Pensieve header: A fresh implementation of baby DoPeGDO. Continues pensieve://2020-09/, pensieve://2020-03/Testing123.nb, and pensieve://People/VanDerVeen/TimidHeisenbergRGeneralForm@.nb.

𝔼[ω, Q, P_ϵSeries] represents ωe^{Q+P}, where ω is a scalar, Q is an ϵ-free quadratic, and $P = \sum_{k=0}^{\$k} P[\![k]\!]\,\epsilon^k$ is a perturbation (it is ill-advised to include ω in P because then it will have log terms).

Scheme: 𝔼_[_] // 𝔼_[_] calls FZip or Zip, which are functionally the same. Zip works by handling the quadratic part and calling PZip for the perturbation-only part. PZip works by iteratively solving the synthesis equation. FZip works by encapsulating coefficients, calling Zip, and back-substituting.

Initialization, minor utilities, and “Define” Code

```
SetDirectory["C:\\drorbn\\AcademicPensieve\\Projects\\BabyDoPeGDO"];
Once[<< KnotTheory`];
Once[Get@"../Profile/Profile.m"];
```

```
Loading KnotTheory` version of February 2, 2020, 10:53:45.2097.
Read more at http://katlas.org/wiki/KnotTheory.
This is Profile.m of http://www.drorbn.net/AcademicPensieve/Projects/Profile/.
This version: April 2020. Original version: July 1994.
```

In[∘]:=
```
$k=1;
```

In[∘]:=
```
CCF[ℰ_] := ExpandDenominator@ExpandNumerator@Together[ℰ];
CF[ℰ_List] := CF /@ ℰ;
CF[ℰ_ϵSeries] := CF /@ ℰ;
CF[ℰ_] := PP_CF@Module[
    {vs = Cases[ℰ, (y | x | η | ξ)_, ∞] ⋃ {y | x | η | ξ}},
    Total[CoefficientRules[Expand[ℰ], vs] /. (ps_ → c_) :→ CCF[c] (Times @@ vs^ps)]
   ];
(*CF[ℰ_] := PP_CF@CCF[ℰ];*)
CF[ℰ_𝔼] := CF /@ ℰ;
CF[𝔼_sp___[ℰs___]] := CF /@ 𝔼_sp[ℰs];
```

In[∘]:=
```
ϵSeries /: S1_ϵSeries ≡ S2_ϵSeries :=
  Length[S1] == Length[S2] ∧ Inner[CF[#1] == CF[#2] &, S1, S2, And];
ϵSeries[0] := ϵSeries @@ Table[0, $k + 1];
ϵSeries /: S1_ϵSeries + S2_ϵSeries :=
  ϵSeries @@ Table[S1[[k]] + S2[[k]], {k, Min[Length@S1, Length@S2]}];
ϵSeries /: S1_ϵSeries * S2_ϵSeries := ϵSeries @@
   Table[Sum[S1[[j + 1]] * S2[[k - j + 1]], {j, 0, k}], {k, 0, Min[Length@S1, Length@S2] - 1}];
ϵSeries /: c_ * S_ϵSeries := (c #) & /@ S;
ϵSeries /: ∂_vs___ S_ϵSeries := (s ↦ ∂_vs s) /@ S;
```

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_}
     }] ]]
```

The Basic Tensors

In[∘]:=
```
Define[m_{i,j→k} = 𝔼_{{i,j}→{k}}[1, -ξ_i η_j + (η_i + η_j) y_k + (ξ_i + ξ_j) x_k, ϵSeries[0]]]
```

In[∘]:=
```
Define[
 R_{i,j} = 𝔼_{{}→{i,j}}[1, (-1 + T) x_j (y_i - y_j),
   ϵSeries[0, -1/2 (1 - T) x_j^2 y_i^2 + x_i x_j y_i y_j + 1/2 (1 - 3 T) x_j^2 y_i y_j]],
 R̄_{i,j} = 𝔼_{{}→{i,j}}[1, (-1 + 1/T) x_j (y_i - y_j),
   ϵSeries[0, -((-1 + T) x_i x_j y_i^2)/T^2 - ((1 - T) x_j^2 y_i^2)/(2 T^3) - (x_i x_j y_i y_j)/T^2 - ((-1 - T) x_j^2 y_i y_j)/(2 T^3)]],
 CC_i = 𝔼_{{}→{i}}[√T, 0, ϵSeries[0, -(x_i y_i)/T]],
 C̄C̄_i = 𝔼_{{}→{i}}[1/√T, 0, ϵSeries[0, (x_i y_i)/T]]
]
```

In[∘]:=
```
Define[Kink_i = CC_3 R_{1,2} // m_{2,3→2} // m_{2,1→i}, K̄ink_i = CC_3 R̄_{1,2} // m_{1,3→1} // m_{1,2→i}]
```

The Main Program

Variables and their duals:

In[∘]:=
```
{y*, x*, η*, ξ*} = {η, ξ, y, x};
(vs_List)* := (v ↦ v*) /@ vs;
(u_i_)* := (u*)_i;
```

𝔼 operations:

```
In[∘]:=
𝔼 /: 𝔼[ω1_, Q1_, P1_] ≡ 𝔼[ω2_, Q2_, P2_] := CF[ω1 == ω2] ∧ CF[Q1 == Q2] ∧ (P1 ≡ P2);
𝔼 /: 𝔼[ω1_, Q1_, P1_] × 𝔼[ω2_, Q2_, P2_] := 𝔼[ω1 ω2, Q1 + Q2, P1 + P2];
𝔼_{d1_→r1_}[ℰ1s___] ≡ 𝔼_{d2_→r2_}[ℰ2s___] ^:= (d1 == d2) ∧ (r1 == r2) ∧ (𝔼[ℰ1s] ≡ 𝔼[ℰ2s]);
𝔼_{d1_→r1_}[ℰ1s___] 𝔼_{d2_→r2_}[ℰ2s___] ^:= 𝔼_{(d1⋃d2)→(r1⋃r2)} @@ (𝔼[ℰ1s] × 𝔼[ℰ2s]);
𝔼_{dr_}[ℰs___]_{$k_} := 𝔼_{dr} @@ 𝔼[ℰs]_{$k};
```

```
In[∘]:=
𝔼_{d1_→r1_}[ℰ1s___] // 𝔼_{d2_→r2_}[ℰ2s___] := Module[{is = r1 ⋂ d2, lvs},
  lvs = Flatten@Table[{x_{$@i}, y_{$@i}}, {i, is}];
  𝔼_{(d1⋃Complement[d2,is])→(r2⋃Complement[r1,is])} @@ (Zip_{lvs⋃lvs*}[lvs*.lvs, Times[
      𝔼[ℰ1s] /. Table[(v : x | y)_i → v_{$@i}, {i, is}],
      𝔼[ℰ2s] /. Table[(v : ξ | η)_i → v_{$@i}, {i, is}]
     ]])
 ]
```

$$[F:\mathcal{E}]_B := e^{\frac{1}{2}\sum_{i,j\in B} F_{ij}\partial_{z_i}\partial_{z_j}}\mathcal{E} \quad \text{and} \quad \langle F:\mathcal{E}\rangle_B := [F:\mathcal{E}]_B|_{z_B\to 0},$$

where $\mathcal{E}$ is a docile perturbed Gaussian. The following lemma allows us to restrict to the case where $\mathcal{E}$ has no B-B quadratic part:

Lemma 1. With convergences left to the reader,

$$\left\langle F:\mathcal{E}e^{\frac{1}{2}\sum_{i,j\in B} G_{ij}z_iz_j}\right\rangle_B = \det(1-GF)^{-1/2}\left\langle F(1-GF)^{-1}:\mathcal{E}\right\rangle_B.$$

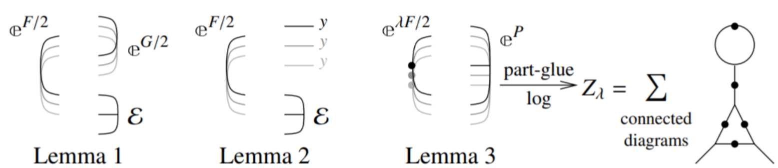
The next lemma dispatches the case where $\mathcal{E}$ has a B-linear part:

Lemma 2. $\left\langle F:\mathcal{E}e^{\sum_{i\in B} y_iz_i}\right\rangle_B = e^{\frac{1}{2}\sum_{i,j\in B} F_{ij}y_iy_j}\left\langle F:\ \mathcal{E}|_{z_B\to z_B+Fy_B}\right\rangle_B$.

Finally, we deal with the docile perturbation case:

Lemma 3. With an extra variable λ, $Z_\lambda := \log[\lambda F: e^P]_B$ satisfies and is determined by the following PDE / IVP:

$$Z_0 = P \quad \text{and} \quad \partial_\lambda Z_\lambda = \frac{1}{2}\sum_{i,j\in B} F_{ij}\left(\partial_{z_i}\partial_{z_j}Z_\lambda + (\partial_{z_i}Z_\lambda)(\partial_{z_j}Z_\lambda)\right).$$




