České vysoké učení technické v Praze

Fakulta jaderná a fyzikálně inženýrská

Katedra matematiky

Propagátor časově závislého harmonického oscilátoru

Bakalářská práce

Autor práce: **Pavel Neškudla**

Školitel: **Prof. Ing. Pavel Šťovíček, DrSc.**

Školní rok: **2006/2007**

Oficiální zadání

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

Praha, 25.5.2007 Pavel Neškudla

Děkuji panu Prof. Ing. Pavlu Šťovíčkovi, DrSc. za cenné rady, připomínky a za jeho trpělivé a ochotné vedení práce.

Název práce:

Propagátor časově závislého harmonického oscilátoru

Autor:

Pavel Neškudla

Obor:

Inženýrská informatika

Abstrakt:

Během práce je používán systém Mathematica pro ověřování vztahů v Lieových algebrách a k úpravě vybraných pojmů kvantové mechaniky. Konkrétněji tu je uveden program vyjadřující Baker-Campbell-Hausdorffovu formuli a upravován tvar propagátoru v případě časově závislého harmonického oscilátoru.

Klíčová slova:

propagátor, Baker-Campbell-Hausdorffova formule

Title:

Propagator of a time-dependent harmonic oscillator

Author:

Pavel Neškudla

Abstract:

In this work there is used the system Mathematica to verify the relations in Lie algebras and to edit some terms of the quantum mechanics. To be more specific, there is a program for expressing the Baker-Campbell-Hausdorff‘s formula introduced. There is also edited a form of propagator in case of the time-dependent harmonic oscillator.

Keywords:

propagator, Baker-Campbell-Hausdorff formula

Obsah

Obsah vztahů

Úvod

Cílem této bakalářské práce bylo seznámit se s počítačovým algebraickým systémem *Mathematica* a aplikovat tento systém na vybrané problémy. *Mathematica* je systém, který slouží k manipulaci s matematickými výrazy, k jejich úpravě, symbolickým i numerickým výpočtům a v neposlední řadě také k prezentaci získaných výsledků.

Základní principy systému *Mathematica* jsem si osvojil na matematických operacích s Lieovými algebrami, které byly také jedním z témat, se kterými jsem se v průběhu své prace seznámil.

Následujícím úkolem byla aplikace získaných znalostí, která spočívala v použití systému *Mathematica* k vyjádření explicitního tvaru propagátoru v případě časově závislého harmonického oscilátoru.

1. Lieovy algebry

1.1. Definice Lieovy algebry

V této první kapitole se zaměřím na základy Lieových algeber. Nejprve uvedu základní definici.

■ Definice 1:

Lieova algebra $\mathfrak{g}$ nad tělesem T je uspořádaná dvojice $\mathfrak{g} = (V, [.,.])$, kde V je vektorový prostor nad tělesem T a $[.,.] : V \times V \to V$ je binární operace (nazývaná **Lieova závorka**), která má následující vlastnosti:

1) **bilinearita**

$(\forall \alpha, \beta \in T, \forall x, y, z \in V)\ ([\alpha x + \beta y, z] = \alpha[x, z] + \beta[y, z])$

$(\forall \alpha, \beta \in T, \forall x, y, z \in V)\ ([z, \alpha x + \beta y] = \alpha[z, x] + \beta[z, y])$

2) **antisymetrie**

$(\forall x, y \in V)\ ([x, y] = -[y, x])$

3) **Jacobiho identita**

$(\forall x, y, z \in V)\ ([x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0)$

Poznámka: Důsledkem 2) je vztah $(\forall x \in V)\ ([x, x] = 0)$.

1.2. Ověřování Jacobiho identity

Nyní se budu zabývat třetím bodem z definice Lieovy algebry. Platí totiž, že

Lieovu algebru plně definují:
1) bazické prvky generující vektorový prostor
2) Lieovy závorky mezi těmito bazickými prvky ve tvaru lineární kombinace bazických prvků, kde koeficienty jsou voleny tak, aby Lieova závorka byla antisymetrická a splňovala Jacobiho identitu.

Nechť mám tedy zadány prvky generující vektorový prostor a Lieovy závorky mezi těmito bazickými prvky ve tvaru lineární kombinace bazických prvků.

Budu tedy ověřovat, zda platí Jacobiho identity pro tyto závorky. Pokud totiž platí, pak se jedná o správně zadanou Lieovu algebru.

Lieova závorka `[A, B]` je ve výpočtech ve zbývající části práce značena `k[A, B]`. Takto jsem si tuto operaci pojmenoval pro použití v systému *Mathematica*.

1.2.1. Funkce pro ověření Jacobiho identity

■ 1., 2. a 3. vlastnost z definice 1 převedeny do řeči systému *Mathematica*

```
pLieAlg = {k[A_, A_] → 0,
   k[A_ + B_, D_] → k[A, D] + k[B, D],
   k[A_, B_ + C_] → k[A, B] + k[A, C],
   k[
     (m : (_Integer | _Rational | _Real | α | β | χ | δ | ϵ | ϕ | γ) |
         a | b | c | d | e | f | g) A_, B_] → m k[A, B],
   k[A_,
     (m : (_Integer | _Rational | _Real | α | β | χ | δ | ϵ | ϕ | γ) |
         a | b | c | d | e | f | g) B_] → m k[A, B],
   k[0, A_] → 0, k[A_, 0] → 0 };
```

■ Jacobiho identita

```
jacobi[u_, v_, w_] = k[u, k[v, w]] + k[v, k[w, u]] + k[w, k[u, v]];
```

■ Samotná funkce, která ověřuje platnost Jacobiho identity

```
ZkontrolujPlatnostJacobihoIdentity[bazovePrvky_, pVztahy_,
  bPrint_] :=
 Block[{},
  pocetBazi = Dimensions[bazovePrvky][[1]];
  p = 0; (*počítadlo kombinací*)
  suma = 0; (*součet  Jacobiho identity*)
  vysl = 0; (*výsledky po dosazení do Jacobiho identity*)
  For [i1 = 1, i1 ≤ pocetBazi, i1++, (*iterace v A*)
   For[ i2 = i1 + 1, i2 ≤ pocetBazi, i2++, (*iterace v B*)
    For [i3 = i2 + 1, i3 ≤ pocetBazi, i3++, (*iterace v C*)
     p++;
     (*Vytvoření substitučního pravidla pro dosazovaní
       do identity. Za A,B,C se dosazují prvky báze*)
     pSubstABC = {A → bazovePrvky[[i1]], B → bazovePrvky[[i2]],
       C → bazovePrvky[[i3]]};
     (*Nyní následuje samotná úprava  identity do co
       nejjednoduššího tvaru*)
     vysl =
      jacobi[A, B, C] /. pSubstABC /. pVztahy /. pLieAlg /. pVztahy /.
           pLieAlg /. pLieAlg /. pVztahy /. pLieAlg // Expand;
     suma = suma + Abs[vysl];
     (*sčítání jednotlivých výsledků v absolutní hodnotě
       do proměnné suma*)
     If[bPrint == 1,
      Print[p, ":", jacobi[A, B, C] /. pSubstABC, " = ", vysl];];
    ];
   ];
  ];
  If[suma == 0,
   Print["Součet je ", suma, ". ",
     StyleForm["Jacobiho identita platí.",
      FontColor → RGBColor[0, 0.8, 0] ] ];,
   Print ["Součet je ", suma // Simplify, ". ",
     StyleForm["Jacobiho identita neplatí.",
      FontColor → RGBColor[0.9, 0, 0] ] ];,
   Print ["Součet je ", suma // Simplify, ". ",
     StyleForm["Jacobiho identita neplatí.",
      FontColor → RGBColor[0.9, 0, 0] ] ];
  ];
 ];
```

Proměnné vstupující přímo do této funkce jsou **bazovePrvk**y, **pVztah**y a **bPrint**. Proměnná **bazovePrvky** obsahuje seznam prvků generující vektorový prostor. Proměnná **pVztahy** obsahuje definované Lieovy závorky mezi těmito bázovými prvky v požadovaném tvaru lineárních kombinací. Tento seznam musí tyto Lieovy závorky obsahovat ve formě přepisovacích pravidel typu `[A, B] → -C` (čtu Lieovu závorku A B přepiš na C). Poslední proměnná **bPrint** má pouze informativní význam. Pokud se rovná jedné, pak se budou v průběhu výpočtu zobrazovat jednotlivé kalkulované vztahy. Pokud

se nerovná jedné, pak se tyto vztahy nezobrazí. Princip funkce je následující. Vychází se z výrazu `[A, [B, C]] + [B, [C, A]] + [C, [A, B]]`, do kterého se dosazují za `A, B, C` postupně všechny prvky báze, které jsou v seznamu **bazovePrvky**. A k úpravě Jacobiho identity se používá vyjádření Lieových závorek pro prvky báze. Tyto závorky jsou v proměnné **pVztahy**.

1.2.2. Použití funkce - Lieova algebra g

Nyní naprogramovanou funkci využiji pro ověření konkrétních Lieových algeber. Nejprve ověřím speciální případ Lieovy algebry, se kterou se budu setkávat i nadále v této práci. Poté zkontroluji zadání algebry načtené z externího souboru, který je vytvořen pomocí jednoduché aplikace. V dalším průběhu práce budu pracovat s jednotkovým prvkem `I`, který budu pro systém *Mathematica* označovat `II`, protože `I` v *Mathematice* označuje imaginární jednotku.

■ Lieova algebra g

Definuji Lieovu algebru g (nad tělesem $\mathbb{C}$) s bází `{I, P, Q, U, V, W}` následujícími Lieovými závorkami:
`[P,Q]=II, [P,II]=0, [Q,II]=0, [P,U]=0, [P,V]=2Q, [P,W]=P, [Q,U]=-2P, [Q,V]=0, [Q,W]=-Q, [II,U]=0, [II,V]=0, [II,W]=0, [U,V]=-2II+4W, [U,W]=2U, [V,W]=-2V`

Tato Lieova algebra g souvisí s tvarem propagátoru pro harmonický oscilátor. Tedy nadále budu pod Lieovou algebrou g rozumět tuto právě definovanou algebru.

Pro systém *Mathematica* tyto skutečnosti reflektuji následovně:

```
PrvkyBaze = {II, P, Q, U, V, W};

DefRel = {k[P, Q] == II, k[P, II] == 0, k[Q, II] == 0, k[P, U] == 0,
   k[P, V] == 2 Q, k[P, W] == P, k[Q, U] == -2 P, k[Q, V] == 0,
   k[Q, W] == -Q, k[II, U] == 0, k[II, V] == 0, k[II, W] == 0,
   k[U, V] == -2 II + 4 W, k[U, W] == 2 U, k[V, W] == -2 V};
```

Vysvětlivky: proměnná **PrvkyBaze** obsahuje seznam bazických prvků
proměnná **DefRel** obsahuje Lieovy závorky mezi prvky báze

Lieovu závorku stačí zadat vždy jen pro jednu z dvojice `[A, B]`, k ní příslušná Lieova závorka `[B, A]` je k ní antisymetrická z definice, tedy `[B, A] = -[A, B]`.

■ Vytvoření množiny pravidel jako parametru pVztahy pro funkci

Z množiny **DefRel** vytvořím množinu pravidel **pKomRel**, která danou Lieovu závorku nahradí lineární kombinací bazických prvků, která jí odpovídá. Tato množina je tvořena dvěma částmi. První část je množina **DefRel**, v níž jsou pouze rovnosti změněny na přepisovací pravidla. Druhou částí jsou Lieovy závorky antisymetrické k závorkám v množině **DefRel** a jsou opět převedeny do tvaru pravidel. Vše je zřejmé z následujícího zápisu.

```
pKomRel =
 {DefRel /. {Equal[X_, Y_] → Rule[X, Y]},
   (*1. část množiny - tady se jen převedou definiční
      vztahy do formy pravidel*)
  DefRel /. {Equal[k[X_, Y_], C_] → Rule[k[Y, X], -C]}}
   (*2. část množiny - vytvoření antisymetrických pravidel*) //
 Flatten;
```

Nyní mám tedy definovanou bázi ve formě proměnné **PrvkyBaze** a také Lieovy závorky v požadovaném tvaru uložené v proměnné **pKomRel**. Tyto proměnné vložím na vstup definované funkce pro ověřování platnosti Jacobiho identity.

```
ZkontrolujPlatnostJacobihoIdentity[PrvkyBaze, pKomRel, 0];
```

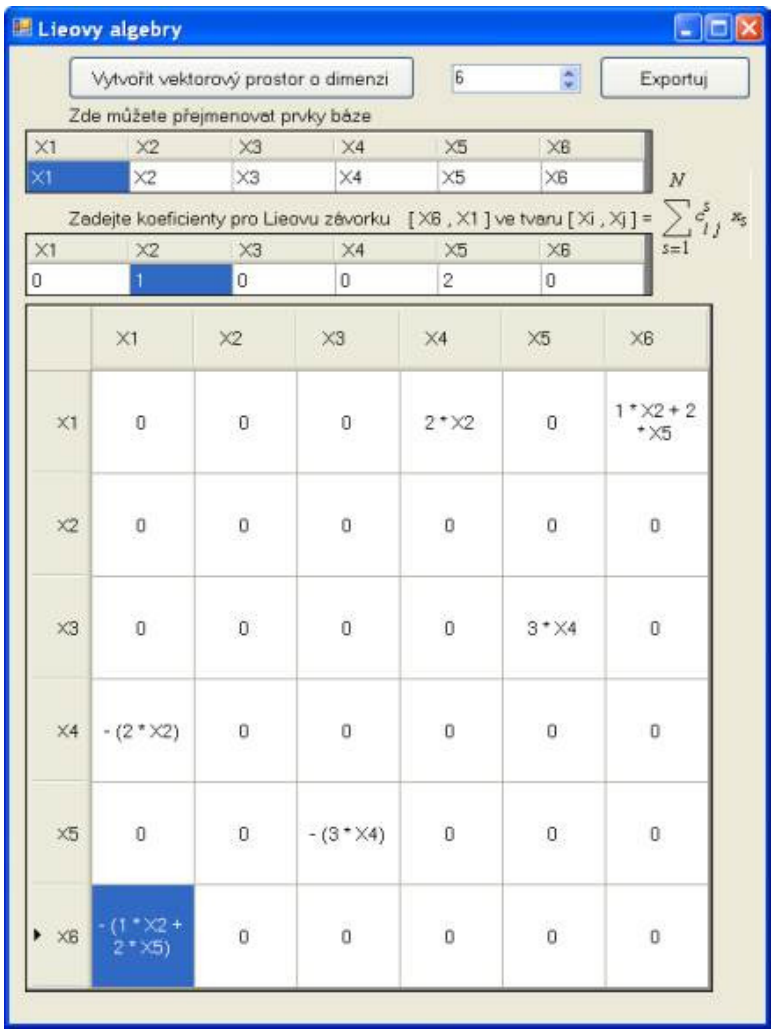
```
Součet je 0. Jacobiho identita platí.
```

■ Algebra načtená z externího souboru

■ Program pro definování Lieových algeber

Pro definování Lieových algeber určením prvků generujících vektorový prostor a Lieových závorek mezi nimi jsem naprogramoval malou aplikaci v jazyce C# pro prostředí .NET. Hlavní část zdrojového kódu je k nahlédnutí v dodatku 1.

Použití programu je velmi jednoduché. Postup práce s programem demonstruji na následujícím obrázku, který vyobrazuje jediné okno programu zobrazované ihned po spuštění.



(Okno aplikace Lieovy algebry)

V programu nejprve uživatel nastaví, kolik prvků bude mít báze, tedy dimenzi prostoru, na kterém bude definovat svou algebru. Na začátku se vygenerují názvy těchto bazických prvků ve tvaru `X1, X2, ..., Xn` pro n-rozměrný prostor. Tyto názvy lze libovolně měnit v příslušné tabulce (viz obrázek - Zde můžete přejmenovat prvky báze).

Po vytvoření vektorového prostoru a případné změně jmen prvků báze se uživateli objeví hlavní velká čtvercová tabulka. Ta má v popisu jednotlivých sloupců a řádků názvy prvků báze. Výběrem patřičného pole provede uživatel výběr Lieovy závorky, kterou bude definovat. Jednotlivé řádky odpovídají levému prvku v Lieově závorce. Sloupce odpovídají pravému prvku. Tak například pokud by měl uživatel tabulku jako je na obrázku, tak vybráním řádku s popiskem X6 a sloupce s popiskem X1 dá programu najevo, že chce definovat Lieovu závorku `[X6, X1]`.

Tato závorka se definuje pomocí druhé jednořádkové tabulky. Vzhledem k tomu, že Lieovy závorky mezi bazickými prvky mají být zadány ve tvaru lineární kombinace bazických prvků, tedy Lieova závorka $[x_i, x_j] = \sum_{s=1}^{n} c_{ij}^{s} x_s$, tak se v této tabulce zadávají jednotlivé koeficienty c_{ij}^{s} pro již vybrané i a j.

Po příslušném zadefinování relací výsledek exportujeme do souboru tlačítkem Exportuj.

Výstup programu je ve formátu :

$$\mathbf{PrvkyBazeExt} = \{X_1, X_2, \ldots, X_n\}$$

$$\mathbf{DefRelExt} = \{\ldots, [X_i, X_j] = \sum_{k=1}^{n} \alpha_k X_k, \ldots\},$$

kde $\alpha_i \in T$, $X_i \in V$, $i \in \{1, \ldots, n\}$.

■ Načtení příkladu ze souboru a ověření identity

V programu představeném v předešlém odstavci jsem vytvořil jednoduchou definici algebry, kterou jsem uložil do souboru "C:/algebra". Jednoduchým způsobem ji načtu a zobrazím proměnné, které jsou v ni obsaženy. To znamená proměnné **PrvkyBazeEx** a **DefRelExt**.

```
<< "C:/algebra";
```

A zobrazíme načtenou definici algebry

```
Print["PrvkyBazeExt = ", PrvkyBazeExt]
Print["DefRelExt = ", DefRelExt]
```

```
PrvkyBazeExt = {A, B, C}
```

```
DefRelExt = {k[B, A] == 2 B + C, k[C, A] == 0, k[C, B] == A}
```

Opět jako v případě algebry $\mathfrak{g}$ vytvořím celý soubor Lieových závorek mezi všemi prvky báze.

```
pKomRelExt =
 {DefRelExt /. {Equal[X_, Y_] → Rule[X, Y]},
   (*1. část množiny - tady se jen převedou definiční
      vztahy do formy pravidel*)
   DefRelExt /. {Equal[k[X_, Y_], C_] → Rule[k[Y, X], -C]}}
   (*2. část množiny - vytvoření antisymetrických pravidel*) //
  Flatten;
```

A zkontroluji, jestli tato algebra načtená z externího souboru, je, či není Lieovou.

```
ZkontrolujPlatnostJacobihoIdentity[PrvkyBazeExt, pKomRelExt,
 1]
```

```
1:k[A, k[B, C]] + k[B, k[C, A]] + k[C, k[A, B]] = -2 A
```

```
Součet je 2 Abs[A]. Jacobiho identita neplatí.
```

2. Exponenciální zobrazení a Baker-Campbell-Hausdorffova formule

2.1. Baker-Campbell-Hausdorffova formule

2.1.1. Obecný tvar Baker-Campbell-Hausdorffovy formule

■ Baker-Campbell-Hausdorffova (BCH) formule ([3])

Buď $\mathfrak{g}$ libovolná Lieova algebra a G příslušná souvislá a jednoduše souvislá Lieova grupa . Buď $exp : \mathfrak{g} \to G$ exponenciální zobrazení. Definujme $Z := X * Y = log\,(exp\,X\,exp\,Y)$. Pak Baker-Campbell-Hausdorffova formule pro výpočet Z má tvar:

$$Z = \sum_{n>0} \frac{(-1)^{n-1}}{n} \sum_{\substack{r_i+s_i>0 \\ 1\le i\le n}} \frac{\left(\sum_{i=1}^{n}(r_i+s_i)\right)^{-1}}{r_1!\,s_1!\ldots r_n!\,s_n!}\ (ad\,X)^{r_1}\,(ad\,Y)^{s_1}\ \ldots\ (ad\,X)^{r_n}\,(ad\,Y)^{s_{n-1}}\,Y,$$

kde zobrazení $(ad\,A) : \mathfrak{g} \to \mathfrak{g}$ je definováno vztahem $(ad\,A)\,B = [A, B]$. A pokud $s_n = 0$, tak se poslední tři členy interpretují jako $(ad\,X)^{r_n-1}\,X$. Tato formule platí, pokud suma na pravé straně konverguje. A ta konverguje vždy pro X a Y z jistého okolí počátku (pro X, Y z vektorového prostoru se počátkem myslí okolí nulového vektoru).

