build

Hamiltonian Gauge Gravity Surveyor (HiGGS)

build

Source notebook for the HiGGS session and binary files

build

Initialisation

build

Notebook build functions

build

```
(*List of all print cells in front end before this notebook starts torun*)
$PrintCellsBeforeBuildHiGGS =
  Flatten@Cells[SelectedNotebook[], CellStyle → {"Print"}];
(*Purge all print cells produced since notebook starts to run*)
ClearBuild[] :=
  NotebookDelete@(Flatten@Cells[SelectedNotebook[], CellStyle → {"Print"}] ~
     Complement ~ $PrintCellsBeforeBuildHiGGS);
(*ClearBuild[]:=Print@"clearb"*)
(*This setup works for CellTags*)
DirectoryEnvironment[] := Module[{RelevantTag, CurrentCell},
   CurrentCell = EvaluationCell[];
   RelevantTag = CurrentValue[CurrentCell, CellTags];
   Print["DiectoryEnvironment[] was called and will return CellTag ",
    RelevantTag, "."];
   FileNameJoin@{$HiGGSInstallDirectory, "bin/build/" <> RelevantTag <> ".mx"}];
PreviousDirectoryEnvironment[] := Module[{RelevantTag, LastCell},
   LastCell = PreviousCell[];
   RelevantTag = CurrentValue[LastCell, CellTags];
   Print["PreviousDiectoryEnvironment[] was called and will return CellTag ",
    RelevantTag, "."];
   FileNameJoin@{$HiGGSInstallDirectory, "bin/build/" <> RelevantTag <> ".mx"}];
PreviousCellTag[] := Module[{RelevantTag, LastCell},
   LastCell = PreviousCell[];
   RelevantTag = CurrentValue[LastCell, CellTags];
```

```
    RelevantTag];
(*Switch this off to prevent loading through CellTags*)
(*
OpenLastCache[]:=If[PrematureCellTags~MemberQ~PreviousCellTag[],
    Print[
      "The binary at "<>PreviousDirectoryEnvironment[]<>" has been ignored."];,
    Check[ToExpression["<<"<>PreviousDirectoryEnvironment[]<>";"],
      Print["The binary at "<>
        PreviousDirectoryEnvironment[]<>" cannot be found: quitting."];
      Quit[];];
  ];
*)

(*This construction supercedes the use of CellTags*)
BinaryLocation[RelevantTag_String] :=
  FileNameJoin@{$HiGGSInstallDirectory, "bin/build/" <> RelevantTag <> ".mx"};
BuildHiGGS::nobin = "The binary at `1` cannot be found; quitting.";
SetAttributes[IfBuild, HoldAll];
IfBuild[RelevantTag_String, expr_] :=
  Catch@If[PrematureCellTags~MemberQ~RelevantTag,
    Print[" ** BuildHiGGS: The binary at " <>
       BinaryLocation@RelevantTag <> " has been ignored."];,
    If[ActiveCellTags~MemberQ~RelevantTag,
      Print[" ** BuildHiGGS: Building the binary at " <>
        BinaryLocation@RelevantTag <> "..."];
      $BinaryLocation = BinaryLocation@RelevantTag;
      Evaluate@expr;,
      If[UnitTests~MemberQ~RelevantTag,
        Print[" ** BuildHiGGS: The unit test labelled " <>
           RelevantTag <> " has been ignored."];,
        Print[" ** BuildHiGGS: Incorporating the binary at " <>
          BinaryLocation@RelevantTag <> "..."];
        Check[ToExpression["<<" <> BinaryLocation@RelevantTag <> ";"],
         Throw@Message[BuildHiGGS::nobin, BinaryLocation@RelevantTag];
         Quit[];];];
    ];
  ];

(*Incorporate borrowed scripts from Cyril Pitrou's contributions*)
Get["xAct/HiGGS/HiGGS_variations.m"];
ClearBuild[];
```

build

Manifold and geometry

build

```
dimension = 4;                (* dimension of space-time manifold *)
DefManifold[M4, dimension, IndexRange[{a, z}]];
AddIndices[TangentM4, {a1, b1, c1, d1, e1, f1, g1, h1, i1,
   j1, k1, l1, n1, m1, o1, p1, q1, r1, s1, t1, u1, v1, w1, x1, y1, z1}];
Quiet@DefMetric[-1, G[-a, -c], CD, {",", "∂"}, PrintAs → "ɣ",
   FlatMetric → True, SymCovDQ → True];
(*DefMetric[-1,G[-a,-c],CD,{",","∂"},PrintAs→"ɣ",
  SymCovDQ→True,FlatMetric→True];*)
ClearBuild[];
```

build

Administrative functions

build

```
If[$PaperPrint,
  If[NotebookDirectory[] == $Failed,
    Print[" ** BuildHiGGS: Purging figures directory at " <>
       FileNameJoin@{NotebookDirectory[], "fig/*"} <> "..."];,
    Run@("rm -rf " <> FileNameJoin@{NotebookDirectory[], "fig/*"});,
    Run@("rm -rf " <> FileNameJoin@{NotebookDirectory[], "fig/*"});];
 ];

$OldLine = $Line;
$SubLine = 1;
(*$PaperPrint=False;*)

HiGGSOutput[x_String] := Module[{},
   $ListingsOutput = x;
   Run@("rm " <> FileNameJoin@{$WorkingDirectory, "fig", $ListingsOutput});
  ];

SetAttributes[HiGGSEcho, HoldAll]
HiGGSEcho[x_] := Block[{str, res, $ListingsFile},
   str = ToString[Unevaluated[x] ~ToString~ InputForm];
   $ListingsFile = OpenAppend[FileNameJoin@
      {$WorkingDirectory, "fig", $ListingsOutput}, PageWidth -> Infinity];
   WriteString[$ListingsFile, "In[]:= " <> str <> "\nOut[]= "];
   (*WriteString[$ListingsFile,"|\nIn[]:= "<>str<>"\n|\n"];*)
   Close@$ListingsFile;
```

```
    res = Evaluate@x;
    res];

$Widetext = False;

Options[HiGGSPrint] = {"Widetext" -> False};
HiGGSPrint[expr__, OptionsPattern[]] := Block[{res, $ListingsFile, size},
    If[$Widetext, size = (510 / 246) * 350, size = 350];
    (*If[OptionValue@"Widetext",size=(510/246)*300,size=300];*)
    res = expr;
    Print@res;
    If[$PaperPrint,
     If[$Line == $OldLine,
      $SubLine = $SubLine + 1,
      $SubLine = 1;
      $OldLine = $Line;
     ];
     $ListingsFile = OpenAppend[FileNameJoin@
        {$WorkingDirectory, "fig", $ListingsOutput}, PageWidth -> Infinity];
     If[{res}~AllTrue~StringQ,
      WriteString[$ListingsFile,
        "|\n\\vspace{-10pt}\n|\n" <> "" <> StringJoin@{res} <> "\n"];,
      (*WriteString[$ListingsFile,"\\vspace{-10pt}\n|\n"<>
         "      "<>StringJoin@{res}<>"\n|"<>"\n"];,*)
      res = Panel[Row@{"", res}, ImageSize → size, Background -> RGBColor[0.95, 1.,
          0.8], FrameMargins → None, ContentPadding -> True, Alignment -> Right];
      Print@res;
      (*order below was 4-7 now 5-5*)
      FileNameJoin@{$WorkingDirectory, "fig", $ListingsOutput <>
          ToString@$OldLine <> "-" <> ToString@$SubLine <> "fig.pdf"}~Export~res;
      WriteString[$ListingsFile,
       "|\n\\vspace{-4pt}\n\\begin{flushleft}\n\\includegraphics[width=\\
          linewidth]{figures/" <>
        $ListingsOutput <> ToString@$OldLine <> "-" <> ToString@$SubLine <>
        "fig.pdf}\n\\end{flushleft}\n\\vspace{-5pt}\n|\n"];
      (*WriteString[$ListingsFile,
        "\\vspace{-10pt}\n\\begin{flushleft}\n\\includegraphics[width=\\
           linewidth]{figures/"<>ToString@
          $OldLine<>"-"<>ToString@$SubLine<>"fig.pdf}\n\\end{flushleft}\n"];*)
     ];
     Close@$ListingsFile;
    ];
   ];
ClearBuild[];
```

build

```
(*Probably a better place to put this at the top*)
ToNewCanonical[x_] := "ToNewCanonical"~TimeWrapper~
   Module[{temp, res, time, duration, filename, printer},
    printer = PrintTemporary[" ** ToNewCanonical..."];
    (*Beep[];*)
    temp = x;
    temp = ToCanonical@temp;
    temp = temp // ContractMetric;
    temp = temp // ScreenDollarIndices;
    NotebookDelete[printer];
    temp];

(*To suppress the error message from VarD when
 CyrilPitrou's VarAction runs on indexed tensors*)
NewVarAction[x_, y_] :=
  "VarAction"~TimeWrapper~Quiet[VarAction[x, y], {VarD::nouse}];
ClearBuild[];
```

Perturbation theory, which does not use xPert (but probably should)

build

```
(*This constant symbol will parametrise the perturbation*)
DefConstantSymbol[Prt, PrintAs → "ϵ"];
$ToNormalOrderRules = {};
$ToEHOrderRules = {};
Options[DeclareOrder] = {"IsUnityWithEHTerm" → False, "approximation" → False};
DeclareOrder[tensor_, order_, OptionsPattern[]] :=
  "DeclareOrder"~TimeWrapper~Module[{tmp},
    If[OptionValue["approximation"] == False,
     tmp = MakeRule[{tensor, Evaluate[Prt^order tensor]},
         MetricOn → All, ContractMetrics → True];,
     tmp = MakeRule[{tensor, Evaluate[Prt^order Evaluate[OptionValue[
              "approximation"]]]}, MetricOn → All, ContractMetrics → True];,
     tmp = MakeRule[{tensor, Evaluate[Prt^order Evaluate[OptionValue[
              "approximation"]]]}, MetricOn → All, ContractMetrics → True];
    ];
    If[OptionValue["IsUnityWithEHTerm"] == False,
     $ToNormalOrderRules = Join[$ToNormalOrderRules, tmp];
     $ToEHOrderRules = Join[$ToEHOrderRules, tmp];,
     $ToNormalOrderRules = Join[$ToNormalOrderRules, tmp];
    ];
   ];
ClearBuild[];
```

Formatting of spin irreps

build

```
(*it is better that coupling constants format in colour*)
$Coupling = RGBColor[1, 0, 0];
Colour[x_String, ColorKey_] := ColorString[x, ColorKey];
(*a more systematic way to format tensors*)
$TensorColour = RGBColor[0, 0, 0];
$IrrepColour = RGBColor[0, 0, 1];
Spin0p = "0̊⁺";
Spin0m = "0̊⁻";
Spin1p = "1̊⁺";
Spin1m = "1̊⁻";
Spin2p = "2̊⁺";
Spin2m = "2̊⁻";
SO0 = "0̩";
SO1 = "1̩";
SO2 = "2̩";
SO3 = "3̩";
SO4 = "4̩";
SO5 = "5̩";
SO6 = "6̩";
dSpin0p = "."0⁺;
dSpin0m = "."0⁻;
dSpin1p = "."1⁺;
dSpin1m = "."1⁻;
dSpin2p = "."2⁺;
dSpin2m = "."2⁻;
dSO0 = "."0;
dSO1 = "."1;
dSO2 = "."2;
dSO3 = "."3;
dSO4 = "."4;
dSO5 = "."5;
dSO6 = "."6;
```

build

```
Options[SymbolBuild] = {"Derivative" -> 0, "Constant" -> False};
SymbolBuild[TensorSymbol_,
   IrrepSymbol : _?StringQ : "", OptionsPattern[]] := Module[{res},
   If[PossibleZeroQ@StringLength@IrrepSymbol,
    res = ColorString[TensorSymbol, $TensorColour];,
    If[OptionValue@"Constant",
      res = ColorString[TensorSymbol, $Coupling] ~
          StringJoin ~ ColorString[IrrepSymbol, $IrrepColour];,
      res = ColorString[IrrepSymbol, $IrrepColour] ~ StringJoin ~
          ColorString[TensorSymbol, $TensorColour];
     ];
   ];
   If[OptionValue@"Derivative" == 1,
    res = "D" ~ StringJoin ~ res;
   ];
   If[OptionValue@"Derivative" == 2,
    res = "(𝒟" ~ StringJoin ~ res;
    res = res ~ StringJoin ~ ")""";
   ];
   res];
ClearBuild[];
```

build

Irreducible decomposition of the fields using $SO^+(1,3)$

build

Initial definitions

build

```
SectorNames =
  {"B0p", "B1p", "B1m", "B2p", "A0p", "A0m", "A1p", "A1m", "A2p", "A2m"};
ASectorNames = {"A0p", "A0m", "A1p", "A1m", "A2p", "A2m"};
BSectorNames = {"B0p", "B0m", "B1p", "B1m", "B2p", "B2m"};
RSymb = "ℛ";
DefTensor[R[a, b, -d, -e], M4, {Antisymmetric[{a, b}],
   Antisymmetric[{-d, -e}]}, PrintAs -> SymbolBuild[RSymb]];
DeclareOrder[R[a, b, -d, -e], 1];
TSymb = "𝒯";
DefTensor[T[a, -b, -c], M4,
  Antisymmetric[{-b, -c}], PrintAs -> SymbolBuild[TSymb]];
DeclareOrder[T[a, -b, -c], 1];
WSymb = "𝒲";
DefTensor[W[a, b, -d, -e], M4, PrintAs -> SymbolBuild[WSymb]];
DeclareOrder[W[a, b, -d, -e], 1];
RLambdaSymb = "λ_ℛ"
DefTensor[RLambda[a, b, -d, -e], M4,
  {Antisymmetric[{a, b}], Antisymmetric[{-d, -e}]},
  PrintAs -> SymbolBuild[RLambdaSymb]];
DeclareOrder[RLambda[a, b, -d, -e], 1];
TLambdaSymb = "λ_𝒯";
DefTensor[TLambda[a, -d, -e], M4,
  Antisymmetric[{-d, -e}], PrintAs -> SymbolBuild[TLambdaSymb]];
DeclareOrder[TLambda[a, -d, -e], 1];
ClearBuild[];
```

build

Basic/Nester forms of R and T

build

```
(*
(*This is where we get the notation for generating sets of permutations from,
not the documentation!*)
HiGGSPrint[RiemannSymmetry[{-i,-j,-m,-n}]];
*)
DefTensor[R1[-i, -j, -m, -n], M4, StrongGenSet[{-i, -j, -m, -n},
   GenSet[Cycles[{-i, -j}, {-m, -n}], Cycles[{-i, -m}], Cycles[{-j, -n}]]],
```

```
  PrintAs → SymbolBuild[RSymb, SO1]];
DeclareOrder[R1[-i, -j, -m, -n], 1];
DefTensor[R2[-i, -j, -m, -n], M4, StrongGenSet[{-i, -j, -m, -n},
   GenSet[-Cycles[{-i, -m}, {-j, -n}], -Cycles[{-i, -j}], -Cycles[{-m, -n}]]],
  PrintAs -> SymbolBuild[RSymb, SO2]];
DeclareOrder[R2[-i, -j, -m, -n], 1];
DefTensor[R3[-i, -j, -m, -n], M4,
  Antisymmetric[{-i, -j, -m, -n}], PrintAs -> SymbolBuild[RSymb, SO3]];
DeclareOrder[R3[-i, -j, -m, -n], 1];
DefTensor[R4[-i, -j], M4,
  Symmetric[{-i, -j}], PrintAs -> SymbolBuild[RSymb, SO4]];
DeclareOrder[R4[-i, -j], 1];
DefTensor[R5[-i, -j], M4,
  Antisymmetric[{-i, -j}], PrintAs -> SymbolBuild[RSymb, SO5]];
DeclareOrder[R5[-i, -j], 1];
DefTensor[R6[], M4, PrintAs -> SymbolBuild[RSymb, SO6]];
DeclareOrder[R6[], 1];
DefTensor[T1[-i, -j, -k], M4,
  Symmetric[{-i, -j}], PrintAs -> SymbolBuild[TSymb, SO1]];
DeclareOrder[T1[-i, -j, -k], 1];
DefTensor[T2[-i], M4, PrintAs -> SymbolBuild[TSymb, SO2]];
DeclareOrder[T2[-i], 1];
DefTensor[T3[-i], M4, PrintAs -> SymbolBuild[TSymb, SO3]];
DeclareOrder[T3[-i], 1];
AutomaticRules[R1,
  MakeRule[{R1[a, a1, b, -b], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[R1, MakeRule[{R1[a, b, a1, -b], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[R2, MakeRule[{R2[a, b, a1, -b], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[R4, MakeRule[{R4[a, -a], 0}, MetricOn → All,
   ContractMetrics → True]];
AutomaticRules[T1, MakeRule[{T1[a, a1, -a1], 0},
   MetricOn → All, ContractMetrics → True]];

AutomaticRules[T1,
  MakeRule[{T1[a, -a, -k], 0}, MetricOn → All, ContractMetrics → True]];

RDefinition = R3[-i, -j, -m, -n] +
   (2/3) (2 R1[-i, -j, -m, -n] +
      R1[-i, -m, -j, -n]) +
   R2[-i, -j, -m, -n] +
   (1/2) (G[-i, -m] (R5[-j, -n] + R4[-j, -n]) +
```

```
      G[-j, -n] (R5[-i, -m] + R4[-i, -m]) -
      G[-j, -m] (R5[-i, -n] + R4[-i, -n]) -
      G[-i, -n] (R5[-j, -m] + R4[-j, -m])) -
   (1/12) (G[-i, -m] G[-j, -n] - G[-i, -n] G[-j, -m]) R6[];

TDefinition = (2/3) (T1[-i, -j, -k] - T1[-i, -k, -j]) +
   (1/3) (G[-i, -j] T2[-k] - G[-i, -k] T2[-j]) +
   epsilonG[-i, -j, -k, -m] T3[m];

RSO13Activate = MakeRule[{R[-i, -j, -m, -n], Evaluate[RDefinition]},
   MetricOn → All, ContractMetrics → True];
TSO13Activate = MakeRule[{T[-i, -j, -k], Evaluate[TDefinition]},
   MetricOn → All, ContractMetrics → True];

StrengthSO13Activate = Join[RSO13Activate, TSO13Activate];
ClearBuild[];
```

build

Basic/Nester forms of B and A

```
ASymb = "𝒜";
DefTensor[A[a, c, -d], M4,
  Antisymmetric[{a, c}], PrintAs -> SymbolBuild[ASymb]];
DeclareOrder[A[a, c, -d], 1];
DefTensor[A1[-k, -i, -j], M4,
  Symmetric[{-i, -j}], PrintAs -> SymbolBuild[ASymb, SO1]];
DeclareOrder[A1[-k, -i, -j], 1];
DefTensor[A2[-i], M4, PrintAs -> SymbolBuild[ASymb, SO2]];
DeclareOrder[A2[-i], 1];
DefTensor[A3[-i], M4, PrintAs -> SymbolBuild[ASymb, SO3]];
DeclareOrder[A3[-i], 1];
AutomaticRules[A1,
  MakeRule[{A1[a, a1, -a1], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[A1, MakeRule[{A1[a, -a, -k], 0},
   MetricOn → All, ContractMetrics → True]];

ADefinition = (2/3) (A1[-k, -i, -j] - A1[-j, -i, -k]) +
   (1/3) (G[-i, -j] A2[-k] - G[-i, -k] A2[-j]) +
   epsilonG[-i, -j, -k, -m] A3[m];

ASO13Activate = MakeRule[{A[-j, -k, -i], Evaluate[ADefinition]},
   MetricOn → All, ContractMetrics → True];
```

```
BSymb = "b";
FSymb = "f";
DefTensor[F[-i, -j], M4, PrintAs → SymbolBuild[FSymb]];
DefTensor[F1[-i, -j], M4,
  Antisymmetric[{-i, -j}], PrintAs -> SymbolBuild[FSymb, SO1]];
DeclareOrder[F1[-i, -j], 1];
DefTensor[F2[-i, -j], M4,
  Symmetric[{-i, -j}], PrintAs -> SymbolBuild[FSymb, SO2]];
DeclareOrder[F2[-i, -j], 1];
DefTensor[F3[], M4, PrintAs -> SymbolBuild[FSymb, SO3]];
DeclareOrder[F3[], 1];
AutomaticRules[F2,
  MakeRule[{F2[a1, -a1], 0}, MetricOn → All, ContractMetrics → True]];

FDefinition = F1[-i, -j] + F2[-i, -j] + (1 / 4) G[-i, -j] F3[];

FSO13Activate = MakeRule[{F[-i, -j], Evaluate[FDefinition]},
   MetricOn → All, ContractMetrics → True];

GaugeSO13Activate = Join[FSO13Activate, ASO13Activate];
ClearBuild[];
```

build

Basic/Nester forms of Rλ and Tλ

build

```
DefTensor[RLambda1[-i, -j, -m, -n], M4, StrongGenSet[{-i, -j, -m, -n},
   GenSet[Cycles[{-i, -j}, {-m, -n}], Cycles[{-i, -m}], Cycles[{-j, -n}]]],
  PrintAs -> SymbolBuild[RLambdaSymb, SO1]];
DeclareOrder[RLambda1[-i, -j, -m, -n], 1];
DefTensor[RLambda2[-i, -j, -m, -n], M4, StrongGenSet[{-i, -j, -m, -n},
   GenSet[-Cycles[{-i, -m}, {-j, -n}], -Cycles[{-i, -j}], -Cycles[{-m, -n}]]],
  PrintAs -> SymbolBuild[RLambdaSymb, SO2]];
DeclareOrder[RLambda2[-i, -j, -m, -n], 1];
DefTensor[RLambda3[-i, -j, -m, -n], M4,
  Antisymmetric[{-i, -j, -m, -n}], PrintAs -> SymbolBuild[RLambdaSymb, SO3]];
DeclareOrder[RLambda3[-i, -j, -m, -n], 1];
DefTensor[RLambda4[-i, -j], M4,
  Symmetric[{-i, -j}], PrintAs -> SymbolBuild[RLambdaSymb, SO4]];
DeclareOrder[RLambda4[-i, -j], 1];
DefTensor[RLambda5[-i, -j], M4,
  Antisymmetric[{-i, -j}], PrintAs -> SymbolBuild[RLambdaSymb, SO5]];
DeclareOrder[RLambda5[-i, -j], 1];
DefTensor[RLambda6[], M4, PrintAs -> SymbolBuild[RLambdaSymb, SO6]];
```

```
DeclareOrder[RLambda6[], 1];
DefTensor[TLambda1[-i, -j, -k], M4,
  Symmetric[{-i, -j}], PrintAs -> SymbolBuild[TLambdaSymb, SO1]];
DeclareOrder[TLambda1[-i, -j, -k], 1];
DefTensor[TLambda2[-i], M4, PrintAs -> SymbolBuild[TLambdaSymb, SO2]];
DeclareOrder[TLambda2[-i], 1];
DefTensor[TLambda3[-i], M4, PrintAs -> SymbolBuild[TLambdaSymb, SO3]];
DeclareOrder[TLambda3[-i], 1];
AutomaticRules[RLambda1,
  MakeRule[{RLambda1[a, a1, b, -b], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[RLambda1, MakeRule[{RLambda1[a, b, a1, -b], 0},
   MetricOn → All, ContractMetrics → True]];
(*AutomaticRules[RLambda1,MakeRule[{RLambda1[a,-a,a1,-a1],0},
   MetricOn→All,ContractMetrics→True]];*)(*redundant*)
(*AutomaticRules[RLambda1,MakeRule[{RLambda1[a,a1,-a,-a1],0},
   MetricOn→All,ContractMetrics→True]];*)(*redundant*)
AutomaticRules[RLambda2, MakeRule[{RLambda2[a, b, a1, -b], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[RLambda4, MakeRule[{RLambda4[a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[TLambda1, MakeRule[{TLambda1[a, a1, -a1], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[TLambda1, MakeRule[{TLambda1[a, -a, -a1], 0},
   MetricOn → All, ContractMetrics → True]];

RLambdaDefinition = RLambda3[-i, -j, -m, -n] +
   (2 / 3) (2 RLambda1[-i, -j, -m, -n] +
      RLambda1[-i, -m, -j, -n]) +
   RLambda2[-i, -j, -m, -n] +
   (1 / 2) (G[-i, -m] (RLambda5[-j, -n] + RLambda4[-j, -n]) +
      G[-j, -n] (RLambda5[-i, -m] + RLambda4[-i, -m]) -
      G[-j, -m] (RLambda5[-i, -n] + RLambda4[-i, -n]) -
      G[-i, -n] (RLambda5[-j, -m] + RLambda4[-j, -m])) -
   (1 / 12) (G[-i, -m] G[-j, -n] - G[-i, -n] G[-j, -m]) RLambda6[];

TLambdaDefinition = (2 / 3) (TLambda1[-i, -j, -k] - TLambda1[-i, -k, -j]) +
   (1 / 3) (G[-i, -j] TLambda2[-k] - G[-i, -k] TLambda2[-j]) +
   epsilonG[-i, -j, -k, -m] TLambda3[m];

RLambdaSO13Activate =
  MakeRule[{RLambda[-i, -j, -m, -n], Evaluate[RLambdaDefinition]},
   MetricOn → All, ContractMetrics → True];
TLambdaSO13Activate = MakeRule[{TLambda[-i, -j, -k], Evaluate[TLambdaDefinition]},
```

```
    MetricOn → All, ContractMetrics → True];

StrengthLambdaSO13Activate = Join[RLambdaSO13Activate, TLambdaSO13Activate];
ClearBuild[];
```

build

Basic/Nester forms of σ

build

In[•]:=

```
DefTensor[Spin1[-i, -j, -k], M4, Symmetric[{-i, -j}], PrintAs -> "σ^(1)"];
DeclareOrder[Spin1[-i, -j, -k], 1];
DefTensor[Spin2[-i], M4, PrintAs -> "σ^(2)"];
DeclareOrder[Spin2[-i], 1];
DefTensor[Spin3[-i], M4, PrintAs -> "σ^(3)"];
DeclareOrder[Spin3[-i], 1];
AutomaticRules[Spin1,
  MakeRule[{Spin1[a, a1, -a1], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[Spin1, MakeRule[{Spin1[a, -a, -a1], 0},
    MetricOn → All, ContractMetrics → True]];

SpinDefinition = (2/3) (Spin1[-i, -j, -k] - Spin1[-i, -k, -j]) +
    (1/3) (G[-i, -j] Spin2[-k] - G[-i, -k] Spin2[-j]) +
    epsilonG[-i, -j, -k, -m] Spin3[m];

DefTensor[STensor[-i, -j, -k], M4, Antisymmetric[{-j, -k}], PrintAs → "σ"];
DeclareOrder[STensor[-i, -j, -k], 1];

SpinSO13Activate = MakeRule[{STensor[-i, -j, -k], Evaluate[SpinDefinition]},
    MetricOn → All, ContractMetrics → True];

StrengthLambdaSO13Activate = Join[RLambdaSO13Activate, TLambdaSO13Activate];
ClearBuild[];
```

build

Define complete projections $\{\mathrm{I}\hat{\mathcal{P}}\}$, $\{\mathrm{M}\hat{\mathcal{P}}\}$

build

```
PpRSymb = "$\hat{\mathcal{P}}_{\mathcal{R}}$";
DefTensor[PR1[-a, -b, -c, -d, e, f, g, h],
  M4, PrintAs → SymbolBuild[PpRSymb, SO1]];
DefTensor[PR2[-a, -b, -c, -d, e, f, g, h], M4, PrintAs → SymbolBuild[PpRSymb, SO2]];
DefTensor[PR3[-i, -k, -l, -m, a, b, c, d], M4, PrintAs → SymbolBuild[PpRSymb, SO3]];
DefTensor[PR4[-i, -k, -l, -m, a, b, c, d], M4, PrintAs → SymbolBuild[PpRSymb, SO4]];
DefTensor[PR5[-i, -k, -l, -m, a, b, c, d], M4, PrintAs → SymbolBuild[PpRSymb, SO5]];
DefTensor[PR6[-i, -k, -l, -m, a, b, c, d], M4, PrintAs → SymbolBuild[PpRSymb, SO6]];

ToCanonicalTotal[x_] := ToCanonical[Total[x]];
ToCanonicalParallel[x_] := Module[{Monomials, Ret},
   Monomials = MonomialList[x];
   Ret = Total[ParallelCombine[ToCanonicalTotal, Monomials, List]];
   Ret];

AutomaticRules[PR1, MakeRule[{CD[-x][PR1[-a, -b, -c, -d, e, f, g, h]], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[PR2, MakeRule[{CD[-x][PR2[-a, -b, -c, -d, e, f, g, h]], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[PR3, MakeRule[{CD[-x][PR3[-a, -b, -c, -d, e, f, g, h]], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[PR4, MakeRule[{CD[-x][PR4[-a, -b, -c, -d, e, f, g, h]], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[PR5, MakeRule[{CD[-x][PR5[-a, -b, -c, -d, e, f, g, h]], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[PR6, MakeRule[{CD[-x][PR6[-a, -b, -c, -d, e, f, g, h]], 0},
   MetricOn → All, ContractMetrics → True]];
PWSymb = "$\mathcal{P}_{\mathcal{W}}$";
DefTensor[PW[-i, -k, -l, -m, a, b, c, d], M4, PrintAs → SymbolBuild[PWSymb]];
PpTSymb = "$\hat{\mathcal{P}}_{\mathcal{T}}$";
DefTensor[PT1[-a, -b, -c, e, f, g], M4, PrintAs → SymbolBuild[PpTSymb, SO1]];
DefTensor[PT2[-a, -b, -c, e, f, g], M4, PrintAs → SymbolBuild[PpTSymb, SO2]];
DefTensor[PT3[-a, -b, -c, e, f, g], M4, PrintAs → SymbolBuild[PpTSymb, SO3]];
AutomaticRules[PT1, MakeRule[{CD[-x][PT1[-a, -b, -c, e, f, g]], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[PT2, MakeRule[{CD[-x][PT2[-a, -b, -c, e, f, g]], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[PT3, MakeRule[{CD[-x][PT3[-a, -b, -c, e, f, g]], 0},
   MetricOn → All, ContractMetrics → True]];
ClearBuild[];
```

O13Projections oT ggle

In[ ]:=

```
IfBuild["O13ProjectionsToggle",
```

```
PWActivate = MakeRule[
  {PW[-i, -k, -l, -m, a, b, c, d], G[a, -i] G[b, -k] G[c, -l] G[d, -m] + (1/2)
     (G[b, d] G[a, -i] G[c, -m] G[-k, -l] - G[b, d] G[a, -i] G[c, -l] G[-k, -m] +
       G[b, d] G[a, -k] G[c, -l] G[-i, -m] - G[b, d] G[a, -k] G[c, -m] G[-i, -l]) +
    (1/6) G[a, c] G[b, d] (G[-i, -l] G[-k, -m] - G[-i, -m] G[-k, -l])},
  MetricOn → All, ContractMetrics → True];

PR1Definition =
 Antisymmetrize[Antisymmetrize[Antisymmetrize[Antisymmetrize[(2/3)
       G[s, -i] G[r, -n] (2 G[p, -j] G[q, -m] + G[p, -m] G[q, -j])
       (1/2) (Symmetrize[PW[-s, -p, -q, -r, a, b, c, d] +
          PW[-s, -r, -q, -p, a, b, c, d], {-s, -q}]), {-i, -j}],
     {-m, -n}], {a, b}], {c, d}] /. PWActivate // ToCanonical;
PR1Activate = MakeRule[{PR1[-i, -j, -m, -n, a, b, c, d], Evaluate[PR1Definition]},
  MetricOn → All, ContractMetrics → True];

PR2Definition =
 Antisymmetrize[Antisymmetrize[Antisymmetrize[Antisymmetrize[(1/2)
       (PW[-i, -j, -m, -n, a, b, c, d] - PW[-m, -n, -i, -j, a, b, c, d]),
      {-i, -j}], {-m, -n}], {a, b}], {c, d}] /. PWActivate // ToCanonical;
PR2Activate = MakeRule[{PR2[-i, -j, -m, -n, a, b, c, d], Evaluate[PR2Definition]},
  MetricOn → All, ContractMetrics → True];

PR3Definition = Antisymmetrize[Antisymmetrize[Antisymmetrize[
     Antisymmetrize[(-1/4) (1/6) epsilonG[-i, -j, -m, -n] epsilonG[a, b, c, d],
      {-i, -j}], {-m, -n}], {a, b}], {c, d}] // ToCanonical;
PR3Activate = MakeRule[{PR3[-i, -j, -m, -n, a, b, c, d], Evaluate[PR3Definition]},
  MetricOn → All, ContractMetrics → True];

PR4Definition =
 Antisymmetrize[Antisymmetrize[Antisymmetrize[Antisymmetrize[(1/2)
       (G[-i, -m] G[x, -j] G[y, -n] + G[-j, -n] G[x, -i] G[y, -m] - G[-j, -m]
          G[x, -i] G[y, -n] - G[-i, -n] G[x, -j] G[y, -m]) (Symmetrize[
          G[-x, a] G[-y, c] G[b, d], {-x, -y}] - (1/4) G[-x, -y] G[b, d] G[a, c]),
      {-i, -j}], {-m, -n}], {a, b}], {c, d}] // ToCanonical;
PR4Activate = MakeRule[{PR4[-i, -j, -m, -n, a, b, c, d], Evaluate[PR4Definition]},
  MetricOn → All, ContractMetrics → True];

PR5Definition =
 Antisymmetrize[Antisymmetrize[Antisymmetrize[Antisymmetrize[(1/2)
       (G[-i, -m] G[x, -j] G[y, -n] + G[-j, -n] G[x, -i] G[y, -m] - G[-j, -m]
          G[x, -i] G[y, -n] - G[-i, -n] G[x, -j] G[y, -m])
       Antisymmetrize[G[-x, a] G[-y, c] G[b, d], {-x, -y}], {-i, -j}],
```

```
      {-m, -n}], {a, b}], {c, d}] // ToCanonical;
  PR5Activate = MakeRule[{PR5[-i, -j, -m, -n, a, b, c, d], Evaluate[PR5Definition]},
    MetricOn → All, ContractMetrics → True];

  PR6Definition =
   Antisymmetrize[Antisymmetrize[Antisymmetrize[Antisymmetrize[-(1/6) G[b, d]
         G[a, c] (G[-i, -j] G[-m, -n] - G[-i, -m] G[-j, -n]), {-i, -j}],
       {-m, -n}], {a, b}], {c, d}] // ToCanonical;
  PR6Activate = MakeRule[{PR6[-i, -j, -m, -n, a, b, c, d], Evaluate[PR6Definition]},
    MetricOn → All, ContractMetrics → True];

  PT1Definition =
   Antisymmetrize[Antisymmetrize[(4/3) (Symmetrize[G[-i, a] G[-j, b] G[-k, c] +
           (1/3) G[-k, -i] G[a, b] G[c, -j], {-i, -j}] -
         (1/3) G[-i, -j] G[a, b] G[c, -k]), {-j, -k}], {b, c}] // ToCanonical;
  PT1Activate = MakeRule[{PT1[-i, -j, -k, a, b, c], Evaluate[PT1Definition]},
    MetricOn → All, ContractMetrics → True];

  PT2Definition = Antisymmetrize[Antisymmetrize[
      (2/3) G[-i, -j] G[a, b] G[c, -k], {-j, -k}], {b, c}] // ToCanonical;
  PT2Activate = MakeRule[{PT2[-i, -j, -k, a, b, c], Evaluate[PT2Definition]},
    MetricOn → All, ContractMetrics → True];

  PT3Definition = Antisymmetrize[Antisymmetrize[(1/6) epsilonG[-i, -j, -k, -m]
       epsilonG[m, a, b, c], {-j, -k}], {b, c}] // ToCanonical;
  PT3Activate = MakeRule[{PT3[-i, -j, -k, a, b, c], Evaluate[PT3Definition]},
    MetricOn → All, ContractMetrics → True];

  PActivate = Join[PWActivate, PR1Activate, PR2Activate, PR3Activate, PR4Activate,
    PR5Activate, PR6Activate, PT1Activate, PT2Activate, PT3Activate];

  DumpSave[BinaryLocation["O13ProjectionsToggle"], {PActivate}];
  ClearBuild[];
 ];
```

build

In[•]:=

```
OpenLastCache[];
```

CheckOrthogonality oT ggle

```
IfBuild["CheckOrthogonalityToggle",
  HiGGSPrint[ActiveCellTags];
  HiGGSPrint[Style["checking orthogonality", Blue, 16]];
  For[ii = 1, ii < 7, ii++, For[jj = 1, jj < 7, jj++, If[ii ≠ jj,
     HiGGSPrint[ToExpression["PR" <> ToString[ii] <> "[-i,-k,-l,-m,a,b,c,d]PR" <>
```

```
        ToString[jj] <> "[-a,-b,-c,-d,e,f,g,h]R[-e,-f,-g,-h]"] /.
      PActivate // ToCanonical]]]];
For[ii = 1, ii < 4, ii++, For[jj = 1, jj < 4, jj++, If[ii ≠ jj, HiGGSPrint[
    ToExpression["PT" <> ToString[ii] <> "[-i,-j,-k,a,b,c]PT" <> ToString[jj] <>
        "[-a,-b,-c,e,f,g]T[-e,-f,-g]"] /. PActivate // ToCanonical]]]];

HiGGSPrint[Style["checking inverse orthogonality", Blue, 16]];

For[ii = 1, ii < 7, ii++, For[jj = 1, jj < 7, jj++, If[ii ≠ jj, HiGGSPrint[
     ToExpression["PR" <> ToString[ii] <> "[a,b,c,d,i,j,k,l]R[-i,-j,-k,-l]PR" <>
         ToString[jj] <> "[-a,-b,-c,-d,e,f,g,h]R[-e,-f,-g,-h]"] /.
       PActivate // ToCanonical]]]] ×
 For[ii = 1, ii < 4, ii++, For[jj = 1, jj < 4, jj++,
   If[ii ≠ jj, HiGGSPrint[ToExpression[
        "PT" <> ToString[ii] <> "[a,b,c,i,j,k]T[-i,-j,-k]PT" <> ToString[jj] <>
         "[-a,-b,-c,e,f,g]T[-e,-f,-g]"] /. PActivate // ToCanonical]]]];

HiGGSPrint[Style["checking idempotency", Blue, 16]];

For[ii = 1, ii < 7, ii++,
  HiGGSPrint[ToExpression["(PR" <> ToString[ii] <> "[-i,-k,-l,-m,a,b,c,d]PR" <>
        ToString[ii] <> "[-a,-b,-c,-d,e,f,g,h]-PR" <>
        ToString[ii] <> "[-i,-k,-l,-m,e,f,g,h])R[-e,-f,-g,-h]"] /.
      PActivate // ToCanonical // FullSimplify]] ×
 For[ii = 1, ii < 4, ii++, HiGGSPrint[ToExpression["(PT" <> ToString[ii] <>
        "[-i,-j,-k,a,b,c]PT" <> ToString[ii] <> "[-a,-b,-c,e,f,g]-PT" <>
        ToString[ii] <> "[-i,-j,-k,e,f,g])T[-e,-f,-g]"] /.
      PActivate // ToCanonical // FullSimplify]];

HiGGSPrint[Style["checking completeness", Blue, 16]];

(PR1[-i, -k, -l, -m, a, b, c, d] + PR2[-i, -k, -l, -m, a, b, c, d] +
        PR3[-i, -k, -l, -m, a, b, c, d] + PR4[-i, -k, -l, -m, a, b, c, d] +
        PR5[-i, -k, -l, -m, a, b, c, d] + PR6[-i, -k, -l, -m, a, b, c, d])
      R[-a, -b, -c, -d] /. PActivate // ToCanonical // Simplify ×
     (PT1[-i, -k, -l, a, b, c] + PT2[-i, -k, -l, a, b, c] + PT3[-i, -k, -l, a, b, c])
     T[-a, -b, -c] /. PActivate // ToCanonical // Simplify;

HiGGSPrint[Style["checking invertability", Blue, 16]];

For[ii = 1, ii < 7, ii++,
 HiGGSPrint[ToExpression["(PR" <> ToString[ii] <> "[e,f,g,h,-i,-k,-l,-m]-PR" <>
       ToString[ii] <> "[-i,-k,-l,-m,e,f,g,h])R[-e,-f,-g,-h]"] /.
```

```
      PActivate // ToCanonical // FullSimplify]];
  ClearBuild[];
 ];
```

build

Define Ricci, Ricci scalar and torsion contraction

build

```
(*Define the Ricci \mathcal{R}^a_{\ b}*)
DefTensor[Rc[a, -b], M4, PrintAs → SymbolBuild[RSymb]];
DeclareOrder[Rc[a, -b], 1];
(*Define the Ricci scalar \mathcal{R}*)
DefTensor[Rs[], M4, PrintAs → SymbolBuild[RSymb]];
DeclareOrder[Rs[], 1];
(*Define the torsion contraction \mathcal{T}^a*)
DefTensor[Tc[-a], M4, PrintAs → SymbolBuild[TSymb]];
DeclareOrder[Tc[-a], 1];
(*Rule to expand Ricci*)
ExpandRicci =
  MakeRule[{Rc[a, -b], R[c, a, -c, -b]}, MetricOn → All, ContractMetrics → True];
(*Rule to expand Ricci scalar*)
ExpandRicciScalar =
  MakeRule[{Rs[], R[c, d, -c, -d]}, MetricOn → All, ContractMetrics → True];
(*Rule to expand torsion contraction*)
TorsionExpandContraction =
  MakeRule[{Tc[-a], T[b, -a, -b]}, MetricOn → All, ContractMetrics → True];
(*Total rule to expand contracted field-strength tensors*)
ExpandContractedStrengths =
  Join[ExpandRicci, ExpandRicciScalar, TorsionExpandContraction];

(*Rule to expand Ricci*)
ContractRicci =
  MakeRule[{R[c, a, -c, -b], Rc[a, -b]}, MetricOn → All, ContractMetrics → True];
(*Rule to expand Ricci scalar*)
ContractRicciScalar =
  MakeRule[{R[c, d, -c, -d], Rs[]}, MetricOn → All, ContractMetrics → True];
(*Rule to expand torsion contraction*)
TorsionContractContraction =
  MakeRule[{T[b, -a, -b], Tc[-a]}, MetricOn → All, ContractMetrics → True];
(*Total rule to expand contracted field-strength tensors*)
ContractExpandedStrengths =
  Join[ContractRicci, ContractRicciScalar, TorsionContractContraction];
ClearBuild[];
```

