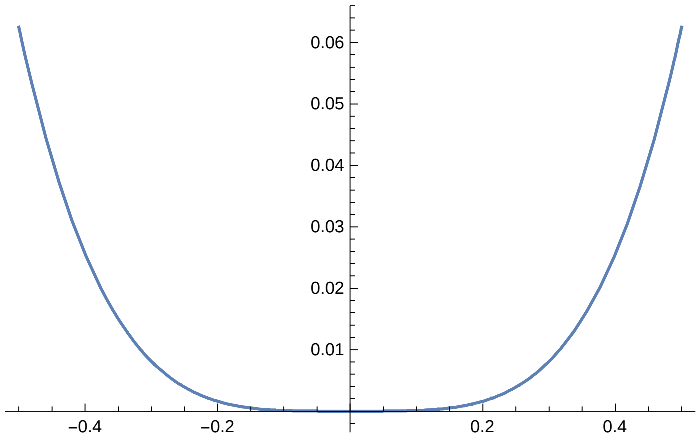
Dosadíme hodnoty x_0 do druhé derivace a rozhodneme, zda se jedná o maximum nebo minimum.

```
In[4]:= 12 × 0^2
Out[4]= 0
```

Došli jsme k výsledku, kde nelze pomocí tohoto obecného postupu stanovit, zda je bod $x_0 = 0$ maximem nebo minimem zkoumané funkce. Graf funkce $f : y = x^4$ je ovšem obecně známý. Z tohoto grafu vidíme, že v bodě $x_0 = 0$ nastává minimum zkoumané funkce.

In[5]:= `Plot[x^4, {x, -0.5, 0.5}]`

Out[5]=



Příklad:

Pomocí obecného postupu hledání extrémů funkce nejděte extrém u funkce $g : y = x^3$.

<u>Řešení</u>:

Nejprve vypočítáme první derivaci funkce g a poté položíme tuto derivaci rovnu nule.

In[6]:= `D[x^3, x]`

Out[6]= $3\,x^2$

In[7]:= `Solve[3 x^2 == 0, x]`

Out[7]= `{{x → 0}, {x → 0}}`

Poté vypočítáme druhou derivaci funkce g a stanovíme, zda “body podezřelé z extrému” jsou minima či maxima zkoumané funkce.

In[8]:= `D[3 x^2, x]`

Out[8]= $6\,x$

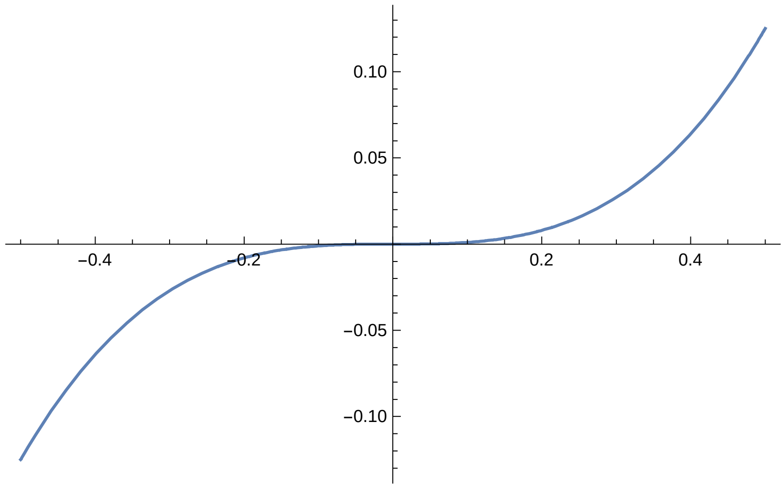
In[9]:= `6 × 0`

Out[9]= 0

Došli jsme k úplně stejnému výsledku, jako v předchozím příkladu (nelze pomocí tohoto obecného postupu stanovit, zda je bod $x_0 = 0$ maximem nebo minimem zkoumané funkce). Graf funkce $g : y = x^3$ je ovšem obecně známý. Z tohoto grafu vidíme, že v bodě $x_0 = 0$ nenastává maximum ani minimum zkoumané funkce, ale jedná se o inflexní bod.

In[10]:= `Plot[x^3, {x, -0.5, 0.5}]`

Out[10]=



Příklad:

Najděte lokální extrémy funkce $h: y = \frac{x^5}{5} + x^4$.

<u>Řešení</u>:

Provedeme obecný postup hledání extrémů funkce.

In[11]:= $D\left[\frac{x^5}{5} + x^4, x\right]$

Out[11]= $4 x^3 + x^4$

In[12]:= $\text{Solve}[4 x^3 + x^4 == 0, x]$

Out[12]= $\{\{x \to -4\}, \{x \to 0\}, \{x \to 0\}, \{x \to 0\}\}$

In[13]:= $D[4 x^3 + x^4, x]$

Out[13]= $12 x^2 + 4 x^3$

In[14]:= $12 (-4)^2 + 4 (-4)^3$

Out[14]= -64

In[15]:= $12 (0)^2 + 4 (0)^3$

Out[15]= 0

Z provedeného postupu vidíme, že bod $x_1 = -4$ je lokálním maximem zkoumané funkce h, ale pro bod $x_2 = 0$ nelze podle tohoto postupu rozhodnout, zda se jedná o maximum, nebo minimum. Zde můžeme využít vlastnosti první derivaci funkce h v okolí bodu $x_2 = 0$. Zvolíme si levé okolí bodu $x_2 = 0$ interval $(-4, 0)$, pravé okolí bodu $(0, \infty)$ a provedeme první derivace v libovolných bodech z těchto dvou intervalů.

In[16]:= $D\left[\frac{x^5}{5} + x^4, x\right]$

Out[16]= $4 x^3 + x^4$

In[17]:= $4 (-2)^3 + (-2)^4$

Out[17]= -16

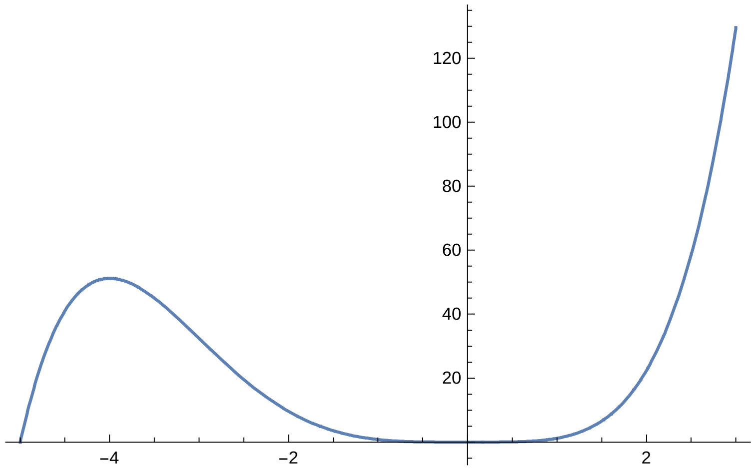
In[18]:= `4 (2)^3 + (2)^4`

Out[18]= 48

Jelikož je první derivace funkce *h* v bodě -2 záporná, je tato funkce v intervalu (-4,0) klesající. První derivace funkce *h* je v bodě 2 kladná, a tedy na intervalu (0,∞) je tato funkce rostoucí. Z toho vyplývá, že v bodě $x_2 = 0$ nastává lokální minimum zkoumané funkce, což si můžeme ověřit pomocí grafu funkce *h*.

In[19]:= $\texttt{Plot}\left[\frac{x^5}{5} + x^4, \{x, -5, 3\}\right]$

Out[19]=



2.2 Fyzikální úlohy vedoucí k zjišťování lokálních extrémů funkcí

Při řešení fyzikálních úloh, kde je třeba nalézt lokální extrém, je vhodné zvolit následující postup:

1. Nejprve zavedeme proměnné a přiřadíme jim symboly.
2. Vyjádříme veličinu, jejíž maximum nebo minimum budeme hledat, pomocí těchto proměnných.
3. Zjistíme vztahy mezi proměnnými ze zadání.
4. Vyjádříme veličinu, která se má maximalizovat nebo minimalizovat pomocí jediné proměnné.

Příklad:

Najděte nejekonomičtější tvar válcové nádoby s jednou podstavou (např. hrnec bez pokličky) s pevně daným objemem. Jinak řečeno, jaký je nejmenší povrch takovéto nádoby?

Řešení:

1. Nejprve zavedeme proměnné a přiřadíme jim symboly:
r … poloměr nádoby
h … výška nádoby

2. Vyjádříme veličinu, jejíž maximum nebo minimum budeme hledat, pomocí těchto proměnných:

$$S = \pi r^2 + 2 \pi r h = \pi (r^2 + 2 r h)$$

3. Zjistíme vztahy mezi proměnnými ze zadání:
Pro objem válcové nádoby s jednou podstavou platí:

$$V = \pi r^2 h$$

4. Vyjádříme veličinu, která se má maximalizovat nebo minimalizovat pomocí jediné proměnné:

Předchozí dva vztahy upravíme a dostaneme:

$S = \pi\left(r^2 + \frac{2V}{\pi r}\right)$

Hledáme minimum funkce S pro proměnnou $r > 0$.

In[20]:=
```
D[π (r^2 + 2 V/(π r)), r]
```

Out[20]= $\pi\left(2r - \frac{2V}{\pi r^2}\right)$

In[21]:=
```
Reduce[π (2 r - 2 V/(π r^2)) == 0 && 0 < r, {r, V}]
```

Out[21]= $r > 0$ && $V == \pi r^3$

Nyní porovnáme výsledný vztahů získáme:

In[22]:=
```
Reduce[π r^3 == π r^2 h && 0 < r, {r, h}]
```

Out[22]= $r > 0$ && $h == r$

Aby měla nádoba ze zadání co nejekonomičtější rozměry musí platit, že výška nádoby je rovna jejímu poloměru $h = r$. Pomocí druhé derivace potvrdíme, že se opravdu jedná o minimum funkce.

In[23]:=
```
D[π (2 r - 2 V/(π r^2)), r]
```

Out[23]= $\pi\left(2 + \frac{4V}{\pi r^3}\right)$

Druhá derivace je pro $V > 0$ a $r > 0$ kladná. Jedná se tedy o minimum funkce.

Příklad:

Nádoba je naplněna kapalinou do výšky h nad dnem nádoby. V jaké výšce h_0 nad dnem nádoby je třeba navrtat otvor, aby měla kapalina maximální dostřik?

Řešení:

Pro souřadnice libovolného bodu trajektorie vodorovného vrhu platí:

$x = v_0 \cdot t,$

$y = h_0 - \frac{1}{2} g t^2.$

Pro výtokovou rychlost kapaliny v_0 platí:

$v = \sqrt{2(h - h_0) \cdot g}\ .$

V okamžiku dopadu platí:

$y = 0, \ x = d,$

$0 = h_0 - \frac{1}{2} g {t_d}^2 \Rightarrow t_d = \sqrt{\frac{2 h_0}{g}}\ ,$

$d = v_0 \cdot t_d = 2\sqrt{h \cdot h_0 - h_0^2}\ .$

Nyní chceme zjistit, kdy vzdálenost dostřiku d bude maximální v závislosti na výšce otvoru nad dnem h_0. Bude-li d maximální, bude také d^2 maximální, a proto budeme hledat maximum funkce d^2.

In[24]:=
```
d2 = 4 (h h0 - h0^2);
```

In[25]:=
```
D[d2, h0]
```
Out[25]= $4 (h - 2 h_0)$

In[26]:=
```
Reduce[4 (h - 2 h0) == 0 , {h, h0}]
```
Out[26]= $h_0 == \frac{h}{2}$

Vidíme, že maximální dostřik nastane, když je otvor navrtán v $\frac{1}{2}$ výšky sloupce kapaliny. Nyní pomocí druhé derivace můžeme potvrdit, že jde opravdu o maximum funkce.

In[27]:=
```
D[4 (h - 2 h0), h0]
```
Out[27]= -8

Příklad:

Z papíru formátu A4 (210 x 297 mm) vystřihni v rozích čtyři stejné čtverečky, aby složením vzniklého obrazce vznikla krabička maximálního objemu.

Řešení:

x ... délka strany čtverečku, který vystřihneme z jednoho rohu papíru
297 - 2*x* ... délka nově vzniklé krabičky
210 - 2*x* ... šířka nově vzniklé krabičky
x ... výška nově vzniklé krabičky

In[28]:=
```
ak = 297 - 2 x;
bk = 210 - 2 x;
ck = x;
```

Krabička bude mít tvar kvádru a pro jeho objem platí: $V = a\,b\,c$.

In[31]:=
```
Vk = ak bk ck
```
Out[31]= $(210 - 2 x) (297 - 2 x) x$

In[32]:=
```
D[Vk, x] // Simplify
```
Out[32]= $6 \left(10\,395 - 338 x + 2 x^2\right)$

Po provedení první derivace položíme tuto derivaci rovnu nule. Výška krabičky *x* nemůže být větší než je polovina šířky papíru.

In[33]:=
```
NSolve[6 (10395 - 338 x + 2 x^2) == 0 && 105 > x, {x}]
```
Out[33]= {{x → 40.4234}}

Z kvadratické rovnice vidíme, že podmínkám vyhovuje pouze jeden kořen a to 40.4234 mm. Nyní ještě potvrdíme, že se jedná o maximum funkce.

In[34]:=
```
D[6 (10395 - 338 x + 2 x^2), x]
```
Out[34]= $6 (-338 + 4 x)$

In[35]:=
```
6 (-338 + 4 (40.4234))
```
Out[35]= -1057.84

Příklad:

Teplotní objemovou roztažnost vody lze v rozmezí teplot od 0° C do 30° C popsat rovnicí $V = V_0 (1 + \beta_1 t + \beta_2 t^2 + \beta_3 t^3)$, kde $\beta_1 = -6.43 \cdot 10^{-5}\,\mathrm{K}^{-1}$, $\beta_2 = 8.51 \cdot 10^{-6}\,\mathrm{K}^{-2}$, $\beta_3 = -6.79 \cdot 10^{-8}\,\mathrm{K}^{-3}$ jsou empiricky určené hodnoty a V_0 je objem při teplotě

0° C. Určete při jaké teplotě je podle uvedené aproximace objem v daném intervalu teplot minimální a při jaké teplotě maximální.

Řešení:

Nejprve provedeme první derivaci funkce a tuto derivaci položíme rovnu nule.

In[36]:=
```
D[V0 (1 + B1 t + B2 t^2 + B3 t^3), t]
```

Out[36]= $\left(B_1 + 2\,t\,B_2 + 3\,t^2\,B_3\right) V_0$

In[37]:=
```
B1 := -6.43 * 10^-5;
B2 := 8.51 * 10^-6;
B3 := -6.79 * 10^-8;
```

In[40]:=
```
Solve[V0 (B1 + 2 t B2 + 3 t^2 B3) == 0, {t}];
```

Výsledkem kvadratické rovnice jsou dva kořeny. Pomocí druhé derivace určíme maximum nebo minimum.

In[41]:=
```
D[V0 (B1 + 2 t B2 + 3 t^2 B3), t]
```

Out[41]= $\left(0.00001702 - 4.074 \times 10^{-7}\,t\right) V_0$

Dosazením $t = 3.96618°$ C:

In[42]:=
```
(0.00001702 - 4.074 × 10^-7 3.96618) V0
```

Out[42]= $0.0000154042\,V_0$

Vidíme, že tato hodnota odpovídá minimu funkce.

Dosazením $t = 79.5881°$ C:

In[43]:=
```
(0.00001702 - 4.074 × 10^-7 79.5881) V0
```

Out[43]= $-0.0000154042\,V_0$

Vidíme, že tato hodnota odpovídá maximu funkce.
Jelikož v zadání je interval omezen na hodnoty od 0° C do 30° C, lze říci, že v tomto intervalu je objem vody nejmenší při teplotě 3.96618° C a největší při teplotě 30° C.

Můžeme zobrazit grafické znázornění závislosti objemu vody na teplotě v intervalu 0° C do 100° C podle kubického polynomu $V = V_0 (1 + \beta_1 t + \beta_2 t^2 + \beta_3 t^3)$.

In[44]:=
$$\text{FindMinimum}\left[1 - \frac{6.43\,t}{10^5} + \frac{8.51\,t^2}{10^6} - \frac{6.79\,t^3}{10^8}, \{t, 0\}\right]$$

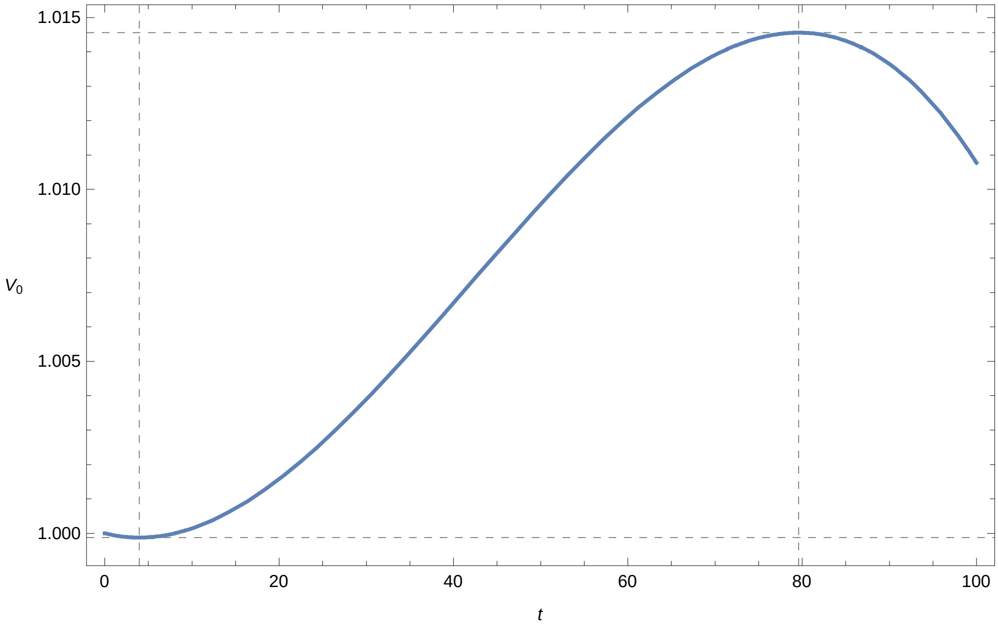
Out[44]= $\{0.999875, \{t \to 3.96618\}\}$

In[45]:=
$$\text{FindMaximum}\left[1 - \frac{6.43\,t}{10^5} + \frac{8.51\,t^2}{10^6} - \frac{6.79\,t^3}{10^8}, \{t, 70\}\right]$$

Out[45]= $\{1.01456, \{t \to 79.5881\}\}$

In[46]:=
```
Plot[1 - 6.43 t/10^5 + 8.51 t^2/10^6 - 6.79 t^3/10^8, {t, 0, 100},
 PlotStyle → Directive[Opacity[1], AbsoluteThickness[2]],
 Frame → True, FrameLabel → {t, V0}, RotateLabel → False, GridLines →
  {{3.9661758658454733, 79.58807057392566}, {0.9998746055908817, 1.0145565116818087 }},
 GridLinesStyle → Directive[Gray, Dashed]]
```

Out[46]=



3 Zpracování dat jednoduchých měření a regresní analýza

Při zpracování naměřených hodnot se už ve fyzikálním praktiku setkáváme s potřebou nalézt závislost mezi měřenými veličinami. Program *Mathematica* poskytuje řadu nástrojů k řešení takových úloh.

Při výpočtech bývá užitečné vypsat nejen číselnou hodnotu výrazu, ale i jeho označení symbolem. Za tímto účelem definujme funkci **nameAndValue**, v níž pro potřeby textu zaokrouhlujeme na 3 platné číslice a *n* desetinných míst, anglosasskou desetinnou tečku můžeme nahradit desetinnou čárkou.

```
In[47]:= SetAttributes[nameAndValue, HoldFirst];
nameAndValue[x_, n_] :=
  Print[Unevaluated[x], "=", NumberForm[x, {3, n}, NumberPoint → ","]];
nameAndValue[aa = N[π, 30], 3]
aa

aa = N[π, 30]=3,140

Out[50]= 3.14159265358979323846264338328
```

3.1 Závislost elektrického odporu na teplotě

Výsledky měření elektrického odporu vodiče v závislosti na teplotě jsou dány v tabulce:

$\frac{t}{°C}$	19.0	25.0	30.2	36.0	40.2	45.3	50.0
$\frac{R}{\Omega}$	76.3	77.8	79.7	80.9	82.4	84.0	85.1

které zadáme ve tvaru výčtu hodnot.

Teplota ve °C:

```
In[51]:= teplota = {19.0, 25.0, 30.2, 36.0, 40.2, 45.3, 50.0}

Out[51]= {19., 25., 30.2, 36., 40.2, 45.3, 50.}
```

Odpor v Ω:

```
In[52]:= odpor = {76.3, 77.8, 79.7, 80.9, 82.4, 84.0, 85.1}

Out[52]= {76.3, 77.8, 79.7, 80.9, 82.4, 84., 85.1}
```

Příkaz **Riffle** vyvoří seznam, kde se střídají hodnoty z prvního a druhého výčtu hodnot, příkaz **Partition** s parametrem 2 vytvoří dvojice ze sousedních hodnot:

```
In[53]:= hodnoty = Partition[Riffle[teplota, odpor], 2]

Out[53]= {{19., 76.3}, {25., 77.8}, {30.2, 79.7},
  {36., 80.9}, {40.2, 82.4}, {45.3, 84.}, {50., 85.1}}
```

Protože můžeme předpokládat lineární závislost odporu na teplotě, můžeme použít funkci **LinearModelFit**:

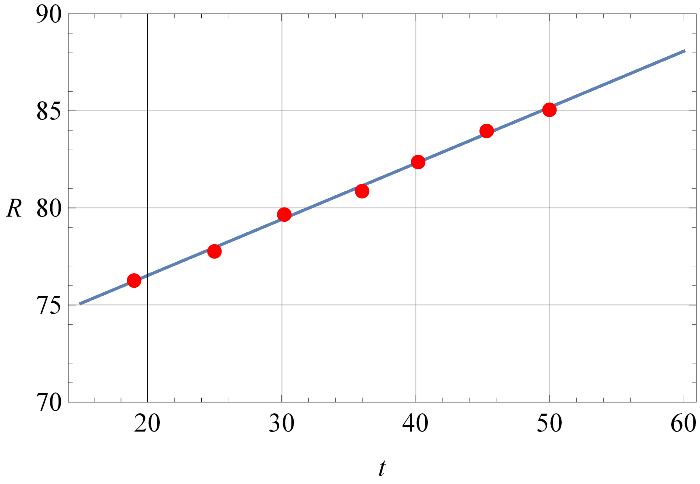
```
In[54]:= model = LinearModelFit[hodnoty, x, x]

Out[54]= FittedModel[ 70.7518 + 0.288714 x ]
```

Naleznou funkční závislost lze uložit pro další použití příkazem

In[55]:=
```
model["BestFit"]
```
Out[55]=
```
70.7518 + 0.288714 x
```

a např. vykreslit v odpovídajícím intervalu spolu s funkčními hodnotami.

In[56]:=
```
Show[{Plot[model["BestFit"], {x, 15, 60}, PlotRange → {Automatic, {70, 90}},
   PlotStyle → {Thickness[0.005]}, FrameLabel → {"t", "R"},
   RotateLabel → False, BaseStyle → {FontFamily → "Times", FontSize → 14},
   Frame → True, GridLines → Automatic],
  ListPlot[hodnoty, PlotMarkers → {Automatic, Medium}, PlotStyle → {Red}]}]
```
Out[56]=



Interpretaci získaných hodnot provedeme v souladu s teoretickým vztahem pro závislost teploty na odporu $R = R_0(1 + \alpha \Delta t) = a + bx$.

Odtud po zaokrouhlení na tři platné číslice (v souladu s naměřenými hodnotami) získáváme $R_0 = 70.8\,\Omega$, $\alpha = \frac{b}{a}$, neboli

In[57]:=
```
α = ScientificForm[Coefficient[model["BestFit"], x, 1] /
   Coefficient[model["BestFit"], x, 0], 3, NumberPoint → ","]
```
Out[57]//ScientificForm=
```
4,08 × 10^-3
```

Teplotní součinitel odporu tak vychází $\alpha = 4.08 \cdot 10^{-3}\,K^{-1}$. Zbývá ještě posoudit kvalitu modelu regrese pomocí běžně užívaných veličin – koeficientu determinace r^2, koeficientu korelace r a chyby v hodnotách získaných parametrů.

In[58]:=
```
NumberForm[{model["RSquared"], Sqrt[model["RSquared"]], model["ParameterErrors"]},
  NumberPoint → "."]
```
Out[58]//NumberForm=
```
{0.997025, 0.998512, {0.257921, 0.00705258}}
```

Koeficient korelace vychází $r = 0{,}9985$. Nyní můžeme odpovědět např. na otázky, jaký odpor bude odpovídat teplotám $t_1 = 20.0\,°C$ a $t_2 = 60.0\,°C$ a při jaké teplotě t_3 bude mít vodič odpor $R_3 = 82.0\,\Omega$. Abychom s vypočtenými hodnotami vypsali na výstupu i označení proměnných, nadefinujeme ještě dříve funkci **nameAndValue**. V případě teploty t_3 pak vidíme, jak z výstupu funkce **Solve** použít pouze číselnou hodnotu namísto přiřazení {x → cislo}:

In[59]:=
```
R1 = model["BestFit"] /. x → 20;
R2 = model["BestFit"] /. x → 60;
t3 = Solve[model["BestFit"] == 82.0, x][[1, 1, 2]];
nameAndValue[R1, 1]
nameAndValue[R2, 1]
nameAndValue[t3, 1]
```

```
R1=76,5

R2=88,1

t3=39,0
```

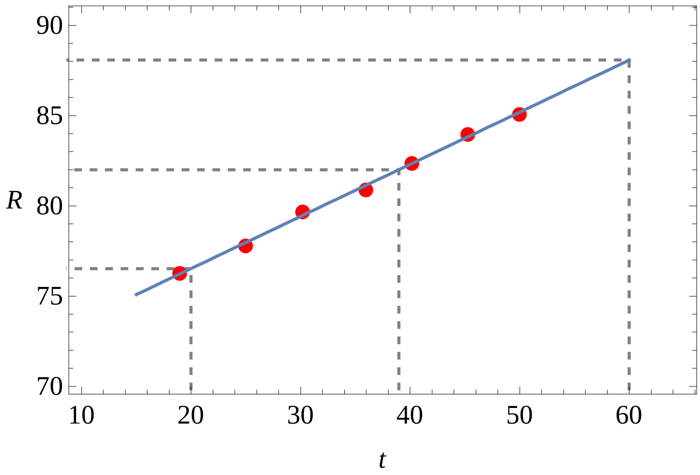
Nyní můžeme graf vykreslit i s vypočtenými hodnotami.

In[65]:=
```
grafvypoctenychhodnot =
  ListLinePlot[{{{20, 0}, {20, R1}, {0, R1}}, {{60, 0}, {60, R2}, {0, R2}},
    {{t3, 0}, {t3, 82}, {0, 82}}}, PlotStyle → {{Dashed, Gray}}];
```

In[66]:=
```
grafmerenychhodnot =
  ListPlot[hodnoty, PlotMarkers → {Automatic, Medium}, PlotStyle → {Red}];
grafmodelu = Plot[model["BestFit"], {x, 15, 60}, PlotStyle → {Thickness[0.005]}];
```

In[68]:=
```
Show[{grafvypoctenychhodnot, grafmerenychhodnot, grafmodelu},
 Frame → True, FrameLabel → {"t", "R"}, RotateLabel → False,
 PlotRange → {{10, 65}, {70, 90}}, BaseStyle → {FontFamily → "Times", FontSize → 14}]
```

Out[68]=



3.2 Určení tíhového zrychlení

Užitím naměřených hodnot dráhy s a času t pro volný pád olověné kuličky (pro kterou můžeme zanedbat odpor vzduchu)

$\frac{s}{m}$	0.00	4.00	5.20	6.00	7.00	8.00	9.00	10.0	11.0
$\frac{t}{s}$	0.00	0.90	1.03	1.10	1.20	1.28	1.35	1.43	1.50

určete tíhové zrychlení. Předpokládáme, že času $t = 0$ s odpovídá $s = 0$ m. Podobně jako v předchozím příkladu zadáme hodnoty:

In[69]:=
```
draha = {0.00, 4.00, 5.20, 6.00, 7.00, 8.00, 9.00, 10.0, 11.0}
```

Out[69]=
```
{0., 4., 5.2, 6., 7., 8., 9., 10., 11.}
```

In[70]:=
```
cas = {0.00, 0.90, 1.03, 1.10, 1.20, 1.28, 1.35, 1.43, 1.50}
```

Out[70]=
```
{0., 0.9, 1.03, 1.1, 1.2, 1.28, 1.35, 1.43, 1.5}
```

In[71]:=
```
hodnoty = Partition[Riffle[cas, draha], 2]
```

Out[71]=
```
{{0., 0.}, {0.9, 4.}, {1.03, 5.2}, {1.1, 6.},
 {1.2, 7.}, {1.28, 8.}, {1.35, 9.}, {1.43, 10.}, {1.5, 11.}}
```

Při analýze volného pádu vycházíme ze známého vztahu $s = \frac{1}{2} g t^2$. Chceme-li použít lineární regresi, musíme hledat vztah mezi $2s$ a t^2:

In[72]:=
```
hodnoty2 = Partition[Riffle[cas^2, 2 draha], 2]
```

Out[72]=
```
{{0., 0.}, {0.81, 8.}, {1.0609, 10.4}, {1.21, 12.},
 {1.44, 14.}, {1.6384, 16.}, {1.8225, 18.}, {2.0449, 20.}, {2.25, 22.}}
```

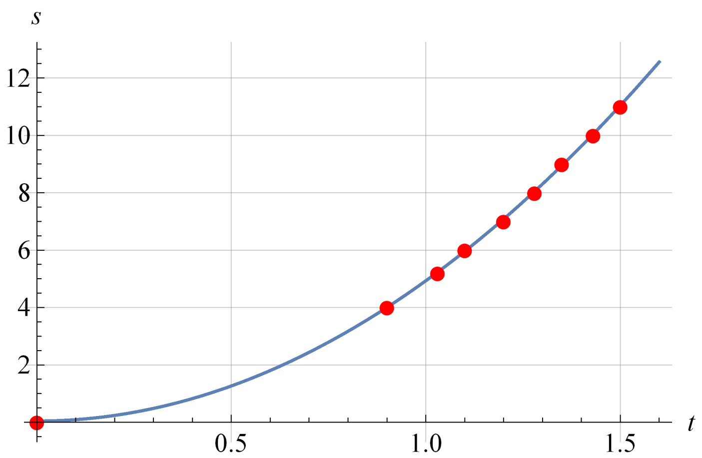
In[73]:=
```
model = LinearModelFit[hodnoty2, y, y];
model["BestFit"]
```

Out[74]= 0.0414801 + 9.77679 y

Získaný model můžeme pro ilustraci opět vykreslit s naměřenými hodnotami a určit jeho charakteristiky:

In[75]:=
```
Style[
 Show[{Plot[model["BestFit"] /. y → t^2 / 2, {t, 0, 1.6}, PlotStyle → {Thickness[0.005]},
    AxesLabel → {"t", "s"}, BaseStyle → {FontFamily → "Times", FontSize → 14},
    GridLines → Automatic, Axes → True],
   ListPlot[hodnoty, PlotMarkers → {Automatic, Medium}, PlotStyle → {Red}]}]]
NumberForm[{model["RSquared"], Sqrt[model["RSquared"]],
  model["BestFitParameters"][[2]], model["ParameterErrors"][[2]]}, NumberPoint → "."]
```

Out[75]=



Out[76]//NumberForm=
{0.999835, 0.999917, 9.77679, 0.0474963}

Koeficient korelace vychází $r = 0.9999$ a pro hledanou hodnotu tíhového zrychlení vychází $g = (9.78 \pm 0.05)\,\mathrm{m\cdot s^{-2}}$. Program *Mathematica* umožňuje najít přímo koeficient hledané kvadratické závislosti.

In[77]:=
```
nlinmodel = NonlinearModelFit[hodnoty, g1 / 2 x^2, g1, x];
NumberForm[{nlinmodel["BestFit"], nlinmodel["ParameterTable"],
  nlinmodel["RSquared"], Sqrt[nlinmodel["RSquared"]]}, NumberPoint → "."]
```

Out[78]//NumberForm=

{4.90077 x^2,

	Estimate	Standard Error	t-Statistic	P-Value
g1	9.80154	0.0196049	499.953	2.869×10^{-19}

, 0.999968, 0.999984}

Použitím tohoto modelu vychází $g = (9.80 \pm 0.02)\,\mathrm{m\cdot s^{-2}}$ s koeficientem korelace $r = 0.99994$ velmi blízkým jedné, tento způsob je v tomto zřejmě rychlejší a přesnější. Funkci **NonlinearModelFit** můžeme samozřejmě použít i na upravené hodnoty výše hledané závislosti $2s$ na t^2:

In[79]:=
```
nlinmodel2 = NonlinearModelFit[hodnoty2, g2 y, g2, y];
NumberForm[{nlinmodel2["BestFit"], nlinmodel2["ParameterTable"],
  nlinmodel2["RSquared"], Sqrt[nlinmodel2["RSquared"]]}, NumberPoint → "."]
```

Out[80]//NumberForm=

{9.80154 y,

	Estimate	Standard Error	t-Statistic	P-Value
g2	9.80154	0.0196049	499.953	2.869×10^{-19}

, 0.999968, 0.999984}

Pokud nás nezajímají statistické charakteristiky modelu, můžeme použít i funkce **Fit** (i pak musíme zadat předpokládaný tvar polynomické závislosti):

In[81]:=
```
Fit[hodnoty, {x^2}, x]
```

Out[81]= 4.90077 x^2

3.3 Anomálie vody

Hodně možností pro hledání závislostí nabízejí i běžné středoškolské fyzikální tabulky. Podívejme se na závislost hustoty vody na teplotě v rozmezí mezi 0 °C a 40 °C. Zadejme teploty po 5 °C a jim odpovídající hodnoty hustoty.

```
In[82]:= teplota = Table[i, {i, 0, 40, 5}]
Out[82]= {0, 5, 10, 15, 20, 25, 30, 35, 40}
```

```
In[83]:= hustota = {999.8426, 999.9668, 999.7201,
           999.1016, 998.2063, 997.048, 995.6511, 994.0359, 992.2204}
Out[83]= {999.843, 999.967, 999.72, 999.102, 998.206, 997.048, 995.651, 994.036, 992.22}
```

```
In[84]:= hodnoty = Partition[Riffle[teplota, hustota], 2]
Out[84]= {{0, 999.843}, {5, 999.967}, {10, 999.72}, {15, 999.102},
          {20, 998.206}, {25, 997.048}, {30, 995.651}, {35, 994.036}, {40, 992.22}}
```

Závislost hustoty na teplotě v uvažovaném intervalu aproximujme kubickým polynomem.

```
In[85]:= nlinmodel = NonlinearModelFit[hodnoty, ρ0 (1 + β1 t + β2 t^2 + β3 t^3), {ρ0, β1, β2, β3}, t]
         NumberForm[{Sqrt[nlinmodel["RSquared"]],
           nlinmodel["BestFitParameters"], nlinmodel["ParameterErrors"]}, NumberPoint → "."]
```

Out[85]= FittedModel[$999.85\,(1 + 0.0000607236\,t - 7.94845\times10^{-6}\,t^2 + 4.16554\times10^{-8}\,t^3)$]

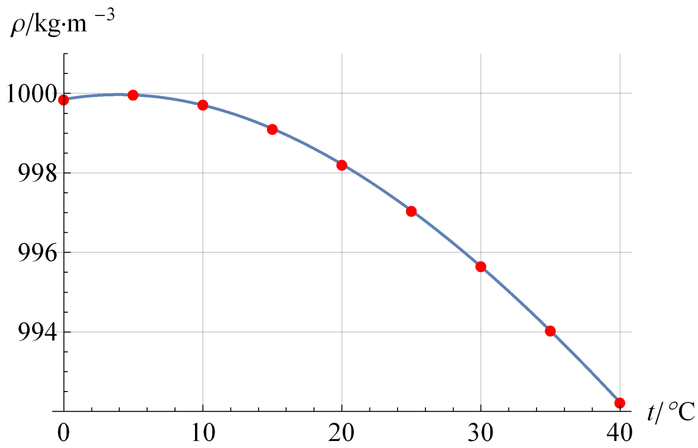
Out[86]//NumberForm=

$\{1.,\ \{\rho_0 \to 999.85,\ \beta_1 \to 0.0000607236,\ \beta_2 \to -7.94845\times10^{-6},\ \beta_3 \to 4.16554\times10^{-8}\},$
$\{0.0123617,\ 2.85443\times10^{-6},\ 1.72388\times10^{-7},\ 2.82737\times10^{-9}\}\}$

Nyní můžeme graficky znázornit proložení modelu daty:

```
In[87]:= Show[{Plot[nlinmodel["BestFit"] /. x → t, {t, 0, 40}, PlotStyle → {Thickness[0.005]},
             AxesLabel → {"t/ °C", "ρ/kg·m -3"}, BaseStyle → {FontFamily → "Times", FontSize → 14},
             GridLines → Automatic, Axes → True, PlotRange → {Automatic, {992, 1001}}],
           ListPlot[hodnoty, PlotMarkers → {Automatic, Small}, PlotStyle → {Red}]}]
```

Out[87]=



Pomocí získané kubické funkce se můžeme pokusit nalézt i teplotu, při které je hustota vody maximální:

In[88]:=
```
FindMaximum[nlinmodel["BestFit"], {t, 0}]
maxrho = FindMaximum[nlinmodel["BestFit"], {t, 0}][[1]] // N
maxt = t /. Last[FindMaximum[nlinmodel["BestFit"], {t, 0}]]
```

Out[88]=
```
{999.968, {t → 3.94199}}
```

Out[89]=
```
999.968
```

Out[90]=
```
3.94199
```

Získané hodnoty $t_m = 3.94\,°\text{C}$ poměrně dobře odpovídají, známe skutečnosti, že hustota vody nabývá maxima pro teploty okolo 4 °C. V našem modelu má tato maximální hustota hodnotu $\rho_m = 999.968\,\text{kg}\cdot\text{m}^{-3}$. Pro názornost můžeme vykreslit náš model pro teploty v okolí maxima a vyznačit maximální hodnotu.

