In[ ]:=
```
PP_ = Identity; $k = 0; γ = 1; ℏ;
```

The “Speedy” Engine

Internal Utilities

Canonical Form:

In[ ]:=
```
CCF[ℰ_] := ExpandDenominator@ExpandNumerator@Together[
    Expand[ℰ] //. e^x_ e^y_ :> e^(x+y) /. e^x_ :> e^CCF[x]];
CF[ℰ_List] := CF /@ ℰ;
CF[sd_SeriesData] := MapAt[CF, sd, 3];
CF[ℰ_] := Module[
   {vs = Cases[ℰ, (y | b | t | a | x | η | β | τ | α | ξ)_, ∞] ⋃ {y, b, t, a, x, η, β, τ, α, ξ}},
   Total[CoefficientRules[Expand[ℰ], vs] /. (ps_ → c_) :> CCF[c] (Times @@ vs^ps)]
  ];
CF[ℰ_𝔼] := CF /@ ℰ; CF[𝔼_sp___[ℰs___]] := CF /@ 𝔼_sp[ℰs];
```

The Kronecker δ:

In[ ]:=
```
Kδ /: Kδ_{i_,j_} := If[i === j, 1, 0];
```

Equality, multiplication, and degree-adjustment of perturbed Gaussians; $\mathbb{E}[L, Q, P]$ stands for $e^{L+Q} P$:

In[ ]:=
```
𝔼 /: 𝔼[L1_, Q1_, P1_] ≡ 𝔼[L2_, Q2_, P2_] :=
  CF[L1 == L2] ∧ CF[Q1 == Q2] ∧ CF[Normal[P1 - P2] == 0];
𝔼 /: 𝔼[L1_, Q1_, P1_] 𝔼[L2_, Q2_, P2_] := 𝔼[L1 + L2, Q1 + Q2, P1 * P2];
𝔼[L_, Q_, P_]_$k_ := 𝔼[L, Q, Series[Normal@P, {ϵ, 0, $k}]];
```

Zip and Bind

Variables and their duals:

In[ ]:=
```
{t*, b*, y*, a*, x*, z*} = {τ, β, η, α, ξ, ζ};
{τ*, β*, η*, α*, ξ*, ζ*} = {t, b, y, a, x, z}; (u_i_)* := (u*)_i;
```

Upper to lower and lower to Upper:

In[ ]:=
```
U2l = {B_i_^p_. :> e^(-p ℏ γ b_i), B^p_. :> e^(-p ℏ γ b), T_i_^p_. :> e^(-p ℏ t_i), T^p_. :> e^(-p ℏ t), 𝒜_i_^p_. :> e^(p γ α_i), 𝒜^p_. :> e^(p γ α)};
l2U = {e^(c_. b_i_ + d_.) :> B_i^(-c/(ℏ γ)) e^d, e^(c_. b + d_.) :> B^(-c/(ℏ γ)) e^d,
    e^(c_. t_i_ + d_.) :> T_i^(-c/ℏ) e^d, e^(c_. t + d_.) :> T^(-c/ℏ) e^d,
    e^(c_. α_i_ + d_.) :> 𝒜_i^(c/γ) e^d, e^(c_. α + d_.) :> 𝒜^(c/γ) e^d,
    e^ℰ_ :> e^Expand@ℰ};
```

Derivatives in the presence of exponentiated variables:

In[ ]:=
```
Db[f_] := ∂b f - ℏ γ B ∂B f; Dbi_[f_] := ∂bi f - ℏ γ Bi ∂Bi f;
Dt[f_] := ∂t f - ℏ T ∂T f; Dti_[f_] := ∂ti f - ℏ Ti ∂Ti f;
Dα[f_] := ∂α f + γ 𝒜 ∂𝒜 f; Dαi_[f_] := ∂αi f + γ 𝒜i ∂𝒜i f;
Dv_[f_] := ∂v f; D{v_,0}[f_] := f; D{}[f_] := f; D{v_,n_Integer}[f_] := Dv[D{v,n-1}[f]];
D{l_List,ls___}[f_] := D{ls}[Dl[f]];
```