ShowIrreps oT ggle

```
IfBuild["ShowIrrepsToggle",
```

```
(*Irreducible decompositions*)
AutomaticRules[R, MakeRule[
  {R[c, a, -c, -b], Rc[a, -b]}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[Rc, MakeRule[{Rc[c, -c], Rs[]},
  MetricOn → All, ContractMetrics → True]];
AutomaticRules[T, MakeRule[{T[c, -a, -c], Tc[-a]},
  MetricOn → All, ContractMetrics → True]];
PR1[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate // ToCanonical //
 ContractMetric;
HiGGSPrint[%];
PR2[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate // ToCanonical //
 ContractMetric;
HiGGSPrint[%];
PR3[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate // ToCanonical //
 ContractMetric;
HiGGSPrint[%];
PR4[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate // ToCanonical //
 ContractMetric;
HiGGSPrint[%];
PR5[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate // ToCanonical //
 ContractMetric;
HiGGSPrint[%];
PR6[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate // ToCanonical //
 ContractMetric;
HiGGSPrint[%];
PT1[-i, -j, -k, a, b, c] T[-a, -b, -c] /. PActivate // ToCanonical //
 ContractMetric;
HiGGSPrint[%];
PT2[-i, -j, -k, a, b, c] T[-a, -b, -c] /. PActivate // ToCanonical //
 ContractMetric;
HiGGSPrint[%];
PT3[-i, -j, -k, a, b, c] T[-a, -b, -c] /. PActivate // ToCanonical //
 ContractMetric;
HiGGSPrint[%];

tmp = PR1[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate /.
    StrengthSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PR2[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate /.
    StrengthSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PR3[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate /.
    StrengthSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
```

```
tmp = PR4[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate /.
   StrengthSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PR5[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate /.
   StrengthSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PR6[-i, -j, -k, -l, a, b, c, d] R[-a, -b, -c, -d] /. PActivate /.
   StrengthSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PT1[-i, -j, -k, a, b, c] T[-a, -b, -c] /. PActivate /.
   StrengthSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PT2[-i, -j, -k, a, b, c] T[-a, -b, -c] /. PActivate /.
   StrengthSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PT3[-i, -j, -k, a, b, c] T[-a, -b, -c] /. PActivate /.
   StrengthSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];

tmp = PR1[-i, -j, -k, -l, a, b, c, d] RLambda[-a, -b, -c, -d] /. PActivate /.
   StrengthLambdaSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PR2[-i, -j, -k, -l, a, b, c, d] RLambda[-a, -b, -c, -d] /. PActivate /.
   StrengthLambdaSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PR3[-i, -j, -k, -l, a, b, c, d] RLambda[-a, -b, -c, -d] /. PActivate /.
   StrengthLambdaSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PR4[-i, -j, -k, -l, a, b, c, d] RLambda[-a, -b, -c, -d] /. PActivate /.
   StrengthLambdaSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PR5[-i, -j, -k, -l, a, b, c, d] RLambda[-a, -b, -c, -d] /. PActivate /.
   StrengthLambdaSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PR6[-i, -j, -k, -l, a, b, c, d] RLambda[-a, -b, -c, -d] /. PActivate /.
   StrengthLambdaSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PT1[-i, -j, -k, a, b, c] TLambda[-a, -b, -c] /. PActivate /.
   StrengthLambdaSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PT2[-i, -j, -k, a, b, c] TLambda[-a, -b, -c] /. PActivate /.
   StrengthLambdaSO13Activate // ToNewCanonical;
HiGGSPrint[tmp];
tmp = PT3[-i, -j, -k, a, b, c] TLambda[-a, -b, -c] /. PActivate /.
```

```
    StrengthLambdaSO13Activate // ToNewCanonical;
  HiGGSPrint[tmp];
  ClearBuild[];
 ];
```

build

Multiplier couplings $\{\bar{\alpha}_I\}, \{\bar{\beta}_M\}$

build

```
(*My couplings for irrep Lorentz constraints*)
cAlpSymb = "ᾱ";
DefConstantSymbol[cAlp1,
  PrintAs → SymbolBuild[cAlpSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[cAlp2, PrintAs →
   SymbolBuild[cAlpSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[cAlp3, PrintAs →
   SymbolBuild[cAlpSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[cAlp4, PrintAs →
   SymbolBuild[cAlpSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[cAlp5, PrintAs →
   SymbolBuild[cAlpSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[cAlp6, PrintAs →
   SymbolBuild[cAlpSymb, dSO6, "Constant" -> True]];

cAlp = {cAlp1, cAlp2, cAlp3, cAlp4, cAlp5, cAlp6};

(*My couplings for irrep Lorentz constraints*)
gAlpSymb = "á";
DefConstantSymbol[gAlp1,
  PrintAs → SymbolBuild[gAlpSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[gAlp2, PrintAs →
   SymbolBuild[gAlpSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[gAlp3, PrintAs →
   SymbolBuild[gAlpSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[gAlp4, PrintAs →
   SymbolBuild[gAlpSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[gAlp5, PrintAs →
   SymbolBuild[gAlpSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[gAlp6, PrintAs →
   SymbolBuild[gAlpSymb, dSO6, "Constant" -> True]];

gAlp = {gAlp1, gAlp2, gAlp3, gAlp4, gAlp5, gAlp6};

cAlpParaParaSymb = "ᾱ"";
```

```
DefConstantSymbol[cAlpParaPara0p,
  PrintAs → SymbolBuild[cAlpParaParaSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[cAlpParaPara0m,
  PrintAs → SymbolBuild[cAlpParaParaSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[cAlpParaPara1p,
  PrintAs → SymbolBuild[cAlpParaParaSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[cAlpParaPara1m,
  PrintAs → SymbolBuild[cAlpParaParaSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[cAlpParaPara2p,
  PrintAs → SymbolBuild[cAlpParaParaSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[cAlpParaPara2m,
  PrintAs → SymbolBuild[cAlpParaParaSymb, dSO6, "Constant" -> True]];

cAlpParaPara = {cAlpParaPara0p, cAlpParaPara0m,
   cAlpParaPara1p, cAlpParaPara1m, cAlpParaPara2p, cAlpParaPara2m};

cAlpPerpPerpSymb = "\!\(\*OverscriptBox[\(\[Alpha]\), \(\[Perpendicular]\[Perpendicular]\)]\)";
DefConstantSymbol[cAlpPerpPerp0p,
  PrintAs → SymbolBuild[cAlpPerpPerpSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[cAlpPerpPerp0m,
  PrintAs → SymbolBuild[cAlpPerpPerpSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[cAlpPerpPerp1p,
  PrintAs → SymbolBuild[cAlpPerpPerpSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[cAlpPerpPerp1m,
  PrintAs → SymbolBuild[cAlpPerpPerpSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[cAlpPerpPerp2p,
  PrintAs → SymbolBuild[cAlpPerpPerpSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[cAlpPerpPerp2m,
  PrintAs → SymbolBuild[cAlpPerpPerpSymb, dSO6, "Constant" -> True]];

cAlpPerpPerp = {cAlpPerpPerp0p, cAlpPerpPerp0m,
   cAlpPerpPerp1p, cAlpPerpPerp1m, cAlpPerpPerp2p, cAlpPerpPerp2m};

cAlpPerpParaSymb = "\!\(\*OverscriptBox[\(\[Alpha]\), \(\[Perpendicular]\[Parallel]\)]\)";
DefConstantSymbol[cAlpPerpPara0p,
  PrintAs → SymbolBuild[cAlpPerpParaSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[cAlpPerpPara0m,
  PrintAs → SymbolBuild[cAlpPerpParaSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[cAlpPerpPara1p,
  PrintAs → SymbolBuild[cAlpPerpParaSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[cAlpPerpPara1m,
  PrintAs → SymbolBuild[cAlpPerpParaSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[cAlpPerpPara2p,
  PrintAs → SymbolBuild[cAlpPerpParaSymb, dSO5, "Constant" -> True]];
```

```
DefConstantSymbol[cAlpPerpPara2m,
  PrintAs → SymbolBuild[cAlpPerpParaSymb, dSO6, "Constant" -> True]];

cAlpPerpPara = {cAlpPerpPara0p, cAlpPerpPara0m,
    cAlpPerpPara1p, cAlpPerpPara1m, cAlpPerpPara2p, cAlpPerpPara2m};

cAlpParaPerpSymb = "\!\(\*OverscriptBox[\(\[Alpha]\), \(\[Perpendicular]\)]\)";
DefConstantSymbol[cAlpParaPerp0p,
  PrintAs → SymbolBuild[cAlpParaPerpSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[cAlpParaPerp0m,
  PrintAs → SymbolBuild[cAlpParaPerpSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[cAlpParaPerp1p,
  PrintAs → SymbolBuild[cAlpParaPerpSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[cAlpParaPerp1m,
  PrintAs → SymbolBuild[cAlpParaPerpSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[cAlpParaPerp2p,
  PrintAs → SymbolBuild[cAlpParaPerpSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[cAlpParaPerp2m,
  PrintAs → SymbolBuild[cAlpParaPerpSymb, dSO6, "Constant" -> True]];

cAlpParaPerp = {cAlpParaPerp0p, cAlpParaPerp0m,
    cAlpParaPerp1p, cAlpParaPerp1m, cAlpParaPerp2p, cAlpParaPerp2m};

cBetSymb = "\!\(\*OverscriptBox[\(\[Beta]\), \(\)]\)";
DefConstantSymbol[cBet1,
  PrintAs → SymbolBuild[cBetSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[cBet2, PrintAs →
    SymbolBuild[cBetSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[cBet3, PrintAs →
    SymbolBuild[cBetSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[cBet4, PrintAs →
    SymbolBuild[cBetSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[cBet5, PrintAs →
    SymbolBuild[cBetSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[cBet6, PrintAs →
    SymbolBuild[cBetSymb, dSO6, "Constant" -> True]];

cBet = {cBet1, cBet2, cBet3};

gBetSymb = "\[Beta]";
DefConstantSymbol[gBet1,
  PrintAs → SymbolBuild[gBetSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[gBet2, PrintAs →
    SymbolBuild[gBetSymb, dSO2, "Constant" -> True]];
```

```
DefConstantSymbol[gBet3, PrintAs →
    SymbolBuild[gBetSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[gBet4, PrintAs →
    SymbolBuild[gBetSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[gBet5, PrintAs →
    SymbolBuild[gBetSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[gBet6, PrintAs →
    SymbolBuild[gBetSymb, dSO6, "Constant" -> True]];

gBet = {gBet1, gBet2, gBet3};

cBetParaPara = "β̄∥∥";
DefConstantSymbol[cBetParaPara0p,
  PrintAs → SymbolBuild[cBetParaPara, dSO1, "Constant" -> True]];
DefConstantSymbol[cBetParaPara0m,
  PrintAs → SymbolBuild[cBetParaPara, dSO2, "Constant" -> True]];
DefConstantSymbol[cBetParaPara1p,
  PrintAs → SymbolBuild[cBetParaPara, dSO3, "Constant" -> True]];
DefConstantSymbol[cBetParaPara1m,
  PrintAs → SymbolBuild[cBetParaPara, dSO4, "Constant" -> True]];
DefConstantSymbol[cBetParaPara2p,
  PrintAs → SymbolBuild[cBetParaPara, dSO5, "Constant" -> True]];
DefConstantSymbol[cBetParaPara2m,
  PrintAs → SymbolBuild[cBetParaPara, dSO6, "Constant" -> True]];

cBetParaPara = {cBetParaPara0p, cBetParaPara0m,
    cBetParaPara1p, cBetParaPara1m, cBetParaPara2p, cBetParaPara2m};

cBetPerpPerp = "β̄⊥⊥";
DefConstantSymbol[cBetPerpPerp0p,
  PrintAs → SymbolBuild[cBetPerpPerp, dSO1, "Constant" -> True]];
DefConstantSymbol[cBetPerpPerp0m,
  PrintAs → SymbolBuild[cBetPerpPerp, dSO2, "Constant" -> True]];
DefConstantSymbol[cBetPerpPerp1p,
  PrintAs → SymbolBuild[cBetPerpPerp, dSO3, "Constant" -> True]];
DefConstantSymbol[cBetPerpPerp1m,
  PrintAs → SymbolBuild[cBetPerpPerp, dSO4, "Constant" -> True]];
DefConstantSymbol[cBetPerpPerp2p,
  PrintAs → SymbolBuild[cBetPerpPerp, dSO5, "Constant" -> True]];
DefConstantSymbol[cBetPerpPerp2m,
  PrintAs → SymbolBuild[cBetPerpPerp, dSO6, "Constant" -> True]];

cBetPerpPerp = {cBetPerpPerp0p, cBetPerpPerp0m,
    cBetPerpPerp1p, cBetPerpPerp1m, cBetPerpPerp2p, cBetPerpPerp2m};
```

```
cBetPerpPara = "β⃗^(⊥∥)";
DefConstantSymbol[cBetPerpPara0p,
  PrintAs → SymbolBuild[cBetPerpPara, dSO1, "Constant" -> True]];
DefConstantSymbol[cBetPerpPara0m,
  PrintAs → SymbolBuild[cBetPerpPara, dSO2, "Constant" -> True]];
DefConstantSymbol[cBetPerpPara1p,
  PrintAs → SymbolBuild[cBetPerpPara, dSO3, "Constant" -> True]];
DefConstantSymbol[cBetPerpPara1m,
  PrintAs → SymbolBuild[cBetPerpPara, dSO4, "Constant" -> True]];
DefConstantSymbol[cBetPerpPara2p,
  PrintAs → SymbolBuild[cBetPerpPara, dSO5, "Constant" -> True]];
DefConstantSymbol[cBetPerpPara2m,
  PrintAs → SymbolBuild[cBetPerpPara, dSO6, "Constant" -> True]];

cBetPerpPara = {cBetPerpPara0p, cBetPerpPara0m,
    cBetPerpPara1p, cBetPerpPara1m, cBetPerpPara2p, cBetPerpPara2m};

cBetParaPerp = "β⃗^(∥⊥)";
DefConstantSymbol[cBetParaPerp0p,
  PrintAs → SymbolBuild[cBetParaPerp, dSO1, "Constant" -> True]];
DefConstantSymbol[cBetParaPerp0m,
  PrintAs → SymbolBuild[cBetParaPerp, dSO2, "Constant" -> True]];
DefConstantSymbol[cBetParaPerp1p,
  PrintAs → SymbolBuild[cBetParaPerp, dSO3, "Constant" -> True]];
DefConstantSymbol[cBetParaPerp1m,
  PrintAs → SymbolBuild[cBetParaPerp, dSO4, "Constant" -> True]];
DefConstantSymbol[cBetParaPerp2p,
  PrintAs → SymbolBuild[cBetParaPerp, dSO5, "Constant" -> True]];
DefConstantSymbol[cBetParaPerp2m,
  PrintAs → SymbolBuild[cBetParaPerp, dSO6, "Constant" -> True]];

cBetParaPerp = {cBetParaPerp0p, cBetParaPerp0m,
    cBetParaPerp1p, cBetParaPerp1m, cBetParaPerp2p, cBetParaPerp2m};
ClearBuild[];
```

build

Quadratic couplings $\hat{\alpha}_0$, $\{\hat{\alpha}_I\}$, $\{\hat{\beta}_M\}$

build

```
(*Mike's couplings for irrep Lorentz constraints*)
mAlpSymb = "α";
DefConstantSymbol[mAlp0,
  PrintAs → SymbolBuild[mAlpSymb, dSO0, "Constant" -> True]];
```

```
DefConstantSymbol[mAlp1, PrintAs →
    SymbolBuild[mAlpSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[mAlp2, PrintAs →
    SymbolBuild[mAlpSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[mAlp3, PrintAs →
    SymbolBuild[mAlpSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[mAlp4, PrintAs →
    SymbolBuild[mAlpSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[mAlp5, PrintAs →
    SymbolBuild[mAlpSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[mAlp6, PrintAs →
    SymbolBuild[mAlpSymb, dSO6, "Constant" -> True]];

mAlp = {mAlp1, mAlp2, mAlp3, mAlp4, mAlp5, mAlp6};

(*My couplings for irrep Lorentz constraints*)
AlpSymb = "α̂";
DefConstantSymbol[Alp0,
  PrintAs → SymbolBuild[AlpSymb, dSO0, "Constant" -> True]];
DefConstantSymbol[Alp1, PrintAs → SymbolBuild[AlpSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[Alp2, PrintAs → SymbolBuild[AlpSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[Alp3, PrintAs → SymbolBuild[AlpSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[Alp4, PrintAs → SymbolBuild[AlpSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[Alp5, PrintAs → SymbolBuild[AlpSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[Alp6, PrintAs → SymbolBuild[AlpSymb, dSO6, "Constant" -> True]];

Alp = {Alp1, Alp2, Alp3, Alp4, Alp5, Alp6};

mBetSymb = "β";
DefConstantSymbol[mBet1,
  PrintAs → SymbolBuild[mBetSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[mBet2, PrintAs →
    SymbolBuild[mBetSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[mBet3, PrintAs →
    SymbolBuild[mBetSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[mBet4, PrintAs →
    SymbolBuild[mBetSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[mBet5, PrintAs →
    SymbolBuild[mBetSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[mBet6, PrintAs →
    SymbolBuild[mBetSymb, dSO6, "Constant" -> True]];

mBet = {mBet1, mBet2, mBet3};
```

```
BetSymb = "β̂";
DefConstantSymbol[Bet1,
  PrintAs → SymbolBuild[BetSymb, dSO1, "Constant" -> True]];
DefConstantSymbol[Bet2, PrintAs → SymbolBuild[BetSymb, dSO2, "Constant" -> True]];
DefConstantSymbol[Bet3, PrintAs → SymbolBuild[BetSymb, dSO3, "Constant" -> True]];
DefConstantSymbol[Bet4, PrintAs → SymbolBuild[BetSymb, dSO4, "Constant" -> True]];
DefConstantSymbol[Bet5, PrintAs → SymbolBuild[BetSymb, dSO5, "Constant" -> True]];
DefConstantSymbol[Bet6, PrintAs → SymbolBuild[BetSymb, dSO6, "Constant" -> True]];

Bet = {Bet1, Bet2, Bet3};
ClearBuild[];
```

build

Dynamical variables

build

Define variables

build

```
VSymb = "n";
DefTensor[V[-a], M4, PrintAs → SymbolBuild[VSymb]];
AutomaticRules[V, MakeRule[{V[-a] V[a], 1}]];
NSymb = "𝒩";
DefTensor[Lapse[], M4, PrintAs → SymbolBuild[NSymb]];
JiSymb = "𝒥^-1";
DefTensor[Ji[], M4, PrintAs → SymbolBuild[JiSymb]];
JSymb = "𝒥";
DefTensor[J[], M4, PrintAs → SymbolBuild[JSymb]];
AutomaticRules[J,
  MakeRule[{J[] Ji[], 1}, MetricOn → All, ContractMetrics → True]];
CollapseJ1 = MakeRule[{J[] Ji[], 1}, MetricOn → All, ContractMetrics → True];
CollapseJ2 = MakeRule[{J[] Ji[]^2, Ji[]}, MetricOn → All, ContractMetrics → True];
CollapseJ3 = MakeRule[{J[]^2 Ji[], J[]}, MetricOn → All, ContractMetrics → True];
CollapseJ = Join[CollapseJ1, CollapseJ2, CollapseJ3];

JiToJ = MakeRule[{Ji[], 1 / J[]}, MetricOn → All, ContractMetrics → True];

APiSymb = "π_𝔄";
DefTensor[APi[-a, -b, -c], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[APiSymb]];
DeclareOrder[APi[-a, -b, -c], 1, "IsUnityWithEHTerm" → True];
APiPSymb = "π̂_𝔄";
DefTensor[APiP[-a, -b, -c], M4, Antisymmetric[{-a, -b}],
```

```
  PrintAs -> SymbolBuild[APiPSymb], OrthogonalTo → {V[c]}];
DeclareOrder[APiP[-a, -b, -c], 1, "IsUnityWithEHTerm" → True];
BPiSymb = "πb";
DefTensor[BPi[-a, -c], M4, PrintAs → SymbolBuild[BPiSymb]];
DeclareOrder[BPi[-a, -c], 1];
BPiPSymb = "π̂b";
DefTensor[BPiP[-a, -c], M4,
  PrintAs → SymbolBuild[BPiPSymb], OrthogonalTo → {V[c]}];
DeclareOrder[BPiP[-a, -c], 1];
HSymb = "h";
DefTensor[H[-a, c], M4, PrintAs → SymbolBuild[HSymb]];
DefTensor[B[a, -c], M4, PrintAs → SymbolBuild[BSymb]];

(*this section inserted to allow
 perturbative expansion of the H and B fields*)
HToF = MakeRule[{H[-i, -j], G[-i, -j] + F[-i, -j]},
   MetricOn → All, ContractMetrics → True];
BToF = MakeRule[{B[-i, -j], G[-i, -j] - F[-i, -j] + F[-i, -m] F[m, -j]},
   MetricOn → All, ContractMetrics → True];
ToF = Join[HToF, BToF];

(*Rule to contract Roman indices*)
AutomaticRules[H,
  MakeRule[{H[-a, i] B[a, -j], G[i, -j]}, MetricOn → All, ContractMetrics → True]];
(*Rule to contract Greek indices*)
AutomaticRules[H,
  MakeRule[{H[-a, i] B[c, -i], G[-a, c]}, MetricOn → All, ContractMetrics → True]];


G3Symb = "γ"";
DefTensor[G3[-a, -b], M4, Symmetric[{-a, -b}], PrintAs → SymbolBuild[G3Symb]];
AutomaticRules[G3, MakeRule[{G3[-a, -b] G3[b, -d], G3[-a, -d]},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[G3, MakeRule[{G3[-a, a], 3}, MetricOn → All,
   ContractMetrics → True]];
AutomaticRules[G3, MakeRule[{B[a, -b] G3[b, -c] V[-a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[G3, MakeRule[{CD[-a][G3[-c, b]], 0},
   MetricOn → All, ContractMetrics → True]];

EpsSymb = "ϵ"";
DefTensor[Eps[-a, -b, -c], M4, Antisymmetric[{-a, -b, -c}],
  OrthogonalTo → {V[a], V[b], V[c]}, PrintAs → SymbolBuild[EpsSymb]];
DeclareOrder[CD[-z][Eps[-a, -b, -c]], 1];
```

```
FoliGSymb = "η"";
DefTensor[FoliG[-a, -b], M4, Symmetric[{-a, -b}],
  OrthogonalTo → {V[a], V[b]}, PrintAs → SymbolBuild[FoliGSymb]];
DeclareOrder[CD[-z][FoliG[-a, -b]], 1];
epsilonGVToEps = MakeRule[{V[d] epsilonG[-a, -b, -c, -d], Eps[-a, -b, -c]},
    MetricOn → All, ContractMetrics → True];
EpsToepsilonGV = MakeRule[{Eps[-a, -b, -c], V[d] epsilonG[-a, -b, -c, -d]},
    MetricOn → All, ContractMetrics → True];
GToFoliG = MakeRule[{G[-a, -b], FoliG[-a, -b] + V[-a] V[-b]},
    MetricOn → All, ContractMetrics → True];
FoliGToG = MakeRule[{FoliG[-a, -b], G[-a, -b] - V[-a] V[-b]},
    MetricOn → All, ContractMetrics → True];

HCompSymb = "H";
DefTensor[HComp[], M4, PrintAs → SymbolBuild[HCompSymb]];
(*A dummy variable which we will use to construct Poisson brackets*)
ClearBuild[];
```

build

ADM projections

build

```
PPerpSymb = "p̂⊥";
DefTensor[PPerp[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[PPerpSymb]];
PParaSymb = "p̂"";
DefTensor[PPara[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[PParaSymb]];
PPerpDefinition = V[-a] V[b];
PPerpActivate = MakeRule[{PPerp[-a, b], Evaluate[PPerpDefinition]},
    MetricOn → All, ContractMetrics → True];
PParaDefinition = G[-a, b] - V[-a] V[b];
PParaActivate = MakeRule[{PPara[-a, b], Evaluate[PParaDefinition]},
    MetricOn → All, ContractMetrics → True];
PADMActivate = Join[PPerpActivate, PParaActivate];
ClearBuild[];
```

build

Automatic rules for converting derivatives to ∇b

build

```
(*Rules for converting all derivatives into
```

```
 derivatives of translational gauge fields by chain rule*)
DefTensor[DVDB[-a, -b, c], M4];
DefTensor[DHDB[-a, b, -c, d], M4];
DefTensor[DJDB[-c, d], M4];
DefTensor[DJiDB[-c, d], M4];
DefTensor[DLapseDB[-c, d], M4];

DVDBDefinition = -V[-b] PPara[i, -a] H[-i, c] /. PADMActivate // ToCanonical;
AutomaticRules[DVDB, MakeRule[{DVDB[-a, -b, c], Evaluate[DVDBDefinition]},
    MetricOn → All, ContractMetrics → True]];
DHDBDefinition = -H[-c, b] H[-a, d] // ToCanonical;
AutomaticRules[DHDB, MakeRule[{DHDB[-a, b, -c, d], Evaluate[DHDBDefinition]},
    MetricOn → All, ContractMetrics → True]];
DJDBDefinition = J[] PPara[-c, e] H[-e, d] /. PADMActivate // ToCanonical;
AutomaticRules[DJDB, MakeRule[{DJDB[-c, d], Evaluate[DJDBDefinition]},
    MetricOn → All, ContractMetrics → True]];
DJiDBDefinition = -Ji[] PPara[-c, e] H[-e, d] /. PADMActivate // ToCanonical;
AutomaticRules[DJiDB, MakeRule[{DJiDB[-c, d], Evaluate[DJiDBDefinition]},
    MetricOn → All, ContractMetrics → True]];
(*DLapseDBDefinition=Lapse[]PPerp[-c,e]H[-e,d]/.PADMActivate//ToCanonical;*)
(*this fixed, could have led to catastrophic
 errors: I even wrote it correctly in the paper but in HiGGS it was clearly
       copied from J rule and never edited..! Finally noticed because
    when I tried to explore theories with Einstein--Hilbert term,
I would occasionally get non-ADM "X" tensor in the velocities.*)
DLapseDBDefinition = Lapse[] V[-c] V[e] H[-e, d] /. PADMActivate // ToCanonical;
(*this ought to be correct*)
AutomaticRules[DLapseDB,
  MakeRule[{DLapseDB[-c, d], Evaluate[DLapseDBDefinition]},
    MetricOn → All, ContractMetrics → True]];

AutomaticRules[V, MakeRule[{CD[-a][V[-j]],
     Evaluate[-V[-i] PPara[-j, k] H[-k, m] CD[-a][B[i, -m]] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[H, MakeRule[{CD[-a][H[-j, n]],
     Evaluate[-H[-i, n] H[-j, m] CD[-a][B[i, -m]]]},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[J, MakeRule[{CD[-a][J[]],
     Evaluate[J[] H[-k, n] PPara[k, -i] CD[-a][B[i, -n]] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[Ji, MakeRule[{CD[-a][Ji[]],
     Evaluate[-Ji[] H[-k, n] PPara[k, -i] CD[-a][B[i, -n]] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[Lapse, MakeRule[{CD[-a][Lapse[]],
```

```
    Evaluate[Lapse[] H[-k, n] PPerp[k, -i] CD[-a][B[i, -n]] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True]];

DeclareOrder[CD[-a][B[i, -m]], 1];
ClearBuild[];
```