In[91]:=
```
kreslimodel = Plot[nlinmodel["BestFit"] /. x → t, {t, 0, 10},
    PlotStyle → {Thickness[0.005]}, BaseStyle → {FontFamily → "Times", FontSize → 14},
    Frame → True, FrameLabel → {"t/°C", "ρ/kg·m -3"},
    RotateLabel → False, PlotRange → {Automatic, {999.7, 1000.05}}];
```

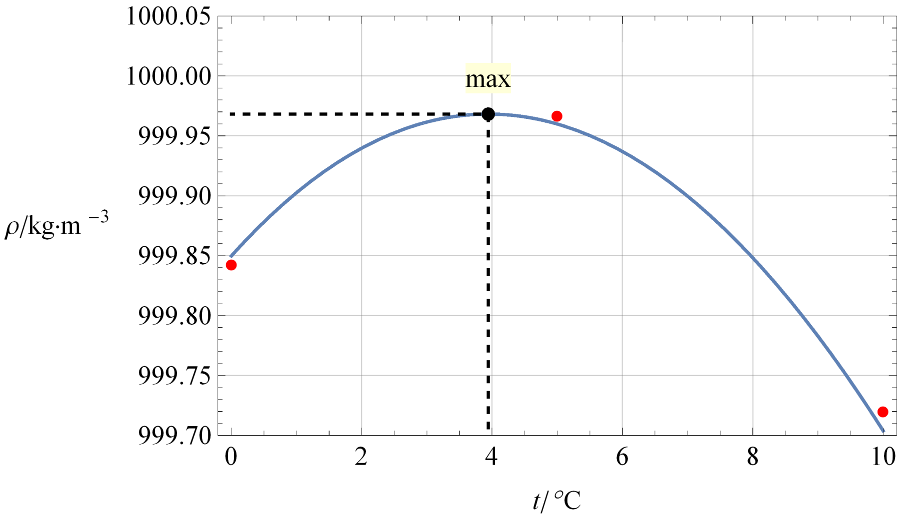
In[92]:=
```
kreslihodnoty = ListPlot[hodnoty, PlotMarkers → {Automatic, Small}, PlotStyle → {Red}];
```

In[93]:=
```
kreslimaxcary = ListLinePlot[
    {{{maxt, 997}, {maxt, maxrho}, {0, maxrho}}}, PlotStyle → {{Dashed, Black}}];
```

In[94]:=
```
kreslimaxbod = Graphics[{PointSize[.02], Point[{maxt, maxrho}]}];
```

In[95]:=
```
Style[Show[{kreslimodel, kreslihodnoty, kreslimaxcary, kreslimaxbod},
   GridLines → Automatic, Epilog →
    {Text["max", {maxt, maxrho + 0.03}, Background → LightYellow]}], NumberPoint → "."]
```

Out[95]=



Pokud bychom chtěli odhadnout parametry lineární teplotní objemové roztažnosti užívané ve středoškolské fyzice a v základním vysokoškolském kurzu, budeme předpokládat lineání závislost hustoty na teplotě:

In[96]:=
```
roztaznost = NonlinearModelFit[hodnoty, ρ0 (1 - β t), {ρ0, β}, t]
NumberForm[{Sqrt[roztaznost["RSquared"]], roztaznost["BestFitParameters"],
   roztaznost["ParameterErrors"]}, NumberPoint → "."]
```

Out[96]=
```
FittedModel[ 1001.21 (1 – 0.000194675 t) ]
```

Out[97]//NumberForm=
```
{1., {ρ0 → 1001.21, β → 0.000194675}, {0.557241, 0.0000232896}}
```

Pro součinitel teplotní objemové roztažnosti tak dostáváme hodnotu $\beta = (0.00019 \pm 0.00002)\,\text{K}^{-1}$, ve slušném souladu s tabulkovou hodnotou pro 20 °C, tj. $\beta_{20} = 0.00018\,\text{K}^{-1}$. Dodejme, že náš soubor dat byl pro jednoduchost zvolen poměrně malý.

3.4 Tlak sytých vodních par

Využijeme data o závislosti tlaku sytých vodních par na teplotě z tabulek.

Teplota ve °C:

```
In[98]:= teplota = Join[Table[i, {i, 0, 80, 5}], Table[i, {i, 81, 110}]]

Out[98]= {0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75,
  80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96,
  97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110}
```

Tlak v kPa:

```
In[99]:= tlak = {0.611, 0.873, 1.228, 1.706, 2.339, 3.169, 4.245, 5.627, 7.381, 9.590, 12.344,
   15.752, 19.932, 25.022, 31.176, 38.6, 47.4, 49.3, 51.3, 53.4, 55.6, 57.8,
   60.1, 62.5, 64.9, 67.5, 70.1, 72.8, 75.6, 78.5, 81.5, 84.5, 87.7, 90.9, 94.3,
   97.8, 101.3, 105.0, 108.8, 112.7, 116.7, 120.8, 125.0, 129.4, 133.9, 138.5, 143}

Out[99]= {0.611, 0.873, 1.228, 1.706, 2.339, 3.169, 4.245, 5.627, 7.381, 9.59, 12.344,
  15.752, 19.932, 25.022, 31.176, 38.6, 47.4, 49.3, 51.3, 53.4, 55.6, 57.8,
  60.1, 62.5, 64.9, 67.5, 70.1, 72.8, 75.6, 78.5, 81.5, 84.5, 87.7, 90.9, 94.3,
  97.8, 101.3, 105., 108.8, 112.7, 116.7, 120.8, 125., 129.4, 133.9, 138.5, 143}
```

Uspořádání do dvojic:

```
In[100]:= hodnoty = Partition[Riffle[teplota, tlak], 2]

Out[100]= {{0, 0.611}, {5, 0.873}, {10, 1.228}, {15, 1.706}, {20, 2.339}, {25, 3.169},
  {30, 4.245}, {35, 5.627}, {40, 7.381}, {45, 9.59}, {50, 12.344}, {55, 15.752},
  {60, 19.932}, {65, 25.022}, {70, 31.176}, {75, 38.6}, {80, 47.4}, {81, 49.3},
  {82, 51.3}, {83, 53.4}, {84, 55.6}, {85, 57.8}, {86, 60.1}, {87, 62.5},
  {88, 64.9}, {89, 67.5}, {90, 70.1}, {91, 72.8}, {92, 75.6}, {93, 78.5},
  {94, 81.5}, {95, 84.5}, {96, 87.7}, {97, 90.9}, {98, 94.3}, {99, 97.8},
  {100, 101.3}, {101, 105.}, {102, 108.8}, {103, 112.7}, {104, 116.7},
  {105, 120.8}, {106, 125.}, {107, 129.4}, {108, 133.9}, {109, 138.5}, {110, 143}}
```

```
In[101]:= nlinmodel = NonlinearModelFit[hodnoty, a + b t + c t^2 + d t^3, {a, b, c, d}, t]
NumberForm[{Sqrt[nlinmodel["RSquared"]],
  nlinmodel["BestFitParameters"], nlinmodel["ParameterErrors"]}, NumberPoint → "."]
```

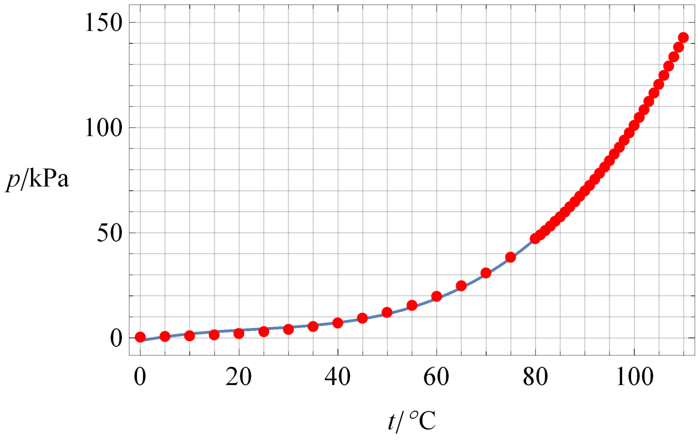
Out[101]= FittedModel[$-1.33733 + 0.437518\,t - 0.0132457\,t^2 + 0.000191825\,t^3$]

Out[102]//NumberForm=

{0.999954, {a → −1.33733, b → 0.437518, c → −0.0132457, d → 0.000191825},
 {0.529562, 0.0390413, 0.00076192, 4.26568×10^{-6}}}

In[103]:=
```
Show[{Plot[nlinmodel["BestFit"], {t, 0, 100},
   PlotStyle → {Thickness[0.005]}, FrameLabel → {"t/ °C", "p/kPa"},
   RotateLabel → False, BaseStyle → {FontFamily → "Times", FontSize → 14},
   GridLines → {Table[i, {i, 0, 110, 5}], Table[i, {i, 0, 150, 10}]}, Frame → True],
  ListPlot[hodnoty, PlotMarkers → {Automatic, Small}, PlotStyle → {Red}]},
 PlotRange → {{0, 110}, {0, 150}}]
```

Out[103]=



V některých zdrojích (např. v anglické části Wikipedie) je zmiňována empirická Antoinova rovnice, která předpokládá exponenciální závislost. I takový model se můžeme pokusit nalézt, předpokládáme-li závislost $p = A \cdot \exp(B/t)$. Lomený výraz nás nutí vynechat $t = 0$ °C z tabulky hodnot pomocí příkazu `Drop[hodnoty,1]`, který vypustí ze seznamu hodnot první uspořádanou dvojici. Pro model tak získáváme:

In[104]:=
```
nlinmodel2 = NonlinearModelFit[Drop[hodnoty, 1], A Exp[B / t], {A, B}, t]
NumberForm[{Sqrt[nlinmodel2["RSquared"]], nlinmodel2["BestFitParameters"],
  nlinmodel2["ParameterErrors"]}, NumberPoint → "."]
```

Out[104]= FittedModel[$2525.22\, e^{-319.882/t}$]

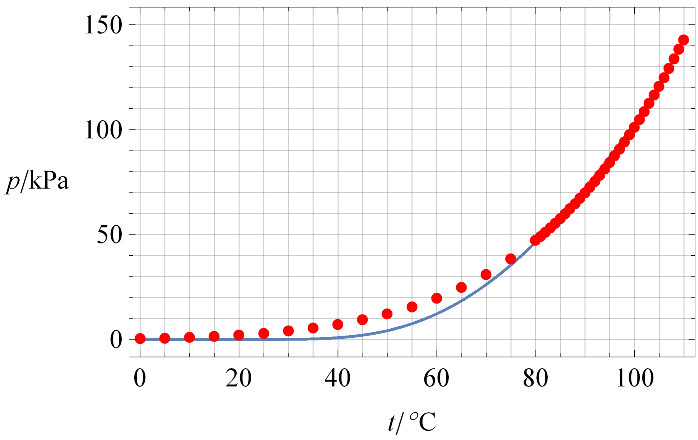
Out[105]//NumberForm=

```
{0.998936, {A → 2525.22, B → -319.882}, {189.667, 7.40947}}
```

Vidíme, že koeficient korelace je o něco menší než u modelu pomocí kubického polynomu, rozptyl hodnot koeficientů naopak větší. Názorně to ukáže i vykreslení modelu spolu s tabulkovými hodnotami.

In[106]:=
```
Show[{Plot[nlinmodel2["BestFit"], {t, 0, 100},
   PlotStyle → {Thickness[0.005]}, FrameLabel → {"t/ °C", "p/kPa"},
   RotateLabel → False, BaseStyle → {FontFamily → "Times", FontSize → 14},
   GridLines → {Table[i, {i, 0, 110, 5}], Table[i, {i, 0, 150, 10}]}, Frame → True],
  ListPlot[hodnoty, PlotMarkers → {Automatic, Small}, PlotStyle → {Red}]},
 PlotRange → {{0, 110}, {0, 150}}]
```

Out[106]=



Vidíme, že v okolí 100 °C by bylo možné závislost aproximovat lineární funkcí a nalézt tak závislost teploty varu vody na tlaku (lze říci, že k varu dochází přibližně tehdy, když vnější atmosférický tlak je roven tlaku sytých par). Vyberme proto hodnoty od 90 °C do 108 °C tak, že vezmeme posledních 20 uspořádaných dvojic a potom vypustíme ještě 2 poslední. Na tuto podmožinu pak aplikujeme opět lineární regresi.

In[107]:=
```
hodnoty2 = Drop[Take[hodnoty, -20], -2]
model = LinearModelFit[hodnoty2, t, t];
NumberForm[{Sqrt[model["RSquared"]],
  model["BestFitParameters"], model["ParameterErrors"]}, NumberPoint → "."]
```

Out[107]=
```
{{91, 72.8}, {92, 75.6}, {93, 78.5}, {94, 81.5}, {95, 84.5}, {96, 87.7},
 {97, 90.9}, {98, 94.3}, {99, 97.8}, {100, 101.3}, {101, 105.}, {102, 108.8},
 {103, 112.7}, {104, 116.7}, {105, 120.8}, {106, 125.}, {107, 129.4}, {108, 133.9}}
```

Out[109]//NumberForm=
```
{0.997693, {-255.602, 3.58349}, {6.07428, 0.0609653}}
```

In[110]:=
```
model["BestFit"]
```

Out[110]=
```
-255.602 + 3.58349 t
```

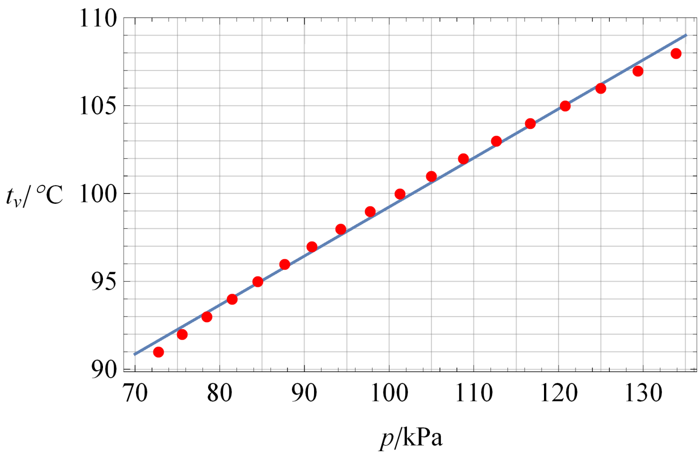
Koeficient korelace $r = 0.997693$ nalezené závislosti je opět celkem uspokojivý. Nyní nalezněme “obrácenou” závislost teploty (varu vody) na tlaku, kdy použijeme rovnici modelu k sestavení lineární rovnice. Pro vykreslení musíme přeuspořádat data (přehodit nezávisle a závisle proměnnou).

In[111]:=
```
tv = Expand[Solve[model["BestFit"] == p, t][[1, 1, 2]]]
prevracenehodnoty =
 Partition[Riffle[Transpose[hodnoty2][[2]], Transpose[hodnoty2][[1]]], 2]
Show[{Plot[tv, {p, 70, 135}, PlotStyle → {Thickness[0.005]},
   FrameLabel → {"p/kPa", "tv/ °C"}, RotateLabel → False,
   BaseStyle → {FontFamily → "Times", FontSize → 14},
   GridLines → {Table[i, {i, 70, 140, 5}], Table[i, {i, 90, 110, 1}]}, Frame → True],
  ListPlot[prevracenehodnoty, PlotMarkers → {Automatic, Small}, PlotStyle → {Red}]}]
```

Out[111]=
```
71.3276 + 0.279058 p
```

Out[112]=
```
{{72.8, 91}, {75.6, 92}, {78.5, 93}, {81.5, 94}, {84.5, 95}, {87.7, 96},
 {90.9, 97}, {94.3, 98}, {97.8, 99}, {101.3, 100}, {105., 101}, {108.8, 102},
 {112.7, 103}, {116.7, 104}, {120.8, 105}, {125., 106}, {129.4, 107}, {133.9, 108}}
```

Out[113]=



Získaný vztah se jen málo liší od rovnice uvedené v tabulkách $t_v = 71.6 + 0.28\,p$.

4 Měření tíhového zrychlení reverzním a matematickým kyvadlem

4.1 Tíhové zrychlení

Tíhové zrychlení $\boldsymbol{g}$ je zrychlení volně puštěného tělesa v gravitačním poli Země působením tíhové síly F_G. Tíhová síla F_G je vektorovým součtem gravitační síly F_g a setrvačné odstředivé síly F_S. Platí tedy:

$$\boldsymbol{F}_G = \boldsymbol{F}_g + \boldsymbol{F}_s.$$

Podle druhého Newtonova gravitačního zákona udílí gravitační síla $\boldsymbol{F}_g$ tělesu o hmotnosti m gravitační zrychlení $\boldsymbol{a}_g = \frac{\boldsymbol{F}_g}{m}$. Pro velikost gravitačního zrychlení $\boldsymbol{a}_g$, které uděluje Země každému tělesu ve výšce h nad povrchem Země, resp. na povrchu Země, platí fyzikální vztahy:

$$\boldsymbol{a}_{\text{gh}} = \frac{\kappa M_Z}{(R_Z + h)^2},$$

$$\boldsymbol{a}_{\text{g0}} = \frac{\kappa M_Z}{R_Z{}^2}.$$

K získání konstant ve vztazích lze využít WolframAlpha.

Hodnota gravitační konstanty κ je:

In[114]:=
```
WolframAlpha["Newtonian gravitational constant", IncludePods → "Value",
 AppearanceElements → {"Pods"}, TimeConstraint → {30, Automatic, Automatic, Automatic}]
```

Out[114]=

Value:

6.67×10^{-11} N m²/kg² (newton square meters per kilogram squared)

6.67×10^{-8} dyne cm²/g² (dyne square centimeters per gram squared)

3.44×10^{-8} ft³/(slug s²) (feet cubed per slug per second squared)

Hmotnost Země M_Z je:

In[115]:=
```
WolframAlpha["mass of Earth", IncludePods → "Result",
 AppearanceElements → {"Pods"}, TimeConstraint → {30, Automatic, Automatic, Automatic}]
```

Out[115]=

Result:

5.9721986×10^{24} kg (kilograms)

Poloměr Země R_Z je:

In[116]:=
```
WolframAlpha["average equatorial radius of earth", IncludePods → "Result",
 AppearanceElements → {"Pods"}, TimeConstraint → {30, Automatic, Automatic, Automatic}]
```

Out[116]=

Result: Show non–metric

6378.137 km (kilometers)

Velikost gravitačního zrychlení $\boldsymbol{a}_g$ se s rostoucí vzdáleností od povrchu Země zmenšuje, největší hodnotu má na povrchu Země:

In[117]:=
$$\texttt{Solve}\left[\texttt{a0} == \frac{6.67\times 10^{-11}\times 5.972\times 10^{24}}{\left(6378\times 10^{3}\right)^{2}}, \texttt{a0}\right]$$

Out[117]= {{a0 → 9.79212}}

Velikost gravitačního zrychlení, které uděluje Země tělesu, které je např. ve vzdálenosti 800 m od povrchu Země je:

In[118]:=
$$\texttt{Solve}\left[\texttt{a800} == \frac{6.67\times 10^{-11}\times 5.972\times 10^{24}}{\left(6378\times 10^{3}+800\right)^{2}}, \texttt{a800}\right]$$

Out[118]= {{a800 → 9.78966}}

Protože se velikost setrvačné odstředivé síly $\boldsymbol{F}_s$ mění se zeměpisnou šířkou místa na zemském povrchu, mění se se zeměpisnou šířkou místa také velikost tíhové síly $\boldsymbol{F}_G$. Se změnami tíhové síly $\boldsymbol{F}_G$ se mění i velikost tíhového zrychlení $\boldsymbol{g}$. Na rovníku je vliv setrvačné odstředivé síly největší, tudíž hodnota tíhového zrychlení je nejmenší. Dále se pak tíhové zrychlení zvětšuje směrem od rovníku k pólům.

Pokud chceme zjistit velikost tíhového zrychlení např. v Olomouci, využijeme WolframAlpha.

In[119]:=
```
WolframAlpha["acceleration of gravity in Olomouc",
 IncludePods → "GravitationalFieldStrength", AppearanceElements → {"Pods"},
 TimeConstraint → {30, Automatic, Automatic, Automatic}]
```

Out[119]=

Gravitational field strength
for Olomouc, Czech Republic:

Show non-metric units

total field	9.81323 m/s^2 (meters per second squared)
angular deviation from local vertical	0.00332° (degrees)
down component	9.81317 m/s^2 (meters per second squared)
west component	4.3×10^{-4} m/s^2 (meters per second squared)
south component	0.03255 m/s^2 (meters per second squared)

(based on EGM2008 12th order model; 213 meters above sea level)

In[120]:=
```
WolframAlpha["acceleration of gravity in Oslo",
 IncludePods → "GravitationalFieldStrength", AppearanceElements → {"Pods"},
 TimeConstraint → {30, Automatic, Automatic, Automatic}]
```

Out[120]=

Gravitational field strength for Oslo, Norway:

Show non-metric units

total field	9.825 m/s^2 (meters per second squared)
angular deviation from local vertical	0.00292° (degrees)
down component	9.82496 m/s^2 (meters per second squared)
west component	5×10^{-4} m/s^2 (meters per second squared)
south component	0.02867 m/s^2 (meters per second squared)

(based on EGM2008 12th order model; 4 meters above sea level)

In[121]:=
```
WolframAlpha["acceleration of gravity in Nairobi",
 IncludePods → "GravitationalFieldStrength", AppearanceElements → {"Pods"},
 TimeConstraint → {30, Automatic, Automatic, Automatic}]
```

Out[121]=

Gravitational field strength for Nairobi: Show non–metric units

total field	$9.77087\ \mathrm{m/s^2}$ (meters per second squared)
angular deviation from local vertical	$1.5\times10^{-4\circ}$ (degrees)
down component	$9.77087\ \mathrm{m/s^2}$ (meters per second squared)
east component	$9\times10^{-5}\ \mathrm{m/s^2}$ (meters per second squared)
north component	$0.00147\ \mathrm{m/s^2}$ (meters per second squared)

(based on EGM2008 12th order model; 1670 meters above sea level)

Z WolframAlpha vidíme, že velikost tíhového zrychlení v Olomouci je 9.813 $\frac{\mathrm{m}}{\mathrm{s}^2}$, např. v Oslu je velikost tíhového zrychlení 9.825 $\frac{\mathrm{m}}{\mathrm{s}^2}$ a např. v Nairobi je 9.771 $\frac{\mathrm{m}}{\mathrm{s}^2}$.

Pro výpočet tíhového zrychlení $\boldsymbol{g}$, lze např. také použít vztah:

$$\boldsymbol{g} = 9,\ 78\,031\,846\left(1 + 0,\ 005\,278\sin^2\phi + 0,\ 000\,023\,462\sin^4\phi\right)\frac{\mathrm{m}}{\mathrm{s}^2},$$

který je součástí Geodetic Reference System a byl přijat v roce 1967 International Union of Geodesy and Geophysics. ϕ ve vztahu je zeměpisná šířka.

Zeměpisnou šířku Olomouce lze opět zjistit pomocí WolframAlpha.

In[122]:=
```
WolframAlpha["latitude Olomouc", IncludePods → "Result",
 AppearanceElements → {"Pods"}, TimeConstraint → {30, Automatic, Automatic, Automatic}]
```

Out[122]=

Result: Show DMS

49.61°N

Zeměpisnou šířku převedeme ze stupňů na radiány a dosadíme do vztahu pro výpočet velikosti tíhového zrychlení.

In[123]:=
```
NSolve[radiany == 49.61 Pi/180, radiany]
```

Out[123]=
```
{{radiany → 0.865858}}
```

In[124]:=
```
Solve[gOL == 9.78031846
   (1 + 0.005278 (Sin[0.865857841914387])^2 + 0.000023462 (Sin[0.865857841914387])^4), gOL]
```

Out[124]=
```
{{gOL → 9.81034}}
```

Velikost tíhového zrychlení v Olomouci je 9.810 $\frac{\mathrm{m}}{\mathrm{s}^2}$. Tyto dva početní výsledky pro velikost tíhového zrychlení pro Olomouc jsou pro první tři platné číslice stejné a liší se až na pozici čtvrté platné číslice.

Dohodou byla stanovena hodnota tzv. normálního tíhového zrychlení $\boldsymbol{g}_n$, která odpovídá hodnotě tíhového zrychlení pro 45° severní zeměpisné šířky při hladině moře.

In[125]:=
```
NSolve[gnrad == 45 Pi/180, gnrad]
```

Out[125]=
```
{{gnrad → 0.785398}}
```

In[126]:=
```
Solve[gn == 9.78031846
    (1 + 0.005278 (Sin[0.7853981633974483])^2 + 0.000023462 (Sin[0.7853981633974483])^4), gn]
```
Out[126]=
```
{{gn → 9.80619}}
```

Hodnota tíhového zrychlení ***g*** závisí také na nadmořské výšce, s rostoucí nadmořskou výškou se tíhové zrychlení zmenšuje. Můžeme ovšem dokázat, že v malých nadmořských výškách lze tíhové zrychlení považovat za konstantní. Např. ve výšce asi 3 km nad zemským povrchem klesne ***g*** pouze o 1 ‰.

4.2 Matematické kyvadlo, fyzické kyvadlo a reverzní kyvadlo

Matematické kyvadlo

Matematickým kyvadlem rozumíme abstraktní model mechanického oscilátoru, kde je malé těleso hmotnosti m zavěšeno na pevném vlákně zanedbatelné hmotnosti a konstantní délky l.
Při výpočtu se omezíme pouze na malé výchylky, abychom mohli oblouk, po kterém se kulička pohybuje, považovat za úsečku. Pro výchylku $\alpha \leq 5°$ platí, že výraz sinα je přibližně roven úhlu α vyjádřenému v radiánech (sin$\alpha \approx \alpha$).

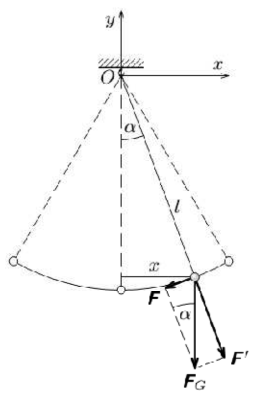
Např. pro $\alpha = 5°$ platí:

In[127]:=
```
NSolve[A5a == 5 Pi/180, A5a]
```
Out[127]=
```
{{A5a → 0.0872665}}
```

In[128]:=
```
NSolve[A5b == Sin[5 Pi/180], A5b]
```
Out[128]=
```
{{A5b → 0.0871557}}
```

Příčinou kmitavého pohybu matematického kyvadla je síla ***F***, která je výslednicí tíhové síly $m\boldsymbol{g}$ a tahové síly $\boldsymbol{F}'$, kterou působí vlákno závěsu na těleso. Síla ***F*** působí ve směru proti výchylce kuličky (koncového bodu závěsu kyvadla) a snaží se ji vrátit do rovnovážné polohy.
Hodnotu periody kmitu matematického kyvadla o délce L umístěného v tíhovém poli Země o tíhovém zrychlení ***g*** s počáteční výchylkou menší než 5°, můžeme vypočítat podle následujícího vztahu: $T_0 = 2\pi\sqrt{\frac{L}{g}}$.

Ze vztahu vidíme, že perioda kmitání matematického kyvadla nezávisí na hmotnosti tělesa ani na výchylce z rovnovážné polohy. Pro tíhové zrychlení ***g*** platí tedy vztah: $\boldsymbol{g} = \frac{4\pi^2 L}{T^2}$.



Model matematického kyvadla.

Ačkoliv se ve většině (středoškolských) učebnic zdůrazňuje, že výchylka matematického kyvadla nesmí překročit 5°, aby platil vztah $T_0 = 2\pi\sqrt{\frac{L}{g}}$, lze při experimentech matematické kyvadlo vychýlit o více než 5°. Vztah pro periodu kmitů matematického kyvadla pro výchylky větší než 5° je:

$$T = 2\pi\sqrt{\frac{L}{g}}\left(1 + \frac{1}{4}\sin^2\frac{\alpha}{2} + \frac{9}{64}\sin^4\frac{\alpha}{2} + \frac{225}{2304}\sin^6\frac{\alpha}{2} + ...\right)$$

Pro výkmit např. 10° je podle tohoto vztahu perioda matematického kyvadla T:

In[129]:=
```
NSolve[A10 == 10 Pi/180, A10]
```

Out[129]=
```
{{A10 → 0.174533}}
```

In[130]:=
```
NSolve[
 T == T0 (1 + 1/4 (Sin[0.174533/2])^2 + 9/64 (Sin[0.174533/2])^4 + 225/2304 (Sin[0.174533/2])^6), T]
```

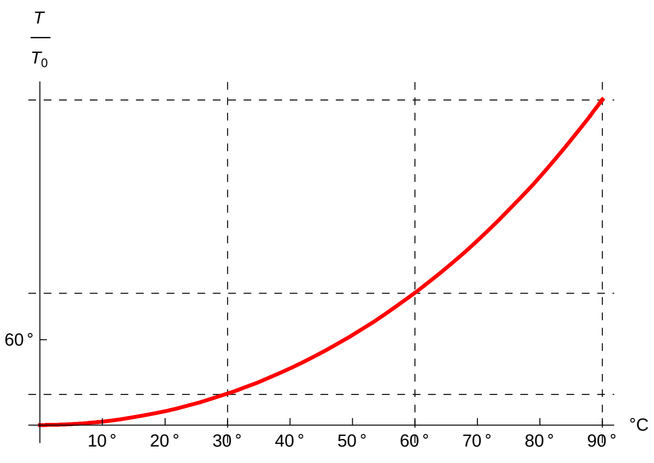
Out[130]=
```
{{T → 1.00191 T0}}
```

Vidíme, že pro periodu matematického kyvadla T, T_0 platí $T = 1,00191\ T_0$. Jestliže pro výchylku 10° použijeme klasický středoškolský vztah $T_0 = 2\pi\sqrt{\frac{L}{g}}$ místo vztahu $T = 2\pi\sqrt{\frac{L}{g}}\left(1 + \frac{1}{4}\sin^2\frac{\alpha}{2} + \frac{9}{64}\sin^4\frac{\alpha}{2} + \frac{225}{2304}\sin^6\frac{\alpha}{2} + ...\right)$ dopouštíme se chyby pouze 0,2 %.

Rozdíl mezi vypočtenou periodou matematického kyvadla pomocí těchto dvou vztahů pro počáteční výchylku v intervalu <0 °,90 °> je zobrazen na grafu níže.

In[131]:=
```
Plot[Sum[((2 n)!/(2^n n!)^2)^2 Sin[x/2]^(2 n), {n, 0, 100}], {x, 0 Degree, 90 Degree}, AxesOrigin → {0, 1},
 Ticks → {{10 °, 20 °, 30 °, 40 °, 50 °, 60 °, 70 °, 80 °, 90 °}}, AxesLabel → {"°C", T/T0},
 PlotStyle → {Red, Thick}, GridLines → {{30 °, 60 °, 90 °}, {1.017, 1.073, 1.18}},
 GridLinesStyle → {{Dashed}, {Dashed}}]
```

Out[131]=



Fyzické kyvadlo

Za reálné kyvadlo považujeme fyzické kyvadlo. Fyzické kyvadlo je libovolné tuhé těleso zavěšené nad svým těžištěm, které vykonává kmity kolem rovnovážné polohy jako matematické kyvadlo. Pro výpočet periody kyvu platí vztah:

$$T = 2\pi\sqrt{\frac{J}{m g a}},$$

kde m je hmotnost tělesa, J moment setrvačnosti tuhého tělesa vzhledem k těžišti a a je vzdálenost těžiště kyvadla od osy otáčení. Matematické kyvadlo je opravdu jen speciálním případem fyzického kyvadla, kdy kmitá hmotný bod o hmotnosti m ve vzdálenosti L od osy otáčení. Jeho moment setrvačnosti je $J = m L^2$.

$$T = 2\pi \sqrt{\frac{m L^2}{m g L}} = T = 2\pi \sqrt{\frac{L}{g}}.$$

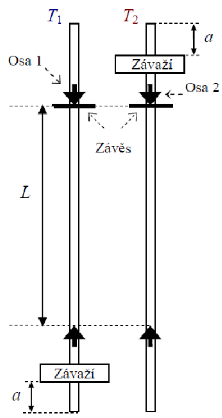
Reverzní kyvadlo

Henry Kater, britský fyzik, vynalezl reverzní kyvadlo pro měření lokálního tíhového zrychlení. Reverzní kyvadlo je založeno na definici redukované délky fyzického kyvadla. Redukovaná délky fyzického kyvadla je taková délka matematického kyvadla, kdy perioda kmitu matematického kyvadla, o redukované délce fyzického kyvadla, má stejnou periodu jako fyzické kyvadlo. Pro fyzické kyvadlo existují právě dvě různé vzdálenosti osy otáčení od těžiště (a_1, a_2), pro které platí, že perioda kmitání je stejná. Součtem těchto vzdáleností dostaneme právě redukovanou délku fyzického kyvadla.

Aby bylo reverzní kyvadlo použitelné na všech místech Zeměkoule, skládá se reverzní kyvadlo z ocelové tyče se dvěma pevnými hroty a další součástí reverzního kyvadla je posuvné závaží na jednom konci tyče. Posunem tohoto závaží, lze zkrátit, nebo prodloužit periody kmitání reverzního kyvadla. Cílem měření je, najít správnou polohu posuvného závaží, aby perioda kmitání kolem obou os otáčení byla stejná. Tímto nastavením se z fyzického kyvadla stává reverzní kyvadlo. Po zjištění periody reverzního kyvadla můžeme k výpočtu tíhového zrychlení použít vztah pro výpočet tíhového zrychlení matematického kyvadla vztah:

$$g = \frac{T^2 4\pi^2}{L},$$

kde T je perioda kmitání reverzního kyvadla, a L je redukovaná délka fyzického kyvadla, která je rovna vzdálenosti mezi osami otáčení.



Reverzní kyvadlo.

4.3 Postup měření tíhového zrychlení reverzním kyvadlem a zpracování výsledků

Postup měření:

1. Změřte 10 krát vzdálenost břitů na kyvadle (L). Nastavte závaží na nejkratší vzdálenosti od břitu. Změřte vzdálenost závaží od konce tyče (vzdálenost značíme a). Zavěste kyvadlo tak, že závaží je pod osou otáčení. Nastavte na čítači měření 1 periody.

2. Vychylte kyvadlo z rovnovážné polohy o méně než 5° a změřte periodu T_1 10 krát.

3. Nyní kyvadlo otočte a změřte periodu T_2 při zavěšení na druhém břitu. Změřte periodu opět 10 krát.

4. Posuňte závaží o cca 1 cm od břitu a opět změřte vzdálenost a a periody T_1 a T_2. Tento proces opakujte do doby až perioda T_2 bude menší než perioda T_1. Poté závaží posuňte ještě 2x.

5. Sestavte graf závislosti vzdálenosti závaží na periodách T_1 a T_2 a z grafu zjistěte vzdálenost a0, pro které budou periody T_1 a T_2 rovny.