Poznámka : Prvních několik členů formule má tvar :

$$Z = X + Y + \frac{1}{2}[X, Y] + \frac{1}{12}[X, [X, Y]] - \frac{1}{12}[Y, [X, Y]] - \frac{1}{48}[Y, [X, [X, Y]]] - \frac{1}{48}[X, [Y, [X, Y]]] + \ldots$$

2.1.2. Baker-Campbell-Hausdorffova formule pro maticovou Lieovu grupu

V případě, že Lieova grupa z BCH formule je maticovou Lieovou grupou $G \subset GL\,(n, \mathbb{R})$ (tj. množina regulárních reálných matic typu $n \times n$) , pak příslušná Lieova algebra coby tečný prostor ke G v bodě I (nyní jednotková matice) je rovněž tvořena maticemi $n \times n$ a Lieova závorka je dána komutátorem $[X, Y] = XY - YX$.

Exponenciální relace je v tomto případě definována následovně.

■ Definice 2:

Buď $G \subset GL(n, \mathbb{R})$ Lieova grupa a $\mathfrak{g}$ k ní příslušná Lieova algebra. Zobrazení $\boldsymbol{exp} : \mathfrak{g} \to G$ definované předpisem:

$(\forall X \in \mathfrak{g}) \qquad \left(\exp X = e^X = I + \sum_{k=1} \frac{1}{k!} X^k \right)$ nazveme **exponenciálním zobrazením**.

S maticovými Lieovými grupami se budu v práci dále setkávat a budu také používat této exponenciální relace pro konkrétní výpočty v systému *Mathematica*. Proto si tyto definice zapíši do formy, s nimiž budu dále pracovat.

■ Převedení definice do systému *Mathematica*

Nejprve zapíši krátkou funkci pro získání výrazu nekomutativní mocniny. Pro nekomutativní násobení se v systému *Mathematica* užívá označení **.

```
ncPower[x_, n_Integer] :=
  If[n == 0, 1, Block[{y = x}, Do[y = y ** x, {n - 1}]; y]];
```

Ukázka použití.

```
ncPower[X, 5]
```

```
X ** X ** X ** X ** X
```

Nyní již zadefinuji exponenciální relaci z definice 3.

```
expRelace[X_, MAX_] := II + Sum[(1 / i !) * ncPower[X, i], {i, MAX}];
```

Opět malá ukázka použití.

```
expRelace[X, 3]
```

$$II + X + \frac{X ** X}{2} + \frac{X ** X ** X}{6}$$

Pokud bychom do BCH formule dosadili za G maticovou Lieovu grupu, pak by se tato formule zjednodušila a obdrželi bychom následující výraz.

■ Speciální případ Baker-Campbell-Hausdorffovy formule pro maticovou Lieovu grupu G

$$Z = \sum_{n>0} \frac{(-1)^{n-1}}{n} \sum_{\substack{r_i + s_i > 0 \\ 1 \le i \le n}} \frac{X^{r_1} Y^{s_1} \ldots X^{r_n} Y^{s_n}}{r_1 ! \, s_1 ! \ldots r_n ! \, s_n !} .$$

Poznámka: Výrazy prvního, druhého, třetího a čtvrtého řádu mají následující tvary:

$$z_1 = X + Y$$
$$z_2 = \frac{1}{2}\,(XY - YX)$$
$$z_3 = \frac{1}{12}\,(X^2 Y + XY^2 - 2\,XYX + Y^2 X + Y\,X^2 - 2\,YXY)$$
$$z_4 = \frac{1}{24}\,(X^2 Y^2 - 2\,XYXY - Y^2 X^2 + 2\,YXYX)$$

2.2. Vyjádření Baker-Campbell-Hausdorffovy formule do určitého řádu

Mým úkolem bylo vytvořit nějakou funkci, která by dokázala vypsat vzorec Baker-Campbell-Hausdorffovy formule z definice 4, který budu dále v této práci označovat jako formule. Chtěl jsem tedy vyjádřit formuli pro maticové Lieovy algebry do určitého řádu přesnosti.

S tímto problémem jsem se vypořádal následovně. Nejprve jsem napsal funkci v systému *Mathematica*, kterou jsem ale z důvodu rychlosti výpočtu mohl testovat jen do pátého řádu přesnosti.

Pak jsem se zajímal o to, jakých rychlostních zlepšení bych dosáhl, pokud bych tu samou funkci napsal v nějakém nízkoúrovňovém jazyce. Při tom jsem zjistil, že systém *Mathematica* je schopen využívat určité externí aplikace napsané v jazyce C. Takovou externí aplikaci se mi podařilo napsat a nakonec jsem ji umístil do balíku, který je jednoduše přístupný ze systému *Mathematica*. Tento výsledek považuji za nejdůležitější bod mé práce.

Pomocí aplikace nakonec vyjádřím formuli do 11. tého řádu přesnosti. Limitujícím faktorem při výpočtu je alespoň v případě mého počítače spíše velikost operační paměti, než jeho rychlost.

Neupravenou formu formule, která je čistě výstupem mé funkce napsané v jazyce C, jsem schopen vyjádřit přibližně po třiceti minutách výpočtů a s použitím asi 300 MB operační paměti až do řádu 13 (AMD Mobile Sempron 1,8 GHz, 640 MB RAM). Odhaduji, že výpočet 14. tého řádu by vyžadoval asi 1,2 GB paměti a trval by kolem tří hodin.

Ještě zmíním, že při hledání informací o tomto tématu jsem na internetu narazil na článek [2], kde je uveden jiný, efektivní algoritmus pro vyjádření formule. Tento článek jsem ale objevil až po naprogramování své funkce.

2.2.1. Vyjádření formule v systému *Mathematica*

Po mnoha hodinách přemýšlení a ladění jsem naprogramoval funkci v systému *Mathematica* na jehož vstupu byly právě proměnné X a Y a řád přesnosti. Proměnné X a Y odpovídají X a Y z formule. Výstupem funkce byl výraz, který reprezentoval formuli vyjádřenou s přesností do zadaného řádu. Tento výraz bylo možno běžnými algebraickými operacemi s využitím vlastností z definice Lieovy algebry upravit do skutečně požadovaného tvaru.

Nejtěžší část problému spočívala v druhé sumě ve formuli. Jedná se totiž o vícenásobnou sumu, jejíž násobnost je závislá na sčítacím indexu n první sumy. Pro připomenutí zde formuli znovu uvedu.

Nejtěžší část problému spočívala v druhé sumě ve formuli. Jedná se totiž o vícenásobnou sumu, jejíž násobnost je závislá na sčítacím indexu n první sumy. Pro připomenutí zde formuli znovu uvedu:

$$Z = \sum_{n>0} \frac{(-1)^{n-1}}{n} \sum_{\substack{r_i+s_i>0 \\ 1\le i\le n}} \frac{X^{r_1} Y^{s_1} \ldots X^{r_n} Y^{s_n}}{r_1!\, s_1! \ldots r_n!\, s_n!}$$

V první řadě v tomto vzorci není nikde explicitní závislost na požadované přesnosti. Musel jsem ji tedy nalézt a pomocí ní omezit obě sumy. Označím si M jako požadovaný řád přesnosti hledané formule. Prvním omezením je že n vystupující jako sčítací index ve vnější sumě je menší nebo roven M. Dalším omezením je, že vnitřní suma je omezena shora následujícím faktem. Suma přes všechny dvojice s_i a r_i je menší nebo rovna M. To znamená, že když sečtu všechna s_i a přičtu k výsledku ještě součet všech r_i, tak tento celkový součet je shora omezen M. Pokud bych totiž tyto hranice překročil, tak by se uvnitř vnitřní sumy objevilo násobení více než M prvků. Tím pádem bych překročil stanovený řád přesnosti, což by bylo zbytečné.

Celkem je omezení následující. Na začátku zadám M. Pro dané M volím $n = 1 \ldots M$. Dále pro dané n vyplývá, že indexů r_i je n a stejně tak indexů s_i je n. Za tyto indexy mohu volit libovolně tak, aby vždy $r_i + s_i > 0$ a zároveň $\sum_{i=1}^{n} r_i + \sum_{i=1}^{n} s_i < M$.

Způsob, kterým formuli v programu vyjadřuji vychází přímo ze vzorce a uvedených omezení. Podstatou algoritmu bylo iterování přes všechny možné hodnoty indexů s_i a r_i od nuly až po stanovená omezení. Těchto kombinací, jak indexy s_i a r_i zvolit, je ovšem velmi mnoho a implementace v systému *Mathematica* nebyla příliš efektivní. Jazyk systému *Mathematica* je totiž interpretovaný a proto jsou rozsáhlé iterace v jeho nekompilovaném kódu docela neefektivní. Formuli vyjádřenou s přesností pátého řádu jsem obdržel po několikahodinovém běhu programu.

2.2.2. Vyjádření formule pomocí jazyka C

Z předešlých důvodů pomalého běhu funkce v systému *Mathematica* jsem se tedy rozhodl napsat tuto funkci v nějakém nízkoúrovňovém jazyce. Zdrojový kód programu je obsažen v dodatku 2.

Začal jsem se zajímat o to, zda by systém *Mathematica* nějakým způsobem neumožňoval kompilaci funkcí psaných například v jazyce C++. Při studiu této problematiky jsem objevil jednu možnost jak toto v podstatě provést. *Mathematica* umí používat externí aplikace pomocí rozhraní, které se nazývá *Mathlink*. Avšak tyto programy musí být pro rozhraní *Mathlink* řádně uzpůsobeny. Omezením například je, že není možné využívat jazyk C++, ale pouze jeho předchůdce, jazyk C.

Nyní zde popíši, jakým způsobem jsem při vytváření aplikace v jazyce C, kterou lze v systému *Mathematica* využívat, postupoval. Hlavním výkonným kódem programu byla funkce z předcházející kapitoly přepsaná do jazyka C. Tento program jsem se pak ještě snažil zoptimalizovat. Hledal jsem způsoby, jak zrychlit algoritmus nejen samotného generování kombinací indexů s_i a r_i ale i části kódu přistupující k paměti a zapisující do ní jednotlivé části formule (To znamená členy sum a pak i řetězce při vytváření nekomutativního součinu uvnitř formule). Nakonec jsem dosáhl celkem velkého zrychlení. V přijatelném čase jsem na svém počítači dosáhl vyjádření formule s přesností do 13. tého řádu. Limitem pro vyjádření formule s vyšší přesností už nebyla rychlost výpočtu, ale velikost operační paměti mého počítače. Během výpočtu je totiž vzorec neupravovaný a obsahuje mnoho členů, které by se měli nakonec odečíst. Z tohoto důvodu vzorec 13. tého řádu zabírá přibližně 300 MB.

Velmi důležitým byl odhad velikosti potřebné paměti pro uchování výrazů jednotlivých částí formule a odhad délky celkového výsledku. Pro tento odhad je důležité spočítat, přes kolik různých kombinací indexů s_i a r_i se musí při výpočtu sčítat. To znamená, kolik takových kombinací vyhovuje hraničním podmínkám uvedeným v předcházející kapitole 2.2.1. Jakmile jsou totiž hraniční podmínky splněny, začne se v programu vytvářet další člen formule, který samozřejmě zabere nějaké to místo v paměti.

Následuje rekurentní vzorec pro určení počtu těchto kombinací. M značí řád přesnosti. Malé n značí n z formule.

$$P(n, M) = \sum_{k=1}^{M} P(n-1, M-k)\,(k+1), \quad P(0, 1) = 1$$

V tomto vzorci je skrytá následující věc. Jedná se o to, že si představím posloupnosti indexů s_i a r_i jakoby ve dvou řádcích psaných pod sebou. Nahoře nechť jsou psány indexy r_i a pod nimi indexy s_i. Tím tedy vznikne dvouřádková tabulka s n sloupci, protože $i = 1 \ldots n$. Z předpisu druhé sumy formule vyplývá, že v každém sloupci musí být součet minimálně jedna a zároveň (z omezujících podmínek) maximálně M.

Počet možností, jak dostat ve sloupečku součet právě k, je roven $(k + 1)$. Nechť

se zajímám o i-tý sloupeček. Za indexy s_i a r_i musím volit jednu z následujících kombinací.

$$r_i = 0,\ 1,\ \ 2,\ \ldots,\ k-1,\ k$$
$$s_i = k,\ k-1,\ k-2,\ \ldots,\ 1,\ 0$$

Představme si, že z naší tabulky vynechám právě p-tý sloupec. Pak vlastně zbytek sloupců tvoří novou tabulku s $n-1$ sloupci a novým maximem, které je teď rovno $M-k$. Protože k jsem již rozdělil mezi indexy s_p a r_p. Celkový počet možností, kterými lze tedy navolit indexy s_i a r_i je pak roven součtu všech možností pro různé varianty pevně voleného $k = 1 \ldots M$.

Následující blok kódu vyčísluje prvních M hodnot tohoto rekurentního vztahu. V podstatě se jedná o jiný zápis vztahu a to takový, který napočítává hodnoty od nejnižších při $k = 1$ až po nejvyšší při $k = M$.

```
MAX = 5;
p = Table[0, {i, 1, MAX}, {j, 1, MAX}];
Do[
 {Do[
   {
    Do[
     {If[(n - 1) == 0, {p[[n, M]] += (k + 1);},
       (*počáteční podmínky*)
       If[(n - 1) > (M - k), {(*nic se nepřičte*)},
        p[[n, M]] += p[[n - 1, M - k]] (k + 1);
       ];];},
     {k, 1, M}];
   },
   {n, 1, M}]},
 {M, 1, MAX}];
p = Transpose[p];
```

Pole, které je tímto krátkým programem vygenerováno jsem použil při programování programu Formule.exe pro odhad potřebného množství paměti při vyjadřování jednotlivých sčítanců vnější sumy formule. Za odhad jsem použil největší hodnotu, která může při daném maximu M nastat. Tuto hodnotu jsem korigoval s přihlédnutím na průměrnou odchylku skutečně spotřebované paměti od této maximální hodnoty tím, že ji násobím konstantou 0.6, abych ušetřil ještě nějaké, jinak zbytečně alokované, místo.

Výsledkem odhadu je, že alokace paměti pro výslednou formuli se pohybuje kolem 115 procent skutečně spotřebované paměti.

■ Jak napsat a zkompilovat kód jazyka C, aby byl zkompilovaný kód přístupný v systému *Mathematica* ?

Jak jsem již uvedl, systém *Mathematica* může používat externí aplikace pomocí rozhraní *Mathlink*, ale nemohou to být aplikace ledajaké. Už při jejich tvorbě se musí dodržet určité postupy, které umožní spojení aplikace s rozhraním *Mathlink*. Tyto postupy se v první řadě dotknou samotného zdrojového kódu.

■ Zdrojový kódu programu

První podmínkou je, že pokud chceme kód psát v nějakém jazyce z rodiny céčkovských jazyků, pak musíme použít jazyk C. Integrace s jazykem C++ není podporována, alespoň co se týče systému *Mathematica* ve verzi 5.0. Na začátku zdrojového kódu (soubor Formule.c) musí být uveden, kromě jiných, řádek **#include "mathlink.h" .**

Dále musí kód obsahovat deklaraci funkce, kterou budeme chtít používat. Deklarace musí být typu extern. V mém případě tedy kód obsahuje řádek: **extern char * Formule (char *strX, char *strY, int Max);**. Definice této funkce je uvedena až ke konci zdrojového kódu.

Tady se pozastavím nad tím, jaké typy v systému *Mathematice* odpovídají typům jazyka C. Například int v jazyce C je Integer v systému *Mathematica*. Pro moji funkci Formule chci v systému *Mathematica* použít jako první dva parametry řetězce tvarů "X" a "Y". Řetězec se v *Mathematice* označuje jako String a k němu odpovídající typ v jazyce C je char* (tedy ukazatel do paměti na první znak řetězce).

Nakonec kód musí obsahovat definici funkce main v následujícím tvaru:

```
int main(int argc, char* argv[])
{
    return MLMain(argc, argv);
}
```

Přes tuto funkci přistupuje Mathlink k naší exportované funkci. Kód v dodatku obsahuje ještě jiné tvary funkce main, které jsou jen speciálními případy této pro různé typy systémů (MACITOSH, WINDOWS, LINUX).

■ Vytvoření souboru s příponou .tm, který obsahuje informace o naší exportované funkci (tedy jen o funkci Formule)

Po napsání zdrojového kódu je nutné vytvořit soubor s příponou .tm. V tomto souboru musí být obsaženy funkce, které chceme přímo zpřístupnit přes rozhraní *Mathlink*.

Takto vypadá obsah souboru Formule.tm pro moji aplikaci:

char *Formule P((char *, char *, int));

:Begin:

:Function: Formule
:Pattern: Formule[strX_String,strY_String,Max_Integer]
:Arguments: { strX, strY, Max}
:ArgumentTypes: { String, String, Integer }
:ReturnType: String
:End:

:Evaluate: Funkce1::usage = "Formule[X_String, Y_String, M_Integer] gives the expression of Baker-Campbell-Hausdorff formula of elements X and Y to M-th order."

Soubor začíná uvedením deklarace funkce Formule ve tvaru jazyka C. P( ) je jakési makro, jehož úplný význam není pro základní princip důležité znát.

Mezi titulky **:Begin**: a **:End:** jsou uvedeny všechny důležité vlastnosti funkce, které systém *Mathematica* potřebuje pro její volání a pro manipulaci s její vrácenou hodnotou. Nejprve je za položkou :Function: uveden název funkce. Za **:Pattern:** (česky vzor) je její tvar v *Mathematice*. Tím je dáno jakým způsobem potom bude možné funkci ze systému *Mathematica* volat. Dále jsou uvedeny názvy jejích argumentů a jejich příslušné typy v systému *Mathematica*. A nakonec ještě jakého typu je vracená hodnota.

Na úplném konci souboru je značka **:Evaluate: :usage =** " ". Kde je v uvozovkách uvedena zpráva o tom, jak se má funkce používat a co provádí. Ta se objeví například při nesprávném použití funkce.

■ Kompilace připravených souborů a finální vytvoření aplikace

Poté, co jsou připraveny soubory Formule.c a Formule.tm, můžeme konečně vytvořit spustitelný soubor aplikace. Uvedu zde sekvenci příkazů, které z těchto dvou souborů nakonec vytvoří spustitelný soubor. V této sekvenci jsou postupně volány tři programy. Nejprve MPREP, který je součástí systému *Mathematica,* dále pak kompilátor jazyka C a linker.