build

Nester form rules

build

```
G3HExpand = MakeRule[{G3[n, -m] H[-i, m],
    Evaluate[V[-i] V[j] G3[n, -m] H[-j, m] + PPara[-i, j] H[-j, n] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
HG3BExpand = MakeRule[{H[-a, b] G3[-b, c] B[d, -c],
    Evaluate[PPara[-a, b] PPara[-b, d] + V[-a] V[c] H[-c, e] G3[-e, f] B[d, -f] /.
       PADMActivate // ToCanonical]}, MetricOn → All, ContractMetrics → True];
DefTensor[X[k], M4];
AutomaticRules[X,
  MakeRule[{X[-a] V[a], 1}, MetricOn → All, ContractMetrics → True]];
HG3BExpandLazy = MakeRule[{B[d, -b] G3[b, -a] H[-e, a],
    Evaluate[G[d, -e] - V[-e] X[d] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
UnprocessedX = MakeRule[{X[d], Evaluate[
     V[d] + PPara[d, -c] B[c, -b] G3[b, -e] H[-f, e] V[f] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];(*seems I never used this below,
and I'd like to know why X didn't cause problems
 with previous velocities,
since it commonly cropps up in brackets
 with the Lapse (but not always)*)
(*this still seems a problem -- must check!*)
XToV = MakeRule[{X[d], Evaluate[V[d]]}, MetricOn → All, ContractMetrics → True];
HExpandedDefinition =
  G3[-k, j] H[-i, k] + V[-i] V[k] H[-k, j] - V[-i] G3[-k, j] V[l] H[-l, k];
 (*there was a sign error here, since corrected*)
HExpand = MakeRule[{H[-i, j], Evaluate[HExpandedDefinition]},
   MetricOn → All, ContractMetrics → True];

RiemannCartanExpand =
  MakeRule[{R[a, b, -d, -e], H[-d, i] H[-e, j] (CD[-i][A[a, b, -j]] -
       CD[-j][A[a, b, -i]] + A[a, -k, -i] A[k, b, -j] - A[a, -k, -j] A[k, b, -i])},
   MetricOn → All, ContractMetrics → True];
TorsionExpand = MakeRule[{T[a, -b, -c],
    H[-b, i] H[-c, j] (CD[-i][B[a, -j]] - CD[-j][B[a, -i]] + A[a, -k, -i] B[k, -j] -
       A[a, -k, -j] B[k, -i])}, MetricOn → All, ContractMetrics → True];
```

```
ExpandStrengths = Join[RiemannCartanExpand, TorsionExpand];
ToTorsion =
  MakeRule[{CD[-s][B[a, -r]], Evaluate[Symmetrize[CD[-s][B[a, -r]], {-s, -r}] -
      Antisymmetrize[A[a, -k, -s] B[k, -r], {-s, -r}] + (1/2) B[b, -s]
       B[c, -r] T[a, -b, -c]]}, MetricOn → All, ContractMetrics → True];
ToRiemannCartan = MakeRule[{CD[-s][A[i, j, -r]],
    Evaluate[Symmetrize[CD[-s][A[i, j, -r]], {-s, -r}] -
      Antisymmetrize[A[i, -m, -s] A[m, j, -r], {-s, -r}] +
      (1/2) B[k, -s] B[l, -r] R[i, j, -k, -l]]},
   MetricOn → All, ContractMetrics → True];
ToStrengths = Join[ToTorsion, ToRiemannCartan];

(*would be good to put parallel momenta up here also*)

(*Defining parallel field strengths, i.e. the canonical parts*)
TPpSymb = "𝒯";
DefTensor[TP[-a, -b, -c], M4, Antisymmetric[{-b, -c}],
  PrintAs → SymbolBuild[TPpSymb], OrthogonalTo → {V[b], V[c]}];
DeclareOrder[TP[-a, -b, -c], 1];
RPpSymb = "ℛ";
DefTensor[RP[-a, -b, -c, -d], M4,
  {Antisymmetric[{-a, -b}], Antisymmetric[{-c, -d}]},
  PrintAs → SymbolBuild[RPpSymb], OrthogonalTo → {V[c], V[d]}];
DeclareOrder[RP[-a, -b, -c, -d], 1];
TPToT = MakeRule[{TP[-a, -b, -c], PPara[-b, e] PPara[-c, f] T[-a, -e, -f]},
   MetricOn → All, ContractMetrics → True];
RPToR = MakeRule[{RP[-a, -b, -c, -d], PPara[-c, e] PPara[-d, f] R[-a, -b, -e, -f]},
   MetricOn → All, ContractMetrics → True];
StrengthPToStrength = Join[TPToT, RPToR];

(*Defining parallel field strength multipliers*)
RLambdaPpSymb = "λ_ℛ";
DefTensor[RLambdaP[-a, -b, -c, -d], M4,
  {Antisymmetric[{-a, -b}], Antisymmetric[{-c, -d}]},
  PrintAs → SymbolBuild[RLambdaPpSymb], OrthogonalTo → {V[c], V[d]}];
DeclareOrder[RLambdaP[-a, -b, -c, -d], 1];
TLambdaPpSymb = "λ_𝒯";
DefTensor[TLambdaP[-a, -c, -d], M4, Antisymmetric[{-c, -d}],
  PrintAs → SymbolBuild[TLambdaPpSymb], OrthogonalTo → {V[c], V[d]}];
DeclareOrder[TLambdaP[-a, -c, -d], 1];
TLambdaPToTLambda =
  MakeRule[{TLambdaP[-a, -b, -c], PPara[-b, e] PPara[-c, f] TLambda[-a, -e, -f]},
   MetricOn → All, ContractMetrics → True];
```

```
RLambdaPToRLambda = MakeRule[{RLambdaP[-a, -b, -c, -d], PPara[-c, e] PPara[-d, f]
      RLambda[-a, -b, -e, -f]}, MetricOn → All, ContractMetrics → True];
StrengthLambdaPToStrengthLambda = Join[RLambdaPToRLambda, TLambdaPToTLambda];

(*Defining perpendicular field strengths, i.e. the non-canonical parts*)
TPerppSymb = "𝒯⊥";
DefTensor[TPerp[-a, -b], M4,
  PrintAs → SymbolBuild[TPerppSymb], OrthogonalTo → {V[b]}];
DeclareOrder[TPerp[-a, -b], 1];
RPerppSymb = "ℛ⊥";
DefTensor[RPerp[-a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPerppSymb], OrthogonalTo → {V[c]}];
DeclareOrder[RPerp[-a, -b, -c], 1];
TPerpToT = MakeRule[{TPerp[-a, -b], PPara[-b, f] V[g] T[-a, -f, -g]},
    MetricOn → All, ContractMetrics → True];
RPerpToR = MakeRule[{RPerp[-a, -b, -c], PPara[-c, e] V[f] R[-a, -b, -e, -f]},
    MetricOn → All, ContractMetrics → True];
StrengthPerpToStrength = Join[TPerpToT, RPerpToR];

(*Defining perpendicular field strength multipliers*)
TLambdaPerppSymb = "λ𝒯⊥";
DefTensor[TLambdaPerp[-a, -b], M4,
  PrintAs → SymbolBuild[TLambdaPerppSymb], OrthogonalTo → {V[b]}];
DeclareOrder[TLambdaPerp[-a, -b], 1];
RLambdaPerppSymb = "λℛ⊥";
DefTensor[RLambdaPerp[-a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPerppSymb], OrthogonalTo → {V[c]}];
DeclareOrder[RLambdaPerp[-a, -b, -c], 1];
TLambdaPerpToTLambda =
  MakeRule[{TLambdaPerp[-a, -b], PPara[-b, f] V[g] TLambda[-a, -f, -g]},
    MetricOn → All, ContractMetrics → True];
RLambdaPerpToRLambda = MakeRule[{RLambdaPerp[-a, -b, -c], PPara[-c, e] V[f]
      RLambda[-a, -b, -e, -f]}, MetricOn → All, ContractMetrics → True];
StrengthLambdaPerpToStrengthLambda = Join[RLambdaPerpToRLambda,
    TLambdaPerpToTLambda];

RDecomposeDefinition =
  RP[-a, -b, -c, -d] + 2 Antisymmetrize[V[-d] RPerp[-a, -b, -c], {-c, -d}] /.
        ExpandStrengths /. PADMActivate // ToCanonical //
      CollectTensors // ScreenDollarIndices // CollectTensors;
TDecomposeDefinition = TP[-a, -c, -d] + 2 Antisymmetrize[V[-d] TPerp[-a, -c],
            {-c, -d}] /. ExpandStrengths /. PADMActivate // ToCanonical //
      CollectTensors // ScreenDollarIndices // CollectTensors;
RDecompose = MakeRule[{R[-a, -b, -c, -d], Evaluate[RDecomposeDefinition]},
```

```
    MetricOn → All, ContractMetrics → True];
TDecompose = MakeRule[{T[-a, -c, -d], Evaluate[TDecomposeDefinition]},
    MetricOn → All, ContractMetrics → True];
StrengthDecompose = Join[RDecompose, TDecompose];

RLambdaDecomposeDefinition =
  RLambdaP[-a, -b, -c, -d] + 2 Antisymmetrize[V[-d] RLambdaPerp[-a, -b, -c],
            {-c, -d}] /. ExpandStrengths /. PADMActivate // ToCanonical //
      CollectTensors // ScreenDollarIndices // CollectTensors;
TLambdaDecomposeDefinition =
  TLambdaP[-a, -c, -d] + 2 Antisymmetrize[V[-d] TLambdaPerp[-a, -c], {-c, -d}] /.
        ExpandStrengths /. PADMActivate // ToCanonical //
      CollectTensors // ScreenDollarIndices // CollectTensors;
RLambdaDecompose = MakeRule[{RLambda[-a, -b, -c, -d], Evaluate[
      RLambdaDecomposeDefinition]}, MetricOn → All, ContractMetrics → True];
TLambdaDecompose = MakeRule[{TLambda[-a, -c, -d], Evaluate[
      TLambdaDecomposeDefinition]}, MetricOn → All, ContractMetrics → True];
StrengthLambdaDecompose = Join[RLambdaDecompose, TLambdaDecompose];

(*
TPToT=MakeRule[{TP[-a,-b,-c],T[-a,-i,-j]PPara[i,-b]PPara[j,-c]},
  MetricOn→All,ContractMetrics→True];
RPToR=MakeRule[{RP[-a,-b,-c,-d],R[-a,-b,-i,-j]PPara[i,-c]PPara[j,-d]},
  MetricOn→All,ContractMetrics→True];
StrengthPToStrength=Join[TPToT,RPToR];
*)(*scheduled for decomission*)

CDBCommute = MakeRule[{CD[-s][B[a, -r]],
    Evaluate[CD[-r][B[a, -s]] - 2 Antisymmetrize[A[a, -k, -s] B[k, -r], {-s, -r}] +
      B[b, -s] B[c, -r] T[a, -b, -c]]}, MetricOn → All, ContractMetrics → True];
 (*Might want to write an equivalent version for Riemann
 Cartan curvature*)

DefTensor[DV[-a, -j], M4, OrthogonalTo → {V[j]},
  PrintAs → SymbolBuild[VSymb, "Derivative" -> 1]];
(*DeclareOrder[DV[-a,-j],1];*)
DefTensor[DJ[-a], M4, PrintAs -> SymbolBuild[JSymb, "Derivative" -> 1]];
(*DeclareOrder[DJ[-a],1];*)

G3VCDBToG3DV = MakeRule[{G3[-l, n] V[-k] CD[-m][B[k, -n]],
    -G3[-l, n] B[j, -n] A[k, -j, -m] V[-k] - G3[-l, n] B[j, -n] DV[-m, -j]},
   MetricOn → All, ContractMetrics → True];

G3HCDBToDJ = MakeRule[{G3[n, -s] H[-k, s] CD[-m][B[k, -n]], Ji[] DJ[-m] -
```

```
      V[k] H[-k, a] G3[-a, b] (B[j, -b] DV[-m, -j] + V[-l] A[l, -j, -m] B[j, -b])},
    MetricOn → All, ContractMetrics → True];

(*we want to be able to reverse the v and J derivatives also,
this below just some syntax for that time*)
(*
G3DVToG3VCDB=MakeRule[{G3[-l,n]V[-k]CD[-m][B[k,-n]],
   -G3[-l,n]B[j,-n]A[k,-j,-m]V[-k]-G3[-l,n]B[j,-n]DV[-m,-j]},
  MetricOn→All,ContractMetrics→True];
(*the rules below should of course be generalised beyond simply the
 momenta -- these below now generalise to the field strengths*)
DTP0mDeactivate=MakeRule[{DTP0m[-z],CD[-z][TP0m[]]},
  MetricOn→All,ContractMetrics→True];
DTP1pDeactivate=MakeRule[{DTP1p[-z,-a,-b],
   CD[-z][TP1p[-a,-b]]-A[i,-a,-z]TP1p[-i,-b]-A[i,-b,-z]TP1p[-a,-i]},
  MetricOn→All,ContractMetrics→True];
DTP1mDeactivate=MakeRule[{DTP1m[-z,-a],CD[-z][TP1m[-a]]-A[i,-a,-z]TP1m[-i]},
  MetricOn→All,ContractMetrics→True];
DTP2mDeactivate=MakeRule[{DTP2m[-z,-a,-b,-c],
   CD[-z][TP2m[-a,-b,-c]]-A[i,-a,-z]TP2m[-i,-b,-c]-A[i,-b,-z]TP2m[-a,-i,-c]-
    A[i,-c,-z]TP2m[-a,-b,-i]},MetricOn→All,ContractMetrics→True];
DRP0pDeactivate=MakeRule[{DRP0p[-z],CD[-z][RP0p[]]},
  MetricOn→All,ContractMetrics→True];
DRP0mDeactivate=MakeRule[{DRP0m[-z],CD[-z][RP0m[]]},
  MetricOn→All,ContractMetrics→True];
DRP1pDeactivate=MakeRule[{DRP1p[-z,-a,-b],
   CD[-z][RP1p[-a,-b]]-A[i,-a,-z]RP1p[-i,-b]-A[i,-b,-z]RP1p[-a,-i]},
  MetricOn→All,ContractMetrics→True];
DRP1mDeactivate=MakeRule[{DRP1m[-z,-a],CD[-z][RP1m[-a]]-A[i,-a,-z]RP1m[-i]},
  MetricOn→All,ContractMetrics→True];
DRP2pDeactivate=MakeRule[{DRP2p[-z,-a,-b],
   CD[-z][RP2p[-a,-b]]-A[i,-a,-z]RP2p[-i,-b]-A[i,-b,-z]RP2p[-a,-i]},
  MetricOn→All,ContractMetrics→True];
DRP2mDeactivate=MakeRule[{DRP2m[-z,-a,-b,-c],
   CD[-z][RP2m[-a,-b,-c]]-A[i,-a,-z]RP2m[-i,-b,-c]-A[i,-b,-z]RP2m[-a,-i,-c]-
    A[i,-c,-z]RP2m[-a,-b,-i]},MetricOn→All,ContractMetrics→True];
DRPDeactivate=Join[DTP0mDeactivate,DTP1pDeactivate,DTP1mDeactivate,
  DTP2mDeactivate,DRP0pDeactivate,DRP0mDeactivate,DRP1pDeactivate,
  DRP1mDeactivate,DRP2pDeactivate,DRP2mDeactivate];
*)

DefTensor[DpJ[-z], M4,
  PrintAs → SymbolBuild[JSymb, "Derivative" -> 2], OrthogonalTo → {V[z]}];
```

```
DeclareOrder[DpJ[-z], 1];
DeclareOrder[DJ[-z], 1,
  "approximation" -> B[w, -z] DpJ[-w] + V[-v] B[v, -z] V[u] H[-u, w] DJ[-w]];
DpJActivate = MakeRule[{G3[-y, z] DJ[-z], G3[-y, z] B[x, -z] DpJ[-x]},
   MetricOn → All, ContractMetrics → True];
DefTensor[DpV[-z, -a], M4, PrintAs → SymbolBuild[VSymb, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a]}];
DeclareOrder[DpV[-z, -a], 1];
DeclareOrder[DV[-z, -a], 1,
  "approximation" -> B[w, -z] DpV[-w, -a] + V[-v] B[v, -z] V[u] H[-u, w] DV[-w, -a]];
DpVActivate = MakeRule[{G3[-y, z] DV[-z, -a],
    Evaluate[G3[-y, z] B[x, -z] DpV[-x, -a] + (G[-a, i] - PPara[-a, i]) G3[-y, z]
        DV[-z, -i] /. PADMActivate]}, MetricOn → All, ContractMetrics → True];

DpVExpand = MakeRule[{DpV[-m, -j],
    Evaluate[Symmetrize[DpV[-m, -j], {-m, -j}] - (1 / 2) V[-i] TP[i, -m, -j] /.
      PADMActivate]}, MetricOn → All, ContractMetrics → True];

AVepsilonGToAVEps =
  MakeRule[{A[-e, d, -f] epsilonG[-d, -a, -b, -c] V[e], A[-e, d, -f] V[e]
     (V[-a] Eps[-d, -b, -c] - V[-b] Eps[-d, -a, -c] + V[-c] Eps[-d, -a, -b])},
   MetricOn → All, ContractMetrics → True];
HEpsToHG3Eps = MakeRule[{Eps[-a, -b, c] H[-c, e], Eps[-a, -b, c] H[-c, f] G3[e, -f]},
   MetricOn → All, ContractMetrics → True];
epsilonGToEpsV = MakeRule[{epsilonG[-a, -b, -c, -d],
    -V[-a] Eps[-b, -c, -d] + V[-b] Eps[-a, -c, -d] - V[-c] Eps[-a, -b, -d] +
     V[-d] Eps[-a, -b, -c]}, MetricOn → All, ContractMetrics → True];
DefTensor[Q[-a, -b], M4, OrthogonalTo → {V[a], V[b]}];
DeclareOrder[Q[-a, -b], 1];
AHEpsExpand = MakeRule[{A[-i, j, -m] Eps[-j, -p, -q] H[-k, m],
    Evaluate[Eps[-i, j, -z] Q[z, -k] Eps[-j, -p, -q] +
       PPerp[-i, a] PPara[-k, b] A[-a, j, -m] Eps[-j, -p, -q] H[-b, m] +
       PPara[-i, a] PPerp[-k, b] A[-a, j, -m] Eps[-j, -p, -q] H[-b, m] +
       PPerp[-i, a] PPerp[-k, b] A[-a, j, -m] Eps[-j, -p, -q] H[-b, m] /.
      PADMActivate]}, MetricOn → All, ContractMetrics → True];
EpsEpsExpand = MakeRule[{Eps[i, a, b] Eps[-i, -c, -d], Evaluate[
     PPara[a, -c] PPara[b, -d] - PPara[a, -d] PPara[b, -c] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];

DefTensor[CDAInert[-a, -b, -c, -d], M4, Antisymmetric[{-b, -c}]];
DeclareOrder[CDAInert[-a, -b, -c, -d], 1];
CDAToCDAInert = MakeRule[{CD[-a][A[-b, -c, -d]], CDAInert[-a, -b, -c, -d]},
   MetricOn → All, ContractMetrics → True];
```

```
CDAInertToCDA = MakeRule[{CDAInert[-a, -b, -c, -d], CD[-a][A[-b, -c, -d]]},
    MetricOn → All, ContractMetrics → True];
AExpandedDefinition = PPara[-a, i] PPara[-b, j] A[-i, -j, -c] +
     PPerp[-a, i] PPara[-b, j] A[-i, -j, -c] -
     PPerp[-b, i] PPara[-a, j] A[-i, -j, -c] /. PADMActivate;
CDAExpandedDefinition = PPara[-a, i] PPara[-b, j] CDAInert[-k, -i, -j, -c] +
     PPerp[-a, i] PPara[-b, j] CDAInert[-k, -i, -j, -c] -
     PPerp[-b, i] PPara[-a, j] CDAInert[-k, -i, -j, -c] /. PADMActivate;
AToAExpanded = MakeRule[{A[-a, -b, -c], Evaluate[AExpandedDefinition]},
    MetricOn → All, ContractMetrics → True];
CDAToCDAExpanded = MakeRule[{CDAInert[-k, -a, -b, -c],
     Evaluate[CDAExpandedDefinition]}, MetricOn → All, ContractMetrics → True];
AExpand = Join[AToAExpanded, CDAToCDAExpanded];
HVCDADefinition = H[-i, m] V[b] CDAInert[-k, i, -b, -c] /. PADMActivate;
HVADefinition = H[-i, m] V[b] A[i, -b, -c] /. PADMActivate;
HG3VCDAToHVCDA = MakeRule[{H[-i, j] G3[-j, m] V[b] CDAInert[-k, i, -b, -c],
     Evaluate[HVCDADefinition]}, MetricOn → All, ContractMetrics → True];
HG3VAToHVA = MakeRule[{H[-i, j] G3[-j, m] V[b] A[i, -b, -c],
     Evaluate[HVADefinition]}, MetricOn → All, ContractMetrics → True];
ClearBuild[];
```

build

Basic form covariance check on $\mathbb{R}^{1,3} \rtimes SO^{+}(1,3)$

build

```
(*Tools for covariance check,
which is useful for emergencies but otherwise commented out*)
(*
DefTensor[CCoord[-a,-b,c],M4,Symmetric[{-a,-b}]]
 DefTensor[FLorentz[-a,-b,-c],M4,PrintAs→"FAILΛ"]
 DefTensor[FCoord[-a,-b,-c],M4,PrintAs→"FAILx"]
 DefTensor[Lorentz[a,-b],M4,PrintAs→"Λ"]
 AutomaticRules[Lorentz,MakeRule[
   {Lorentz[-a,-b]Lorentz[a,-c],G[-b,-c]},MetricOn→All,ContractMetrics→True]];
AutomaticRules[Lorentz,MakeRule[{Lorentz[-b,-a]Lorentz[-c,a],G[-c,-b]},
  MetricOn→All,ContractMetrics→True]];
DefTensor[Coord[a,-b],M4,PrintAs→"x"]
 AutomaticRules[Coord,MakeRule[
   {Coord[-a,-b]Coord[a,-c],G[-b,-c]},MetricOn→All,ContractMetrics→True]];
AutomaticRules[Coord,MakeRule[{Coord[-b,-a]Coord[-c,a],G[-c,-b]},
  MetricOn→All,ContractMetrics→True]];

DefTensor[CDBInert[-a,b,-c],M4];
```

```
DefTensor[CDAInert[-a,b,c,-d],M4,Antisymmetric[{b,c}]];
ToCDBInert=MakeRule[{CD[-a][B[b,-c]],CDBInert[-a,b,-c]},
  MetricOn→All,ContractMetrics→True];
ToCDAInert=MakeRule[{CD[-a][A[b,c,-d]],CDAInert[-a,b,c,-d]},
  MetricOn→All,ContractMetrics→True];
ToCDInert=Join[ToCDBInert,ToCDAInert];

GaugeB=MakeRule[{B[b,-c],Lorentz[b,-j]Coord[-c,k]B[j,-k]},
  MetricOn→All,ContractMetrics→True];
GaugeH=MakeRule[{H[-b,c],Lorentz[-b,j]Coord[c,-k]H[-j,k]},
  MetricOn→All,ContractMetrics→True];
GaugeV=MakeRule[{V[b],Lorentz[b,-j]V[j]},MetricOn→All,ContractMetrics→True];
GaugeA=
 MakeRule[{A[b,c,-d],Lorentz[b,-j]Lorentz[c,-k]Coord[-d,l]A[j,k,-l]-Lorentz[c,j]
     Coord[-d,l]CD[-l][Lorentz[b,-j]]},MetricOn→All,ContractMetrics→True];
GaugeMe=Join[GaugeB,GaugeH,GaugeV,GaugeA];

GaugeCDA=MakeRule[{CDAInert[-a,b,c,-d],
   Coord[-a,i]CD[-i][Lorentz[b,-j]Lorentz[c,-k]Coord[-d,l]A[j,k,-l]-
      Lorentz[c,j]Coord[-d,l]CD[-l][Lorentz[b,-j]]]},
  MetricOn→All,ContractMetrics→True];
GaugeCDB=MakeRule[{CDBInert[-a,b,-c],
   Coord[-a,i]CD[-i][Lorentz[b,-j]Coord[-c,k]B[j,-k]]},
  MetricOn→All,ContractMetrics→True];
GaugeMeInert=Join[GaugeCDB,GaugeCDA];

ToCCoord=MakeRule[{CD[-a][Coord[-b,c]],Coord[s,-a]CCoord[-s,-b,c]},
  MetricOn→All,ContractMetrics→True];

DefTensor[Toten[b,-c,d],M4,Symmetric[{b,d}]];
(*SwitchMe=MakeRule[{Lorentz[a,-b]CD[-c][Lorentz[-a,-d]],
    Toten[-b,-c,-d]-Lorentz[-a,-d]CD[-c][Lorentz[a,-b]]},
   MetricOn→All,ContractMetrics→True];*)
(*SwitchMe=MakeRule[{Lorentz[a,-b]CD[-c][Lorentz[-a,-d]],-Lorentz[-a,-d]
     CD[-c][Lorentz[a,-b]]},MetricOn→All,ContractMetrics→True];*)
CommuteMe=MakeRule[{Lorentz[a,-b]CD[-c][Lorentz[-a,-d]],
   Evaluate[Antisymmetrize[Lorentz[a,-b]CD[-c][Lorentz[-a,-d]],{-b,-d}]]},
  MetricOn→All,ContractMetrics→True];
SwitchMe=MakeRule[{Lorentz[a,-b]CD[-c][Lorentz[-a,-d]],
   -Lorentz[-a,-d]CD[-c][Lorentz[a,-b]]},MetricOn→All,ContractMetrics→True];

FlagLorentz=MakeRule[{CD[-a][Lorentz[-b,-c]],FLorentz[-a,-b,-c]},
  MetricOn→All,ContractMetrics→True];
FlagCoord=MakeRule[{CD[-a][Coord[-b,-c]],FCoord[-a,-b,-c]},
```

```
  MetricOn→All,ContractMetrics→True];
FlagBroken=Join[FlagLorentz,FlagCoord];

ManRemoveG3=MakeRule[{G3[-b,c],G[-b,c]},MetricOn→All,ContractMetrics→True];

GaugeShift[x_]:=Module[{exp},
  exp=x;
  HiGGSPrint[Style["Manually removing G3",Blue,10]];
  exp=exp/.ManRemoveG3;
  HiGGSPrint[Style["simplifying",Blue,10]];
  exp=exp//ToCanonical//ScreenDollarIndices//ContractMetric//CollectTensors;
  HiGGSPrint[Style["Converting to inert",Blue,10]];
  exp=exp/.ToCDInert;
  HiGGSPrint[Style["simplifying",Blue,10]];
  exp=exp//ToCanonical//ScreenDollarIndices//ContractMetric//CollectTensors;
  HiGGSPrint[Style["transforming gauge",Blue,10]];
  exp=exp/.GaugeMe;
  HiGGSPrint[Style["simplifying",Blue,10]];
  exp=exp//ToCanonical//ScreenDollarIndices//ContractMetric//CollectTensors;
  HiGGSPrint[Style["transforming CD gauge",Blue,10]];
  exp=exp/.GaugeMeInert;
  HiGGSPrint[Style["simplifying",Blue,10]];
  exp=exp//ToCanonical//ScreenDollarIndices//ContractMetric//CollectTensors;
  HiGGSPrint[Style["transforming to coordinate Hessian",Blue,10]];
  exp=exp/.ToCCoord;
  HiGGSPrint[Style["simplifying",Blue,10]];
  exp=exp//ToCanonical//ScreenDollarIndices//ContractMetric//CollectTensors;
  HiGGSPrint[Style["removing scalar",Blue,10]];
  exp=exp//NoScalar;
  HiGGSPrint[Style["commuting Lorentz gradients",Blue,10]];
  exp=exp/.SwitchMe;
  HiGGSPrint[Style["simplifying",Blue,10]];
  exp=exp//ToCanonical//ScreenDollarIndices//ContractMetric//CollectTensors;
  HiGGSPrint[Style["removing scalar",Blue,10]];
  exp=exp//NoScalar;
  HiGGSPrint[Style["commuting Lorentz gradients",Blue,10]];
  exp=exp/.CommuteMe;
  HiGGSPrint[Style["simplifying",Blue,10]];
  exp=exp//ToCanonical//ScreenDollarIndices//ContractMetric//CollectTensors;
  HiGGSPrint[Style["removing scalar",Blue,10]];
  exp=exp//NoScalar;
  HiGGSPrint[Style["commuting Lorentz gradients",Blue,10]];
  exp=exp/.SwitchMe;
  HiGGSPrint[Style["simplifying",Blue,10]];
```

```
  exp=exp//ToCanonical//ScreenDollarIndices//ContractMetric//CollectTensors;
  HiGGSPrint[Style["raising flags",Blue,10]];
  exp=exp/.FlagBroken;
  exp];
*)
```

build

Irreducible decomposition of the fields using SO(3)

build

Human-readable projections {Aᵖ̌}, {Eᵖ̌}

build

```
DefTensor[PThreePara[-a, -b, -c, d, e, f],
  M4, {Antisymmetric[{-a, -b}], Antisymmetric[{d, e}]}];
PThreeParaDefinition =
  Antisymmetrize [Antisymmetrize[PPara[-a, d] PPara[-b, e] PPara[-c, f], {-a, -b}],
     {d, e}] /. PADMActivate // ToCanonical;
PThreeParaActivate = MakeRule[{PThreePara[-a, -b, -c, d, e, f],
    Evaluate[PThreeParaDefinition]}, MetricOn → All, ContractMetrics → True];
DefTensor[PThreePerp[-a, -b, -c, d, e, f], M4,
  {Antisymmetric[{-a, -b}], Antisymmetric[{d, e}]}];
PThreePerpDefinition = Antisymmetrize[Antisymmetrize[
      (PPara[-a, d] PPerp[-b, e] + PPerp[-a, d] PPara[-b, e]) PPara[-c, f],
      {-a, -b}], {d, e}] /. PADMActivate // ToCanonical;
PPerpActivate = MakeRule[{PThreePerp[-a, -b, -c, d, e, f],
    Evaluate[PThreePerpDefinition]}, MetricOn → All, ContractMetrics → True];

DefTensor[PAPerp[-a, -b, d, e, f], M4];
DefTensor[PAPara[-a, -b, -c, d, e, f], M4];
DefTensor[PBPerp[-a, d, e], M4];
DefTensor[PBPara[-a, -b, d, e], M4];

PAPerpDefinition = V[d] PPara[-a, e] G[-b, f] /. PADMActivate // ToCanonical;
PAPerpActivate = MakeRule[{PAPerp[-a, -b, d, e, f], Evaluate[PAPerpDefinition]},
   MetricOn → All, ContractMetrics → True];
PAParaDefinition = PPara[-a, d] PPara[-b, e] G[-c, f] /. PADMActivate //
   ToCanonical;
PAParaActivate = MakeRule[{PAPara[-a, -b, -c, d, e, f],
    Evaluate[PAParaDefinition]}, MetricOn → All, ContractMetrics → True];

PBPerpDefinition = V[d] G[-a, e] /. PADMActivate // ToCanonical;
```

```
PBPerpActivate = MakeRule[{PBPerp[-a, d, e], Evaluate[PBPerpDefinition]},
    MetricOn → All, ContractMetrics → True];
PBParaDefinition = PPara[-a, d] G[-b, e] /. PADMActivate // ToCanonical;
PBParaActivate = MakeRule[{PBPara[-a, -b, d, e], Evaluate[PBParaDefinition]},
    MetricOn → All, ContractMetrics → True];

PADMPiActivate =
  Join[PAPerpActivate, PAParaActivate, PBPerpActivate, PBParaActivate];

PASymb = "P̌ₐ";
DefTensor[PA0p[c, d], M4, PrintAs → SymbolBuild[PASymb, Spin0p]];
DefTensor[PA1p[-a, -b, c, d], M4, PrintAs -> SymbolBuild[PASymb, Spin1p]];
DefTensor[PA2p[-a, -b, c, d], M4, PrintAs -> SymbolBuild[PASymb, Spin2p]];

PA0pDefinition = PPara[c, -k] PPara[d, -l] G[k, l] /. PADMActivate // ToCanonical;
PA1pDefinition = PPara[-a, i] PPara[-b, j] PPara[c, -k] PPara[d, -l]
     Antisymmetrize[G[-i, k] G[-j, l], {-i, -j}] /. PADMActivate // ToCanonical;
PA2pDefinition = PPara[-a, i] PPara[-b, j] PPara[c, -k] PPara[d, -l]
     (Symmetrize[G[-i, k] G[-j, l], {-i, -j}] - (1/3) G[-i, -j] G[k, l]) /.
    PADMActivate // ToCanonical;

DefTensor[PA0m[d, e, f], M4, PrintAs -> SymbolBuild[PASymb, Spin0m]];
DefTensor[PA1m[-a, d, e, f], M4, PrintAs -> SymbolBuild[PASymb, Spin1m]];
DefTensor[PA2m[-a, -b, -c, d, e, f], M4, PrintAs -> SymbolBuild[PASymb, Spin2m]];

PA0mDefinition = PPara[-i, d] PPara[-j, e] PPara[-k, f] epsilonG[i, j, k, g] V[-g] /.
    PADMActivate // ToCanonical;
PA1mDefinition = PPara[-i, d] PPara[-j, f] PPara[k, -a] PPara[-l, e]
     G[i, j] G[-k, l] /. PADMActivate // ToCanonical;
PA2mDefinition = PPara[-a, i] PPara[-b, j] PPara[-c, k] PPara[d, -l]
     PPara[e, -n] PPara[f, -m] (3/4) ((1/3) (2 G[-i, l] G[-j, n] G[-k, m] -
          G[-j, l] G[-k, n] G[-i, m] - G[-k, l] G[-i, n] G[-j, m]) - Antisymmetrize[
        G[-i, -k] G[-j, n] G[l, m], {-i, -j}]) /. PADMActivate // ToCanonical;

PA0pActivate = MakeRule[{PA0p[c, d], Evaluate[PA0pDefinition]},
    MetricOn → All, ContractMetrics → True];
PA1pActivate = MakeRule[{PA1p[-a, -b, c, d], Evaluate[PA1pDefinition]},
    MetricOn → All, ContractMetrics → True];
PA2pActivate = MakeRule[{PA2p[-a, -b, c, d], Evaluate[PA2pDefinition]},
    MetricOn → All, ContractMetrics → True];
PA0mActivate = MakeRule[{PA0m[d, e, f], Evaluate[PA0mDefinition]},
    MetricOn → All, ContractMetrics → True];
PA1mActivate = MakeRule[{PA1m[-a, d, e, f], Evaluate[PA1mDefinition]},
```

```
    MetricOn → All, ContractMetrics → True];
PA2mActivate = MakeRule[{PA2m[-a, -b, -c, d, e, f], Evaluate[PA2mDefinition]},
    MetricOn → All, ContractMetrics → True];

PBSymb = "P̌ᵦ";
DefTensor[PB0p[c, d], M4, PrintAs -> SymbolBuild[PBSymb, Spin0p]];
DefTensor[PB1p[-a, -b, c, d], M4, PrintAs -> SymbolBuild[PBSymb, Spin1p]];
DefTensor[PB2p[-a, -b, c, d], M4, PrintAs -> SymbolBuild[PBSymb, Spin2p]];
DefTensor[PB1m[-a, d], M4, PrintAs -> SymbolBuild[PBSymb, Spin1m]];

PB0pDefinition = PPara[c, -k] PPara[d, -l] G[k, l] /. PADMActivate // ToCanonical;
PB1pDefinition = PPara[-a, i] PPara[-b, j] PPara[c, -k] PPara[d, -l]
      Antisymmetrize[G[-i, k] G[-j, l], {-i, -j}] /. PADMActivate // ToCanonical;
PB2pDefinition = PPara[-a, i] PPara[-b, j] PPara[c, -k] PPara[d, -l]
      (Symmetrize[G[-i, k] G[-j, l], {-i, -j}] - (1/3) G[-i, -j] G[k, l]) /.
     PADMActivate // ToCanonical;
PB1mDefinition = PPara[d, -j] PPara[-a, i] G[-i, j] /. PADMActivate // ToCanonical;

PB0pActivate = MakeRule[{PB0p[c, d], Evaluate[PB0pDefinition]},
    MetricOn → All, ContractMetrics → True];
PB1pActivate = MakeRule[{PB1p[-a, -b, c, d], Evaluate[PB1pDefinition]},
    MetricOn → All, ContractMetrics → True];
PB2pActivate = MakeRule[{PB2p[-a, -b, c, d], Evaluate[PB2pDefinition]},
    MetricOn → All, ContractMetrics → True];
PB1mActivate = MakeRule[{PB1m[-a, d], Evaluate[PB1mDefinition]},
    MetricOn → All, ContractMetrics → True];

PO3PiActivate =
  Join[PA0pActivate, PA1pActivate, PA2pActivate, PA0mActivate, PA1mActivate,
    PA2mActivate, PB0pActivate, PB1pActivate, PB2pActivate, PB1mActivate];

APiToAPiP = MakeRule[{APi[-i, -j, k] G3[-k, a] B[l, -a], APiP[-i, -j, l]},
    MetricOn → All, ContractMetrics → True];
BPiToBPiP = MakeRule[{BPi[-i, k] G3[-k, a] B[l, -a], BPiP[-i, l]},
    MetricOn → All, ContractMetrics → True];
PiToPiP = Join[APiToAPiP, BPiToBPiP];
CDAPiToCDAPiP = MakeRule[{CD[-z][APi[-i, -j, k]] G3[-k, a] B[l, -a],
     CD[-z][APiP[-i, -j, l]] - APi[-i, -j, k] G3[-k, a] CD[-z][B[l, -a]]},
    MetricOn → All, ContractMetrics → True];
CDBPiToCDBPiP = MakeRule[{CD[-z][BPi[-i, k]] G3[-k, a] B[l, -a],
     CD[-z][BPiP[-i, l]] - BPi[-i, k] G3[-k, a] CD[-z][B[l, -a]]},
    MetricOn → All, ContractMetrics → True];
CDPiToCDPiP = Join[CDAPiToCDAPiP, CDBPiToCDBPiP];
APiToAPiPHard = MakeRule[{APi[-i, -j, k] G3[-k, a],
```

```
    Evaluate[APiP[-i, -j, l] PPara[-l, s] H[-s, f] G3[-f, a] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
BPiToBPiPHard = MakeRule[{BPi[-i, k] G3[-k, a],
    Evaluate[BPiP[-i, l] PPara[-l, s] H[-s, f] G3[-f, a] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
PiToPiPHard = Join[APiToAPiPHard, BPiToBPiPHard];
(*PADMActivate added above two lines on 14/04*)
CDAPiToCDAPiPHard = MakeRule[{CD[-z][APi[-i, -j, k]] G3[-k, a],
    Evaluate[CD[-z][APiP[-i, -j, l]] PPara[-l, s] H[-s, f] G3[-f, a] +
       APiP[-i, -j, l] CD[-z][PPara[-l, s] H[-s, f] G3[-f, a]] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
CDBPiToCDBPiPHard = MakeRule[{CD[-z][BPi[-i, k]] G3[-k, a],
    Evaluate[CD[-z][BPiP[-i, l]] PPara[-l, s] H[-s, f] G3[-f, a] +
       BPiP[-i, l] CD[-z][PPara[-l, s] H[-s, f] G3[-f, a]] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
CDPiToCDPiPHard = Join[CDAPiToCDAPiPHard, CDBPiToCDBPiPHard];
APiPToAPi = MakeRule[{APiP[-i, -j, l], APi[-i, -j, k] G3[-k, a] B[l, -a]},
   MetricOn → All, ContractMetrics → True];
BPiPToBPi = MakeRule[{BPiP[-i, l], BPi[-i, k] G3[-k, a] B[l, -a]},
   MetricOn → All, ContractMetrics → True];
PiPToPi = Join[APiPToAPi, BPiPToBPi];

ActivateGeneralO3Projections[expr_] := Module[{exp, kern}, exp = Evaluate[expr];
   exp = exp // ToCanonical;
   exp = exp /. PActivate;
   exp = exp // ToCanonical;
   exp = exp /. PADMActivate;
   exp = exp // ToCanonical;
   exp = exp /. PADMPiActivate;
   exp = exp // ToCanonical;
   exp = exp /. PO3PiActivate;
   exp = exp // ToCanonical;
   exp = exp /. HG3BExpandLazy;
   exp = exp // ContractMetric;
   exp = exp // ToCanonical;
   exp = exp // CollectTensors; exp];
ClearBuild[];
```

build

Complete projections $\{^{A}\hat{P}\}$, $\{^{E}\hat{P}\}$

build

In[ ]:=
```
PBTSymb = "P̂_b";
DefTensor[PB0pT[-n, -m, a, c], M4, PrintAs -> SymbolBuild[PBTSymb, Spin0p]];
DefTensor[PB1pT[-n, -m, a, c], M4, PrintAs -> SymbolBuild[PBTSymb, Spin1p]];
DefTensor[PB2pT[-n, -m, a, c], M4, PrintAs -> SymbolBuild[PBTSymb, Spin2p]];
DefTensor[PB1mT[-n, -m, a, c], M4, PrintAs -> SymbolBuild[PBTSymb, Spin1m]];

PATSymb = "P̂_A";
DefTensor[PA0pT[-n, -m, -o, a, b, c],
  M4, PrintAs -> SymbolBuild[PATSymb, Spin0p]];
DefTensor[PA1pT[-n, -m, -o, a, b, c], M4, PrintAs -> SymbolBuild[PATSymb, Spin1p]];
DefTensor[PA2pT[-n, -m, -o, a, b, c], M4, PrintAs -> SymbolBuild[PATSymb, Spin2p]];
DefTensor[PA0mT[-n, -m, -o, a, b, c], M4, PrintAs -> SymbolBuild[PATSymb, Spin0m]];
DefTensor[PA1mT[-n, -m, -o, a, b, c], M4, PrintAs -> SymbolBuild[PATSymb, Spin1m]];
DefTensor[PA2mT[-n, -m, -o, a, b, c], M4, PrintAs -> SymbolBuild[PATSymb, Spin2m]];
ClearBuild[];
```