6. Nastavte tuto vzdálenost na kyvadle a změřte opět 10 krát periody T_1 a T_2.

7. Vypočítejte tíhové zrychlení a chybu měření.

Výsledky měření:

1. Změřili jsme 10 krát délku mezi dvěma břity L.

```
In[132]:= delkaL = {1.146, 1.146, 1.145, 1.145, 1.145, 1.145, 1.146, 1.145, 1.145, 1.146}

Out[132]= {1.146, 1.146, 1.145, 1.145, 1.145, 1.145, 1.146, 1.145, 1.145, 1.146}
```

2. Nastavili jsme vzdálenost závaží a_1 a změřili 10 krát dobu kyvu T_1 pro polohu závaží u břitu a poté periodu kmitu T_2, kdy závaží je na vzdálenější straně od osy otáčení. Postupně jsme dále měřili periody kmitů kyvadla pro další polohy závaží.
Data z našeho měření jsou v souboru DATA, které jsme do programu *Mathematica* importovali z vytvořeného textového souboru.
Data jsou seřazena v následujících tabulkách:

```
In[133]:= a = {12.65, 11.885, 10.56, 9.10, 8.14, 6.96}

Out[133]= {12.65, 11.885, 10.56, 9.1, 8.14, 6.96}

In[134]:= SetDirectory[NotebookDirectory[]]

Out[134]= C:\Users\20022969\Desktop\2_Fyzika

In[135]:= DATA = Import["DATA.txt", "Data"]

Out[135]= {{2.12742, 2.13264, 2.13879, 2.15192, 2.15896,
   2.16754, 2.05674, 2.07752, 2.12325, 2.17541, 2.20518, 2.25955},
  {2.12746, 2.13267, 2.13941, 2.15225, 2.15899, 2.16755, 2.05674, 2.0776,
   2.12325, 2.17545, 2.20579, 2.25956}, {2.12757, 2.13267, 2.13943, 2.15225,
   2.15909, 2.16756, 2.05675, 2.07761, 2.12335, 2.17546, 2.20627, 2.25964},
  {2.12758, 2.13268, 2.13944, 2.15226, 2.15911, 2.16757, 2.05677, 2.07767,
   2.12336, 2.17551, 2.20664, 2.25966}, {2.12758, 2.1327, 2.13952, 2.15229,
   2.15912, 2.16759, 2.05677, 2.07783, 2.12336, 2.17551, 2.20724, 2.2597},
  {2.12759, 2.13274, 2.13953, 2.15231, 2.15912, 2.16759, 2.05678, 2.078,
   2.12339, 2.17553, 2.20744, 2.25976}, {2.12759, 2.13274, 2.1396, 2.15232,
   2.15914, 2.16761, 2.05678, 2.07802, 2.1234, 2.17553, 2.20776, 2.25978},
  {2.12759, 2.13274, 2.13961, 2.15236, 2.15918, 2.16773, 2.05681, 2.07804,
   2.12344, 2.17553, 2.20804, 2.25982}, {2.1276, 2.13275, 2.13965, 2.15238,
   2.1592, 2.16775, 2.05682, 2.07809, 2.1235, 2.17556, 2.20844, 2.25985},
  {2.12763, 2.13275, 2.13966, 2.1524, 2.15921, 2.16779, 2.05684, 2.07813,
   2.12351, 2.17558, 2.20865, 2.25989}, {2.12763, 2.13278, 2.13967, 2.15244,
   2.15921, 2.16782, 2.05684, 2.07824, 2.12352, 2.17559, 2.20915, 2.26002},
  {2.12764, 2.13278, 2.13971, 2.15246, 2.15926, 2.16784, 2.05685, 2.07839,
   2.12357, 2.1756, 2.20919, 2.26008}, {2.12766, 2.13279, 2.1398, 2.15248,
   2.15926, 2.16784, 2.05686, 2.0784, 2.12357, 2.17569, 2.20948, 2.26011},
  {2.12766, 2.1328, 2.13989, 2.15262, 2.15929, 2.16786, 2.05687, 2.07842,
   2.12358, 2.17572, 2.20972, 2.2602}, {2.12767, 2.13281, 2.14006, 2.15266,
   2.15931, 2.16796, 2.05688, 2.07852, 2.1236, 2.17575, 2.20989, 2.25716}}

In[136]:= Dimensions[DATA]

Out[136]= {15, 12}
```

In[137]:=
```
T1a1 = Table[DATA[[i, 1]], {i, 15}];
T1a2 = Table[DATA[[i, 2]], {i, 15}];
T1a3 = Table[DATA[[i, 3]], {i, 15}];
T1a4 = Table[DATA[[i, 4]], {i, 15}];
T1a5 = Table[DATA[[i, 5]], {i, 15}];
T1a6 = Table[DATA[[i, 6]], {i, 15}];
T2a1 = Table[DATA[[i, 7]], {i, 15}];
T2a2 = Table[DATA[[i, 8]], {i, 15}];
T2a3 = Table[DATA[[i, 9]], {i, 15}];
T2a4 = Table[DATA[[i, 10]], {i, 15}];
T2a5 = Table[DATA[[i, 11]], {i, 15}];
T2a6 = Table[DATA[[i, 12]], {i, 15}];
```

In[149]:=
```
T1 = {T1a1, T1a2, T1a3, T1a4, T1a5, T1a6}
```

Out[149]=
```
{{2.12742, 2.12746, 2.12757, 2.12758, 2.12758, 2.12759, 2.12759,
  2.12759, 2.1276, 2.12763, 2.12763, 2.12764, 2.12766, 2.12766, 2.12767},
 {2.13264, 2.13267, 2.13267, 2.13268, 2.1327, 2.13274, 2.13274, 2.13274,
  2.13275, 2.13275, 2.13278, 2.13278, 2.13279, 2.1328, 2.13281},
 {2.13879, 2.13941, 2.13943, 2.13944, 2.13952, 2.13953, 2.1396, 2.13961,
  2.13965, 2.13966, 2.13967, 2.13971, 2.1398, 2.13989, 2.14006},
 {2.15192, 2.15225, 2.15225, 2.15226, 2.15229, 2.15231, 2.15232, 2.15236,
  2.15238, 2.1524, 2.15244, 2.15246, 2.15248, 2.15262, 2.15266},
 {2.15896, 2.15899, 2.15909, 2.15911, 2.15912, 2.15912, 2.15914, 2.15918,
  2.1592, 2.15921, 2.15921, 2.15926, 2.15926, 2.15929, 2.15931},
 {2.16754, 2.16755, 2.16756, 2.16757, 2.16759, 2.16759, 2.16761, 2.16773,
  2.16775, 2.16779, 2.16782, 2.16784, 2.16784, 2.16786, 2.16796}}
```

In[150]:=
```
T2 = {T2a1, T2a2, T2a3, T2a4, T2a5, T2a6}
```

Out[150]=
```
{{2.05674, 2.05674, 2.05675, 2.05677, 2.05677, 2.05678, 2.05678,
  2.05681, 2.05682, 2.05684, 2.05684, 2.05685, 2.05686, 2.05687, 2.05688},
 {2.07752, 2.0776, 2.07761, 2.07767, 2.07783, 2.078, 2.07802, 2.07804,
  2.07809, 2.07813, 2.07824, 2.07839, 2.0784, 2.07842, 2.07852},
 {2.12325, 2.12325, 2.12335, 2.12336, 2.12336, 2.12339, 2.1234, 2.12344,
  2.1235, 2.12351, 2.12352, 2.12357, 2.12357, 2.12358, 2.1236},
 {2.17541, 2.17545, 2.17546, 2.17551, 2.17551, 2.17553, 2.17553, 2.17553,
  2.17556, 2.17558, 2.17559, 2.1756, 2.17569, 2.17572, 2.17575},
 {2.20518, 2.20579, 2.20627, 2.20664, 2.20724, 2.20744, 2.20776, 2.20804,
  2.20844, 2.20865, 2.20915, 2.20919, 2.20948, 2.20972, 2.20989},
 {2.25955, 2.25956, 2.25964, 2.25966, 2.2597, 2.25976, 2.25978, 2.25982,
  2.25985, 2.25989, 2.26002, 2.26008, 2.26011, 2.2602, 2.25716}}
```

In[151]:=
```
Grid[Transpose[{{"měření T1 [s]", 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
   {"a = 12.65 cm", 2.127417, 2.127463, 2.127574, 2.127582,
    2.127582, 2.127586, 2.127594, 2.127594, 2.127601, 2.127627,
    2.127629, 2.12764, 2.127662, 2.127664, 2.127667},
   {"a = 11.885 cm", 2.132639`, 2.132665`, 2.132667`, 2.132677`,
    2.132703`, 2.132737`, 2.132739`, 2.132744`, 2.132747`, 2.132751`,
    2.132782`, 2.132783`, 2.13279`, 2.132801`, 2.132809`},
   {"a = 10.56 cm", 2.138794`, 2.139408`, 2.139429`, 2.139439`,
    2.139518`, 2.139533`, 2.139597`, 2.139605`, 2.139649`, 2.139655`,
    2.139665`, 2.139711`, 2.139799`, 2.139889`, 2.140059`},
   {"a = 9.10 cm", 2.151919`, 2.152245`, 2.152251`, 2.152259`,
    2.152285`, 2.152309`, 2.152319`, 2.152361`, 2.152383`, 2.152398`,
    2.152442`, 2.152459`, 2.152476`, 2.152621`, 2.152659`},
   {"a = 8.14 cm", 2.158958`, 2.158991`, 2.159089`, 2.159106`,
    2.159122`, 2.159123`, 2.159135`, 2.159184`, 2.159202`, 2.159211`,
    2.159214`, 2.159255`, 2.15926`, 2.159293`, 2.159305`},
   {"a = 6.96 cm", 2.167535`, 2.167551`, 2.16756`, 2.167569`, 2.167586`,
    2.167593`, 2.167609`, 2.167725`, 2.167749`, 2.167794`,
    2.167818`, 2.167835`, 2.167841`, 2.167864`, 2.167956`}
  }], Alignment → Left, Spacings → {2, 1}, Frame → All, ItemStyle → "Text",
 Background → {{Gray, None}, {LightGray, None}}]
```

Out[151]=

měření T_1 [s]	a = 12.65 cm	a = 11.885 cm	a = 10.56 cm	a = 9.10 cm	a = 8.14 cm	a = 6.96 cm
1	2.12742	2.13264	2.13879	2.15192	2.15896	2.16754
2	2.12746	2.13267	2.13941	2.15225	2.15899	2.16755
3	2.12757	2.13267	2.13943	2.15225	2.15909	2.16756
4	2.12758	2.13268	2.13944	2.15226	2.15911	2.16757
5	2.12758	2.1327	2.13952	2.15229	2.15912	2.16759
6	2.12759	2.13274	2.13953	2.15231	2.15912	2.16759
7	2.12759	2.13274	2.1396	2.15232	2.15914	2.16761
8	2.12759	2.13274	2.13961	2.15236	2.15918	2.16773
9	2.1276	2.13275	2.13965	2.15238	2.1592	2.16775
10	2.12763	2.13275	2.13966	2.1524	2.15921	2.16779
11	2.12763	2.13278	2.13967	2.15244	2.15921	2.16782
12	2.12764	2.13278	2.13971	2.15246	2.15926	2.16784
13	2.12766	2.13279	2.1398	2.15248	2.15926	2.16784
14	2.12766	2.1328	2.13989	2.15262	2.15929	2.16786
15	2.12767	2.13281	2.14006	2.15266	2.15931	2.16796

In[152]:=
```
Grid[Transpose[{{"měření T2 [s]", 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
   {"a = 12.65 cm", 2.05674`, 2.056742`, 2.056749`, 2.05677`,
    2.056773`, 2.056776`, 2.05678`, 2.056814`, 2.056815`, 2.05684`,
    2.056842`, 2.056846`, 2.056855`, 2.05687`, 2.056878`},
   {"a = 11.885 cm", 2.077518`, 2.077601`, 2.077612`, 2.077674`,
    2.077832`, 2.077997`, 2.078016`, 2.078042`, 2.078085`, 2.078129`,
    2.078241`, 2.078391`, 2.078398`, 2.07842`, 2.078519`},
   {"a = 10.56 cm", 2.123248`, 2.123253`, 2.123353`, 2.12336`,
    2.123364`, 2.123393`, 2.1234`, 2.123444`, 2.1235`, 2.123505`,
    2.123518`, 2.123569`, 2.123571`, 2.123577`, 2.123595`},
   {"a = 9.10 cm", 2.175406`, 2.175447`, 2.175464`, 2.175506`,
    2.175508`, 2.175528`, 2.175528`, 2.175531`, 2.175559`, 2.175576`,
    2.175591`, 2.175604`, 2.175685`, 2.175724`, 2.17575`},
   {"a = 8.14 cm", 2.205181`, 2.205791`, 2.206268`, 2.206644`,
    2.207238`, 2.207438`, 2.207756`, 2.208043`, 2.208435`, 2.208649`,
    2.209151`, 2.209187`, 2.209479`, 2.209724`, 2.209885`},
   {"a = 6.96 cm", 2.259553`, 2.259557`, 2.259638`, 2.259658`,
    2.259695`, 2.259758`, 2.259778`, 2.259816`, 2.259853`, 2.259886`,
    2.260024`, 2.260076`, 2.260114`, 2.260199`, 2.257163`}
  }], Alignment → Left, Spacings → {2, 1}, Frame → All, ItemStyle → "Text",
 Background → {{Gray, None}, {LightGray, None}}]
```

Out[152]=

měření T_2 [s]	a = 12.65 cm	a = 11.885 cm	a = 10.56 cm	a = 9.10 cm	a = 8.14 cm	a = 6.96 cm
1	2.05674	2.07752	2.12325	2.17541	2.20518	2.25955
2	2.05674	2.0776	2.12325	2.17545	2.20579	2.25956
3	2.05675	2.07761	2.12335	2.17546	2.20627	2.25964
4	2.05677	2.07767	2.12336	2.17551	2.20664	2.25966
5	2.05677	2.07783	2.12336	2.17551	2.20724	2.2597
6	2.05678	2.078	2.12339	2.17553	2.20744	2.25976
7	2.05678	2.07802	2.1234	2.17553	2.20776	2.25978
8	2.05681	2.07804	2.12344	2.17553	2.20804	2.25982
9	2.05682	2.07809	2.1235	2.17556	2.20844	2.25985
10	2.05684	2.07813	2.12351	2.17558	2.20865	2.25989
11	2.05684	2.07824	2.12352	2.17559	2.20915	2.26002
12	2.05685	2.07839	2.12357	2.1756	2.20919	2.26008
13	2.05686	2.0784	2.12357	2.17569	2.20948	2.26011
14	2.05687	2.07842	2.12358	2.17572	2.20972	2.2602
15	2.05688	2.07852	2.1236	2.17575	2.20989	2.25716

Ze získaných dat vypočítáme aritmetické průměry, které použijeme pro výpočet vzdálenosti a_0.

In[153]:=
```
AritmPrumT1 = {Mean[T1a1], Mean[T1a2], Mean[T1a3], Mean[T1a4], Mean[T1a5], Mean[T1a6]}
```

Out[153]=
```
{2.12759, 2.13274, 2.13958, 2.15236, 2.15916, 2.16771}
```

In[154]:=
```
AritmPrumT2 = {Mean[T2a1], Mean[T2a2], Mean[T2a3], Mean[T2a4], Mean[T2a5], Mean[T2a6]}
```

Out[154]=
```
{2.05681, 2.07803, 2.12344, 2.17556, 2.20792, 2.25965}
```

In[155]:=
```
Grid[Transpose[{{"vzdálenost závaží", "12.65 cm", "11.885 cm", "10.56 cm",
    "9.1 cm", "8.14 cm", "6.96 cm"}, {"T1 [s]", 2.1275, 2.1327355999999997`,
    2.1395833333333334`, 2.152359066666667`, 2.1591632`, 2.167705666666667`},
   {"T2 [s]", 2.056806`, 2.0780316666666665`, 2.1234433333333333`,
    2.1755604666666666`, 2.2079245999999997`, 2.2596512`}}],
 Alignment → Left, Spacings → {2, 1}, Frame → All, ItemStyle → "Text",
 Background → {{Gray, None}, {LightGray, None}}]
```

Out[155]=

vzdálenost závaží	T_1 [s]	T_2 [s]
12.65 cm	2.1275	2.05681
11.885 cm	2.13274	2.07803
10.56 cm	2.13958	2.12344
9.1 cm	2.15236	2.17556
8.14 cm	2.15916	2.20792
6.96 cm	2.16771	2.25965

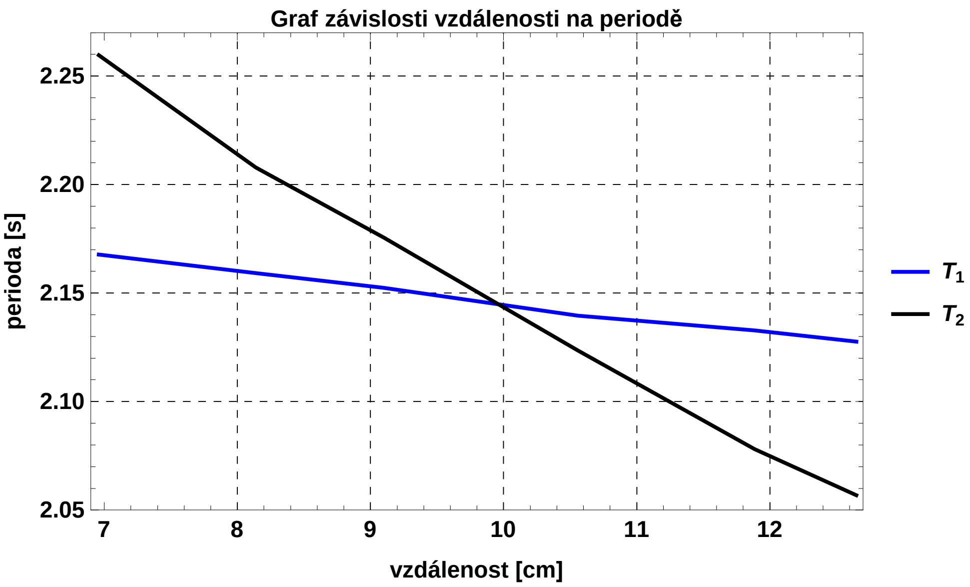
Nyní sestavíme graf závislosti vzdálenosti závaží na periodách T_1 a T_2 a vypočteme vzdálenost a_0.

In[156]:=
```
DG1 = {{6.96, 2.16771}, {8.14, 2.15916}, {9.1, 2.15236},
   {10.56, 2.13958}, {11.885, 2.13274}, {12.65, 2.12759}};
```

In[157]:=
```
DG2 = {{6.96, 2.25965}, {8.14, 2.20792}, {9.1, 2.17556},
   {10.56, 2.12344}, {11.885, 2.07803}, {12.65, 2.05681}};
```

In[158]:=
```
ListLinePlot[{DG1, DG2}, PlotRange → {{6.9, 12.7}, {2.05, 2.27}},
 PlotStyle → {{Blue, Thick}, {Black, Thick}},
 FrameLabel → { "vzdálenost [cm]", "perioda [s]"},
 PlotLabel → Style["Graf závislosti vzdálenosti na periodě", FontSize → 12 ],
 LabelStyle → Directive[12, Bold], ImageSize → 450 , PlotLegends → {"T1", "T2"},
 Frame → True, GridLines → Automatic, GridLinesStyle → Dashed]
```

Out[158]=



Průsečík křivek můžeme z grafu určit jednoduše. Graf si zvětšíme. Po kliknutí (pravým tlačítkem) do grafu zvolíme možnost **Get Coordinates** a odečteme souřadnice průsečíku křivek.

In[159]:=
```
a0T = {{9.963, 2.145}};
```

V poslední části měření nastavíme vypočtenou polohu závaží na reverzním kyvadle a měříme periodu pohybu pro tuto vzdálenost, opět pro dvě osy otáčení. Pro výpočet tíhového zrychlení použijeme aritmetický průměr periody T_1 a T_2.

In[160]:=
```
T1a0 = {2.147, 2.141, 2.142, 2.147, 2.138, 2.154, 2.138, 2.143, 2.150, 2.150};
```

In[161]:=
```
T2a0 = {2.141, 2.150, 2.144, 2.140, 2.144, 2.144, 2.159, 2.147, 2.147, 2.150};
```

In[162]:= $\overline{L}$ = Mean[delkaL]

Out[162]= 1.1454

In[163]:=
```
AritmPrumT1a0 = Mean[T1a0]
AritmPrumT2a0 = Mean[T2a0]
```

Out[163]= 2.145

Out[164]= 2.1466

In[165]:= $\overline{T} = \frac{\text{AritmPrumT1a0} + \text{AritmPrumT2a0}}{2}$

Out[165]= 2.1458

In[166]:= $\overline{g} = \frac{4 \pi^2 \overline{L}}{(\overline{T})^2}$

Out[166]= 9.82061

V poslední části měření je vypočítána chyba měření. Tíhové zrychlení je fukncí dvou měřených veličin; periody a délky: $g = g(L, T)$.

Chybu měření veličin L, T_1 a T_2 vypočítáme jako standardní kvadratickou odchylku rozptylu veličiny.

In[167]:=
$\sigma L = \text{MeanDeviation[delkaL]} / \sqrt{10}$;
$\sigma T1 = \text{MeanDeviation[T1a0]} / \sqrt{10}$;
$\sigma T2 = \text{MeanDeviation[T2a0]} / \sqrt{10}$;

In[170]:= $\sigma T = \frac{\sigma T2 + \sigma T1}{2}$;

Chybu měření vypočítáme podle vztahu $\sigma g = g \sqrt{\left(\frac{\sigma L}{L}\right)^2 + \left(\frac{2\,\sigma T}{T}\right)^2}$ a chybu měření veličin L, T_1 a T_2 vypočítáme jako standardní kvadratickou odchylku rozptylu veličiny.

In[171]:= $\sigma g = \overline{g} \sqrt{\left(\frac{\sigma L}{\overline{L}}\right)^2 + \left(\frac{2\,\sigma T}{\overline{T}}\right)^2}$

Out[171]= 0.0125144

In[172]:= gravityAcceleration = $(\overline{g} \pm \sigma g)$

Out[172]= 9.82061 ± 0.0125144

Relativní chyba v procentech je:

In[173]:= $\frac{0.012514366985729261\text{`}}{9.820609300693011\text{`}}\ 100$

Out[173]= 0.12743

Závěr měření a diskuze:

Úkolem měření bylo stanovit hodnotu tíhového zrychlení pomocí reverzního kyvadla v Olomouci (poloha: 49,61°severní zeměpisné šířky, 213 metrů nad mořem). Naše výsledná hodnota tíhového zrychlení je $\boldsymbol{g} = (9.82 \pm 0.01)\ \frac{\mathrm{m}}{s^2}$. Při srovnání s hodnotou ze serveru WolframAlpha $\boldsymbol{g} = 9.81\ \frac{\mathrm{m}}{s^2}$ lze měření považovat za přesné. Relativní chyba měření je 0,13 %.

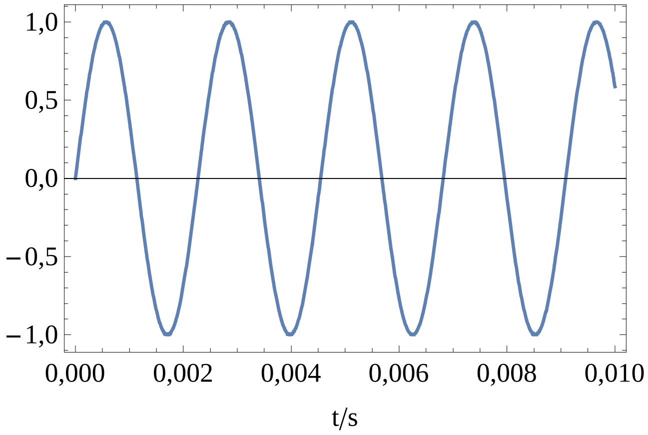
5 Klasická mechanika

5.1 Skládání kmitů

Klasická mechanika nabízí celou řadu zajímavých problémů. Zejména ve spojení s řešením pohybových rovnic. Vykresleme harmonický signál odpovídající komornímu a, tj. frekvenci $f = 440$ Hz, amplitudu volme jednotkovou.

In[174]:=
```
f₁ = 440;
Style[Plot[Sin[2 π f₁ t], {t, 0, 0.01},
  PlotStyle → {Thickness[0.006]}, FrameLabel → {"t/s", ""}, Frame → True,
  BaseStyle → {FontFamily → "Times", FontSize → 14}], NumberPoint → ","]
```

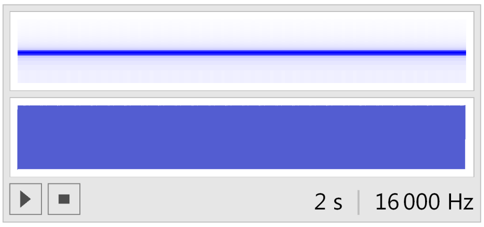
Out[175]=



Pokud to instalace programu *Mathematica* ve Vašem počítači umožňuje, můžete si signál i přehrát příkazem **Play**, v našem případě volíme délku zvuku 2 s.

In[176]:=
```
Play[Sin[2 π f₁ t], {t, 0, 2}, SampleRate → 16 000]
```

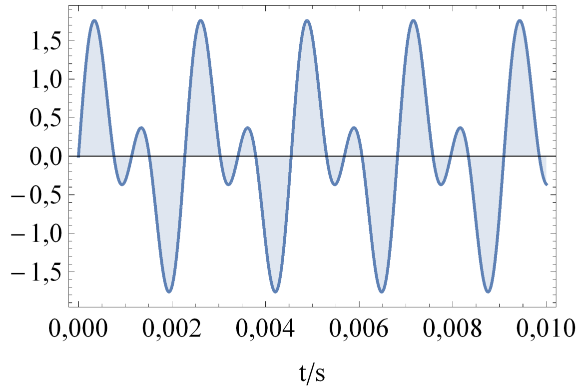
Out[176]=



Nyní složíme signál se signálem s dvojnásobnou frekvencí.

In[177]:=
```
signal1 = Sin[2 π f1 t] + Sin[2 π 2 f1 t];
Style[Plot[signal1, {t, 0, 0.01}, PlotStyle → {Thickness[0.006]},
  FrameLabel → {"t/s", ""}, Frame → True, Filling → Axis,
  BaseStyle → {FontFamily → "Times", FontSize → 14}], NumberPoint → ","]
```

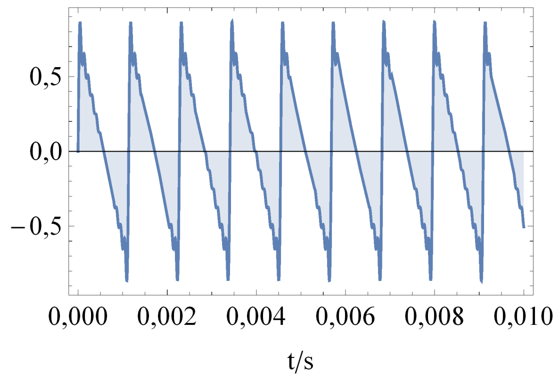
Out[178]=



Kmitů můžeme skládat i více např. definováním vhodné tabulky a sečtením jejích prvků příkazem **`Total`**.

In[179]:=
```
signal2 = Total[Table[Sin[2 π n f1 t] / n, {n, 2, 24, 2}]];
Style[Plot[signal2, {t, 0, .01}, PlotStyle → {Thickness[0.006]},
  FrameLabel → {"t/s", ""}, Frame → True, Filling → Axis,
  BaseStyle → {FontFamily → "Times", FontSize → 14}], NumberPoint → ","]
```

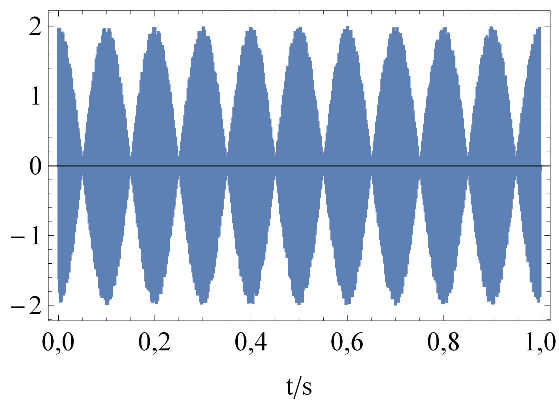
Out[180]=



Složením kmitů blízké frekvence vznikají rázy.

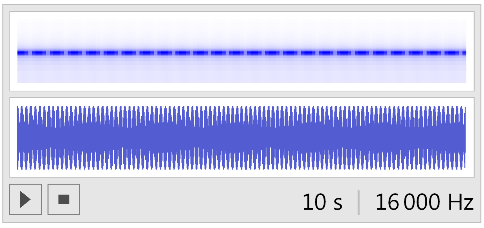
In[181]:=
```
signal3 = Sin[2 π f1 t] + Sin[2 π (f1 + 10) t];
Style[Plot[signal3, {t, 0, 1}, PlotStyle → {Thickness[0.006]},
  FrameLabel → {"t/s", ""}, Frame → True, Filling → Axis, Axes → {True, False},
  BaseStyle → {FontFamily → "Times", FontSize → 14}], NumberPoint → ","]
```

Out[182]=



In[183]:=
```
Play[signal3, {t, 0, 10}, SampleRate → 16 000]
```

Out[183]=



Data získaná skládáním jsou poměrně vděčná na zkoumání Fourierovy analýzy – víme totiž dopředu, jaký výsledek bychom měli očekávat. Omezíme se na vzorky s trváním 0,1 s. Nejprve vygenerujeme datové vzorky.

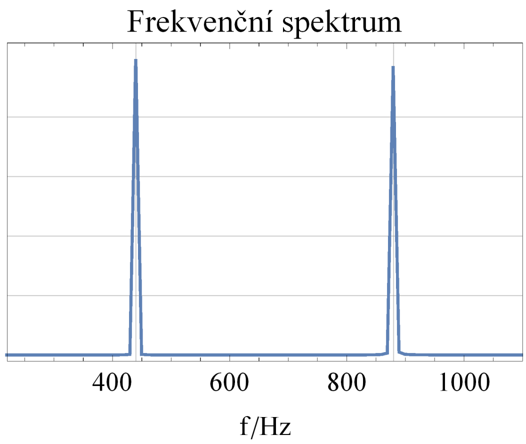
In[184]:=
```
caskrok = 0.0001;
data1 = Table[N[signal1], {t, 0, 0.1, caskrok}];
data2 = Table[N[signal2], {t, 0, 0.1, caskrok / 24}];
```

Nyní použijeme příkaz **Fourier** a vykreslíme frekvenční spektrum.

In[187]:=
```
ffts = Fourier[data1];
len = Length[data1] / 2;
freq = 0.5 * 1 / caskrok * Range[len] / len;
power = (Abs[ffts] ^2) [[Range[2, len + 1]]];
```

In[191]:=
```
ListPlot[Transpose[{freq, power}], Joined → True,
 PlotStyle → {Thickness[0.006]}, PlotRange → {{0.5 f1, 2.5 f1}, All},
 FrameLabel → {"f/Hz", ""}, PlotLabel → "Frekvenční spektrum",
 GridLines → {{f1, 2 f1}, Automatic}, Frame → True, Axes → False,
 FrameTicks → {Automatic, None}, BaseStyle → {FontFamily → "Times", FontSize → 14}]
```

Out[191]=



Polohu maxima v získaných datech můžeme najít např. pomocí příkazu **Max**.

In[192]:=
```
maxpos = Position[power, Max[power]]
fm = freq[[maxpos[[1, 1]]]];
nameAndValue[fm, 1]
```

Out[192]= {{44}}

f_m=440,0

Vidíme, že maximum odpovídající frekvenci 440 Hz je v naší matici na 44. pozici, v rozmezí 50.–100. pozice můžeme očekávat polohu 2. maxima.

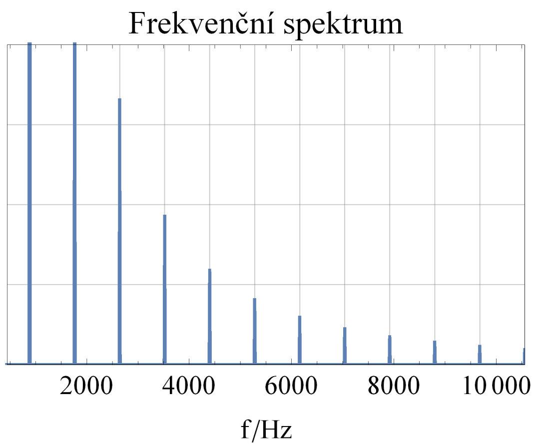
In[195]:=
```
maxpos2 = Position[power, Max[power[[Round[50] ;; Round[100]]]]];
fm2 = freq[[maxpos2[[1, 1]]]];
nameAndValue[fm2, 1]
```

f_{m2}=879,0

In[198]:=
```
ffts = Fourier[data2];
len = Length[data2] / 2;
freq = 0.5 × 24 / caskrok * Range[len] / len;
power = (Abs[ffts] ^2)[[Range[2, len + 1]]];
```

In[202]:=
```
ListPlot[Transpose[{freq, power}], Joined → True,
 PlotStyle → {Thickness[0.006]}, PlotRange → {{f1, 24 f1}, {0, 200}},
 FrameLabel → {"f/Hz", ""}, PlotLabel → "Frekvenční spektrum",
 GridLines → {Table[i f1, {i, 2, 24, 2}], Automatic}, Frame → True, Axes → False,
 FrameTicks → {Automatic, None}, BaseStyle → {FontFamily → "Times", FontSize → 14}]
```

Out[202]=



5.2 Keplerova úloha

Keplerova úloha o pohybu tělesa v centrálním silovém poli patří ke klasickým problémům mechaniky a umožňuje zavést i pochopit význam zavedení efektivního potenciálu. Právě na tento pojem a nikoli na řešení samotné pohybové rovnice se zde zaměříme. Při výpočtech použijeme – jak je při řešení obvyklé – polárních souřadnic a rovnic kuželoseček v polárních souřadnicích. Pomocí efektivního potenciálu lze ukázat, že charakter trajektorie při zadaném momentu hybosti L závisí na celkové mechanické energii částice E_n; pro vázané trajektorie (kružnice, elipsa) je $E_n < 0$, pro nevázané $E_n = 0$ (parabola) nebo $E_n > 0$ (hyperbola). Pohyb částice je možná pouze v oblasti, kde $E_n \geq U_{ef}$.
Nejprve vykresleme vázanou eliptickou trajektorii. Pro jednoduchost položíme hmotnost pohybující se testovací částice m a gravitační konstantu G rovnu jedné, neboť cílem je pouze kvalitativní popis pohybu a průběhu efektivního potenciálu.

$$U_{ef} = \frac{L^2}{2 r^2} - \frac{M}{r}.$$

Ze zadaných hodnot energie a momentu hybnosti vypočteme číselnou výstřednost dráhy ε, vzdálenost v periheliu r_p (pokud podle tradice uvažujeme pohyb v gravitačním poli Slunce) a vzdálenost afeliu r_a. V našich zjednodušených jednotkách pak má rovnice kuželosečky v polárních souřadnicích tvar:

$$r = \frac{L^2}{-1+\varepsilon \cos(\varphi)}.$$

Směr $\varphi = 0$ odpovídá periheliu. K vykreslení trajektorie proto použijeme příkaz **`ParametricPlot3D`**. Pohyb částice na pozadí vykresleného efektivního potenciálu podél kuželosečky můžeme interaktivně vykreslit pomocí příkazu **`Manipulate`**, osu z omezíme pomocí zjištění minimální hodnoty efektivního potenciálu přílazem **`FindMinimum`**. V obrázku jsou také načrtnuty kružnice odpovídající vzdálenostem v periheliu a afeliu, kterých se skutečná eliptická trajektorie dotýká vždy v jednom bodě.