Finite Zips:

In[ ]:=
```
collect[sd_SeriesData, ξ_] := MapAt[collect[#, ξ] &, sd, 3];
collect[ℰ_, ξ_] := Collect[ℰ, ξ];
Zip{}[P_] := P;
Zipξs_[Ps_List] := Zipξs /@ Ps;
Zip{ξ_,ξs___}[P_] :=
  (collect[P // Zip{ξs}, ξ] /. f_. ξ^d_. :> (D{ξ*,d}[f])) /. ξ* → 0 /.
   ((ξ* /. {b → B, t → T, α → 𝒜}) → 1)
```

QZip implements the "Q-level zips" on $\mathbb{E}(L, Q, P) = e^{L+Q} P(\epsilon)$. Such zips regard the L variables as scalars.

$$\left\langle P(z_i,\zeta^j)e^{c+\eta^i z_i+y_j\zeta^j+q^i_j z_i\zeta^j}\right\rangle = |\tilde{q}|\left\langle P(z_i,\zeta^j)e^{c+\eta^i z_i}\Big|_{z_i\to\tilde{q}^k_i(z_k+y_k)}\right\rangle$$
$$= |\tilde{q}|e^{c+\eta^i\tilde{q}^k_i y_k}\left\langle P\left(\tilde{q}^k_i(z_k+y_k),\zeta^j+\eta^i\tilde{q}^j_i\right)\right\rangle.$$

In[ ]:=
```
QZipξs_List@𝔼[L_, Q_, P_] := Module[{ξ, z, zs, c, ys, ηs, qt, zrule, ξrule, out},
   zs = Table[ξ*, {ξ, ξs}];
   c = CF[Q /. Alternatives @@ (ξs ⋃ zs) → 0];
   ys = CF@Table[∂ξ (Q /. Alternatives @@ zs → 0), {ξ, ξs}];
   ηs = CF@Table[∂z (Q /. Alternatives @@ ξs → 0), {z, zs}];
   qt = CF@Inverse@Table[Kδz,ξ* - ∂z,ξ Q, {ξ, ξs}, {z, zs}];
   zrule = Thread[zs → CF[qt.(zs + ys)]];
   ξrule = Thread[ξs → ξs + ηs.qt];
   CF /@ 𝔼[L, c + ηs.qt.ys, Det[qt] Zipξs[P /. (zrule ⋃ ξrule)]] ];
```

LZip implements the "L-level zips" on $\mathbb{E}(L, Q, P) = Pe^{L+Q}$. Such zips regard all of Pe^{Q} as a single"P". Here the z's are b and α and the ζ's are β and a.

In[◦]:=
```
LZip_{ζs_List}@𝔼[L_, Q_, P_] :=
  Module[{ζ, z, zs, Zs, c, ys, ηs, lt, zrule, Zrule, ζrule, Q1, EEQ, EQ},
   zs = Table[ζ*, {ζ, ζs}];
   Zs = zs /. {b → B, t → T, α → 𝒜};
   c = L /. Alternatives @@ (ζs ⋃ zs) → 0 /. Alternatives @@ Zs → 1;
   ys = Table[∂_ζ (L /. Alternatives @@ zs → 0), {ζ, ζs}];
   ηs = Table[∂_z (L /. Alternatives @@ ζs → 0), {z, zs}];
   lt = Inverse@Table[Kδ_{z,ζ*} - ∂_{z,ζ} L, {ζ, ζs}, {z, zs}];
   zrule = Thread[zs → lt.(zs + ys)];
   Zrule = Join[zrule,
     zrule /. r_Rule :> ((U = r[[1]] /. {b → B, t → T, α → 𝒜}) → (U /. U2l /. r //. l2U))];
   ζrule = Thread[ζs → ζs + ηs.lt];
   Q1 = Q /. (Zrule ⋃ ζrule);
   EEQ[ps___] := EEQ[ps] =
     (CF[e^-Q1 D_Thread[{zs,{ps}}][e^Q1]] /. {Alternatives @@ zs → 0, Alternatives @@ Zs → 1});
   CF@𝔼[c + ηs.lt.ys, Q1 /. {Alternatives @@ zs → 0, Alternatives @@ Zs → 1},
     Det[lt] (Zip_ζs[(EQ @@ zs) (P /. (Zrule ⋃ ζrule))] /.
         Derivative[ps___][EQ][___] :> EEQ[ps] /. _EQ → 1) ]];
```