CompleteO3ProjectionsToggle

```
IfBuild["CompleteO3ProjectionsToggle",
  PB0pTDefinition =
   (1/3) PPara[-n, -m] PB0p[e, f] PBPara[-e, -f, a, c] /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical;
  PB1pTDefinition = PB1p[-n, -m, e, f] PBPara[-e, -f, a, c] /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical;
  PB2pTDefinition = PB2p[-n, -m, e, f] PBPara[-e, -f, a, c] /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical;
  PB1mTDefinition = V[-n] PB1m[-m, f] PBPerp[-f, a, c] /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical;

  PB0pTActivate = MakeRule[{PB0pT[-n, -m, a, c], Evaluate[PB0pTDefinition]},
    MetricOn → All, ContractMetrics → True];
  PB1pTActivate = MakeRule[{PB1pT[-n, -m, a, c], Evaluate[PB1pTDefinition]},
    MetricOn → All, ContractMetrics → True];
  PB2pTActivate = MakeRule[{PB2pT[-n, -m, a, c], Evaluate[PB2pTDefinition]},
    MetricOn → All, ContractMetrics → True];
  PB1mTActivate = MakeRule[{PB1mT[-n, -m, a, c], Evaluate[PB1mTDefinition]},
    MetricOn → All, ContractMetrics → True];

  PA0pTDefinition =
   Antisymmetrize[Antisymmetrize[2 Antisymmetrize[V[-n] (1/3) PPara[-m, -o]
            PA0p[e, f] PAPerp[-e, -f, a, b, c], {-n, -m}], {-n, -m}], {a, b}] /.
     PO3PiActivate /. PADMPiActivate /. PADMActivate // ToCanonical;
  PA1pTDefinition = Antisymmetrize[Antisymmetrize[2 Antisymmetrize[
```

```
          V[-n] PA1p[-m, -o, e, f] PAPerp[-e, -f, a, b, c], {-n, -m}], {-n, -m}],
       {a, b}] /. PO3PiActivate /. PADMPiActivate /. PADMActivate // ToCanonical;
PA2pTDefinition = Antisymmetrize[Antisymmetrize[2 Antisymmetrize[
          V[-n] PA2p[-m, -o, e, f] PAPerp[-e, -f, a, b, c], {-n, -m}], {-n, -m}],
       {a, b}] /. PO3PiActivate /. PADMPiActivate /. PADMActivate // ToCanonical;
PA0mTDefinition = Antisymmetrize[Antisymmetrize[(-1/6) PA0m[-n, -m, -o]
         PA0m[i, j, k] PAPara[-i, -j, -k, a, b, c], {-n, -m}], {a, b}] /.
     PO3PiActivate /. PADMPiActivate /. PADMActivate // ToCanonical;
PA1mTDefinition = Antisymmetrize[Antisymmetrize[Antisymmetrize[-PPara[-m, -o]
          PA1m[-n, i, j, k] PAPara[-i, -j, -k, a, b, c], {-m, -n}], {-n, -m}],
       {a, b}] /. PO3PiActivate /. PADMPiActivate /. PADMActivate // ToCanonical;
PA2mTDefinition = Antisymmetrize[Antisymmetrize[(4/3) PA2m[-n, -m, -o, d, e, f]
         PAPara[-d, -e, -f, a, b, c], {-n, -m}], {a, b}] /.
     PO3PiActivate /. PADMPiActivate /. PADMActivate // ToCanonical;

PA0pTActivate = MakeRule[{PA0pT[-n, -m, -o, a, b, c], Evaluate[PA0pTDefinition]},
  MetricOn → All, ContractMetrics → True];
PA1pTActivate = MakeRule[{PA1pT[-n, -m, -o, a, b, c], Evaluate[PA1pTDefinition]},
  MetricOn → All, ContractMetrics → True];
PA2pTActivate = MakeRule[{PA2pT[-n, -m, -o, a, b, c], Evaluate[PA2pTDefinition]},
  MetricOn → All, ContractMetrics → True];
PA0mTActivate = MakeRule[{PA0mT[-n, -m, -o, a, b, c], Evaluate[PA0mTDefinition]},
  MetricOn → All, ContractMetrics → True];
PA1mTActivate = MakeRule[{PA1mT[-n, -m, -o, a, b, c], Evaluate[PA1mTDefinition]},
  MetricOn → All, ContractMetrics → True];
PA2mTActivate = MakeRule[{PA2mT[-n, -m, -o, a, b, c], Evaluate[PA2mTDefinition]},
  MetricOn → All, ContractMetrics → True];

NewPO3TActivate = Join[PB0pTActivate, PB1pTActivate,
  PB2pTActivate, PB1mTActivate, PA0pTActivate, PA1pTActivate,
  PA2pTActivate, PA0mTActivate, PA1mTActivate, PA2mTActivate];

tmp =
 (PA0pT[-n, -m, -o, a, b, c] + PA1pT[-n, -m, -o, a, b, c] + PA2pT[-n, -m, -o, a, b,
          c] + PA0mT[-n, -m, -o, a, b, c] + PA1mT[-n, -m, -o, a, b, c] +
         PA2mT[-n, -m, -o, a, b, c]) APi[-a, -b, -e]
       G3[e, -f] B[-c, f] /. NewPO3TActivate /. PO3PiActivate /.
    PADMPiActivate /. PADMActivate // ToCanonical;
HiGGSPrint[tmp];

tmp =
 (PB0pT[-n, -m, a, c] + PB1pT[-n, -m, a, c] + PB2pT[-n, -m, a, c] + PB1mT[-n, -m, a,
          c]) BPi[-a, -e] G3[e, -f] B[-c, f] /. NewPO3TActivate /.
```

```
      PO3PiActivate /. PADMPiActivate /. PADMActivate // ToCanonical;
  HiGGSPrint[tmp];

  DumpSave[BinaryLocation["CompleteO3ProjectionsToggle"], {NewPO3TActivate}];
  ClearBuild["CompleteO3ProjectionsToggle"];
 ];
```

build

In[ ]:=
```
OpenLastCache[];
```

build

Projection normalisations $\{c^{\pm}_{A}\}$. $\{c^{\pm}_{E}\}$

build

In[ ]:=
```
DefConstantSymbol[cPerpA0p, PrintAs → "c^+_A0+"];
DefConstantSymbol[cPerpA0m, PrintAs → "c^+_A0-"];
DefConstantSymbol[cPerpA1p, PrintAs → "c^+_A1+"];
DefConstantSymbol[cPerpA1m, PrintAs → "c^+_A1-"];
DefConstantSymbol[cPerpA2p, PrintAs → "c^+_A2+"];
DefConstantSymbol[cPerpA2m, PrintAs → "c^+_A2-"];

DefConstantSymbol[cPerpB0p, PrintAs → "c^+_b0+"];
DefConstantSymbol[cPerpB0m, PrintAs → "c^+_b0-"];
DefConstantSymbol[cPerpB1p, PrintAs → "c^+_b1+"];
DefConstantSymbol[cPerpB1m, PrintAs → "c^+_b1-"];
DefConstantSymbol[cPerpB2p, PrintAs → "c^+_b2+"];
DefConstantSymbol[cPerpB2m, PrintAs → "c^+_b2-"];
ClearBuild[];
```

ProjectionNormalisations oT ggle

In[ ]:=
```
IfBuild["ProjectionNormalisationsToggle",
  Solutions = {};
  tmp =
   PB0pT[-n, -m, a, c] - cPerpB0p PB0p[g, h] PBPara[-g, -h, -n, -m] PB0p[e, f] PBPara[
            -e, -f, a, c] /. NewPO3TActivate /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
  Solutions = Join[Solutions, Solve[ToConstantSymbolEquations[tmp == 0],
      cPerpB0p][[1]]];
  tmp = PB1pT[-n, -m, a, c] - cPerpB1p PB1p[-x, -y, g, h] PBPara[-g, -h, -n, -m] PB1p[
            x, y, e, f] PBPara[-e, -f, a, c] /. NewPO3TActivate /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
  Solutions = Join[Solutions, Solve[ToConstantSymbolEquations[tmp == 0],
      cPerpB1p][[1]]];
  tmp = PB1mT[-n, -m, a, c] - cPerpB1m PB1m[-x, h] PBPerp[-h, -n, -m]
```

```
          PB1m[x, f] PBPerp[-f, a, c] /. NewPO3TActivate /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
Solutions = Join[Solutions, Solve[ToConstantSymbolEquations[tmp == 0],
    cPerpB1m][[1]]];
tmp = PB2pT[-n, -m, a, c] - cPerpB2p PB2p[-x, -y, g, h] PBPara[-g, -h, -n, -m] PB2p[
           x, y, e, f] PBPara[-e, -f, a, c] /. NewPO3TActivate /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
Solutions = Join[Solutions, Solve[ToConstantSymbolEquations[tmp == 0],
    cPerpB2p][[1]]];
tmp = Antisymmetrize[Antisymmetrize[PA0pT[-n, -m, -o, a, b, c] - cPerpA0p PA0p[g,
             h] PAPerp[-g, -h, -n, -m, -o] PA0p[e, f] PAPerp[-e, -f, a, b, c],
          {-n, -m}], {a, b}] /. NewPO3TActivate /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
Solutions = Join[Solutions, Solve[ToConstantSymbolEquations[tmp == 0],
    cPerpA0p][[1]]];
tmp = Antisymmetrize[Antisymmetrize[PA0mT[-n, -m, -o, a, b, c] -
           cPerpA0m PA0m[g, h, i] PAPara[-g, -h, -i, -n, -m, -o]
            PA0m[e, f, j] PAPara[-e, -f, -j, a, b, c], {-n, -m}], {a, b}] /.
        NewPO3TActivate /. PO3PiActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
Solutions = Join[Solutions, Solve[ToConstantSymbolEquations[tmp == 0],
    cPerpA0m][[1]]];
tmp = Antisymmetrize[Antisymmetrize[PA1pT[-n, -m, -o, a, b, c] -
           cPerpA1p PA1p[-x, -y, g, h] PAPerp[-g, -h, -n, -m, -o]
            PA1p[x, y, e, f] PAPerp[-e, -f, a, b, c], {-n, -m}], {a, b}] /.
        NewPO3TActivate /. PO3PiActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
Solutions = Join[Solutions, Solve[ToConstantSymbolEquations[tmp == 0],
    cPerpA1p][[1]]];
tmp = Antisymmetrize[Antisymmetrize[PA1mT[-n, -m, -o, a, b, c] -
           cPerpA1m PA1m[-x, g, h, i] PAPara[-g, -h, -i, -n, -m, -o]
            PA1m[x, e, f, j] PAPara[-e, -f, -j, a, b, c], {-n, -m}], {a, b}] /.
        NewPO3TActivate /. PO3PiActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
Solutions = Join[Solutions, Solve[ToConstantSymbolEquations[tmp == 0],
    cPerpA1m][[1]]];
tmp = Antisymmetrize[Antisymmetrize[PA2pT[-n, -m, -o, a, b, c] -
           cPerpA2p PA2p[-x, -y, g, h] PAPerp[-g, -h, -n, -m, -o]
            PA2p[x, y, e, f] PAPerp[-e, -f, a, b, c], {-n, -m}], {a, b}] /.
        NewPO3TActivate /. PO3PiActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
Solutions = Join[Solutions, Solve[ToConstantSymbolEquations[tmp == 0],
    cPerpA2p][[1]]];
tmp = Antisymmetrize[Antisymmetrize[PA2mT[-n, -m, -o, a, b, c] -
```

```
            cPerpA2m PA2m[-x, -y, -z, g, h, i] PAPara[-g, -h, -i, -n, -m, -o]
             PA2m[x, y, z, e, f, j] PAPara[-e, -f, -j, a, b, c], {-n, -m}],
          {a, b}] /. NewPO3TActivate /. PO3PiActivate /. PADMPiActivate /.
      PADMActivate // ToCanonical // CollectTensors;
  Solutions = Join[Solutions, Solve[ToConstantSymbolEquations[tmp == 0],
      cPerpA2m][[1]]];
  TocPerp = Solutions;

  DumpSave[BinaryLocation[], {TocPerp}];
  ClearBuild[];
 ];
```

build

In[•]:=
```
OpenLastCache[];
```

ProjectionNormalisationsCheckToggle

```
IfBuild["ProjectionNormalisationsCheckToggle",
  HiGGSPrint[Style["B0p", Blue, 20]];
  tmp = PB0p[g, h] PBPara[-g, -h, -n, -m] PB0p[e, f] PBPara[-e, -f, n, m] -
          (1/cPerpB0p) /. TocPerp /. NewPO3TActivate /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
  HiGGSPrint[tmp];
  HiGGSPrint[Style["B1p", Blue, 20]];
  tmp =
   PB1p[-x, -y, g, h] PBPara[-g, -h, -n, -m] PB1p[u, v, e, f] PBPara[-e, -f, n, m] -
          (1/cPerpB1p) Antisymmetrize[Antisymmetrize[PPara[-x, u] PPara[-y, v],
              {-x, -y}], {u, v}] /. TocPerp /. NewPO3TActivate /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
  HiGGSPrint[tmp];
  HiGGSPrint[Style["B1m", Blue, 20]];
  tmp =
   PB1m[-x, h] PBPerp[-h, -n, -m] PB1m[u, f] PBPerp[-f, n, m] - (1/cPerpB1m) PPara[
            -x, u] /. TocPerp /. NewPO3TActivate /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
  HiGGSPrint[tmp];
  HiGGSPrint[Style["B2p", Blue, 20]];
  tmp =
   PB2p[-x, -y, g, h] PBPara[-g, -h, -n, -m] PB2p[u, v, e, f] PBPara[-e, -f, n, m] -
          (1/cPerpB2p) Symmetrize[Symmetrize[PPara[-x, u] PPara[-y, v],
              {-x, -y}], {u, v}] /. TocPerp /. NewPO3TActivate /. PO3PiActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
  HiGGSPrint[tmp];
  HiGGSPrint[Style["A0p", Red, 20]];
  tmp = Antisymmetrize[ PA0p[g, h] PAPerp[-g, -h, -n, -m, -o], {-n, -m}] PA0p[e, f]
```

```
          PAPerp[-e, -f, a, b, c] G[n, -a] G[m, -b] G[o, -c] - (1 / cPerpA0p) /.
        TocPerp /. NewPO3TActivate /. PO3PiActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
HiGGSPrint[tmp];
HiGGSPrint[Style["A0m", Red, 20]];
tmp =
 Antisymmetrize[PA0m[g, h, i] PAPara[-g, -h, -i, -n, -m, -o], {-n, -m}] PA0m[e,
           f, j] PAPara[-e, -f, -j, a, b, c] G[n, -a] G[m, -b] G[o, -c] -
         (1 / cPerpA0m) /. TocPerp /. NewPO3TActivate /. PO3PiActivate /.
     PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
HiGGSPrint[tmp];
HiGGSPrint[Style["A1p", Red, 20]];
tmp =
 Antisymmetrize[PA1p[-x, -y, g, h] PAPerp[-g, -h, -n, -m, -o], {-n, -m}] PA1p[
           u, v, e, f] PAPerp[-e, -f, a, b, c] G[n, -a] G[m, -b] G[o, -c] -
         (1 / cPerpA1p) Antisymmetrize[Antisymmetrize[PPara[-x, u] PPara[-y, v],
            {-x, -y}], {u, v}] /. TocPerp /. NewPO3TActivate /. PO3PiActivate /.
     PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
HiGGSPrint[tmp];
HiGGSPrint[Style["A1m", Red, 20]];
tmp =
 Antisymmetrize[PA1m[-x, g, h, i] PAPara[-g, -h, -i, -n, -m, -o], {-n, -m}] PA1m[
           u, e, f, j] PAPara[-e, -f, -j, a, b, c] G[n, -a]
          G[m, -b] G[o, -c] - (1 / cPerpA1m) PPara[-x, u] /. TocPerp /.
       NewPO3TActivate /. PO3PiActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
HiGGSPrint[tmp];
HiGGSPrint[Style["A2p", Red, 20]];
tmp =
 Antisymmetrize[PA2p[-x, -y, g, h] PAPerp[-g, -h, -n, -m, -o], {-n, -m}] PA2p[
           u, v, e, f] PAPerp[-e, -f, a, b, c] G[n, -a] G[m, -b] G[o, -c] -
         (1 / cPerpA2p) Symmetrize[Symmetrize[PPara[-x, u] PPara[-y, v],
            {-x, -y}], {u, v}] /. TocPerp /. NewPO3TActivate /. PO3PiActivate /.
     PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
HiGGSPrint[tmp];
HiGGSPrint[Style["A2m", Red, 20]];
tmp =
 Antisymmetrize[ PA2m[-x, -y, -z, g, h, i] PAPara[-g, -h, -i, -n, -m, -o], {-n,
            -m}] PA2m[u, v, w, e, f, j] PAPara[-e, -f, -j, a, b, c] G[n, -a]
          G[m, -b] G[o, -c] - (1 / cPerpA2m) Antisymmetrize[Antisymmetrize[
           PPara[-x, u] PPara[-y, v] PPara[-z, w], {-x, -y}], {u, v}] /.
        TocPerp /. NewPO3TActivate /. PO3PiActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
```

```
  HiGGSPrint[tmp];
  ClearBuild[];
 ];
```

build

Transfer couplings $\{\hat{\alpha}_A^{++}\}$, $\{\hat{\beta}_E^{++}\}$

build

In[•]:=

```
DefConstantSymbol[BetPerpPerp0p, PrintAs → Colour["β̂₀₊⁺⁺", $Coupling]];
DefConstantSymbol[BetPerpPerp0m, PrintAs → Colour["β̂₀₋⁺⁺", $Coupling]];
DefConstantSymbol[BetPerpPerp1p, PrintAs → Colour["β̂₁₊⁺⁺", $Coupling]];
DefConstantSymbol[BetPerpPerp1m, PrintAs → Colour["β̂₁₋⁺⁺", $Coupling]];
DefConstantSymbol[BetPerpPerp2p, PrintAs → Colour["β̂₂₊⁺⁺", $Coupling]];
DefConstantSymbol[BetPerpPerp2m, PrintAs → Colour["β̂₂₋⁺⁺", $Coupling]];

BetPerpPerp = {BetPerpPerp0p, BetPerpPerp0m,
   BetPerpPerp1p, BetPerpPerp1m, BetPerpPerp2p, BetPerpPerp2m};

DefConstantSymbol[AlpPerpPerp0p, PrintAs → Colour["α̂₀₊⁺⁺", $Coupling]];
DefConstantSymbol[AlpPerpPerp0m, PrintAs → Colour["α̂₀₋⁺⁺", $Coupling]];
DefConstantSymbol[AlpPerpPerp1p, PrintAs → Colour["α̂₁₊⁺⁺", $Coupling]];
DefConstantSymbol[AlpPerpPerp1m, PrintAs → Colour["α̂₁₋⁺⁺", $Coupling]];
DefConstantSymbol[AlpPerpPerp2p, PrintAs → Colour["α̂₂₊⁺⁺", $Coupling]];
DefConstantSymbol[AlpPerpPerp2m, PrintAs → Colour["α̂₂₋⁺⁺", $Coupling]];

AlpPerpPerp = {AlpPerpPerp0p, AlpPerpPerp0m,
   AlpPerpPerp1p, AlpPerpPerp1m, AlpPerpPerp2p, AlpPerpPerp2m};
ClearBuild[];
```

TransferCouplingsPerpPerpToggle

In[•]:=

```
IfBuild["TransferCouplingsPerpPerpToggle",
  Transfer$CouplingsPerpPerpSolutions = {};
  tmp =
   BetPerpPerp0p PB0p[g, h] PBPara[-g, -h, a, e] - PB0p[x, z] PBPara[-x, -z, i, f]
            V[g] PPara[-f, h] V[-c] PPara[e, -d] (Bet1 PT1[-i, -g, -h, a, c, d] +
              Bet2 PT2[-i, -g, -h, a, c, d] + Bet3 PT3[-i, -g, -h, a, c, d]) /.
          PO3PiActivate /. PActivate /. PADMPiActivate /.
       PADMActivate // ToCanonical // CollectTensors;
  Transfer$CouplingsPerpPerpSolutions = Join[Transfer$CouplingsPerpPerpSolutions,
    Solve[ToConstantSymbolEquations[tmp == 0], BetPerpPerp0p][[1]]];
  tmp = BetPerpPerp1p PB1p[-q, -r, g, h] PBPara[-g, -h, a, e] - PB1p[-q, -r, x, z]
            PBPara[-x, -z, i, f] V[g] PPara[-f, h] V[-c] PPara[e, -d]
            (Bet1 PT1[-i, -g, -h, a, c, d] + Bet2 PT2[-i, -g, -h, a, c, d] +
```

```
            Bet3 PT3[-i, -g, -h, a, c, d]) /. PO3PiActivate /. PActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpPerpSolutions = Join[Transfer$CouplingsPerpPerpSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], BetPerpPerp1p][[1]]];
tmp = BetPerpPerp1m PB1m[-q, h] PBPerp[-h, a, e] - PB1m[-q, z] PBPerp[-z, i, f]
          V[g] PPara[-f, h] V[-c] PPara[e, -d] (Bet1 PT1[-i, -g, -h, a, c, d] +
            Bet2 PT2[-i, -g, -h, a, c, d] + Bet3 PT3[-i, -g, -h, a, c, d]) /.
        PO3PiActivate /. PActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpPerpSolutions = Join[Transfer$CouplingsPerpPerpSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], BetPerpPerp1m][[1]]];
tmp = BetPerpPerp2p PB2p[-q, -r, g, h] PBPara[-g, -h, a, e] - PB2p[-q, -r, x, z]
          PBPara[-x, -z, i, f] V[g] PPara[-f, h] V[-c] PPara[e, -d]
          (Bet1 PT1[-i, -g, -h, a, c, d] + Bet2 PT2[-i, -g, -h, a, c, d] +
            Bet3 PT3[-i, -g, -h, a, c, d]) /. PO3PiActivate /. PActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpPerpSolutions = Join[Transfer$CouplingsPerpPerpSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], BetPerpPerp2p][[1]]];
tmp = AlpPerpPerp0p PA0p[g, h] Antisymmetrize[PAPerp[-g, -h, a, b, e], {a, b}] -
         PA0p[x, z] PAPerp[-x, -z, i, j, f] V[g] PPara[-f, h] V[-c] PPara[e, -d]
          (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2 PR2[-i, -j, -g, -h,
              a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b, c, d] + Alp4
            PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
            Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /. PO3PiActivate /. PActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpPerpSolutions = Join[Transfer$CouplingsPerpPerpSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPerp0p][[1]]];
tmp = AlpPerpPerp0m PA0m[g, h, i] Antisymmetrize[PAPara[-g, -h, -i, a, b, e],
           {a, b}] - PA0m[x, y, z] PAPara[-x, -y, -z, i, j, f] V[g] PPara[-f, h]
          V[-c] PPara[e, -d] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2
            PR2[-i, -j, -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
            Alp4 PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j, -g,
              -h, a, b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /.
        PO3PiActivate /. PActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpPerpSolutions = Join[Transfer$CouplingsPerpPerpSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPerp0m][[1]]];
tmp = AlpPerpPerp1p PA1p[-p, -q, g, h] Antisymmetrize[PAPerp[-g, -h, a, b, e],
           {a, b}] - PA1p[-p, -q, x, z] PAPerp[-x, -z, i, j, f] V[g] PPara[-f, h]
          V[-c] PPara[e, -d] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2
            PR2[-i, -j, -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
            Alp4 PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j, -g,
              -h, a, b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /.
```

```
        PO3PiActivate /. PActivate /. PADMPiActivate /.
      PADMActivate // ToCanonical // CollectTensors;
  Transfer$CouplingsPerpPerpSolutions = Join[Transfer$CouplingsPerpPerpSolutions,
    Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPerp1p][[1]]];
  tmp = AlpPerpPerp1m PA1m[-p, g, h, i] Antisymmetrize[PAPara[-g, -h, -i, a, b, e],
            {a, b}] - PA1m[-p, x, y, z] PAPara[-x, -y, -z, i, j, f] V[g] PPara[-f, h]
          V[-c] PPara[e, -d] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2
             PR2[-i, -j, -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
           Alp4 PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j, -g,
             -h, a, b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /.
        PO3PiActivate /. PActivate /. PADMPiActivate /.
      PADMActivate // ToCanonical // CollectTensors;
  Transfer$CouplingsPerpPerpSolutions = Join[Transfer$CouplingsPerpPerpSolutions,
    Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPerp1m][[1]]];
  tmp = AlpPerpPerp2p PA2p[-p, -q, g, h] Antisymmetrize[PAPerp[-g, -h, a, b, e],
            {a, b}] - PA2p[-p, -q, x, z] PAPerp[-x, -z, i, j, f] V[g] PPara[-f, h]
          V[-c] PPara[e, -d] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2
             PR2[-i, -j, -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
           Alp4 PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j, -g,
             -h, a, b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /.
        PO3PiActivate /. PActivate /. PADMPiActivate /.
      PADMActivate // ToCanonical // CollectTensors;
  Transfer$CouplingsPerpPerpSolutions = Join[Transfer$CouplingsPerpPerpSolutions,
    Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPerp2p][[1]]];
  tmp = AlpPerpPerp2m PA2m[-q, -p, -r, g, h, i]
            Antisymmetrize[PAPara[-g, -h, -i, a, b, e], {a, b}] -
          PA2m[-q, -p, -r, x, y, z] PAPara[-x, -y, -z, i, j, f] V[g] PPara[-f, h]
          V[-c] PPara[e, -d] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2
             PR2[-i, -j, -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
           Alp4 PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j, -g,
             -h, a, b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /.
        PO3PiActivate /. PActivate /. PADMPiActivate /.
      PADMActivate // ToCanonical // CollectTensors;
  Transfer$CouplingsPerpPerpSolutions = Join[Transfer$CouplingsPerpPerpSolutions,
    Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPerp2m][[1]]];

  DumpSave[BinaryLocation[], {Transfer$CouplingsPerpPerpSolutionsDUMMY}];
  ClearBuild[];
 ];
```

build

In[•]:=

```
OpenLastCache[];
```

build

Human-readable projections {ᴬ$\check{\mathcal{P}}$}, {ᴱ$\check{\mathcal{P}}$}

build

In[•]:=

```
(*Projection operators which define the O(3)
 decomposition of the canonical parts of field strengths*)
PTSymb = "$\check{\mathcal{P}}_{\mathcal{T}}$";
DefTensor[PT0m[d, e, f], M4, PrintAs -> SymbolBuild[PTSymb, Spin0m]];
DefTensor[PT1p[-a, -b, c, d], M4, PrintAs -> SymbolBuild[PTSymb, Spin1p]];
DefTensor[PT1m[-a, d, e, f], M4, PrintAs -> SymbolBuild[PTSymb, Spin1m]];
DefTensor[PT2m[-a, -b, -c, d, e, f], M4, PrintAs -> SymbolBuild[PTSymb, Spin2m]];
PRSymb = "$\check{\mathcal{P}}_{\mathcal{R}}$";
DefTensor[PR0p[e, f, g, h], M4, PrintAs -> SymbolBuild[PRSymb, Spin0p]];
DefTensor[PR0m[e, f, g], M4, PrintAs -> SymbolBuild[PRSymb, Spin0m]];
DefTensor[PR1p[-n, -m, e, f, g, h], M4, PrintAs -> SymbolBuild[PRSymb, Spin1p]];
DefTensor[PR1m[-n, e, f, g], M4, PrintAs -> SymbolBuild[PRSymb, Spin1m]];
DefTensor[PR2p[-n, -m, e, f, g, h], M4, PrintAs -> SymbolBuild[PRSymb, Spin2p]];
DefTensor[PR2m[-n, -m, -o, e, f, g], M4, PrintAs -> SymbolBuild[PRSymb, Spin2m]];
PT0mDefinition = PPara[-i, d] PPara[-j, e] PPara[-k, f] epsilonG[i, j, k, g] V[-g] /.
    PADMActivate // ToCanonical;
PT1pDefinition = PPara[-a, i] PPara[-b, j] PPara[c, -k] PPara[d, -l]
     Antisymmetrize[G[-i, k] G[-j, l], {-i, -j}] /. PADMActivate // ToCanonical;
PT1mDefinition = PPara[-i, d] PPara[-j, f] PPara[k, -a] PPara[-l, e]
     G[i, j] G[-k, l] /. PADMActivate // ToCanonical;
PT2mDefinition = PPara[-a, i] PPara[-b, j] PPara[-c, k] PPara[e, -l]
     PPara[f, -n] PPara[d, -m] (3/4) ((1/3) (2 G[-i, l] G[-j, n] G[-k, m] -
          G[-j, l] G[-k, n] G[-i, m] - G[-k, l] G[-i, n] G[-j, m]) - Antisymmetrize[
        G[-i, -k] G[-j, n] G[l, m], {-i, -j}]) /. PADMActivate // ToCanonical;
PR0pDefinition = PPara[-e, -g] PPara[-f, -h] /. PADMActivate // ToCanonical;
PR0mDefinition =
  PPara[-i, -e] PPara[-j, -f] PPara[-k, -g] epsilonG[i, j, k, p] V[-p] /.
    PADMActivate // ToCanonical;
PR1pDefinition = PPara[-e, -g] Antisymmetrize[PPara[-n, -f] PPara[-m, -h],
      {-n, -m}] /. PADMActivate // ToCanonical;
PR1mDefinition = PPara[-e, -g] PPara[-n, -f] /. PADMActivate // ToCanonical;
PR2pDefinition =
  PPara[-e, -g] (Symmetrize[PPara[-n, -f] PPara[-m, -h], {-n, -m}] - (1/3)
       PPara[-n, -m] PPara[-f, -h]) /. PADMActivate // ToCanonical;
PR2mDefinition = PPara[-a, i] PPara[-b, j] PPara[-c, k] PPara[e, -l]
     PPara[f, -n] PPara[d, -m] (3/4) ((1/3) (2 G[-i, l] G[-j, n] G[-k, m] -
          G[-j, l] G[-k, n] G[-i, m] - G[-k, l] G[-i, n] G[-j, m]) - Antisymmetrize[
        G[-i, -k] G[-j, n] G[l, m], {-i, -j}]) /. PADMActivate // ToCanonical;
PT0mActivate = MakeRule[{PT0m[d, e, f], Evaluate[PT0mDefinition]},
```

```
    MetricOn → All, ContractMetrics → True];
PT1pActivate = MakeRule[{PT1p[-a, -b, c, d], Evaluate[PT1pDefinition]},
    MetricOn → All, ContractMetrics → True];
PT1mActivate = MakeRule[{PT1m[-a, d, e, f], Evaluate[PT1mDefinition]},
    MetricOn → All, ContractMetrics → True];
PT2mActivate = MakeRule[{PT2m[-a, -b, -c, d, e, f], Evaluate[PT2mDefinition]},
    MetricOn → All, ContractMetrics → True];
PR0pActivate = MakeRule[{PR0p[-e, -f, -g, -h], Evaluate[PR0pDefinition]},
    MetricOn → All, ContractMetrics → True];
PR0mActivate = MakeRule[{PR0m[-e, -f, -g], Evaluate[PR0mDefinition]},
    MetricOn → All, ContractMetrics → True];
PR1pActivate = MakeRule[{PR1p[-n, -m, -e, -f, -g, -h], Evaluate[PR1pDefinition]},
    MetricOn → All, ContractMetrics → True];
PR1mActivate = MakeRule[{PR1m[-n, -e, -f, -g], Evaluate[PR1mDefinition]},
    MetricOn → All, ContractMetrics → True];
PR2pActivate = MakeRule[{PR2p[-n, -m, -e, -f, -g, -h], Evaluate[PR2pDefinition]},
    MetricOn → All, ContractMetrics → True];
PR2mActivate = MakeRule[{PR2m[-a, -b, -c, d, e, f], Evaluate[PR2mDefinition]},
    MetricOn → All, ContractMetrics → True];

(*These rules then expand those canonical
 field strength O(3) projection operators*)
PO3TActivate = Join[PT0mActivate, PT1pActivate, PT1mActivate, PT2mActivate];
PO3RActivate = Join[PR0pActivate, PR0mActivate,
    PR1pActivate, PR1mActivate, PR2pActivate, PR2mActivate];
ClearBuild[];
```

build

Projection normalisations $\{c_A^{\shortparallel}\}$. $\{c_E^{\shortparallel}\}$

build

```
In[ ]:= DefConstantSymbol[cParaA0p, PrintAs → "c^H_A0+"];
        DefConstantSymbol[cParaA0m, PrintAs → "c^H_A0-"];
        DefConstantSymbol[cParaA1p, PrintAs → "c^H_A1+"];
        DefConstantSymbol[cParaA1m, PrintAs → "c^H_A1-"];
        DefConstantSymbol[cParaA2p, PrintAs → "c^H_A2+"];
        DefConstantSymbol[cParaA2m, PrintAs → "c^H_A2-"];

        DefConstantSymbol[cParaB0p, PrintAs → "c^H_b0+"];
        DefConstantSymbol[cParaB0m, PrintAs → "c^H_b0-"];
        DefConstantSymbol[cParaB1p, PrintAs → "c^H_b1+"];
        DefConstantSymbol[cParaB1m, PrintAs → "c^H_b1-"];
        DefConstantSymbol[cParaB2p, PrintAs → "c^H_b2+"];
        DefConstantSymbol[cParaB2m, PrintAs → "c^H_b2-"];
        ClearBuild[];
```

build

Transfer couplings $\{\hat{\alpha}_A^{\pm H}\}$, $\{\hat{\beta}_E^{\pm H}\}$

build

```
In[ ]:= DefConstantSymbol[AlpPerpPara0p, PrintAs → Colour["α̂0+^H", $Coupling]];
        DefConstantSymbol[AlpPerpPara0m, PrintAs → Colour["α̂0-^H", $Coupling]];
        DefConstantSymbol[AlpPerpPara1p, PrintAs → Colour["α̂1+^H", $Coupling]];
        DefConstantSymbol[AlpPerpPara1m, PrintAs → Colour["α̂1-^H", $Coupling]];
        DefConstantSymbol[AlpPerpPara2p, PrintAs → Colour["α̂2+^H", $Coupling]];
        DefConstantSymbol[AlpPerpPara2m, PrintAs → Colour["α̂2-^H", $Coupling]];

        AlpPerpPara = {AlpPerpPara0p, AlpPerpPara0m,
           AlpPerpPara1p, AlpPerpPara1m, AlpPerpPara2p, AlpPerpPara2m};

        DefConstantSymbol[BetPerpPara0p, PrintAs → Colour["β̂0+^H", $Coupling]];
        DefConstantSymbol[BetPerpPara0m, PrintAs → Colour["β̂0-^H", $Coupling]];
        DefConstantSymbol[BetPerpPara1p, PrintAs → Colour["β̂1+^H", $Coupling]];
        DefConstantSymbol[BetPerpPara1m, PrintAs → Colour["β̂1-^H", $Coupling]];
        DefConstantSymbol[BetPerpPara2p, PrintAs → Colour["β̂2+^H", $Coupling]];
        DefConstantSymbol[BetPerpPara2m, PrintAs → Colour["β̂2-^H", $Coupling]];

        BetPerpPara = {BetPerpPara0p, BetPerpPara0m,
           BetPerpPara1p, BetPerpPara1m, BetPerpPara2p, BetPerpPara2m};
        ClearBuild[];
```