Pro vytvoření spustitelného souboru aplikace jsem v adresáři se soubory Formule.c a Formule.tm spustil následující příkazy:

```
 "..\Microsoft Visual Studio 8\VC\bin\VCVARS32"  x86
SET CL=/nologo /c /DWIN32 /D_WINDOWS /W3 /ML /O2 /Oi /Ob2 /Ot
/DNDEBUG
SET LINK=/NOLOGO /SUBSYSTEM:windows /INCREMENTAL:no /PDB:NONE
kernel32.lib user32.lib gdi32.lib
MPREP Formule.tm -o Formuletm.c
CL Formule.c Formuletm.c
LINK  Formule.obj Formuletm.obj ml32i2m.lib /OUT:Formule.exe
```

Pomocí příkazů na první třech řádcích se nastaví vlastnosti kompilátoru a linkeru, které budou při překladu zdrojového kódu použity. SET CL nastavuje parametry kompilátoru. Například /O2 znamená, že se bude výsledný program optimalizovat pro rychlost. SET LINK nastavuje parametry linkeru, mezi nimiž jsou i knihovny, které se

mají při kompilaci přilinkovat.

Čtvrtý řádek osahuje příkaz pro spuštění programu MPREP, který vytvoří zdrojový kód v jazyce C ze souboru Formule.tm. Tím vznikne soubor Formuletm.c, který obsahuje kód pro spolupráci aplikace s rozhraním *Mathlink*. Nakonec se oba zdrojové soubory zkompilují a slinkují.

Výstupem je spustitelný soubor. V tomto případě Formule.exe.

■ Příklad použití aplikace Formule.exe v systému *Mathematica*

Následující příkazy předpokládají, že soubor Formule.exe je umístěný v kořenovém adresáři disku D. Použití tohoto souboru v systému *Mathematica* pak vypadá následovně.

```
link = Install["D:/Formule.exe"];

LinkPatterns[link]

 {Formule[strX_String, strY_String, Max_Integer]}

Formule["A", "B", 2]

 +1/1*(+A*1/1+A**A*1/2+B*1/1+A**B*1/1+B**B*
   1/2)-1/2*(+A**A*1/1+A**B*1/1+B**A*1/1+B**B*1/1)

LinkClose[link]
```

Je vidět, že nejprve je nutné aplikaci nainstalovat. To znamená, připravit ji k tomu, aby mohl systém *Mathematica* používat její funkce. K aplikacise tedy vytvoří takzvaný link. K tomu slouží funkce **Install**, jejíž parametrem je cesta k souboru aplikace v souborovém systému. V některých případech lze dokonce uvést pouze název souboru bez cesty, ale to ukáži až později v této kapitole.

Funkce **Linkpatterns[link]** vypíše funkce, které jsou z aplikace uživateli k dispozici.

Pak spustím jedinou funkci z aplikace, kterou je funkce Formule. Je zde vidět, že na vstupu funkce jsou dva řetězce "A" a "B" a požadovaný řád formule, který je v tomto příkladu 2. Výstupem této funkce je řetězec, který obsahuje zápis součtu dvou členů vnější sumy vyjádřených z formule. Tento výstup zde ještě není upravený, obsahuje členy navíc, které lze dalšími úpravami spolu sečíst a výraz celkově zkrátit. Těmito dodatečnými úpravami se zabývá balíček, o kterém budu psát dále v následující kapitole.

Nakonec se spojení s aplikací ukončí pomocí funkce **LinkClose[link]**.

2.2.3. Vytvoření balíku s aplikací

Potřebné soubory jsou k nalezení v dodatku 3.

Package (česky balík) je v *Mathematice* označení pro samostatný soubor, který obsahuje funkce, které spolu nějakým způsobem souvisí. Pokud pak v systému *Mathematica* tento package otevřeme (nahrajeme), můžeme všechny funkce z tohoto souboru jednoduše používat.

Mým cíle bylo upravit výsledek z externí funkce psané v jazyce C do přijatelnější podoby. Chtěl jsem, aby uživatel obdržel co nejkratší zápis formule a sám už ho nemusel nijak upravovat. Tedy, svoji externí aplikaci jsem vložil do package spolu s funkcí, která výstup z aplikace zjednoduší. Uživatel pak volá tuto funkci z balíčku, která již vrací upravený a zkrácený tvar formule.

Jak tedy postupovat při vytváření package? Nejprve se vytvoří typický soubor v *Mathematice* s příponou .nb. V tomto souboru je v mém případě následující text a kód.

This packgage mediates an access to the external program Formule.exe, that is programmed in C and contains a function Formule [A, B, m]. It returns the Baker-Campbell-Hausdorff formula of the elements A and B computed to the m-th order of precision.

```
BeginPackage["Propagator`BCHfle`"]
BCHfle::usage =
 "BCHfle[A_String, B_String,RAD_Integer] gives the
   expression of the Baker-Campbell-Hausdorff formula
   calculated to the RAD-th order."
Begin["`Private`"]
BCHfle[A_, B_, RAD_Integer] := Module[{fle},
   link = Install["Propagator/Formule.exe"];
   fle =
    ToExpression[StringReplace[Formule["X", "Y", RAD],
       {"X" → StringJoin["(", ToString[A], ")"],
        "Y" → StringJoin["(", ToString[B], ")"]}]] // Expand;
fle = fle // Expand //. {X^2 → X^2};

   LinkClose[link];
   fle];
End[]

EndPackage[]
```

Tento soubor se pak uloží pomocí volby menu File→Save As Special→Package Format. Tím se vytvoří soubor s příponou .m, kde budou všechny buňky zakomentovány. Jako výkonný kód se do tohoto package formátu uloží pouze buňky, které jsou označeny jako inicializační. Položka menu Cell→Cell Properties→Initialization Cell.

První část textu až před BeginPackage je pouze informativní, tedy není označena jako inicializační. Zde by mělo být uvedeno, které funkce balík nabízí.

Zbývající část textu musí být označena jako inicializační, protože se jedná o výkonný kód.

Na této výkonné části je vidět základní struktura souboru package v systému

Mathematica. Kód začíná funkcí **BeginPackage["Propagator`BCHfle`"]**, jejímž parametrem je název balíčku zakončený zpětným apostrofem `. Taková je konvence značení pro package. Název mého balíčku je BCHfle. Je obvyklé nazývat package stejně jako funkci, kterou obsahuje, případně nějak příznačně funkcím, které by obsahoval. Před názvem BCHfle je ještě název Propagator`, což značí jakousi hierarchii, že balík BCHfle je podstrom kořenu Propagator. To intuitivně odpovídá adresářové hierarchii. Systém *Mathematica* podle toho také balík nalezene, pokud bychom chtěli package načíst a použít v jakémkoliv dalším souboru v systému *Mathematica,* tak, že v adresáři Propagator hledá balík BCHfle. V systému *Mathematica* se řekne, že balík BCHfle je v kontextu Propagator.

Dále je zde krátká instrukce o tom, jak se funkce obsažená v balíčku používá, **BCHfle::usage**. Následuje samotná definice funkcí. Ta je uvedena uvnitř bloku **Begin["`Private`"]** a **End[]**. Tento blok určuje že uvnitř se pracuje v takzvaném kontextu Private. A celý kód končí funkcí **End[]**. Taková je tedy struktura souboru package.

A co vlastně dělá hlavní funkce kódu, tedy funkce BCHfle? Jejími vstupními parametry jsou **A** a **B** libovolné a **RAD** typu celé číslo. Uvnitř této funkce se volá můj program Formule.exe. Protože jsem svůj package umístil v kontextu Propagator a tedy tento balík je pak hledán v adresáři Propagator, tak je příhodné umístit i soubor Formule.exe do tohoto adresáře. Kde má ale být umístěn adresář Propagator určeno není. Proto musí být umístěn v některém z adresářů uvedených v proměnné prostředí $Path.

Program Formule.exe v balíku nainstaluje. Dále je zavolána funkce Formule s pevnými parametry typu String "X" a "Y" a právě **RAD** typu Integer, který značí požadovaný řád přesnosti, s kterým chceme formuli vypsat. Tím se tedy formule vyjádří v neupraveném tvaru řetězce, v kterém se následně provede nahrazení pevných symbolů "X" a "Y" za požadované výrazy **A** a **B** a celý tento řetězec se převede do formátu výrazu příkazem ToExpression. Tímto převodem se výraz částečně automaticky upraví. Odečtou se některé členy a celkově se tedy zkrátí. Pak se ještě provede úprava příkazem **Expand** a spojí se násobky do mocnin. Funkce nakonec vrátí celkem přijatelnou podobu formule s elementy A a B s požadovanou přesností řádu **RAD**.

Svůj balík jsem tedy uložil v Package formátu pod názvem BCHfle.m v podadresáři Propagator v jednom z adresářů proměnné **$Path** . Na tento balík se pak mohu odvolat z jiného souboru příkazem **<<Propagator`BCHfle`** a tím nahraju funkce, které jsou v něm obsažené.

Konkrétně jsem vše provedl takto. V adresáři ../mathematica/5.0/AddOns/ExtraPackages jsem vytvořil podadresář Propagator, do kterého jsem přesunul soubory BCHfle.m a Formule.exe.

2.2.4. Práce s package

Následujícím příkazem **<<Propagator`BCHfle`** se balíček v systému *Mathematica* inicializuj a dalším příkazem, **?BCHfle**, se zobrazí instruktáž k funkci.

```
<< Propagator`BCHfle`

?BCHfle
```

```
BCHfle[A_String, B_String,RAD_Integer]
  gives the expression of the Baker-Campbell-
  Hausdorff formula calculated to the RAD-th order.
```

Balíček je připraven k použití. Vyzkouším funkčnost balíčku požadavkem na vygenerování BCH formule pro prvky X a Y do čtvrtého řádu.

```
BCHfle[X, Y, 4]
```

$$X + Y + \frac{X ** Y}{2} - \frac{Y ** X}{2} + \frac{X ** X ** Y}{12} - \frac{X ** Y ** X}{6} + \frac{X ** Y ** Y}{12} + \frac{Y ** X ** X}{12} - \frac{Y ** X ** Y}{6} + \frac{Y ** Y ** X}{12} + \frac{1}{24} X ** X ** Y ** Y - \frac{1}{12} X ** Y ** X ** Y + \frac{1}{12} Y ** X ** Y ** X - \frac{1}{24} Y ** Y ** X ** X$$

Pro kontrolu, že výsledný vzorec pro formuli opravdu odpovídá skutečnosti přikládám znovu členy prvních čtyřech řádů z poznámky z kapitoly 2.1.2. (za výrazem speciálního tvaru BCH formule).

$$z_1 = X + Y$$
$$z_2 = \frac{1}{2}\,(XY - YX)$$
$$z_3 = \frac{1}{12}\,(X^2 Y + XY^2 - 2\,XYX + Y^2 X + Y X^2 - 2\,YXY)$$
$$z_4 = \frac{1}{24}\,(X^2 Y^2 - 2\,XYXY - Y^2 X^2 + 2\,YXYX)$$

2.3. Test funkčnosti programu vypisujícího formuli

Nyní ověřím, že formule, které generuji v připraveném balíku pomocí mnou naprogramované funkce jsou skutečně správně. Přitom budu vycházet z toho, že musí platit následující identita.

$$exp\,X\,exp\,Y = exp\,Z \qquad (1)$$

Levou stranu tohoto vztahu vyjádřím běžným způsobem a to jako součinu dvou rozvojů exponenciálních funkcí do určitého řádu. Pravou stranu vyjádřím právě pomocí svého balíku, který vyjadřuje formuli. Tu vyjádřím do stejného řádu jako levou stranu. Nechť vyjádřím formuli s přesností řádu M, pak se musí všechny členy od nultého až po M-tý řád shodovat na levé a pravé straně vztahu. Za X budu dosazovat $\in A$ a za Y budu

dosazovat ϵB, kde ϵ je konstanta, pomocí níž identifikuji jednotlivé řády přesnosti ve vztahu.

Pro tento test použiji krátkých funkcí vyjadřujících nekomutativní mocninu a exponenciální relaci, které jsem zavedl v kapitole 2.1.2. za definicí 2 a ještě zavedu další pravidla, pomocí nichž budu vztah upravovat a ověřovat jeho platnost.

2.3.1. Pravidla sloužící k úpravě výrazů

```
pSpojMocniny = {
    (X_ ** X_) → (X^2),
    (X_ ** X_^m_) → (X^(m+1)),
    (X_^m_ ** X_) → (X^(m+1)),
    (X_ ** X_ ** Y_) → (X^2 ** Y),
    (X_ ** Y_ ** Y_) → (X ** Y^2)
  };
```

Toto pravidlo zkracuje zápis nekomutativního násobení do přehlednější podoby mocnin. Tato spíše estetická funkce tohoto pravidla však není jedinou. Důležitější je, že se zápis zkrátí a výraz je pak připraven pro rychlejší manipulaci zaokrouhlovacího pravidla **pRound**, které je uvedeno dále. To totiž zanedbá členy obsahující mocniny ϵ s nadbytečně vysokým exponentem, tedy s exponentem vyšším než je požadovaný řád přesnosti při kontrole vztahu.

```
pSpojStejnaEpsUvnitrMocniny = {(A_)^m_ :> (Collect[A, ϵ])^m};
```

Pravidlo, které upraví mocněnce ve výrazu do přehlednější podoby tím, že k sobě "sesbírá" členy obsahující stejné mocniny ϵ. Toto pravidlo slouží podobně jako předchozí k úpravě výrazu před použitím pravidla **pRound**, které pak jednodušeji nalezne členy s příliš vysokým exponentem u ϵ.

```
pRound[M_] = {
    ϵ^m_/;m>M A_ → 0,
    ((A___ + ϵ^m_ C__ + B___)^n_) /; (m + n - 1 > M) → (A + B)^n
  };
```

Zaokrouhlovací pravidlo **pRound** zanedbá ve výrazu členy vyššího řádu, než je M. První část pravidla je velmi jednoduchá. Výraz se prohledá a naleznou se všechny členy obsahující ϵ, jehož exponent je větší než M a tyto členy se z výrazu odstraní. Druhá část pravidla zanedbá určité členy rovnou v mocnině a to ty, které by po roznásobení této mocniny zaručeně vytvořily výrazy řádu vyššího než M. Před tímto krokem je potřeba právě použít předchozích pravidel **pSpojMocniny** a **pSpojStejnaEpsUvnitrMocniny**, která výraz převedou do tvaru mocnin a zjednoduší následnou manipulaci s nimi. Zanedbání nadbytečných členů se dělá hlavně kvůli dalšímu průběhu výpočtu, které spočívá v roznásobování této mocniny. Pro vysoké řády přesnosti by po roznásobení

vzniklo mnoho sčítanců a další výpočet, zejména pro vyšší řády, by byl velmi pomalý. Proto je dobré se nadbytečných členů zbavit ještě před roznásobením.

```
pRoznasobMocniny =
  {(A_ ** B_ ** C___)^m_ → (A ** B ** C) ** (A ** B ** C)^(m-1),
   (X : (A | B) c_.)^m_ → (X c) ** (X c)^(m-1)};
```

Toto je první z roznásobovacích pravidel. Použiji ho hlavně proto, že spojování mocnin pomocí pravidla **pSpojMocniny** není jednoznačné. Abych tedy mohl upravit výrazy tvaru $(A ** B)^2 ** A - A ** (B ** A)^2$, musím oba sčítance tohoto výrazu nejprve roznásobit do stejného tvaru $A ** B ** A ** B ** A$.

```
pExpandCR = {
    (A_ + B_) ** (C_) → (A ** C + B ** C) ,
    (A_) ** (B_ + C_) → (A ** B + A ** C),
    (A_ + B_) ** (C_ + D_) → (A ** C + B ** C + A ** D + B ** D) ,
    (A_ + B_ + C___)^m_ → (A + B + C) ** (A + B + C)^(m-1)
  };
```

Pravidlo **pExpandCR** je druhé z roznásobovacích pravidel. Jedná se o nekomutativní roznásobování závorek.

```
pII = {II ** II^m_. → II, II^m_. ** X_ → X, X_ ** II^m_. → X,
   II^m_. X_ → II X};
```

Jednoduché pravidlo pro násobení jednotkovým prvkem.

```
KK = (_Integer | _Rational | ψ | ψ^q_Integer | t | t^q_Integer |
     s | s^q_Integer | φ1 | φ1^q_Integer | φ2 | φ2^q_Integer |
     _Real | ϵ | ϵ^q_Integer | α | α^q_Integer | γ | γ^q_Integer |
     β | β^q_Integer | ξ | ξ^q_Integer | ζ | ζ^q_Integer | η |
     η^q_Integer | -i | i | i / 2 | 1 / ω);

pVytkniKonstanty = {
    (-X_) ** Y → -X ** Y,
    ((m : KK) X_) ** (Y_) → (m) (X ** Y),
    (X_) ** ((m : KK) Y_) → (m) (X ** Y),
    ((n : KK) X_) ** ((m : KK) Y_) → (n m) (X ** Y)
  };
```

Do proměnné **KK** jsou uloženy všechny druhy konstant, které v mé práci v podobných výrazech vystupují. Pravidlo **pVytkniKonstanty** pak není nic jiného než vytýkaní konstant z nekomutativních součinů před tyto součiny. Je velmi důležité pro další práci s výrazy.

2.3.2. Vlastní testovací procedura

Ukáži postup testování funkce **BCHfle** z balíku na kontrole vztahu (1) do řádu přesnosti 2. Na tomto nízkém řádu bude výsledek formule ještě poměrný krátký a úpravy, které na něm budu provádět budou proto viditelnější a bude na nich jasně patrný význam předcházejících pravidel.