In[◦]:=
```
B_{}[L_, R_] := L R;
B_{is__}[L_𝔼, R_𝔼] := Module[{n},
   Times[
     L /. Table[(v : b | B | t | T | a | x | y)_i → v_{n@i}, {i, {is}}],
     R /. Table[(v : β | τ | α | 𝒜 | ξ | η)_i → v_{n@i}, {i, {is}}]
   ] // LZip_{Join@@Table[{β_{n@i}, τ_{n@i}, a_{n@i}}, {i, {is}}]} // QZip_{Join@@Table[{ξ_{n@i}, y_{n@i}}, {i, {is}}]} ];
B_{is___}[L_, R_] := B_{{is}}[L, R];
```

𝔼 morphisms with domain and range.

In[◦]:=
```
B_{is_List}[𝔼_{d1_→r1_}[L1_, Q1_, P1_], 𝔼_{d2_→r2_}[L2_, Q2_, P2_]] :=
  𝔼_{(d1⋃Complement[d2,is])→(r2⋃Complement[r1,is])} @@ B_is[𝔼[L1, Q1, P1], 𝔼[L2, Q2, P2]];
𝔼_{d1_→r1_}[L1_, Q1_, P1_] // 𝔼_{d2_→r2_}[L2_, Q2_, P2_] :=
  B_{r1⋂d2}[𝔼_{d1→r1}[L1, Q1, P1], 𝔼_{d2→r2}[L2, Q2, P2]];
𝔼_{d1_→r1_}[L1_, Q1_, P1_] ≡ 𝔼_{d2_→r2_}[L2_, Q2_, P2_] ^:=
  (d1 == d2) ∧ (r1 == r2) ∧ (𝔼[L1, Q1, P1] ≡ 𝔼[L2, Q2, P2]);
𝔼_{d1_→r1_}[L1_, Q1_, P1_] 𝔼_{d2_→r2_}[L2_, Q2_, P2_] ^:=
  𝔼_{(d1⋃d2)→(r1⋃r2)} @@ (𝔼[L1, Q1, P1] 𝔼[L2, Q2, P2]);
𝔼_{dr_}[L_, Q_, P_]_{$k_} := 𝔼_dr @@ 𝔼[L, Q, P]_{$k};
𝔼_[ℰ___][i_] := {ℰ}[[i]];
```

“Define” Code

Define[lhs = rhs, ...] defines the lhs to be rhs, except that rhs is computed only once for each value of $k. Fancy Mathematica not for the faint of heart. Most readers should ignore.

In[∘]:=
```
SetAttributes[Define, HoldAll];
Define[def_, defs__] := (Define[def]; Define[defs];);
Define[op_is__ = ℰ_] := Module[{SD, ii, jj, kk, isp, nis, nisp, sis}, Block[{i, j, k},
   ReleaseHold[Hold[
      SD[op_{nisp,$k_Integer}, Block[{i, j, k}, op_{isp,$k} = ℰ; op_{nis,$k}]];
      SD[op_isp, op_{{is},$k}]; SD[op_sis__, op_{sis}];
     ] /. {SD → SetDelayed,
      isp → {is} /. {i → i_, j → j_, k → k_},
      nis → {is} /. {i → ii, j → jj, k → kk},
      nisp → {is} /. {i → ii_, j → jj_, k → kk_}
     }] ]]
```

```
Define[m_{i,j→k} = 𝔼_{{i,j}→{k}}[0, -ξ_i η_j + (η_i + η_j) y_k + (ξ_i + ξ_j) x_k, 1]]
(*Heisenberg multiplication*)
```