TransferCouplingsPerpParaToggle

```
In[ ]:= IfBuild["TransferCouplingsPerpParaToggle",
```

```
Transfer$CouplingsPerpParaSolutions = {};
tmp =
 BetPerpPara0m PT0m[e, f, g] PTPara[-e, -f, -g, a, v, w] - PB0p[x, z] PBPara[-x,
             -z, i, f] V[g] PPara[-f, h] PPara[v, -c] PPara[w, -d]
            (Bet1 PT1[-i, -g, -h, a, c, d] + Bet2 PT2[-i, -g, -h, a, c, d] +
              Bet3 PT3[-i, -g, -h, a, c, d]) /. PO3TActivate /.
         PADMTActivate /. PO3PiActivate /. PActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpParaSolutions = Join[Transfer$CouplingsPerpParaSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], BetPerpPara0p][[1]]];
tmp = BetPerpPara1p PT1p[-n, -m, e, f] PTPerp[-e, -f, a, v, w] -
           PB1p[-q, -r, x, z] PBPara[-x, -z, i, f] V[g] PPara[-f, h] PPara[v, -c]
            PPara[w, -d] (Bet1 PT1[-i, -g, -h, a, c, d] + Bet2 PT2[-i, -g, -h,
                a, c, d] + Bet3 PT3[-i, -g, -h, a, c, d]) /. PO3TActivate /.
         PADMTActivate /. PO3PiActivate /. PActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpParaSolutions = Join[Transfer$CouplingsPerpParaSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], BetPerpPara1p][[1]]];
tmp = BetPerpPara1m PT1m[-n, e, f, g] PTPara[-e, -f, -g, a, v, w] -
           PB1m[-q, z] PBPerp[-z, i, f] V[g] PPara[-f, h] PPara[v, -c]
            PPara[w, -d] (Bet1 PT1[-i, -g, -h, a, c, d] + Bet2 PT2[-i, -g, -h,
                a, c, d] + Bet3 PT3[-i, -g, -h, a, c, d]) /. PO3TActivate /.
         PADMTActivate /. PO3PiActivate /. PActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpParaSolutions = Join[Transfer$CouplingsPerpParaSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], BetPerpPara1m][[1]]];
tmp = BetPerpPara2p PT2m[-n, -m, -o, e, f, g] PTPara[-e, -f, -g, a, v, w] -
          PB2p[-q, -r, x, z] PBPara[-x, -z, i, f] V[g] PPara[-f, h] PPara[v, -c]
            PPara[w, -d] (Bet1 PT1[-i, -g, -h, a, c, d] + Bet2 PT2[-i, -g, -h,
                a, c, d] + Bet3 PT3[-i, -g, -h, a, c, d]) /. PO3TActivate /.
        PADMTActivate.PO3PiActivate /. PActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpParaSolutions = Join[Transfer$CouplingsPerpParaSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], BetPerpPara2p][[1]]];
tmp = AlpPerpPara0p PR0p[e, f, g, h] Antisymmetrize[
             PRPara[-e, -f, -g, -h, a, b, v, w], {a, b}] -
           PA0p[x, z] PAPerp[-x, -z, i, j, f] V[g] PPara[-f, h] PPara[v, -c]
            PPara[w, -d] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2 PR2[-i, -j,
                -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b, c, d] + Alp4
               PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j, -g, -h, a,
                b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /. PO3TActivate /.
         PADMTActivate /. PO3PiActivate /. PActivate /. PADMPiActivate /.
    PADMActivate // ToCanonical // CollectTensors;
```

```
Transfer$CouplingsPerpParaSolutions = Join[Transfer$CouplingsPerpParaSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPara0p][[1]]];
tmp = AlpPerpPara0m PR0m[e, f, g] Antisymmetrize[PRPerp[-e, -f, -g, a, b, v, w],
        {a, b}] - PA0m[x, y, z] PAPara[-x, -y, -z, i, j, f] V[g] PPara[-f, h]
       PPara[v, -c] PPara[w, -d] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] +
         Alp2 PR2[-i, -j, -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a,
           b, c, d] + Alp4 PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j,
           -g, -h, a, b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /.
     PO3TActivate /. PADMTActivate /. PO3PiActivate /. PActivate /.
   PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpParaSolutions = Join[Transfer$CouplingsPerpParaSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPara0m][[1]]];
tmp = AlpPerpPara1p PR1p[-n, -m, e, f, g, h]
        Antisymmetrize[PRPara[-e, -f, -g, -h, a, b, v, w], {a, b}] -
       PA1p[-p, -q, x, z] PAPerp[-x, -z, i, j, f] V[g] PPara[-f, h]
        PPara[v, -c] PPara[w, -d] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2
           PR2[-i, -j, -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b,
            c, d] + Alp4 PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j,
            -g, -h, a, b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /.
     PO3TActivate /. PADMTActivate /. PO3PiActivate /. PActivate /.
   PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpParaSolutions = Join[Transfer$CouplingsPerpParaSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPara1p][[1]]];
tmp = AlpPerpPara1m PR1m[-n, e, f, g] Antisymmetrize[
        PRPerp[-e, -f, -g, a, b, v, w], {a, b}] - PA1m[-p, x, y, z]
       PAPara[-x, -y, -z, i, j, f] V[g] PPara[-f, h] PPara[v, -c]
       PPara[w, -d] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2 PR2[-i, -j,
            -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b, c, d] + Alp4
           PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j, -g, -h, a,
            b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /. PO3TActivate /.
     PADMTActivate /. PO3PiActivate /. PActivate /. PADMPiActivate /.
   PADMActivate // ToCanonical // CollectTensors;
Transfer$CouplingsPerpParaSolutions = Join[Transfer$CouplingsPerpParaSolutions,
  Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPara1m][[1]]];
tmp = AlpPerpPara2p PR2p[-n, -m, e, f, g, h]
        Antisymmetrize[PRPara[-e, -f, -g, -h, a, b, v, w], {a, b}] -
       PA2p[-p, -q, x, z] PAPerp[-x, -z, i, j, f] V[g] PPara[-f, h]
        PPara[v, -c] PPara[w, -d] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2
           PR2[-i, -j, -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b,
            c, d] + Alp4 PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j,
            -g, -h, a, b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) /.
     PO3TActivate /. PADMTActivate /. PO3PiActivate /. PActivate /.
   PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
```

```
  Transfer$CouplingsPerpParaSolutions = Join[Transfer$CouplingsPerpParaSolutions,
    Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPara2p][[1]]];
  tmp = AlpPerpPara2m PR2m[-n, -m, -o, e, f, g]
              Antisymmetrize[PRPerp[-e, -f, -g, a, b, v, w], {a, b}]
            - PA2m[-q, -p, -r, x, y, z] PAPara[-x, -y, -z, i, j, f] V[g]
              PPara[-f, h] PPara[v, -c] PPara[w, -d] (Alp1 PR1[-i, -j, -g, -h, a, b,
                   c, d] + Alp2 PR2[-i, -j, -g, -h, a, b, c, d] + Alp3 PR3[-i, -j, -g,
                   -h, a, b, c, d] + Alp4 PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i,
                   -j, -g, -h, a, b, c, d]) /.
           PO3TActivate /. PADMTActivate /. PO3PiActivate /. PActivate /.
      PADMPiActivate /. PADMActivate // ToCanonical // CollectTensors;
  Transfer$CouplingsPerpParaSolutions = Join[Transfer$CouplingsPerpParaSolutions,
    Solve[ToConstantSymbolEquations[tmp == 0], AlpPerpPara2m][[1]]];

  DumpSave[BinaryLocation[], {Transfer$CouplingsPerpParaSolutionsDUMMY}];
  ClearBuild[];
 ];
```

build

In[•]:=

```
OpenLastCache[];
```

build

Transfer couplings $\{\hat{\alpha}_A^{\parallel\perp}\}, \{\hat{\beta}_E^{\parallel\perp}\}$

build

In[•]:=

```
DefConstantSymbol[AlpParaPerp0p, PrintAs → Colour["α̂₀₊^∥⊥", $Coupling]];
DefConstantSymbol[AlpParaPerp0m, PrintAs → Colour["α̂₀₋^∥⊥", $Coupling]];
DefConstantSymbol[AlpParaPerp1p, PrintAs → Colour["α̂₁₊^∥⊥", $Coupling]];
DefConstantSymbol[AlpParaPerp1m, PrintAs → Colour["α̂₁₋^∥⊥", $Coupling]];
DefConstantSymbol[AlpParaPerp2p, PrintAs → Colour["α̂₂₊^∥⊥", $Coupling]];
DefConstantSymbol[AlpParaPerp2m, PrintAs → Colour["α̂₂₋^∥⊥", $Coupling]];

AlpParaPerp = {AlpParaPerp0p, AlpParaPerp0m,
   AlpParaPerp1p, AlpParaPerp1m, AlpParaPerp2p, AlpParaPerp2m};

DefConstantSymbol[BetParaPerp0p, PrintAs → Colour["β̂₀₊^∥⊥", $Coupling]];
DefConstantSymbol[BetParaPerp0m, PrintAs → Colour["β̂₀₋^∥⊥", $Coupling]];
DefConstantSymbol[BetParaPerp1p, PrintAs → Colour["β̂₁₊^∥⊥", $Coupling]];
DefConstantSymbol[BetParaPerp1m, PrintAs → Colour["β̂₁₋^∥⊥", $Coupling]];
DefConstantSymbol[BetParaPerp2p, PrintAs → Colour["β̂₂₊^∥⊥", $Coupling]];
DefConstantSymbol[BetParaPerp2m, PrintAs → Colour["β̂₂₋^∥⊥", $Coupling]];

BetParaPerp = {BetParaPerp0p, BetParaPerp0m,
   BetParaPerp1p, BetParaPerp1m, BetParaPerp2p, BetParaPerp2m};
ClearBuild[];
```

build

Transfer couplings $\{\hat{\alpha}_A^{""}\}$, $\{\hat{\beta}_E^{""}\}$

build

In[ ]:=
```
DefConstantSymbol[AlpParaPara0p, PrintAs → Colour["α̂₀⁺⁺", $Coupling]];
DefConstantSymbol[AlpParaPara0m, PrintAs → Colour["α̂₀⁻⁻", $Coupling]];
DefConstantSymbol[AlpParaPara1p, PrintAs → Colour["α̂₁⁺⁺", $Coupling]];
DefConstantSymbol[AlpParaPara1m, PrintAs → Colour["α̂₁⁻⁻", $Coupling]];
DefConstantSymbol[AlpParaPara2p, PrintAs → Colour["α̂₂⁺⁺", $Coupling]];
DefConstantSymbol[AlpParaPara2m, PrintAs → Colour["α̂₂⁻⁻", $Coupling]];

AlpParaPara = {AlpParaPara0p, AlpParaPara0m,
   AlpParaPara1p, AlpParaPara1m, AlpParaPara2p, AlpParaPara2m};

DefConstantSymbol[BetParaPara0p, PrintAs → Colour["β̂₀⁺⁺", $Coupling]];
DefConstantSymbol[BetParaPara0m, PrintAs → Colour["β̂₀⁻⁻", $Coupling]];
DefConstantSymbol[BetParaPara1p, PrintAs → Colour["β̂₁⁺⁺", $Coupling]];
DefConstantSymbol[BetParaPara1m, PrintAs → Colour["β̂₁⁻⁻", $Coupling]];
DefConstantSymbol[BetParaPara2p, PrintAs → Colour["β̂₂⁺⁺", $Coupling]];
DefConstantSymbol[BetParaPara2m, PrintAs → Colour["β̂₂⁻⁻", $Coupling]];

BetParaPara = {BetParaPara0p, BetParaPara0m,
   BetParaPara1p, BetParaPara1m, BetParaPara2p, BetParaPara2m};
ClearBuild[];
```

build

Transfer solutions calculated by hand

build

In[ ]:=
```
AlpDetRelations = {AlpParaPara0p == (Alp4 + Alp6) / 2,
   AlpParaPara0m == (Alp2 + Alp3) / 2,
   AlpParaPara1p == - (Alp2 + Alp5) / 2,
   AlpParaPara1m == (Alp4 + Alp5) / 2,
   AlpParaPara2p == (Alp1 + Alp4) / 2,
   AlpParaPara2m == - (Alp1 + Alp2) / 2,
   AlpPerpPara0p == - (Alp4 - Alp6) / 4,
   AlpPerpPara0m == (Alp2 - Alp3) / 2,
   AlpPerpPara1p == - (Alp2 - Alp5) / 2,
   AlpPerpPara1m == (Alp4 - Alp5) / 2,
   AlpPerpPara2p == (Alp1 - Alp4) / 2,
   AlpPerpPara2m == - (Alp1 - Alp2) / 2,
   AlpParaPerp0p == - (Alp4 - Alp6) / 2,
   AlpParaPerp0m == (Alp2 - Alp3) / 4,
```

```
    AlpParaPerp1p == (Alp2 - Alp5) / 4,
    AlpParaPerp1m == (Alp4 - Alp5) / 4,
    AlpParaPerp2p == (Alp1 - Alp4) / 4,
    AlpParaPerp2m == - (Alp1 - Alp2) / 4,
    AlpPerpPerp0p == (Alp4 + Alp6) / 4,
    AlpPerpPerp0m == (Alp2 + Alp3) / 4,
    AlpPerpPerp1p == (Alp2 + Alp5) / 4,
    AlpPerpPerp1m == (Alp4 + Alp5) / 4,
    AlpPerpPerp2p == (Alp1 + Alp4) / 4,
    AlpPerpPerp2m == - (Alp1 + Alp2) / 4};

BetDetRelations = {BetParaPara0p == 0,
    BetParaPara0m == Bet3 / 6,
    BetParaPara1p == (2 Bet1 + Bet3) / 3,
    BetParaPara1m == (Bet1 + 2 Bet2) / 3,
    BetParaPara2p == 0,
    BetParaPara2m == Bet1,
    BetPerpPara0p == 0,
    BetPerpPara0m == 0,
    BetPerpPara1p == - (Bet1 - Bet3) / 3,
    BetPerpPara1m == - (Bet1 - Bet2) / 3,
    BetPerpPara2p == 0,
    BetPerpPara2m == 0,
    BetParaPerp0p == 0,
    BetParaPerp0m == 0,
    BetParaPerp1p == - (Bet1 - Bet3) / 3,
    BetParaPerp1m == - (Bet1 - Bet2) / 3,
    BetParaPerp2p == 0,
    BetParaPerp2m == 0,
    BetPerpPerp0p == Bet2 / 2,
    BetPerpPerp0m == 0,
    BetPerpPerp1p == (Bet1 + 2 Bet3) / 6,
    BetPerpPerp1m == (2 Bet1 + Bet2) / 6,
    BetPerpPerp2p == Bet1 / 2,
    BetPerpPerp2m == 0};

AlpDeterminants = {AlpParaPara0p AlpPerpPerp0p - AlpParaPerp0p AlpPerpPara0p,
    AlpParaPara0m AlpPerpPerp0m - AlpParaPerp0m AlpPerpPara0m,
    AlpParaPara1p AlpPerpPerp1p - AlpParaPerp1p AlpPerpPara1p,
    AlpParaPara1m AlpPerpPerp1m - AlpParaPerp1m AlpPerpPara1m,
    AlpParaPara2p AlpPerpPerp2p - AlpParaPerp2p AlpPerpPara2p,
    AlpParaPara2m AlpPerpPerp2m - AlpParaPerp2m AlpPerpPara2m};
```

```
BetDeterminants = {BetParaPara0p BetPerpPerp0p - BetParaPerp0p BetPerpPara0p,
    BetParaPara0m BetPerpPerp0m - BetParaPerp0m BetPerpPara0m,
    BetParaPara1p BetPerpPerp1p - BetParaPerp1p BetPerpPara1p,
    BetParaPara1m BetPerpPerp1m - BetParaPerp1m BetPerpPara1m,
    BetParaPara2p BetPerpPerp2p - BetParaPerp2p BetPerpPara2p,
    BetParaPara2m BetPerpPerp2m - BetParaPerp2m BetPerpPara2m};

ToAlp = SolveConstants[AlpDetRelations,
     Join[AlpPerpPara, AlpPerpPerp, AlpParaPara, AlpParaPerp]][[1]];

ToBet = SolveConstants[BetDetRelations,
     Join[BetPerpPara, BetPerpPerp, BetParaPara, BetParaPerp]][[1]];

cAlpDetRelations = {cAlpParaPara0p == (cAlp4 + cAlp6) / 2,
    cAlpParaPara0m == (cAlp2 + cAlp3) / 2,
    cAlpParaPara1p == - (cAlp2 + cAlp5) / 2,
    cAlpParaPara1m == (cAlp4 + cAlp5) / 2,
    cAlpParaPara2p == (cAlp1 + cAlp4) / 2,
    cAlpParaPara2m == - (cAlp1 + cAlp2) / 2,
    cAlpPerpPara0p == - (cAlp4 - cAlp6) / 4,
    cAlpPerpPara0m == (cAlp2 - cAlp3) / 2,
    cAlpPerpPara1p == - (cAlp2 - cAlp5) / 2,
    cAlpPerpPara1m == (cAlp4 - cAlp5) / 2,
    cAlpPerpPara2p == (cAlp1 - cAlp4) / 2,
    cAlpPerpPara2m == - (cAlp1 - cAlp2) / 2,
    cAlpParaPerp0p == - (cAlp4 - cAlp6) / 2,
    cAlpParaPerp0m == (cAlp2 - cAlp3) / 4,
    cAlpParaPerp1p == (cAlp2 - cAlp5) / 4,
    cAlpParaPerp1m == (cAlp4 - cAlp5) / 4,
    cAlpParaPerp2p == (cAlp1 - cAlp4) / 4,
    cAlpParaPerp2m == - (cAlp1 - cAlp2) / 4,
    cAlpPerpPerp0p == (cAlp4 + cAlp6) / 4,
    cAlpPerpPerp0m == (cAlp2 + cAlp3) / 4,
    cAlpPerpPerp1p == (cAlp2 + cAlp5) / 4,
    cAlpPerpPerp1m == (cAlp4 + cAlp5) / 4,
    cAlpPerpPerp2p == (cAlp1 + cAlp4) / 4,
    cAlpPerpPerp2m == - (cAlp1 + cAlp2) / 4};

cBetDetRelations = {cBetParaPara0p == 0,
    cBetParaPara0m == cBet3 / 6,
    cBetParaPara1p == (2 cBet1 + cBet3) / 3,
    cBetParaPara1m == (cBet1 + 2 cBet2) / 3,
    cBetParaPara2p == 0,
```

```
    cBetParaPara2m == cBet1,
    cBetPerpPara0p == 0,
    cBetPerpPara0m == 0,
    cBetPerpPara1p == -(cBet1 - cBet3) / 3,
    cBetPerpPara1m == -(cBet1 - cBet2) / 3,
    cBetPerpPara2p == 0,
    cBetPerpPara2m == 0,
    cBetParaPerp0p == 0,
    cBetParaPerp0m == 0,
    cBetParaPerp1p == -(cBet1 - cBet3) / 3,
    cBetParaPerp1m == -(cBet1 - cBet2) / 3,
    cBetParaPerp2p == 0,
    cBetParaPerp2m == 0,
    cBetPerpPerp0p == cBet2 / 2,
    cBetPerpPerp0m == 0,
    cBetPerpPerp1p == (cBet1 + 2 cBet3) / 6,
    cBetPerpPerp1m == (2 cBet1 + cBet2) / 6,
    cBetPerpPerp2p == cBet1 / 2,
    cBetPerpPerp2m == 0};

cAlpDeterminants = {cAlpParaPara0p cAlpPerpPerp0p - cAlpParaPerp0p cAlpPerpPara0p,
    cAlpParaPara0m cAlpPerpPerp0m - cAlpParaPerp0m cAlpPerpPara0m,
    cAlpParaPara1p cAlpPerpPerp1p - cAlpParaPerp1p cAlpPerpPara1p,
    cAlpParaPara1m cAlpPerpPerp1m - cAlpParaPerp1m cAlpPerpPara1m,
    cAlpParaPara2p cAlpPerpPerp2p - cAlpParaPerp2p cAlpPerpPara2p,
    cAlpParaPara2m cAlpPerpPerp2m - cAlpParaPerp2m cAlpPerpPara2m};

cBetDeterminants = {cBetParaPara0p cBetPerpPerp0p - cBetParaPerp0p cBetPerpPara0p,
    cBetParaPara0m cBetPerpPerp0m - cBetParaPerp0m cBetPerpPara0m,
    cBetParaPara1p cBetPerpPerp1p - cBetParaPerp1p cBetPerpPara1p,
    cBetParaPara1m cBetPerpPerp1m - cBetParaPerp1m cBetPerpPara1m,
    cBetParaPara2p cBetPerpPerp2p - cBetParaPerp2p cBetPerpPara2p,
    cBetParaPara2m cBetPerpPerp2m - cBetParaPerp2m cBetPerpPara2m};

TocAlp = SolveConstants[cAlpDetRelations,
     Join[cAlpPerpPara, cAlpPerpPerp, cAlpParaPara, cAlpParaPerp]][[1]];

TocBet = SolveConstants[cBetDetRelations,
     Join[cBetPerpPara, cBetPerpPerp, cBetParaPara, cBetParaPerp]][[1]];
ClearBuild[];
```

build

Alternative human-readable projections $\{^{A}\check{\mathcal{P}}\}$, $\{^{E}\check{\mathcal{P}}\}$

build

In[ ]:=

```
(*Projections to break the field strengths up into canonical and non-
```

```
 canonical parts*)
DefTensor[PPerpTPerp[-e, a, b], M4];
DefTensor[PPerpTPara[-e, -f, a, b], M4];
DefTensor[PPerpRPerp[-e, -f, a, b, c], M4];
DefTensor[PPerpRPara[-e, -f, -g, a, b, c], M4];
PPerpTPerpDefinition = V[a] PPara[-e, b] /. PADMActivate // ToCanonical;
PPerpTPerpActivate =
  MakeRule[{PPerpTPerp[-e, a, b], Evaluate[PPerpTPerpDefinition]},
   MetricOn → All, ContractMetrics → True];
PPerpTParaDefinition = PPara[-e, a] PPara[-f, b] /. PADMActivate // ToCanonical;
PPerpTParaActivate =
  MakeRule[{PPerpTPara[-e, -f, a, b], Evaluate[PPerpTParaDefinition]},
   MetricOn → All, ContractMetrics → True];
PPerpRPerpDefinition = V[a] PPara[-e, b] PPara[-f, c] /. PADMActivate //
   ToCanonical;
PPerpRPerpActivate = MakeRule[{PPerpRPerp[-e, -f, a, b, c],
    Evaluate[PPerpRPerpDefinition]}, MetricOn → All, ContractMetrics → True];
PPerpRParaDefinition = PPara[-e, a] PPara[-f, b] PPara[-g, c] /. PADMActivate //
   ToCanonical;
PPerpRParaActivate = MakeRule[{PPerpRPara[-e, -f, -g, a, b, c],
    Evaluate[PPerpRParaDefinition]}, MetricOn → All, ContractMetrics → True];
PPerpADMTActivate = Join[PPerpTPerpActivate, PPerpTParaActivate];
PPerpADMRActivate = Join[PPerpRPerpActivate, PPerpRParaActivate];

(*Projection operators which define the O(3)
 decomposition of the canonical parts of field strengths*)
DefTensor[PPerpT0p[e, f], M4, PrintAs -> SymbolBuild[PTSymb, Spin0p]];
DefTensor[PPerpT1p[-a, -b, e, f], M4, PrintAs -> SymbolBuild[PTSymb, Spin1p]];
DefTensor[PPerpT1m[-a, e, f], M4, PrintAs -> SymbolBuild[PTSymb, Spin1m]];
DefTensor[PPerpT2p[-a, -b, e, f], M4, PrintAs -> SymbolBuild[PTSymb, Spin2p]];

DefTensor[PPerpR0p[e, f], M4, PrintAs -> SymbolBuild[PRSymb, Spin0p]];
DefTensor[PPerpR0m[e, f, g], M4, PrintAs -> SymbolBuild[PRSymb, Spin0m]];
DefTensor[PPerpR1p[-n, -m, e, f], M4, PrintAs -> SymbolBuild[PRSymb, Spin1p]];
DefTensor[PPerpR1m[-n, e, f, g], M4, PrintAs -> SymbolBuild[PRSymb, Spin1m]];
DefTensor[PPerpR2p[-n, -m, e, f], M4, PrintAs -> SymbolBuild[PRSymb, Spin2p]];
DefTensor[PPerpR2m[-n, -m, -o, e, f, g],
  M4, PrintAs -> SymbolBuild[PRSymb, Spin2m]];

PPerpT0pDefinition = PPara[e, f] /. PADMActivate // ToCanonical;
PPerpT1pDefinition =
  Antisymmetrize[PPara[-n, e] PPara[-m, f], {-n, -m}] /. PADMActivate //
   ToCanonical;
PPerpT1mDefinition = PPara[-n, e] /. PADMActivate // ToCanonical;
```

```
PPerpT2pDefinition = (Symmetrize[PPara[-n, e] PPara[-m, f], {-n, -m}] -
       (1/3) PPara[-n, -m] PPara[e, f]) /. PADMActivate // ToCanonical;

PPerpR0pDefinition = -PPara[e, f] /. PADMActivate // ToCanonical;
PPerpR0mDefinition =
  PPara[-i, e] PPara[-j, f] PPara[-k, g] epsilonG[i, j, k, p] V[-p] /.
    PADMActivate // ToCanonical;
PPerpR1pDefinition = Antisymmetrize[PPara[-n, e] PPara[-m, f], {-n, -m}] /.
    PADMActivate // ToCanonical;
PPerpR1mDefinition = PPara[e, g] PPara[-n, f] /. PADMActivate // ToCanonical;
PPerpR2pDefinition = (Symmetrize[PPara[-n, e] PPara[-m, f], {-n, -m}] -
       (1/3) PPara[-n, -m] PPara[e, f]) /. PADMActivate // ToCanonical;
PPerpR2mDefinition = PPara[-a, i] PPara[-b, j] PPara[-c, k] PPara[e, -l]
     PPara[f, -n] PPara[d, -m] (3/4) ((1/3) (2 G[-i, l] G[-j, n] G[-k, m] -
          G[-j, l] G[-k, n] G[-i, m] - G[-k, l] G[-i, n] G[-j, m]) - Antisymmetrize[
        G[-i, -k] G[-j, n] G[l, m], {-i, -j}]) /. PADMActivate // ToCanonical;

PPerpT0pActivate = MakeRule[{PPerpT0p[e, f], Evaluate[PPerpT0pDefinition]},
   MetricOn → All, ContractMetrics → True];
PPerpT1pActivate = MakeRule[{PPerpT1p[-n, -m, e, f],
    Evaluate[PPerpT1pDefinition]}, MetricOn → All, ContractMetrics → True];
PPerpT1mActivate = MakeRule[{PPerpT1m[-n, e], Evaluate[PPerpT1mDefinition]},
   MetricOn → All, ContractMetrics → True];
PPerpT2pActivate = MakeRule[{PPerpT2p[-n, -m, e, f],
    Evaluate[PPerpT2pDefinition]}, MetricOn → All, ContractMetrics → True];

PPerpR0pActivate = MakeRule[{PPerpR0p[e, f], Evaluate[PPerpR0pDefinition]},
   MetricOn → All, ContractMetrics → True];
PPerpR0mActivate = MakeRule[{PPerpR0m[e, f, g], Evaluate[PPerpR0mDefinition]},
   MetricOn → All, ContractMetrics → True];
PPerpR1pActivate = MakeRule[{PPerpR1p[-n, -m, e, f],
    Evaluate[PPerpR1pDefinition]}, MetricOn → All, ContractMetrics → True];
PPerpR1mActivate = MakeRule[{PPerpR1m[-n, e, f, g], Evaluate[PPerpR1mDefinition]},
   MetricOn → All, ContractMetrics → True];
PPerpR2pActivate = MakeRule[{PPerpR2p[-n, -m, e, f],
    Evaluate[PPerpR2pDefinition]}, MetricOn → All, ContractMetrics → True];
PPerpR2mActivate = MakeRule[{PPerpR2m[-a, -b, -c, e, f, d],
    Evaluate[PPerpR2mDefinition]}, MetricOn → All, ContractMetrics → True];

(*These rules then expand those canonical
 field strength O(3) projection operators*)
PPerpO3TActivate = Join[PPerpT0pActivate,
   PPerpT1pActivate, PPerpT1mActivate, PPerpT2pActivate];
```

```
PPerpO3RActivate = Join[PPerpR0pActivate, PPerpR0mActivate,
    PPerpR1pActivate, PPerpR1mActivate, PPerpR2pActivate, PPerpR2mActivate];
ClearBuild[];
```

build

Basic form ϕbJ^P, ϕAJ^P

build

In[•]:=

```
PhiBSymb = "ϕb";
DefTensor[PhiB0p[], M4, PrintAs → SymbolBuild[PhiBSymb, Spin0p]];
DeclareOrder[PhiB0p[], 1];
DefTensor[PhiB1p[-a, -b], M4, Antisymmetric[{-a, -b}],
 PrintAs → SymbolBuild[PhiBSymb, Spin1p]];

  DeclareOrder[PhiB1p[-a, -b], 1];
DefTensor[PhiB1m[-a], M4, PrintAs → SymbolBuild[PhiBSymb, Spin1m]];
DeclareOrder[PhiB1m[-a], 1];
DefTensor[PhiB2p[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[PhiBSymb, Spin2p]];
DeclareOrder[PhiB2p[-a, -b], 1];
PhiASymb = "ϕA";
DefTensor[PhiA0p[], M4, PrintAs → SymbolBuild[PhiASymb, Spin0p]];
DeclareOrder[PhiA0p[], 1, "IsUnityWithEHTerm" → True];
DefTensor[PhiA0m[], M4, PrintAs → SymbolBuild[PhiASymb, Spin0m]];
DeclareOrder[PhiA0m[], 1];
DefTensor[PhiA1p[-a, -b], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[PhiASymb, Spin1p]];
DeclareOrder[PhiA1p[-a, -b], 1];
DefTensor[PhiA1m[-a], M4, PrintAs → SymbolBuild[PhiASymb, Spin1m]];
DeclareOrder[PhiA1m[-a], 1];
DefTensor[PhiA2p[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[PhiASymb, Spin2p]];
DeclareOrder[PhiA2p[-a, -b], 1];
DefTensor[PhiA2m[-a, -b, -c], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[PhiASymb, Spin2m]];
DeclareOrder[PhiA2m[-a, -b, -c], 1];
AutomaticRules[PhiA2m,
  MakeRule[{PhiA2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[PhiA2m, MakeRule[{epsilonG[a, b, c, d] PhiA2m[-a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];

DefTensor[BPhi[-a, -c], M4];
DeclareOrder[BPhi[-a, -c], 1];
BPhiDefinition = Ji[] BPi[-i, z] G3[-z, a] B[-k, -a] -
```

```
    4 V[g] B[-k, -o] G3[o, -z] H[h, z]
     (Bet1 PT1[-i, -g, -h, a, c, d] + Bet2 PT2[-i, -g, -h, a, c, d] +
       Bet3 PT3[-i, -g, -h, a, c, d]) PPara[-c, m] PPara[-d, n] T[-a, -m, -n] -
    2 V[g] B[-k, -o] G3[o, -z] H[h, z] (cBet1 PT1[-i, -g, -h, a, c, d] +
       cBet2 PT2[-i, -g, -h, a, c, d] + cBet3 PT3[-i, -g, -h, a, c, d])
     PPara[-c, m] PPara[-d, n] TLambda[-a, -m, -n] -
    2 V[g] B[-k, -o] G3[o, -z] H[h, z] (cBet1 PT1[-i, -g, -h, a, c, d] +
       cBet2 PT2[-i, -g, -h, a, c, d] + cBet3 PT3[-i, -g, -h, a, c, d])
     (PPerp[-c, m] PPara[-d, n] TLambda[-a, -m, -n] +
       PPara[-c, m] PPerp[-d, n] TLambda[-a, -m, -n]);
BPhiActivate = MakeRule[{BPhi[-i, -k], Evaluate[BPhiDefinition]},
    MetricOn → All, ContractMetrics → True];

DefTensor[APhi[-a, -b, -c], M4, Antisymmetric[{-a, -b}]];
DeclareOrder[APhi[-a, -b, -c], 1, "IsUnityWithEHTerm" → True];
APhiDefinition = Ji[] APi[-i, -j, z] G3[-z, a] B[-k, -a] +
    2 Alp0 Antisymmetrize[ V[-i] PPara[-j, -k], {-i, -j}] -
    8 V[g] B[-k, -o] G3[o, -z] H[h, z]
     (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
       Alp3 PR3[-i, -j, -g, -h, a, b, c, d] + Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
       Alp5 PR5[-i, -j, -g, -h, a, b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d])
     PPara[-c, m] PPara[-d, n] R[-a, -b, -m, -n] - 4 V[g] B[-k, -o] G3[o, -z] H[h, z]
     (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] + cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
       cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] + cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
       cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] + cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
     PPara[-c, m] PPara[-d, n] RLambda[-a, -b, -m, -n] -
    4 V[g] B[-k, -o] G3[o, -z] H[h, z]
     (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] + cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
       cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] + cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
       cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] + cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
     (PPerp[-c, m] PPara[-d, n] RLambda[-a, -b, -m, -n] +
       PPara[-c, m] PPerp[-d, n] RLambda[-a, -b, -m, -n]);
APhiActivate = MakeRule[{APhi[-i, -j, -k], Evaluate[APhiDefinition]},
    MetricOn → All, ContractMetrics → True];
ClearBuild[];
```

CanonicalPhiToggle

In[•]:=

```
IfBuild["CanonicalPhiToggle",
  PhiB0pDefinition = PB0p[e, f] PBPara[-e, -f, a, c] BPhi[-a, -c] /. BPhiActivate //
    ActivateGeneralO3Projections;
  PhiB1pDefinition = PB1p[-n, -m, e, f] PBPara[-e, -f, a, c] BPhi[-a, -c] /.
     BPhiActivate // ActivateGeneralO3Projections;
  PhiB2pDefinition = PB2p[-n, -m, e, f] PBPara[-e, -f, a, c] BPhi[-a, -c] /.
```

```
    BPhiActivate // ActivateGeneralO3Projections;
PhiB1mDefinition = PB1m[-n, f] PBPerp[-f, a, c] BPhi[-a, -c] /. BPhiActivate //
  ActivateGeneralO3Projections;

PhiA0pDefinition =
 PA0p[e, f] PAPerp[-e, -f, a, b, c] APhi[-a, -b, -c] /. APhiActivate //
  ActivateGeneralO3Projections;
PhiA1pDefinition = PA1p[-n, -m, e, f] PAPerp[-e, -f, a, b, c] APhi[-a, -b, -c] /.
    APhiActivate // ActivateGeneralO3Projections;
PhiA2pDefinition = PA2p[-n, -m, e, f] PAPerp[-e, -f, a, b, c] APhi[-a, -b, -c] /.
    APhiActivate // ActivateGeneralO3Projections;
PhiA0mDefinition = PA0m[d, e, f] PAPara[-d, -e, -f, a, b, c] APhi[-a, -b, -c] /.
    APhiActivate // ActivateGeneralO3Projections;
PhiA1mDefinition = PA1m[-n, d, e, f] PAPara[-d, -e, -f, a, b, c] APhi[-a, -b, -c] /.
    APhiActivate // ActivateGeneralO3Projections;
PhiA2mDefinition = PA2m[-n, -m, -o, d, e, f] PAPara[-d, -e, -f, a, b, c]
     APhi[-a, -b, -c] /. APhiActivate // ActivateGeneralO3Projections;

PhiB0pActivate = MakeRule[{PhiB0p[], Scalar[Evaluate[PhiB0pDefinition]]},
  MetricOn → All, ContractMetrics → True];
PhiB1pActivate = MakeRule[{PhiB1p[-n, -m], Evaluate[PhiB1pDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiB1mActivate = MakeRule[{PhiB1m[-n], Evaluate[PhiB1mDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiB2pActivate = MakeRule[{PhiB2p[-n, -m], Evaluate[PhiB2pDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiA0pActivate = MakeRule[{PhiA0p[], Scalar[Evaluate[PhiA0pDefinition]]},
  MetricOn → All, ContractMetrics → True];
PhiA0mActivate = MakeRule[{PhiA0m[], Scalar[Evaluate[PhiA0mDefinition]]},
  MetricOn → All, ContractMetrics → True];
PhiA1pActivate = MakeRule[{PhiA1p[-n, -m], Evaluate[PhiA1pDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiA1mActivate = MakeRule[{PhiA1m[-n], Evaluate[PhiA1mDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiA2pActivate = MakeRule[{PhiA2p[-n, -m], Evaluate[PhiA2pDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiA2mActivate = MakeRule[{PhiA2m[-n, -m, -o], Evaluate[PhiA2mDefinition]},
  MetricOn → All, ContractMetrics → True];

PhiActivate = Join[PhiB0pActivate, PhiB1pActivate,
  PhiB1mActivate, PhiB2pActivate, PhiA0pActivate, PhiA0mActivate,
  PhiA1pActivate, PhiA1mActivate, PhiA2pActivate, PhiA2mActivate];

DumpSave[BinaryLocation[], {PhiActivate}];
```

```
  ClearBuild[];
 ];
```

build

In[•]:=
```
OpenLastCache[];
```

build

Basic form $\neg \phi bJ^P, \neg \phi AJ^P$

build

In[•]:=
```
BPhiNonCanonicalDefinition = 4 V[g] B[-k, -o] G3[o, -z]
    H[h, z] (Bet1 PT1[-i, -g, -h, a, c, d] + Bet2 PT2[-i, -g, -h, a, c, d] +
      Bet3 PT3[-i, -g, -h, a, c, d]) (PPerp[-c, m] PPara[-d, n] T[-a, -m, -n] +
      PPara[-c, m] PPerp[-d, n] T[-a, -m, -n]);
BPhiNonCanonicalActivate = MakeRule[{BPhi[-i, -k], Evaluate[
      BPhiNonCanonicalDefinition]}, MetricOn → All, ContractMetrics → True];

APhiNonCanonicalDefinition = 8 V[g] B[-k, -o] G3[o, -z] H[h, z]
    (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
      Alp3 PR3[-i, -j, -g, -h, a, b, c, d] + Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
      Alp5 PR5[-i, -j, -g, -h, a, b, c, d] + Alp6 PR6[-i, -j, -g, -h, a, b, c, d])
    (PPerp[-c, m] PPara[-d, n] R[-a, -b, -m, -n] +
      PPara[-c, m] PPerp[-d, n] R[-a, -b, -m, -n]);
APhiNonCanonicalActivate = MakeRule[{APhi[-i, -j, -k], Evaluate[
      APhiNonCanonicalDefinition]}, MetricOn → All, ContractMetrics → True];
ClearBuild[];
```