```
In[∘]:=
Zip_{vs_}[ℱ_, ℰ_] := ⟨ℱ, ℰ⟩ // Zip1_{vs} // Zip2_{vs} // Zip3_{vs}
```

Getting rid of the quadratic.

Lemma 1. With convergences left to the reader,

$$\left\langle F:\mathcal{E}e^{\frac{1}{2}\sum_{i,j\in B} G_{ij}z_iz_j}\right\rangle_B = \det(1-GF)^{-1/2}\left\langle F(1-GF)^{-1}:\mathcal{E}\right\rangle_B$$

In[∘]:=
```
Zip1vs_@⟨ℱ_, 𝔼[ω_, Q_, P_]⟩ := PPZip1@Module[{ℐ, F, G, u, v},
  ℐ = IdentityMatrix@Length@vs;
  F = Table[∂u,v ℱ, {u, vs*}, {v, vs*}];
  G = Table[∂u,v Q, {u, vs}, {v, vs}];
  CF /@ ⟨vs*.F.Inverse[ℐ - G.F].vs* / 2,
    𝔼[PowerExpand@Factor[ω Det[ℐ - G.F]^-1/2], Q - vs.G.vs / 2, P]⟩
]
```

Getting rid of linear terms.

Lemma 2. $\left\langle F:\mathcal{E}\,e^{\sum_{i\in B}y_iz_i}\right\rangle_B=e^{\frac{1}{2}\sum_{i,j\in B}F_{ij}y_iy_j}\left\langle F:\ \mathcal{E}|_{z_B\to z_B+Fy_B}\right\rangle_B.$

In[∘]:=
```
Zip2vs_@⟨ℱ_, 𝔼[ω_, Q_, P_]⟩ := PPZip2@Module[{F, Y, u, v},
  F = Table[∂u,v ℱ, {u, vs*}, {v, vs*}];
  Y = Table[∂v Q, {v, vs}];
  CF /@ ⟨ℱ, 𝔼[ω, Q - Y.vs + Y.F.Y / 2, P /. Thread[vs → vs + F.Y]]⟩
]
```

Dealing with Feynman diagrams.

Lemma 3. With an extra variable λ, $Z_\lambda:=\log[\lambda F:e^P]_B$ satisfies and is determined by the following PDE / IVP:

$$Z_0=P\quad\text{and}\quad\partial_\lambda Z_\lambda=\frac{1}{2}\sum_{i,j\in B}F_{ij}\left(\partial_{z_i}\partial_{z_j}Z_\lambda+(\partial_{z_i}Z_\lambda)(\partial_{z_j}Z_\lambda)\right).$$

Note that the power m of λ is at most $k-1+\frac{2k+2}{2}=2k$. We write $Z_\lambda=\sum Z[m]\,\lambda^m$.

In[∘]:=
```
Zip3vs_@⟨ℱ_, 𝔼[ω_, Q_, P_]⟩ := PPZip3@Module[{u, v, m, j},
  Z[0] = P;
  For[m = 0, m < 2 $k, ++m,
   Z[m + 1] = CF[1/(2 (m + 1))
      Sum[∂u*,v* ℱ (∂u,v Z[m] + Sum[(∂u Z[j]) (∂v Z[m - j]), {j, 0, m}]), {u, vs}, {v, vs}]]
  ];
  𝔼[ω, Q, CF[Sum[Z[m], {m, 0, 2 $k}] /. Table[v → 0, {v, vs}]]]
]
```

Some Knot Theory

```
In[∘]:= RVK[pd_PD] := Module[{n, xs, x, rots, front = {0}, k},
    n = Length@pd; rots = Table[0, {2 n}];
    xs = Cases[pd, x_X ⧴ { Xp[x⟦4⟧, x⟦1⟧]  PositiveQ@x
                           Xm[x⟦2⟧, x⟦1⟧]  True        ];
    For[k = 0, k < 2 n, ++k, If[k == 0 ∨ FreeQ[front, -k],
      front = Flatten[front /. k → (xs /. {
              Xp[k + 1, l_] | Xm[l_, k + 1] ⧴ {l, k + 1, 1 - l},
              Xp[l_, k + 1] | Xm[k + 1, l_] ⧴ (++rots⟦l⟧; {1 - l, k + 1, l})
             })],
      Cases[front, k | -k] /. {k, -k} ⧴ --rots⟦k + 1⟧;
     ]];
    RVK[xs, rots] ];
RVK[K_] := RVK[PD[K]];
```