In[203]:=
```
L = 1.8;
En = -.07;
eps = Sqrt[1 + 2 * En * L^2]; Print["ε=", NumberForm[eps, {3, 3}, NumberPoint → ","]]
rp = L^2 / (1 + eps); nameAndValue[rp, 2]
ra = L^2 / (1 - eps); nameAndValue[ra, 1]
M = (L^2 / 2 / rp^2 - En) * rp;
FindMinimum[L^2 / 2 / r^2 - M / r, {r, rp}]
minimumU = %[[1]] ;
kresba1 =
  ParametricPlot3D[{{L^2 / (1 + eps * Cos[u]) * Cos[u], L^2 / (1 + eps * Cos[u]) * Sin[u], En},
     {rp * Cos[u], rp * Sin[u], En}, {ra * Cos[u], ra * Sin[u], En}},
    {u, 0, 2 * π}, Boxed → False, Axes → False, PlotRange → {All, All, All},
    PlotStyle → {{Thickness[0.0075]},
      {Red, Dashed, Thickness[0.005]}, {Red, Dashed, Thickness[0.005]}}];
kresba2 = Plot3D[L^2 / 2 / (x^2 + y^2) - M / Sqrt[x^2 + y^2], {x, -1.1 * ra, 1.1 * ra},
    {y, -1.1 * ra, 1.1 * ra}, Boxed → False, Axes → False, Ticks → None];
grafUa[t_] = Plot[{En, L^2 / 2 / r^2 - M / r}, {r, 0, 1.3 * ra}, Filling → Axis,
    FrameTicks → {{None, None}}, AxesLabel → {"r", "Uef"},
    Ticks → {{{rp, "rp"}, {ra, "ra"}}, {{En, "En"}}},
    Epilog → {Dashed, PointSize[0.02], {Line[{{rp, -1}, {rp, En}}],
       Line[{{ra, -1}, {ra, En}}], Point[{L^2 / (1 + eps * Cos[t]), En}]}},
    BaseStyle → {FontFamily → "Times", FontSize → 14}];
```

ε=0,739

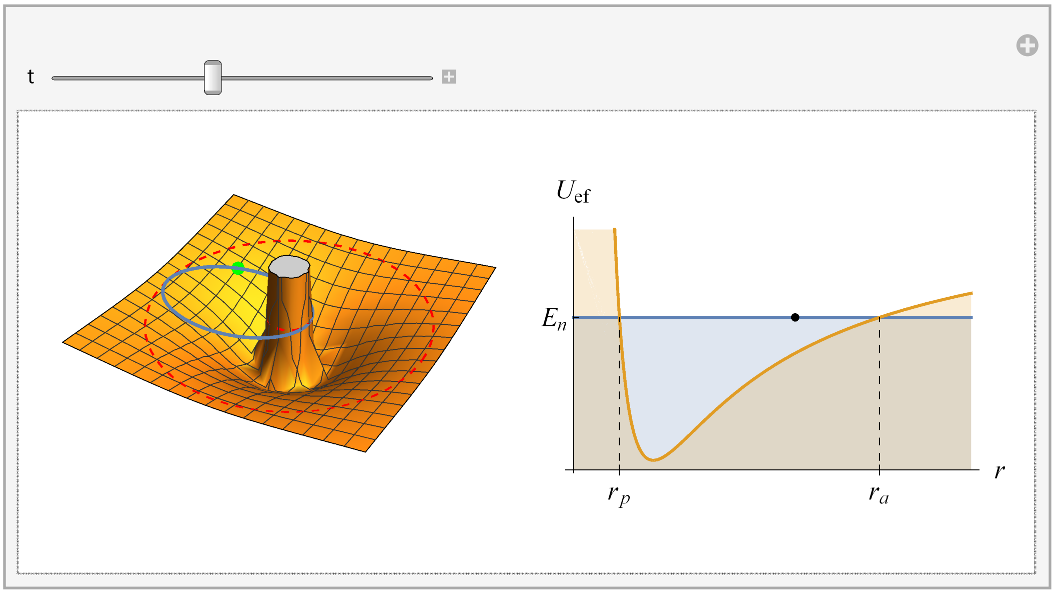
r_p=1,86

r_a=12,4

Out[209]= {-0.154321, {r → 3.24}}

In[214]:=
```
Manipulate[
 GraphicsRow[{Show[{kresba2, kresba1, Graphics3D[{Green, PointSize[0.03], Point[
        {L^2 / (1 + eps * Cos[t]) * Cos[t], L^2 / (1 + eps * Cos[t]) * Sin[t], En}]}]}],
   grafUa[t]} , ImageSize → 500], {t, 0, 2 * π, π / 120}]
```

Out[214]=



V případě nevázané parabolické trajektorie je afelium v nekonečnu, vykreslené hodnoty poloměru proto omezíme seshora podle uvážení. Zde už se spokojíme pouze s vykreslením, případnou animaci přenecháváme čtenáři.

In[215]:=
```
En1 = 0;
eps1 = Sqrt[1 + 2 En1 L^2]; Print["ε=", NumberForm[eps1, {3, 3}, NumberPoint → ","]]
rp1 = L^2 / (1 + eps1); nameAndValue[rp1, 2]
M1 = (L^2 / 2 / rp1^2 - En1) rp1;
FindMinimum[L^2 / 2 / r^2 - M / r, {r, rp1}]
minimumU1 = %[[1]] ;
```

ε=1,000

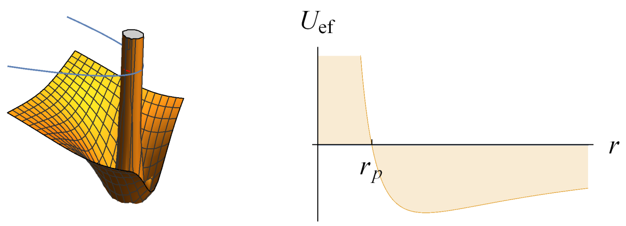
r_{p1}=1,62

Out[219]= {-0.154321, {r → 3.24}}

In[221]:=
```
kresba11a =
  ParametricPlot3D[{L^2 / (1 + eps1 Cos[u]) Cos[u], L^2 / (1 + eps1 Cos[u]) Sin[u], En1},
   {u, -5 π / 6, 5 π / 6}, Boxed → False, Axes → False, PlotStyle → {Thickness[0.0075]}];
kresba11b = ParametricPlot3D[{rp1 Cos[u], rp1 Sin[u], En1}, {u, 0, 2 π},
   Boxed → False, Axes → False, PlotStyle → {Red, Dashed, Thickness[0.005]}];
kresba21 = Plot3D[L^2 / 2 / (x^2 + y^2) - M1 / Sqrt[x^2 + y^2], {x, -10 rp1, 3 rp1},
   {y, -8 rp1, 8 rp1}, Boxed → False, Axes → False, Ticks → None, BoxRatios → {1, 1, 1.5}];
grafU1 = Plot[{En1, L^2 / 2 / r^2 - M / r}, {r, 0, 5 rp1}, Filling → Axis,
   FrameTicks → {{None, None}}, AxesLabel → {"r", "Uef"}, Ticks →
    {{{rp1, "rp"}}, {{En1, "En"}}}, BaseStyle → {FontFamily → "Times", FontSize → 14},
   PlotStyle → {{Thickness[0.005]}, {Thickness[0.002]}}];
```

In[225]:=
```
GraphicsRow[{Show[{kresba21, kresba11b, kresba11a}], grafU1}]
```

Out[225]=



U nevázané hyperbolické trajektorie je bezpečnější určit si úhel asymptot (jinak by mohl být s vykreslením trajektorie problém), zbývající část je pak už analogická předchozím případům.

In[226]:=
```
En2 = 0.07;
eps2 = Sqrt[1 + 2 En2 L^2]; Print["ε=", NumberForm[eps2, {3, 3}, NumberPoint → ","]]
rp2 = L^2 / (1 + eps2); nameAndValue[rp, 2]
M2 = (L^2 / 2 / rp2^2 - En2) rp2;
maxuhel = ArcCos[1 / eps2]; nameAndValue[maxuhel, 2]
FindMinimum[L^2 / 2 / r^2 - M2 / r, {r, rp2}]
minimumU2 = %[[1]] ;
```

ε=1,210

r_p=1,86

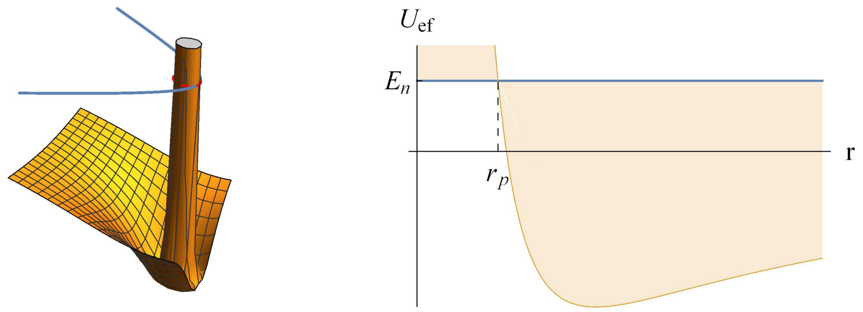
maxuhel=0,59

Out[231]= {-0.154321, {r → 3.24}}

In[233]:=
```
kresba12a =
  ParametricPlot3D[{L^2 / (1 + eps2 Cos[u]) Cos[u], L^2 / (1 + eps2 Cos[u]) Sin[u], En2},
   {u, -π + maxuhel + 0.2, π - maxuhel - 0.2}, Boxed → False,
   Axes → False, PlotStyle → {Thickness[0.01]}];
kresba12b = ParametricPlot3D[{1.1 rp2 Cos[u], 1.1 rp2 Sin[u], En2}, {u, 0, 2 π},
   Boxed → False, Axes → False, PlotStyle → {Red, Dashed, Thickness[0.0075]}];
kresba22 = Plot3D[L^2 / 2 / (x^2 + y^2) - M2 / Sqrt[x^2 + y^2],
   {x, -13 rp2, 2 rp2}, {y, -8 rp2, 8 rp2}, Boxed → False, Axes → False,
   PlotRange → {All, All, {minimumU2, 1.5 En2}}, BoxRatios → {1, 1, 1.5}];
grafU2 = Plot[{En2, L^2 / 2 / r^2 - M / r}, {r, 0, 5 rp2}, Filling → {2 → {1}},
   FrameTicks → {{None, None}}, AxesLabel → {"r", "Uef"}, Ticks →
    {{{rp2, "rp"}}, {{En2, "En"}}}, BaseStyle → {FontFamily → "Times", FontSize → 14},
   PlotStyle → {{Thickness[0.005]}, {Thickness[0.002]}}, Epilog →
    {Dashed, Line[{{rp2, 0}, {rp2, En2}}]}, PlotRange → {All, {minimumU2, 1.5 En2}}];
```

In[237]:=
```
GraphicsRow[{Show[{kresba22, kresba12b, kresba12a}], grafU2}]
```

Out[237]=



5.3 Buquoyova úloha

Tato úloha z dynamiky soustav s proměnnou hmotností má zajímavou historii a historické souvislosti. Je spojena se jménem Jiřího Františka Augusta Buquoye (1781–1851), šlechtice, matematika a vynálezce, vzdáleného potomka velitele císařských vojsk v bělohorské bitvě Karla Bonaventury Buquoye, kterému za prokázané služby v porážce českého stavovského povstání císař Ferdinand II. Habsburský v roce 1620 daroval panství Nové Hrady, Rožmberk, Liběјovice a tvrze Žumberk a Cuknštejn. J. F. A. Buquoy studoval matematiku, přírodní vědy, filozofii a ekonomii ve Vídni a v Praze. Svoje znalosti uplatňoval při správě rodového majetku – mj. sestrojil v roce 1810 parní stroj, který se snažil uplatnit v praxi, zasloužil se o rozvoj novohradských skláren, kde zavedl unikátní a dnes již zapomenutou technologii výroby černého neprůhledného skla — hyalitu (1817), v roce 1838 založil přírodní rezervaci Žofínský prales. Okruh jeho aktivit byl mimořádně široký, zahrnoval i práce a úvahy na témata z filozofie, práva, politiky a umění.

J. F. A. Buquoy byl pravděpodobně první, kdo sestavoval a řešil úlohy na pohyb soustav s proměnnou hmotností. První takovou úlohu uvádí ve své knize z roku 1814 (Buquoy, G.: Weitere Entwickelung und Anwendung des Gesetzes der virtuellen Geschwindigkeiten in mechanischer und statischer Hinsicht. Breitkopf und Härtel: Leipzig, 1814). Její zadání je velmi jednoduché:

Na vodorovné podložce leží smotané dokonale ohebné vlákno. Určete jeho pohyb, působí-li na jeho konec svisle vzhůru konstantní síla.

Při řešení této Buquoyovy úlohy předpokládáme, že
• vlákno je jednorozměrné s konstantní lineární hustotou η (neprotažitelné),
• poloha jeho konce nad vodorovnou podložkou je charakterizována souřadnicí x > 0,
• dokonale ohebné vlákno se při pohybu vzhůru (proti tíhovému zrychlení) bez tření odmotává v počátku osy x,
• při pohybu dolů v počátku osy x vlákno „mizí" — část, která dopadne na podložku, se nepohybuje (hybnost se předává podložce s obrovskou hmotností).

Za těchto podmínek je Buquoyův problém popsán pohybovou rovnicí, která speciálním způsobem závisí na rychlosti:

$$y''(t) = g\left(\frac{y_c}{y(t)} - 1\right) - \frac{0.5\, y'(t)^2\,(\operatorname{sgn}(y'(t))+1)}{y(t)}.$$

Ukažme numerické řešení pomocí funkce **NDSolve**. Význačnou hodnotou – jak ukáže řešení – je “rovnovážná poloha”, délka svislé části vlákna y_c, pro kterou je tíha této svislé části rovna působící síle F.

Lineární hustota:

In[238]:=
```
η = 0.005;
```

Síla působící svisle vzhůru:

In[239]:=
```
F = 0.1;
```

Rovnovážná poloha:

In[240]:=
```
g = 9.8;
yc = F / η / g;
```

Počáteční poloha:

In[242]:=
```
yp = 1;
```

Maximální čas, do kterého bude probíhat numerická integrace:

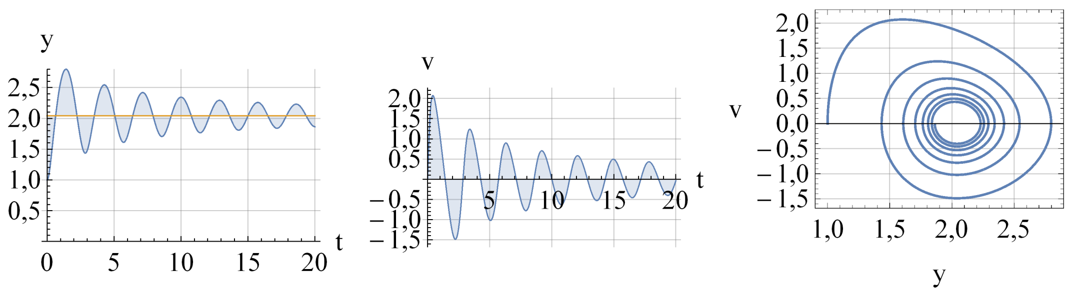
In[243]:=
```
tm = 20;
```

In[244]:=
```
Konst = F / 2 / η yp + g / 3 yp^3;
s = NDSolve[{y''[t] == g (yc / y[t] - 1) - 0.5 (1 + Sign[y'[t]]) y'[t]^2 / y[t],
    y[0] == 1, y'[0] == 0}, y, {t, 0, tm}];
hm = Max[FindMaximum[y[t] /. s, t], yp][[1]];
```

In[247]:=
```
graf1 = Style[Plot[{Evaluate[y[t] /. s], yc},
    {t, 0, tm}, PlotRange → {Automatic, {0, hm}}, Filling → {1 → {2}},
    BaseStyle → {FontFamily → "Times", FontSize → 14}, AxesLabel → {"t", "y"},
    GridLines → Automatic, PlotStyle → {Thickness[0.005]}], NumberPoint → ","];
graf2 = Style[Plot[Evaluate[y'[t] /. s], {t, 0, tm}, Filling → Axis,
    BaseStyle → {FontFamily → "Times", FontSize → 14}, AxesLabel → {"t", "v"},
    GridLines → Automatic, PlotStyle → {Thickness[0.005]}], NumberPoint → ","];
graf3 = Style[ParametricPlot[Evaluate[{y[t], y'[t]} /. s], {t, 0, tm},
    BaseStyle → {FontFamily → "Times", FontSize → 14}, Frame → True,
    FrameLabel → {"y", "v"}, RotateLabel → False, GridLines → Automatic,
    PlotStyle → {Thickness[0.01]}, AspectRatio → 0.8], NumberPoint → ","];
```

In[250]:=
```
GraphicsRow[{graf1, graf2, graf3}, ImageSize → {550, 200}]
```

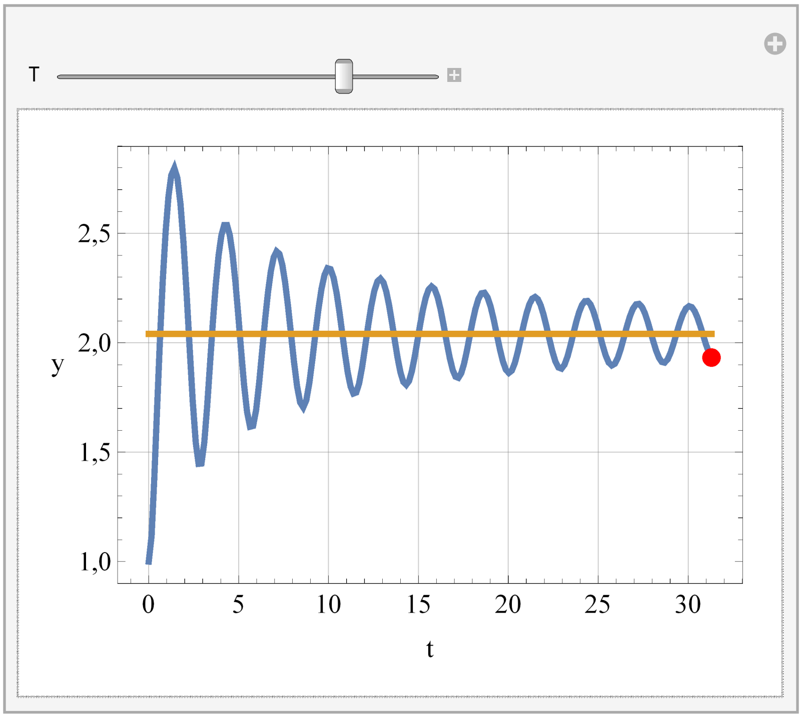
Out[250]=



Vidíme, že pokud nezačínáme v klidu v poloze y_c, podobá se pohyb tlumeným kmitům (ale nejde o harmonické kmity) a směřuje k rovnováze v poloze y_c. Kromě vykreslení závislosti polohy a rychlosti konce vlákna na čase a znázornění pohybu pomocí fázového diagramu můžeme pomocí příkazu **Manipulate** získat interaktivní vykreslení řešení pro různé časové intervaly; postupným zvyšováním horní meze času tak vlastně získáme animaci řešení.

In[251]:=
```
Manipulate[
 Module[{sol = NDSolve[{y''[t] == g (yc / y[t] - 1) - 0.5 (1 + Sign[y'[t]]) y'[t]^2 / y[t],
      y[0] == p[[1]], y'[0] == p[[2]]}, y, {t, 0, T}]},
  Style[ParametricPlot[{Evaluate[{t, y[t]} /. sol], {t, yc}}, {t, 0, T},
    Frame → True, FrameLabel → {"t", "y"}, RotateLabel → False,
    GridLines → Automatic, PlotStyle → {Thickness[0.01]}, PlotRange → All,
    AspectRatio → 0.7, BaseStyle → {FontFamily → "Times", FontSize → 14},
    Epilog → {Red, PointSize[0.03], {Point[Evaluate[{T, y[T]} /. sol]]}}],
   NumberPoint → ","]], {{p, {1, 0}}, Locator}, {{T, 0.01}, 0, 40}]
```

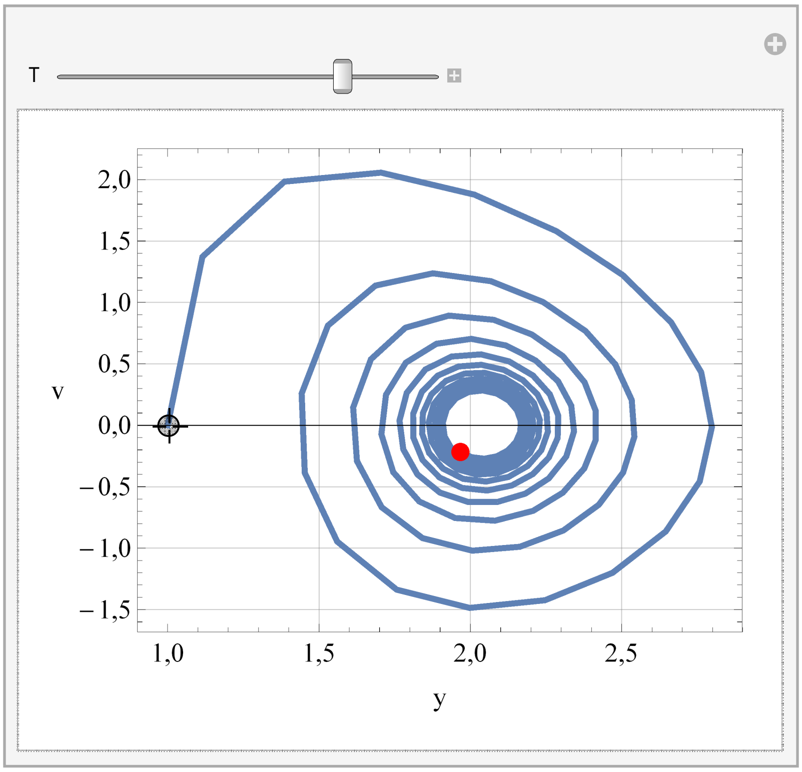
Out[251]=



Totéž můžeme aplikovat i na interaktivní vykreslení fázového diagramu.

In[252]:=
```
Manipulate[
 Module[{sol = NDSolve[{y''[t] == g (yc / y[t] - 1) - 0.5 * (1 + Sign[y'[t]]) y'[t]^2 / y[t],
      y[0] == p[[1]], y'[0] == p[[2]]}, y, {t, 0, T}]},
  Style[ParametricPlot[Evaluate[{y[t], y'[t]} /. sol], {t, 0, T}, Frame → True,
    FrameLabel → {"y", "v"}, RotateLabel → False, GridLines → Automatic,
    PlotStyle → {Thickness[0.01]}, AspectRatio → .8, PlotRange → All,
    BaseStyle → {FontFamily → "Times", FontSize → 14},
    Epilog → {Red, PointSize[0.03], {Point[Evaluate[{y[T], y'[T]} /. sol]}}],
   NumberPoint → ","]], {{p, {1, 0}}, Locator}, {{T, 0.01}, 0, 40}]
```

Out[252]=



5.4 Van der Polův oscilátor

Van der Polův oscilátor sehrál významnou úlohu ve vývoji nelineární dynamiky. Z matematického hlediska je popsán van der Polovou rovnicí:

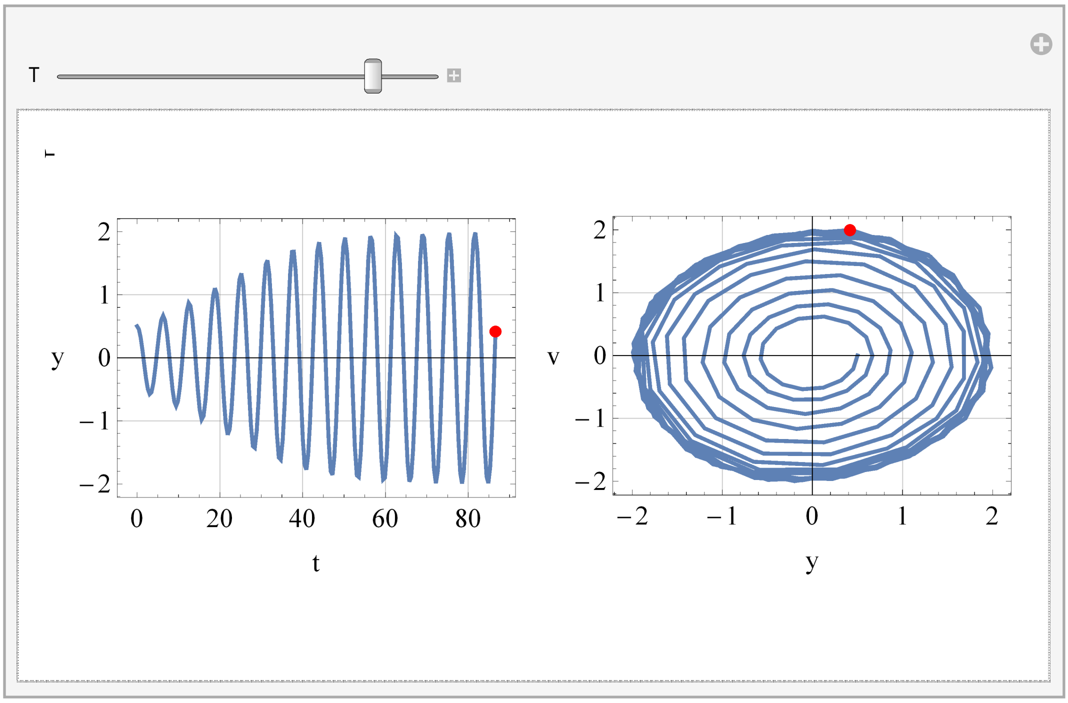
$y''(t) + \mu\,(y(t)^2 - 1)\,y'(t) + y(t) = 0,$

kde μ je konstantní parametr. Rovnice tohoto typu byla objevena při studiu nelineárních elektrických obvodů u prvních radiopřijímačů v roce 1926 holandským inženýrem B. van der Polem, podobnou rovincí se zabýval už lord Rayleigh okolo roku 1880 v souvislosti s nelineárními vibracemi. Pohybová rovnice se od rovnice harmonického oscilátoru liší prostředním členem, jež odpovídá nelineárnímu tlumení. Pro $|y| > 1$ vede ke skutečnému kladnému tlumení, naopak pro $|y| < 1$ představuje negativní tlumení (buzení) systému. Jinými slovy, tlumí velké a budí malé výchylky. Lze odhadnout, že systém může přejít do stavu, kdy se obě tendence vyrovnají a z obecnějších úvah lze ukázat, že pro každé $\mu > 0$ má rovnice jednoznačně určený limitní periodický režim. Jak ukážeme, s rostoucím μ se tyto limitní kmity více a více liší od harmonického pohybu. Čtenář si může na několika případech ověřit, že limitní kmity závisejí pouze na parametru μ a nikoli na zvolených počátečních podmínkách (s výjimkou případu $y_0 = v_0 = 0$, kdy ke kmitům nedojde vůbec).

Nejprve zvolme poměrně malou hodnotu $\mu = 0{,}01$.

In[253]:=
```
μ = 0.1;
Manipulate[
 Module[{solvdp = NDSolve[{y''[t] == -μ (y[t]^2 - 1) y'[t] - y[t], y[0] == p[[1]],
      y'[0] == p[[2]]}, y, {t, 0, T}, PrecisionGoal → 30]},
  GraphicsRow[{Style[ParametricPlot[Evaluate[{t, y[t]} /. solvdp], {t, 0, T},
      Frame → True, FrameLabel → {"t", "y"}, RotateLabel → False,
      GridLines → Automatic, PlotStyle → {Thickness[0.01]}, PlotRange → All,
      AspectRatio → 0.7, BaseStyle → {FontFamily → "Times", FontSize → 14},
      Epilog → {Red, PointSize[0.03], {Point[Evaluate[{T, y[T]}] /. solvdp ]}}],
     NumberPoint → ","], Style[ParametricPlot[Evaluate[{y[t], y'[t]} /. solvdp],
      {t, 0, T}, Frame → True, FrameLabel → {"y", "v"}, RotateLabel → False,
      GridLines → Automatic, PlotStyle → {Thickness[0.01]}, PlotRange → All,
      AspectRatio → 0.7, BaseStyle → {FontFamily → "Times", FontSize → 14},
      Epilog → {Red, PointSize[0.03], {Point[Evaluate[{y[T], y'[T]}] /. solvdp ]}}],
     NumberPoint → ","]}, ImageSize → 500]],
 {{p, {0.5, 0}}, Locator}, {{T, 0.01}, 0, 100}]
```

Out[254]=



6 Teorie elektromagnetického pole

6.1 Znázornění polí elektrostatických multipólů složených z bodových nábojů

Elektrostatické pole je pole potenciálové. Vektor elektrické intenzity hledáme ve tvaru $\vec{E} = -\operatorname{grad}\varphi$, kde φ je skalární potenciál elektrostatického pole. Skalární potenciál bodového náboje je $\varphi = \frac{1}{4\pi\epsilon}\cdot\frac{Q}{r}$, kde Q je velikost bodového náboje a r je jeho vzdálenost od místa, ve kterém je skalární potenciál hledán. Skalární potenciál je aditivní veličina, proto potenciál soustavy bodových nábojů hledáme ve tvaru

$\varphi = \sum_{i=1}^{n} \frac{1}{4\pi\epsilon}\cdot\frac{Q_i}{r_i}$.

Elektrostatické pole se znázorňuje pomocí siločar – křivek, pro které platí, že vektor elektrické intenzity $\vec{E}$ je k nim v každém bodě tečnou. Jednou z možností vykreslení siločar je využití standardních knihoven a funkcí software *Mathematica*, konkrétně příkazů z knihovny **`VectorAnalysis`** a funkcí **`StreamPlot`**, případně je možné použít funkci **`VectorPlot`**.

a) Znázornění elektrostatického pole elementárního dipólu

Uvažujeme dipól složený ze dvou stejně velkých nábojů opačné polarity. Velikosti nábojů volíme $Q_1 = 1$, $Q_2 = -1$. Náboje jsou umístěny na ose x, přičemž první náboj má souřadnici $x_1 = 1$, $x_2 = -1$. Pole je válcově symetrické, proto hledáme rozložení siločar pouze v jedné rovině, která prochází osou x, konkrétně např. v rovině xy. Pro zjednodušení položíme konstantu ve výrazu pro skalární potenciál rovnu jedné. Skalární potenciál pole dipólu je tedy popsán vztahem $\varphi = \frac{1}{\sqrt{(x-1)^2+y^2}} - \frac{1}{\sqrt{(x+1)^2+y^2}}$.

Následující kód vypočítá vektor elektrické intenzity $\vec{E}$ v kartézských souřadnicích a pole vykreslí pomocí siločar:

In[255]:=
```
Clear[F];
F[x_, y_] := 1/Sqrt[(-1 + x)^2 + y^2] - 1/Sqrt[(1 + x)^2 + y^2];
```

$$F[x\_, y\_] := \frac{1}{\sqrt{(-1+x)^2+y^2}} - \frac{1}{\sqrt{(1+x)^2+y^2}};$$

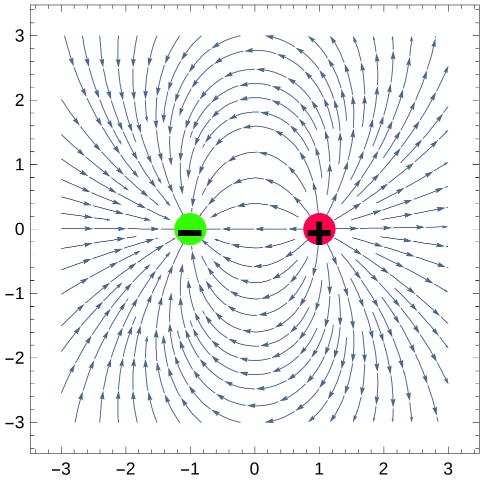
In[257]:=
```
GradientF = -Grad[F[x, y], {x, y, z}]
```

Out[257]=

$$\left\{\frac{-1+x}{\left((-1+x)^2+y^2\right)^{3/2}} - \frac{1+x}{\left((1+x)^2+y^2\right)^{3/2}},\ \frac{y}{\left((-1+x)^2+y^2\right)^{3/2}} - \frac{y}{\left((1+x)^2+y^2\right)^{3/2}},\ 0\right\}$$

In[258]:=
```
StreamPlot[{(-1 + x)/((-1 + x)^2 + y^2)^(3/2) - (1 + x)/((1 + x)^2 + y^2)^(3/2),
  y/((-1 + x)^2 + y^2)^(3/2) - y/((1 + x)^2 + y^2)^(3/2)},
 {x, -3, 3}, {y, -3, 3}, Epilog → {{Hue[0.95`], Disk[{1, 0}, 0.25`],
    GrayLevel[0], Text[Style["+", FontSize → 24, FontWeight → "Bold"], {1, 0}]},
   {Hue[0.3`], Disk[{-1, 0}, 0.25`], GrayLevel[0],
    Text[Style["-", FontSize → 24, FontWeight → "Bold"], {-1, 0}]}}]
```

Out[258]=



b) Znázornění elektrostatického pole axiálního kvadrupólu

Jedná se o soustavu tří bodových nábojů o poměrných velikostech $Q_1 = 1$, $Q_2 = -2$, $Q_3 = 1$, které jsou opět umístěny na ose x v bodech o souřadnicích $x_1 = 1$, $x_2 = 0$, $x_3 = -1$. Skalární potenciál soustavy za předpokladů, uvedených v předchozím odstavci je $\varphi = \frac{1}{\sqrt{(x-1)^2+y^2}} - \frac{2}{\sqrt{x^2+y^2}} + \frac{1}{\sqrt{(x+1)^2+y^2}}$. Postup znázornění pole je stejný, jako v předchozím příkladě:

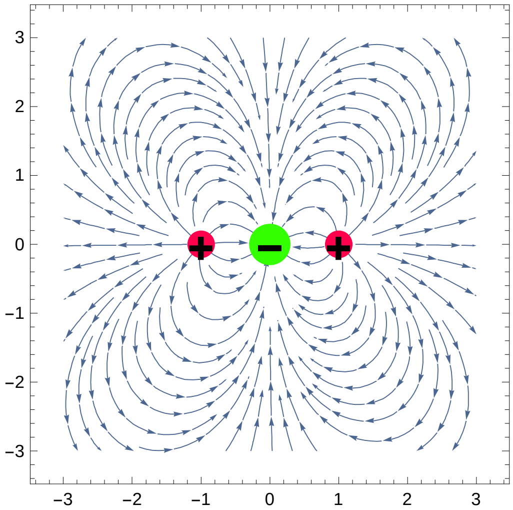
In[259]:=
```
Clear[F1];
F1[x_, y_] = 1/Sqrt[(-1 + x)^2 + y^2] - 2/Sqrt[x^2 + y^2] + 1/Sqrt[(1 + x)^2 + y^2];
```

In[261]:=
```
GradientF1 = -Grad[F1[x, y], {x, y, z}]
```

Out[261]=

$$\left\{\frac{-1+x}{\left((-1+x)^2+y^2\right)^{3/2}} - \frac{2x}{\left(x^2+y^2\right)^{3/2}} + \frac{1+x}{\left((1+x)^2+y^2\right)^{3/2}},\ \frac{y}{\left((-1+x)^2+y^2\right)^{3/2}} - \frac{2y}{\left(x^2+y^2\right)^{3/2}} + \frac{y}{\left((1+x)^2+y^2\right)^{3/2}},\ 0\right\}$$

In[262]:=
```
StreamPlot[{(-1 + x)/((-1 + x)^2 + y^2)^(3/2) - (2 x)/(x^2 + y^2)^(3/2) + (1 + x)/((1 + x)^2 + y^2)^(3/2),
  y/((-1 + x)^2 + y^2)^(3/2) - (2 y)/(x^2 + y^2)^(3/2) + y/((1 + x)^2 + y^2)^(3/2)}, {x, -3, 3},
 {y, -3, 3}, Epilog → {{Hue[0.95`], Disk[{1, 0}, 0.2`], GrayLevel[0],
    Text[Style["+", FontSize → 24, FontWeight → "Bold"], {1, 0}]},
   {Hue[0.95`], Disk[{-1, 0}, 0.2`], GrayLevel[0],
    Text[Style["+", FontSize → 24, FontWeight → "Bold"], {-1, 0}]},
   {Hue[0.3`], Disk[{0, 0}, 0.3`], GrayLevel[0],
    Text[Style["-", FontSize → 24, FontWeight → "Bold"], {0, 0}]}}]
```

Out[262]=


c) Znázornění elektrostatického pole axiálního oktupólu

Elementární axiální oktupól se skládá ze čtyř bodových nábojů o poměrných velikostech $Q_1 = +1$, $Q_2 = -3$, $Q_3 = +3$, $Q_4 = -1$. Náboje jsou opět umístěny na ose x v bodech o souřadnicích $x_1 = 3$, $x_2 = 1$, $x_3 = -1$, $x_4 = -3$. Skalární potenciál této soustavy je $\varphi = \frac{1}{\sqrt{(x-3)^2+y^2}} - \frac{3}{\sqrt{(x-1)^2+y^2}} + \frac{3}{\sqrt{(x+1)^2+y^2}} - \frac{1}{\sqrt{(x+3)^2+y^2}}$. Znázornění elektrostatického pole je následující:

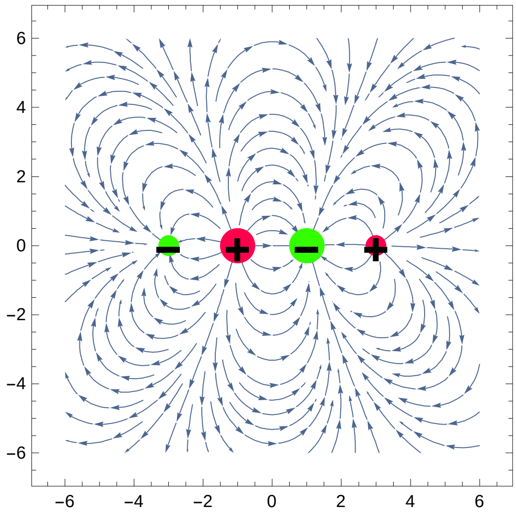
In[263]:=
```
Clear[F2];
F2[x_, y_] = 1/Sqrt[(-3 + x)^2 + y^2] - 3/Sqrt[(-1 + x)^2 + y^2] + 3/Sqrt[(1 + x)^2 + y^2] - 1/Sqrt[(3 + x)^2 + y^2];
```

In[265]:=
```
GradientF2 = -Grad[F2[x, y], {x, y, z}]
```

Out[265]=
$$\left\{\frac{-3+x}{\left((-3+x)^2+y^2\right)^{3/2}} - \frac{3(-1+x)}{\left((-1+x)^2+y^2\right)^{3/2}} + \frac{3(1+x)}{\left((1+x)^2+y^2\right)^{3/2}} - \frac{3+x}{\left((3+x)^2+y^2\right)^{3/2}},\right.$$
$$\left.\frac{y}{\left((-3+x)^2+y^2\right)^{3/2}} - \frac{3y}{\left((-1+x)^2+y^2\right)^{3/2}} + \frac{3y}{\left((1+x)^2+y^2\right)^{3/2}} - \frac{y}{\left((3+x)^2+y^2\right)^{3/2}},\ 0\right\}$$

In[266]:=
```
StreamPlot[{(-3 + x)/((-3 + x)^2 + y^2)^(3/2) - (3 (-1 + x))/((-1 + x)^2 + y^2)^(3/2) + (3 (1 + x))/((1 + x)^2 + y^2)^(3/2) - (3 + x)/((3 + x)^2 + y^2)^(3/2),
  y/((-3 + x)^2 + y^2)^(3/2) - (3 y)/((-1 + x)^2 + y^2)^(3/2) + (3 y)/((1 + x)^2 + y^2)^(3/2) - y/((3 + x)^2 + y^2)^(3/2)},
 {x, -6, 6}, {y, -6, 6}, Epilog → {{Hue[0.95`], Disk[{3, 0}, 0.3`], GrayLevel[0],
    Text[Style["+", FontSize → 24, FontWeight → "Bold"], {3, 0}]},
   {Hue[0.95`], Disk[{-1, 0}, 0.51`], GrayLevel[0],
    Text[Style["+", FontSize → 24, FontWeight → "Bold"], {-1, 0}]},
   {Hue[0.3`], Disk[{-3, 0}, 0.3`], GrayLevel[0],
    Text[Style["-", FontSize → 24, FontWeight → "Bold"], {-3, 0}]},
   {Hue[0.3`], Disk[{1, 0}, 0.51`], GrayLevel[0],
    Text[Style["-", FontSize → 24, FontWeight → "Bold"], {1, 0}]}}]
```

Out[266]=



6.2 Znázornění polí elektrostatických multipólů pomocí postupného vykreslování siločar

Získáme-li výpočtem výrazy pro složky vektoru elektrické intenzity $\vec{E}$ polí elektrostatických multipólů, můžeme je dosadit do obecné rovnice siločar. Tato rovnice má v rovině kartézských souřadnic tvar $\frac{dx}{E_x} = \frac{dy}{E_y}$.