■ Ukázkový test pro formuli druhého řádu

Nejprve do proměnné **RAD** uložím požadovaný řád přesnosti.

```
RAD = 2;
```

■ Vyjádření levé strany vztahu (1)

```
LS = expRelace[ϵ A, RAD] ** expRelace[ϵ B, RAD]
```

$$\left(\mathrm{II} + \mathrm{A}\,\epsilon + \frac{1}{2}\,(\mathrm{A}\,\epsilon) ** (\mathrm{A}\,\epsilon)\right) ** \left(\mathrm{II} + \mathrm{B}\,\epsilon + \frac{1}{2}\,(\mathrm{B}\,\epsilon) ** (\mathrm{B}\,\epsilon)\right)$$

Tento základní tvar upravím vytknutím konstant.

```
% //. pVytkniKonstanty
```

$$\left(\mathrm{II} + \mathrm{A}\,\epsilon + \frac{1}{2}\,\epsilon^2\,\mathrm{A} ** \mathrm{A}\right) ** \left(\mathrm{II} + \mathrm{B}\,\epsilon + \frac{1}{2}\,\epsilon^2\,\mathrm{B} ** \mathrm{B}\right)$$

Poté roznásobím závorky.

```
% //. pExpandCR
```

$$\begin{aligned}&\mathrm{II} ** \mathrm{II} + \mathrm{II} ** (\mathrm{B}\,\epsilon) + \mathrm{II} ** \left(\frac{1}{2}\,\epsilon^2\,\mathrm{B} ** \mathrm{B}\right) + (\mathrm{A}\,\epsilon) ** \mathrm{II} + \\ &(\mathrm{A}\,\epsilon) ** (\mathrm{B}\,\epsilon) + (\mathrm{A}\,\epsilon) ** \left(\frac{1}{2}\,\epsilon^2\,\mathrm{B} ** \mathrm{B}\right) + \left(\frac{1}{2}\,\epsilon^2\,\mathrm{A} ** \mathrm{A}\right) ** \mathrm{II} + \\ &\left(\frac{1}{2}\,\epsilon^2\,\mathrm{A} ** \mathrm{A}\right) ** (\mathrm{B}\,\epsilon) + \left(\frac{1}{2}\,\epsilon^2\,\mathrm{A} ** \mathrm{A}\right) ** \left(\frac{1}{2}\,\epsilon^2\,\mathrm{B} ** \mathrm{B}\right)\end{aligned}$$

A použiji pravidlo o násobení jednotkovým prvkem. A znovu vytknu konstanty.

```
% //. pII //. pVytkniKonstanty
```

$$\begin{aligned}&\mathrm{II} + \mathrm{A}\,\epsilon + \mathrm{B}\,\epsilon + \frac{1}{2}\,\epsilon^2\,\mathrm{A} ** \mathrm{A} + \epsilon^2\,\mathrm{A} ** \mathrm{B} + \frac{1}{2}\,\epsilon^2\,\mathrm{B} ** \mathrm{B} + \\ &\frac{1}{2}\,\epsilon^3\,\mathrm{A} ** \mathrm{A} ** \mathrm{B} + \frac{1}{2}\,\epsilon^3\,\mathrm{A} ** \mathrm{B} ** \mathrm{B} + \frac{1}{4}\,\epsilon^4\,\mathrm{A} ** \mathrm{A} ** \mathrm{B} ** \mathrm{B}\end{aligned}$$

A zápis ještě zkrátím použitím pravidla **pSpojMocniny** a spojím členy obsahující stejné mocniny ϵ (tzn. členy stejných řádů).

```
Collect[% //. pSpojMocniny, ϵ]
```

$$II + (A + B)\,\epsilon + \epsilon^2\left(\frac{A^2}{2} + \frac{B^2}{2} + A ** B\right) + \epsilon^3\left(\frac{A ** B^2}{2} + \frac{A^2 ** B}{2}\right) + \frac{1}{4}\epsilon^4 A^2 ** B^2$$

A nakonec zanedbám členy, které obsahují ϵ v mocnině větší než je požadovaný řád.

```
Print["LS = ", LS = % /. pRound[RAD]];
```

$$LS = II + (A + B)\,\epsilon + \epsilon^2\left(\frac{A^2}{2} + \frac{B^2}{2} + A ** B\right)$$

■ Vyjádření pravé strany vztahu (1) pomocí formule

```
PS = expRelace[BCHfle[ϵ A, ϵ B, RAD], RAD]
```

$$II + A\epsilon + B\epsilon + \frac{1}{2}(A\epsilon) ** (B\epsilon) - \frac{1}{2}(B\epsilon) ** (A\epsilon) + \frac{1}{2}\left(A\epsilon + B\epsilon + \frac{1}{2}(A\epsilon) ** (B\epsilon) - \frac{1}{2}(B\epsilon) ** (A\epsilon)\right) ** \left(A\epsilon + B\epsilon + \frac{1}{2}(A\epsilon) ** (B\epsilon) - \frac{1}{2}(B\epsilon) ** (A\epsilon)\right)$$

Na tento výraz nejprve aplikuji pravidlo, které ho zkrátí do tvaru mocnin. Dále budu směřovat k tomu, abych mohl zanedbat některé členy ještě před roznásobením. Nyní se zaměřím na poslední nekomutativní součin. Vezmu-li poslední člen prvního součinitele $\frac{1}{2}(A\epsilon) ** (B\epsilon)$, pak je zřejmé že tento člen je již řádu ϵ^2 a tedy při roznásobení s kterýmkoliv členem druhého součinitele (ten obsahuje členy nejméně řádu ϵ^1) dostanu výraz nejméně řádu ϵ^3, který již překračuje stanovenou přesnost a tudíž by toto roznásobování bylo zbytečné.

```
% //. pSpojMocniny
```

$$II + A\epsilon + B\epsilon + \frac{1}{2}(A\epsilon) ** (B\epsilon) + \frac{1}{2}\left(A\epsilon + B\epsilon + \frac{1}{2}(A\epsilon) ** (B\epsilon) - \frac{1}{2}(B\epsilon) ** (A\epsilon)\right)^2 - \frac{1}{2}(B\epsilon) ** (A\epsilon)$$

Opět vytknu konstanty. Nejen z estetických důvodu, ale hlavně pro další manipulaci s pravidly. A pak znovu spojím mocniny. V tomto případě nízkého řádu nemá toto pravidlo uplatnění, ale při vyšších už ano.

```
% //. pVytkniKonstanty //. pSpojMocniny
```

$$II + A\epsilon + B\epsilon + \frac{1}{2}\epsilon^2 A ** B - \frac{1}{2}\epsilon^2 B ** A + \frac{1}{2}\left(A\epsilon + B\epsilon + \frac{1}{2}\epsilon^2 A ** B - \frac{1}{2}\epsilon^2 B ** A\right)^2$$

Nyní spojím mocniny ϵ stejných řádů uvnitř mocnin součtů. Tím připravím výraz do podoby, s kterou pracuje zaokrouhlovací pravidlo mnohem efektivněji.

```
% //. pSpojStejnaEpsUvnitrMocniny
```

$$II + A\epsilon + B\epsilon + \frac{1}{2}\epsilon^2 A ** B + \frac{1}{2}\left((A+B)\epsilon + \epsilon^2\left(\frac{A ** B}{2} - \frac{B ** A}{2}\right)\right)^2 - \frac{1}{2}\epsilon^2 B ** A$$

Teď můžu konečně zanedbat členy, které by po roznásobení utvořili výrazy s ϵ ve vyšší mocnině než je řád **RAD**.

```
% //. pRound[RAD]
```

$$II + A\epsilon + B\epsilon + \frac{1}{2}(A+B)^2\epsilon^2 + \frac{1}{2}\epsilon^2 A ** B - \frac{1}{2}\epsilon^2 B ** A$$

Je tedy vidět, že zmizel člen $\epsilon^2\left(\frac{A**B}{2} - \frac{B**A}{2}\right)$ v druhé mocnině součtu.

Nyní výraz roznásobím a vytknu konstanty. Zde se toto pravidlo opět neuplatní, protože roznásobovaná mocnina obsahovala jen ϵ řádu 2, ale kdyby obsahovala více druhů mocnin, pak by se pravidlo uplatnilo. A ještě zápis zkrátím spojením mocnin.

```
% //. pExpandCR //. pVytkniKonstanty //. pSpojMocniny
```

$$II + A\epsilon + B\epsilon + \frac{1}{2}\epsilon^2 A ** B - \frac{1}{2}\epsilon^2 B ** A + \frac{1}{2}\epsilon^2\left(A^2 + B^2 + A ** B + B ** A\right)$$

Následující úpravy nejsou pro tento řád nutná, ale pro vyšší řády ano, protože výraz v takových chvílích ještě obsahuje neroznásobené členy. Následující sekvencí pravidel se výraz upraví podobně jako v předchozím do podoby, v které pak zanedbám některé zbytečné členy. Pak vše roznásobím a znovu nadbytečné členy zanedbám.

```
Collect[%, ϵ]
```

$$II + (A+B)\epsilon + \epsilon^2\left(\frac{A ** B}{2} - \frac{B ** A}{2} + \frac{1}{2}\left(A^2 + B^2 + A ** B + B ** A\right)\right)$$

```
% //. pRound[RAD] //. pVytkniKonstanty //. pExpandCR //.
    pRound[RAD] //. pVytkniKonstanty //.
  pSpojStejnaEpsUvnitrMocniny //. pSpojMocniny //. pRound[RAD]
```

$$II + (A+B)\epsilon + \epsilon^2\left(\frac{A ** B}{2} - \frac{B ** A}{2} + \frac{1}{2}\left(A^2 + B^2 + A ** B + B ** A\right)\right)$$

Dále je třeba roznásobit členy v závorkách tím, co stojí před nimi. Tím dostanu situaci, kdy se některé členy sečtou dohromady, případně se úplně odečtou. Pozor! Pravidlo Expand by mohlo způsobit chybu, pokud bych jej použil na výraz obsahující mocninu. V těchto výrazech se vyskytují nekomutativní mocniny, zatímco pravidlo Expand by je roznásobilo komutativně $(A+B)^2 \neq A^2 + 2AB + B^2$.

```
Collect[% // Expand , ϵ]
```

$$II + (A + B)\,\epsilon + \epsilon^2\left(\frac{A^2}{2} + \frac{B^2}{2} + A ** B\right)$$

Nakonec ještě roznásobím zkrácené zápisy mocnin. Tato zkrácení jsou totiž nejednoznačná a tudíž se ve výrazu mohou vyskytovat například členy $(A ** B)^2 ** A - A ** (B ** A)^2$, které se automaticky neodečtou, i když by měly. Poté, co se tyto členy odečtou mohu zápis zbývajících členů opět zkrátit.

```
Print["PS = ", PS = % //. pRoznasobMocniny //. pSpojMocniny];
```

$$PS = II + (A + B)\,\epsilon + \epsilon^2\left(\frac{A^2}{2} + \frac{B^2}{2} + A ** B\right)$$

V tomto případě již ihned vidíme, že levá i pravá strana je po zanedbání nadbytečných členů stejná.

```
LS - PS
```

```
0
```

■ Test pro formuli obecného řádu

Postup je stejný jako v ukázce pro druhý řád. Proto zápis zkrátím. V rozumném čase se mi podařilo otestovat platnost formule do devátého řádu. Vyšší řády se již rozvinou do příliš dlouhých výrazů a úpravy s nimi trvají déle.

```
RAD = 6;
```

■ Vyjádření levé strany vztahu (1)

```
LS = expRelace[ϵ A, RAD] ** expRelace[ϵ B, RAD];

% //. pVytkniKonstanty //. pExpandCR //. pII //. pVytkniKonstanty //.
  pSpojMocniny;

Print["LS = ", LS = Collect[%, ϵ] /. pRound[RAD]]
```

$$LS = II + (A + B)\,\epsilon + \epsilon^2\left(\frac{A^2}{2} + \frac{B^2}{2} + A ** B\right) +$$
$$\epsilon^3\left(\frac{A^3}{6} + \frac{B^3}{6} + \frac{A ** B^2}{2} + \frac{A^2 ** B}{2}\right) +$$
$$\epsilon^4\left(\frac{A^4}{24} + \frac{B^4}{24} + \frac{A ** B^3}{6} + \frac{A^2 ** B^2}{4} + \frac{A^3 ** B}{6}\right) +$$
$$\epsilon^5\left(\frac{A^5}{120} + \frac{B^5}{120} + \frac{A ** B^4}{24} + \frac{A^2 ** B^3}{12} + \frac{A^3 ** B^2}{12} + \frac{A^4 ** B}{24}\right) +$$
$$\epsilon^6\left(\frac{A^6}{720} + \frac{B^6}{720} + \frac{A ** B^5}{120} + \frac{A^2 ** B^4}{48} + \frac{A^3 ** B^3}{36} + \frac{A^4 ** B^2}{48} + \frac{A^5 ** B}{120}\right)$$

■ Vyjádření pravé strany pomocí formule

```
PS = expRelace[BCHfle[ϵ A, ϵ B, RAD], RAD];

% //. pSpojMocniny //. pVytkniKonstanty //. pSpojMocniny //.
          pSpojStejnaEpsUvnitrMocniny //. pRound[RAD] //.
        pExpandCR //. pVytkniKonstanty //. pSpojMocniny //.
      pSpojStejnaEpsUvnitrMocniny //. pRound[RAD] //.
    pVytkniKonstanty //. pExpandCR;

% //. pRound[RAD] //. pVytkniKonstanty //.
      pSpojStejnaEpsUvnitrMocniny //. pSpojMocniny //.
    pRound[RAD] // Expand ;

Print["PS=",
  PS = Collect[%, ϵ] //. pRoznasobMocniny //. pSpojMocniny];
```

$$PS=II + (A+B)\,\epsilon + \epsilon^2\left(\frac{A^2}{2} + \frac{B^2}{2} + A ** B\right) +$$
$$\epsilon^3\left(\frac{A^3}{6} + \frac{B^3}{6} + \frac{A ** B^2}{2} + \frac{A^2 ** B}{2}\right) +$$
$$\epsilon^4\left(\frac{A^4}{24} + \frac{B^4}{24} + \frac{A ** B^3}{6} + \frac{A^2 ** B^2}{4} + \frac{A^3 ** B}{6}\right) +$$
$$\epsilon^5\left(\frac{A^5}{120} + \frac{B^5}{120} + \frac{A ** B^4}{24} + \frac{A^2 ** B^3}{12} + \frac{A^3 ** B^2}{12} + \frac{A^4 ** B}{24}\right) +$$
$$\epsilon^6\left(\frac{A^6}{720} + \frac{B^6}{720} + \frac{A ** B^5}{120} + \frac{A^2 ** B^4}{48} + \frac{A^3 ** B^3}{36} + \frac{A^4 ** B^2}{48} + \frac{A^5 ** B}{120}\right)$$

A nakonec obě strany srovnám a vidím, že formule je vyjádřena správně.

```
LS - PS
```

```
0
```

3. Aplikace formule pro úpravu propagátoru

3.1. Vybrané pojmy kvantové mechaniky

3.1.1. Teorie - propagátor harmonického oscilátoru

■ Časově nezávislý kvantový harmonický oscilátor

Kvantový harmonický oscilátor je plně popsán Hamiltonovým operátorem. Případ časově nezávislého harmonického oscilátoru odpovídá následujícímu Hamiltoniánu.

$$H_\omega(t) = -\frac{\hbar^2}{2m}\partial_x^2 + \frac{m\omega^2 x^2}{2}$$

Pro další práci položme bez újmy na obecnosti konstanty $m = 1,\ \hbar = 1$. Tedy

$$H_\omega(t) = -\frac{1}{2}\partial_x^2 + \frac{\omega^2 x^2}{2}$$

■ Časově závislý kvantový harmonický oscilátor

Dále se ovšem budu zabývat časově závislým harmonickým oscilátorem, v jehož Hamiltoniánu se objevuje ještě právě časově závislý člen.

$$H(t) = -\frac{1}{2}\partial_x^2 + \frac{\omega^2 x^2}{2} + f(t)\,x$$

Tento operátor si zadefinuji v systému *Mathematica* následovně:

```
Hω[ff_] := -1/2 ∂{x,2} ff + ω^2 x^2/2 ff;

H[ff_] := Hω[ff] + f[t] x ff;
```

■ Schrödingerova rovnice

Časový vývoj systému je v kvantové mechanice popsán Schrödingerovou rovnicí, která má následující tvar:

$$i\hbar\,\partial_t\,\Psi(t, x) = H(t)\,\Psi(t, x)$$

A opět zjednodušme položením $\hbar = 1$.

$$i\,\partial_t\,\Psi\,(t,\,x)\,=\,H\,(t)\,\Psi\,(t,\,x)$$

Vysvětlivky:

$\Psi(t,\,x)$ - časově závislá vlnová funkce popisující stav kvantového systému

$H\,(t)$ - Hamiltonův operátor zkoumaného systému

$\hbar$ - redukovaná Planckova konstanta (Diracova konstanta)

Pro případ časově závislého harmonického oscilátoru má Schrödingerova rovnice tvar:

$$i\,\partial_t\,\Psi\,(t,\,x)\,=\,-\,\frac{1}{2}\,\partial_x^2\,\Psi\,(t,\,x)\,+\,\frac{\omega^2\,x^2}{2}\,\Psi\,(t,\,x)\,+\,f\,(t)\,x\,\Psi\,(t,\,x)$$

```
SchroedingerEq[Ψ_, H_] := i ∂t Ψ - H[Ψ];
```

■ Propagátor

Hledáme řešení Schrödingerovy rovnice. Nechť je pro systém zadána počáteční podmínka $\phi\,(x)$. Pak časový vývoj systému z času s do času t určuje vlnová funkce $\Psi\,(t,\,s,\,x)$. Lze psát:

$$\Psi\,(t,\,s,\,x)\,=\,U\,(t,\,s)\,\phi\,(x),$$

kde $U\,(t,\,s)$ je unitární operátor, který se nazývá **propagátor**.

Je známo, jak vypadají propagátory pro kvantový harmonický oscilátor. V případě, že se jedná o časově nezávislý, má propagátor následující snadno odvoditelný tvar:

$$U_0\,(t,\,s)\,=\,e^{-i\,H_\omega\,(t-s)}$$

A pro vlnovou funkci v případě časově nezávislého harmonického oscilátoru platí:

$$\Psi_0\,(t,\,s,\,x)\,=\,U_0\,(t,\,s)\,\phi\,(x),$$

Pro časově závislý případ je situace složitější. Podle Ennse a Veseliće [1] má propagátor tvar:

$$U\,(t,\,s)\,=\,e^{-i\,\varphi_1(t,s)\,x}\,e^{\frac{\varphi_2(t,s)}{\omega}\,\partial_x}\,e^{-i\,H_\omega\,(t-s)\,-\,i\,\psi(t,s)},$$

kde $\varphi_1\,(t,\,s)\,=\,\int_s^t\,cos\,((t-u)\,\omega)\,f\,(u)\,du,$

$\varphi_2\,(t,\,s)\,=\,\int_s^t sin\,((t-u)\,\omega)\,f\,(u)\,du,$

$\psi\,(t,\,s)\,=\,\frac{1}{2}\,\int_s^t\,(\varphi_1\,(v,\,s)^2\,-\,\varphi_2\,(v,\,s)^2)\,dv.$

V následující kapitole 3.2. se budu zabývat právě tímto výrazem pro propagátor. Cílem bude přepsat tento výraz do tvaru s jediným exponentem, tedy do tvaru e^Z.

Zkombinováním předchozích vztahů lze psát:

$$\Psi(t, s, x) = e^{-i\varphi_1(t,s)\,x}\, e^{\frac{\varphi_2(t,s)}{\omega}\partial_x}\, e^{-i\psi(t,s)}\, \Psi_0(t, s, x).$$

V následujícím odstavci 3.1.2. budu oprávněnost tohoto tvaru vlnové funkce ověřovat s tím, že funkce $\Psi_0(t, s, x)$ splňuje Schrödingerovu rovnici pro časově nezávislý případ.

3.1.2. Splňuje řešení podle Ennse a Veseliće Schrödingerovu rovnici?

■ Příprava vztahů

Nejprve upravím tvar vlnové funkce, kterou poté dosadím do Schrödingerovy rovnice a ověřím její platnost.