NonCanonicalPhiToggle

In[•]:=
```
IfBuild["NonCanonicalPhiToggle",
  PhiNonCanonicalB0pDefinition =
   PB0p[e, f] PBPara[-e, -f, a, c] BPhi[-a, -c] /. BPhiNonCanonicalActivate //
    ActivateGeneralO3Projections;
  PhiNonCanonicalB1pDefinition = PB1p[-n, -m, e, f] PBPara[-e, -f, a, c]
      BPhi[-a, -c] /. BPhiNonCanonicalActivate // ActivateGeneralO3Projections;
  PhiNonCanonicalB2pDefinition = PB2p[-n, -m, e, f] PBPara[-e, -f, a, c]
      BPhi[-a, -c] /. BPhiNonCanonicalActivate // ActivateGeneralO3Projections;
  PhiNonCanonicalB1mDefinition = PB1m[-n, f] PBPerp[-f, a, c] BPhi[-a, -c] /.
     BPhiNonCanonicalActivate // ActivateGeneralO3Projections;

  PhiNonCanonicalA0pDefinition =
   PA0p[e, f] PAPerp[-e, -f, a, b, c] APhi[-a, -b, -c] /.
     APhiNonCanonicalActivate // ActivateGeneralO3Projections;
  PhiNonCanonicalA1pDefinition = PA1p[-n, -m, e, f] PAPerp[-e, -f, a, b, c] APhi[
       -a, -b, -c] /. APhiNonCanonicalActivate // ActivateGeneralO3Projections;
```

```
PhiNonCanonicalA2pDefinition = PA2p[-n, -m, e, f] PAPerp[-e, -f, a, b, c] APhi[
     -a, -b, -c] /. APhiNonCanonicalActivate // ActivateGeneralO3Projections;
PhiNonCanonicalA0mDefinition = PA0m[d, e, f] PAPara[-d, -e, -f, a, b, c] APhi[
     -a, -b, -c] /. APhiNonCanonicalActivate // ActivateGeneralO3Projections;
PhiNonCanonicalA1mDefinition = PA1m[-n, d, e, f] PAPara[-d, -e, -f, a, b, c] APhi[
     -a, -b, -c] /. APhiNonCanonicalActivate // ActivateGeneralO3Projections;
PhiNonCanonicalA2mDefinition = PA2m[-n, -m, -o, d, e, f]
    PAPara[-d, -e, -f, a, b, c] APhi[-a, -b, -c] /.
   APhiNonCanonicalActivate // ActivateGeneralO3Projections;

PhiNonCanonicalB0pActivate =
 MakeRule[{PhiB0p[], Scalar[Evaluate[PhiNonCanonicalB0pDefinition]]},
  MetricOn → All, ContractMetrics → True];
PhiNonCanonicalB1pActivate = MakeRule[
  {PhiB1p[-n, -m], Evaluate[PhiNonCanonicalB1pDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiNonCanonicalB1mActivate = MakeRule[
  {PhiB1m[-n], Evaluate[PhiNonCanonicalB1mDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiNonCanonicalB2pActivate = MakeRule[
  {PhiB2p[-n, -m], Evaluate[PhiNonCanonicalB2pDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiNonCanonicalA0pActivate = MakeRule[
  {PhiA0p[], Scalar[Evaluate[PhiNonCanonicalA0pDefinition]]},
  MetricOn → All, ContractMetrics → True];
PhiNonCanonicalA0mActivate = MakeRule[
  {PhiA0m[], Scalar[Evaluate[PhiNonCanonicalA0mDefinition]]},
  MetricOn → All, ContractMetrics → True];
PhiNonCanonicalA1pActivate = MakeRule[
  {PhiA1p[-n, -m], Evaluate[PhiNonCanonicalA1pDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiNonCanonicalA1mActivate = MakeRule[
  {PhiA1m[-n], Evaluate[PhiNonCanonicalA1mDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiNonCanonicalA2pActivate = MakeRule[
  {PhiA2p[-n, -m], Evaluate[PhiNonCanonicalA2pDefinition]},
  MetricOn → All, ContractMetrics → True];
PhiNonCanonicalA2mActivate = MakeRule[
  {PhiA2m[-n, -m, -o], Evaluate[PhiNonCanonicalA2mDefinition]},
  MetricOn → All, ContractMetrics → True];

PhiNonCanonicalActivate =
 Join[PhiNonCanonicalB0pActivate, PhiNonCanonicalB1pActivate,
  PhiNonCanonicalB1mActivate, PhiNonCanonicalB2pActivate,
```

```
    PhiNonCanonicalA0pActivate, PhiNonCanonicalA0mActivate,
    PhiNonCanonicalA1pActivate, PhiNonCanonicalA1mActivate,
    PhiNonCanonicalA2pActivate, PhiNonCanonicalA2mActivate];

  DumpSave[BinaryLocation[], {PhiNonCanonicalActivate}];
  ClearBuild[];
 ];
```

build

In[•]:=

```
OpenLastCache[];
```

build

Define χbJ^P, χAJ^P

build

In[•]:=

```
ChiBSymb = "χb";
DefTensor[ChiB0p[], M4, PrintAs → SymbolBuild[ChiBSymb, Spin0p]];
DeclareOrder[ChiB0p[], 1];
DefTensor[ChiB1p[-a, -b], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[ChiBSymb, Spin1p]];
DeclareOrder[ChiB1p[-a, -b], 1];
DefTensor[ChiB1m[-a], M4, PrintAs → SymbolBuild[ChiBSymb, Spin1m]];
DeclareOrder[ChiB1m[-a], 1];
DefTensor[ChiB2p[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[ChiBSymb, Spin2p]];
DeclareOrder[ChiB2p[-a, -b], 1];
ChiASymb = "χa";
DefTensor[ChiA0p[], M4, PrintAs → SymbolBuild[ChiASymb, Spin0p]];
DeclareOrder[ChiA0p[], 1];
DefTensor[ChiA0m[], M4, PrintAs → SymbolBuild[ChiASymb, Spin0m]];
DeclareOrder[ChiA0m[], 1];
DefTensor[ChiA1p[-a, -b], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[ChiASymb, Spin1p]];
DeclareOrder[ChiA1p[-a, -b], 1];
DefTensor[ChiA1m[-a], M4, PrintAs → SymbolBuild[ChiASymb, Spin1m]];
DeclareOrder[ChiA1m[-a], 1];
DefTensor[ChiA2p[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[ChiASymb, Spin2p]];
DeclareOrder[ChiA2p[-a, -b], 1];
DefTensor[ChiA2m[-a, -b, -c], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[ChiASymb, Spin2m]];
DeclareOrder[ChiA2m[-a, -b, -c], 1];
AutomaticRules[ChiA2m,
  MakeRule[{ChiA2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[ChiA2m, MakeRule[{epsilonG[a, b, c, d] ChiA2m[-a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
ClearBuild[];
```

build

Basic form $\chi^{\perp}$ bJ^P, $\chi^{\perp}$ AJ^P

build

In[•]:=

```
ChiPerpBSymb = "χb⊥";
DefTensor[ChiPerpB0p[], M4, PrintAs → SymbolBuild[ChiPerpBSymb, Spin0p]];
DeclareOrder[ChiPerpB0p[], 1];
```

```
DefTensor[ChiPerpB1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[ChiPerpBSymb, Spin1p]];
DeclareOrder[ChiPerpB1p[-a, -b], 1];
DefTensor[ChiPerpB1m[-a], M4, PrintAs → SymbolBuild[ChiPerpBSymb, Spin1m]];
DeclareOrder[ChiPerpB1m[-a], 1];
DefTensor[ChiPerpB2p[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[ChiPerpBSymb, Spin2p]];
DeclareOrder[ChiPerpB2p[-a, -b], 1];
ChiPerpASymb = "χ⁺_𝔄";
DefTensor[ChiPerpA0p[], M4, PrintAs → SymbolBuild[ChiPerpASymb, Spin0p]];
DeclareOrder[ChiPerpA0p[], 1];
DefTensor[ChiPerpA0m[], M4, PrintAs → SymbolBuild[ChiPerpASymb, Spin0m]];
DeclareOrder[ChiPerpA0m[], 1];
DefTensor[ChiPerpA1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[ChiPerpASymb, Spin1p]];
DeclareOrder[ChiPerpA1p[-a, -b], 1];
DefTensor[ChiPerpA1m[-a], M4, PrintAs → SymbolBuild[ChiPerpASymb, Spin1m]];
DeclareOrder[ChiPerpA1m[-a], 1];
DefTensor[ChiPerpA2p[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[ChiPerpASymb, Spin2p]];
DeclareOrder[ChiPerpA2p[-a, -b], 1];
DefTensor[ChiPerpA2m[-a, -b, -c], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[ChiPerpASymb, Spin2m]];
DeclareOrder[ChiPerpA2m[-a, -b, -c], 1];
AutomaticRules[ChiPerpA2m,
  MakeRule[{ChiPerpA2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[ChiPerpA2m, MakeRule[{epsilonG[a, b, c, d] ChiPerpA2m[-a, -b, -c],
    0}, MetricOn → All, ContractMetrics → True]];

DefTensor[BChiPerp[-a, -c], M4];
DeclareOrder[BChiPerp[-a, -c], 1];
BChiPerpDefinition = Ji[] BPi[-i, z] G3[-z, a] B[-k, -a] -
   2 V[g] B[-k, -o] G3[o, -z] H[h, z] (cBet1 PT1[-i, -g, -h, a, c, d] +
      cBet2 PT2[-i, -g, -h, a, c, d] + cBet3 PT3[-i, -g, -h, a, c, d])
    PPara[-c, m] PPara[-d, n] TLambda[-a, -m, -n] -
   2 V[g] B[-k, -o] G3[o, -z] H[h, z] (cBet1 PT1[-i, -g, -h, a, c, d] +
      cBet2 PT2[-i, -g, -h, a, c, d] + cBet3 PT3[-i, -g, -h, a, c, d])
    (PPerp[-c, m] PPara[-d, n] TLambda[-a, -m, -n] +
      PPara[-c, m] PPerp[-d, n] TLambda[-a, -m, -n]);
BChiPerpActivate = MakeRule[{BChiPerp[-i, -k], Evaluate[BChiPerpDefinition]},
   MetricOn → All, ContractMetrics → True];

DefTensor[AChiPerp[-a, -b, -c], M4, Antisymmetric[{-a, -b}]];
```

```
DeclareOrder[AChiPerp[-a, -b, -c], 1];
AChiPerpDefinition =
  Ji[] APi[-i, -j, z] G3[-z, a] B[-k, -a] - 4 V[g] B[-k, -o] G3[o, -z] H[h, z]
    (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] + cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
      cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] + cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
      cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] + cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
    PPara[-c, m] PPara[-d, n] RLambda[-a, -b, -m, -n] -
   4 V[g] B[-k, -o] G3[o, -z] H[h, z]
    (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] + cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
      cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] + cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
      cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] + cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
    (PPerp[-c, m] PPara[-d, n] RLambda[-a, -b, -m, -n] +
      PPara[-c, m] PPerp[-d, n] RLambda[-a, -b, -m, -n]);
AChiPerpActivate = MakeRule[{AChiPerp[-i, -j, -k], Evaluate[AChiPerpDefinition]},
   MetricOn → All, ContractMetrics → True];
ClearBuild[];
```

ChiPerpToggle

In[ ]:=

```
IfBuild["ChiPerpToggle",
  ChiPerpB0pDefinition =
   PB0p[e, f] PBPara[-e, -f, a, c] BChiPerp[-a, -c] /. BChiPerpActivate //
    ActivateGeneralO3Projections;
  ChiPerpB1pDefinition = PB1p[-n, -m, e, f] PBPara[-e, -f, a, c] BChiPerp[-a, -c] /.
     BChiPerpActivate // ActivateGeneralO3Projections;
  ChiPerpB2pDefinition = PB2p[-n, -m, e, f] PBPara[-e, -f, a, c] BChiPerp[-a, -c] /.
     BChiPerpActivate // ActivateGeneralO3Projections;
  ChiPerpB1mDefinition = PB1m[-n, f] PBPerp[-f, a, c] BChiPerp[-a, -c] /.
     BChiPerpActivate // ActivateGeneralO3Projections;

  ChiPerpA0pDefinition = PA0p[e, f] PAPerp[-e, -f, a, b, c] AChiPerp[-a, -b, -c] /.
     AChiPerpActivate // ActivateGeneralO3Projections;
  ChiPerpA1pDefinition = PA1p[-n, -m, e, f] PAPerp[-e, -f, a, b, c]
      AChiPerp[-a, -b, -c] /. AChiPerpActivate // ActivateGeneralO3Projections;
  ChiPerpA2pDefinition = PA2p[-n, -m, e, f] PAPerp[-e, -f, a, b, c]
      AChiPerp[-a, -b, -c] /. AChiPerpActivate // ActivateGeneralO3Projections;
  ChiPerpA0mDefinition = PA0m[d, e, f] PAPara[-d, -e, -f, a, b, c]
      AChiPerp[-a, -b, -c] /. AChiPerpActivate // ActivateGeneralO3Projections;
  ChiPerpA1mDefinition = PA1m[-n, d, e, f] PAPara[-d, -e, -f, a, b, c]
      AChiPerp[-a, -b, -c] /. AChiPerpActivate // ActivateGeneralO3Projections;
  ChiPerpA2mDefinition = PA2m[-n, -m, -o, d, e, f] PAPara[-d, -e, -f, a, b, c]
      AChiPerp[-a, -b, -c] /. AChiPerpActivate // ActivateGeneralO3Projections;

  ChiPerpB0pActivate =
   MakeRule[{ChiPerpB0p[], Scalar[Evaluate[ChiPerpB0pDefinition]]},
```

```
    MetricOn → All, ContractMetrics → True];
  ChiPerpB1pActivate = MakeRule[{ChiPerpB1p[-n, -m],
     Evaluate[ChiPerpB1pDefinition]}, MetricOn → All, ContractMetrics → True];
  ChiPerpB1mActivate = MakeRule[{ChiPerpB1m[-n], Evaluate[ChiPerpB1mDefinition]},
    MetricOn → All, ContractMetrics → True];
  ChiPerpB2pActivate = MakeRule[{ChiPerpB2p[-n, -m],
     Evaluate[ChiPerpB2pDefinition]}, MetricOn → All, ContractMetrics → True];
  ChiPerpA0pActivate = MakeRule[{ChiPerpA0p[], Scalar[
      Evaluate[ChiPerpA0pDefinition]]}, MetricOn → All, ContractMetrics → True];
  ChiPerpA0mActivate = MakeRule[{ChiPerpA0m[], Scalar[
      Evaluate[ChiPerpA0mDefinition]]}, MetricOn → All, ContractMetrics → True];
  ChiPerpA1pActivate = MakeRule[{ChiPerpA1p[-n, -m],
     Evaluate[ChiPerpA1pDefinition]}, MetricOn → All, ContractMetrics → True];
  ChiPerpA1mActivate = MakeRule[{ChiPerpA1m[-n], Evaluate[ChiPerpA1mDefinition]},
    MetricOn → All, ContractMetrics → True];
  ChiPerpA2pActivate = MakeRule[{ChiPerpA2p[-n, -m],
     Evaluate[ChiPerpA2pDefinition]}, MetricOn → All, ContractMetrics → True];
  ChiPerpA2mActivate = MakeRule[{ChiPerpA2m[-n, -m, -o],
     Evaluate[ChiPerpA2mDefinition]}, MetricOn → All, ContractMetrics → True];

  ChiPerpActivate = Join[ChiPerpB0pActivate,
    ChiPerpB1pActivate, ChiPerpB1mActivate, ChiPerpB2pActivate,
    ChiPerpA0pActivate, ChiPerpA0mActivate, ChiPerpA1pActivate,
    ChiPerpA1mActivate, ChiPerpA2pActivate, ChiPerpA2mActivate];

  DumpSave[BinaryLocation[], {ChiPerpActivate}];
  ClearBuild[];
 ];
```

build

In[•]:=
```
OpenLastCache[];
```

build

Basic form $\chi^{\vDash}$ bJ^P, $\chi^{\vDash}$ AJ^P

build

In[•]:=
```
ChiSingBSymb = "χ_b^⊨";
DefTensor[ChiSingB0p[], M4, PrintAs → SymbolBuild[ChiSingBSymb, Spin0p]];
DeclareOrder[ChiSingB0p[], 1];
DefTensor[ChiSingB1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[ChiSingBSymb, Spin1p]];
DeclareOrder[ChiSingB1p[-a, -b], 1];
DefTensor[ChiSingB1m[-a], M4, PrintAs → SymbolBuild[ChiSingBSymb, Spin1m]];
DeclareOrder[ChiSingB1m[-a], 1];
```

```
DefTensor[ChiSingB2p[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[ChiSingBSymb, Spin2p]];
DeclareOrder[ChiSingB2p[-a, -b], 1];
ChiSingASymb = "χ_A^F";
DefTensor[ChiSingA0p[], M4, PrintAs → SymbolBuild[ChiSingASymb, Spin0p]];
DeclareOrder[ChiSingA0p[], 1];
DefTensor[ChiSingA0m[], M4, PrintAs → SymbolBuild[ChiSingASymb, Spin0m]];
DeclareOrder[ChiSingA0m[], 1];
DefTensor[ChiSingA1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[ChiSingASymb, Spin1p]];
DeclareOrder[ChiSingA1p[-a, -b], 1];
DefTensor[ChiSingA1m[-a], M4, PrintAs → SymbolBuild[ChiSingASymb, Spin1m]];
DeclareOrder[ChiSingA1m[-a], 1];
DefTensor[ChiSingA2p[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[ChiSingASymb, Spin2p]];
DeclareOrder[ChiSingA2p[-a, -b], 1];
DefTensor[ChiSingA2m[-a, -b, -c], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[ChiSingASymb, Spin2m]];
DeclareOrder[ChiSingA2m[-a, -b, -c], 1];
AutomaticRules[ChiSingA2m,
  MakeRule[{ChiSingA2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[ChiSingA2m, MakeRule[{epsilonG[a, b, c, d] ChiSingA2m[-a, -b, -c],
    0}, MetricOn → All, ContractMetrics → True]];

DefTensor[BChiSingExtra[-a, -c], M4];
DeclareOrder[BChiSingExtra[-a, -c], 1];
BChiSingExtraDefinition = 4 V[g] B[-k, -o] G3[o, -z] H[h, z]
   (cBet1 PT1[-i, -g, -h, a, c, d] + cBet2 PT2[-i, -g, -h, a, c, d] +
     cBet3 PT3[-i, -g, -h, a, c, d]) PPara[-c, m] PPara[-d, n] T[-a, -m, -n];
BChiSingExtraActivate = MakeRule[{BChiSingExtra[-i, -k],
    Evaluate[BChiSingExtraDefinition]}, MetricOn → All, ContractMetrics → True];

DefTensor[AChiSingExtra[-a, -b, -c], M4, Antisymmetric[{-a, -b}]];
DeclareOrder[AChiSingExtra[-a, -b, -c], 1];
AChiSingExtraDefinition = 8 V[g] B[-k, -o] G3[o, -z] H[h, z]
   (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] + cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
     cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] + cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
     cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] + cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
   PPara[-c, m] PPara[-d, n] R[-a, -b, -m, -n];
AChiSingExtraActivate = MakeRule[{AChiSingExtra[-i, -j, -k],
    Evaluate[AChiSingExtraDefinition]}, MetricOn → All, ContractMetrics → True];
ClearBuild[];
```

ChiSingToggle

In[•]:=

```
IfBuild["ChiSingToggle",
```

```
ChiSingExtraB1pDefinition =
 (BetPerpPerp1p / cBetPerpPerp1p) PB1p[-n, -m, e, f] PBPara[-e, -f, a, c]
      BChiSingExtra[-a, -c] /. ToBet /. TocBet /.
   BChiSingExtraActivate // ActivateGeneralO3Projections;
ChiSingExtraB1mDefinition = (BetPerpPerp1m / cBetPerpPerp1m) PB1m[-n, f]
      PBPerp[-f, a, c] BChiSingExtra[-a, -c] /. ToBet /. TocBet /.
   BChiSingExtraActivate // ActivateGeneralO3Projections;

ChiSingExtraA0pDefinition =
 (AlpPerpPerp0p / cAlpPerpPerp0p) PA0p[e, f] PAPerp[-e, -f, a, b, c]
      AChiSingExtra[-a, -b, -c] /. ToAlp /. TocAlp /.
   AChiSingExtraActivate // ActivateGeneralO3Projections;
ChiSingExtraA1pDefinition = (AlpPerpPerp1p / cAlpPerpPerp1p) PA1p[-n, -m, e, f]
      PAPerp[-e, -f, a, b, c] AChiSingExtra[-a, -b, -c] /. ToAlp /. TocAlp /.
   AChiSingExtraActivate // ActivateGeneralO3Projections;
ChiSingExtraA2pDefinition = (AlpPerpPerp2p / cAlpPerpPerp2p) PA2p[-n, -m, e, f]
      PAPerp[-e, -f, a, b, c] AChiSingExtra[-a, -b, -c] /. ToAlp /. TocAlp /.
   AChiSingExtraActivate // ActivateGeneralO3Projections;
ChiSingExtraA0mDefinition = (AlpPerpPerp0m / cAlpPerpPerp0m) PA0m[d, e, f]
      PAPara[-d, -e, -f, a, b, c] AChiSingExtra[-a, -b, -c] /. ToAlp /. TocAlp /.
   AChiSingExtraActivate // ActivateGeneralO3Projections;
ChiSingExtraA1mDefinition = (AlpPerpPerp1m / cAlpPerpPerp1m) PA1m[-n, d, e, f]
      PAPara[-d, -e, -f, a, b, c] AChiSingExtra[-a, -b, -c] /. ToAlp /. TocAlp /.
   AChiSingExtraActivate // ActivateGeneralO3Projections;
ChiSingExtraA2mDefinition = (AlpPerpPerp2m / cAlpPerpPerp2m) PA2m[-n, -m, -o,
       d, e, f] PAPara[-d, -e, -f, a, b, c] AChiSingExtra[-a, -b, -c] /. ToAlp /.
    TocAlp /. AChiSingExtraActivate // ActivateGeneralO3Projections;

ChiSingB1pDefinition =
 PhiB1p[-n, -m] + ChiSingExtraB1pDefinition /. PhiActivate // NoScalar //
  ToNewCanonical;
ChiSingB1mDefinition = PhiB1m[-n] + ChiSingExtraB1mDefinition /. PhiActivate //
   NoScalar // ToNewCanonical;

ChiSingA0pDefinition =
 PhiA0p[] + ChiSingExtraA0pDefinition /. PhiActivate // NoScalar //
  ToNewCanonical;
ChiSingA0mDefinition = PhiA0m[] + ChiSingExtraA0mDefinition /. PhiActivate //
   NoScalar // ToNewCanonical;
ChiSingA1pDefinition = PhiA1p[-n, -m] + ChiSingExtraA1pDefinition /.
    PhiActivate // NoScalar // ToNewCanonical;
ChiSingA1mDefinition = PhiA1m[-n] + ChiSingExtraA1mDefinition /. PhiActivate //
   NoScalar // ToNewCanonical;
```

```
  ChiSingA2pDefinition = PhiA2p[-n, -m] + ChiSingExtraA2pDefinition /.
       PhiActivate // NoScalar // ToNewCanonical;
  ChiSingA2mDefinition = PhiA2m[-n, -m, -o] + ChiSingExtraA2mDefinition /.
       PhiActivate // NoScalar // ToNewCanonical;

  ChiSingB1pActivate =
   MakeRule[{ChiSingB1p[-n, -m], Evaluate[ChiSingB1pDefinition]},
    MetricOn → All, ContractMetrics → True];
  ChiSingB1mActivate = MakeRule[{ChiSingB1m[-n], Evaluate[ChiSingB1mDefinition]},
    MetricOn → All, ContractMetrics → True];
  ChiSingA0pActivate = MakeRule[{ChiSingA0p[], Scalar[
       Evaluate[ChiSingA0pDefinition]]}, MetricOn → All, ContractMetrics → True];
  ChiSingA0mActivate = MakeRule[{ChiSingA0m[], Scalar[
       Evaluate[ChiSingA0mDefinition]]}, MetricOn → All, ContractMetrics → True];
  ChiSingA1pActivate = MakeRule[{ChiSingA1p[-n, -m],
      Evaluate[ChiSingA1pDefinition]}, MetricOn → All, ContractMetrics → True];
  ChiSingA1mActivate = MakeRule[{ChiSingA1m[-n], Evaluate[ChiSingA1mDefinition]},
    MetricOn → All, ContractMetrics → True];
  ChiSingA2pActivate = MakeRule[{ChiSingA2p[-n, -m],
      Evaluate[ChiSingA2pDefinition]}, MetricOn → All, ContractMetrics → True];
  ChiSingA2mActivate = MakeRule[{ChiSingA2m[-n, -m, -o],
      Evaluate[ChiSingA2mDefinition]}, MetricOn → All, ContractMetrics → True];

  ChiSingActivate = Join[ChiSingB1pActivate, ChiSingB1mActivate,
    ChiSingA0pActivate, ChiSingA0mActivate, ChiSingA1pActivate,
    ChiSingA1mActivate, ChiSingA2pActivate, ChiSingA2mActivate];

  DumpSave[BinaryLocation[], {ChiSingActivate}];
  ClearBuild[];
 ];
```

build

In[ ]:=

```
OpenLastCache[];
```

build

Define ubJ^P, uAJ^P

build

In[•]:=

```
UBSymb = "u_b";
DefTensor[UB0p[], M4, PrintAs → SymbolBuild[UBSymb, Spin0p]];
DeclareOrder[UB0p[], 1];
DefTensor[UB1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[UBSymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[UB1p[-a, -b], 1];
DefTensor[UB1m[-a], M4,
  PrintAs → SymbolBuild[UBSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[UB1m[-a], 1];
DefTensor[UB2p[-a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[UBSymb, Spin2p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[UB2p[-a, -b], 1];
UASymb = "u_Я";
DefTensor[UA0p[], M4, PrintAs → SymbolBuild[UASymb, Spin0p]];
DeclareOrder[UA0p[], 1];
DefTensor[UA0m[], M4, PrintAs → SymbolBuild[UASymb, Spin0m]];
DeclareOrder[UA0m[], 1];
DefTensor[UA1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[UASymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[UA1p[-a, -b], 1];
DefTensor[UA1m[-a], M4,
  PrintAs → SymbolBuild[UASymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[UA1m[-a], 1];
DefTensor[UA2p[-a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[UASymb, Spin2p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[UA2p[-a, -b], 1];
DefTensor[UA2m[-a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[UASymb, Spin2m], OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[UA2m[-a, -b, -c], 1];
AutomaticRules[UA2m,
  MakeRule[{UA2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[UA2m, MakeRule[{epsilonG[a, b, c, d] UA2m[-a, -b, -c], 0},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[UB2p, MakeRule[{UB2p[a, -a], 0},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[UA2p, MakeRule[{UA2p[a, -a], 0},
    MetricOn → All, ContractMetrics → True]];
ClearBuild[];
```

build

Basic form $\hat{\pi}$ bJ^P, $\hat{\pi}$ AJ^P

build

In[•]:=

```
PiPBSymb = "π̂_b";
DefTensor[PiPB0p[], M4, PrintAs → SymbolBuild[PiPBSymb, Spin0p]];
DeclareOrder[PiPB0p[], 1];
DefTensor[PiPB1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPBSymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[PiPB1p[-a, -b], 1];
DefTensor[PiPB1m[-a], M4,
  PrintAs → SymbolBuild[PiPBSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[PiPB1m[-a], 1];
DefTensor[PiPB2p[-a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPBSymb, Spin2p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[PiPB2p[-a, -b], 1];
PiPASymb = "π̂_𝔄";
DefTensor[PiPA0p[], M4, PrintAs → SymbolBuild[PiPASymb, Spin0p]];
DeclareOrder[PiPA0p[], 1, "IsUnityWithEHTerm" → True];
DefTensor[PiPA0m[], M4, PrintAs → SymbolBuild[PiPASymb, Spin0m]];
DeclareOrder[PiPA0m[], 1];
DefTensor[PiPA1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPASymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[PiPA1p[-a, -b], 1];
DefTensor[PiPA1m[-a], M4,
  PrintAs → SymbolBuild[PiPASymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[PiPA1m[-a], 1];
DefTensor[PiPA2p[-a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPASymb, Spin2p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[PiPA2p[-a, -b], 1];
DefTensor[PiPA2m[-a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPASymb, Spin2m], OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[PiPA2m[-a, -b, -c], 1];
AutomaticRules[PiPA2m,
  MakeRule[{PiPA2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[PiPA2m, MakeRule[{epsilonG[a, b, c, d] PiPA2m[-a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[PiPB2p, MakeRule[{PiPB2p[a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[PiPA2p, MakeRule[{PiPA2p[a, -a], 0},
   MetricOn → All, ContractMetrics → True]];

PiPB0pDefinition = PB0p[e, f] PBPara[-e, -f, a, c] BPiP[-a, -c];
```

```
PiPB1pDefinition = PB1p[-n, -m, e, f] PBPara[-e, -f, a, c] BPiP[-a, -c];
PiPB2pDefinition = PB2p[-n, -m, e, f] PBPara[-e, -f, a, c] BPiP[-a, -c];
PiPB1mDefinition = PB1m[-n, f] PBPerp[-f, a, c] BPiP[-a, -c];
PiPA0pDefinition = PA0p[e, f] PAPerp[-e, -f, a, b, c] APiP[-a, -b, -c];
PiPA1pDefinition = PA1p[-n, -m, e, f] PAPerp[-e, -f, a, b, c] APiP[-a, -b, -c];
PiPA2pDefinition = PA2p[-n, -m, e, f] PAPerp[-e, -f, a, b, c] APiP[-a, -b, -c];
PiPA0mDefinition = PA0m[d, e, f] PAPara[-d, -e, -f, a, b, c] APiP[-a, -b, -c];
PiPA1mDefinition = PA1m[-n, d, e, f] PAPara[-d, -e, -f, a, b, c] APiP[-a, -b, -c];
PiPA2mDefinition =
  PA2m[-n, -m, -o, d, e, f] PAPara[-d, -e, -f, a, b, c] APiP[-a, -b, -c];

PiPB0pActivate = MakeRule[{PiPB0p[], Scalar[Evaluate[PiPB0pDefinition]]},
   MetricOn → All, ContractMetrics → True];
PiPB1pActivate = MakeRule[{PiPB1p[-n, -m], Evaluate[PiPB1pDefinition]},
   MetricOn → All, ContractMetrics → True];
PiPB1mActivate = MakeRule[{PiPB1m[-n], Evaluate[PiPB1mDefinition]},
   MetricOn → All, ContractMetrics → True];
PiPB2pActivate = MakeRule[{PiPB2p[-n, -m], Evaluate[PiPB2pDefinition]},
   MetricOn → All, ContractMetrics → True];
PiPA0pActivate = MakeRule[{PiPA0p[], Scalar[Evaluate[PiPA0pDefinition]]},
   MetricOn → All, ContractMetrics → True];
PiPA0mActivate = MakeRule[{PiPA0m[], Scalar[Evaluate[PiPA0mDefinition]]},
   MetricOn → All, ContractMetrics → True];
PiPA1pActivate = MakeRule[{PiPA1p[-n, -m], Evaluate[PiPA1pDefinition]},
   MetricOn → All, ContractMetrics → True];
PiPA1mActivate = MakeRule[{PiPA1m[-n], Evaluate[PiPA1mDefinition]},
   MetricOn → All, ContractMetrics → True];
PiPA2pActivate = MakeRule[{PiPA2p[-n, -m], Evaluate[PiPA2pDefinition]},
   MetricOn → All, ContractMetrics → True];
PiPA2mActivate = MakeRule[{PiPA2m[-n, -m, -o], Evaluate[PiPA2mDefinition]},
   MetricOn → All, ContractMetrics → True];

PiPO3Activate = Join[PiPB0pActivate, PiPB1pActivate,
   PiPB1mActivate, PiPB2pActivate, PiPA0pActivate, PiPA0mActivate,
   PiPA1pActivate, PiPA1mActivate, PiPA2pActivate, PiPA2mActivate];
ClearBuild[];
```

build

Basic form $\hat{T}\,J^P$, $\hat{R}\,J^P$

build

```
(*O(3) decomposition of the canonical parts of field strengths*)
TPSymb = "𝒯";
DefTensor[TP0m[], M4, PrintAs → SymbolBuild[TPSymb, Spin0m]];
```

```
DeclareOrder[TP0m[], 1];
DefTensor[TP1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TPSymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[TP1p[-a, -b], 1];
DefTensor[TP1m[-a], M4,
  PrintAs → SymbolBuild[TPSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[TP1m[-a], 1];
DefTensor[TP2m[-a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TPSymb, Spin2m], OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[TP2m[-a, -b, -c], 1];
RPSymb = "ℛ"";
DefTensor[RP0p[], M4, PrintAs → SymbolBuild[RPSymb, Spin0p]];
DeclareOrder[RP0p[], 1];
DefTensor[RP0m[], M4, PrintAs → SymbolBuild[RPSymb, Spin0m]];
DeclareOrder[RP0m[], 1];
DefTensor[RP1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPSymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RP1p[-a, -b], 1];
DefTensor[RP1m[-a], M4,
  PrintAs → SymbolBuild[RPSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[RP1m[-a], 1];
DefTensor[RP2p[-a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPSymb, Spin2p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RP2p[-a, -b], 1];
DefTensor[RP2m[-a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPSymb, Spin2m], OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[RP2m[-a, -b, -c], 1];
AutomaticRules[TP2m,
  MakeRule[{TP2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[TP2m, MakeRule[{epsilonG[a, b, c, d] TP2m[-a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[TP2m, MakeRule[{Eps[a, b, c] TP2m[-a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[RP2m, MakeRule[{RP2m[a, -b, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[RP2m, MakeRule[{epsilonG[a, b, c, d] RP2m[-a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[RP2p, MakeRule[{RP2p[a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
ClearBuild[];

(*Projections to break the field strengths up into canonical and non-
 canonical parts*)
DefTensor[PTPerp[-e, -f, a, b, c], M4];
```

```
DefTensor[PTPara[-e, -f, -g, a, b, c], M4];
DefTensor[PRPerp[-e, -f, -g, a, b, c, d], M4];
DefTensor[PRPara[-e, -f, -g, -h, a, b, c, d], M4];
PTPerpDefinition = V[a] PPara[-e, b] PPara[-f, c] /. PADMActivate // ToCanonical;
PTPerpActivate = MakeRule[{PTPerp[-e, -f, a, b, c], Evaluate[PTPerpDefinition]},
   MetricOn → All, ContractMetrics → True];
PTParaDefinition = PPara[-e, a] PPara[-f, b] PPara[-g, c] /. PADMActivate //
   ToCanonical;
PTParaActivate = MakeRule[{PTPara[-e, -f, -g, a, b, c],
    Evaluate[PTParaDefinition]}, MetricOn → All, ContractMetrics → True];
PRPerpDefinition = V[a] PPara[-e, b] PPara[-f, c] PPara[-g, d] /. PADMActivate //
   ToCanonical;
PRPerpActivate = MakeRule[{PRPerp[-e, -f, -g, a, b, c, d],
    Evaluate[PRPerpDefinition]}, MetricOn → All, ContractMetrics → True];
PRParaDefinition = PPara[-e, a] PPara[-f, b] PPara[-g, c] PPara[-h, d] /.
    PADMActivate // ToCanonical;
PRParaActivate = MakeRule[{PRPara[-e, -f, -g, -h, a, b, c, d],
    Evaluate[PRParaDefinition]}, MetricOn → All, ContractMetrics → True];
PADMTActivate = Join[PTPerpActivate, PTParaActivate];
PADMRActivate = Join[PRPerpActivate, PRParaActivate];
ClearBuild[];
```

build

Define $\hat{\lambda}\, J^P$

build

```
(*O(3) decomposition of the canonical parts of Riemann-Cartan multiplier*)
TLambdaPSymb = "λ_T^(-1)";
DefTensor[TLambdaP0m[], M4, PrintAs → SymbolBuild[TLambdaPSymb, Spin0m]];
DeclareOrder[TLambdaP0m[], 1];
DefTensor[TLambdaP1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPSymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[TLambdaP1p[-a, -b], 1];
DefTensor[TLambdaP1m[-a], M4,
  PrintAs → SymbolBuild[TLambdaPSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[TLambdaP1m[-a], 1];
DefTensor[TLambdaP2m[-a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPSymb, Spin2m], OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[TLambdaP2m[-a, -b, -c], 1];
RLambdaPSymb = "λ_R^(-1)";
DefTensor[RLambdaP0p[], M4, PrintAs → SymbolBuild[RLambdaPSymb, Spin0p]];
DeclareOrder[RLambdaP0p[], 1];
DefTensor[RLambdaP0m[], M4, PrintAs → SymbolBuild[RLambdaPSymb, Spin0m]];
DeclareOrder[RLambdaP0m[], 1];
DefTensor[RLambdaP1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPSymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RLambdaP1p[-a, -b], 1];
DefTensor[RLambdaP1m[-a], M4,
  PrintAs → SymbolBuild[RLambdaPSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[RLambdaP1m[-a], 1];
DefTensor[RLambdaP2p[-a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPSymb, Spin2p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RLambdaP2p[-a, -b], 1];
DefTensor[RLambdaP2m[-a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPSymb, Spin2m], OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[RLambdaP2m[-a, -b, -c], 1];
AutomaticRules[RLambdaP2m,
  MakeRule[{RLambdaP2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[RLambdaP2m, MakeRule[{epsilonG[a, b, c, d] RLambdaP2m[-a, -b, -c],
    0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[RLambdaP2p, MakeRule[{RLambdaP2p[a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[TLambdaP2m, MakeRule[{TLambdaP2m[a, -b, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[TLambdaP2m, MakeRule[{epsilonG[a, b, c, d] TLambdaP2m[-a, -b, -c],
    0}, MetricOn → All, ContractMetrics → True]];
ClearBuild[];
```

build

Define $\overset{*}{T}J^P$, $\overset{*}{R}J^P$

build

```
(*O(3) decomposition of the non-canonical parts of field strengths*)
TPerpSymb = "𝒯⊥";
DefTensor[TPerp0p[], M4, PrintAs → SymbolBuild[TPerpSymb, Spin0p]];
DeclareOrder[TPerp0p[], 1];
DefTensor[TPerp1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TPerpSymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[TPerp1p[-a, -b], 1];
DefTensor[TPerp1m[-a], M4,
  PrintAs → SymbolBuild[TPerpSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[TPerp1m[-a], 1];
DefTensor[TPerp2p[-a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[TPerpSymb, Spin2p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[TPerp2p[-a, -b], 1];
RPerpSymb = "ℛ⊥";
DefTensor[RPerp0p[], M4, PrintAs → SymbolBuild[RPerpSymb, Spin0p]];
DeclareOrder[RPerp0p[], 1];
DefTensor[RPerp0m[], M4, PrintAs → SymbolBuild[RPerpSymb, Spin0m]];
DeclareOrder[RPerp0m[], 1];
DefTensor[RPerp1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPerpSymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RPerp1p[-a, -b], 1];
DefTensor[RPerp1m[-a], M4,
  PrintAs → SymbolBuild[RPerpSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[RPerp1m[-a], 1];
DefTensor[RPerp2p[-a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPerpSymb, Spin2p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RPerp2p[-a, -b], 1];
DefTensor[RPerp2m[-a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPerpSymb, Spin2m], OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[RPerp2m[-a, -b, -c], 1];
AutomaticRules[TPerp2p,
  MakeRule[{TPerp2p[a, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[RPerp2m, MakeRule[{RPerp2m[a, -b, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[RPerp2m, MakeRule[{epsilonG[a, b, c, d] RPerp2m[-a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[RPerp2p, MakeRule[{RPerp2p[a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
ClearBuild[];
```

build

Define $\overset{*}{\lambda}\, J^P$

build

```
TLambdaPerpSymb = "λ⊥T";
DefTensor[TLambdaPerp0p[], M4, PrintAs → SymbolBuild[TLambdaPerpSymb, Spin0p]];
DeclareOrder[TLambdaPerp0p[], 1];
DefTensor[TLambdaPerp1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPerpSymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[TLambdaPerp1p[-a, -b], 1];
DefTensor[TLambdaPerp1m[-a], M4,
  PrintAs → SymbolBuild[TLambdaPerpSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[TLambdaPerp1m[-a], 1];
DefTensor[TLambdaPerp2p[-a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPerpSymb, Spin2p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[TLambdaPerp2p[-a, -b], 1];
RLambdaPerpSymb = "λ⊥R";
DefTensor[RLambdaPerp0p[], M4, PrintAs → SymbolBuild[RLambdaPerpSymb, Spin0p]];
DeclareOrder[RLambdaPerp0p[], 1];
DefTensor[RLambdaPerp0m[], M4, PrintAs → SymbolBuild[RLambdaPerpSymb, Spin0m]];
DeclareOrder[RLambdaPerp0m[], 1];
DefTensor[RLambdaPerp1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin1p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RLambdaPerp1p[-a, -b], 1];
DefTensor[RLambdaPerp1m[-a], M4,
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[RLambdaPerp1m[-a], 1];
DefTensor[RLambdaPerp2p[-a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin2p], OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RLambdaPerp2p[-a, -b], 1];
DefTensor[RLambdaPerp2m[-a, -b, -c], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[RLambdaPerpSymb, Spin2m],
  OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[RLambdaPerp2m[-a, -b, -c], 1];
AutomaticRules[RLambdaPerp2m, MakeRule[
   {RLambdaPerp2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[RLambdaPerp2m, MakeRule[
   {epsilonG[a, b, c, d] RLambdaPerp2m[-a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[RLambdaPerp2p, MakeRule[{RLambdaPerp2p[a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[TLambdaPerp2p, MakeRule[{TLambdaPerp2p[a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
ClearBuild[];
```