V čitatelích zlomků jsou složky vektorů elementárního posunutí. V obecně křivočarých souřadnicích q_1, q_2 je tedy rovnice siločar vyjádřena vztahem $\frac{l_{q_1}}{E_{q_1}} = \frac{l_{q_2}}{E_{q_2}}$,

kde l_{q_1}, l_{q_2} jsou složky vektoru elementárního posunutí ve směrech křivočarých souřadnic q_1, q_2, které jsou úměrné diferenciálům křivočarých souřadnic přes Laméovy koeficienty h_{q_1}, h_{q_2}. Platí tedy $l_{q_1} = h_{q_1}\,dq_1$, $l_{q_2} = h_{q_2}\,dq_2$. Vyřešíme-li takovou rovnici siločar jako diferenciální rovnici, získáme konkrétní rovnici siločar. Rovnice vždy obsahuje konstantu, za kterou dosazujeme nejlépe zvolené ekvidistatntní hodnoty a pro každou jednu hodnotu vykreslíme právě jednu siločáru. Jako příklady bude uvedeno znázornění pole elementárního dipólu a axiálního oktupólu.

a) Znázornění pole elementárního dipólu

Složky vektoru elektrické intenzity pole elektrostatického dipólu v polárních souřadnicích r, ϑ jsou $E_r = \frac{1}{2\pi\varepsilon_0}\cdot\frac{p\cdot\cos\vartheta}{r^3}$, $E_\vartheta = \frac{1}{4\pi\varepsilon_0}\cdot\frac{p\cdot\sin\vartheta}{r^3}$,

kde p je velikost dipólového momentu. Vyřešením obecné rovnice siločar dostaneme rovnici siločar pole dipólu $r = K\cdot\sin^2\vartheta$.

Následující kód provádí vykreslování jednotlivých siločar pro diskrétní hodnoty konstanty *K*.

In[267]:=
```
X[R_, θ_] := R Cos[θ];
Y[R_, θ_] := R Sin[θ];
pomConst := 0.5;
numOfFieldLines := 10;
Diq1 := 0;
Diq2 := 0;
```

In[273]:=
```
For[Diq1 = 0, Diq1 < numOfFieldLines,
  DiVector1[Diq1] = Table[
    {X[pomConst Diq1 Sin[t]^2, t], Y[pomConst Diq1 Sin[t]^2, t]}, {t, 0, Pi, Pi / 50}],
  Diq1++];
For[Diq2 = 0, Diq2 < numOfFieldLines,
  DiVector2[Diq2] = Table[{X[pomConst Diq2 Sin[t]^2, t],
     Y[pomConst Diq2 Sin[t]^2, t]}, {t, 2 Pi, Pi, -Pi / 50}],
  Diq2++];
```

In[275]:=
```
pomDiVector1 = DiVector1[1];
pomDiVector2 = DiVector2[1];
```

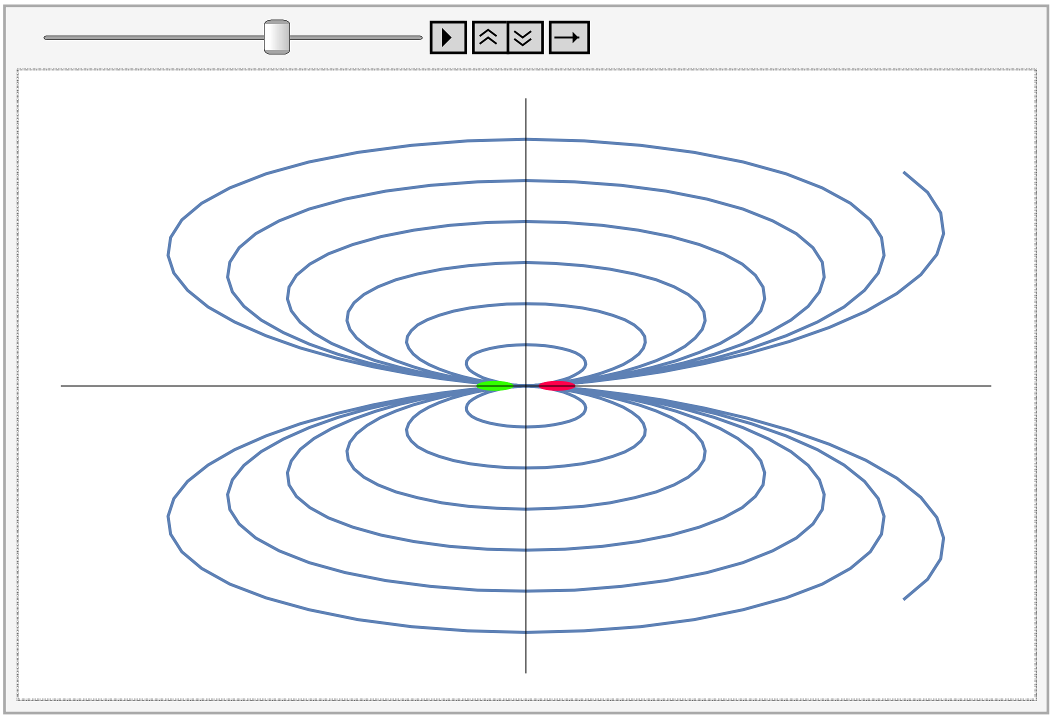
In[277]:=
```
For[DiPom1 = 1, DiPom1 < numOfFieldLines,
  pomDiVector1 = Append[pomDiVector1, DiVector1[DiPom1]],
  DiPom1++];
For[DiPom2 = 1, DiPom2 < numOfFieldLines,
  pomDiVector2 = Append[pomDiVector2, DiVector2[DiPom2]],
  DiPom2++];
```

In[279]:=
```
pomDiVector1 = Flatten[pomDiVector1];
pomDiVector2 = Flatten[pomDiVector2];
pomDiVector1 = Partition[pomDiVector1, 2];
pomDiVector2 = Partition[pomDiVector2, 2];
```

In[283]:=
```
DiMovie = Table[Show[ListPlot[Take[pomDiVector1, DiN],
     PlotRange → {{-1.5, 1.5}, {-3.5, 3.5}}, Joined → True, Ticks → None],
    ListPlot[Take[pomDiVector2, DiN], PlotRange → {{-1.5, 1.5}, {-3.5, 3.5}},
     Joined → True, Ticks → None],
    Graphics[{Hue[0.95], Disk[{1/10, 0}, 0.06], GrayLevel[0]}],
    Graphics[{Hue[0.3], Disk[{-1/10, 0}, 0.06], GrayLevel[0]}],
    ImageSize → {500, 300}], {DiN, 1, Length[pomDiVector2], 2}];
```

In[284]:=
```
ListAnimate[DiMovie, 10, AnimationRunning → False]
```

Out[284]=



b) Znázornění elektrostatického pole axiálního oktupólu

Čistý axiální oktupól, který nevykazuje typové prvky polí jiných multipólů, je popsán v předchozí kapitole. Složky vektoru elektrické intenzity elektrostatického pole axiálního oktupólu v polárních souřadnicích r, ϑ jsou $E_r = \frac{3\,Q l_0^3}{\pi \varepsilon_0} \cdot \frac{5 \cdot \cos^3 \vartheta - 3 \cos\vartheta}{r^5}$, $E_\vartheta = \frac{9\,Q l_0^3}{4\,\pi \varepsilon_0} \cdot \frac{\sin\vartheta (5 \cdot \cos^2 \vartheta - 1)}{r^5}$,

kde Q lze považovat za jednotkový náboj, l_0 je vzdálenost sousedních nábojů v oktupólu. Vyřešením obecné rovnice siločar dostaneme rovnici siločar pole oktupólu ve tvaru $r^3 = K \cdot \sin^2 \vartheta \cdot | 5 \sin^2 \vartheta - 4 |$.

Následujícím programem je nejprve znázorněno pole oktupólu pomocí tří kompletních siločar a poté je ukázáno postupné vykreslování siločar. Na rozdíl od předchozího dipólu jsou všechny uvažované siločáry vykreslovány najednou, tedy vykreslují se najednou křivky pro všechny uvažované hodnoty konstanty K.

In[285]:=
```
X[R_, θ_] := R Cos[θ];
Y[R_, θ_] := R Sin[θ];
```

In[287]:=
```
pomConstOctu := 1 / 2;
numOfFieldLinesOctu := 10;
konstanta = numOfFieldLinesOctu - 2;
ZkOctuInf := ZkOctuInfinitesimal = Pi / 1000;
```

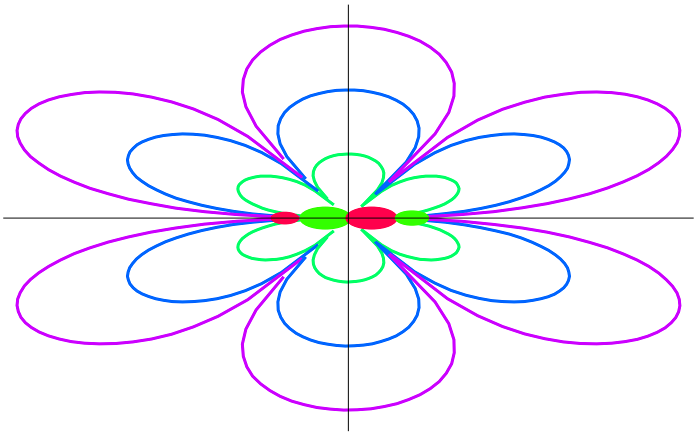
In[291]:=
```
ZkOctuq1 := 0;
ZkOctuq2 := 0;
ZkOctuq3 := 0;
ZkOctuq4 := 0;
ZkOctuq5 := 0;
ZkOctuq6 := 0;
```

```
In[297]:= For[ZkOctuq1 = 0, ZkOctuq1 < numOfFieldLinesOctu, ZkOctuVector1[ZkOctuq1] =
             Table[{X[pomConstOctu ZkOctuq1 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t],
               Y[pomConstOctu ZkOctuq1 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t]},
              {t, ArcSin[2 / Sqrt[5]] - ZkOctuInf, 0, -Pi / 100}], ZkOctuq1++];
          For[ZkOctuq2 = 0, ZkOctuq2 < numOfFieldLinesOctu, ZkOctuVector2[ZkOctuq2] =
             Table[{X[pomConstOctu ZkOctuq2 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t],
               Y[pomConstOctu ZkOctuq2 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t]},
              {t, ArcSin[2 / Sqrt[5]] + ZkOctuInf, Pi - ArcSin[2 / Sqrt[5]] - ZkOctuInf, Pi / 100}],
            ZkOctuq2++];
          For[ZkOctuq3 = 0, ZkOctuq3 < numOfFieldLinesOctu, ZkOctuVector3[ZkOctuq3] =
             Table[{X[pomConstOctu ZkOctuq3 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t],
               Y[pomConstOctu ZkOctuq3 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t]},
              {t, Pi - ZkOctuInf, Pi - ArcSin[2 / Sqrt[5]] + ZkOctuInf, -Pi / 100}], ZkOctuq3++];
          For[ZkOctuq4 = 0, ZkOctuq4 < numOfFieldLinesOctu, ZkOctuVector4[ZkOctuq4] =
             Table[{X[pomConstOctu ZkOctuq4 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t],
               Y[pomConstOctu ZkOctuq4 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t]},
              {t, Pi + ZkOctuInf, Pi + ArcSin[2 / Sqrt[5]] - ZkOctuInf, Pi / 100}], ZkOctuq4++];
          For[ZkOctuq5 = 0, ZkOctuq5 < numOfFieldLinesOctu, ZkOctuVector5[ZkOctuq5] =
             Table[{X[pomConstOctu ZkOctuq5 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t],
               Y[pomConstOctu ZkOctuq5 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t]},
              {t, 2 Pi - ArcSin[2 / Sqrt[5]] - ZkOctuInf,
               Pi + ArcSin[2 / Sqrt[5]] + ZkOctuInf, -Pi / 100}], ZkOctuq5++];
          For[ZkOctuq6 = 0, ZkOctuq6 < numOfFieldLinesOctu, ZkOctuVector6[ZkOctuq6] =
             Table[{X[pomConstOctu ZkOctuq6 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t],
               Y[pomConstOctu ZkOctuq6 (Sin[t]^2 Abs[5 Sin[t]^2 - 4])^(1 / 3), t]},
              {t, 2 Pi - ArcSin[2 / Sqrt[5]] + ZkOctuInf, 2 Pi - ZkOctuInf, Pi / 100}], ZkOctuq6++];

In[303]:= For[i = 31, i ≤ 36, i++, For[j = 1, j ≤ 5, j++,
             ZkOctuVector2[j] = Append[ZkOctuVector2[j], ZkOctuVector2[j][[30]]];
             ZkOctuVector5[j] = Append[ZkOctuVector5[j], ZkOctuVector5[j][[30]]]]];

In[304]:= pomZkOctuVector1 = ZkOctuVector1[1];
          pomZkOctuVector2 = ZkOctuVector2[1];
          pomZkOctuVector3 = ZkOctuVector3[1];
          pomZkOctuVector4 = ZkOctuVector4[1];
          pomZkOctuVector5 = ZkOctuVector5[1];
          pomZkOctuVector6 = ZkOctuVector6[1];

In[310]:= For[ZkOctuPom1 = 1, ZkOctuPom1 < konstanta,
            pomZkOctuVector1 = Append[pomZkOctuVector1, ZkOctuVector1[ZkOctuPom1]],
            ZkOctuPom1++];
          For[ZkOctuPom2 = 1, ZkOctuPom2 < konstanta,
            pomZkOctuVector2 = Append[pomZkOctuVector2, ZkOctuVector2[ZkOctuPom2]],
            ZkOctuPom2++];
          For[ZkOctuPom3 = 1, ZkOctuPom3 < konstanta,
            pomZkOctuVector3 = Append[pomZkOctuVector3, ZkOctuVector3[ZkOctuPom3]],
            ZkOctuPom3++];
          For[ZkOctuPom4 = 1, ZkOctuPom4 < konstanta,
            pomZkOctuVector4 = Append[pomZkOctuVector4, ZkOctuVector4[ZkOctuPom4]],
            ZkOctuPom4++];
          For[ZkOctuPom5 = 1, ZkOctuPom5 < konstanta,
            pomZkOctuVector5 = Append[pomZkOctuVector5, ZkOctuVector5[ZkOctuPom5]],
            ZkOctuPom5++];
          For[ZkOctuPom6 = 1, ZkOctuPom6 < konstanta,
            pomZkOctuVector6 = Append[pomZkOctuVector6, ZkOctuVector6[ZkOctuPom6]],
            ZkOctuPom6++];
```

In[316]:=
```
pomZkOctuVector1 = Flatten[pomZkOctuVector1];
pomZkOctuVector2 = Flatten[pomZkOctuVector2];
pomZkOctuVector3 = Flatten[pomZkOctuVector3];
pomZkOctuVector4 = Flatten[pomZkOctuVector4];
pomZkOctuVector5 = Flatten[pomZkOctuVector5];
pomZkOctuVector6 = Flatten[pomZkOctuVector6];
pomZkOctuVector1 = Partition[pomZkOctuVector1, 2];
pomZkOctuVector2 = Partition[pomZkOctuVector2, 2];
pomZkOctuVector3 = Partition[pomZkOctuVector3, 2];
pomZkOctuVector4 = Partition[pomZkOctuVector4, 2];
pomZkOctuVector5 = Partition[pomZkOctuVector5, 2];
pomZkOctuVector6 = Partition[pomZkOctuVector6, 2];
```

In[328]:=
```
OctuPicture = Show[ListPlot[{
    ZkOctuVector1[1], ZkOctuVector1[2], ZkOctuVector1[3], ZkOctuVector2[1],
    ZkOctuVector2[2], ZkOctuVector2[3], ZkOctuVector3[1], ZkOctuVector3[2],
    ZkOctuVector3[3], ZkOctuVector4[1], ZkOctuVector4[2], ZkOctuVector4[3],
    ZkOctuVector5[1], ZkOctuVector5[2], ZkOctuVector5[3], ZkOctuVector6[1],
    ZkOctuVector6[2], ZkOctuVector6[3]},
   Joined → True, PlotStyle → {{Hue[0.4`]}, {Hue[0.6`]}, {Hue[0.8`]}}],
  Graphics[{Hue[0.95`], Disk[{-0.22`, 0}, 0.05`], GrayLevel[0]}],
  Graphics[{Hue[0.3`], Disk[{-0.08`, 0}, 0.09`], GrayLevel[0]}],
  Graphics[{Hue[0.95`], Disk[{0.08`, 0}, 0.09`], GrayLevel[0]}],
  Graphics[{Hue[0.3`], Disk[{0.22`, 0}, 0.06`], GrayLevel[0]}], Ticks → None]
```

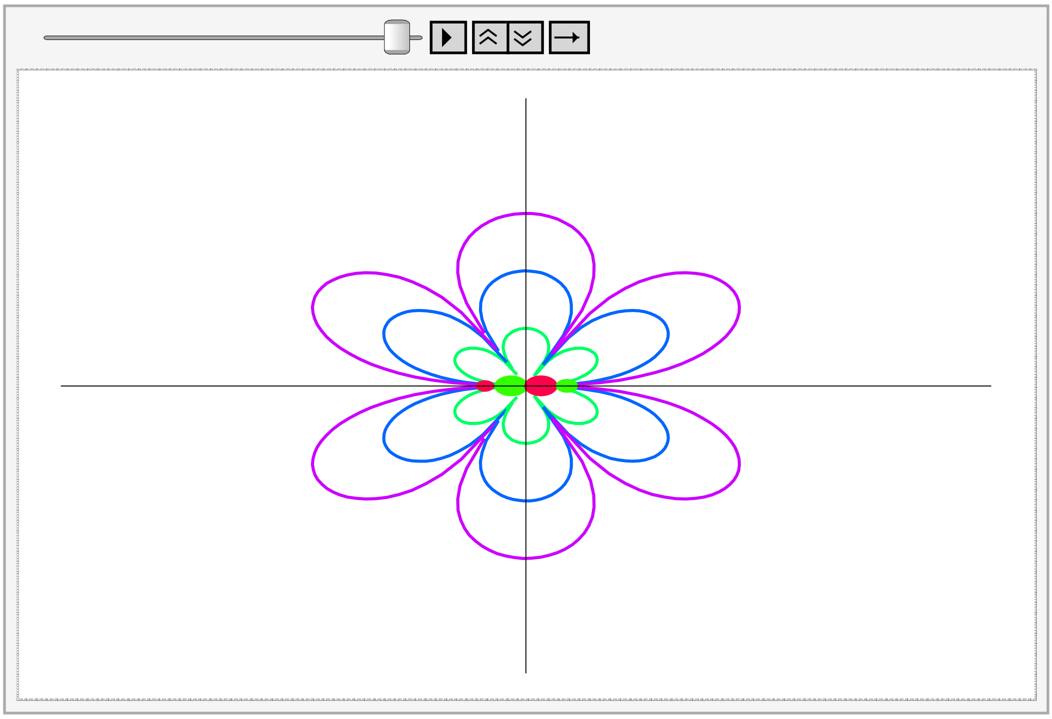
Out[328]=



In[329]:=
```
ZKOctuN2 := 0;
OctuMovie = Table[Show[ListPlot[{
      Take[ZkOctuVector1[1], ZkOctuN2], Take[ZkOctuVector1[2], ZkOctuN2],
      Take[ZkOctuVector1[3], ZkOctuN2], Take[ZkOctuVector2[1], ZkOctuN2],
      Take[ZkOctuVector2[2], ZkOctuN2], Take[ZkOctuVector2[3], ZkOctuN2],
      Take[ZkOctuVector3[1], ZkOctuN2], Take[ZkOctuVector3[2], ZkOctuN2],
      Take[ZkOctuVector3[3], ZkOctuN2], Take[ZkOctuVector4[1], ZkOctuN2],
      Take[ZkOctuVector4[2], ZkOctuN2], Take[ZkOctuVector4[3], ZkOctuN2],
      Take[ZkOctuVector5[1], ZkOctuN2], Take[ZkOctuVector5[2], ZkOctuN2],
      Take[ZkOctuVector5[3], ZkOctuN2], Take[ZkOctuVector6[1], ZkOctuN2],
      Take[ZkOctuVector6[2], ZkOctuN2], Take[ZkOctuVector6[3], ZkOctuN2]},
     Joined → True, PlotStyle → {{Hue[0.4`]}, {Hue[0.6`]}, {Hue[0.8`]}},
     PlotRange → {{-2.5`, 2.5`}, {-2.5`, 2.5`}}],
    Graphics[{Hue[0.95`], Disk[{-0.22`, 0}, 0.05`], GrayLevel[0]}],
    Graphics[{Hue[0.3`], Disk[{-0.08`, 0}, 0.09`], GrayLevel[0]}],
    Graphics[{Hue[0.95`], Disk[{0.08`, 0}, 0.09`], GrayLevel[0]}],
    Graphics[{Hue[0.3`], Disk[{0.22`, 0}, 0.06`], GrayLevel[0]}],
    Ticks → None, ImageSize → {500, 300}], {ZkOctuN2, 1, 36, 1}];
```

In[331]:=
```
ListAnimate[OctuMovie, 4, AnimationRunning → False]
```

Out[331]=



6.3 Vizualizace ploch fázových rychlostí krystalů

Plocha fázových rychlostí je plocha, na kterou se za jednotku času z jednoho bodu ve všech směrech rozšíří v elektricky anizotropním prostředí elektromagnetická vlna. Odvození rovnice této plochy je relativně snadné. Jedná se o dvojlistou plochu šestého stupně, jejíž řezy souřadnými rovinami se rozpadají na kružnice a ovály čtvrtého stupně.

Rovnice plochy fázových rychlostí v ortogonálním pravotočivém kartézském souřadném systému má tvar

$$v_x^2\left(v_2^2-v_x^2-v_y^2-v_z^2\right)\left(v_3^2-v_x^2-v_y^2-v_z^2\right)+ ,$$
$$v_y^2\left(v_1^2-v_x^2-v_y^2-v_z^2\right)\left(v_3^2-v_x^2-v_y^2-v_z^2\right)+v_z^2\left(v_1^2-v_x^2-v_y^2-v_z^2\right)\left(v_2^2-v_x^2-v_y^2-v_z^2\right)=0$$

kde v_x, v_y, v_z jsou složky vektoru fázové rychlosti, orientovaného ve zvoleném směru a v_1, v_2, v_3 jsou hlavní fázové rychlosti vázané s hlavními permitivitami ε_1, ε_2, ε_3 v systému hlavních os.

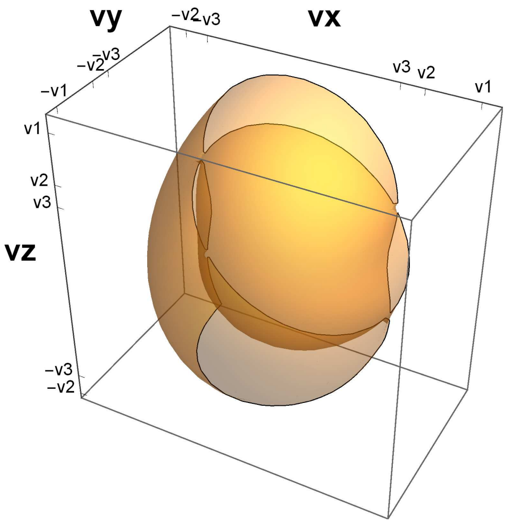
Problém je s představou, jak tato plocha vypadá. S využitím software *Mathematica* může být plocha fázových rychlostí znázorněna následujícím způsobem. Jsou prezentovány 3 ukázky, lišící se omezením prostoru, ve kterém je plocha zobrazena. Pomocí otáčení zobrazených 3D grafů je možné si plochu podrobně prohlédnout.

Pozn. hodnoty hlavních fázových rychlostí v1, v2, v3 mohou být voleny libovolně, v souladu s teorií předpokládáme, že v1 > v2 > v3.

In[332]:=
```
v1 = 3;
v2 = 2.1;
v3 = 1.7;
```

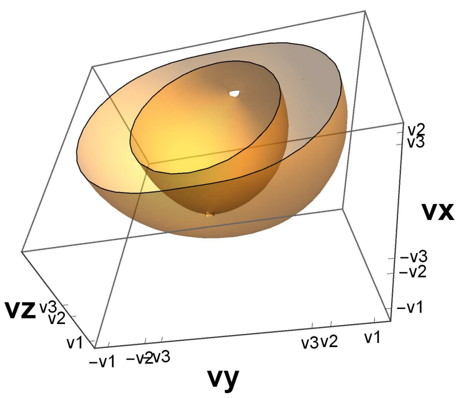
In[335]:=
```
ContourPlot3D[vx^2 (v2^2 - vx^2 - vy^2 - vz^2) (v3^2 - vx^2 - vy^2 - vz^2) +
   vy^2 (v1^2 - vx^2 - vy^2 - vz^2) (v3^2 - vx^2 - vy^2 - vz^2) +
   vz^2 (v1^2 - vx^2 - vy^2 - vz^2) (v2^2 - vx^2 - vy^2 - vz^2) == 0,
  {vx, -v2 - 0.2, v1 + 0.2}, {vy, -v1 - 0.2, 0}, {vz, -v2, v1 + 0.2},
  BoxRatios → Automatic, ContourStyle → Opacity[0.45], Contours → Automatic,
  Mesh → None, Axes → True, AxesLabel → {Style["vx", FontSize → 16, FontWeight → "Bold"],
   Style["vy", FontSize → 16, FontWeight → "Bold"],
   Style["vz", FontSize → 16, FontWeight → "Bold"]}, Ticks → {
   {{-v1, "-v1"}, {-v2, "-v2"}, {-v3, "-v3"}, {v3, "v3"}, {v2, "v2"}, {v1, "v1"}},
   {{-v1, "-v1"}, {-v2, "-v2"}, {-v3, "-v3"}, {v3, "v3"}, {v2, "v2"}, {v1, "v1"}},
   {{-v1, "-v1"}, {-v2, "-v2"}, {-v3, "-v3"}, {v3, "v3"}, {v2, "v2"}, {v1, "v1"}}}]
```

Out[335]=



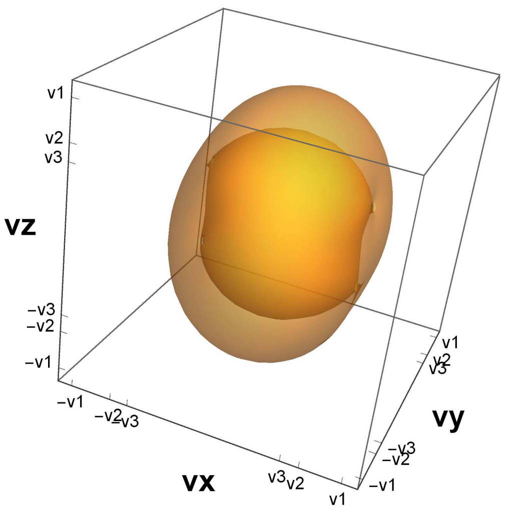
In[336]:=
```
ContourPlot3D[vx^2 (v2^2 - vx^2 - vy^2 - vz^2) (v3^2 - vx^2 - vy^2 - vz^2) +
   vy^2 (v1^2 - vx^2 - vy^2 - vz^2) (v3^2 - vx^2 - vy^2 - vz^2) +
   vz^2 (v1^2 - vx^2 - vy^2 - vz^2) (v2^2 - vx^2 - vy^2 - vz^2) == 0,
  {vx, -v1 - 0.2, v2 + 0.2}, {vy, -v1 - 0.2, v1 + 0.2}, {vz, 0, v1 + 0.2},
  BoxRatios → Automatic, ContourStyle → Opacity[0.45], Contours → Automatic, Mesh → None,
  Axes → True, AxesLabel → {Style["vx", FontSize → 16, FontWeight → "Bold"],
   Style["vy", FontSize → 16, FontWeight → "Bold"],
   Style["vz", FontSize → 16, FontWeight → "Bold"]}, Ticks → {
   {{-v1, "-v1"}, {-v2, "-v2"}, {-v3, "-v3"}, {v3, "v3"}, {v2, "v2"}, {v1, "v1"}},
   {{-v1, "-v1"}, {-v2, "-v2"}, {-v3, "-v3"}, {v3, "v3"}, {v2, "v2"}, {v1, "v1"}},
   {{-v1, "-v1"}, {-v2, "-v2"}, {-v3, "-v3"}, {v3, "v3"}, {v2, "v2"}, {v1, "v1"}}}]
```

Out[336]=



In[337]:=
```
ContourPlot3D[vx^2 (v2^2 - vx^2 - vy^2 - vz^2) (v3^2 - vx^2 - vy^2 - vz^2) +
   vy^2 (v1^2 - vx^2 - vy^2 - vz^2) (v3^2 - vx^2 - vy^2 - vz^2) +
   vz^2 (v1^2 - vx^2 - vy^2 - vz^2) (v2^2 - vx^2 - vy^2 - vz^2) == 0,
 {vx, -v1 - 0.2, v1 + 0.2}, {vy, -v1 - 0.2, v1 + 0.2}, {vz, -v1 - 0.2, v1 + 0.2},
 BoxRatios → Automatic, ContourStyle → Opacity[0.45], Contours → Automatic,
 Mesh → None, Axes → True, AxesLabel → {Style["vx", FontSize → 16, FontWeight → "Bold"],
   Style["vy", FontSize → 16, FontWeight → "Bold"],
   Style["vz", FontSize → 16, FontWeight → "Bold"]}, Ticks → {
   {{-v1, "-v1"}, {-v2, "-v2"}, {-v3, "-v3"}, {v3, "v3"}, {v2, "v2"}, {v1, "v1"}},
   {{-v1, "-v1"}, {-v2, "-v2"}, {-v3, "-v3"}, {v3, "v3"}, {v2, "v2"}, {v1, "v1"}},
   {{-v1, "-v1"}, {-v2, "-v2"}, {-v3, "-v3"}, {v3, "v3"}, {v2, "v2"}, {v1, "v1"}}}]
```

Out[337]=



6.4 Odraznost a propustnost rozhraní dvou homogenních izotropních dielektrik

Jevy na rozhraní jsou popsány Fresnelovými vzorci. Tyto vzorce umožňují výpočet koeficientů odraznosti a propustnosti jako poměrů amplitud vektoru elektrické intenzity vlny odražené nebo lomené k amplitudě vlny dopadající pro lineárně polarizované vlny, kdy vektor elektrické intenzity leží buď v rovině dopadu nebo v rovině k ní kolmé. Vzorce jsou obvykle uváděny ve tvaru

$r^{\parallel} = -\frac{\mathrm{tg}(\alpha_1-\alpha_2)}{\mathrm{tg}(\alpha_1+\alpha_2)}$ pro koeficient odraznosti v rovině dopadu,

$r^{\perp} = -\frac{\sin(\alpha_1-\alpha_2)}{\sin(\alpha_1+\alpha_2)}$ pro koeficient odraznosti v rovině kolmé k rovině dopadu,

$t^{\parallel} = \frac{2\sin\alpha_2\cos\alpha_1}{\sin(\alpha_1+\alpha_2)\cos(\alpha_1-\alpha_2)}$ pro koeficient propustnosti v rovině dopadu a

$t^{\perp} = \frac{2\sin\alpha_2\cos\alpha_1}{\sin(\alpha_1+\alpha_2)}$ pro koeficient propustnosti v rovině kolmé k rovině dopadu,

kde α_1 je úhel dopadu na rozhraní a α_2 je úhel lomu.

Vychází-li koeficienty odraznosti kladné, pak odražená vlna je ve fázi s vlnou dopadající, v opačném případě dochází k odrazu s opačnou fází. Lomená vlna je vždy ve fázi s vlnou dopadající.

S koeficienty odraznosti a propustnosti souvisí odraznost R a propustnost T jako energetické veličiny – popisují poměr energií přenášených vlnou odraženou nebo lomenou k energii vlny dopadající. Odraznosti, související s koeficienty odraznosti, pak jsou definovány vztahy

$R^{\parallel} = (r^{\parallel})^2$, $R^{\perp} = (r^{\perp})^2$, $T^{\parallel} = \frac{n_2\cos\alpha_2}{n_1\cos\alpha_1}(t^{\parallel})^2$, $T^{\perp} = \frac{n_2\cos\alpha_2}{n_1\cos\alpha_1}(t^{\perp})^2$, kde n_1 je index lomu prostředí, ze kterého vlna na rozhraní dopadá, n_2 je index lomu prostředí, do kterého se láme.

Pomocí prostředků Mathematiky můžeme pro názornost vykreslovat závislosti uvedených veličin např. na úhlu dopadu pro zvolená prostředí (indexy lomu), případně pomocí animací zároveň vykreslovat závislosti na dalších veličinách.

Jako příklad uvedeme rozhraní prostředí opticky hustšího a řidšího, tedy $n_1 > n_2$. Předpokládejme, že se jedná o reálné rozhraní optického skla a vzduchu, tedy $n_2 = 1$. Je známo, že pro úhel dopadu $\alpha_1 > \alpha_m$, kde α_m je mezní úhel, dochází k totálnímu odrazu.