```
In[∘]:= rot[i_, 0] := 𝔼_{{}→{i}}[1, 0, ϵSeries@0];
rot[i_, n_] := Module[{j},
    rot[i, n] = If[n > 0, rot[i, n - 1] CC_j, rot[i, n + 1] \overline{CC}_j] // m_{i,j→i}];
```

In[∘]:=
```
Z[K_] := Z[RVK@K];
Z[rvk_RVK] := (*Z[rvk] =*)
 Module[{todo, n, rots, ζ, done, st, cx, ζ1, i, j, k, k1, k2, k3},
  {todo, rots} = List @@ rvk;
  AppendTo[rots, 0];
  n = Length[todo];
  ζ = 𝔼_{{}→{0}}[1, 0, ϵSeries@0];
  done = {0};
  st = Range[0, 2 n + 1];
  While[{} =!= ($M = todo),
   {cx} = MaximalBy[todo, Length[done ⋂ {#⟦1⟧, #⟦2⟧, #⟦1⟧ - 1, #⟦2⟧ - 1}] &, 1];
   {i, j} = List @@ cx;
   ζ1 = Switch[Head[cx],
      Xp, (R_{i,j} OverBar[Kink_k]) // m_{j,k→j},
      Xm, (OverBar[R]_{i,j} Kink_k) // m_{j,k→j}
     ];
   ζ1 = (rot[k, rots⟦i⟧] ζ1) // m_{k,i→i}; rots⟦i⟧ = 0;
   ζ1 = (ζ1 rot[k, rots⟦i + 1⟧]) // m_{i,k→i}; rots⟦i + 1⟧ = 0;
   ζ1 = (rot[k, rots⟦j⟧] ζ1) // m_{k,j→j}; rots⟦j⟧ = 0;
   ζ1 = (ζ1 rot[k, rots⟦j + 1⟧]) // m_{j,k→j}; rots⟦j + 1⟧ = 0;
   ζ *= ζ1;
   If[MemberQ[done, i], ζ = ζ // m_{i,i+1→i}; st = st /. st⟦i + 2⟧ → st⟦i + 1⟧];
   If[MemberQ[done, i - 1], ζ = ζ // m_{st⟦i⟧,i→st⟦i⟧}; st = st /. st⟦i + 1⟧ → st⟦i⟧];
   If[MemberQ[done, j], ζ = ζ // m_{j,j+1→j}; st = st /. st⟦j + 2⟧ → st⟦j + 1⟧];
   If[MemberQ[done, j - 1], ζ = ζ // m_{st⟦j⟧,j→st⟦j⟧}; st = st /. st⟦j + 1⟧ → st⟦j⟧];
   done = done ⋃ {i - 1, i, j - 1, j};
   todo = DeleteCases[todo, cx]
  ];
  CF /@ (ζ (*/. {x_0→x,y_0→y,a_0→a}*))
 ]
```

In[∘]:=
```
BeginProfile[];
PopupWindow[Button["Show Profile Monitor"],
 Dynamic[PrintProfile[], UpdateInterval → 3, TrackedSymbols → {}]]
```

Out[∘]= Show Profile Monitor

In[∘]:=
```
NewBit[K_] := Module[{Alex = Alexander[K][T]},
  T^3 Alex^2/(T - 1) Z[K]⟦3, 2⟧ // Factor]
```

In[∘]:=
```
NewBit /@ AllKnots[{3, 5}]
```

KnotTheory: Loading precomputed data in PD4Knots`.

Out[∘]= $$\left\{2-T+T^2,\ (1+T)\left(1-3T+T^2\right),\ \frac{4-3T+5T^2-3T^3+3T^4-T^5+T^6}{T^2},\ 9-11T+7T^2-T^3\right\}$$

```
In[◦]:= (*Two knots with equal Alexander, new bit does not agree*)
Alexander[Knot[6, 1]] == Alexander[Knot[9, 46]]
Timing[NewBit[Knot[6, 1]] == NewBit[Knot[9, 46]]]

Out[◦]= True

Out[◦]= {46.4531, 5 - 11 T - T^2 + 3 T^3 == 7 - 21 T + 9 T^2 + T^3}

In[◦]:= PrintProfile[]

Out[◦]= ProfileRoot is root. Profiled time: 79.031
  (  24)   0.032/  0.032 above CF
  ( 237)   1.581/  6.183 above Zip1
  ( 237)   0.799/ 38.897 above Zip2
  ( 237)  28.773/ 33.919 above Zip3
CF: called 3816 times, time in 47.878/47.878
  (  24)   0.032/  0.032 under ProfileRoot
  ( 1185)   4.602/  4.602 under Zip1
  ( 1185)  38.098/ 38.098 under Zip2
  ( 1422)   5.146/  5.146 under Zip3
Zip3: called 237 times, time in 28.773/33.919
  ( 237)  28.773/ 33.919 under ProfileRoot
  ( 1422)   5.146/  5.146 above CF
Zip1: called 237 times, time in 1.581/6.183
  ( 237)   1.581/  6.183 under ProfileRoot
  ( 1185)   4.602/  4.602 above CF
Zip2: called 237 times, time in 0.799/38.897
  ( 237)   0.799/ 38.897 under ProfileRoot
  ( 1185)  38.098/ 38.098 above CF
```