build

Nester form $\hat{T}, \hat{R}$

build

```
(*These rules then expand the O(3) parts in terms of the canonical parts*)
TP0mDefinition =
  PT0m[e, f, g] PTPara[-e, -f, -g, a, b, c] TP[-a, -b, -c] /. PO3TActivate /.
    PADMTActivate // ToCanonical;
TP1pDefinition = PT1p[-n, -m, e, f] PTPerp[-e, -f, a, b, c] TP[-a, -b, -c] /.
     PO3TActivate /. PADMTActivate // ToCanonical;
TP1mDefinition = PT1m[-n, e, f, g] PTPara[-e, -f, -g, a, b, c] TP[-a, -b, -c] /.
     PO3TActivate /. PADMTActivate // ToCanonical;
TP2mDefinition = PT2m[-n, -m, -o, e, f, g] PTPara[-e, -f, -g, a, b, c]
      TP[-a, -b, -c] /. PO3TActivate /. PADMTActivate // ToCanonical;

RP0pDefinition =
  PR0p[e, f, g, h] PRPara[-e, -f, -g, -h, a, b, c, d] RP[-a, -b, -c, -d] /.
     PO3RActivate /. PADMRActivate // ToCanonical;
RP0mDefinition = PR0m[e, f, g] PRPerp[-e, -f, -g, a, b, c, d] RP[-a, -b, -c, -d] /.
     PO3RActivate /. PADMRActivate // ToCanonical;
RP1pDefinition = PR1p[-n, -m, e, f, g, h] PRPara[-e, -f, -g, -h, a, b, c, d]
      RP[-a, -b, -c, -d] /. PO3RActivate /. PADMRActivate // ToCanonical;
RP1mDefinition = PR1m[-n, e, f, g] PRPerp[-e, -f, -g, a, b, c, d] RP[-a, -b, -c, -d] /.
     PO3RActivate /. PADMRActivate // ToCanonical;
RP2pDefinition = PR2p[-n, -m, e, f, g, h] PRPara[-e, -f, -g, -h, a, b, c, d]
      RP[-a, -b, -c, -d] /. PO3RActivate /. PADMRActivate // ToCanonical;
RP2mDefinition = PR2m[-n, -m, -o, e, f, g] PRPerp[-e, -f, -g, a, b, c, d]
      RP[-a, -b, -c, -d] /. PO3RActivate /. PADMRActivate // ToCanonical;

TP0mActivate = MakeRule[{TP0m[], Scalar[Evaluate[TP0mDefinition]]},
   MetricOn → All, ContractMetrics → True];
TP1pActivate = MakeRule[{TP1p[-n, -m], Evaluate[TP1pDefinition]},
   MetricOn → All, ContractMetrics → True];
TP1mActivate = MakeRule[{TP1m[-n], Evaluate[TP1mDefinition]},
   MetricOn → All, ContractMetrics → True];
TP2mActivate = MakeRule[{TP2m[-n, -m, -o], Evaluate[TP2mDefinition]},
   MetricOn → All, ContractMetrics → True];

RP0pActivate = MakeRule[{RP0p[], Scalar[Evaluate[RP0pDefinition]]},
   MetricOn → All, ContractMetrics → True];
RP0mActivate = MakeRule[{RP0m[], Scalar[Evaluate[RP0mDefinition]]},
   MetricOn → All, ContractMetrics → True];
RP1pActivate = MakeRule[{RP1p[-n, -m], Evaluate[RP1pDefinition]},
```

```
    MetricOn → All, ContractMetrics → True];
RP1mActivate = MakeRule[{RP1m[-n], Evaluate[RP1mDefinition]},
    MetricOn → All, ContractMetrics → True];
RP2pActivate = MakeRule[{RP2p[-n, -m], Evaluate[RP2pDefinition]},
    MetricOn → All, ContractMetrics → True];
RP2mActivate = MakeRule[{RP2m[-n, -m, -o], Evaluate[RP2mDefinition]},
    MetricOn → All, ContractMetrics → True];

TPO3Activate = Join[TP0mActivate, TP1pActivate, TP1mActivate, TP2mActivate];
RPO3Activate = Join[RP0pActivate, RP0mActivate,
    RP1pActivate, RP1mActivate, RP2pActivate, RP2mActivate];

TPDefinition =  V[-a] TP1p[-b, -c] +
       - (1 / 6) PT0m[-a, -b, -c] TP0m[] +
        Antisymmetrize[-PPara[-a, -b] TP1m[-c], {-b, -c}] +
       (4 / 3) TP2m[-b, -c, -a] /. PO3TActivate /. PADMActivate // ToCanonical;

DefTensor[RPPara[-a, -b, -c, -d], M4,
  {Antisymmetric[{-a, -b}], Antisymmetric[{-c, -d}]},
  OrthogonalTo → {V[a], V[b], V[c], V[d]}];
DeclareOrder[RPPara[-a, -b, -c, -d], 1];
DefTensor[RPPerp[-a, -b, -c], M4,
  Antisymmetric[{-b, -c}], OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[RPPerp[-a, -b, -c], 1];

RPParaDefinition =
  - (1 / 6) (PPara[-a, -d] PPara[-b, -c] - PPara[-a, -c] PPara[-b, -d]) RP0p[] -
    (PPara[-b, -d] RP1p[-a, -c] - PPara[-b, -c] RP1p[-a, -d] -
      PPara[-a, -d] RP1p[-b, -c] + PPara[-a, -c] RP1p[-b, -d]) +
    (PPara[-b, -d] RP2p[-a, -c] - PPara[-b, -c] RP2p[-a, -d] -
      PPara[-a, -d] RP2p[-b, -c] + PPara[-a, -c] RP2p[-b, -d]);
RPPerpDefinition = - (1 / 6) PR0m[-a, -b, -c] RP0m[] +
    Antisymmetrize[-PPara[-a, -b] RP1m[-c], {-b, -c}] + (4 / 3) RP2m[-b, -c, -a];

RPParaActivate = MakeRule[{RPPara[-a, -b, -c, -d], Evaluate[RPParaDefinition]},
    MetricOn → All, ContractMetrics → True];
RPPerpActivate = MakeRule[{RPPerp[-a, -b, -c], Evaluate[RPPerpDefinition]},
    MetricOn → All, ContractMetrics → True];
RPParaPerpActivate = Join[RPParaActivate, RPPerpActivate];

RPDefinition =
  RPPara[-a, -b, -c, -d] + 2 Antisymmetrize[V[-a] RPPerp[-b, -c, -d], {-a, -b}] /.
      RPParaPerpActivate /. PO3RActivate /. PADMActivate // ToCanonical;
```

```
TPDefinition =
  TPDefinition // CollectTensors // ScreenDollarIndices // CollectTensors;
(*TPDefinition=TPDefinition/.TPO3Activate//CollectTensors//
    ScreenDollarIndices//CollectTensors;*)(*removed 19/04*)
RPDefinition = RPDefinition // CollectTensors // ScreenDollarIndices //
   CollectTensors;

TPActivate = MakeRule[{TP[-a, -b, -c], Evaluate[TPDefinition]},
   MetricOn → All, ContractMetrics → True];
RPActivate = MakeRule[{RP[-a, -b, -c, -d], Evaluate[RPDefinition]},
   MetricOn → All, ContractMetrics → True];
StrengthPToStrengthPO3 = Join[TPActivate, RPActivate];
ClearBuild[];
```

build

Nester form $\hat{\lambda}$

build

In[•]:=

```
(*These rules then expand the O(3) parts in terms of the canonical parts*)

TLambdaP0mDefinition =
  PT0m[e, f, g] PTPara[-e, -f, -g, a, b, c] TLambdaP[-a, -b, -c] /. PO3TActivate /.
    PADMTActivate // ToCanonical;
TLambdaP1pDefinition = PT1p[-n, -m, e, f] PTPerp[-e, -f, a, b, c]
      TLambdaP[-a, -b, -c] /. PO3TActivate /. PADMTActivate // ToCanonical;
TLambdaP1mDefinition = PT1m[-n, e, f, g] PTPara[-e, -f, -g, a, b, c]
      TLambdaP[-a, -b, -c] /. PO3TActivate /. PADMTActivate // ToCanonical;
TLambdaP2mDefinition = PT2m[-n, -m, -o, e, f, g] PTPara[-e, -f, -g, a, b, c]
      TLambdaP[-a, -b, -c] /. PO3TActivate /. PADMTActivate // ToCanonical;

RLambdaP0pDefinition =
  PR0p[e, f, g, h] PRPara[-e, -f, -g, -h, a, b, c, d] RLambdaP[-a, -b, -c, -d] /.
     PO3RActivate /. PADMRActivate // ToCanonical;
RLambdaP0mDefinition = PR0m[e, f, g] PRPerp[-e, -f, -g, a, b, c, d]
      RLambdaP[-a, -b, -c, -d] /. PO3RActivate /. PADMRActivate // ToCanonical;
RLambdaP1pDefinition = PR1p[-n, -m, e, f, g, h] PRPara[-e, -f, -g, -h, a, b, c, d]
      RLambdaP[-a, -b, -c, -d] /. PO3RActivate /. PADMRActivate // ToCanonical;
RLambdaP1mDefinition = PR1m[-n, e, f, g] PRPerp[-e, -f, -g, a, b, c, d]
      RLambdaP[-a, -b, -c, -d] /. PO3RActivate /. PADMRActivate // ToCanonical;
RLambdaP2pDefinition = PR2p[-n, -m, e, f, g, h] PRPara[-e, -f, -g, -h, a, b, c, d]
      RLambdaP[-a, -b, -c, -d] /. PO3RActivate /. PADMRActivate // ToCanonical;
RLambdaP2mDefinition = PR2m[-n, -m, -o, e, f, g] PRPerp[-e, -f, -g, a, b, c, d]
      RLambdaP[-a, -b, -c, -d] /. PO3RActivate /. PADMRActivate // ToCanonical;
```

```
TLambdaP0mActivate =
  MakeRule[{TLambdaP0m[], Scalar[Evaluate[TLambdaP0mDefinition]]},
   MetricOn → All, ContractMetrics → True];
TLambdaP1pActivate = MakeRule[{TLambdaP1p[-n, -m],
    Evaluate[TLambdaP1pDefinition]}, MetricOn → All, ContractMetrics → True];
TLambdaP1mActivate = MakeRule[{TLambdaP1m[-n], Evaluate[TLambdaP1mDefinition]},
   MetricOn → All, ContractMetrics → True];
TLambdaP2mActivate = MakeRule[{TLambdaP2m[-n, -m, -o],
    Evaluate[TLambdaP2mDefinition]}, MetricOn → All, ContractMetrics → True];

RLambdaP0pActivate =
  MakeRule[{RLambdaP0p[], Scalar[Evaluate[RLambdaP0pDefinition]]},
   MetricOn → All, ContractMetrics → True];
RLambdaP0mActivate = MakeRule[{RLambdaP0m[], Scalar[
     Evaluate[RLambdaP0mDefinition]]}, MetricOn → All, ContractMetrics → True];
RLambdaP1pActivate = MakeRule[{RLambdaP1p[-n, -m],
    Evaluate[RLambdaP1pDefinition]}, MetricOn → All, ContractMetrics → True];
RLambdaP1mActivate = MakeRule[{RLambdaP1m[-n], Evaluate[RLambdaP1mDefinition]},
   MetricOn → All, ContractMetrics → True];
RLambdaP2pActivate = MakeRule[{RLambdaP2p[-n, -m],
    Evaluate[RLambdaP2pDefinition]}, MetricOn → All, ContractMetrics → True];
RLambdaP2mActivate = MakeRule[{RLambdaP2m[-n, -m, -o],
    Evaluate[RLambdaP2mDefinition]}, MetricOn → All, ContractMetrics → True];

TLambdaPO3Activate = Join[TLambdaP0mActivate,
   TLambdaP1pActivate, TLambdaP1mActivate, TLambdaP2mActivate];
RLambdaPO3Activate = Join[RLambdaP0pActivate, RLambdaP0mActivate,
   RLambdaP1pActivate, RLambdaP1mActivate,
   RLambdaP2pActivate, RLambdaP2mActivate];

TLambdaPDefinition = V[-a] TLambdaP1p[-b, -c] +
      -(1/6) PT0m[-a, -b, -c] TLambdaP0m[] +
       Antisymmetrize[-PPara[-a, -b] TLambdaP1m[-c], {-b, -c}] +
       (4/3) TLambdaP2m[-b, -c, -a] /. PO3TActivate /. PADMActivate // ToCanonical;

DefTensor[RLambdaPPara[-a, -b, -c, -d],
  M4, {Antisymmetric[{-a, -b}], Antisymmetric[{-c, -d}]},
  OrthogonalTo → {V[a], V[b], V[c], V[d]}];
DeclareOrder[RLambdaPPara[-a, -b, -c, -d], 1];
DefTensor[RLambdaPPerp[-a, -b, -c], M4,
  Antisymmetric[{-b, -c}], OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[RLambdaPPerp[-a, -b, -c], 1];
```

```
RLambdaPParaDefinition = -(1/6)
     (PPara[-a, -d] PPara[-b, -c] - PPara[-a, -c] PPara[-b, -d]) RLambdaP0p[] -
    (PPara[-b, -d] RLambdaP1p[-a, -c] - PPara[-b, -c] RLambdaP1p[-a, -d] -
      PPara[-a, -d] RLambdaP1p[-b, -c] + PPara[-a, -c] RLambdaP1p[-b, -d]) +
    (PPara[-b, -d] RLambdaP2p[-a, -c] - PPara[-b, -c] RLambdaP2p[-a, -d] -
      PPara[-a, -d] RLambdaP2p[-b, -c] + PPara[-a, -c] RLambdaP2p[-b, -d]);
RLambdaPPerpDefinition = -(1/6) PR0m[-a, -b, -c] RLambdaP0m[] + Antisymmetrize[
     -PPara[-a, -b] RLambdaP1m[-c], {-b, -c}] + (4/3) RLambdaP2m[-b, -c, -a];

RLambdaPParaActivate =
  MakeRule[{RLambdaPPara[-a, -b, -c, -d], Evaluate[RLambdaPParaDefinition]},
   MetricOn → All, ContractMetrics → True];
RLambdaPPerpActivate = MakeRule[{RLambdaPPerp[-a, -b, -c],
    Evaluate[RLambdaPPerpDefinition]}, MetricOn → All, ContractMetrics → True];
RLambdaPParaPerpActivate = Join[RLambdaPParaActivate, RLambdaPPerpActivate];

RLambdaPDefinition = RLambdaPPara[-a, -b, -c, -d] +
       2 Antisymmetrize[V[-a] RLambdaPPerp[-b, -c, -d], {-a, -b}] /.
      RLambdaPParaPerpActivate /. PO3RActivate /. PADMActivate // ToCanonical;

TLambdaPDefinition =
  TLambdaPDefinition // CollectTensors // ScreenDollarIndices // CollectTensors;
RLambdaPDefinition = RLambdaPDefinition // CollectTensors //
    ScreenDollarIndices // CollectTensors;

TLambdaPActivate = MakeRule[{TLambdaP[-a, -b, -c], Evaluate[TLambdaPDefinition]},
   MetricOn → All, ContractMetrics → True];
RLambdaPActivate = MakeRule[{RLambdaP[-a, -b, -c, -d],
    Evaluate[RLambdaPDefinition]}, MetricOn → All, ContractMetrics → True];
StrengthLambdaPToStrengthLambdaPO3 = Join[TLambdaPActivate, RLambdaPActivate];
ClearBuild[];
```

build

Nester form $\overset{*}{T}, \overset{*}{R}$

build

```
(*These rules then expand the O(3) parts in terms of the canonical parts*)
TPerp0pDefinition =
  PPerpT0p[e, f] PPerpTPara[-e, -f, a, b] TPerp[-a, -b] /. PPerpO3TActivate /.
    PPerpADMTActivate // ToCanonical;
TPerp1pDefinition = PPerpT1p[-n, -m, e, f] PPerpTPara[-e, -f, a, b] TPerp[-a, -b] /.
     PPerpO3TActivate /. PPerpADMTActivate // ToCanonical;
```

```
TPerp1mDefinition = PPerpT1m[-n, e] PPerpTPerp[-e, a, b] TPerp[-a, -b] /.
      PPerpO3TActivate /. PPerpADMTActivate // ToCanonical;
TPerp2pDefinition = PPerpT2p[-n, -m, e, f] PPerpTPara[-e, -f, a, b] TPerp[-a, -b] /.
      PPerpO3TActivate /. PPerpADMTActivate // ToCanonical;

RPerp0pDefinition =
  PPerpR0p[e, f] PPerpRPerp[-e, -f, a, b, c] RPerp[-a, -b, -c] /. PPerpO3RActivate /.
    PPerpADMRActivate // ToCanonical;
RPerp0mDefinition = PPerpR0m[e, f, g] PPerpRPara[-e, -f, -g, a, b, c]
       RPerp[-a, -b, -c] /. PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;
RPerp1pDefinition = PPerpR1p[-n, -m, e, f] PPerpRPerp[-e, -f, a, b, c]
       RPerp[-a, -b, -c] /. PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;
RPerp1mDefinition = PPerpR1m[-n, e, f, g] PPerpRPara[-e, -f, -g, a, b, c]
       RPerp[-a, -b, -c] /. PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;
RPerp2pDefinition = PPerpR2p[-n, -m, e, f] PPerpRPerp[-e, -f, a, b, c]
       RPerp[-a, -b, -c] /. PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;
RPerp2mDefinition = PPerpR2m[-n, -m, -o, e, f, g] PPerpRPara[-e, -f, -g, a, b, c]
       RPerp[-a, -b, -c] /. PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;

TPerp0pActivate = MakeRule[{TPerp0p[], Scalar[Evaluate[TPerp0pDefinition]]},
    MetricOn → All, ContractMetrics → True];
TPerp1pActivate = MakeRule[{TPerp1p[-n, -m], Evaluate[TPerp1pDefinition]},
    MetricOn → All, ContractMetrics → True];
TPerp1mActivate = MakeRule[{TPerp1m[-n], Evaluate[TPerp1mDefinition]},
    MetricOn → All, ContractMetrics → True];
TPerp2pActivate = MakeRule[{TPerp2p[-n, -m], Evaluate[TPerp2pDefinition]},
    MetricOn → All, ContractMetrics → True];

RPerp0pActivate = MakeRule[{RPerp0p[], Scalar[Evaluate[RPerp0pDefinition]]},
    MetricOn → All, ContractMetrics → True];
RPerp0mActivate = MakeRule[{RPerp0m[], Scalar[Evaluate[RPerp0mDefinition]]},
    MetricOn → All, ContractMetrics → True];
RPerp1pActivate = MakeRule[{RPerp1p[-n, -m], Evaluate[RPerp1pDefinition]},
    MetricOn → All, ContractMetrics → True];
RPerp1mActivate = MakeRule[{RPerp1m[-n], Evaluate[RPerp1mDefinition]},
    MetricOn → All, ContractMetrics → True];
RPerp2pActivate = MakeRule[{RPerp2p[-n, -m], Evaluate[RPerp2pDefinition]},
    MetricOn → All, ContractMetrics → True];
RPerp2mActivate = MakeRule[{RPerp2m[-n, -m, -o], Evaluate[RPerp2mDefinition]},
    MetricOn → All, ContractMetrics → True];

TPerpO3Activate =
  Join[TPerp0pActivate, TPerp1pActivate, TPerp1mActivate, TPerp2pActivate];
RPerpO3Activate = Join[RPerp0pActivate, RPerp0mActivate,
```

```
    RPerp1pActivate, RPerp1mActivate, RPerp2pActivate, RPerp2mActivate];

TPerpDefinition = V[-a] TPerp1m[-b] +
       TPerp1p[-a, -b] +
       TPerp2p[-a, -b] +
       (1/3) PPara[-a, -b] TPerp0p[] /. PPerpO3TActivate /. PADMActivate //
    ToCanonical;

DefTensor[RPerpPerp[-a, -b], M4, OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RPerpPerp[-a, -b], 1];
DefTensor[RPerpPara[-a, -b, -c], M4,
  Antisymmetric[{-a, -b}], OrthogonalTo → {V[a], V[b], V[c]}];
DeclareOrder[RPerpPara[-a, -b, -c], 1];

RPerpPerpDefinition = RPerp1p[-a, -b] +
       RPerp2p[-a, -b] -
       (1/3) PPara[-a, -b] RPerp0p[] /. PPerpO3RActivate /. PADMActivate //
    ToCanonical;
RPerpParaDefinition = -(1/6) PR0m[-a, -b, -c] RPerp0m[] - Antisymmetrize[
        -PPara[-c, -a] RPerp1m[-b], {-a, -b}] + (4/3) RPerp2m[-a, -b, -c] /.
      PPerpO3RActivate /. PADMActivate // ToCanonical;

RPerpPerpActivate = MakeRule[{RPerpPerp[-a, -b], Evaluate[RPerpPerpDefinition]},
    MetricOn → All, ContractMetrics → True];
RPerpParaActivate = MakeRule[{RPerpPara[-a, -b, -c],
     Evaluate[RPerpParaDefinition]}, MetricOn → All, ContractMetrics → True];
RPerpParaPerpActivate = Join[RPerpParaActivate, RPerpPerpActivate];

RPerpDefinition =
  RPerpPara[-a, -b, -c] + 2 Antisymmetrize[V[-a] RPerpPerp[-b, -c], {-a, -b}] /.
       RPerpParaPerpActivate /. PO3RActivate /. PADMActivate // ToCanonical;

TPerpDefinition =
  TPerpDefinition // CollectTensors // ScreenDollarIndices // CollectTensors;
(*
TPerpDefinition=TPerpDefinition/.TPerpO3Activate//NoScalar//ToNewCanonical;
HiGGSPrint[TPerpDefinition];
*)
RPerpDefinition =
  RPerpDefinition // CollectTensors // ScreenDollarIndices // CollectTensors;
(*
RPerpDefinition=RPerpDefinition/.RPerpO3Activate//NoScalar;
RPerpDefinition=RPerpDefinition//ToNewCanonical;
```

```
RPerpDefinition=RPerpDefinition//ToCanonical;
HiGGSPrint[RPerpDefinition];
*)

TPerpActivate = MakeRule[{TPerp[-a, -b], Evaluate[TPerpDefinition]},
    MetricOn → All, ContractMetrics → True];
RPerpActivate = MakeRule[{RPerp[-a, -b, -c], Evaluate[RPerpDefinition]},
    MetricOn → All, ContractMetrics → True];
StrengthPerpToStrengthPerpO3 = Join[TPerpActivate, RPerpActivate];
ClearBuild[];
```

build

Nester form $\overset{*}{\lambda}$

build

```
(*These rules then expand the O(3) parts in terms of the canonical parts*)
TLambdaPerp0pDefinition =
  PPerpT0p[e, f] PPerpTPara[-e, -f, a, b] TLambdaPerp[-a, -b] /. PPerpO3TActivate /.
    PPerpADMTActivate // ToCanonical;
TLambdaPerp1pDefinition =
  PPerpT1p[-n, -m, e, f] PPerpTPara[-e, -f, a, b] TLambdaPerp[-a, -b] /.
     PPerpO3TActivate /. PPerpADMTActivate // ToCanonical;
TLambdaPerp1mDefinition = PPerpT1m[-n, e] PPerpTPerp[-e, a, b] TLambdaPerp[
       -a, -b] /. PPerpO3TActivate /. PPerpADMTActivate // ToCanonical;
TLambdaPerp2pDefinition = PPerpT2p[-n, -m, e, f]
      PPerpTPara[-e, -f, a, b] TLambdaPerp[-a, -b] /.
     PPerpO3TActivate /. PPerpADMTActivate // ToCanonical;

RLambdaPerp0pDefinition =
  PPerpR0p[e, f] PPerpRPerp[-e, -f, a, b, c] RLambdaPerp[-a, -b, -c] /.
     PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;
RLambdaPerp0mDefinition = PPerpR0m[e, f, g] PPerpRPara[-e, -f, -g, a, b, c]
      RLambdaPerp[-a, -b, -c] /.
     PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;
RLambdaPerp1pDefinition = PPerpR1p[-n, -m, e, f]
      PPerpRPerp[-e, -f, a, b, c] RLambdaPerp[-a, -b, -c] /.
     PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;
RLambdaPerp1mDefinition = PPerpR1m[-n, e, f, g]
      PPerpRPara[-e, -f, -g, a, b, c] RLambdaPerp[-a, -b, -c] /.
     PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;
RLambdaPerp2pDefinition = PPerpR2p[-n, -m, e, f]
      PPerpRPerp[-e, -f, a, b, c] RLambdaPerp[-a, -b, -c] /.
     PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;
RLambdaPerp2mDefinition = PPerpR2m[-n, -m, -o, e, f, g]
```

```
      PPerpRPara[-e, -f, -g, a, b, c] RLambdaPerp[-a, -b, -c] /.
     PPerpO3RActivate /. PPerpADMRActivate // ToCanonical;

TLambdaPerp0pActivate =
  MakeRule[{TLambdaPerp0p[], Scalar[Evaluate[TLambdaPerp0pDefinition]]},
   MetricOn → All, ContractMetrics → True];
TLambdaPerp1pActivate = MakeRule[{TLambdaPerp1p[-n, -m],
    Evaluate[TLambdaPerp1pDefinition]}, MetricOn → All, ContractMetrics → True];
TLambdaPerp1mActivate = MakeRule[{TLambdaPerp1m[-n],
    Evaluate[TLambdaPerp1mDefinition]}, MetricOn → All, ContractMetrics → True];
TLambdaPerp2pActivate = MakeRule[{TLambdaPerp2p[-n, -m],
    Evaluate[TLambdaPerp2pDefinition]}, MetricOn → All, ContractMetrics → True];

RLambdaPerp0pActivate =
  MakeRule[{RLambdaPerp0p[], Scalar[Evaluate[RLambdaPerp0pDefinition]]},
   MetricOn → All, ContractMetrics → True];
RLambdaPerp0mActivate = MakeRule[{RLambdaPerp0m[],
    Scalar[Evaluate[RLambdaPerp0mDefinition]]},
   MetricOn → All, ContractMetrics → True];
RLambdaPerp1pActivate = MakeRule[{RLambdaPerp1p[-n, -m],
    Evaluate[RLambdaPerp1pDefinition]}, MetricOn → All, ContractMetrics → True];
RLambdaPerp1mActivate = MakeRule[{RLambdaPerp1m[-n],
    Evaluate[RLambdaPerp1mDefinition]}, MetricOn → All, ContractMetrics → True];
RLambdaPerp2pActivate = MakeRule[{RLambdaPerp2p[-n, -m],
    Evaluate[RLambdaPerp2pDefinition]}, MetricOn → All, ContractMetrics → True];
RLambdaPerp2mActivate = MakeRule[{RLambdaPerp2m[-n, -m, -o],
    Evaluate[RLambdaPerp2mDefinition]}, MetricOn → All, ContractMetrics → True];

TLambdaPerpO3Activate = Join[TLambdaPerp0pActivate,
   TLambdaPerp1pActivate, TLambdaPerp1mActivate, TLambdaPerp2pActivate];
RLambdaPerpO3Activate = Join[RLambdaPerp0pActivate, RLambdaPerp0mActivate,
   RLambdaPerp1pActivate, RLambdaPerp1mActivate,
   RLambdaPerp2pActivate, RLambdaPerp2mActivate];

TLambdaPerpDefinition = V[-a] TLambdaPerp1m[-b] +
      TLambdaPerp1p[-a, -b] +
      TLambdaPerp2p[-a, -b] +
      (1 / 3) PPara[-a, -b] TLambdaPerp0p[] /. PPerpO3TActivate /. PADMActivate //
   ToCanonical;

DefTensor[RLambdaPerpPerp[-a, -b], M4, OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RLambdaPerpPerp[-a, -b], 1];
DefTensor[RLambdaPerpPara[-a, -b, -c], M4,
  Antisymmetric[{-a, -b}], OrthogonalTo → {V[a], V[b], V[c]}];
```

```
DeclareOrder[RLambdaPerpPara[-a, -b, -c], 1];

RLambdaPerpPerpDefinition = RLambdaPerp1p[-a, -b] +
      RLambdaPerp2p[-a, -b] -
      (1/3) PPara[-a, -b] RLambdaPerp0p[] /. PPerpO3RActivate /. PADMActivate //
   ToCanonical;
RLambdaPerpParaDefinition = -(1/6) PR0m[-a, -b, -c] RLambdaPerp0m[] -
      Antisymmetrize[-PPara[-c, -a] RLambdaPerp1m[-b], {-a, -b}] +
      (4/3) RLambdaPerp2m[-a, -b, -c] /.
     PPerpO3RActivate /. PADMActivate // ToCanonical;

RLambdaPerpPerpActivate =
  MakeRule[{RLambdaPerpPerp[-a, -b], Evaluate[RLambdaPerpPerpDefinition]},
   MetricOn → All, ContractMetrics → True];
RLambdaPerpParaActivate = MakeRule[
   {RLambdaPerpPara[-a, -b, -c], Evaluate[RLambdaPerpParaDefinition]},
   MetricOn → All, ContractMetrics → True];
RLambdaPerpParaPerpActivate = Join[RLambdaPerpParaActivate,
   RLambdaPerpPerpActivate];

RLambdaPerpDefinition =  RLambdaPerpPara[-a, -b, -c] +
       2 Antisymmetrize[V[-a] RLambdaPerpPerp[-b, -c], {-a, -b}] /.
      RLambdaPerpParaPerpActivate /. PO3RActivate /. PADMActivate // ToCanonical;

TLambdaPerpDefinition =
  TLambdaPerpDefinition // CollectTensors // ScreenDollarIndices //
   CollectTensors;

RLambdaPerpDefinition =
  RLambdaPerpDefinition // CollectTensors // ScreenDollarIndices //
   CollectTensors;
(*
RPerpDefinition=RPerpDefinition/.RPerpO3Activate//NoScalar;
RPerpDefinition=RPerpDefinition//ToNewCanonical;
RPerpDefinition=RPerpDefinition//ToCanonical;
HiGGSPrint[RPerpDefinition];
*)

TLambdaPerpActivate =
  MakeRule[{TLambdaPerp[-a, -b], Evaluate[TLambdaPerpDefinition]},
   MetricOn → All, ContractMetrics → True];
RLambdaPerpActivate = MakeRule[{RLambdaPerp[-a, -b, -c],
    Evaluate[RLambdaPerpDefinition]}, MetricOn → All, ContractMetrics → True];
```

```
StrengthLambdaPerpToStrengthLambdaPerpO3 =
  Join[TLambdaPerpActivate, RLambdaPerpActivate];
(*Again used to be Join...*)
ClearBuild[];
```

build

Nester form $\hat{\pi}\, b$, $\hat{\pi}\, A$

build

In[•]:=

```
BPiPDefinition = ((1/3) PPara[-n, -m] PiPB0p[] +
         PiPB1p[-n, -m] +
         PiPB2p[-n, -m] +
        V[-n] PiPB1m[-m]) /. PO3PiActivate /. PADMActivate // ToNewCanonical;

APiPDefinition = (Antisymmetrize[
        2 Antisymmetrize[V[-n] (1/3) PPara[-m, -o] PiPA0p[], {-n, -m}] +
          2 Antisymmetrize[V[-n] PiPA1p[-m, -o], {-n, -m}] +
          2 Antisymmetrize[V[-n] PiPA2p[-m, -o], {-n, -m}] +
          (-1/6) PA0m[-n, -m, -o] PiPA0m[] +
          Antisymmetrize[-PPara[-m, -o] PiPA1m[-n], {-m, -n}] +
          (4/3) PiPA2m[-n, -m, -o], {-n, -m}]) /. PO3PiActivate /. PADMActivate //
    ToNewCanonical;

BPiPActivate = MakeRule[{BPiP[-n, -m], Evaluate[BPiPDefinition]},
    MetricOn → All, ContractMetrics → True];
APiPActivate = MakeRule[{APiP[-n, -m, -o], Evaluate[APiPDefinition]},
    MetricOn → All, ContractMetrics → True];
PiPToPiPO3 = Join[BPiPActivate, APiPActivate];
ClearBuild[];
```

build

Basic form $\chi^{\parallel}$ bJ^P, $\chi^{\parallel}$ AJ^P

build

In[•]:=

```
ChiParaBSymb = "χ_b^∥";
DefTensor[ChiParaB0m[], M4, PrintAs → SymbolBuild[ChiParaBSymb, Spin0m]];
DeclareOrder[ChiParaB0m[], 1];
DefTensor[ChiParaB1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[ChiParaBSymb, Spin1p]];
DeclareOrder[ChiParaB1p[-a, -b], 1];
DefTensor[ChiParaB1m[-a], M4, PrintAs → SymbolBuild[ChiParaBSymb, Spin1m]];
DeclareOrder[ChiParaB1m[-a], 1];
DefTensor[ChiParaB2m[-a, -b, -c], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[ChiParaBSymb, Spin2m]];
```

```
DeclareOrder[ChiParaB2m[-a, -b, -c], 1];
ChiParaASymb = "χ𝔄";
DefTensor[ChiParaA0p[], M4, PrintAs → SymbolBuild[ChiParaASymb, Spin0p]];
DeclareOrder[ChiParaA0p[], 1];
DefTensor[ChiParaA0m[], M4, PrintAs → SymbolBuild[ChiParaASymb, Spin0m]];
DeclareOrder[ChiParaA0m[], 1];
DefTensor[ChiParaA1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[ChiParaASymb, Spin1p]];
DeclareOrder[ChiParaA1p[-a, -b], 1];
DefTensor[ChiParaA1m[-a], M4, PrintAs → SymbolBuild[ChiParaASymb, Spin1m]];
DeclareOrder[ChiParaA1m[-a], 1];
DefTensor[ChiParaA2p[-a, -b], M4,
  Symmetric[{-a, -b}], PrintAs → SymbolBuild[ChiParaASymb, Spin2p]];
DeclareOrder[ChiParaA2p[-a, -b], 1];
DefTensor[ChiParaA2m[-a, -b, -c], M4,
  Antisymmetric[{-a, -b}], PrintAs → SymbolBuild[ChiParaASymb, Spin2m]];
DeclareOrder[ChiParaA2m[-a, -b, -c], 1];
AutomaticRules[ChiParaB2m,
  MakeRule[{ChiParaB2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[ChiParaB2m, MakeRule[{epsilonG[a, b, c, d] ChiParaB2m[-a, -b, -c],
    0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[ChiParaA2m, MakeRule[{ChiParaA2m[a, -b, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[ChiParaA2m, MakeRule[{epsilonG[a, b, c, d] ChiParaA2m[-a, -b, -c],
    0}, MetricOn → All, ContractMetrics → True]];

ChiParaB0mDefinition = TP0m[] /. PADMActivate // ToCanonical;
ChiParaB1pDefinition = TP1p[-i, -j] /. PADMActivate // ToCanonical;
ChiParaB1mDefinition = TP1m[-i] /. PADMActivate // ToCanonical;
ChiParaB2mDefinition = TP2m[-i, -j, -k] /. PADMActivate // ToCanonical;
ChiParaA0pDefinition = RP0p[] /. PADMActivate // ToCanonical;
ChiParaA0mDefinition = RP0m[] /. PADMActivate // ToCanonical;
ChiParaA1pDefinition = RP1p[-i, -j] /. PADMActivate // ToCanonical;
ChiParaA1mDefinition = RP1m[-i] /. PADMActivate // ToCanonical;
ChiParaA2pDefinition = RP2p[-i, -j] /. PADMActivate // ToCanonical;
ChiParaA2mDefinition = RP2m[-i, -j, -k] /. PADMActivate // ToCanonical;

ChiParaB0mActivate = MakeRule[{ChiParaB0m[], Evaluate[ChiParaB0mDefinition]},
   MetricOn → All, ContractMetrics → True];
ChiParaB1pActivate = MakeRule[{ChiParaB1p[-i, -j],
    Evaluate[ChiParaB1pDefinition]}, MetricOn → All, ContractMetrics → True];
ChiParaB1mActivate = MakeRule[{ChiParaB1m[-i], Evaluate[ChiParaB1mDefinition]},
   MetricOn → All, ContractMetrics → True];
ChiParaB2mActivate = MakeRule[{ChiParaB2m[-i, -j, -k],
```

```
    Evaluate[ChiParaB2mDefinition]}, MetricOn → All, ContractMetrics → True];
ChiParaA0pActivate = MakeRule[{ChiParaA0p[], Evaluate[ChiParaA0pDefinition]},
   MetricOn → All, ContractMetrics → True];
ChiParaA0mActivate = MakeRule[{ChiParaA0m[], Evaluate[ChiParaA0mDefinition]},
   MetricOn → All, ContractMetrics → True];
ChiParaA1pActivate = MakeRule[{ChiParaA1p[-i, -j],
    Evaluate[ChiParaA1pDefinition]}, MetricOn → All, ContractMetrics → True];
ChiParaA1mActivate = MakeRule[{ChiParaA1m[-i], Evaluate[ChiParaA1mDefinition]},
   MetricOn → All, ContractMetrics → True];
ChiParaA2pActivate = MakeRule[{ChiParaA2p[-i, -j],
    Evaluate[ChiParaA2pDefinition]}, MetricOn → All, ContractMetrics → True];
ChiParaA2mActivate = MakeRule[{ChiParaA2m[-i, -j, -k],
    Evaluate[ChiParaA2mDefinition]}, MetricOn → All, ContractMetrics → True];

ChiParaActivate = Join[ChiParaB0mActivate,
   ChiParaB1pActivate, ChiParaB1mActivate, ChiParaB2mActivate,
   ChiParaA0pActivate, ChiParaA0mActivate, ChiParaA1pActivate,
   ChiParaA1mActivate, ChiParaA2pActivate, ChiParaA2mActivate];
ClearBuild[];
```