V tomto případě jsou koeficienty odraznosti a propustnosti dány výrazy

$r^{\parallel} = -e^{i2\psi^{\parallel}}$, kde $\psi^{\parallel} = \operatorname{arctg} \frac{n_1 \sqrt{n_1^2 \sin^2 \alpha_1 - n_2^2}}{n_2^2 \cos\alpha_1}$ a

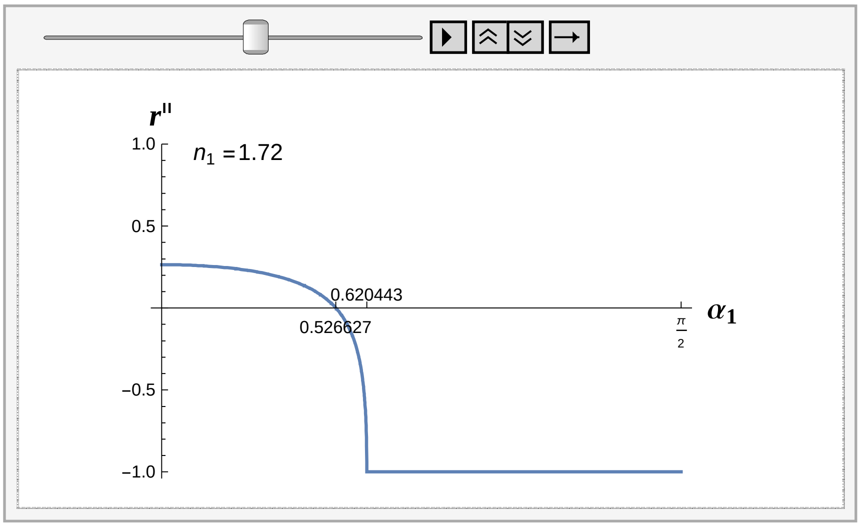
$r^{\perp} = e^{i2\psi^{\perp}}$, kde $\psi^{\perp} = \operatorname{arctg} \frac{\sqrt{n_1^2 \sin^2 \alpha_1 - n_2^2}}{n_1 \cos\alpha_1}$.

Významnou chararteristikou jevů na rozhraní je Brewsterův úhel α_B, pro který platí $\operatorname{tg}\alpha_B = \frac{n_2}{n_1}$. Při dopadu elektromagnetické vlny pod Brewsterovým úhlem je odražená vlna plně lineárně polarizovaná v rovině kolmé k rovině dopadu, protože v tomto případě $r^{\parallel} = 0$.

Ukážeme vykreslení závislostí koeficientů odraznosti a odrazností rozhraní v závislosti na úhlu dopadu $\alpha_{1,}$, přičemž dalším parametrem této závislosti bude změna indexu lomu skla (prvního prostředí) v obvyklém intervalu rozmezí indexů lomu optických skel, tedy od hodnoty $n_1 = 1.4$ do $n_1 = 1.96$. Tato závislost je vyjádřena pomocí animací. Na ose úhlu dopadu jsou pro každou průběžnou hodnotu indexu lomu skla uvedeny hodnoty Brewsterova úhlu α_B (hodnota vlevo) a hodnota mezního úhlu α_m. Pro úhly dopadu větší, než mezní úhel, budou koeficienty odraznosti a propustnosti popsány jen svou amplitudou, tedy hodnotami $r^{\parallel} = -1$, $r^{\perp} = 1$. Vlevo nahoře v poli grafu jsou uvedeny okamžité hodnoty indexu lomu n_1.

```
In[338]:= n2 = 1;
r1 = Table[Show[
    Plot[{ -Tan[a1 - ArcSin[(n1 Sin[a1])/n2]] / Tan[a1 + ArcSin[(n1 Sin[a1])/n2]]   0 ≤ a1 ≤ ArcSin[n2/n1]
           -1                                                                     ArcSin[n2/n1] < a1 ≤ π/2 ,
     {a1, 0, π/2},
     AxesLabel → {
       Style["α1", FontFamily → "Times", FontSize → 16, FontWeight → "Bold"],
       Style["r‖", FontFamily → "Times", FontSize → 16, FontWeight → "Bold"]},
     PlotStyle → {Thickness[0.006]},
     Ticks → {{{π/2, "π/2"}, {ArcSin[n2/n1], ""}, {ArcTan[n2/n1], ArcTan[n2/n1]}}, Automatic},
     PlotRange → {-1.04, 1}, Exclusions → None],
    Graphics[{Text[ArcSin[n2/n1], {ArcSin[n2/n1], 0.08`}]}],
    Graphics[{Text[Style["n1 =     ", 12], {0.2, 0.95}]}],
    Graphics[{Text[Style[n1, 12], {0.3, 0.95}]}],
    ImageSize → {400, 200}], {n1, 1.4, 1.96, 0.02}];

In[340]:= ListAnimate[r1, 3, AnimationRunning → False]
```

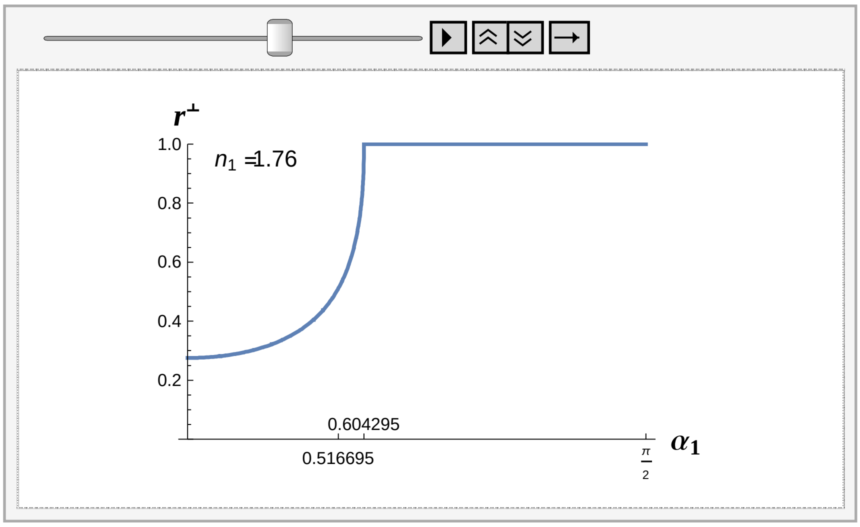
Out[340]=



In[341]:=
```
n2 = 1;
r2 = Table[Show[
    Plot[{ -Sin[a1 - ArcSin[n1 Sin[a1]/n2]] / Sin[a1 + ArcSin[n1 Sin[a1]/n2]]   0 ≤ a1 ≤ ArcSin[n2/n1]
           1                                                                   ArcSin[n2/n1] < a1 ≤ π/2 ,
     {a1, 0, π/2},
     AxesLabel → {
       Style["α1", FontFamily → "Times", FontSize → 16, FontWeight → "Bold"],
       Style["r⊥", FontFamily → "Times", FontSize → 16, FontWeight → "Bold"]},
     PlotStyle → {Thickness[0.008`]},
     Ticks → {{{π/2, "π/2"}, {ArcSin[n2/n1], ""}, {ArcTan[n2/n1], ArcTan[n2/n1]}}, Automatic},
     PlotRange → {0, 1}, Exclusions → None],
    Graphics[{Text[ArcSin[n2/n1], {ArcSin[n2/n1], 0.05`}]}],
    Graphics[{Text[Style["n1 =    ", 12], {0.2, 0.95}]}],
    Graphics[{Text[Style[n1, 12], {0.3, 0.95}]}],
    ImageSize → {400, 200}], {n1, 1.4, 1.96, 0.02}];
```

In[343]:=
```
ListAnimate[r2, 3, AnimationRunning → False]
```

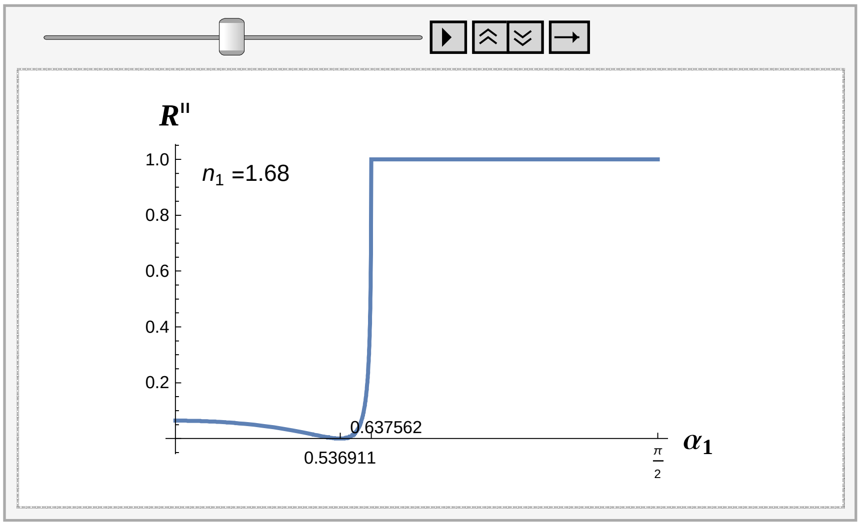
Out[343]=



In[344]:=
```
n2 = 1;
RR1 = Table[Show[
    Plot[{ (-Tan[a1 - ArcSin[n1 Sin[a1]/n2]] / Tan[a1 + ArcSin[n1 Sin[a1]/n2]])^2   0 ≤ a1 ≤ ArcSin[n2/n1]
           1                                                                      ArcSin[n2/n1] < a1 ≤ π/2 ,
      {a1, 0, π/2},
      AxesLabel → {
        Style["α1", FontFamily → "Times", FontSize → 16, FontWeight → "Bold"],
        Style["R''", FontFamily → "Times", FontSize → 16, FontWeight → "Bold"]},
      PlotStyle → {Thickness[0.008`]},
      Ticks → {{{π/2, "π/2"}, {ArcSin[n2/n1], ""}, {ArcTan[n2/n1], ArcTan[n2/n1]}}, Automatic},
      Exclusions → None],
    Graphics[{Text[ArcSin[n2/n1], {ArcSin[n2/n1] + 0.05, 0.04}]}],
    Graphics[{Text[Style["n1 =     ", 12], {0.2, 0.95}]}],
    Graphics[{Text[Style[n1, 12], {0.3, 0.95}]}],
    ImageSize → {400, 200}], {n1, 1.4, 1.96, 0.02}];
```

In[346]:=
```
ListAnimate[RR1, 3, AnimationRunning → False]
```

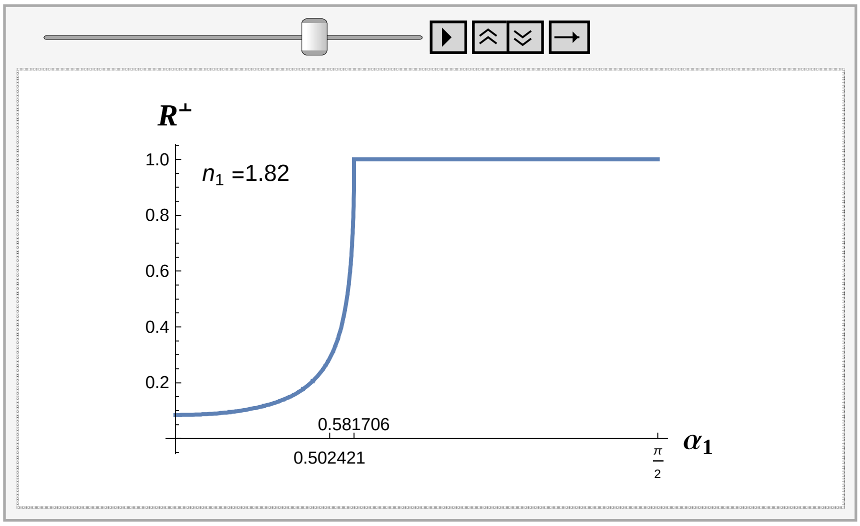
Out[346]=



In[347]:=
```
n2 = 1;
RR2 = Table[Show[
    Plot[{ (-Sin[a1 - ArcSin[n1 Sin[a1]/n2]] / Sin[a1 + ArcSin[n1 Sin[a1]/n2]])^2   0 ≤ a1 ≤ ArcSin[n2/n1]
           1                                                                      ArcSin[n2/n1] < a1 ≤ π/2 ,
      {a1, 0, π/2},
      AxesLabel → {
        Style["α1", FontFamily → "Times", FontSize → 16, FontWeight → "Bold"],
        Style["R⊥", FontFamily → "Times", FontSize → 16, FontWeight → "Bold"]},
      PlotStyle → {Thickness[0.008`]},
      Ticks → {{{π/2, "π/2"}, {ArcSin[n2/n1], ""}, {ArcTan[n2/n1], ArcTan[n2/n1]}}, Automatic},
      Exclusions → None],
    Graphics[{Text[ArcSin[n2/n1], {ArcSin[n2/n1], 0.05`}]}],
    Graphics[{Text[Style["n1 =     ", 12], {0.2, 0.95}]}],
    Graphics[{Text[Style[n1, 12], {0.3, 0.95}]}],
    ImageSize → {400, 200}], {n1, 1.4, 1.96, 0.02}];
```

In[349]:=
```
ListAnimate[RR2, 3, AnimationRunning → False]
```

Out[349]=



7 Základy moderní fyziky

7.1 Planckův vyzařovací zákon

Planckův zákon záření černého tělesa objevený roku 1900 nejen umožnil správný popis záření tohoto fyzikálního modelu, ale díky předpokladu kvantování energie otevřel cestu ke kvantové fyzice a fyzice mikrosvěta. Zákon vyjadřuje závislost *spektrální hustoty intenzity vyzařování* H_λ na vlnové délce λ (popř. na frekvenci) ve tvaru:

In[350]:=
```
Hλ = 2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1)
```

Out[350]=

$$\frac{2\, c^2\, h\, \pi}{\left(-1 + e^{\frac{c\, h}{k\, T\, \lambda}}\right) \lambda^5}$$

Integrací přes všechny vlnové délky od 0 do ∞ získáme celkovou vyzářenou energii, tedy Stefanův – Boltzmannův zákon (pro výpočet integrálu je vhodné přidat předpoklad kladných hodnot absolutní teploty a fyzikálních konstant – doporučujeme zkusit vyčíslení integrálu bez této podmínky).

In[351]:=
```
Integrate[Hλ, {λ, 0, Infinity}, Assumptions → {T > 0, c > 0, k > 0, h > 0}]
```

Out[351]=

$$\frac{2\, k^4\, \pi^5\, T^4}{15\, c^2\, h^3}$$

Pokud by nás zajímala číselná hodnota Stefanovy-Boltzmannovy konstanty v jednotkách $W \cdot m^{-2} \cdot K^{-4}$ (výše vidíme její vyjádření pomocí jiných konstant), dosadíme navíc číselné hodnoty pro rychlost světla $c = 2.998 \cdot 10^8\, m \cdot s^{-1}$, Boltzmannovu konstantu $k = 1.381 \cdot 10^{-23}\, J \cdot K^{-1}$ a Planckovu konstantu $h = 6.626 \cdot 10^{-34}\, J \cdot s$ a vydělíme čtvrtou mocninou absolutní teploty; vychází známý výsledek:

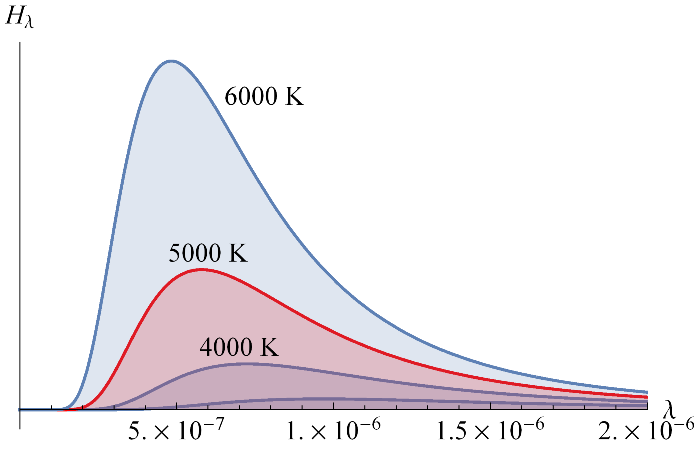
In[352]:=
```
σ = Integrate[Subscript[H, λ], {λ, 0, Infinity}, Assumptions → T > 0] / T^4 /.
  {c → 2.997925 × 10^8, k → 1.3806488 / 10^23, h → 6.626068 / 10^34};
```

Závislost spektrální intenzity vyzařování na vlnové délce (popř. na frekvenci) má typický průběh, který můžeme vykreslit a zkoumat jeho závislost na teplotě. Pro ilustraci volíme teploty 3000 K, 4000 K, 5000 K a 6000 K, které řádově odpovídají teplotě povrchu hvězd slunečního typu. Jednotlivé grafy vykreslíme najednou pomocí příkazu “`Show`”.

In[353]:=
```
graf1 = Plot[2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1) /. {c → 2.997925 * 10^8,
      k → 1.3806488 / 10^23, h → 6.626068 / 10^34, T → 3000} // N, {λ, 0, 10^-5},
   PlotRange → {{0, 0.000002}, All}, Filling → Axis, AxesLabel → {"λ", "Hλ"},
   Ticks → {Automatic, None}, BaseStyle → {FontFamily → "Times", FontSize → 14}];
graf2 = Plot[2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1) /. {c → 2.997925 * 10^8,
      k → 1.3806488 / 10^23, h → 6.626068 / 10^34, T → 4000} // N, {λ, 0, 10^-5},
   PlotRange → {{0, 0.000002}, All}, Filling → Axis, AxesLabel → {"λ", "Hλ"},
   Ticks -> {Automatic, None}, BaseStyle → {FontFamily → "Times", FontSize → 14}];
graf3 = Plot[2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1) /. {c → 2.997925 * 10^8,
      k → 1.3806488 / 10^23, h → 6.626068 / 10^34, T → 5000} // N,
   {λ, 0, 10^-5}, PlotRange → {{0, 0.000002}, All}, PlotStyle → {Red},
   Filling → Axis, AxesLabel → {"λ", "Hλ"}, Ticks -> {Automatic, None},
   BaseStyle → {FontFamily → "Times", FontSize → 14}];
graf4 = Plot[2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1) /. {c → 2.997925 * 10^8,
      k → 1.3806488 / 10^23, h → 6.626068 / 10^34, T → 6000} // N, {λ, 0, 10^-5},
   PlotRange → {{0, 0.000002}, All}, Filling → Axis, AxesLabel → {"λ", "Hλ"},
   Ticks → {Automatic, None}, BaseStyle → {FontFamily → "Times", FontSize → 14}];
```

In[357]:=
```
Show[graf1, graf2, graf3, graf4, Epilog → {Text["4000 K", {7*^-7, 1.8*^13}],
   Text["5000 K", {6*^-7, 4.5*^13}], Text["6000 K", {7.8*^-7, 9*^13}]}]
```

Out[357]=



Vidíme, že vlnová délka, pro kterou má spektrální hustota intenzity vyzařování maximum se s rostoucí teplotou posouvá doleva ke kratším vlnovým délkám. Odpovídající závislost se nazývá Wienův posunovací zákon a odvodíme jej řešením transcendentní rovnice. Polohu extrému najdeme pomocí derivace podle vlnové délky

In[358]:=
```
D[Hλ, λ]
```

Out[358]=

$$\frac{2\,c^3\,e^{\frac{c\,h}{k\,T\,\lambda}}\,h^2\,\pi}{\left(-1+e^{\frac{c\,h}{k\,T\,\lambda}}\right)^2 k\,T\,\lambda^7}-\frac{10\,c^2\,h\,\pi}{\left(-1+e^{\frac{c\,h}{k\,T\,\lambda}}\right)\lambda^6}$$

a před řešením rovnice ještě zjednodušíme:

In[359]:=
```
pom = Factor[%]
```

Out[359]=

$$\frac{2\,c^2\,h\,\pi\left(c\,e^{\frac{c\,h}{k\,T\,\lambda}}\,h+5\,k\,T\,\lambda-5\,e^{\frac{c\,h}{k\,T\,\lambda}}\,k\,T\,\lambda\right)}{\left(-1+e^{\frac{c\,h}{k\,T\,\lambda}}\right)^2 k\,T\,\lambda^7}$$

Pro položení první derivace rovné 0 je postačující uvažovat dále pouze čitatele zlomku a řešení získané transcendentní rovnice hledat pomocí příkazu `FindRoot`; předtím však provedeme substituci zavedením nové neznámé $x = \lambda T$. Postupně získáváme

In[360]:=
```
rovnice = Expand[Numerator[pom] /.
   {c → 2.997925 * 10^8, k → 1.3806488 / 10^23, h → 6.626068 / 10^34, λ → x / T}]
FindRoot[rovnice, {x, 0.0025}]
```

Out[360]= $7.43283\times10^{-41}\,e^{0.0143878/x}+2.58304\times10^{-38}\,x-2.58304\times10^{-38}\,e^{0.0143878/x}\,x$

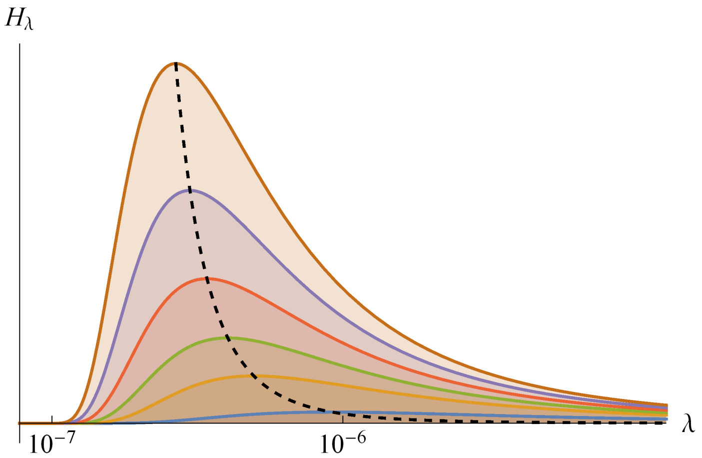
Out[361]= {x → 0.00289777}

Pro vlnovou délku λ_m, jíž při dané teplotě T odpovídá maximum spektrální hustoty vyzařování tak získáváme *Wienův posunovací zákon* $\lambda_m T = 2.898\ \mathrm{mK\cdot m}$. Nyní můžeme předcházející grafy doplnit o maxima jednotlivých funkčních závislostí.

In[362]:=
```
vgraf1 = Plot[{2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1) /.
     {c → 2.997925 * 10^8, k → 1.3806488 / 10^23, h → 6.626068 / 10^34, T → 3000},
    2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1) /. {c → 2.997925 * 10^8,
      k → 1.3806488 / 10^23, h → 6.626068 / 10^34, T → 4000},
    2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1) /. {c → 2.997925 * 10^8,
      k → 1.3806488 / 10^23, h → 6.626068 / 10^34, T → 4500},
    2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1) /. {c → 2.997925 * 10^8,
      k → 1.3806488 / 10^23, h → 6.626068 / 10^34, T → 5000},
    2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1) /. {c → 2.997925 * 10^8,
      k → 1.3806488 / 10^23, h → 6.626068 / 10^34, T → 5500},
    2 Pi h c^2 / λ^5 / (Exp[h c / k / λ / T] - 1) /. {c → 2.997925 * 10^8,
      k → 1.3806488 / 10^23, h → 6.626068 / 10^34, T → 6000}}, {λ, 0, 2*^-6},
   PlotRange → {{0, 0.000002}, All}, Filling → Axis, AxesLabel → {"λ", "Hλ"},
   Ticks → {Table[{10^i, Superscript[10, i]}, {i, -7, -5}], None},
   BaseStyle → {FontFamily → "Times", FontSize → 14}];
vgraf2 = Plot[2 Pi h c^2 / λ^5 / (Exp[h * c / k / 0.0028977] - 1) /.
    {c -> 2.997925 * 10^8, k → 1.3806488 / 10^23, h → 6.626068 / 10^34},
   {λ, 1*^-7, 2*^-6}, PlotRange → {{0, 0.000002}, {0, 1*^14}}, AxesLabel → {"λ", "Hλ"},
   Ticks → {Table[{10^i, Superscript[10, i]}, {i, -7, -5}], None},
   BaseStyle → {FontFamily → "Times", FontSize → 14}, PlotStyle → {Black, Dashed}];
```

In[364]:=
```
Show[vgraf1, vgraf2]
```

Out[364]=

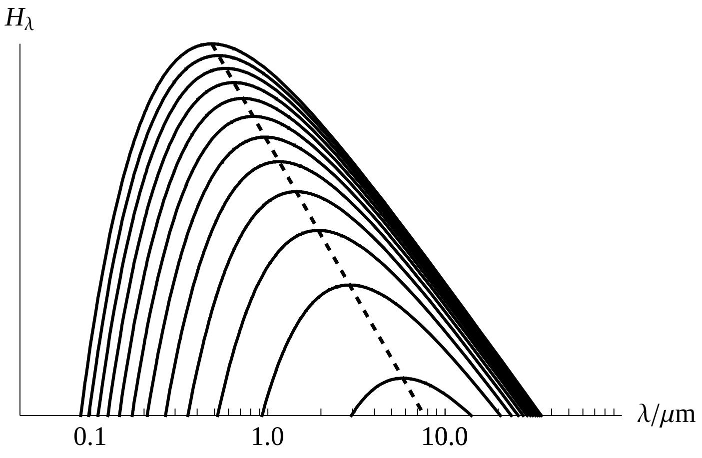


V řadě publikací je při grafickém znázornění využito logaritmické škály os, rozsah teplot volíme od 500 K do 6000 K. Předtím ještě definujeme popisky osy vlnových délek.

In[365]:=
```
popisky = Table[{j * 10^-7 // N, If[Element[Log10[j * 10^-7], Integers],
      NumberForm[N[j / 10, 3], {3, 1}, NumberPoint → "."], ""]}, {j, Join[
      Table[i, {i, 1, 9, 1}], Table[i, {i, 10, 100, 10}], Table[i, {i, 100, 1000, 100}]]}];
```

In[366]:=
```
LogLogPlot[{2 Pi h c^2 / λ^5 / (Exp[h * c / k / λ / T] - 1) /. {c → 2.997925 * 10^8,
    k -> 1.3806488 / 10^23, h → 6.626068 / 10^34, T → {Table[500 * i, {i, 1, 12}]}},
  2 Pi h c^2 / λ^5 / (Exp[h c / k / 0.0028977] - 1) /.
   {c → 2.997925 * 10^8, k → 1.3806488 / 10^23, h → 6.626068 / 10^34}},
 {λ, 5*^-8, 1*^-4}, PlotRange → {{4*^-8, 1*^-4}, {1*^8, 1*^14}},
 AxesLabel → {"λ/μm", "Hλ"}, Ticks → {popisky, None},
 BaseStyle → {FontFamily → "Times", FontSize → 14},
 PlotStyle → {Black, {Black, Dashed, Thickness[0.006]}}]
```

Out[366]=



7.2 Vývoj hustoty pravděpodobnosti částice v jednorozměrné nekonečně hluboké potenciálové jámě

Tato úloha patří mezi základní úlohy kvantové mechaniky. V případě částice o hmotnosti m v nekonečně hluboké potenciálové jámě šířky a pro energii n-tého kvantového stavu platí

$$E_n = \frac{\pi^2 \hbar^2}{2 m a^2} n^2.$$

Definujme příslušnou funkci:

In[367]:=
```
energie1[n_] := π^2 ℏ^2 / (2 m a^2) n^2
```

Podobně pro vlnovou funkci n-tého kvantového stavu:

$$\varphi_n = \sqrt{\frac{2}{a}} \sin\left(\frac{\pi n}{a} x\right)$$

In[368]:=
```
vlfce1[x_, n_] := Sqrt[2 / a] Sin[π n / a x]
```

Pro časovou závislost vlnové funkce v jednotlivých kvantových stavech pak platí:

$$\phi_n = \varphi_n \exp\left(-\frac{\mathrm{i} E_n}{\hbar} t\right)$$

In[369]:=
```
vlfcecas1[t_, x_, n_] := vlfce1[x, n] Exp[-I energie1[n] / ℏ t]
```

V dalších výpočtech je vhodné zavést vhodné položit konstanty, které neovlivňují výsledek po kvalitativní stránce, rovny 1, proto:

In[370]:=
```
ℏ = 1;
m = 1;
a = 1;
```

Nyní pomocí předpisu **Table** vybereme kvantové stavy, které chceme skládat a pomocí součtu prvků **Total** vytvoříme výslednou vlnovou funkci a pro kontrolu vypíšeme její tvar.

In[373]:=
```
n1 = Table[i, {i, 2, 8, 2}]
vyslfce1[t_, x_] := Total[vlfcecas1[t, x, n1]]
vyslfce1[t, x]
```

Out[373]= {2, 4, 6, 8}

Out[375]= $\sqrt{2}\, e^{-2 i \pi^2 t} \text{Sin}[2 \pi x] + \sqrt{2}\, e^{-8 i \pi^2 t} \text{Sin}[4 \pi x] + \sqrt{2}\, e^{-18 i \pi^2 t} \text{Sin}[6 \pi x] + \sqrt{2}\, e^{-32 i \pi^2 t} \text{Sin}[8 \pi x]$

Normovací konstantu výsledné funkce (částice se nachází mezi polohami $x = 0$ a $x = a$) zjistíme integrací pro nějaký konkrétní čas, např. $t = 0$.

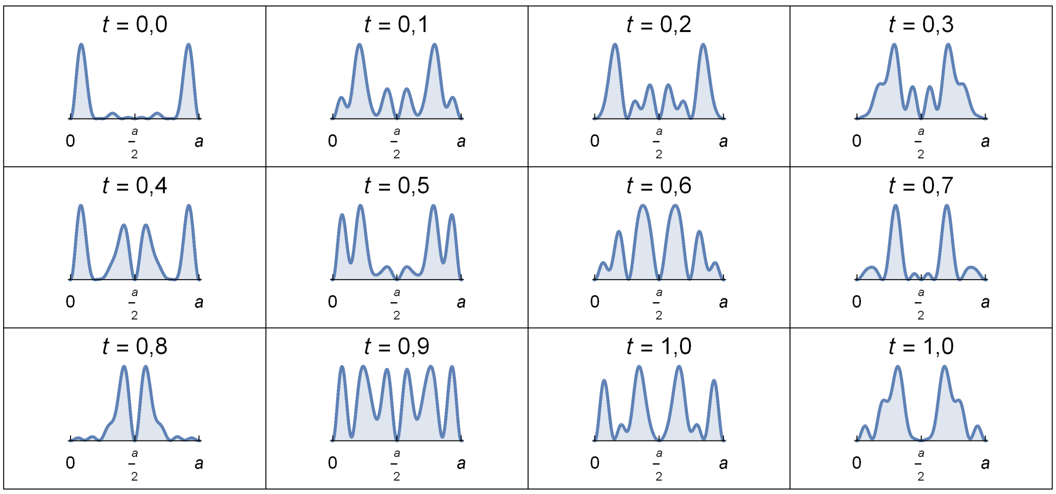
In[376]:=
```
A1 = Sqrt[1 / Integrate[vyslfce1[0, x] * Conjugate[vyslfce1[0, x]], {x, 0, a}]]
```

Out[376]= $\frac{1}{2}$

Nyní již můžeme vykreslit závislost hustoty $\phi \cdot \phi^*$ pravděpodobnosti na poloze v různých časech.

In[377]:=
```
GraphicsGrid[
 {Table[Plot[A1^2 vyslfce1[t, x] Conjugate[vyslfce1[t, x]], {x, 0, a}, Filling → Axis,
    PlotLabel → "t = " ~~ ToString[NumberForm[t, {1, 1}, NumberPoint → ","]],
    Ticks → {{{0, "0"}, {0.5, "a/2"}, {1, "a"}}, None}, Axes → {True, False},
    PlotRange → {Automatic, All}], {t , 0, 0.3, 0.1}],
  Table[Plot[A1^2 vyslfce1[t, x] Conjugate[vyslfce1[t, x]], {x, 0, a}, Filling → Axis,
    PlotLabel → "t = " ~~ ToString[NumberForm[t, {1, 1}, NumberPoint → ","]],
    Ticks → {{{0, "0"}, {0.5, "a/2"}, {1, "a"}}, None}, Axes → {True, False},
    PlotRange → {Automatic, All}], {t , 0.4, 0.7, 0.1}],
  Table[Plot[A1^2 vyslfce1[t, x] Conjugate[vyslfce1[t, x]], {x, 0, a}, Filling → Axis,
    PlotLabel → "t = " ~~ ToString[NumberForm[t, {1, 1}, NumberPoint → ","]],
    Ticks → {{{0, "0"}, {0.5, "a/2"}, {1, "a"}}, None}, Axes → {True, False},
    PlotRange → {Automatic, All}], {t , 0.8, 1.1, 0.1}]}, ImageSize → 550, Frame → All]
```

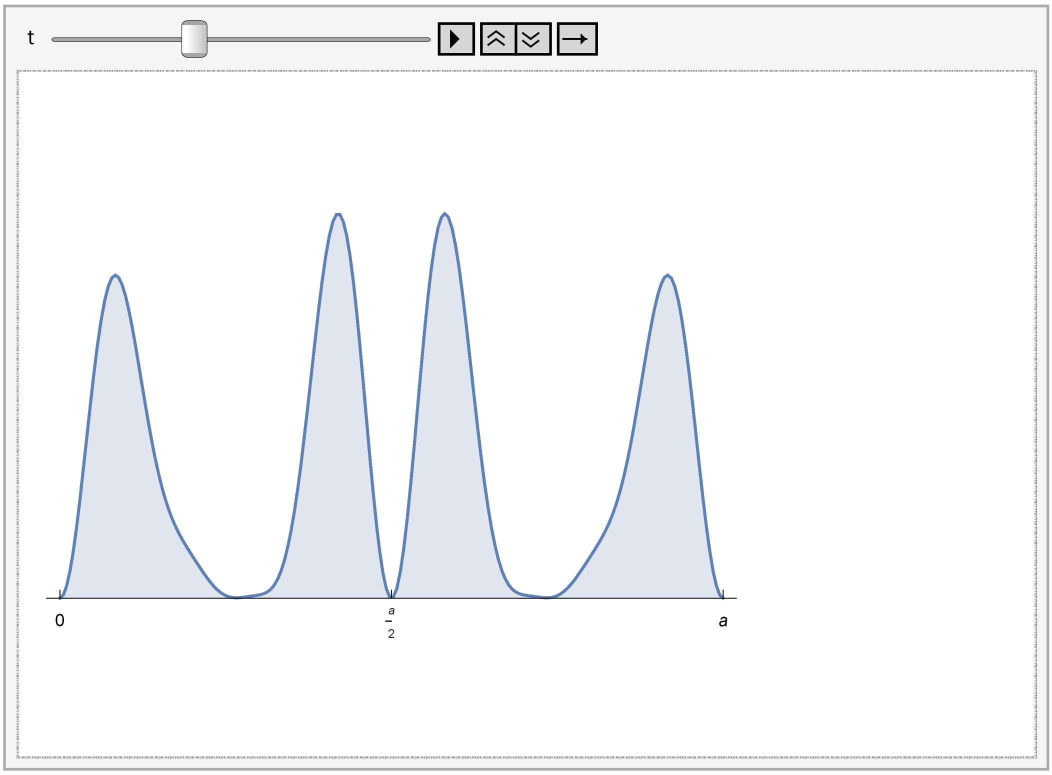
Out[377]=



Pro animaci můžeme použít příkaz **Animate**:

In[378]:=
```
Animate[Plot[A1^2 vyslfce1[t, x] Conjugate[vyslfce1[t, x]], {x, 0, a}, Filling → Axis,
  Ticks → {{{0, "0"}, {0.5, "a/2"}, {1, "a"}}, None}, Axes → {True, False},
  PlotRange → {Automatic, All}], {t, 0, 2, 1*^-5}, AnimationRunning → False]
```

Out[378]=



Vyzkoušejte, že pokud zvolíte pouze stacionární kvantový stav (např. $n = 2$), hustota pravděpodobnosti se s časem nemění.
Pokud bychom chtěli animaci uložit jako videosekvenci, není výhodné vycházet z příkazu **Animate** (ve videu jsou pak zařazeny snímky jak dopředu tak pozpátku). Místo toho vytvoříme **Table** z jednotlivých snímků a ten poté převedeme na videosekvenci (nezapomeňme na středník, aby se všechny jednotlivé obrázky nezobrazovaly). Oproti animacím v programu *Mathematica* se při exportu vyplatí zmenšit počet snímků (v našem případě časový interval a použít jeho hrubší dělení) a obrnit se trpělivostí.