A very docile R-matrix to order 3 in ϵ. Is there an R-matrix such that the coefficient of ϵ^k has degree 2k in y,x?

```
X_{i_,j_} := 𝔼_{{}→{i,j}}[0, (y_i - y_j) x_j, 1 + ϵ (x_i y_i - x_j y_i - x_j y_j) +
    ϵ^2 (Sum[r_{If[#===i,0,1]&@k,If[#===i,0,1]&@l} y_k x_l, {k, {i, j}}, {l, {i, j}}] +
       Sum[s_{If[#===i,0,1]&@k,If[#===i,0,1]&@l,If[#===i,0,1]&@m,If[#===i,0,1]&@n} y_k x_l y_m x_n,
        {k, {i, j}}, {l, {i, j}}, {m, {i, j}}, {n, {i, j}}]) + O[ϵ]^3] /.
  {r_{1,0} → 0, r_{1,1} → 1, r_{0,0} → -1, r_{0,1} → 0, s_{0,0,0,0} → 1/2, s_{1,0,1,0} → 0, s_{1,0,0,0} → 1,
   s_{0,0,1,0} → -1, s_{1,1,1,1} → 1/2, s_{1,1,1,0} → 1, s_{1,0,1,1} → -1,
   s_{0,0,1,1} → 1, s_{0,1,0,1} → 1, s_{0,1,1,0} → -1, s_{1,0,0,1} → 1, s_{1,1,0,0} → -1,
   s_{0,0,0,1} → -1, s_{0,1,0,0} → -1, s_{0,1,1,1} → 1, s_{1,1,0,1} → -1/2}
(*It looks pretty enough but some random choices were made already.*)
X_{1,2} /. {r_{1,0} → 0, r_{1,1} → 1, r_{0,0} → -1, r_{0,1} → 0, s_{0,0,0,0} → 1/2, s_{1,0,1,0} → 0, s_{1,0,0,0} → 1,
  s_{0,0,1,0} → -1, s_{1,1,1,1} → 1/2, s_{1,1,1,0} → 1, s_{1,0,1,1} → -1, s_{0,0,1,1} → 1, s_{0,1,0,1} → 1, s_{0,1,1,0} → -1,
  s_{1,0,0,1} → 1, s_{1,1,0,0} → -1, s_{0,0,0,1} → -1, s_{0,1,0,0} → -1, s_{0,1,1,1} → 1, s_{1,1,0,1} → -1/2}
```

Out[∘]= $\mathbb{E}_{\{\}\to\{1,2\}}\left[0,\ x_2\,(y_1 - y_2),\ 1 + (x_1 y_1 - x_2 y_1 - x_2 y_2)\,\epsilon + \left(-x_1 y_1 + \frac{1}{2} x_1^2 y_1^2 - 2 x_1 x_2 y_1^2 + x_2^2 y_1^2 + x_2 y_2 + \frac{1}{2} x_2^2 y_1 y_2 + \frac{1}{2} x_2^2 y_2^2\right)\epsilon^2 + O[\epsilon]^3\right]$

```
(*Checking Yang-Baxter:*)
(X_{1,2} X_{4,3} X_{5,6} // m_{1,4→1} // m_{2,5→2} // m_{3,6→3}) ≡ (X_{2,3} X_{4,5} X_{1,6} // m_{1,4→1} // m_{2,5→2} // m_{3,6→3})
```

Out[∘]= True