K tomu budu potřebovat vědět, jaká je funkce operátorů typu $(e^{\xi\,\partial_x} f)(x)$ a $(e^{i\eta x} f)(x)$. Z formálního rozvoje těchto operátorů do řad lze odvodit, že $(e^{\xi\,\partial_x} f)(x)$ je takzvaný operátor posunutí a platí pro něj $(e^{\xi\,\partial_x} f)(x) = f(x+\xi)$ a $(e^{i\eta x} f)(x)$ je operátor násobení a platí $(e^{i\eta x} f)(x) = e^{i\eta x} f(x)$. Nyní tyto operátory zadefinuji do systému *Mathematica.*

```
ExpT[η_, f_] := e^(i η x) f;  (*operátor násobení*)
ExpM[ξ_, f_] := f /. {x → x + ξ}(*operátor posunutí*);
```

Ještě také zadefinuji funkce vyskytující se ve výrazu pro propagátor.

```
pIntFunkce = {
   φ1[t_, s_] :> Integrate[f[u] Cos[ω (t - u)], {u, s, t}],
   φ2[t_, s_] :> Integrate[f[u] Sin[ω (t - u)], {u, s, t}],
   ψ[t_, s_] :> 1/2 Integrate[(φ1[v, s]^2 - φ2[v, s]^2), {v, s, t}]};
```

A samotný ověřovaný tvar vlnové funkce.

```
Ψ[t_, s_, x_] :=
 e^(- i ψ[t, s]) ExpT[- φ1[t, s], ExpM[φ2[t, s]/ω, Ψ0[t, s, x]]]
```

Po aplikaci operátorů dostane vlnová funkce tvar:

```
Ψ[t, s, x]
```

$$e^{-i\psi[t,s]-i\,x\,\varphi_1[t,s]}\,\Psi_0\left[t, s, x + \frac{\varphi_2[t, s]}{\omega}\right]$$

■ Dosazení do Schrödingerovy rovnice

```
SchroedingerEq[Ψ[t, s, x] //. pIntFunkce, H];
```

Následně použiji předpoklad, že Ψ_0 splňuje Schrödingerovu rovnici v případě časově nezávislého harmonického oscilátoru. Tedy $\partial_t \Psi_0 = -i H_\omega \Psi_0$.

```
% //. {Ψ0^(1,0,0)[t, s, xx_] → -i Hω[Ψ0[t, s, x]] /. x → xx} //
 Simplify
```

$$\frac{1}{\omega}\left(e^{-\frac{1}{2}i\left(2x\int_s^t \text{Cos}[(t-u)\,\omega]\,f[u]\,du+\int_s^t\left(\left(\int_s^v \text{Cos}[(-u+v)\,\omega]\,f[u]\,du\right)^2-\left(\int_s^v f[u]\,\text{Sin}\right.\right.\right.\right.$$

$$\left(x\,\omega^2\left(\int_s^t f[u]\,\text{Sin}[(t-u)\,\omega]\,du\right)\right.$$

$$\Psi_0\left[t,\ s,\ x+\frac{\int_s^t f[u]\,\text{Sin}[(t-u)\,\omega]\,du}{\omega}\right]+$$

$$x\,\omega\left(\int_s^t -\omega\,f[u]\,\text{Sin}[(t-u)\,\omega]\,du\right)$$

$$\Psi_0\left[t,\ s,\ x+\frac{\int_s^t f[u]\,\text{Sin}[(t-u)\,\omega]\,du}{\omega}\right]-$$

$$i\left(\omega\int_s^t \text{Cos}[(t-u)\,\omega]\,f[u]\,du-\int_s^t \omega\,\text{Cos}[(t-u)\,\omega]\,f[u]\,du\right)$$

$$\left.\left.\left.\left.\Psi_0^{(0,0,1)}\left[t,\ s,\ x+\frac{\int_s^t f[u]\,\text{Sin}[(t-u)\,\omega]\,du}{\omega}\right]\right)\right)\right)$$

S úpravou integrálů je bohužel nutno systému *Mathematica* pomoci. Následuje tedy vytknutí parametru ω před integrály a z toho ihned vyplyne platnost ověřovaného řešení.

```
% /. {∫_s^t -ω f_ du :→ -ω ∫_s^t f du, ∫_s^t ω f_ du :→ ω ∫_s^t f du}
```

$$0$$

3.2. Úprava výrazu analogického k tvaru propagátoru

3.2.1. Používané pojmy a funkce

Nyní se vrátím k Lieově algebře $\mathcal{g}$ uvedené v kapitole 1.1.3. Budu vyšetřovat vztah, který v této kapitole postupně přiblížím. Oprávněnost postupu, který dále používám, zaručuje následující věta.

■ Věta (Poincarè-Birkhoff-Witt) ([4])

Buď g Lieova algebra. Univerzální obalující algebru označíme $\mathcal{U}(g)$. $\mathcal{U}(g)$ je nekonečnorozměrná asociativní algebra. Je-li $\{x_1, x_2, \ldots, x_N\}$ libovolná pevně zvolená báze v g, pak prvky $\left\{x_1^{n_1} x_2^{n_2} \ldots x_N^{n_N};\ n_1, n_2, \ldots, n_N \in \mathbb{Z}_+\right\}$ tvoří (algebraickou) bázi v $\mathcal{U}(g)$.

■ Převedení výrazu s klasickým komutátorem na jednoznačný tvar

Z předcházející věty vyplývá, že libovolný polynom v Lieově algebry lze přepsat do jednoznačného tvaru, polynomu v uspořádaných prvcích báze dané Lieovy algebry.

Ve zbývající části textu budu často využívat následující funkci, která mi libovolný polynom v prvcích báze Lieovy algebry převede právě do jednoznačného tvaru. Jedná se o to, že libovolný polynom převedu na polynom ve tvaru, ve kterém budou prvky báze v každém členu výsledného polynomu uspořádány podle předem určeného pořadí. Díky tomuto jednoznačnému tvaru je pak možné provádět zjednodušující úpravy takovýchto polynomů. Ty jsou například výsledkem Baker-Campbell-Hausdorffovy formule.

Mým dalším cílem tedy bylo naprogramovat takový algoritmus, který toto převádění polynomů provede.

■ Definice komutačních vztahů

Vytvořím pravidla, která nahradí výrazy zapsané ve špatném pořadí výrazy napsanými ve správném pořadí. Vyjdu z toho, že mám zadány Lieovy závorky ve tvaru lineární kombinace bazický prvků. Nechť mám dánu Lieovu závorku $[X, Y] = X ** Y - Y ** X$, pak určím, že z dvojice sčítanců, které jsou na pravé straně, je právě první zapsán v souladu s definovaným pořadím bazických prvků a druhý ne. Tento druhý člen tedy mohu vyjádřit pomocí prvního. Následující seznam pravidel bude právě klíčem, podle kterého se budou tyto druhé členy vyjadřovat pomocí prvních.

```
pRel = pKomRel //. {k[X_, Y_] → X ** Y - Y ** X} //.
  {Rule[A_ + B_, C_] → Rule[B, C - A]}
```

```
{-Q ** P → II - P ** Q, P ** II → II ** P, Q ** II → II ** Q,
 -U ** P → -P ** U, -V ** P → 2 Q - P ** V, -W ** P → P - P ** W,
 -U ** Q → -2 P - Q ** U, -V ** Q → -Q ** V, -W ** Q → -Q - Q ** W,
 -U ** II → -II ** U, -V ** II → -II ** V, -W ** II → -II ** W,
 -V ** U → -2 II + 4 W - U ** V, -W ** U → 2 U - U ** W,
 -W ** V → -2 V - V ** W, Q ** P → -II + P ** Q, -P ** II → -II ** P,
 -Q ** II → -II ** Q, U ** P → P ** U, V ** P → -2 Q + P ** V,
 W ** P → -P + P ** W, U ** Q → 2 P + Q ** U, V ** Q → Q ** V,
 W ** Q → Q + Q ** W, U ** II → II ** U, V ** II → II ** V, W ** II → II ** W,
 V ** U → 2 II - 4 W + U ** V, W ** U → -2 U + U ** W, W ** V → 2 V + V ** W}
```

■ Příklad použití pro algebru $\mathcal{g}$ vzhledem uspořádání {I,P,Q,U,V,W}

```
P ** Q /. pRel (*správné pořadí-výraz zůstane tokový, jaký byl*)

 P ** Q

Q ** P /. pRel(*špatné pořadí-výraz se převede*)

 -I I + P ** Q
```

Bohužel, takovéto jednoduché pravidlo v systému *Mathematica* nestačí pro převádění složitějších polynomů. Pro ty je nutné vytvořit program, který bude v daném polynomu postupovat po jednotlivých jeho členech a každý člen zpracuje po jednotlivých jeho činitelích.

■ Funkce pro převedení polynomů do jednoznačného tvaru

Pro takovéto operace jsem vytvořil následující tři funkce, z nichž každá pracuje s různě složitými výrazy.

První z těchto funkcí zpracovává nejsložitější výrazy a uvnitř volá další funkce pro úpravu dílčích částí výrazu. Tyto funkce volá, dokud se předchozí celkový výsledný tvar nerovná tomu aktuálnímu. To pak totiž znamená, že dále již žádné úpravy provádět nelze a tedy že už byl nalezen požadovaný jednoznačný tvar.

Parametry této funkce jsou: **vyraz**, což je libovolný polynom, který chceme upravit a to ve tvaru podobnému `a A ** B ** C + b A ** B ** C + ...`; a **poradnik**, což je seznam obsahující prvky báze, zapsané v pořadí, ve kterém chceme vyjádřit členy výsledku. Polynom se tedy rozdělí na jednotlivé sčítance a na ně se volá funkce **pPrehodClen**, jejíž výstupem je onen člen s provedeným jedním přehozením do správného pořadí.

```
pPrevedNaZakladniTvar[vyraz_, poradnik_List, pComRels_] :=
 Block[{ven, iPoc, preved, iPocClenu, out, prev},
  preved = vyraz;
  Label[Preved];
  out = 0;
  If[Head[preved] == Plus,
   {iPocClenu = Length[preved];
    (*pokud je to součet členů,
     tak se bude upravovat každý zvlášť*)
    For[iPoc = 1, iPoc ≤ iPocClenu,
     prev = preved[[iPoc]];
     ven = pPrehodClen[prev, poradnik, pComRels] /. pII // Expand;
     iPoc++; out = out + ven
    ];
   }, , out = pPrehodClen[preved, poradnik, pComRels] /. pII //
     Expand; ven = out;];
  (*kontrola, zda se nám již výstup neopakuje,
   pokud ne tak se výraz pokusíme ještě upravit*)
  If[out == preved, , preved = out; Goto[Preved];, preved = out;
   Goto[Preved];];
  out
 ];
```

Funkce **pPrehodClen** má také dva vstupní parametry. Prvním je opět výraz, který chceme upravit. Tentokrát by jím měl být pouze jednoduchý součin prvků báze a druhým parametrem je opět seznam definující pořadí. Tato funkce je rekurzivní. Analyzuje výraz (který si nyní představuji zapsaný ve tvaru stromu) postupně seshora, dokud nenarazí na podstrom, jehož hlavní operací je nekomutativním násobením. A na tento výraz pak aplikuje funkci **pPrehod**, která provede jedno prohození prvků.

Pokud by byl na vstupu funkce například výraz $(1/2)\ A ** B$, tak by výpočet probíhal následovně. Nejprve by funkce zjistila, že výraz je tvaru násobení jednou polovinou, konstantou. Tudíž si program zapamatuje, že při zpětném vyhodnocování bude násobit 1/2 a dále zpracuje zbytek výrazu, tedy $A ** B$. U $A ** B$ je již hlavní operací nekomutativní násobení, a tak je tento výraz poslán funkci pPrehod.

Omezením této funkce je, aby výraz, který má jako hlavní operaci nekomutativní násobení, obsahoval již jen součinitele, které jsou v seznamu **poradnik**, jinak funkce selže. To znamená, že si funkce neporadí s výrazem tvaru $A ** (B + C)$, ale poradí si s každým ze sčítanců součtu $A ** B + A ** C$ zvlášť. Operace nekomutativního násobení tedy musí být poslední operací, která se ve výrazu vyskytuje.

```
pPrehodClen[vyraz_, poradnik_List, pComRels_] :=
 Block[{ven},
  If[Head[vyraz] == NonCommutativeMultiply,
   {ven = pPrehod[vyraz, poradnik, pComRels];}, {},
   {
    If[Head[vyraz] == Symbol, ven = vyraz; Goto[Konec];];
    If[Head[vyraz] == Rational, ven = vyraz; Goto[Konec];];
    If[Head[vyraz] == Integer, ven = vyraz; Goto[Konec];];
    If[Head[vyraz] == Real, ven = vyraz; Goto[Konec];];
    If[Head[vyraz] == Complex, ven = vyraz; Goto[Konec];];
    ven = Head[vyraz][pPrehodClen[First[vyraz], poradnik,
       pComRels], pPrehodClen[Take[vyraz, -(Length[vyraz] - 1)],
       poradnik, pComRels]];
   }];
  Label[Konec];
  ven
 ]
```

Poslední z trojice je nejzákladnější funkce pro převod. Parametry funkce **pPrehod** jsou obdobné jako u předcházejících funkcí. Výraz musí ovšem obsahovat jediný druh operace a to nekomutativní násobení. Mezi činiteli nesmí být jiné prvky než ty, které jsou uvedeny v seznamu **poradnik**.

```
pPrehod[vyraz_, poradnik_List, pComRels_] :=
 Block[{iDelkaListu, Vystup = II, iPoc, Prohozeno = 0, cleny},
  {cleny = vyraz //. NonCommutativeMultiply → List;
   iDelkaListu = Length[cleny],
   For[iPoc = 1, iPoc < iDelkaListu,
    {
     If[Position[poradnik, cleny[[iPoc]]][[1]][[1]] >
      Position[poradnik, cleny[[iPoc + 1]]][[1]][[1]],
      Vystup = Vystup **
       ((cleny[[iPoc]] ** cleny[[iPoc + 1]]) /. pComRels);
      iPoc = iPoc + 2; Goto[ZaFor];,
      Vystup = Vystup ** cleny[[iPoc]]]}
     iPoc++];
   Label[ZaFor];
   (*Prohodily se některé členy. Zbytek tam pouze dopíšeme.*)
   Vystup =
    Vystup ** ((Take[cleny, -(iDelkaListu - iPoc + 1)]) /.
       {List -> NonCommutativeMultiply});
   Vystup = Vystup //. pExpandCR /. pII //. pVytkniKonstanty /. pII;
  };
  iDelkaListu;
  Label[KonecPrehod];
  Vystup
 ]
```

■ Malá ukázka použití funkce pPrevedNaZakladniTvar opět na algebře g

```
pPrevedNaZakladniTvar[(1 / 2) (P ** Q - Q ** P), PrvkyBaze,
 pRel]
```

$$\frac{II}{2}$$

3.2.2. Vlastní úprava výrazu

Nyní budu upravovat následující výraz do tvaru jediného exponenciálního zobrazení. Tento výraz je analogický s výrazem pro propagátor Ennse a Veseliće.

$$e^{\alpha P}\, e^{\beta Q}\, e^{\gamma (V-U)} \qquad (2)$$

Pro dostatečně malé reálné koeficienty α, β a γ jsou násobky $\alpha P, \beta Q$ a $\gamma (V - U)$ z okolí počátku a proto lze pro úpravu tohoto vtahu použít Baker-Campbell-Hausdorffovu formuli zavedenou ve druhé kapitole.

V první fázi úpravy použiji Baker-Campbell-Hausdorffovu formuli na první dva součinitele, konkrétně na součin $e^{\alpha P}\, e^{\beta Q}$.

■ Hledání Z_1 z rovnice $e^{\alpha P}\, e^{\beta Q} = e^{Z_1}$

Pomocí svého package vyjádřím Z_1 do sedmého řádu přesnosti.

```
<< Propagator`BCHfle`
```

```
Z1 = BCHfle[α P, β Q, 7] //. pVytkniKonstanty;
```

Nyní použiji funkce, které jsem si připravil pro převod polynomů do jednoznačného tvaru. Tímto převedením se zároveň aplikují zjednodušující identity a výsledek nám odkrývá, že řády vyššího stupně než dva se výrazu nevyskytují.

```
Z1 = pPrevedNaZakladniTvar[Z1, PrvkyBaze, pRel]
```

$$P\,\alpha + Q\,\beta + \frac{II\,\alpha\,\beta}{2}$$

Tedy s přesností do sedmého řádu jsem nalezl $Z_1 = \alpha P + \beta Q + \frac{\alpha\beta}{2} I$. Z_1 by takto vypadalo i kdybych počítal s nekonečnou přesností, protože:

```
Print["[Q,P] = " k[Q, P] //. pKomRel, "     a    ", "[P,Q] = ",
 k[P, Q] //. pKomRel]
```

```
-[Q,P] = II      a    [P,Q] = II
```

A všechny členů vyšších řádů než dva důsledkem toho obsahují výrazy tvaru $[Q, I]$ či $[P, I]$ a ty jsou v případě naší Lieovy algebry g rovny nule.

Například:

```
Print["[P,[Q,P]] = ", k[P, k[Q, P]] //. pKomRel //. pLieAlg,
 " = ", k[P, k[Q, P]] //. pKomRel //. pLieAlg //. pKomRel]
```

```
[P,[Q,P]] = -k[P, II] = 0
```

Důsledkem toho lze vztah (2) přepsat na následující vztah.

$$e^{\alpha P}\, e^{\beta Q}\, e^{\gamma (V-U)} = e^{\alpha P + \beta Q + \frac{\alpha\beta}{2} I}\, e^{\gamma (V-U)} \tag{3}$$

■ Hledání *Z* z rovnice $e^{\alpha P + \beta Q + \frac{\alpha\beta}{2} I}\, e^{\gamma (V-U)} = e^{Z}$

Nejprve vyjádřím Z do sedmého řádu pomocí funkce z package.

```
Z = BCHfle[A, B, 7] /. {A → Z1, B → γ (V - U)};
```

```
Z =
  Z //. pVytkniKonstanty //. pExpandCR //. pVytkniKonstanty //.
    pII //. pVytkniKonstanty // Expand;
```

Následuje převedení do jednoznačného tvaru, při kterém se výraz opět velmi zjednoduší a zpřehlední. Výpočet při tomto řádu přesnosti je však časově náročný.

```
Z = pPrevedNaZakladniTvar[Z, PrvkyBaze]
```

$$\begin{aligned} &P\alpha + Q\beta + \frac{II\,\alpha\beta}{2} - U\gamma + V\gamma + Q\alpha\gamma + \frac{1}{6} II\,\alpha^2\gamma + P\beta\gamma - \frac{1}{6} II\,\beta^2\gamma + \\ &\frac{1}{3} P\alpha\gamma^2 + \frac{1}{3} Q\beta\gamma^2 - \frac{1}{45} II\,\alpha^2\gamma^3 + \frac{1}{45} II\,\beta^2\gamma^3 - \frac{1}{45} P\alpha\gamma^4 - \\ &\frac{1}{45} Q\beta\gamma^4 + \frac{1}{315} II\,\alpha^2\gamma^5 - \frac{1}{315} II\,\beta^2\gamma^5 + \frac{2}{945} P\alpha\gamma^6 + \frac{2}{945} Q\beta\gamma^6 \end{aligned}$$

Vytknu prvky báze pro přehlednější identifikaci koeficientů.

```
Z = Collect[Z, {P, Q, U, V, W, II}]; Print["Z = ", Z]
```

$$\begin{aligned} Z = \; &-U\gamma + V\gamma + II\left(\frac{\alpha\beta}{2} + \frac{\alpha^2\gamma}{6} - \frac{\beta^2\gamma}{6} - \frac{\alpha^2\gamma^3}{45} + \frac{\beta^2\gamma^3}{45} + \frac{\alpha^2\gamma^5}{315} - \frac{\beta^2\gamma^5}{315}\right) + \\ &P\left(\alpha + \beta\gamma + \frac{\alpha\gamma^2}{3} - \frac{\alpha\gamma^4}{45} + \frac{2\alpha\gamma^6}{945}\right) + Q\left(\beta + \alpha\gamma + \frac{\beta\gamma^2}{3} - \frac{\beta\gamma^4}{45} + \frac{2\beta\gamma^6}{945}\right) \end{aligned}$$

Při použití celého nekonečného rozvoje formule by měl výraz následující tvar.

$$\begin{aligned} Z = &-U\gamma + V\gamma + \\ &I\left(\frac{\alpha\beta}{2} + \frac{\alpha^2\gamma}{6} - \frac{\beta^2\gamma}{6} - \frac{\alpha^2\gamma^3}{45} + \frac{\beta^2\gamma^3}{45} + \frac{\alpha^2\gamma^5}{315} - \frac{\beta^2\gamma^5}{315} + \ldots\right) + \\ &P\left(\alpha + \beta\gamma + \frac{\alpha\gamma^2}{3} - \frac{\alpha\gamma^4}{45} + \frac{2\alpha\gamma^6}{945} + \ldots\right) + \\ &Q\left(\beta + \alpha\gamma + \frac{\beta\gamma^2}{3} - \frac{\beta\gamma^4}{45} + \frac{2\beta\gamma^6}{945} + \ldots\right) \end{aligned} \tag{4}$$

Lze dokonce ihned říci, že koeficienty u prvků U a V jsou $-\gamma$ a γ. Protože z předpisu formule je zřejmé, že ve výsledku se vyskytují pouze komutátory prvků P, Q či

I s prvky U či V a ty lze podle definice algebry g vyjádřit jako lineární kombinaci pouze z prvků P, Q či I. Tedy nikde ve vyšších řádech se už nemohou vyskytovat prvky U, V či W. Z toho lze také usoudit, že koeficient u W je roven nule. Korektněji tyto koeficienty a i ty u zbývajících prvků P, Q a I získám řešením rovnic v kapitole 3.3.