In[379]:=
```
(* video =
 Table[Plot[A^2 vyslfce[t,x] Conjugate[vyslfce[t,x]],{x,0,a},Filling → Axis,
   Ticks → {{{0,"0"},{0.5,"a/2"},{1,"a"}},None},Axes → {True,False}],{t,0,1,1*^-3}] *)
```

In[380]:=
```
(* Export["jama.flv",video] *)
```

7.3 Vývoj hustoty pravděpodobnosti lineárního harmonického oscilátoru

Program *Mathematica* nabízí i velké množství předdefinovaných speciálních funkcí, v případě lineárního harmonického oscilátoru (LHO) využijeme Hermiteovy polynomy, jinak bude postup podobný jako v předcházejícím příkladu. Také tato úloha patří mezi základní úlohy kvantové mechaniky. V případě LHO s úhlovou frekvencí ω pro energii n-tého kvantového stavu platí
$E_n = \hbar\,\omega\,(n + \frac{1}{2})$.
Definujme příslušnou funkci:

In[381]:=
```
energie2[n_] := ℏ ω (n + 1 / 2)
```

Podobně pro vlnovou funkci n-tého kvantového stavu

$$\varphi_n = \sqrt[4]{\frac{m\,\omega}{\hbar\pi}}\ \frac{1}{\sqrt{2^n\, n!}}\, e^{-\frac{\xi^2}{2}}\, H_n(\xi)$$

kde m je hmotnost oscilátoru a $\xi = \sqrt{\frac{m\,\omega}{\hbar\pi}}\ x$ přeškálovaná souřadnice.

In[382]:=
```
vlfce2[ξ_, n_] := (m ω / ℏ / π)^(1 / 4) 1 / Sqrt[2^n n!] HermiteH[n, ξ] Exp[-ξ^2 / 2]
```

Pro časovou závislost vlnové funkce v jednotlivých kvantových stavech pak platí

$\phi_n = \varphi_n \exp\left(-\frac{i\, E_n}{\hbar}\, t\right)$

In[383]:=
```
vlfcecas2[t_, ξ_, n_] := vlfce2[ξ, n] * Exp[-I * energie2[n] / ℏ * t]
```

V dalších výpočtech je vhodné zavést vhodné položit konstanty, teré neovlivňují výsledek po kvalitativní stránce, rovny 1, proto:

In[384]:=
```
ℏ = 1;
m = 1;
ω = 1;
```

Nyní pomocí předpisu **Table** vybereme kvantové stavy, které chceme skládat a pomocí součtu prvků **Total** vytvoříme výslednou vlnovou funkci a pro kontrolu vypíšeme její tvar

In[387]:=
```
n2 = Table[i, {i, 2, 6, 2}]
vyslfce2[t_, ξ_] := Total[vlfcecas2[t, ξ, n2]]
Simplify[vyslfce2[t, ξ]]
```

Out[387]= {2, 4, 6}

Out[389]=

$$\frac{1}{60\,\pi^{1/4}} e^{-\frac{13\,i\,t}{2}-\frac{\xi^2}{2}} \left(30\sqrt{2}\, e^{4\,i\,t}\left(-1+2\,\xi^2\right)+5\sqrt{6}\, e^{2\,i\,t}\left(3-12\,\xi^2+4\,\xi^4\right)+\sqrt{5}\left(-15+90\,\xi^2-60\,\xi^4+8\,\xi^6\right)\right)$$

Normovací konstantu výsledné funkce (částice se nachází mezi polohami $x = 0$ a $x = a$) zjistíme integrací pro nějaký konkrétní čas, např. $t = 0$ (pro součet většího počtu stavů bude chvíli trvat, můžrme proto změřit, jak dlouho bude trvat získání výsledku pomocí příkazu **AbsoluteTiming**, který zahrnuje celkový čas k získání výsledku včetně stažení dat z interentu apod.; pokud by nás zajímal jen čas využitý procesorem, stačil by příkaz **Timing**):

In[390]:=
```
AbsoluteTiming[
 A2 = Sqrt[1 / Integrate[vyslfce2[0, ξ] Conjugate[vyslfce2[0, ξ]], {ξ, -∞, ∞}]] // N]
nameAndValue[A2, 2]
```

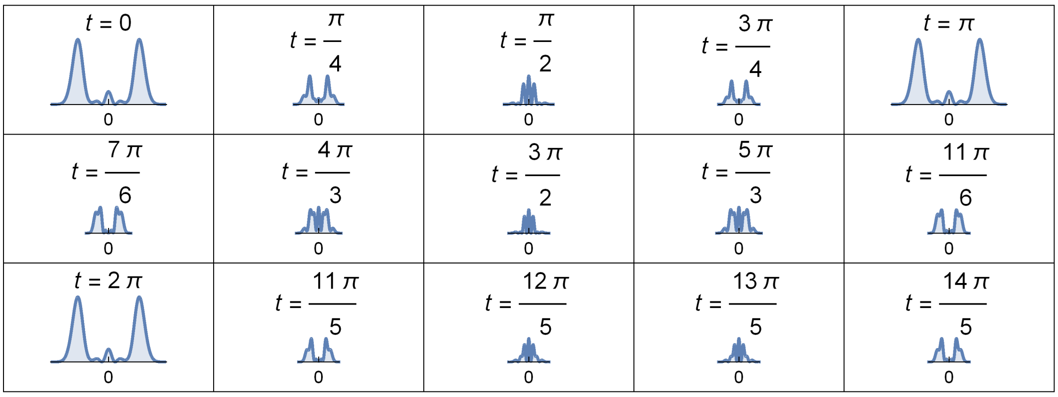
Out[390]= {8.227359, 0.57735}

A2=0,58

Nyní již můžeme vykreslit závislost hustoty $\phi \cdot \phi^*$ pravděpodobnosti na poloze v různých časech.

In[392]:=
```
GraphicsGrid[
 {Table[Plot[A2^2 vyslfce2[t, ξ] Conjugate[vyslfce2[t, ξ]], {ξ, -5, 5}, Filling → Axis,
    PlotLabel → "t = "  ~~ ToString[TraditionalForm[t]], Ticks → {{{0, "0"}}, None},
    Axes → {True, False}, PlotRange → {Automatic, All}], {t , 0, π, π / 4}],
  Table[Plot[A2^2 vyslfce2[t, ξ] Conjugate[vyslfce2[t, ξ]], {ξ, -5, 5}, Filling → Axis,
    PlotLabel → "t = " ~~ ToString[TraditionalForm[t]], Ticks → {{{0, "0"}}, None},
    Axes → {True, False}, PlotRange → {Automatic, All}], {t , π + π / 6, 11 π / 6, π / 6}],
  Table[Plot[A2^2 vyslfce2[t, ξ] Conjugate[vyslfce2[t, ξ]], {ξ, -5, 5},
    Filling → Axis, PlotLabel → "t = " ~~ ToString[TraditionalForm[t]],
    Ticks → {{{0, "0"}}, None}, Axes → {True, False}, PlotRange → {Automatic, All}],
   {t , 2 π, 14 π / 5, π / 5}]}, ImageSize → 550, Frame → All]
```

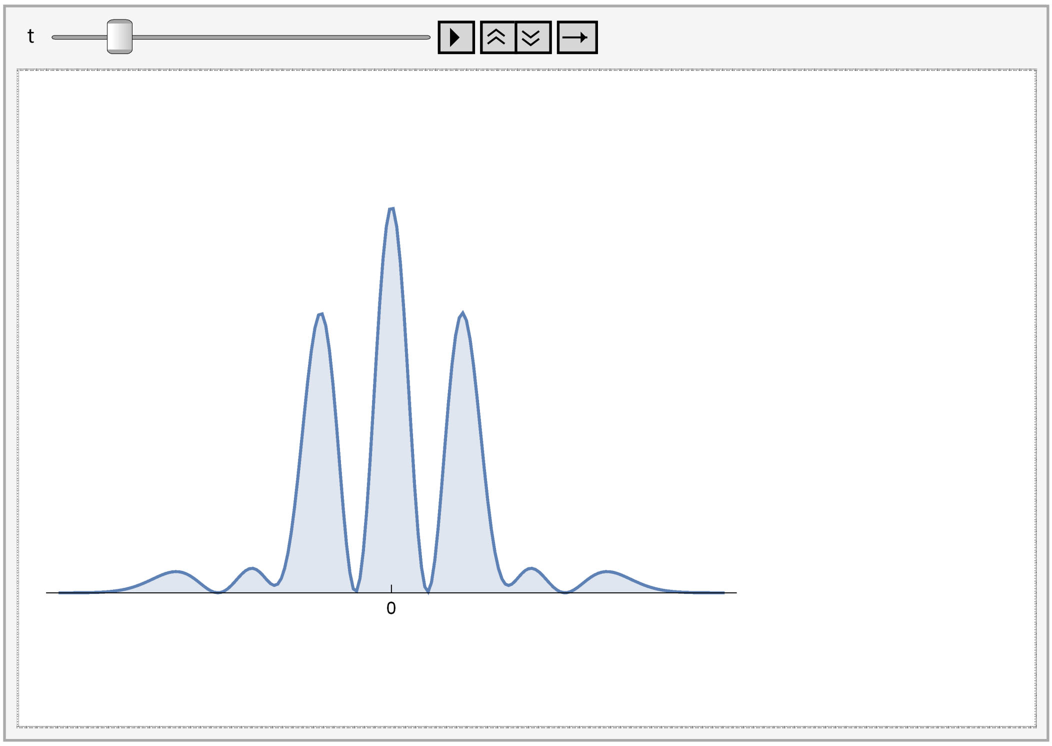
Out[392]=



Pro animaci můžeme použít příkaz **Animate**:

In[393]:=
```
Animate[Plot[A2^2 vyslfce2[t, ξ] Conjugate[vyslfce2[t, ξ]], {ξ, -5, 5}, Filling → Axis,
  Ticks → {{{0, "0"}}, None}, Axes → {True, False}, PlotRange → {Automatic, All}],
 {t, 0, 4 π, 1*^-5}, AnimationRunning → False]
```

Out[393]=



Vyzkoušejte, že pokud zvolíte pouze stacionární kvantový stav (např. $n = 2$), hustota pravděpodobnosti se s časem nemění.

7.4 Sférické harmonické funkce

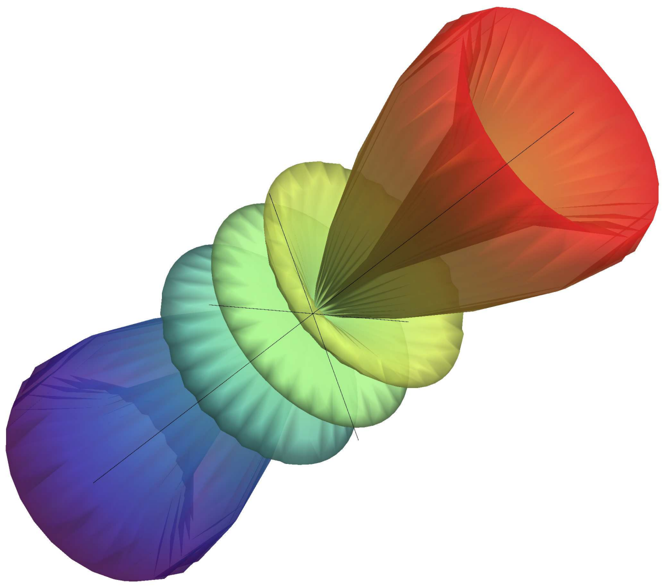
Velmi jednoduše lze pomocí programu *Mathematica* vykreslit sférické harmonické funkce, jejichž čtverec absolutní hodnoty určuje i známé tvary orbitalů atomu vodíku. Stačí zadat vedlejší kvantové číslo *l* a magnetické kvantové číslo *m* a podle potřeby definovat parametry zobrazení. Příkaz **SphericalPlot3D** pak umožnuje přímo vykreslit funkci zadanou ve sférických souřadnicích. Při zjednodušování čtverce absolutní hodnoty je vhodné přidat předpoklad o práci s reálnými proměnnými, zjednodušování goniometrických funkcí s (automaticky předpokládanými komplexními argumenty) většinou neupraví funkci na kompaktní tvar.

In[394]:=
```
l = 5;
m = 1;
SphericalHarmonicY[l, m, ϑ, φ]
grafY = Simplify[SphericalHarmonicY[l, m, ϑ, φ] *
   Conjugate[SphericalHarmonicY[l, m, ϑ, φ]], ϑ ∈ Reals && φ ∈ Reals]
SphericalPlot3D[grafY, {ϑ, 0, π}, {φ, 0, 2 * π}, Mesh → None,
 Boxed → False, Axes → True, ColorFunction → "Rainbow",
 PlotStyle → Directive[Opacity[0.7]], AxesOrigin → {0, 0, 0}, Ticks → None]
```

Out[396]= $-\frac{1}{16} e^{i\varphi} \sqrt{\frac{165}{2\pi}} \left(1 - 14 \operatorname{Cos}[\vartheta]^2 + 21 \operatorname{Cos}[\vartheta]^4\right) \operatorname{Sin}[\vartheta]$

Out[397]= $\frac{165 \left(1 - 14 \operatorname{Cos}[\vartheta]^2 + 21 \operatorname{Cos}[\vartheta]^4\right)^2 \operatorname{Sin}[\vartheta]^2}{512\pi}$

Out[398]=



Snadno také ověříme normování druhé mocniny sférických harmonických funkcí vzhledem k integraci přes prostorový úhel (dΩ = $2\pi \sin\vartheta$):

In[399]:=
```
Integrate[grafY * 2 * π * Sin[ϑ], {ϑ, 0, π}]
```

Out[399]= 1

7.5 Některé fraktály

Mnoho přírodních tvarů je možné modelovat fraktální geometrií, například hory, mraky, sněhové vločky, řeky nebo cévní systém. Krásným příkladem organického fraktálu je romanesko (druh květáku). Často se tvary stromů a kapradí v přírodě modelují na počítačích použitím rekurzivních algoritmů. Jedním z nejznámějších a působivých příkladů je i Madelbrotova množina. Termín fraktál použil poprvé matematik Benoît Mandelbrot v roce 1975 a pochází z latinského *fractus* – rozbitý. Rozumíme jím geometrický objekt, který je sobě podobný (pokud ho pozorujeme v jakémkoliv měřítku či rozlišení, pozorujeme stále opakující se určitý charakteristický motiv) a mívá na první pohled velmi složitý tvar, i když je generován opakovaným použitím jednoduchých pravidel.

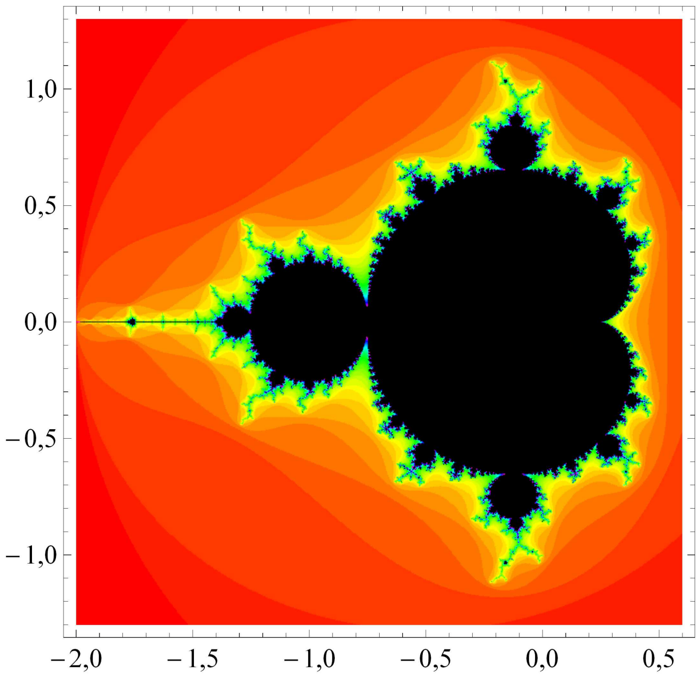
Mandelbrotova množina je množina bodů komplexní roviny, které jsou odvozeny od rekurzivních zobrazení v komplexní rovině, a její okraj je fraktálem. K vykreslení určení se používá zobrazení, které každému komplexnímu číslu c přiřazuje určitou posloupnost komplexních čísel z_n určenou následujícím rekurzivním předpisem

$z_0 = 0$, $z_{n+1} = z_n^2 + c$.

Madelbrotova množina je pak definována jako množina komplexních čísel c, pro která je posloupnost z_0, z_1, z_2,... omezená, tzn. že splňuje následující podmínku existence reálného čísla m takového, že pro všechna n je $|z_n| \leq m$. Lze dokázat, že překročí-li absolutní hodnota některého členu posloupnosti $|z_n|$ hodnotu 2, pak tato poslupnost není omezená, odtud je zřejmé, že lze ve výše uvedené definici položit $m = 2$. V algoritmu se pro každý určený bod komplexní roviny postupně vyčíslují členy posloupnosti z_n a zjišťuje se, jestli splňují podmínku $|z_n| \leq 2$; v případě, že tato podmínka není splněna, bod nepatří do Mandelbrotovy množiny. Při zobrazování se podle hodnoty n, při níž došlo k nesplnění podmínky, zvolí barva, kterou bude bod zobrazen. Pro dosažení dobrého vzhledu se pro blízká „vyřazovací“ n volí podobné barvy. Pokud po vhodně zvoleném počtu iterací (v příkladu níže je zvoleno 50 opakování) zůstává uvedená podmínka splněna, je bod považován za součást Mandelbrotovy množiny (zobrazuje se obvykle černou barvou). Nastavení této hranice ovlivňuje výsledný obrázek: pro příliš malou hodnotu budou některé body chybně označeny jako patřící do množiny, ale velký počet iterací vyžaduje pochopitelně delší čas výpočtu.

In[400]:=
```
Mandel[x_, y_] :=
 Length[FixedPointList[(#^2 + x + y * I) &, 0, 50, SameTest → (Abs[#] > 2 &)]]
Style[DensityPlot[Mandel[x, y], {x, -2, 0.6}, {y, -1.3, 1.3},
   Mesh → False, AspectRatio → Automatic, Frame → True, PlotPoints → 250,
   ColorFunction → (If[# == 1, RGBColor[0, 0, 0], Hue[.9 #]] &),
   BaseStyle → {FontFamily → "Times", FontSize → 14}], NumberPoint → ","]
```

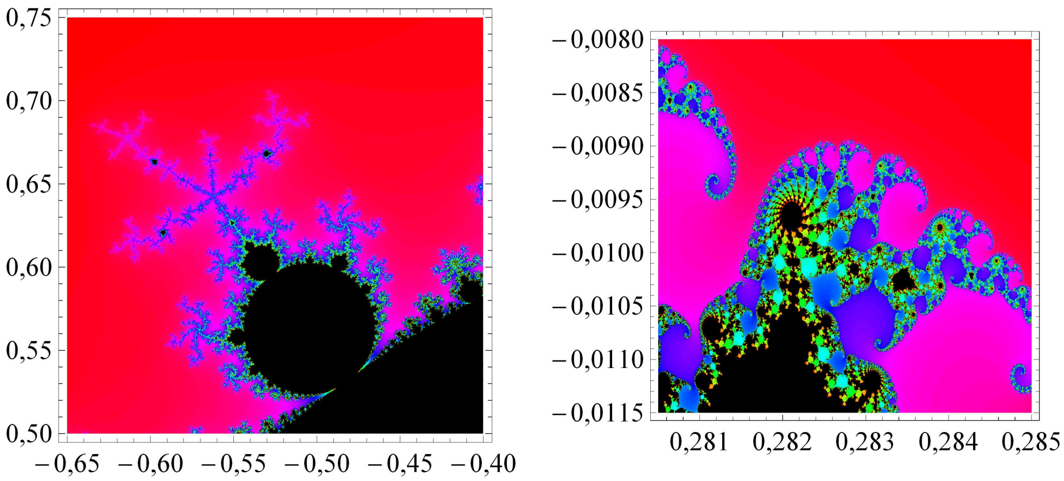
Out[401]=



Na internetu lze najít řadu návodů, jak Mandelbrotovu množinu vykreslit. Ukažme ještě jednu o něco efektivnější možnost, pomocí níž zobrazíme některé detaily množiny, pohrát si můžeme i s barvami pomocí parametru **ColorFunction**.

In[402]:=
```
compiledMandel = Compile[{{c, _Complex}},
   Length[FixedPointList[#^2 + c &, c, 150, SameTest → (Abs[#] > 2.0 &)]]];
GraphicsRow[{Style[DensityPlot[compiledMandel[x + I y], {x, -.65, -.4},
     {y, .5, .75}, PlotPoints → 300, Mesh → False, Frame → True,
     ColorFunction → (If[# == 1, RGBColor[0, 0, 0], Hue[1 - #]] &),
     BaseStyle → {FontFamily → "Times", FontSize → 14}], NumberPoint → ","],
  Style[DensityPlot[compiledMandel[x + I y], {x, 0.2805, 0.285},
     {y, -0.0115, -0.008}, PlotPoints → 300, Mesh → False, Frame → True,
     ColorFunction → (If[# == 1, RGBColor[0, 0, 0], Hue[1 - #]] &), BaseStyle →
      {FontFamily → "Times", FontSize → 14}], NumberPoint → ","]}, ImageSize → 550]
```

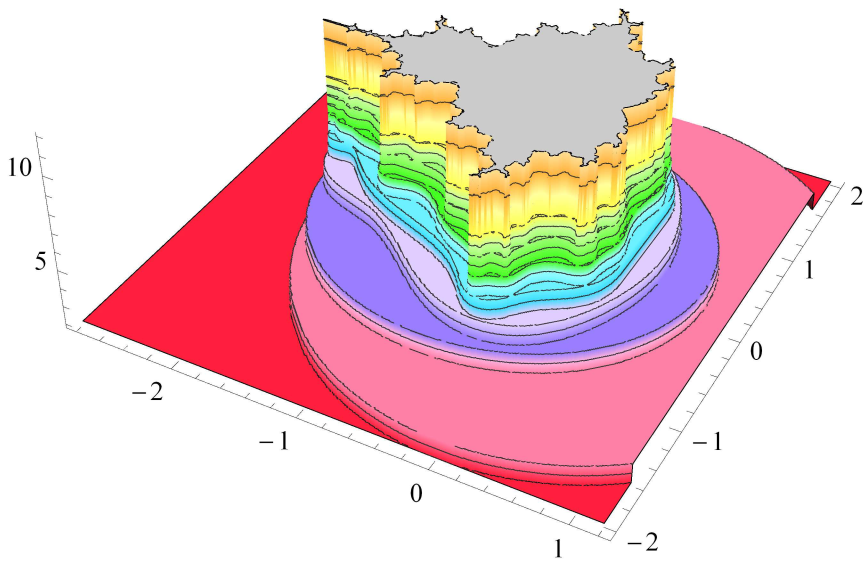
Out[403]=



V neposlední řadě můžeme počet iterací pro jednotlivé body vykreslit jako 3D funkci pomocí příkazu **Plot3D**.

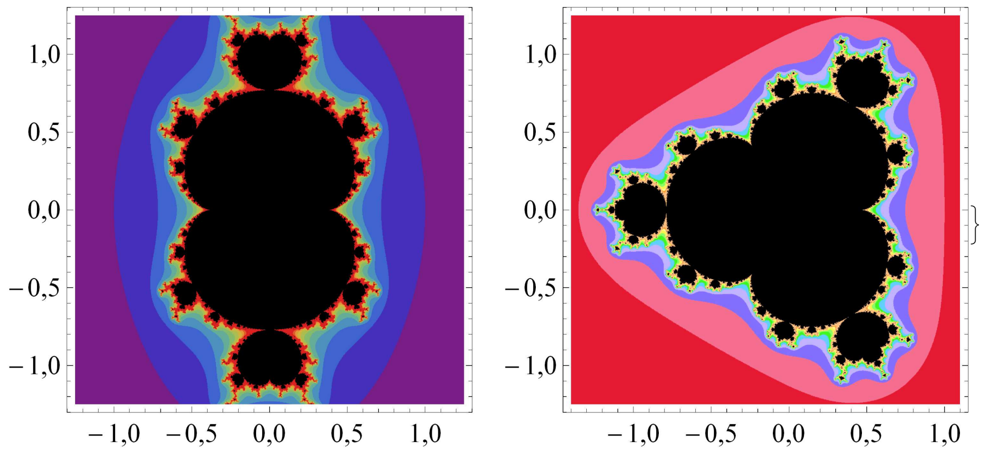
In[404]:=
```
Plot3D[compiledMandel[x + y I], {x, -2.8, 1.2}, {y, -2, 2}, PlotPoints → 200,
 MeshFunctions → {#3 &}, Mesh → {Table[i, {i, 2, 9, .5}]}, Boxed → False,
 Axes → True, ColorFunction → Function[{x, y, z}, ColorData["BrightBands"][z]],
 BaseStyle → {FontFamily → "Times", FontSize → 14}, ImageSize → 450]
```

Out[404]=



Analogicky můžeme zkoumat čísla se stejnou vlastností vzhledem k jinému rekurzivnímu předpisu, např. $z^3 + c$ nebo $z^4 + c$. Vykreslování odpovídajících množin je pak výzvou "pohrát si s barvičkami" a zkusit různé možnosti zavedení parametru **ColorFunction**. Vzhledem k tomu, že počet iterací je zvolen 150, můžeme si i změřit čas, který *Mathematica* na našem počítači (a při jeho daném vytížení) potřebuje k zobrazení (nejen k samotnému výpočtu) pomocí příkazu **Timing** (výsledek se zobrazí vlevo vedle grafického výstupu).

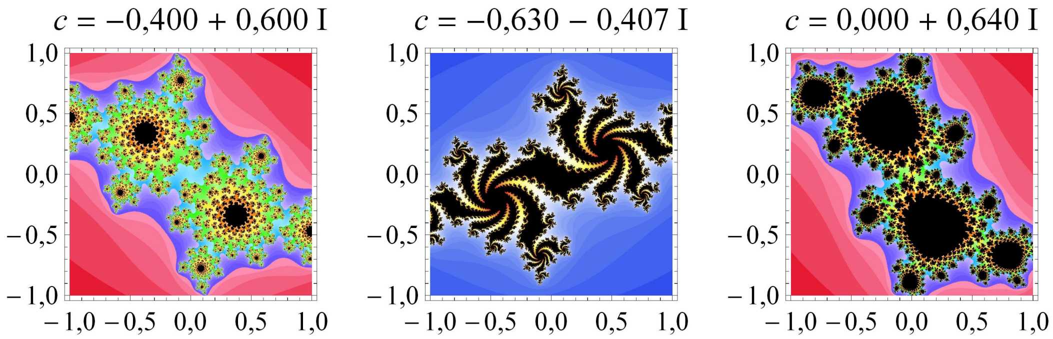
In[405]:=
```
compiledMandel3 = Compile[{{c, _Complex}},
   Length[FixedPointList[#^3 + c &, c, 150, SameTest → (Abs[#] > 2.0 &)]]];
compiledMandel4 = Compile[{{c, _Complex}},
   Length[FixedPointList[#^4 + c &, c, 150, SameTest → (Abs[#] > 2.0 &)]]];
Timing[GraphicsRow[{Style[DensityPlot[compiledMandel3[x + I y], {x, -1.25, 1.25},
     {y, -1.25, 1.25}, PlotPoints → 300, Mesh → False, Frame → True,
     ColorFunction → (If[# == 1, RGBColor[0, 0, 0], ColorData["Rainbow"][15 #]] &),
     BaseStyle → {FontFamily → "Times", FontSize → 14}], NumberPoint → ","],
   Style[DensityPlot[compiledMandel4[x + I y], {x, -1.4, 1.1},
     {y, -1.25, 1.25}, PlotPoints → 300, Mesh → False, Frame → True,
     ColorFunction → (If[# == 1, RGBColor[0, 0, 0], ColorData["BrightBands"][15 #]] &),
     BaseStyle → {FontFamily → "Times", FontSize → 14}],
    NumberPoint → ","]}, ImageSize → 500]]
```

Out[407]= {17.659313,

}

S Mandelbrotovou množinou úzce souvisí i pojem Juliovy množiny. Je definována jako množina všech bodů z v komplexní rovině, pro které posloupnost $z_{n+1} = z_n^2 + c$, kde c je libovolné komplexní číslo, nediverguje. Hranice takových množin tvoří fraktál a poprvé byly popsány francouzskými matematiky Gastonem Juliou a Pierrem Fatou. Podle volby bodu c lze vygenerovat nejrůznější tvary a bodům přiřadit barvy podle počtu kroků, po nichž $|z_n|$ překročí hodnotu 2. Zatímco u Mandelbrotovy množiny měníme c a začínáme v jedné konkrétní hodnotě z (konkrétně $z = 0$), u Juliovy množiny volíme konkrétní c a zkoumáme chování bodů v určité části komplexní roviny v dané posloupnosti. Počet iterací zde volíme 100.

In[408]:=
```
Julia[zo_, c_] := Module[{z = zo, i = 0}, While[i < 100 && Abs[z] < 2, z = z^2 + c;
    i++]; i];
GraphicsRow[{Style[DensityPlot[Sqrt[Julia[x + I y, -0.4 + .6 I]],
    {x, -1, 1}, {y, -1, 1}, PlotPoints → 275, Mesh → False, Frame → True,
    ColorFunction → (If[# == 1, RGBColor[0, 0, 0], ColorData["BrightBands"][#]] &),
    BaseStyle → {FontFamily → "Times", FontSize → 14},
    PlotLabel → "c = " ~~ ToString[NumberForm[-.4 + .6 I, {4, 3}, NumberPoint → ","]]],
   NumberPoint → ","], Style[DensityPlot[Sqrt[Julia[x + I y, -0.63 - 0.407 I]],
    {x, -1, 1}, {y, -1, 1}, PlotPoints → 275, Mesh → False, Frame → True,
    ColorFunction → (If[# == 1, RGBColor[0, 0, 0], ColorData["TemperatureMap"][#]] &),
    BaseStyle → {FontFamily → "Times", FontSize → 14}, PlotLabel →
     "c = " ~~ ToString[NumberForm[-0.63 - 0.407 I, {4, 3}, NumberPoint → ","]]],
   NumberPoint → ","], Style[DensityPlot[Sqrt[Julia[x + I y, .64 I]],
    {x, -1, 1}, {y, -1, 1}, PlotPoints → 275, Mesh → False, Frame → True,
    ColorFunction → (If[# == 1, RGBColor[0, 0, 0], ColorData["BrightBands"][#]] &),
    BaseStyle → {FontFamily → "Times", FontSize → 14},
    PlotLabel → "c = " ~~ ToString[NumberForm[.64 I, {4, 3}, NumberPoint → ","]]],
   NumberPoint → ","]}, ImageSize → 550]
```

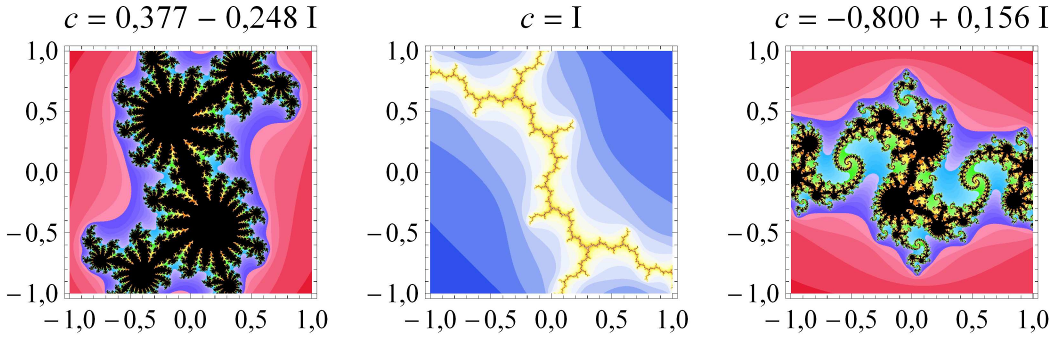
Out[409]=



Vyzkoušíme ještě jednu trojici čísel c.

In[410]:=
```
GraphicsRow[{Style[DensityPlot[Sqrt[Julia[x + I y, 0.377 - .248 I]],
    {x, -1, 1}, {y, -1, 1}, PlotPoints → 275, Mesh → False, Frame → True,
    ColorFunction → (If[# == 1, RGBColor[0, 0, 0], ColorData["BrightBands"][#]] &),
    BaseStyle → {FontFamily → "Times", FontSize → 14}, PlotLabel →
     "c = " ~~ ToString[NumberForm[.377 - .248 I, {4, 3}, NumberPoint → ","]]],
   NumberPoint → ","], Style[DensityPlot[Sqrt[Julia[x + I y, I]], {x, -1, 1},
    {y, -1, 1}, PlotPoints → 275, Mesh → False, Frame → True,
    ColorFunction → (If[# == 1, RGBColor[0, 0, 0], ColorData["TemperatureMap"][#]] &),
    BaseStyle → {FontFamily → "Times", FontSize → 14},
    PlotLabel → "c = " ~~ ToString[NumberForm[I, {4, 3}, NumberPoint → ","]]],
   NumberPoint → ","], Style[DensityPlot[Sqrt[Julia[x + I y, -0.8 + 0.156 I]],
    {x, -1, 1}, {y, -1, 1}, PlotPoints → 275, Mesh → False, Frame → True,
    ColorFunction → (If[# == 1, RGBColor[0, 0, 0], ColorData["BrightBands"][#]] &),
    BaseStyle → {FontFamily → "Times", FontSize → 14}, PlotLabel →
     "c = " ~~ ToString[NumberForm[-0.8 + 0.156 I, {4, 3}, NumberPoint → ","]]],
   NumberPoint → ","]}, ImageSize → 550]
```

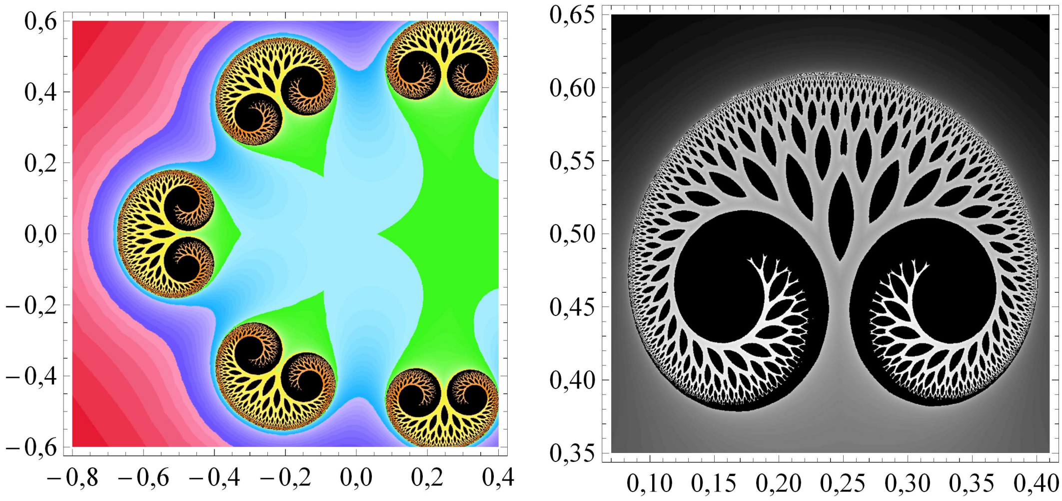
Out[410]=



Analogicky lze postupovat i pro jiné rekurentní předpisy, např. posloupnost $z_{n+1} = z_n^{2,5}$ - 0,2.

In[411]:=
```
Julia2[zo_] := Module[{z = zo, i = 0}, While[i < 100 && Abs[z] < 2, z = z^1.5 - 0.2;
    i++]; i];
GraphicsRow[{Style[DensityPlot[Sqrt[Julia2[x + I y]],
    {x, -0.8, 0.4}, {y, -.6, .6}, PlotPoints → 275, Mesh → False, Frame → True,
    ColorFunction → (If[# == 1, RGBColor[0, 0, 0], ColorData["BrightBands"][#]] &),
    BaseStyle → {FontFamily → "Times", FontSize → 14}], NumberPoint → ","],
  Style[DensityPlot[Sqrt[Julia2[x + I y]], {x, .07, 0.41}, {y, .35, .65},
    PlotPoints → 275, Mesh → False, Frame → True,
    ColorFunction → (If[# == 1, RGBColor[0, 0, 0], GrayLevel[#]] &), BaseStyle →
     {FontFamily → "Times", FontSize → 14}], NumberPoint → ","]}, ImageSize → 550]
```

Out[412]=



8 Obecná teorie relativity

8.1 Určení Ricciho a Einsteinova tenzoru pro sféricky symetrický prostoročas

Rutinní výpočty složek tenzorů v obecné teorii relativity dnes většinou svěřujeme programům CAS (Computer Algebra Systems), mezi něž patří i *Mathematica*. Za nejjednodušší a základní aplikaci lze považovat výpočet Christoffelových symbolů, Ricciho a Einsteinova tenzoru pro daný metrický tenzor. Ukažme si výpočet na příkladu sféricky symetrického prostoročasu. Nejprve (pro případ opakování výpočtu) vymažeme z paměti potřebné proměnné:

In[413]:=
```
Clear[n, coord, metric, inversemetric,
  affine, riemann, ricci, scalar, einstein, t, r, ϑ, φ, A, B];
```

Poté zadáme rozměr prostoročasu (počet souřadnic), souřadnice, kovariantní metrický tenzor a necháme vypsat ve tvaru matice:

In[414]:=
```
n = 4;
coord = {t, r, ϑ, φ};
metric = {{-A[r], 0, 0, 0}, {0, B[r], 0, 0}, {0, 0, r^2, 0}, {0, 0, 0, r^2 * Sin[ϑ]^2}}
metric // MatrixForm
```

Out[416]= $\{\{-A[r], 0, 0, 0\}, \{0, B[r], 0, 0\}, \{0, 0, r^2, 0\}, \{0, 0, 0, r^2 \mathrm{Sin}[\vartheta]^2\}\}$

Out[417]//MatrixForm=

$$\begin{pmatrix} -A[r] & 0 & 0 & 0 \\ 0 & B[r] & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2 \mathrm{Sin}[\vartheta]^2 \end{pmatrix}$$

Dále určíme matici inverzní k matici s kovariantními složkami metrického tenzoru, která obsahuje kontravariatní složky metriky; pro přehlednost zvětšíme font.

In[418]:=
```
inversemetric = Simplify[Inverse[metric]]
Style[inversemetric // MatrixForm, FontSize → 16]
```

Out[418]= $\left\{\left\{-\frac{1}{A[r]}, 0, 0, 0\right\}, \left\{0, \frac{1}{B[r]}, 0, 0\right\}, \left\{0, 0, \frac{1}{r^2}, 0\right\}, \left\{0, 0, 0, \frac{\mathrm{Csc}[\vartheta]^2}{r^2}\right\}\right\}$

Out[419]=

$$\begin{pmatrix} -\frac{1}{A[r]} & 0 & 0 & 0 \\ 0 & \frac{1}{B[r]} & 0 & 0 \\ 0 & 0 & \frac{1}{r^2} & 0 \\ 0 & 0 & 0 & \frac{\mathrm{Csc}[\vartheta]^2}{r^2} \end{pmatrix}$$

Výpočet Christoffelových symbolů

Christoffelovy symboly zavedeme podle definice užívané v dostupné literatuře (pozor si musíme dát na znaménkovou konvenci) pomocí složek metrického tenzoru $g_{\mu\nu}$,

$$\Gamma^{\lambda}{}_{\mu\nu} = \frac{1}{2}\, g^{\lambda\sigma}\left(-\frac{\partial g_{\mu\nu}}{\partial x^{\sigma}} + \frac{\partial g_{\sigma\mu}}{\partial x^{\nu}} + \frac{\partial g_{\sigma\nu}}{\partial x^{\mu}}\right),$$

po výpočtu pomocí funkce listaffine vypíšeme nenulové složky:

In[420]:=
```
affine := affine =
   Simplify[Table[(1 / 2) * Sum[(inversemetric[[i, s]]) * (D[metric[[s, j]], coord[[k]]] +
          D[metric[[s, k]], coord[[j]]] - D[metric[[j, k]], coord[[s]]]),
        {s, 1, n}], {i, 1, n}, {j, 1, n}, {k, 1, n}]];
listaffine := Table[If[UnsameQ[affine[[i, j, k]], 0], {Style[
      Subscript[Superscript[Γ, coord[[i]]], coord[[j]] , coord[[k]]], FontSize → 16],
     "=", Style[affine[[i, j, k]], FontSize → 16]}], {i, 1, n}, {j, 1, n}, {k, 1, j}]
TableForm[Partition[DeleteCases[Flatten[listaffine], Null], 3], TableSpacing → {2, 2}]
```

Out[422]//TableForm=

$$\Gamma^{t}{}_{r,t} = \frac{A'[r]}{2\,A[r]}$$

$$\Gamma^{r}{}_{t,t} = \frac{A'[r]}{2\,B[r]}$$

$$\Gamma^{r}{}_{r,r} = \frac{B'[r]}{2\,B[r]}$$

$$\Gamma^{r}{}_{\vartheta,\vartheta} = -\frac{r}{B[r]}$$

$$\Gamma^{r}{}_{\varphi,\varphi} = -\frac{r\,\mathrm{Sin}[\vartheta]^2}{B[r]}$$

$$\Gamma^{\vartheta}{}_{\vartheta,r} = \frac{1}{r}$$

$$\Gamma^{\vartheta}{}_{\varphi,\varphi} = -\mathrm{Cos}[\vartheta]\,\mathrm{Sin}[\vartheta]$$

$$\Gamma^{\varphi}{}_{\varphi,r} = \frac{1}{r}$$

$$\Gamma^{\varphi}{}_{\varphi,\vartheta} = \mathrm{Cot}[\vartheta]$$

Výpočet Ricciho tenzoru (kovariantní složky)

Podobně zavedeme vztahy pro výpočet Ricciho tenzoru

$$R_{\alpha\beta} = \Gamma^{\gamma}{}_{\alpha\beta}\,\Gamma^{\delta}{}_{\alpha\delta} - \Gamma^{\gamma}{}_{\alpha\delta}\,\Gamma^{\delta}{}_{\beta\gamma} + \frac{\partial\Gamma^{\gamma}{}_{\alpha\beta}}{\partial x^{\gamma}} - \frac{\partial\Gamma^{\gamma}{}_{\alpha\gamma}}{\partial x^{\beta}}$$

In[423]:=
```
ricci := ricci = Simplify[
    Table[Sum[D[affine[[i, j, k]], coord[[i]]] - D[affine[[i, j, i]], coord[[k]]] +
       Sum[affine[[s, j, k]] * affine[[i, i, s]] - affine[[s, j, i]] * affine[[i, k, s]],
        {s, 1, n}], {i, 1, n}], {j, 1, n}, {k, 1, n}]];
listricci := Table[If[UnsameQ[ricci[[j, l]], 0],
    {Style[Subscript[R, coord[[j]], coord[[l]]], FontSize → 16],
     "=", Style[ricci[[j, l]], FontSize → 16]}], {j, 1, n}, {l, 1, j}]
TableForm[Partition[DeleteCases[Flatten[listricci], Null], 3], TableSpacing → {2, 2}]
```

Out[425]//TableForm=

$$R_{t,t} = \frac{-r\,B[r]\,A'[r]^2 + A[r]\,(-r\,A'[r]\,B'[r] + 2\,B[r]\,(2\,A'[r] + r\,A''[r]))}{4\,r\,A[r]\,B[r]^2}$$

$$R_{r,r} = \frac{A[r]\,(4\,A[r] + r\,A'[r])\,B'[r] + r\,B[r]\,\left(A'[r]^2 - 2\,A[r]\,A''[r]\right)}{4\,r\,A[r]^2\,B[r]}$$

$$R_{\vartheta,\vartheta} = \frac{1}{2}\left(2 - \frac{2 + \frac{r\,A'[r]}{A[r]}}{B[r]} + \frac{r\,B'[r]}{B[r]^2}\right)$$

$$R_{\varphi,\varphi} = \frac{\mathrm{Sin}[\vartheta]^2\,\left(-r\,B[r]\,A'[r] + A[r]\,\left(-2\,B[r] + 2\,B[r]^2 + r\,B'[r]\right)\right)}{2\,A[r]\,B[r]^2}$$

Ricciho tenzor

In[426]:=
```
ricci
```

Out[426]=

$$\left\{\left\{\frac{-r\,B[r]\,A'[r]^2+A[r]\,(-r\,A'[r]\,B'[r]+2\,B[r]\,(2\,A'[r]+r\,A''[r]))}{4\,r\,A[r]\,B[r]^2},0,0,0\right\},\right.$$
$$\left\{0,\frac{A[r]\,(4\,A[r]+r\,A'[r])\,B'[r]+r\,B[r]\,\left(A'[r]^2-2\,A[r]\,A''[r]\right)}{4\,r\,A[r]^2\,B[r]},0,0\right\},$$
$$\left\{0,0,\frac{1}{2}\left(2-\frac{2+\frac{r\,A'[r]}{A[r]}}{B[r]}+\frac{r\,B'[r]}{B[r]^2}\right),0\right\},$$
$$\left.\left\{0,0,0,\frac{\mathrm{Sin}[\vartheta]^2\left(-r\,B[r]\,A'[r]+A[r]\left(-2\,B[r]+2\,B[r]^2+r\,B'[r]\right)\right)}{2\,A[r]\,B[r]^2}\right\}\right\}$$

Skalární křivost

Skalární křivost je definována pomocí Ricciho tenzoru
$R = g^{\mu\nu} R_{\mu\nu}$

In[427]:=
```
scalar = Simplify[Sum[inversemetric[[i, j]] ricci[[i, j]], {i, 1, n}, {j, 1, n}]]
```

Out[427]=

$$\frac{1}{2\,r^2\,A[r]^2\,B[r]^2}\left(r^2\,B[r]\,A'[r]^2+4\,A[r]^2\left(-B[r]+B[r]^2+r\,B'[r]\right)+r\,A[r]\,(r\,A'[r]\,B'[r]-2\,B[r]\,(2\,A'[r]+r\,A''[r]))\right)$$

Einsteinův tenzor (kovariantní složky)

Einsteinův tenzor pak zavedeme podle vztahu
$G_{\mu\nu} = R_{\mu\nu} - 1/2\, g_{\mu\nu} R$

In[428]:=
```
einstein := einstein = Simplify[ricci - (1 / 2) scalar * metric]
listeinstein := Table[If[UnsameQ[einstein[[j, l]], 0],
   {Style[Subscript[G, coord[[j]], coord[[l]]], FontSize → 16], "=",
    Style[einstein[[j, l]], FontSize → 16]}], {j, 1, n}, {l, 1, n}]
TableForm[Partition[DeleteCases[Flatten[listeinstein], Null], 3], TableSpacing → {2, 2}]
```

Out[430]//TableForm=

$$G_{t,t} = \frac{A[r]\left(-B[r]+B[r]^2+r\,B'[r]\right)}{r^2\,B[r]^2}$$

$$G_{r,r} = \frac{A[r]-A[r]\,B[r]+r\,A'[r]}{r^2\,A[r]}$$

$$G_{\vartheta,\vartheta} = \frac{r\left(-r\,B[r]\,A'[r]^2-2\,A[r]^2\,B'[r]+A[r]\,(-r\,A'[r]\,B'[r]+2\,B[r]\,(A'[r]+r\,A''[r]))\right)}{4\,A[r]^2\,B[r]^2}$$

$$G_{\varphi,\varphi} = \frac{r\,\mathrm{Sin}[\vartheta]^2\left(-r\,B[r]\,A'[r]^2-2\,A[r]^2\,B'[r]+A[r]\,(-r\,A'[r]\,B'[r]+2\,B[r]\,(A'[r]+r\,A''[r]))\right)}{4\,A[r]^2\,B[r]^2}$$

In[431]:=
```
einstein
```

Out[431]=

$$\left\{\left\{\frac{A[r]\left(-B[r]+B[r]^2+r B'[r]\right)}{r^2 B[r]^2}, 0, 0, 0\right\}, \left\{0, \frac{A[r]-A[r] B[r]+r A'[r]}{r^2 A[r]}, 0, 0\right\}, \left\{0, 0, \frac{r\left(-r B[r] A'[r]^2-2 A[r]^2 B'[r]+A[r]\left(-r A'[r] B'[r]+2 B[r]\left(A'[r]+r A''[r]\right)\right)\right)}{4 A[r]^2 B[r]^2}, 0\right\}, \left\{0, 0, 0, \frac{r \operatorname{Sin}[\vartheta]^2\left(-r B[r] A'[r]^2-2 A[r]^2 B'[r]+A[r]\left(-r A'[r] B'[r]+2 B[r]\left(A'[r]+r A''[r]\right)\right)\right)}{4 A[r]^2 B[r]^2}\right\}\right\}$$

In[432]:=
```
einsteinup :=
  einsteinup = Table[Sum[inversemetric[[i, k]] * inversemetric[[j, l]] * einstein[[k, l]],
     {k, 1, 4}, {l, 1, 4}], {i, 1, 4}, {j, 1, 4}];
listeinsteinup := Table[If[UnsameQ[einsteinup[[j, l]], 0],
   {Superscript[G, StringJoin[{ToString[coord[[j]]], ToString[coord[[l]]]}]],
    "=", einstein[[j, l]]}], {j, 1, n}, {l, 1, n}]
TableForm[Partition[DeleteCases[Flatten[listeinsteinup], Null], 3],
 TableSpacing → {2, 2}]
```

Out[434]//TableForm=

$$G^{tt} = \frac{A[r]\left(-B[r]+B[r]^2+r B'[r]\right)}{r^2 B[r]^2}$$

$$G^{rr} = \frac{A[r]-A[r] B[r]+r A'[r]}{r^2 A[r]}$$

$$G^{\vartheta\vartheta} = \frac{r\left(-r B[r] A'[r]^2-2 A[r]^2 B'[r]+A[r]\left(-r A'[r] B'[r]+2 B[r]\left(A'[r]+r A''[r]\right)\right)\right)}{4 A[r]^2 B[r]^2}$$

$$G^{\varphi\varphi} = \frac{r \operatorname{Sin}[\vartheta]^2\left(-r B[r] A'[r]^2-2 A[r]^2 B'[r]+A[r]\left(-r A'[r] B'[r]+2 B[r]\left(A'[r]+r A''[r]\right)\right)\right)}{4 A[r]^2 B[r]^2}$$

Pro kontrolu definujeme funkce A a B pro známé Schwarzschildovo řešení a po zjednodušení ověříme, že jde o plochý prostoročas s nulovou skalární křivostí, Ricciho i Einsteinovým tenzorem. Pro výpis nenulových Christoffelových symbolů použijeme opět listaffine.

In[435]:=
```
A[r_] := (1 - 2 * G * M / r / c^2);
B[r_] := 1 / (1 - 2 * G * M / r / c^2);
Simplify[scalar];
Simplify[ricci];
Simplify[einstein];
Simplify[affine];
listaffine := Table[If[UnsameQ[affine[[i, j, k]], 0],
   {Subscript[Superscript[Γ, coord[[i]]], coord[[j]] , coord[[k]]],
    "=", Simplify[affine[[i, j, k]]]}], {i, 1, n}, {j, 1, n}, {k, 1, j}]
TableForm[Partition[DeleteCases[Flatten[listaffine], Null], 3], TableSpacing → {2, 2}]
```

Out[442]//TableForm=

$\Gamma^{t}{}_{r,t}$	=	$\frac{1.\,G}{r\,(-2.\,G+c^2\,r)}$
$\Gamma^{r}{}_{t,t}$	=	$\frac{1.\,G\,(-2.\,G+c^2\,r)}{c^4\,r^3}$
$\Gamma^{r}{}_{r,r}$	=	$-\frac{1.\,G}{r\,(-2.\,G+c^2\,r)}$
$\Gamma^{r}{}_{\vartheta,\vartheta}$	=	$\frac{2.\,G}{c^2} - 1.\,r$
$\Gamma^{r}{}_{\varphi,\varphi}$	=	$-\frac{1.\,(-2.\,G+c^2\,r)\,\mathrm{Sin}[\vartheta]^2}{c^2}$
$\Gamma^{\vartheta}{}_{\vartheta,r}$	=	$\frac{1}{r}$
$\Gamma^{\vartheta}{}_{\varphi,\varphi}$	=	$-\mathrm{Cos}[\vartheta]\,\mathrm{Sin}[\vartheta]$
$\Gamma^{\varphi}{}_{\varphi,r}$	=	$\frac{1}{r}$
$\Gamma^{\varphi}{}_{\varphi,\vartheta}$	=	$\mathrm{Cot}[\vartheta]$

8.2 Pohyb částic v okolí Schwarzschildovy černé díry

Pohyb částic lze popsat pomocí integrálů pohybu – momentu hybnosti ℓ a energie $\mathcal{E}$; pro jednoduchost budeme všechny veličiny vztahovat k hmotnosti černé díry M (neboli také lze říci, že tuto hmotnost formálně položíme rovnu 1, i vzdálenost budeme uvádět v násobcích M v obvyklých geometrodynamických jednotkách). Charakter trajektorie při daném momentu hybnosti ℓ vhodně popisuje efektivní potenciál, namísto radiální souřadnice můžeme s výhodou použít její převrácenou hodnotu $u = 1/r$:

In[443]:=
```
V[u_, ℓ_] := -u + ℓ^2 u^2 / 2 - ℓ^2 u^3
```

Specifický moment hybnosti ℓ :

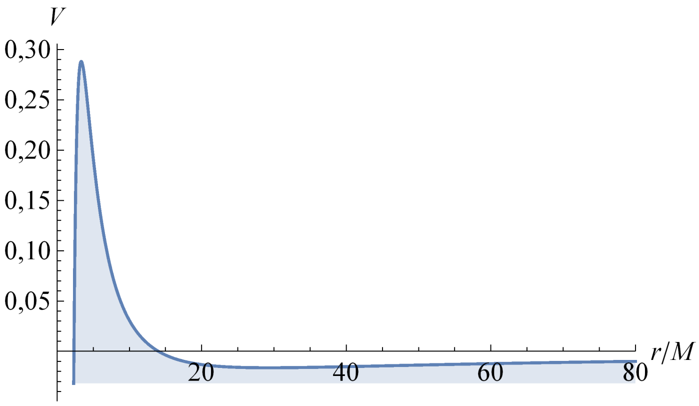
In[444]:=
```
ℓ = 3.3 Sqrt[3]
```

Out[444]= 5.71577

Efektivní potenciál vykreslíme jako funkce r/M a ℓ:

In[445]:=
```
veff = Style[Plot[V[1 / r, ℓ], {r, 2.3, 80},
    AxesLabel -> {"r/M", "V"}, PlotRange → {{0, 80}, All}, Filling → Bottom,
    BaseStyle → {FontFamily → "Times", FontSize → 14}], NumberPoint → ","]
```

Out[445]=



Extrémy potenciálu určují polohy stabilní (minimum) a labilní (maximum) kruhové orbity:

In[446]:=
```
dV[u_, ℓ] := -1 + ℓ^2 u  - 3 ℓ^2 u^2

maxmin = NSolve[dV[ex, ℓ] == 0, ex]
```

Out[447]= `{{ex → 0.0340969}, {ex → 0.299236}}`

Dosazením hodnot najdeme i minimální a maximální hodnotu efektivního potenciálu pro volbu energie částice.

In[448]:=
```
vmin = V[ex /. maxmin[[1]], ℓ]
```

Out[448]= `-0.0164009`

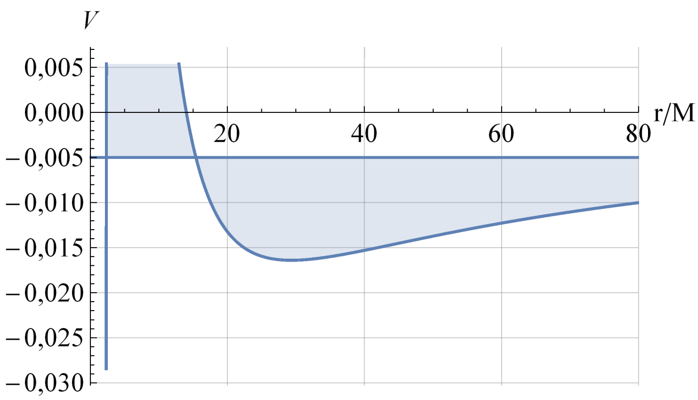
In[449]:=
```
vmax = V[ex /. maxmin[[2]], ℓ]
```

Out[449]= `0.288068`

Vykreslení potenciálu i s hladinou energie:

In[450]:=
```
ℰ = -0.005;
veff = Plot[V[1 / r, ℓ], {r, 2.1, 80}, AxesLabel → {"r/M", "V"},
    PlotRange → {{0, 80}, Automatic}, Filling → ℰ, GridLines → Automatic,
    BaseStyle → {FontFamily → "Times", FontSize → 14}];
sce := Plot[ℰ, {r, 0, 80}, DisplayFunction → Identity];
Style[Show[veff, sce, DisplayFunction → $DisplayFunction], NumberPoint → ","]
```

Out[453]=



Najdeme vzdálenosti, v nichž je zvolená energie částice rovna efektivnímu potenciálu a přepočteme z proměnné *u* na *r*:

In[454]:=
```
soln = NSolve[V[tp, ℓ] == ℰ, tp]
tp1 = 1 / tp /. soln[[1]]
tp2 = 1 / tp /. soln[[2]]
tp3 = 1 / tp /. soln[[3]]
```

Out[454]= {{tp → 0.00548628}, {tp → 0.0649388}, {tp → 0.429575}}

Out[455]= 182.273

Out[456]= 15.3991

Out[457]= 2.32788

Nyní zvolíme počáteční polohu, z energie a momentu hybnosti dopočítáme radiální rychlost.

In[458]:=
```
r0 = 25;
v0 = -Sqrt[2 * ℰ + 1 - (1 - 2 / r0) * (1 + ℓ^2 / r0^2)]
```

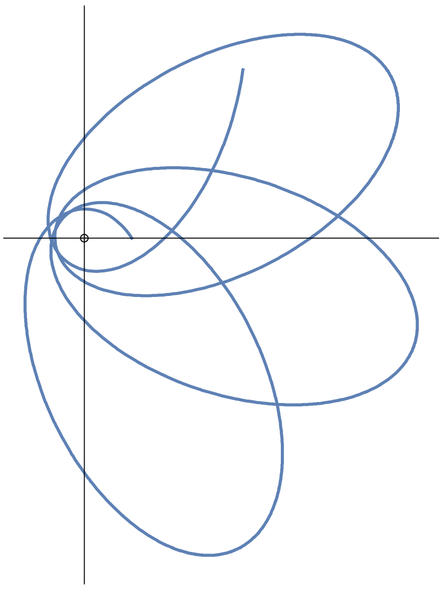
Out[459]= -0.148019

Nyní budeme řešit pohybovou rovnici a řešení vykreslíme spolu s vyznačením hranice horizontu černé díry ($r = 2M$).

In[460]:=
```
reseni =
  NDSolve[{rp''[t] == -1 / rp[t]^2 + ℓ^2 / rp[t]^3 - 3 * ℓ^2 / rp[t]^4, phi'[t] == ℓ / rp[t]^2,
    rp[0] == r0, rp'[0] == v0, phi[0] == 0}, {rp, phi}, {t, 0, 20000}];
```

In[461]:=
```
kr1 = ParametricPlot[Evaluate[{rp[t] * Cos[phi[t]], rp[t] * Sin[phi[t]]} /. reseni],
   {t, 0, 20000}, PlotStyle → {Thickness[0.007]}, Ticks → None];
kr2 = Graphics[Circle[{0, 0}, 2]];
kr3 = Graphics[{LightGray, Disk[{0, 0}, 2]}];
```

In[464]:=
```
Show[{kr1, kr3, kr2}, PlotRange → All]
```

Out[464]=


Vidíme, že výsledkem je neuzavřená vázaná trajektorie.

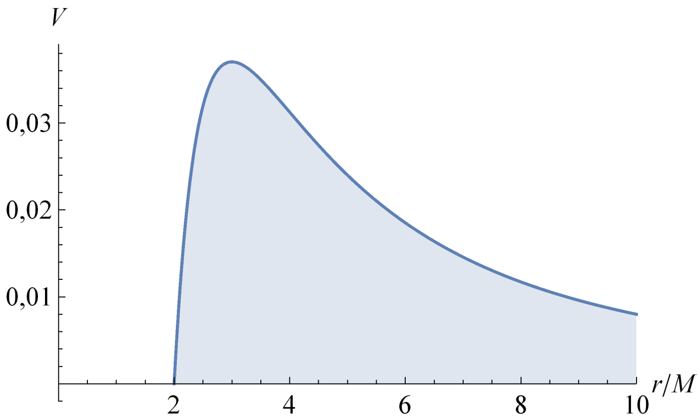
8.3 Pohyb fotonů v okolí Schwarzschildovy černé díry

Pro fotony (obecně částice s nulovou klidovou hmotností) má efektivní potenciál tvar:

In[465]:=
```
V[u_] := u^2 * (1 - 2 * u)
```

Vykreslíme jej jako funkci *r/M* :

In[466]:=
```
veff = Style[Plot[V[1 / r], {r, 2, 10},
    AxesLabel -> {"r/M", "V"}, PlotRange → {{0, 10}, All}, Filling → Bottom,
    BaseStyle → {FontFamily → "Times", FontSize → 14}], NumberPoint → ","]
```

Out[466]=


Analogicky jako v případě částic najdeme extrém efektivního potenciálu (odpovídá nestabilní tzv. kruhové fotonové orbitě, po níž fotony mohou obíhat dokola kolem černé díry)

In[467]:=
```
dV[u_] := 2 u  - 6 u^2

maxmin = NSolve[dV[ex] == 0, ex]
```

Out[468]= {{ex → 0.}, {ex → 0.333333}}

Dopočítáme hodnotu efektivního potenciálu v maximu

In[469]:=
```
vmax = V[ex /. maxmin[[2]]]
```

Out[469]= 0.037037

Poté zvolíme impaktní parametr fotonu a vykreslíme odpovídající průsečíky s efektivním potenciálem

In[470]:=
```
b = 10;
veff = Plot[V[1 / r], {r, 2, 10}, AxesLabel -> {"r/M", "V"},
    PlotRange → {{0, 10}, {1.2 * vmin, 1.2 * vmax}}, Filling → 1 / b^2,
    GridLines → Automatic, BaseStyle → {FontFamily → "Times", FontSize → 14}];
sce := Plot[1 / b^2, {r, 0, 10}, DisplayFunction → Identity];
Style[Show[veff, sce, DisplayFunction → $DisplayFunction], NumberPoint → ","]
```

Out[473]=


Nyní průsečíky najdeme numericky, podle grafu bude největší z nich odpovídat maximálnímu přiblížení fotonu, který dopadá z velké vzdálenosti, k černé díře:

In[474]:=
```
soln = NSolve[V[up] == 1 / b^2, up]
rp1 = 1 / up /. soln[[1]]
rp2 = 1 / up /. soln[[2]]
rp3 = 1 / up /. soln[[3]]
```

Out[474]= {{up → -0.0919089}, {up → 0.113781}, {up → 0.478128}}

Out[475]= -10.8803

Out[476]= 8.78885

Out[477]= 2.09149

Nakonec zvolíme počáteční polohu, dopočítáme radiální rychlost a úhel, z něhož foton přichází a horní časovou mez pro integraci:

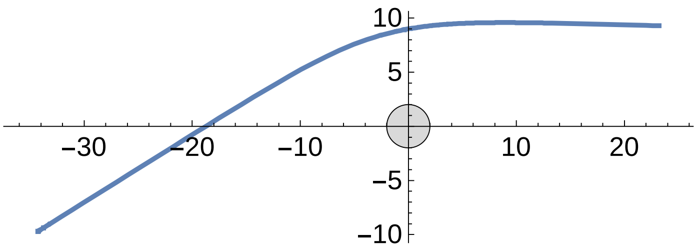
In[478]:=
```
r0 = 25;
v0 = -Sqrt[1 / b^2 - V[1 / r0]]
phi0 = ArcTan[r0, b]
tm = 600;
```

Out[479]= $-\frac{\sqrt{533}}{250}$

Out[480]= $\text{ArcTan}\left[\frac{2}{5}\right]$

Potom numericky vyřešíme pohybovou rovnici a vykreslíme trajektorii fotonu:

In[482]:=
```
reseni = NDSolve[{rp''[t] == 1 / rp[t]^3 - 3 / rp[t]^4, phi'[t] == 1 / rp[t]^2,
     rp[0] == r0, rp'[0] == v0, phi[0] == phi0}, {rp, phi}, {t, 0, tm}];
kr1 = ParametricPlot[Evaluate[{rp[t] * Cos[phi[t]], rp[t] * Sin[phi[t]]} /. reseni],
    {t, 0, tm}, PlotStyle → {Thickness[0.007]}];
kr2 = Graphics[Circle[{0, 0}, 2]];
kr3 = Graphics[{LightGray, Disk[{0, 0}, 2]}];
Show[{kr1, kr3, kr2}, PlotRange → All, BaseStyle → {FontSize → 14}]
```

Out[486]=


V souvislosti se studiem ohybu světelných paprsků v okolí černé díry je zajímavé spočítat změnu úhlu v rovině trajektorie pro daný foton případně ji převést na stupně:

In[487]:=
```
Δϕ = 2 * NIntegrate[1 / (r^2 * Sqrt[1 / b^2 - V[1 / r]]), {r, Max[rp1, rp2, rp3], ∞}]
Δϕ / Degree
```

Out[487]= $3.73199 - 6.14959 \times 10^{-11}\, i$

Out[488]= $213.827 - 3.52345 \times 10^{-9}\, i$

9 Závěr

Anotace

Publikace se věnuje využití programu *Mathematica* (verze 10.0) při řešení náročnějších fyzikálních problémů. V textu jsou vybrány některé zajímavé úlohy, které jsou zpracovány a popsány s využitím programu *Mathematica*. Matematika je dnes nezbytný obor pro studium v přírodovědných disciplínách a odpovídajících učitelských oborech. V době výkonných počítačů, notebooků či moderních tabletů je nezbytné zahrnout tyto progresivní prostředky do výuky a vést studenty k jejich smysluplnému a efektivnímu využívání. Díky moderním technologiím vzniká prostor k řešení náročnějších problémů. Implementace profesionálního matematického softwaru *Mathematica* přispívá ke zkvalitnění vyučovacího procesu dnes velmi náročných oborů, jako je fyzika, biologie i chemie.

Publikace *Software Mathematica pro fyziky* je rozdělena do osmi kapitol včetně úvodu, kde jsou shrnuta základní pravidla pro práci se softwarem *Mathematica*. V druhé kapitole jsou uvedeny některé fyzikální úlohy, které lze v softwaru *Mathematica* jednoduše řešit pomocí hledání lokálních extrémů funkcí. Následující dvě kapitoly textu se zaměřují na zpracování dat jednoduchých měření, např. zkoumání závislosti elektrického odporu na teplotě, analýza závislosti tlaku sytých vodních par na teplotě či zpracování dat z měření tíhového zrychlení pomocí matematického a reverzního kyvadla. Tyto příklady a výpočty lze využívat v rámci praktických měření v laboratoři fyziky. V kapitole číslo pět je uvedeno několik příkladů a aplikací při řešení problémů z oblasti klasické mechaniky, např. Keplerova úloha, Buquoyova úloha a Van der Polův oscilátor. Další kapitola se zabývá interpretací výsledků řešených problémů z teorie elektromagnetického pole. Pozornost je věnována vybraným aplikacím v elektrostatice a v nestacionárním elektromagnetickém poli. Kromě klasické mechaniky jsou řešeny i otázky z moderní mechaniky, např. Planckův vyzařovací zákon, vývoj hustoty pravděpodobnosti částice v jednorozměrné nekonečně hluboké potenciálové jámě nebo vývoj hustoty pravděpodobnosti lineárního harmonického oscilátoru. Díky profesionálnímu softwaru je jeden oddíl textu věnován modelování fraktálů ve 2D i 3D rozměru, což by bez použití kvalitního programu nebylo nikdy možné realizovat. Poslední kapitola Obecná teorie relativity se zabývá výpočty Christoffelových symbolů, Ricciho a Einsteinova tenzoru pro daný metrický tenzor nebo pohybem částic v okolí Schwarzschildovy černé díry.

Závěr

Software *Mathematica* je velmi výkonný matematický program, vhodný pro studenty a učitele fyziky, kteří chtějí rychle a efektivnì řešit náročnější fyzikální problémy. Software *Mathematica* obsahuje téměř 5000 funkcí, které umožňují řešit složité technické výpočty, vizualizovat reálné naměřené hodnoty nebo srovnávat a analyzovat data získaná z internetu. Publikace *Software Mathematica pro fyziky* se věnuje využití programu *Mathematica* (verze 10.0). Text publikace se snaží prohloubit a rozšířit znalosti čtenáře z učiva vysokoškolské fyziky. V jednotlivých kapitolách této publikace jsou popsány příklady z oblasti klasické mechaniky, moderní fyziky, obecné teorie relativity nebo teorie elektromagnetického pole. V publikaci *Software Mathematica pro fyziky* jsou uvedeny také kapitoly, které se věnují fyzikálním úlohám, které vedou k hledání lokálních extrémù funkcí nebo kapitoly popisují zpracování dat reálných měření.

Summary

Software *Mathematica* is a mathematical program with a very high output, suitable for students and teachers of physics looking for a quick and effective solution of challenging problems in the field of physics. Software *Mathematica* has almost 5000 functions that allow solving complex engineering calculations, visualizing real measured values or comparing and analyzing data obtained from the Internet. Publication *Software Mathematica for physics* is devoted to the usage of *Mathematica* (version 10.0). The text of the publication is aimed to enhance readers' knowledge of the university physics curriculum. The individual chapters of the publication describe the examples in the field of classical mechanics, modern physics, theory of general relativity and electromagnetic theory. The publication *Software Mathematica for physics* also contains some chapters dealing with the physical tasks that lead to seeking of local extremes of a function or chapters describing the real measurement data processing.

10 Použitá literatura

BAJER, J. *Mechanika* 2. Olomouc: Univerzita Palackého, 2004.
BEISER, A. *Úvod do moderní fyziky*. Praha: Academia, 1978.
BRDIČKA, M., HLADÍK, A. *Teoretická mechanika*. Praha: Academia, 1987.
BROŽ, J., ROSKOVEC, V., VALOUCH, M. *Fyzikální a matematické tabulky*. Praha: SNTL, 1980.
HLAVIČKA, A. a kol. *Fyzika pro pedagogické fakulty – 1. díl*. Praha: SPN, 1971.
Homepage of Wolfram Research [online]. Dostupné z: http://www.wolfram.com.
HORÁK, Z., KRUPKA, F. *Fyzika. Příručka pro vysoké školy technického směru*. Praha: SNTL, 1981.
HOSTE, J. *Mathematica DeMYSTiFied*. New York: McGraw-Hill Companies, 2009.
JAREŠOVÁ, M., VOLF, I. *Diferenciální počet ve fyzice* [online]. Dostupné z: http://fyzikalniolympiada.cz/texty/matematika/difpoc.pdf.
MIKULČÁK, J., CHARVÁT, J., MACHÁČEK, M., ZEMÁNEK, F. *Matematické, fyzikální a chemické tabulky a vzorce pro střední školy*. Praha: Prometheus, 2009.
OPATRNÝ, T. a kol. *Základy moderní fyziky*. Olomouc: Univerzita Palackého, 2013. Dostupné z: http://mofy.upol.cz/vystupy/02_texty/modul_zmf.pdf.
REICHL J. *Encyklopedie fyziky* [online]. Dostupné z: http://fyzika.jreichl.com/.
ŘÍHA, J. a kol. *Software Mathematica v přírodních vědách a ekonomii*. Olomouc: Univerzita Palackého, 2012.
SKÁLA, L. *Úvod do kvantové mechaniky*. Praha: Academia, 2005.
STROGATZ, S. H. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Colorado: Westview Press, 2000.
SVOBODA, E. a kol. *Přehled středoškolské fyziky*. Praha: Prometheus, 1996.
ŠEDIVÝ, P. *Teplotní závislosti fyzikálních veličin* [online]. Dostupné z: http://fyzikalniolympiada.cz/texty/teplota.pdf.
ŠEDIVÝ, P., VOLF, I., HORÁKOVÁ, R. *Harmonické kmity mechanických soustav* [online]. Dostupné z: http://fyzikalniolympiada.cz/texty/kmity.pdf.
ŠÍMA, V. a PODOLSKÝ, J. *Buquoyova úloha*. PMFA, roč. 51, č. 3, s. 177–186, 2006.
TELFORD, W. M., GELDART, L. P., SHERIFF, R. E. *Applied Geophysics –2nd ed.*Cambridge: Cambridge University Press, 2004.
The Mandelbrot Set [online]. Dostupné z: http://facstaff.unca.edu/mcmcclur/mathematicaGraphics/Mandelbrot/.
VYBÍRAL, B. *Zpracování dat fyzikálních měření* [online]. Dostupné z: http://fyzikalniolympiada.cz/texty/mereni.pdf.
WolframAlpha [online]. Dostupné z: http://www.wolframalpha.com.
WolframAlpha Blog [online]. Dostupné z: http://blog.wolframalpha.com.

Mgr. František Látal, Ph.D. a kolektiv

Software Mathematica pro fyziky

Výkonný redaktor Prof. RNDr. Zdeněk Dvořák, DrSc., Ph.D.
Odpovědná redaktorka Mgr. Jana Kreiselová
Technická redakce autor

Určeno pro studenty a akademické pracovníky VŠ

Vydala a vytiskla Univerzita Palackého v Olomouci
Křížkovského 8, 771 47 Olomouc
www.upol.cz/vup
e-mail: vup@upol.cz

Olomouc 2015
1. vydání

z. č. 2015/0077

Edice - Odborné publikace

ISBN 978-80-244-4470-3

Neprodejná publikace