Jednotkový prvek I komutuje se všemi prvky báze. To znamená, že platí $e^{\alpha I} e^{\beta X} = e^{\alpha I + \beta X}$; $\forall X \in \{P, Q, U, V, W\}$, neboť členy formule vyšších řádů než jedna obsahují vždy komutátor $[I, X]$ nebo $[X, I]$, který je roven nule.

Celkově lze tedy psát finální vztah, který jsem hledal:

$$e^{\alpha P} e^{\beta Q} e^{\gamma (V-U)} = e^{f I} e^{a U + b V + c W + d P + e Q} \qquad (5)$$

Předpokládám, že koeficienty α, β, γ jsou dány. Budu tedy hledat koeficienty a, b, c, d, e, f jako funkce proměnných α, β, γ. Z teorie víme, že tyto koeficienty jednoznačně existují a že to jsou analytické funkce proměnných α, β, γ.

Jak tedy tyto koeficienty vypadají?

3.3. Hledání koeficientů

V této kapitole naleznu všechny neznámé koeficienty vystupující ve vztahu (5).

3.3.1. Používané pojmy a identity

■ Definice 3:

Nechť g a h jsou libovolné Lieovy algebry. Zobrazení $\rho : g \to h$ nazveme **homomorfismem Lieových algeber** právě tehdy, když platí:

1) ρ je lineární

2) $(\forall X, Y \in g)\ (\rho([X, Y]) = [\rho(X), \rho(Y)])$

■ Definice 4:

Homomorfismus Lieových algeber, který je zobrazením $\rho : g \to gl(n, \mathbb{C})$ (kde $gl(n, \mathbb{C})$ se jako množina shoduje s $\mathbb{C}^{n,n}$ a Lieova závorka je dána vztahem $[A, B] = AB - BA$) nazveme **reprezentací** Lieovy algebry g ve vektorovém prostoru $\mathbb{C}^n$.

Poznámka: To znamená, že se každému prvku $X \in g$ reprezentací přiřadí matice $\rho(X) \in \mathbb{C}^{n,n}$ (matice typu $n \times n$, jejíž prvky jsou komplexní čísla).

Zápis vztahu (5) pro matice do systému *Mathematica*.

```
MatrixEq[II_, PP_, QQ_, UU_, VV_, WW_, Nula_] :=
  MatrixExp[α PP].MatrixExp[β QQ].MatrixExp[γ (VV - UU)] -
    MatrixExp[f II].MatrixExp[a UU + b VV + c WW + d PP + e QQ] ==
   Nula;
```

3.3.2. Nalezení koeficientů *a*, *b* a *c* pomocí 2-rozměrné reprezentace g

```
I2 = {{0, 0}, {0, 0}}; P2 = {{0, 0}, {0, 0}}; Q2 = {{0, 0}, {0, 0}};
U2 = {{0, 0}, {2, 0}}; V2 = {{0, -2}, {0, 0}};
W2 = {{1, 0}, {0, -1}};
```

```
Print["I = ", I2 // MatrixForm, "   P = ", P2 // MatrixForm,
 "   Q = ", Q2 // MatrixForm, "   U = ", U2 // MatrixForm,
 "   V = ", V2 // MatrixForm, "   W = ", W2 // MatrixForm]
```

$$I = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad P = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad Q = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$U = \begin{pmatrix} 0 & 0 \\ 2 & 0 \end{pmatrix} \quad V = \begin{pmatrix} 0 & -2 \\ 0 & 0 \end{pmatrix} \quad W = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Ověřím, zda pro tuto reprezentaci platí definované Lieovy závorky pro Lieovu algebru g. Nejprve převedu Lieovy závorky na klasický komutátor s maticovým násobením.

```
MatrixDefRel = DefRel //. k[A_, B_] → A.B - B.A //.
  {Equal[A_ + B_, 0] → Equal[A, -B]}
```

```
{P.Q - Q.P == II, -II.P == -P.II, -II.Q == -Q.II, P.U == U.P,
 P.V - V.P == 2 Q, P.W - W.P == P, Q.U - U.Q == -2 P, Q.V == V.Q,
 Q.W - W.Q == -Q, II.U == U.II, II.V == V.II, II.W == W.II,
 U.V - V.U == -2 II + 4 W, U.W - W.U == 2 U, V.W - W.V == -2 V}
```

A dosadím zavedené matice.

```
% //. {II → I2, P → P2, Q → Q2, U → U2, V → V2, W → W2}
```

```
{True, True, True, True, True, True, True,
 True, True, True, True, True, True, True, True}
```

Tedy definice komutačních relací pro tuto reprezentaci platí. A jak jsem již ověřil v kapitole 1, pro algebru definovanou těmito komutačními relacemi platí Jacobiho identita, tedy jedná se o reprezentaci Lieovy algebry.

Aplikace této reprezentace na výchozí vztah (5)

```
NulovyPrvek = {{0, 0}, {0, 0}};
```

```
MatrixEq[I2, P2, Q2, U2, V2, W2, NulovyPrvek] // FullSimplify
```

$$\left\{\left\{-\text{Cosh}\left[\sqrt{-4\,a\,b+c^2}\right]+\text{Cosh}[2\,\gamma]-\frac{c\,\text{Sinh}\left[\sqrt{-4\,a\,b+c^2}\right]}{\sqrt{-4\,a\,b+c^2}},\right.\right.$$
$$\left.\frac{2\,b\,\text{Sinh}\left[\sqrt{-4\,a\,b+c^2}\right]}{\sqrt{-4\,a\,b+c^2}}-2\,\text{Cosh}[\gamma]\,\text{Sinh}[\gamma]\right\},$$
$$\left\{-\frac{2\,a\,\text{Sinh}\left[\sqrt{-4\,a\,b+c^2}\right]}{\sqrt{-4\,a\,b+c^2}}-2\,\text{Cosh}[\gamma]\,\text{Sinh}[\gamma],\ -\text{Cosh}\left[\sqrt{-4\,a\,b+c^2}\right]+\right.$$
$$\left.\left.\text{Cosh}[2\,\gamma]+\frac{c\,\text{Sinh}\left[\sqrt{-4\,a\,b+c^2}\right]}{\sqrt{-4\,a\,b+c^2}}\right\}\right\}==\{\{0,\ 0\},\ \{0,\ 0\}\}$$

```
sols1 = Solve[%, {a, b, c}]
```

```
Solve::incnst : Inconsistent or redundant transcendental equation.
  After reduction, the bad equation is ArcCosh[Cosh[√<<1>>]]^2
   (-2 + 2 <<1>>^4 - 4 <<1>>^2 <<1>>^2 + 2 Sinh[γ]^4) == 0. More…

Solve::incnst :
 Inconsistent or redundant transcendental equation. After reduction,
  the bad equation is ArcCosh[Cosh[√(-4 a b + c^2)]] <<2>>
   (-2 + 2 <<1>>^4 - 4 <<1>> <<1>>^2 + 2 Sinh[γ]^4) == 0. More…

Solve::ifun :
 Inverse functions are being used by Solve, so some solutions may not
  be found; use Reduce for complete solution information. More…

{{a → -γ, b → γ, c → 0}, {a → γ, b → -γ, c → 0}}
```

Funkce **Solve** nalezla dvě řešení. Jedno ale musí být špatné. Které je to správné zjistím dosazením do původního vztahu.

```
Print[sols1[[1]], ":",
  (MatrixEq[I2, P2, Q2, U2, V2, W2, NulovyPrvek] //. sols1[[1]] //
    FullSimplify), "\n", sols1[[2]], ":",
  (MatrixEq[I2, P2, Q2, U2, V2, W2, NulovyPrvek] //. sols1[[2]] //
    FullSimplify)];
```

```
{a → -γ, b → γ, c → 0}:True
{a → γ, b → -γ, c → 0}:
 {{0, -4 Cosh[γ] Sinh[γ]}, {-4 Cosh[γ] Sinh[γ], 0}} ==
  {{0, 0}, {0, 0}}
```

Správné řešení je tedy $a = -\gamma,\ \ b = \gamma,\ c = 0$.

3.3.3. Nalezení koeficientů *d* a *e* pomocí 3-rozměrné reprezentace g

```
I3 = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
P3 = {{0, 0, 0}, {0, 0, 1}, {0, 0, 0}};
Q3 = {{0, 0, 1}, {0, 0, 0}, {0, 0, 0}};
U3 = {{0, 0, 0}, {2, 0, 0}, {0, 0, 0}};
V3 = {{0, -2, 0}, {0, 0, 0}, {0, 0, 0}};
W3 = {{1, 0, 0}, {0, -1, 0}, {0, 0, 0}};

Print["I = ", I3 // MatrixForm, "   P = ", P3 // MatrixForm,
 "   Q = ", Q3 // MatrixForm, "   U = ", U3 // MatrixForm,
 "   V = ", V3 // MatrixForm, "   W = ", W3 // MatrixForm]
```

$$I = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad P = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad Q = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$U = \begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad V = \begin{pmatrix} 0 & -2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad W = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Opět ověřím, zda platí definované Lieovy závorky i pro tuto reprezentaci.

```
MatrixDefRel //. {II → I3, P → P3, Q → Q3, U → U3, V → V3, W → W3}
```

```
{True, True, True, True, True, True, True,
 True, True, True, True, True, True, True, True}
```

Je to tedy reprezentace Lieovy algebry g. Dosazením do vztahu (5) získám další rovnice pro určení zbývajících koeficientů d a e.

```
NulovyPrvek = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};

MatrixEq[I3, P3, Q3, U3, V3, W3, NulovyPrvek] //. sols1[[1]] //
 FullSimplify
```

$$\left\{\left\{0,\ 0,\ \frac{1}{2\,\gamma^2}\left(e^{-2\sqrt{\gamma^2}}\ (-d + 2\,\beta\,\gamma + d\,\text{Cosh}[2\,\gamma] - e\,\text{Sinh}[2\,\gamma])\right.\right.\right.$$
$$\left.\left.\left.\left(\gamma\,\text{Cosh}[2\,\gamma] + \sqrt{\gamma^2}\,\text{Sinh}[2\,\gamma]\right)\right)\right\},$$
$$\left\{0,\ 0,\ \frac{1}{2\,\gamma^2}\left(e^{-2\sqrt{\gamma^2}}\ (-e + 2\,\alpha\,\gamma + e\,\text{Cosh}[2\,\gamma] - d\,\text{Sinh}[2\,\gamma])\right.\right.$$
$$\left.\left.\left(\gamma\,\text{Cosh}[2\,\gamma] + \sqrt{\gamma^2}\,\text{Sinh}[2\,\gamma]\right)\right)\right\},$$
$$\{0,\ 0,\ 0\}\Big\} == \{\{0,\ 0,\ 0\},\ \{0,\ 0,\ 0\},\ \{0,\ 0,\ 0\}\}$$

```
sols2 = Solve[%, {d, e}] // FullSimplify // Flatten
```

$$\{d \to \gamma\,(\beta + \alpha\,\text{Coth}[\gamma]),\ e \to \gamma\,(\alpha + \beta\,\text{Coth}[\gamma])\}$$

Celkové řešení až na koeficient f je tedy:

```
sols = {sols1[[1]], sols2} // Flatten
```

```
{a → -γ, b → γ, c → 0, d → γ (β + α Coth[γ]), e → γ (α + β Coth[γ])}
```

Vztah (5) lze tedy zapsat ve tvaru:

$$e^{\alpha P}\, e^{\beta Q}\, e^{\gamma (V-U)} = e^{f I}\, e^{a U + b V + d P + e Q}, \qquad (6)$$

$$a = -\gamma,\ b = \gamma,\ d = \gamma\,(\beta + \alpha\, Coth[\gamma]),\ e = \gamma\,(\alpha + \beta\, Coth[\gamma])$$

3.3.4. Nalezení koeficientů *f* pomocí nekonečnorozměrné reprezentace g

■ Zavedení reprezentace

V této reprezentaci jsou jednotlivé prvky báze reprezentovány následujícími operátory: $I = 1$, $P = \partial_x$, $Q = x$, $U = P^2$, $V = Q^2$, $W = P\,Q$.

```
Iinf = #1 &;  Pinf = ∂x #1 &;  Qinf = x #1 &;  Uinf = ∂{x,2} #1 &;
Vinf = x^2 #1 &;  Winf = Composition[Pinf, Qinf];
```

Pravidlo, které aplikuje uvedenou reprezentaci:

```
pUseInfRepre = {II → Iinf, P → Pinf, Q → Qinf, U → Uinf, V → Vinf,
    W → Winf};
```

Opět ověřím, že pro tuto reprezentaci platí dané Lieovy závorky algebry g. Nejprve si upravím definice komutačních vztahů. V této reprezentaci je Lieova závorka komutátorem, v němž operace AB je skládání zobrazení. Definice upravím tak, že všechny členy z rovnosti v proměnné **DefRel** zapíši na levou stranu a pravá strana tak bude nulová. V další práci tedy tuto nulovou pravou stranu ze zápisu vynechám. Na upravené komutační relace se budu dívat jako na operátory tvořené součtem operátorů definovaných výše.

```
DefRelToOperators =
  DefRel //. k[A_, B_] → Composition[A, B] - Composition[B, A] //.
   {Equal[A_, C_] → A - C};
```

Dosadím konkrétní operátory této reprezentace.

```
MyOperators = DefRelToOperators //. pUseInfRepre;
```

V proměnné **MyOperators** se tedy skrývá seznam operátorů a nyní každý jeden operátor z tohoto seznamu aplikuji na libovolnou funkci $fce(x)$. Budu tedy ověřovat, zda aplikací těchto operátorů na libovolnou funkci $fce(x)$, dostanu ve výsledku nulu. Pomocí vestavěné funkce systému *Mathematica* **Through** zajistím aplikaci každého z členů operátoru na tuto funkci. Aplikaci jednoho operátoru ukáži na následujícím příkladu.

```
MyOperators[[1]][fce[x]]
```

```
(Composition[∂x #1 &, x #1 &] - Composition[x #1 &, ∂x #1 &] - (#1 &))[
 fce[x]]
```

```
Through[%, Plus]
```

```
fce[x] + (-Composition[x #1 &, ∂x #1 &])[fce[x]] +
 (-(#1 &))[fce[x]] + x fce'[x]
```

Jak je patrné, *Mathematica* si bohužel neporadila s operátorem (**-Composition**), stejně tak si neporadí pokud je operátor tvaru (cOperátor, kde c je konstanta). Musím tedy ještě vytknout konstanty před operátor.

```
% //. {(a_Integer A_)[x_] → a A[x]}
```

```
0
```

Výsledkem aplikace operátoru je tedy skutečně nula. Nyní předchozí postup aplikuji na všechny prvky ze seznamu operátorů. Ještě upravím aplikaci nulového operátoru a nakonec roznásobením dostanu očekávané výsledky.

```
Table[Through[MyOperators[[i]][fce[x]]],
    {i, 1, Length[MyOperators]}] //.
   {(a_Integer A_)[x_] → a A[x]} //. {Through[0[fce[x]]] → 0} //
 Expand
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

Pro další postup (odvození tvaru propagátoru) ještě reprezentaci přeškáluji parametrem ω následovně:

$$I = 1, \quad P = \frac{1}{\sqrt{\omega}}\,\partial_x, \quad Q = \sqrt{\omega}\,x, \quad U = P^2, \quad V = Q^2, \quad W = P\,Q$$

```
IinfW = Iinf; PinfW = 1/Sqrt[ω] ∂x #1 &; QinfW = Sqrt[ω] x #1 &;
UinfW = Composition[PinfW, PinfW];
VinfW = Composition[QinfW, QinfW];
WinfW = Composition[PinfW, QinfW];

pUseInfWRepre = {II → IinfW, P → PinfW, Q → QinfW, U → UinfW,
   V → VinfW, W → WinfW};
```

Následuje ověření:

```
MyOperatorsW = DefRelToOperators //. pUseInfWRepre;
```

```
Table[Through[MyOperatorsW[[i]][fce[x]]],
    {i, 1, Length[MyOperators]}] //.
   {(a_Integer A_)[x_] → a A[x]} //. {Through[0[fce[x]]] → 0} //
 Expand
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

Tato reprezentace tedy také splňuje definované komutační relace.

■ Hledání koeficientu f

■ Přepsání vztahu (6) pomocí substituce a použití nekonečnorozměrné reprezentace

Vyjdu ze vztahu (6), ve kterém provedu následující substituce:

$$\alpha = \xi,\ \beta = i\,\eta,\ \gamma = i\,\zeta,\ f = i\,\phi,\ d = \mu,\ \nu = -i\,e$$

Následující příkazy provedou substituci v koeficientech d a e vztahu (6).

```
pSubstFirstCoeffs := {α → ξ, β → i η, γ → i ζ, f → i ϕ};

d = γ (β + α Coth[γ]); e = γ (α + β Coth[γ]);

Print["μ = ", μ = Collect[d //. pSubstFirstCoeffs // Simplify, ξ],
 ";ν = ", ν = Collect[-i e //. pSubstFirstCoeffs // Simplify, ξ]]
```

```
μ = ζ (-η + ξ Cot[ζ]);ν = ζ (ξ + η Cot[ζ])
```

Nyní vytvořím kompletní substituční pravidla.

```
Clear[d, e, μ, ν];
pSubstCoeffs =
 List[pSubstFirstCoeffs,
   {d -> μ, e → i ν, μ → ζ (-η + ξ Cot[ζ]), ν → ζ (ξ + η Cot[ζ])}] //
  Flatten; Clear[d, e, f, γ, α, η, ν, μ]
```

Zapíši vztah (6) a provedu v něm kompletní substituci.

```
Eq = e^(α P) ** e^(β Q) ** e^(γ (V-U)) == e^(f II) ** e^(d P+e Q+γ[V-U]) //. pSubstCoeffs
```

```
e^(P ξ) ** e^(i Q η) ** e^(i (-U+V) ζ) ==
 e^(i II ϕ) ** e^(i Q ζ (ξ+η Cot[ζ])+P ζ (-η+ξ Cot[ζ])+(i ζ)[-U+V])
```

Po substituci tedy přejde vztah (6) na následující vztah.

$$e^{\xi P}\ e^{i\eta Q}\ e^{i\zeta(V-U)} = e^{i\phi}\ e^{i\zeta(V-U)\ +\ \mu P\ +\ i\nu Q}, \qquad (7)$$
$$\mu = \zeta\ (\xi\ Cot[\zeta]\ -\eta)\ ,\ \ \nu = \zeta\ (\xi\ +\eta\ Cot[\zeta])$$

Hledání koeficientu f se převedlo na hledání koeficientu ϕ. A k jeho nalezení použiji nekonečnorozměrnou reprezentaci s parametrem ω.

A po použití nekonečnorozměrné reprezentace s parametrem ω dostanu:

```
StylePrint[Eq //. pUseInfWRepre //. {Composition → C},
 FontSize → 7]
```

$$e^{\xi\left(\frac{\partial_x \#1}{\sqrt{\omega}}\&\right)} \star\star e^{i\,\eta\,(\sqrt{\omega}\,x\,\#1\&)} \star\star e^{i\,\zeta\left(-C\left[\frac{\partial_x \#1}{\sqrt{\omega}}\&,\frac{\partial_x \#1}{\sqrt{\omega}}\&\right]+C[\sqrt{\omega}\,x\,\#1\&,\sqrt{\omega}\,x\,\#1\&]\right)} == e^{i\,\phi\,(\#1\&)} \star\star$$
$$e^{\zeta\,(-\eta+\xi\,\mathrm{Cot}[\zeta])\left(\frac{\partial_x \#1}{\sqrt{\omega}}\&\right)+i\,\zeta\,(\xi+\eta\,\mathrm{Cot}[\zeta])\,(\sqrt{\omega}\,x\,\#1\&)+(i\,\zeta)\left[-C\left[\frac{\partial_x \#1}{\sqrt{\omega}}\&,\frac{\partial_x \#1}{\sqrt{\omega}}\&\right]+C[\sqrt{\omega}\,x\,\#1\&,\sqrt{\omega}\,x\,\#1\&]\right]}$$

$$e^{\frac{\xi}{\sqrt{\omega}}\,\partial_x}\; e^{i\sqrt{\omega}\;\eta\,x}\; e^{i\,\zeta\left(-\frac{\partial_x^2}{\omega}+\omega\,x^2\right)} = e^{i\,\phi}\; e^{i\,\zeta\left(\frac{\partial_x^2}{\omega}+\omega\,x^2\right)+\frac{\mu}{\sqrt{\omega}}\,\partial_x+i\sqrt{\omega}\;\nu\,x} \qquad (8)$$

A po použití nekonečnorozměrné reprezentace bez parametru dostanu:

```
StylePrint[Eq //. pUseInfRepre //. {Composition → C},
 FontSize → 8]
```

$$e^{\xi\,(\partial_x \#1\&)} \star\star e^{i\,\eta\,(x\,\#1\&)} \star\star e^{i\,\zeta\,\left(-(\partial_{\{x,2\}}\#1\&)+\left(x^2\,\#1\&\right)\right)} == e^{i\,\phi\,(\#1\&)} \star\star$$
$$e^{\zeta\,(-\eta+\xi\,\mathrm{Cot}[\zeta])\,(\partial_x \#1\&)+i\,\zeta\,(\xi+\eta\,\mathrm{Cot}[\zeta])\,(x\,\#1\&)+(i\,\zeta)\left[-(\partial_{\{x,2\}}\#1\&)+\left(x^2\,\#1\&\right)\right]}$$

$$e^{\xi\,\partial_x}\; e^{i\,\eta\,x}\; e^{i\,\zeta\,(-\partial_x^2+x^2)} = e^{i\,\phi}\; e^{i\,\zeta\,(-\partial_x^2+x^2)+\mu\,\partial_x+i\,\nu\,x} \qquad (9)$$

■ Vyjádření operátorů

▲ První dva operátory

Obě strany vztahů (7),(8) a (9) jsou dobře definovány v $L^2(\mathbb{R}, \mathrm{d}x)$ v případě, že koeficienty ξ, η a ζ jsou reálné. Operátory $e^{\xi\,\partial_x}$, $e^{i\,\eta\,x}$, $e^{i\,\zeta\left(-\partial_x^2+x^2\right)}$ jsou unitární. Výynam první dvou jsem již objasnil v kapitole 3.1.2.

```
Print["e^(ξ ∂x) f = ", ExpM[ξ, f[x]], "\t", "e^(i η x) f = ",
 ExpT[η, f[x]]]
```

$$e^{\xi\,\partial_x}\;\mathtt{f} = \mathtt{f}[\mathtt{x}+\xi] \qquad e^{\mathrm{i}\,\eta\,\mathtt{x}}\;\mathtt{f} = e^{\mathrm{i}\,\mathtt{x}\,\eta}\,\mathtt{f}[\mathtt{x}]$$

▲ Úprava operátoru $e^{i\,\zeta\left(-\partial_x^2+x^2\right)}$

Rozepíšu operátor do Taylorovy řady:

$$e^{i\,\zeta\left(-\partial_x^2+x^2\right)} = \sum_{k=0}^{\infty} \frac{\left(i\,\zeta\,\left(-\partial_{\{x,2\}}\#1+x^2\,\#1\right)\&\right)^k}{k!}$$

A dále platí: $e^{\alpha\,\partial_x}\,(-\partial_x^2+x^2)\;e^{-\alpha\,\partial_x} = -\partial_x^2+(x+\alpha)^2$ a
$e^{-\mathrm{i}\,\beta\,x}\;(-\partial_x^2+x^2)\;e^{\mathrm{i}\,\beta\,x} = -(\partial_x+\mathrm{i}\beta)^2+x^2$

```
ExpM[α, (- ∂{x,2} #1 + x^2 #1) &[ExpM[-α, fce[x]]]]
```

$$(x+\alpha)^2\,\mathtt{fce}[\mathtt{x}] - \mathtt{fce}''[\mathtt{x}]$$

$$(x+\alpha)^2\,\mathtt{fce}[\mathtt{x}] - \mathtt{fce}''[\mathtt{x}]$$

```
ExpT[-β, (-∂{x,2} #1 + x^2 #1) &[ExpT[β, fce[x]]]] -
  (-Composition[(∂x #1 + i β #1) &, (∂x #1 + i β #1) &][fce[x]] +
     x^2 #1 &[fce[x]]) // FullSimplify
```

```
0
```

```
0
```

Dále na základě předchozích úvah srovnám následující výrazy:

```
i ζ (-∂{x,2} fce[x] + x^2 fce[x]) + μ ∂x fce[x] + i ν x fce[x] -
  ExpT[-β,
   ExpM[α, i ζ (-∂{x,2} #1 + x^2 #1) &[ExpM[-α, ExpT[β, fce[x]]]]]] //.
 {β → μ/(2 ζ), α → ν/(2 ζ)} // Simplify
```

$$-\frac{i\,(\mu^2+\nu^2)\,\mathrm{fce}[x]}{4\,\zeta}$$

Platí tedy:

$$i\,\zeta\,(-\partial_x{}^2+x^2)+\mu\,\partial_x+i\,\nu\,x = e^{-\frac{i\,\mu}{2\,\zeta}\,x}\,e^{\frac{\nu}{2\,\zeta}\,\partial_x}\,i\,\zeta\,(-\partial_x{}^2+x^2)\,e^{-\frac{\nu}{2\,\zeta}\,\partial_x}\,e^{\frac{i\,\mu}{2\,\zeta}\,x}-\frac{i\,(\mu^2+\nu^2)}{4\,\zeta}$$

A dosadím do rozvoje výrazu vyskytujícího se na pravé straně vztahu (9).

$$e^{i\,\zeta\,(-\partial_x{}^2+x^2)+\mu\,\partial_x+i\,\nu\,x+\frac{i\,(\mu^2+\nu^2)}{4\,\zeta}} = \sum_{k=0}^{\infty}\frac{\left(i\,\zeta\,(-\partial_x{}^2+x^2)+\mu\,\partial_x+i\,\nu\,x+\frac{i\,(\mu^2+\nu^2)}{4\,\zeta}\right)^k}{k!}$$

Po několika krocích úprav:

$$e^{i\,\zeta\,(-\partial_x{}^2+x^2)+\mu\,\partial_x+i\,\nu\,x} = e^{-\frac{i\,(\mu^2+\nu^2)}{4\,\zeta}}\,e^{-\frac{i\,\mu}{2\,\zeta}\,x}\,e^{\frac{\nu}{2\,\zeta}\,\partial_x}\,e^{i\,\zeta\,(-\partial_x{}^2+x^2)}\,e^{-\frac{\nu}{2\,\zeta}\,\partial_x}\,e^{\frac{i\,\mu}{2\,\zeta}\,x} \qquad (10)$$

A po dosazení do vztahu (9) dostanu:

$$e^{\xi\,\partial_x}\,e^{i\,\eta\,x}\,e^{i\,\zeta\,(-\partial_x{}^2+x^2)} = e^{i\,\phi}\,e^{-\frac{i\,(\mu^2+\nu^2)}{4\,\zeta}}\,e^{-\frac{i\,\mu}{2\,\zeta}\,x}\,e^{\frac{\nu}{2\,\zeta}\,\partial_x}\,e^{i\,\zeta\,(-\partial_x{}^2+x^2)}\,e^{-\frac{\nu}{2\,\zeta}\,\partial_x}\,e^{\frac{i\,\mu}{2\,\zeta}\,x}$$

A upravím použitím operátorů $e^{\frac{i\,\mu}{2\,\zeta}\,x}$ a $e^{-\frac{\nu}{2\,\zeta}\,\partial_x}$ zleva na obě strany rovnosti. Přičemž využiji toho, že $e^{\frac{i\,\mu}{2\,\zeta}\,x}\,e^{\xi\,\partial_x} = e^{-\frac{i\,\mu\,\xi}{2\,\zeta}}\,e^{\xi\,\partial_x}\,e^{\frac{i\,\mu}{2\,\zeta}\,x}$, neboť:

```
ExpT[μ/(2 ζ), ExpM[ξ, fce[x]]] - e^(-i μ ξ/(2 ζ)) ExpM[ξ, ExpT[μ/(2 ζ), fce[x]]] //
 FullSimplify
```

```
0
```

Nakonec tedy dostávám rovnost:

$$e^{\left(\xi-\frac{\nu}{2\zeta}\right)\partial_x}\, e^{i\left(\eta+\frac{\mu}{2\zeta}\right)x}\, e^{i\zeta\left(-\partial_x^2+x^2\right)} = \tag{11}$$

$$e^{i\phi-\frac{i\left(\mu^2+\nu^2\right)}{4\zeta}+\frac{i\mu\xi}{2\zeta}}\, e^{i\zeta\left(-\partial_x^2+x^2\right)}\, e^{-\frac{\nu}{2\zeta}\partial_x}\, e^{\frac{i\mu}{2\zeta}x}$$

▲ Vlastní funkce operátoru $(-\partial_x^2+x^2)$

Jaký tvar mají vlastní funkce operátoru zjistím řešením rovnice:

```
DSolve[{-∂{x,2} ψ[x] + x^2 ψ[x] == λ ψ[x]}, ψ[x], x]
```

```
{{ψ[x] → e^(-x^2/2) C[1] HermiteH[1/2 (-1 + λ), x] +
     e^(-x^2/2) C[2] Hypergeometric1F1[(1 - λ)/4, 1/2, x^2]}}
```

Ze základního kurzu kvantové mechaniky je známo, že vlastní čísla jsou $\lambda_n = 1+2n,\ n\in\mathbb{N}$ a základní stav oscilátoru s nejnižší energií odpovídá volbě $n=0$.

Proto volím vlastní funkci $\psi_0(x) = e^{-\frac{x^2}{2}}$ a k ní vlastní číslo $\lambda_0 = 1$.

```
ψ0[x_] := e^(-x^2/2);
```

Pro tuto funkci platí, že $\left(e^{i\zeta\left(-\partial_x^2+x^2\right)}\,\psi_0\right)(x) = e^{i\zeta\lambda_0}\,\psi_0(x)$.

Následně obě strany rovnosti (11) aplikuji na funkci ψ_0 a ještě je obě vynásobím zleva touto funkcí.

$$\psi_0(x)\, e^{\left(\xi-\frac{\nu}{2\zeta}\right)\partial_x}\, e^{i\left(\eta+\frac{\mu}{2\zeta}\right)x}\, e^{i\zeta\left(-\partial_x^2+x^2\right)}\,\psi_0(x) =$$

$$\psi_0(x)\, e^{i\phi-\frac{i\left(\mu^2+\nu^2\right)}{4\zeta}+\frac{i\mu\xi}{2\zeta}}\, e^{i\zeta\left(-\partial_x^2+x^2\right)}\, e^{-\frac{\nu}{2\zeta}\partial_x}\, e^{\frac{i\mu}{2\zeta}x}\,\psi_0(x)$$

Což lze díky vlastnosti vlastní funkce přepsat na:

$$e^{i\zeta}\,\psi_0(x)\, e^{\left(\xi-\frac{\nu}{2\zeta}\right)\partial_x}\, e^{i\left(\eta+\frac{\mu}{2\zeta}\right)x}\,\psi_0(x) =$$

$$e^{i\phi-\frac{i\left(\mu^2+\nu^2\right)}{4\zeta}+\frac{i\mu\xi}{2\zeta}}\,\psi_0(x)\, e^{i\zeta\left(-\partial_x^2+x^2\right)}\, e^{-\frac{\nu}{2\zeta}\partial_x}\, e^{\frac{i\mu}{2\zeta}x}\,\psi_0(x)$$

■ Řešení rovnice

Obě strany rovnosti zintegruji. Při integrování lze provést záměnu $\psi_0(x)\, e^{i\zeta\left(-\partial_x{}^2+x^2\right)} = e^{i\zeta\left(-\partial_x{}^2+x^2\right)}\psi_0(x) = e^{i\zeta\lambda_0}\psi_0(x) = e^{i\zeta}\psi_0(x)$, po které se zkrátí členy $e^{i\zeta}$.

```
LProd = ∫_-∞^∞ ψ0[x] ExpM[(ξ - ν/(2 ζ)), ExpT[(η + μ/(2 ζ)), ψ0[x]]] dx
```

```
Integrate::gener : Unable to check convergence. More…
```

$$e^{-\frac{\mu^2+2\,i\,\mu\,\nu+\nu^2+4\,\zeta\,(\eta\,(\mu+i\,\nu)-i\,\mu\,\xi-\nu\,\xi)+4\,\zeta^2\left(\eta^2-2\,i\,\eta\,\xi+\xi^2\right)}{16\,\zeta^2}}\sqrt{\pi}$$

```
RProd =
e^(i ϕ - (i (μ^2 + ν^2))/(4 ζ) + (i μ ξ)/(2 ζ))
  ∫_-∞^∞ ψ0[x] ExpM[(-ν/(2 ζ)), ExpT[(μ/(2 ζ)), ψ0[x]]] dx // Simplify
```

$$\frac{e^{\frac{i\left((i-4\,\zeta)\,\mu^2+(i-4\,\zeta)\,\nu^2-2\,\mu\,(\nu-4\,\zeta\,\xi)+16\,\zeta^2\,\phi\right)}{16\,\zeta^2}}}{\sqrt{\pi}}$$

A rovnici vyřeším vzhledem k jediné neznámé ϕ.

```
pϕ =
 Solve[Exponent[LProd, E] == Exponent[RProd, E], ϕ] //.
   pSubstCoeffs // Flatten // FullSimplify
```

$$\left\{\phi \to \frac{1}{16}\left(4\,\eta\,\xi - 2\left(\eta^2+3\,\xi^2\right)\mathrm{Cot}[\zeta] + \left((-i+4\,\zeta)\,\eta^2+2\,\eta\,\xi+(-i+4\,\zeta)\,\xi^2\right)\mathrm{Csc}[\zeta]^2\right)\right\}$$

Tedy $f = i\,\phi$ a $\phi = \frac{1}{2}\,\xi\,\eta + \frac{2\,\zeta-\sin(2\,\zeta)}{8\,\sin^2(\zeta)}\,(\xi^2+\eta^2)$, protože

```
ϕ - (1/2 ξ η + (2 ζ - Sin[2 ζ])/(8 Sin[ζ]^2) (ξ^2 + η^2)) //. pϕ // FullSimplify
```

```
0
```

3.3.5. Shrnutí získaných výsledků

Po nalezení posledního koeficientu $f = i\,\phi$ mohu zapsat celý vztah (9).

$$e^{\xi\,\partial_x}\, e^{i\,\eta\,x}\, e^{i\,\zeta\left(-\partial_x{}^2+x^2\right)} = e^{i\,\phi}\, e^{i\,\zeta\left(-\partial_x{}^2+x^2\right)+\mu\,\partial_x+i\,\nu\,x} \qquad (12)$$

A s parametrem ω doplním vztah (8):

$$e^{\xi\,\frac{1}{\sqrt{\omega}}\,\partial_x}\, e^{i\,\eta\,\sqrt{\omega}\,x}\, e^{i\,\zeta\left(-\frac{1}{\omega}\,\partial_x{}^2+\omega\,x^2\right)} = e^{i\,\phi}\, e^{i\,\zeta\left(-\frac{1}{\omega}\,\partial_x{}^2+\omega\,x^2\right)+\mu\,\frac{1}{\sqrt{\omega}}\,\partial_x+i\,\nu\,\sqrt{\omega}\,x} \qquad (13)$$

A obecně pro Lieovu algebru g:

$$e^{\xi P}\, e^{i\eta Q}\, e^{i\zeta(V-U)} = e^{i\phi}\, e^{i\zeta(V-U)+\mu P+i\nu Q} \tag{14}$$

Přičemž vždy:

$$\mu = \zeta\,(ctg(\zeta)\,\xi - \eta),\quad \nu = \zeta\,(\xi + ctg(\zeta)\,\eta),$$
$$\phi = \frac{1}{2}\,\xi\,\eta + \frac{2\zeta - Sin(2\zeta)}{8\,Sin^2(\zeta)}\,(\xi^2+\eta^2) \tag{15}$$

A ještě po zpětné substituci $f = i\,\phi$ dostávám úplný vztah (5).

```
Solve[ϕ == 1/2 ξ η + (2 ζ - Sin[2 ζ])/(8 Sin[ζ]^2) (ξ^2 + η^2) //.
  {ξ → α, η → -i β, ζ → -i γ, ϕ → -i f, μ → d, ν → -i e}, f]
```

```
{{f → 1/8 (4 α β - 2 α^2 γ Csch[γ]^2 + 2 β^2 γ Csch[γ]^2 +
      α^2 Csch[γ]^2 Sinh[2 γ] - β^2 Csch[γ]^2 Sinh[2 γ])}}
```

$$e^{\alpha P}\, e^{\beta Q}\, e^{\gamma(V-U)} = e^{f I}\, e^{\gamma(V-U)+dP+eQ},$$
$$d = \gamma\,(\beta + \alpha\,Coth[\gamma]),\quad e = \gamma\,(\alpha + \beta\,Coth[\gamma]), \tag{16}$$
$$f = \frac{1}{2}\,\alpha\beta - (\alpha^2-\beta^2)\,\frac{2\gamma - Sinh(2\gamma)}{8\,Sinh^2(\gamma)}$$

3.3.6. Srovnání vztahu (4) s rozvojem získaných analytických funkcí

$$Vztah\ (4) : Z = -U\gamma + V\gamma +$$
$$I\left(\frac{\alpha\beta}{2} + \frac{\alpha^2\gamma}{6} - \frac{\beta^2\gamma}{6} - \frac{\alpha^2\gamma^3}{45} + \frac{\beta^2\gamma^3}{45} + \frac{\alpha^2\gamma^5}{315} - \frac{\beta^2\gamma^5}{315} + \ldots\right) +$$
$$P\left(\alpha + \beta\gamma + \frac{\alpha\gamma^2}{3} - \frac{\alpha\gamma^4}{45} + \frac{2\alpha\gamma^6}{945} + \ldots\right) +$$
$$Q\left(\beta + \alpha\gamma + \frac{\beta\gamma^2}{3} - \frac{\beta\gamma^4}{45} + \frac{2\beta\gamma^6}{945} + \ldots\right)$$

```
Print["P:d = ", Series[γ (β + α Coth[γ]), {γ, 0, 6}]];
Print["Q:e = ", Series[γ (β + α Coth[γ]), {γ, 0, 6}]];
Print["I:f = ", Series[1/2 α β - (α^2 - β^2) (2 γ - Sinh[2 γ])/(8 Sinh[γ]^2),
  {γ, 0, 6}]];
```

$$P{:}d = \alpha + \beta\gamma + \frac{\alpha\gamma^2}{3} - \frac{\alpha\gamma^4}{45} + \frac{2\alpha\gamma^6}{945} + O[\gamma]^7$$

$$Q{:}e = \alpha + \beta\gamma + \frac{\alpha\gamma^2}{3} - \frac{\alpha\gamma^4}{45} + \frac{2\alpha\gamma^6}{945} + O[\gamma]^7$$

$$I{:}f = \frac{\alpha\beta}{2} + \left(\frac{\alpha^2}{6} - \frac{\beta^2}{6}\right)\gamma + \left(-\frac{\alpha^2}{45} + \frac{\beta^2}{45}\right)\gamma^3 + \left(\frac{\alpha^2}{315} - \frac{\beta^2}{315}\right)\gamma^5 + O[\gamma]^7$$

Je vidět, že tyto koeficienty se do sledovaných řádů opravdu shodují.

3.4. Hledaný tvar propagátoru

3.4.1. Odvození

Vyjdu z tvaru propagátoru podle Ensse a Veseliće a upravím ho pomocí výsledků kapitol 3.2. a 3.3.: $U(t,\ s) = e^{-i\,\varphi_1(t,s)\,x}\, e^{\frac{\varphi_2(t,s)}{\omega}\,\partial_x}\, e^{-i\,H_\omega\,(t-s)-i\,\psi(t,s)}$.

Ověřím následující rovnost, kterou použiji dále.

$$e^{-i\,\varphi_1(t,s)\,x}\, e^{\frac{\varphi_2(t,s)}{\omega}\,\partial_x} = e^{i\,\frac{\varphi_2(t,s)}{\omega}\,\varphi_1(t,s)}\, e^{\frac{\varphi_2(t,s)}{\omega}\,\partial_x}\, e^{-i\,\varphi_1(t,s)\,x}.$$

```
ExpT[-φ1[t, s] , ExpM[φ2[t, s]/ω, f[x]]] -
  e^(i φ1[t,s] φ2[t,s]/ω) ExpM[φ2[t, s]/ω, ExpT[-φ1[t, s], f[x]]] //
 Simplify

0
```

Lze tedy psát:

$$U(t,\ s) = e^{i\,\varphi_1(t,s)\,\frac{\varphi_2(t,s)}{\omega}}\, e^{\frac{\varphi_2(t,s)}{\omega}\,\partial_x}\, e^{-i\,\varphi_1(t,s)\,x}\, e^{-i\,H_\omega\,(t-s)-i\,\psi(t,s)} = e^{i\,\varphi_1(t,s)\,\frac{\varphi_2(t,s)}{\omega}-i\,\psi(t,s)}\, e^{\frac{\varphi_2(t,s)}{\omega}\,\partial_x}\, e^{-i\,\varphi_1(t,s)\,x}\, e^{-i\,H_\omega\,(t-s)}.$$

A výraz $e^{\frac{\varphi_2(t,s)}{\omega}\,\partial_x}\, e^{-i\,\varphi_1(t,s)\,x}\, e^{-i\,\frac{1}{2}\,\left(-\partial_x{}^2+\omega^2\,x^2\right)\,(t-s)}$ srovnám s levou stranou vztahu (13), který zní:

$$e^{\xi\,\frac{1}{\sqrt{\omega}}\,\partial_x}\, e^{i\,\eta\,\sqrt{\omega}\,x}\, e^{i\,\zeta\left(-\frac{1}{\omega}\,\partial_x{}^2+\omega\,x^2\right)} = e^{i\,\phi}\, e^{i\,\zeta\left(-\frac{1}{\omega}\,\partial_x{}^2+\omega\,x^2\right)+\mu\,\frac{1}{\sqrt{\omega}}\,\partial_x+i\,\nu\,\sqrt{\omega}\,x}.$$

```
pSubst[ξ, η, ζ] =
 Solve[ξ 1/√ω == φ2[t, s]/ω && i η √ω == -i φ1[t, s] &&
   -i 1/2 (t - s) == i ζ/ω, {η, ξ, ζ}] // Flatten

{ξ → φ2[t, s]/√ω, η → -φ1[t, s]/√ω, ζ → 1/2 (s - t) ω}
```

Tím jsem získal substituční vztahy pro koeficienty ξ, η a ζ. Nyní tedy mohu dosadit pravou stranu vztahu (13) do výrazu pro propagátor a po substituci dostanu hledaný tvar propagátoru, ve formě jediného exponenciálního zobrazení.

$$U(t,\ s)\ =\ e^{i\,\varphi_1(t,s)\,\frac{\varphi_2(t,s)}{\omega}\,-i\,\psi(t,s)\,+i\,\phi\,-i\,H_\omega(t-s)\,+\,\frac{\mu}{\sqrt{\omega}}\,\partial_x\,+i\,\nu\,\sqrt{\omega}\,x},$$

$$\mu = \xi\ (ctg(\zeta)\ \xi - \eta),\ \nu = \xi\ (\xi + ctg(\zeta)\ \eta)$$

$$\phi = \frac{1}{2}\ \xi\ \eta + \frac{2\,\zeta - Sin(2\,\zeta)}{8\,Sin^2(\zeta)}\ (\xi^2 + \eta^2) \qquad (17)$$

$$\xi = \frac{\varphi_2[t,s]}{\sqrt{\omega}},\ \eta = -\frac{\varphi_1[t,s]}{\sqrt{\omega}},\ \zeta = \frac{1}{2}\ (s - t)\ \omega$$

3.4.2. Konečná úprava

Tento výraz ještě upravím dosazením funkcí φ_1 a φ_2. Pro práci s integrály budu potřebovat následující pravidlo pro vytýkání konstant před integrál.

```
pIntegrals =
  {A_. ∫_{s_}^{t_} a_ f_[u_] du_ + B_. ∫_{s_}^{t_} b_ f_[u_] du_ →
    ∫_s^t (A a + B b) f[u] du};
```

- **Koeficient u operátoru ∂_x**

```
μ/√ω //. pSubstCoeffs //. pSubstξηζ //. pIntFunkce //. pIntegrals //
FullSimplify
```

$$\frac{1}{2}\ (s - t)\ \sqrt{\omega}\ \int_s^t \frac{\mathrm{Csc}\left[\frac{1}{2}\ (s - t)\ \omega\right]\ f[u]\ \mathrm{Sin}\left[\frac{1}{2}\ (s + t - 2\,u)\ \omega\right]}{\sqrt{\omega}}\ du$$

Celkově tedy označím:

$$\mu\ (t,\ s)\ =\ \frac{\mu}{\sqrt{\omega}}\ =\ \frac{(t-s)}{2\ sin\left(\frac{\omega}{2}\ (t-s)\right)}\ \int_s^t\ sin\left(\omega\ \left(\frac{t+s}{2}\ -u\right)\right)\ f\ (u)\ du$$

- **Koeficient u operátoru x**

```
i ν √ω //. pSubstCoeffs //. pSubstξηζ //. pIntFunkce //.
  pIntegrals // FullSimplify
```

$$\frac{1}{2}\ i\ (s - t)\ \omega^{3/2}\ \int_s^t -\frac{\mathrm{Cos}\left[\frac{1}{2}\ (s + t - 2\,u)\ \omega\right]\ \mathrm{Csc}\left[\frac{1}{2}\ (s - t)\ \omega\right]\ f[u]}{\sqrt{\omega}}\ du$$

Celkově tedy označím:

$$\nu(t,\ s)\ =\ \nu\ \sqrt{\omega}\ =\ -\frac{\omega\ (t-s)}{2\ \sin\left(\frac{\omega}{2}\ (t-s)\right)}\ \int_s^t \cos\left(\omega\ \left(\frac{t+s}{2}\ -u\right)\right)\ f(u)\ du$$

■ **A ještě upravíme zbývající část exponentu**

```
φ1[t, s] φ2[t, s] / ω - ψ[t, s] + ϕ //. pϕ //. pSubstξηζ //
FullSimplify
```

```
(-8 ω (-1 + Cos[(s - t) ω]) ψ[t, s] +
   (i - 2 s ω + 2 t ω + Sin[(s - t) ω]) φ1[t, s]^2 +
   2 (-2 + 3 Cos[(s - t) ω]) φ1[t, s] φ2[t, s] +
   (i - 2 s ω + 2 t ω + 3 Sin[(s - t) ω]) φ2[t, s]^2) /
 (8 ω (-1 + Cos[(s - t) ω]))
```

Celkově tedy označím:

$$\sigma(t,\,s) = -\psi[t,\,s] + \frac{1}{2\,\omega}\,\varphi_1(t,\,s)\,\varphi_2(t,\,s) - \frac{\omega\,(t-s) - \sin(\omega\,(t-s))}{8\,\omega\,\sin\left(\frac{\omega}{2}\,(t-s)\right)^2}\,\left(\varphi_1(t,\,s)^2 + \varphi_2(t,\,s)^2\right)$$

A propagátor lze tedy zapsat v následujícím tvaru:

$$\begin{aligned}
U(t,\,s) &= exp\left(-i\,H_\omega(t-s) + \mu(t,\,s)\,\partial_x + i\,\nu(t,\,s)\,x + i\,\sigma(t,\,s)\right),\\
\mu(t,\,s) &= \frac{(t-s)}{2\,sin\left(\frac{\omega}{2}\,(t-s)\right)}\int_s^t sin\left(\omega\left(\frac{t+s}{2} - u\right)\right) f(u)\,du,\\
\nu(t,\,s) &= -\frac{\omega\,(t-s)}{2\,sin\left(\frac{\omega}{2}\,(t-s)\right)}\int_s^t cos\left(\omega\left(\frac{t+s}{2} - u\right)\right) f(u)\,du,
\end{aligned} \tag{18}$$

$$\sigma(t,\,s) = -\frac{1}{2}\int_s^t \left(\varphi_1(v,\,s)^2 - \varphi_2(v,\,s)^2\right)dv + \frac{1}{2\,\omega}\,\varphi_1(t,\,s)\,\varphi_2(t,\,s) - \frac{\omega\,(t-s) - sin(\omega\,(t-s))}{8\,\omega\,sin\left(\frac{\omega}{2}\,(t-s)\right)^2}\,\left(\varphi_1(t,\,s)^2 + \varphi_2(t,\,s)^2\right)$$

3.4.3. Kontrola tvaru propagátoru

V této kapitole ověřím správnost odvozeného vztahu pro propagátor.

■ Potřebná pravidla

■ **Pravidlo pro dosazení Hamiltoniánu**

```
pHω = {Hω → (1/2 (-∂{x,2} #1 + ω^2 x^2 #1) &)};
```

■ Pravidla pro práci se složenými operátory

```
pCompositionOuterLinearity = {

   (A_ Composition[a_, b_])[f_] → A Composition[a, b][f],
   (A_ (B_))[f_] → A B[f]};

pCompositionInnerLinearity =
  {Composition[(c : (_Integer | _Rational | _Real)) a_,
     (d : (_Integer | _Rational | _Real)) b_] → c d Composition[a, b],
   Composition[(c : (_Integer | _Rational | _Real)) a_, b_] →
    c Composition[a, b],
   Composition[ a_, (d : (_Integer | _Rational | _Real)) b_] →
    d Composition[a, b]
  };
```

■ Ověření tvaru propagátoru (18)

Ověření provedu tím, že aplikuji původní tvar propagátoru podle Ensse a Veseliće: $U(t, s) = e^{-i\varphi_1(t,s)\,x}\, e^{\frac{\varphi_2(t,s)}{\omega}\partial_x}\, e^{-i H_\omega (t-s) - i\psi(t,s)}$ a odvozený tvar (18) na stejnou libovolnou funkci $fce(x)$. Srovnám vlastně pouze exponenty aplikované na tuto funkci. V prvním případě vyjádřím exponent pomocí Baker-Campbell-Hausdorffovy formule vyjádřené do určitého řádu **RAD** a aplikuji na funkci $fce(x)$ (toto vyjádření označím **LS**), v druhém případě je exponent již daný a tak ho rovnou aplikuji na tuto funkci $fce(x)$ a výsledek rozvinu do řady také do řádu **RAD** (označení **PS**).

Vztahy pro koeficienty z výrazu (18) převedu na substituční pravidla.

$$p\mu\nu\phi = \Big\{\mu[t, s] \to \frac{(t-s)}{2\,\mathrm{Sin}\left[\frac{\omega}{2}(t-s)\right]} \int_s^t \mathrm{Sin}\left[\omega\left(\frac{t+s}{2} - u\right)\right] f[u]\,du,$$

$$\nu[t, s] \to -\frac{\omega(t-s)}{2\,\mathrm{Sin}\left[\frac{\omega}{2}(t-s)\right]} \int_s^t \mathrm{Cos}\left[\omega\left(\frac{t+s}{2} - u\right)\right] f[u]\,du,$$

$$\sigma[t, s] \to -\frac{1}{2}\int_s^t \left(\varphi_1[v, s]^2 - \varphi_2[v, s]^2\right) dv + \frac{1}{2\omega}\varphi_1[t, s]\,\varphi_2[t, s] - \frac{\omega(t-s) - \mathrm{Sin}[\omega(t-s)]}{8\,\omega\,\mathrm{Sin}\left[\frac{\omega}{2}(t-s)\right]^2}\left(\varphi_1[t, s]^2 + \varphi_2[t, s]^2\right)\Big\};$$

$$p\xi\eta\zeta = \Big\{\xi \to \frac{\varphi_2[t, s]}{\sqrt{\omega}},\ \eta \to -\frac{\varphi_1[t, s]}{\sqrt{\omega}},\ \zeta \to \frac{1}{2}(s-t)\,\omega\Big\};$$

Nastavení požadovaného řádu přesnosti pro ověření. Korektního výsledku jsem dosáhl pro řád 8. Vyšší řád už nezvládl hardware mého počítače. Nyní z důvodu kratšího zápisu ověřím řád 3.

```
RAD = 3;
```

■ Vyjádření tvaru (18) do mocninné řady v ξ, η, ζ.

```
PS =
  (-i (t - s) (-1/2 ∂{x,2} #1 + ω^2 x^2/2 #1) + μ[t, s] ∂x #1 +
      i ν[t, s] x #1 + i σ[t, s] #1) &[fce[x]] //. pμνφ //.
  pIntFunkce;

PS = Normal[Series[PS, {t, s, RAD}]] // Simplify // Expand
```

$$\frac{1}{2} i s x^2 \omega^2 fce[x] - \frac{1}{2} i t x^2 \omega^2 fce[x] + i s x f[s] fce[x] -$$
$$i t x f[s] fce[x] - \frac{1}{2} i s^2 x fce[x] f'[s] + i s t x fce[x] f'[s] -$$
$$\frac{1}{2} i t^2 x fce[x] f'[s] + \frac{1}{12} s^3 f'[s] fce'[x] - \frac{1}{4} s^2 t f'[s] fce'[x] +$$
$$\frac{1}{4} s t^2 f'[s] fce'[x] - \frac{1}{12} t^3 f'[s] fce'[x] + \frac{1}{6} i s^3 x fce[x] f''[s] -$$
$$\frac{1}{2} i s^2 t x fce[x] f''[s] + \frac{1}{2} i s t^2 x fce[x] f''[s] -$$
$$\frac{1}{6} i t^3 x fce[x] f''[s] - \frac{1}{2} i s fce''[x] + \frac{1}{2} i t fce''[x]$$

■ Vyjádření propagátoru pomocí formule

```
Clear[aOp, bOp, cOp];
pOp = {aOp → -i φ1 Q, bOp → φ2/ω P, cOp → -i t Hω + i s Hω - i ψ II};
zOp = BCHfle[aOp, bOp, RAD] //. pVytkniKonstanty //. pExpandCR //
   Expand;
zOp = zOp /. pOp //. pVytkniKonstanty;

zOp =
pPrevedNaZakladniTvar[zOp, PrvkyBaze, pRel]
```

$$-i Q \varphi 1 + \frac{P \varphi 2}{\omega} + \frac{i II \varphi 1 \varphi 2}{2 \omega}$$

A členy vyšších řádů se v **zOp** nikdy nevyskytnou. Lze tedy psát:

```
zOp = ϵ aOp + ϵ bOp + III ϵ^2/2;
```

Kde pod ***III*** jsem označil $i \frac{\varphi_1[t,s]\,\varphi_2[t,s]}{\omega} II$.

Nyní vyjádřím zbývající část propagátoru.

```
BCHfle[dOp, ϵ cOp, RAD] /. {dOp → zOp} //. pExpandCR // Expand;

% //. pVytkniKonstanty //. pRound[RAD] //. {ϵ → 1} //. pOp //.
     {III → i φ1 φ2/ω II} //. pVytkniKonstanty //. pExpandCR // Expand;
```

Operaci násobení převedu na složení zobrazení a použiji nekonečnou reprezentaci, která odpovídá operátorům v propagátoru.

```
% //. pVytkniKonstanty //.
    {NonCommutativeMultiply → Composition} //. pUseInfRepre //.
  pHω;
```

Operátor mám již připraven a nyní ho aplikuji na funkci. Pak ještě dosadím funkce z definice propagátoru a výraz rozvinu do řady.

```
Through[%[fce[x]], Plus] //. pCompositionOuterLinearity //
  FullSimplify;
```

Ještě dosadím funkce z definice propagátoru a výraz rozvinu do řady.

```
% /. {φ1 → φ1[t, s], φ2 → φ2[t, s], ψ → ψ[t, s]} //. pIntFunkce;
```

```
LS = Normal[Series[%, {t, s, RAD}]] // FullSimplify // Expand
```

$$\frac{1}{2}\,i\,s\,x^2\,\omega^2\,\text{fce}[x] - \frac{1}{2}\,i\,t\,x^2\,\omega^2\,\text{fce}[x] + i\,s\,x\,f[s]\,\text{fce}[x] -$$
$$i\,t\,x\,f[s]\,\text{fce}[x] - \frac{1}{2}\,i\,s^2\,x\,\text{fce}[x]\,f'[s] + i\,s\,t\,x\,\text{fce}[x]\,f'[s] -$$
$$\frac{1}{2}\,i\,t^2\,x\,\text{fce}[x]\,f'[s] + \frac{1}{12}\,s^3\,f'[s]\,\text{fce}'[x] - \frac{1}{4}\,s^2\,t\,f'[s]\,\text{fce}'[x] +$$
$$\frac{1}{4}\,s\,t^2\,f'[s]\,\text{fce}'[x] - \frac{1}{12}\,t^3\,f'[s]\,\text{fce}'[x] + \frac{1}{6}\,i\,s^3\,x\,\text{fce}[x]\,f''[s] -$$
$$\frac{1}{2}\,i\,s^2\,t\,x\,\text{fce}[x]\,f''[s] + \frac{1}{2}\,i\,s\,t^2\,x\,\text{fce}[x]\,f''[s] -$$
$$\frac{1}{6}\,i\,t^3\,x\,\text{fce}[x]\,f''[s] - \frac{1}{2}\,i\,s\,\text{fce}''[x] + \frac{1}{2}\,i\,t\,\text{fce}''[x]$$

A nakonec srovnám oba získané výrazy. Je vidět, že kontrola proběhla úspěšně.

```
LS - PS
```

```
0
```

Závěr

Seznámit se systémem *Mathematica* se mi myslím celkem povedlo. Nakonec jsem v tomto systému napsal i celou tuto práci.

Podařilo se mi naprogramovat funkci pro výpočet koeficientů v Baker-Campbell-Hausdorffově formuli a tuto formuli integrovat do takzvaného package, který je snadno použitelný v systému *Mathematica*. Funkci jsem s úspěchem otestoval do devátého řádu. Vyšší řády bohužel přesahují hranice výkonu mého počítače.

Ve třetí kapitole jsem vyjádřil propagátor podle Ensse a Veseliće v požadovaném tvaru a správnost tohoto tvaru jsem zkontroloval pomocí naprogramované formule a dalších funkcí.

Do budoucna se naskýtá možnost zefektivnit algoritmus pro vyjádření formule a implementovat verzi vyjadřující formuli nejen pro maticové, ale i obecné Lieovy algebry.

Seznam použité literatury

[1] V. Enns, K. Veselić: Bound states and propagating states for a time-dependent Hamiltonians, Ann. Inst. H. Poincaré, 1983

[2] Matthias W. Reinsch: A simple expression for the terms in the Baker-Campbell-Hausdorff series, Department of Physics, University of California, Berkeley, 2006

[3] Yu.A. Bakhturin, Campbell-Hausdorff formula, SpringerLink Encyclopaedia of Mathematics, 2001, http://eom.springer.de/C/c020090.htm

[4] T.S. Fofanova, Birkhoff–Witt theorem, SpringerLink Encyclopaedia of Mathematics, 2001, http://eom.springer.de/B/b016540.htm

Dodatky

Všechny potřebné materiály jsou umístěny na přiloženém CD. Jednotlivé části dodatků jsou členěny do adresářů.