build

Define $\mathbb{D}\pitchfork bJ^P$, $\mathbb{D}\pitchfork AJ^P$

build

```
DefTensor[DPiPB0p[-z], M4,
  PrintAs → SymbolBuild[PiPBSymb, Spin0p, "Derivative" -> 1]];
(*DeclareOrder[DPiPB0p[-z],1];*)
DefTensor[DPiPB1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPBSymb, Spin1p, "Derivative" -> 1]];
(*DeclareOrder[DPiPB1p[-z,-a,-b],1];*)
DefTensor[DPiPB1m[-z, -a], M4,
  PrintAs → SymbolBuild[PiPBSymb, Spin1m, "Derivative" -> 1]];
(*DeclareOrder[DPiPB1m[-z,-a],1];*)
DefTensor[DPiPB2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPBSymb, Spin2p, "Derivative" -> 1]];
(*DeclareOrder[DPiPB2p[-z,-a,-b],1];*)
AutomaticRules[DPiPB2p,
  MakeRule[{DPiPB2p[-z, a, -a], 0}, MetricOn → All, ContractMetrics → True]];
DefTensor[DPiPA0p[-z], M4,
  PrintAs → SymbolBuild[PiPASymb, Spin0p, "Derivative" -> 1]];
(*DeclareOrder[DPiPA0p[-z],1];*)
DefTensor[DPiPA0m[-z], M4,
  PrintAs → SymbolBuild[PiPASymb, Spin0m, "Derivative" -> 1]];
(*DeclareOrder[DPiPA0m[-z],1];*)
```

```
DefTensor[DPiPA1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPASymb, Spin1p, "Derivative" -> 1]];
(*DeclareOrder[DPiPA1p[-z,-a,-b],1];*)
DefTensor[DPiPA1m[-z, -a], M4,
  PrintAs → SymbolBuild[PiPASymb, Spin1m, "Derivative" -> 1]];
(*DeclareOrder[DPiPA1m[-z,-a],1];*)
DefTensor[DPiPA2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPASymb, Spin2p, "Derivative" -> 1]];
(*DeclareOrder[DPiPA2p[-z,-a,-b],1];*)
DefTensor[DPiPA2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPASymb, Spin2m, "Derivative" -> 1]];
(*DeclareOrder[DPiPA2m[-z,-a,-b,-c],1];*)
AutomaticRules[DPiPA2m,
  MakeRule[{DPiPA2m[-z, a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[DPiPA2m, MakeRule[{epsilonG[a, b, c, d] DPiPA2m[-z, -a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DPiPA2p, MakeRule[{DPiPA2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
DPiPB0pActivate = MakeRule[{CD[-z][PiPB0p[]], DPiPB0p[-z]},
   MetricOn → All, ContractMetrics → True];
DPiPB1pActivate = MakeRule[{CD[-z][PiPB1p[-a, -b]],
    DPiPB1p[-z, -a, -b] + A[i, -a, -z] PiPB1p[-i, -b] + A[i, -b, -z] PiPB1p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DPiPB1mActivate = MakeRule[{CD[-z][PiPB1m[-a]], DPiPB1m[-z, -a] +
     A[i, -a, -z] PiPB1m[-i]}, MetricOn → All, ContractMetrics → True];
DPiPB2pActivate = MakeRule[{CD[-z][PiPB2p[-a, -b]],
    DPiPB2p[-z, -a, -b] + A[i, -a, -z] PiPB2p[-i, -b] + A[i, -b, -z] PiPB2p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DPiPA0pActivate = MakeRule[{CD[-z][PiPA0p[]], DPiPA0p[-z]},
   MetricOn → All, ContractMetrics → True];
DPiPA0mActivate = MakeRule[{CD[-z][PiPA0m[]], DPiPA0m[-z]},
   MetricOn → All, ContractMetrics → True];
DPiPA1pActivate = MakeRule[{CD[-z][PiPA1p[-a, -b]],
    DPiPA1p[-z, -a, -b] + A[i, -a, -z] PiPA1p[-i, -b] + A[i, -b, -z] PiPA1p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DPiPA1mActivate = MakeRule[{CD[-z][PiPA1m[-a]], DPiPA1m[-z, -a] +
     A[i, -a, -z] PiPA1m[-i]}, MetricOn → All, ContractMetrics → True];
DPiPA2pActivate = MakeRule[{CD[-z][PiPA2p[-a, -b]],
    DPiPA2p[-z, -a, -b] + A[i, -a, -z] PiPA2p[-i, -b] + A[i, -b, -z] PiPA2p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DPiPA2mActivate = MakeRule[{CD[-z][PiPA2m[-a, -b, -c]],
    DPiPA2m[-z, -a, -b, -c] + A[i, -a, -z] PiPA2m[-i, -b, -c] +
     A[i, -b, -z] PiPA2m[-a, -i, -c] + A[i, -c, -z] PiPA2m[-a, -b, -i]},
   MetricOn → All, ContractMetrics → True];
```

```
DPiPActivate = Join[DPiPB0pActivate, DPiPB1pActivate, DPiPB1mActivate,
   DPiPB2pActivate, DPiPA0pActivate, DPiPA0mActivate, DPiPA1pActivate,
   DPiPA1mActivate, DPiPA2pActivate, DPiPA2mActivate];

(*the rules below should of course be generalised beyond simply the momenta*)
DPiPB0pDeactivate = MakeRule[
   {DPiPB0p[-z], CD[-z][PiPB0p[]]}, MetricOn → All, ContractMetrics → True];
DPiPB1pDeactivate = MakeRule[{DPiPB1p[-z, -a, -b], CD[-z][PiPB1p[-a, -b]] -
     A[i, -a, -z] PiPB1p[-i, -b] - A[i, -b, -z] PiPB1p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DPiPB1mDeactivate = MakeRule[{DPiPB1m[-z, -a], CD[-z][PiPB1m[-a]] -
     A[i, -a, -z] PiPB1m[-i]}, MetricOn → All, ContractMetrics → True];
DPiPB2pDeactivate = MakeRule[{DPiPB2p[-z, -a, -b], CD[-z][PiPB2p[-a, -b]] -
     A[i, -a, -z] PiPB2p[-i, -b] - A[i, -b, -z] PiPB2p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DPiPA0pDeactivate = MakeRule[{DPiPA0p[-z], CD[-z][PiPA0p[]]},
   MetricOn → All, ContractMetrics → True];
DPiPA0mDeactivate = MakeRule[{DPiPA0m[-z], CD[-z][PiPA0m[]]},
   MetricOn → All, ContractMetrics → True];
DPiPA1pDeactivate = MakeRule[{DPiPA1p[-z, -a, -b], CD[-z][PiPA1p[-a, -b]] -
     A[i, -a, -z] PiPA1p[-i, -b] - A[i, -b, -z] PiPA1p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DPiPA1mDeactivate = MakeRule[{DPiPA1m[-z, -a], CD[-z][PiPA1m[-a]] -
     A[i, -a, -z] PiPA1m[-i]}, MetricOn → All, ContractMetrics → True];
DPiPA2pDeactivate = MakeRule[{DPiPA2p[-z, -a, -b], CD[-z][PiPA2p[-a, -b]] -
     A[i, -a, -z] PiPA2p[-i, -b] - A[i, -b, -z] PiPA2p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DPiPA2mDeactivate = MakeRule[{DPiPA2m[-z, -a, -b, -c],
    CD[-z][PiPA2m[-a, -b, -c]] - A[i, -a, -z] PiPA2m[-i, -b, -c] -
     A[i, -b, -z] PiPA2m[-a, -i, -c] - A[i, -c, -z] PiPA2m[-a, -b, -i]},
   MetricOn → All, ContractMetrics → True];
DPiPDeactivate = Join[DPiPB0pDeactivate, DPiPB1pDeactivate, DPiPB1mDeactivate,
   DPiPB2pDeactivate, DPiPA0pDeactivate, DPiPA0mDeactivate, DPiPA1pDeactivate,
   DPiPA1mDeactivate, DPiPA2pDeactivate, DPiPA2mDeactivate];
ClearBuild[];
```

build

Define $\hat{\mathbb{D}}\pi\, \mathrm{bJ}^P$, $\hat{\mathbb{D}}\pi\, \mathrm{AJ}^P$

build

```
DefTensor[DpPiPB0p[-z], M4, PrintAs →
   SymbolBuild[PiPBSymb, Spin0p, "Derivative" -> 2], OrthogonalTo → {V[z]}];
DeclareOrder[DpPiPB0p[-z], 1];
DeclareOrder[DPiPB0p[-z], 1, "approximation" →
```

```
    B[w, -z] DpPiPB0p[-w] + V[-v] B[v, -z] V[u] H[-u, w] DPiPB0p[-w]];
DefTensor[DpPiPB1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPBSymb, Spin1p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpPiPB1p[-z, -a, -b], 1];
DeclareOrder[DPiPB1p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpPiPB1p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DPiPB1p[-w, -a, -b]];
DefTensor[DpPiPB1m[-z, -a], M4, PrintAs → SymbolBuild[PiPBSymb,
    Spin1m, "Derivative" -> 2], OrthogonalTo → {V[z], V[a]}];
DeclareOrder[DpPiPB1m[-z, -a], 1];
DeclareOrder[DPiPB1m[-z, -a], 1, "approximation" →
    B[w, -z] DpPiPB1m[-w, -a] + V[-v] B[v, -z] V[u] H[-u, w] DPiPB1m[-w, -a]];
DefTensor[DpPiPB2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPBSymb, Spin2p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpPiPB2p[-z, -a, -b], 1];
DeclareOrder[DPiPB2p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpPiPB2p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DPiPB2p[-w, -a, -b]];
AutomaticRules[DpPiPB2p, MakeRule[{DpPiPB2p[-z, a, -a], 0},
    MetricOn → All, ContractMetrics → True]];
DefTensor[DpPiPA0p[-z], M4, PrintAs → SymbolBuild[PiPASymb,
    Spin0p, "Derivative" -> 2], OrthogonalTo → {V[z]}];
DeclareOrder[DpPiPA0p[-z], 1];
DeclareOrder[DPiPA0p[-z], 1, "approximation" →
    B[w, -z] DpPiPA0p[-w] + V[-v] B[v, -z] V[u] H[-u, w] DPiPA0p[-w]];
DefTensor[DpPiPA0m[-z], M4, PrintAs → SymbolBuild[PiPASymb,
    Spin0m, "Derivative" -> 2], OrthogonalTo → {V[z]}];
DeclareOrder[DpPiPA0m[-z], 1];
DeclareOrder[DPiPA0m[-z], 1, "approximation" →
    B[w, -z] DpPiPA0m[-w] + V[-v] B[v, -z] V[u] H[-u, w] DPiPA0m[-w]];
DefTensor[DpPiPA1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPASymb, Spin1p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpPiPA1p[-z, -a, -b], 1];
DeclareOrder[DPiPA1p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpPiPA1p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DPiPA1p[-w, -a, -b]];
DefTensor[DpPiPA1m[-z, -a], M4, PrintAs → SymbolBuild[PiPASymb,
    Spin1m, "Derivative" -> 2], OrthogonalTo → {V[z], V[a]}];
DeclareOrder[DpPiPA1m[-z, -a], 1];
DeclareOrder[DPiPA1m[-z, -a], 1, "approximation" →
    B[w, -z] DpPiPA1m[-w, -a] + V[-v] B[v, -z] V[u] H[-u, w] DPiPA1m[-w, -a]];
```

```
DefTensor[DpPiPA2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPASymb, Spin2p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpPiPA2p[-z, -a, -b], 1];
DeclareOrder[DPiPA2p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpPiPA2p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DPiPA2p[-w, -a, -b]];
DefTensor[DpPiPA2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[PiPASymb, Spin2m, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b], V[c]}];
DeclareOrder[DpPiPA2m[-z, -a, -b, -c], 1];
DeclareOrder[DPiPA2m[-z, -a, -b, -c],
  1, "approximation" → B[w, -z] DpPiPA2m[-w, -a, -b, -c] +
    V[-v] B[v, -z] V[u] H[-u, w] DPiPA2m[-w, -a, -b, -c]];
AutomaticRules[DpPiPA2m, MakeRule[{DpPiPA2m[-z, a, -b, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpPiPA2m, MakeRule[{epsilonG[a, b, c, d] DpPiPA2m[-z, -a, -b, -c],
    0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpPiPA2p, MakeRule[{DpPiPA2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
DpPiPB0pActivate = MakeRule[{G3[-y, z] DPiPB0p[-z],
    G3[-y, z] B[x, -z] DpPiPB0p[-x]}, MetricOn → All, ContractMetrics → True];
DpPiPB1pActivate = MakeRule[{G3[-y, z] DPiPB1p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpPiPB1p[-x, -a, -b] +
        (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
         G3[-y, z] DPiPB1p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPB1mActivate = MakeRule[{G3[-y, z] DPiPB1m[-z, -a],
    Evaluate[G3[-y, z] B[x, -z] DpPiPB1m[-x, -a] +
        (G[-a, i] - PPara[-a, i]) G3[-y, z] DPiPB1m[-z, -i] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPB2pActivate = MakeRule[{G3[-y, z] DPiPB2p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpPiPB2p[-x, -a, -b] +
        (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
         G3[-y, z] DPiPB2p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPA0pActivate = MakeRule[{G3[-y, z] DPiPA0p[-z],
    G3[-y, z] B[x, -z] DpPiPA0p[-x]}, MetricOn → All, ContractMetrics → True];
DpPiPA0mActivate = MakeRule[{G3[-y, z] DPiPA0m[-z],
    G3[-y, z] B[x, -z] DpPiPA0m[-x]}, MetricOn → All, ContractMetrics → True];
DpPiPA1pActivate = MakeRule[{G3[-y, z] DPiPA1p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpPiPA1p[-x, -a, -b] +
        (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
```

```
        G3[-y, z] DPiPA1p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPA1mActivate = MakeRule[{G3[-y, z] DPiPA1m[-z, -a],
    Evaluate[G3[-y, z] B[x, -z] DpPiPA1m[-x, -a] +
       (G[-a, i] - PPara[-a, i]) G[-y, z] DPiPA1m[-z, -i] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPA2pActivate = MakeRule[{G3[-y, z] DPiPA2p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpPiPA2p[-x, -a, -b] +
       (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
        G3[-y, z] DPiPA2p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPA2mActivate = MakeRule[{G3[-y, z] DPiPA2m[-z, -a, -b, -c],
    Evaluate[G3[-y, z] B[x, -z] DpPiPA2m[-x, -a, -b, -c] +
       (G[-a, i] G[-b, j] G[-c, k] - PPara[-a, i] PPara[-b, j] PPara[-c, k])
        G3[-y, z] DPiPA2m[-z, -i, -j, -k] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPActivate = Join[DpPiPB0pActivate, DpPiPB1pActivate, DpPiPB1mActivate,
   DpPiPB2pActivate, DpPiPA0pActivate, DpPiPA0mActivate, DpPiPA1pActivate,
   DpPiPA1mActivate, DpPiPA2pActivate, DpPiPA2mActivate];

(*again this should be extended over other derivatives,
multiply the above by PPara[-w,v]H[-v,y]*)
DpPiPB0pDeactivate =
  MakeRule[{DpPiPB0p[-w], PPara[-w, v] H[-v, y] G3[-y, z] DPiPB0p[-z]},
   MetricOn → All, ContractMetrics → True];
DpPiPB1pDeactivate = MakeRule[{DpPiPB1p[-w, -a, -b],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DPiPB1p[-z, -a, -b] -
       PPara[-w, v] H[-v, y] (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
        G3[-y, z] DPiPB1p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPB1mDeactivate = MakeRule[{DpPiPB1m[-w, -a],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DPiPB1m[-z, -a] -
       PPara[-w, v] H[-v, y] (G[-a, i] - PPara[-a, i]) G3[-y, z] DPiPB1m[-z, -i] /.
      PADMActivate]}, MetricOn → All, ContractMetrics → True];
DpPiPB2pDeactivate = MakeRule[{DpPiPB2p[-w, -a, -b],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DPiPB2p[-z, -a, -b] -
       PPara[-w, v] H[-v, y] (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
        G3[-y, z] DPiPB2p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPA0pDeactivate = MakeRule[{DpPiPA0p[-w], PPara[-w, v] H[-v, y]
     G3[-y, z] DPiPA0p[-z]}, MetricOn → All, ContractMetrics → True];
DpPiPA0mDeactivate = MakeRule[{DpPiPA0m[-w], PPara[-w, v] H[-v, y]
```

```
      G3[-y, z] DPiPA0m[-z]}, MetricOn → All, ContractMetrics → True];
DpPiPA1pDeactivate = MakeRule[{DpPiPA1p[-w, -a, -b],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DPiPA1p[-z, -a, -b] -
       PPara[-w, v] H[-v, y] (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
        G3[-y, z] DPiPA1p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPA1mDeactivate = MakeRule[{DpPiPA1m[-w, -a],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DPiPA1m[-z, -a] -
       PPara[-w, v] H[-v, y] (G[-a, i] - PPara[-a, i]) G3[-y, z] DPiPA1m[-z, -i] /.
      PADMActivate]}, MetricOn → All, ContractMetrics → True];
DpPiPA2pDeactivate = MakeRule[{DpPiPA2p[-w, -a, -b],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DPiPA2p[-z, -a, -b] -
       PPara[-w, v] H[-v, y] (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
        G3[-y, z] DPiPA2p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPA2mDeactivate = MakeRule[{DpPiPA2m[-w, -a, -b, -c], Evaluate[
     PPara[-w, v] H[-v, y] G3[-y, z] DPiPA2m[-z, -a, -b, -c] - PPara[-w, v] H[-v, y]
        (G[-a, i] G[-b, j] G[-c, k] - PPara[-a, i] PPara[-b, j] PPara[-c, k])
        G3[-y, z] DPiPA2m[-z, -i, -j, -k] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpPiPDeactivate = Join[DpPiPB0pDeactivate, DpPiPB1pDeactivate,
   DpPiPB1mDeactivate, DpPiPB2pDeactivate, DpPiPA0pDeactivate,
   DpPiPA0mDeactivate, DpPiPA1pDeactivate, DpPiPA1mDeactivate,
   DpPiPA2pDeactivate, DpPiPA2mDeactivate];
ClearBuild[];
```

build

Define $\mathbb{D}$ubJ^P, $\mathbb{D}$uAJ^P

build

```
DefTensor[DUB0p[-z], M4,
  PrintAs → SymbolBuild[UBSymb, Spin0p, "Derivative" -> 1]];
DeclareOrder[DUB0p[-z], 1];
DefTensor[DUB1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[UBSymb, Spin1p, "Derivative" -> 1]];
DeclareOrder[DUB1p[-z, -a, -b], 1];
DefTensor[DUB1m[-z, -a], M4,
  PrintAs → SymbolBuild[UBSymb, Spin1m, "Derivative" -> 1]];
DeclareOrder[DUB1m[-z, -a], 1];
DefTensor[DUB2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[UBSymb, Spin2p, "Derivative" -> 1]];
DeclareOrder[DUB2p[-z, -a, -b], 1];
AutomaticRules[DUB2p,
  MakeRule[{DUB2p[-z, a, -a], 0}, MetricOn → All, ContractMetrics → True]];
DefTensor[DUA0p[-z], M4, PrintAs →
   SymbolBuild[UASymb, Spin0p, "Derivative" -> 1]];
DeclareOrder[DUA0p[-z], 1];
DefTensor[DUA0m[-z], M4,
  PrintAs → SymbolBuild[UASymb, Spin0m, "Derivative" -> 1]];
DeclareOrder[DUA0m[-z], 1];
DefTensor[DUA1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[UASymb, Spin1p, "Derivative" -> 1]];
DeclareOrder[DUA1p[-z, -a, -b], 1];
DefTensor[DUA1m[-z, -a], M4,
  PrintAs → SymbolBuild[UASymb, Spin1m, "Derivative" -> 1]];
DeclareOrder[DUA1m[-z, -a], 1];
DefTensor[DUA2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[UASymb, Spin2p, "Derivative" -> 1]];
DeclareOrder[DUA2p[-z, -a, -b], 1];
DefTensor[DUA2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[UASymb, Spin2m, "Derivative" -> 1]];
DeclareOrder[DUA2m[-z, -a, -b, -c], 1];
AutomaticRules[DUA2m,
  MakeRule[{DUA2m[-z, a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[DUA2m, MakeRule[{epsilonG[a, b, c, d] DUA2m[-z, -a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DUA2p, MakeRule[{DUA2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
ClearBuild[];
```

build

Define $\hat{\mathbb{D}}$ubJ^P, $\hat{\mathbb{D}}$uAJ^P

build

```
DefTensor[DpUB0p[-z], M4,
```

```
  PrintAs → SymbolBuild[UBSymb, Spin0p, "Derivative" -> 2], OrthogonalTo → {V[z]}];
DeclareOrder[DpUB0p[-z], 1];
DeclareOrder[DUB0p[-z], 1,
  "approximation" → B[w, -z] DpUB0p[-w] + V[-v] B[v, -z] V[u] H[-u, w] DUB0p[-w]];
DefTensor[DpUB1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[UBSymb, Spin1p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpUB1p[-z, -a, -b], 1];
DeclareOrder[DUB1p[-z, -a, -b], 1, "approximation" →
   B[w, -z] DpUB1p[-w, -a, -b] + V[-v] B[v, -z] V[u] H[-u, w] DUB1p[-w, -a, -b]];
DefTensor[DpUB1m[-z, -a], M4, PrintAs → SymbolBuild[UBSymb,
    Spin1m, "Derivative" -> 2], OrthogonalTo → {V[z], V[a]}];
DeclareOrder[DpUB1m[-z, -a], 1];
DeclareOrder[DUB1m[-z, -a], 1, "approximation" →
   B[w, -z] DpUB1m[-w, -a] + V[-v] B[v, -z] V[u] H[-u, w] DUB1m[-w, -a]];
DefTensor[DpUB2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[UBSymb, Spin2p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpUB2p[-z, -a, -b], 1];
DeclareOrder[DUB2p[-z, -a, -b], 1, "approximation" →
   B[w, -z] DpUB2p[-w, -a, -b] + V[-v] B[v, -z] V[u] H[-u, w] DUB2p[-w, -a, -b]];
AutomaticRules[DpUB2p, MakeRule[{DpUB2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
DefTensor[DpUA0p[-z], M4, PrintAs → SymbolBuild[UASymb, Spin0p, "Derivative" -> 2],
  OrthogonalTo → {V[z]}];
DeclareOrder[DpUA0p[-z], 1];
DeclareOrder[DUA0p[-z], 1,
  "approximation" → B[w, -z] DpUA0p[-w] + V[-v] B[v, -z] V[u] H[-u, w] DUA0p[-w]];
DefTensor[DpUA0m[-z], M4, PrintAs → SymbolBuild[UASymb, Spin0m, "Derivative" -> 2],
  OrthogonalTo → {V[z]}];
DeclareOrder[DpUA0m[-z], 1];
DeclareOrder[DUA0m[-z], 1,
  "approximation" → B[w, -z] DpUA0m[-w] + V[-v] B[v, -z] V[u] H[-u, w] DUA0m[-w]];
DefTensor[DpUA1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[UASymb, Spin1p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpUA1p[-z, -a, -b], 1];
DeclareOrder[DUA1p[-z, -a, -b], 1, "approximation" →
   B[w, -z] DpUA1p[-w, -a, -b] + V[-v] B[v, -z] V[u] H[-u, w] DUA1p[-w, -a, -b]];
DefTensor[DpUA1m[-z, -a], M4, PrintAs → SymbolBuild[UASymb,
    Spin1m, "Derivative" -> 2], OrthogonalTo → {V[z], V[a]}];
DeclareOrder[DpUA1m[-z, -a], 1];
DeclareOrder[DUA1m[-z, -a], 1, "approximation" →
   B[w, -z] DpUA1m[-w, -a] + V[-v] B[v, -z] V[u] H[-u, w] DUA1m[-w, -a]];
```

```
DefTensor[DpUA2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[UASymb, Spin2p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpUA2p[-z, -a, -b], 1];
DeclareOrder[DUA2p[-z, -a, -b], 1, "approximation" →
   B[w, -z] DpUA2p[-w, -a, -b] + V[-v] B[v, -z] V[u] H[-u, w] DUA2p[-w, -a, -b]];
DefTensor[DpUA2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[UASymb, Spin2m, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b], V[c]}];
DeclareOrder[DpUA2m[-z, -a, -b, -c], 1];
DeclareOrder[DUA2m[-z, -a, -b, -c],
  1, "approximation" → B[w, -z] DpUA2m[-w, -a, -b, -c] +
    V[-v] B[v, -z] V[u] H[-u, w] DUA2m[-w, -a, -b, -c]];
AutomaticRules[DpUA2m, MakeRule[{DpUA2m[-z, a, -b, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpUA2m, MakeRule[{epsilonG[a, b, c, d] DpUA2m[-z, -a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpUA2p, MakeRule[{DpUA2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
ClearBuild[];
```

build

Define $\mathbb{D}\hat{T}^{J^P}, \mathbb{D}\hat{R}^{J^P}$

build

```
DefTensor[DTP0m[-z], M4,
  PrintAs → SymbolBuild[TPSymb, Spin0m, "Derivative" -> 1]];
(*DeclareOrder[DTP0m[-z],1];*)
DefTensor[DTP1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TPSymb, Spin1p, "Derivative" -> 1]];
(*DeclareOrder[DTP1p[-z,-a,-b],1];*)
DefTensor[DTP1m[-z, -a], M4,
  PrintAs → SymbolBuild[TPSymb, Spin1m, "Derivative" -> 1]];
(*DeclareOrder[DTP1m[-z,-a],1];*)
DefTensor[DTP2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TPSymb, Spin2m, "Derivative" -> 1]];
(*DeclareOrder[DTP2m[-z,-a,-b,-c],1];*)
AutomaticRules[DTP2m,
  MakeRule[{DTP2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[DTP2m, MakeRule[{epsilonG[a, b, c, d] DTP2m[-a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
DefTensor[DRP0p[-z], M4, PrintAs →
   SymbolBuild[RPSymb, Spin0p, "Derivative" -> 1]];
(*DeclareOrder[DRP0p[-z],1];*)
```

```
DefTensor[DRP0m[-z], M4,
  PrintAs → SymbolBuild[RPSymb, Spin0m, "Derivative" -> 1]];
(*DeclareOrder[DRP0m[-z],1];*)
DefTensor[DRP1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPSymb, Spin1p, "Derivative" -> 1]];
(*DeclareOrder[DRP1p[-z,-a,-b],1];*)
DefTensor[DRP1m[-z, -a], M4,
  PrintAs → SymbolBuild[RPSymb, Spin1m, "Derivative" -> 1]];
(*DeclareOrder[DRP1m[-z,-a],1];*)
DefTensor[DRP2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPSymb, Spin2p, "Derivative" -> 1]];
(*DeclareOrder[DRP2p[-z,-a,-b],1];*)
DefTensor[DRP2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPSymb, Spin2m, "Derivative" -> 1]];
(*DeclareOrder[DRP2m[-z,-a,-b,-c],1];*)
AutomaticRules[DRP2m,
  MakeRule[{DRP2m[-z, a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[DRP2m, MakeRule[{epsilonG[a, b, c, d] DRP2m[-z, -a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DRP2p, MakeRule[{DRP2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
DTP0mActivate = MakeRule[{CD[-z][TP0m[]], DTP0m[-z]},
   MetricOn → All, ContractMetrics → True];
DTP1pActivate = MakeRule[{CD[-z][TP1p[-a, -b]],
    DTP1p[-z, -a, -b] + A[i, -a, -z] TP1p[-i, -b] + A[i, -b, -z] TP1p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DTP1mActivate = MakeRule[{CD[-z][TP1m[-a]], DTP1m[-z, -a] + A[i, -a, -z] TP1m[-i]},
   MetricOn → All, ContractMetrics → True];
DTP2mActivate = MakeRule[{CD[-z][TP2m[-a, -b, -c]],
    DTP2m[-z, -a, -b, -c] + A[i, -a, -z] TP2m[-i, -b, -c] +
     A[i, -b, -z] TP2m[-a, -i, -c] + A[i, -c, -z] TP2m[-a, -b, -i]},
   MetricOn → All, ContractMetrics → True];
DRP0pActivate = MakeRule[{CD[-z][RP0p[]], DRP0p[-z]},
   MetricOn → All, ContractMetrics → True];
DRP0mActivate = MakeRule[{CD[-z][RP0m[]], DRP0m[-z]},
   MetricOn → All, ContractMetrics → True];
DRP1pActivate = MakeRule[{CD[-z][RP1p[-a, -b]],
    DRP1p[-z, -a, -b] + A[i, -a, -z] RP1p[-i, -b] + A[i, -b, -z] RP1p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DRP1mActivate = MakeRule[{CD[-z][RP1m[-a]], DRP1m[-z, -a] + A[i, -a, -z] RP1m[-i]},
   MetricOn → All, ContractMetrics → True];
DRP2pActivate = MakeRule[{CD[-z][RP2p[-a, -b]],
    DRP2p[-z, -a, -b] + A[i, -a, -z] RP2p[-i, -b] + A[i, -b, -z] RP2p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
```

```
DRP2mActivate = MakeRule[{CD[-z][RP2m[-a, -b, -c]],
    DRP2m[-z, -a, -b, -c] + A[i, -a, -z] RP2m[-i, -b, -c] +
     A[i, -b, -z] RP2m[-a, -i, -c] + A[i, -c, -z] RP2m[-a, -b, -i]},
   MetricOn → All, ContractMetrics → True];
DRPActivate = Join[DTP0mActivate, DTP1pActivate, DTP1mActivate,
   DTP2mActivate, DRP0pActivate, DRP0mActivate, DRP1pActivate,
   DRP1mActivate, DRP2pActivate, DRP2mActivate];

(*the rules below should of course be generalised beyond simply
 the momenta -- these below now generalise to the field strengths*)
DTP0mDeactivate = MakeRule[{DTP0m[-z], CD[-z][TP0m[]]},
   MetricOn → All, ContractMetrics → True];
DTP1pDeactivate = MakeRule[{DTP1p[-z, -a, -b],
    CD[-z][TP1p[-a, -b]] - A[i, -a, -z] TP1p[-i, -b] - A[i, -b, -z] TP1p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DTP1mDeactivate = MakeRule[{DTP1m[-z, -a], CD[-z][TP1m[-a]] -
     A[i, -a, -z] TP1m[-i]}, MetricOn → All, ContractMetrics → True];
DTP2mDeactivate = MakeRule[{DTP2m[-z, -a, -b, -c], CD[-z][TP2m[-a, -b, -c]] -
     A[i, -a, -z] TP2m[-i, -b, -c] - A[i, -b, -z] TP2m[-a, -i, -c] -
     A[i, -c, -z] TP2m[-a, -b, -i]}, MetricOn → All, ContractMetrics → True];
DRP0pDeactivate = MakeRule[{DRP0p[-z], CD[-z][RP0p[]]},
   MetricOn → All, ContractMetrics → True];
DRP0mDeactivate = MakeRule[{DRP0m[-z], CD[-z][RP0m[]]},
   MetricOn → All, ContractMetrics → True];
DRP1pDeactivate = MakeRule[{DRP1p[-z, -a, -b],
    CD[-z][RP1p[-a, -b]] - A[i, -a, -z] RP1p[-i, -b] - A[i, -b, -z] RP1p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DRP1mDeactivate = MakeRule[{DRP1m[-z, -a], CD[-z][RP1m[-a]] -
     A[i, -a, -z] RP1m[-i]}, MetricOn → All, ContractMetrics → True];
DRP2pDeactivate = MakeRule[{DRP2p[-z, -a, -b],
    CD[-z][RP2p[-a, -b]] - A[i, -a, -z] RP2p[-i, -b] - A[i, -b, -z] RP2p[-a, -i]},
   MetricOn → All, ContractMetrics → True];
DRP2mDeactivate = MakeRule[{DRP2m[-z, -a, -b, -c],
    CD[-z][RP2m[-a, -b, -c]] - A[i, -a, -z] RP2m[-i, -b, -c] -
     A[i, -b, -z] RP2m[-a, -i, -c] - A[i, -c, -z] RP2m[-a, -b, -i]},
   MetricOn → All, ContractMetrics → True];
DRPDeactivate = Join[DTP0mDeactivate, DTP1pDeactivate, DTP1mDeactivate,
   DTP2mDeactivate, DRP0pDeactivate, DRP0mDeactivate, DRP1pDeactivate,
   DRP1mDeactivate, DRP2pDeactivate, DRP2mDeactivate];
ClearBuild[];
```

build

Define $\mathbb{D}\hat{\lambda}J^P$

build

```
DefTensor[DTLambdaP0m[-z], M4,
```

```
  PrintAs → SymbolBuild[TLambdaPSymb, Spin0m, "Derivative" -> 1]];
(*DeclareOrder[DTLambdaP0m[-z],1];*)
DefTensor[DTLambdaP1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPSymb, Spin1p, "Derivative" -> 1]];
(*DeclareOrder[DTLambdaP1p[-z,-a,-b],1];*)
DefTensor[DTLambdaP1m[-z, -a], M4,
  PrintAs → SymbolBuild[TLambdaPSymb, Spin1m, "Derivative" -> 1]];
(*DeclareOrder[DTLambdaP1m[-z,-a],1];*)
DefTensor[DTLambdaP2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPSymb, Spin2m, "Derivative" -> 1]];
(*DeclareOrder[DTLambdaP2m[-z,-a,-b,-c],1];*)
AutomaticRules[DTLambdaP2m,
  MakeRule[{DTLambdaP2m[a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[DTLambdaP2m, MakeRule[{epsilonG[a, b, c, d] DTLambdaP2m[-a, -b, -c],
    0}, MetricOn → All, ContractMetrics → True]];
DefTensor[DRLambdaP0p[-z], M4, PrintAs →
   SymbolBuild[RLambdaPSymb, Spin0p, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaP0p[-z],1];*)
DefTensor[DRLambdaP0m[-z], M4,
  PrintAs → SymbolBuild[RLambdaPSymb, Spin0m, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaP0m[-z],1];*)
DefTensor[DRLambdaP1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPSymb, Spin1p, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaP1p[-z,-a,-b],1];*)
DefTensor[DRLambdaP1m[-z, -a], M4,
  PrintAs → SymbolBuild[RLambdaPSymb, Spin1m, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaP1m[-z,-a],1];*)
DefTensor[DRLambdaP2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPSymb, Spin2p, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaP2p[-z,-a,-b],1];*)
DefTensor[DRLambdaP2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPSymb, Spin2m, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaP2m[-z,-a,-b,-c],1];*)
AutomaticRules[DRLambdaP2m, MakeRule[
   {DRLambdaP2m[-z, a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[DRLambdaP2m, MakeRule[
   {epsilonG[a, b, c, d] DRLambdaP2m[-z, -a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DRLambdaP2p, MakeRule[{DRLambdaP2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
DTLambdaP0mActivate = MakeRule[{CD[-z][TLambdaP0m[]], DTLambdaP0m[-z]},
   MetricOn → All, ContractMetrics → True];
DTLambdaP1pActivate = MakeRule[{CD[-z][TLambdaP1p[-a, -b]],
    DTLambdaP1p[-z, -a, -b] + A[i, -a, -z] TLambdaP1p[-i, -b] +
```

```
      A[i, -b, -z] TLambdaP1p[-a, -i]}, MetricOn → All, ContractMetrics → True];
DTLambdaP1mActivate = MakeRule[{CD[-z][TLambdaP1m[-a]], DTLambdaP1m[-z, -a] +
      A[i, -a, -z] TLambdaP1m[-i]}, MetricOn → All, ContractMetrics → True];
DTLambdaP2mActivate = MakeRule[{CD[-z][TLambdaP2m[-a, -b, -c]],
     DTLambdaP2m[-z, -a, -b, -c] + A[i, -a, -z] TLambdaP2m[-i, -b, -c] +
      A[i, -b, -z] TLambdaP2m[-a, -i, -c] + A[i, -c, -z] TLambdaP2m[-a, -b, -i]},
    MetricOn → All, ContractMetrics → True];
DRLambdaP0pActivate = MakeRule[{CD[-z][RLambdaP0p[]], DRLambdaP0p[-z]},
    MetricOn → All, ContractMetrics → True];
DRLambdaP0mActivate = MakeRule[{CD[-z][RLambdaP0m[]], DRLambdaP0m[-z]},
    MetricOn → All, ContractMetrics → True];
DRLambdaP1pActivate = MakeRule[{CD[-z][RLambdaP1p[-a, -b]],
     DRLambdaP1p[-z, -a, -b] + A[i, -a, -z] RLambdaP1p[-i, -b] +
      A[i, -b, -z] RLambdaP1p[-a, -i]}, MetricOn → All, ContractMetrics → True];
DRLambdaP1mActivate = MakeRule[{CD[-z][RLambdaP1m[-a]], DRLambdaP1m[-z, -a] +
      A[i, -a, -z] RLambdaP1m[-i]}, MetricOn → All, ContractMetrics → True];
DRLambdaP2pActivate = MakeRule[{CD[-z][RLambdaP2p[-a, -b]],
     DRLambdaP2p[-z, -a, -b] + A[i, -a, -z] RLambdaP2p[-i, -b] +
      A[i, -b, -z] RLambdaP2p[-a, -i]}, MetricOn → All, ContractMetrics → True];
DRLambdaP2mActivate = MakeRule[{CD[-z][RLambdaP2m[-a, -b, -c]],
     DRLambdaP2m[-z, -a, -b, -c] + A[i, -a, -z] RLambdaP2m[-i, -b, -c] +
      A[i, -b, -z] RLambdaP2m[-a, -i, -c] + A[i, -c, -z] RLambdaP2m[-a, -b, -i]},
    MetricOn → All, ContractMetrics → True];
DRLambdaPActivate = Join[DTLambdaP0mActivate, DTLambdaP1pActivate,
    DTLambdaP1mActivate, DTLambdaP2mActivate, DRLambdaP0pActivate,
    DRLambdaP0mActivate, DRLambdaP1pActivate, DRLambdaP1mActivate,
    DRLambdaP2pActivate, DRLambdaP2mActivate];
ClearBuild[];
```

build

Define $\mathbb{D}\overset{*}{\lambda}J^P$

build

```
DefTensor[DTLambdaPerp0p[-z], M4,
  PrintAs → SymbolBuild[TLambdaPerpSymb, Spin0p, "Derivative" -> 1]];
(*DeclareOrder[DTLambdaPerp0p[-z],1];*)
DefTensor[DTLambdaPerp1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPerpSymb, Spin1p, "Derivative" -> 1]];
(*DeclareOrder[DTLambdaPerp1p[-z,-a,-b],1];*)
DefTensor[DTLambdaPerp1m[-z, -a], M4,
  PrintAs → SymbolBuild[TLambdaPerpSymb, Spin1m, "Derivative" -> 1]];
(*DeclareOrder[DTLambdaPerp1m[-z,-a],1];*)
DefTensor[DTLambdaPerp2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPerpSymb, Spin2p, "Derivative" -> 1]];
```

```
(*DeclareOrder[DTLambdaPerp2p[-z,-a,-b],1];*)
AutomaticRules[DTLambdaPerp2p, MakeRule[
    {DTLambdaPerp2p[-z, a, -a], 0}, MetricOn → All, ContractMetrics → True]];
DefTensor[DRLambdaPerp0p[-z], M4, PrintAs →
    SymbolBuild[RLambdaPerpSymb, Spin0p, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaPerp0p[-z],1];*)
DefTensor[DRLambdaPerp0m[-z], M4,
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin0m, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaPerp0m[-z],1];*)
DefTensor[DRLambdaPerp1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin1p, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaPerp1p[-z,-a,-b],1];*)
DefTensor[DRLambdaPerp1m[-z, -a], M4,
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin1m, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaPerp1m[-z,-a],1];*)
DefTensor[DRLambdaPerp2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin2p, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaPerp2p[-z,-a,-b],1];*)
DefTensor[DRLambdaPerp2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin2m, "Derivative" -> 1]];
(*DeclareOrder[DRLambdaPerp2m[-z,-a,-b,-c],1];*)
AutomaticRules[DRLambdaPerp2m, MakeRule[
    {DRLambdaPerp2m[-z, a, -b, -a], 0}, MetricOn → All, ContractMetrics → True]];
AutomaticRules[DRLambdaPerp2m, MakeRule[
    {epsilonG[a, b, c, d] DRLambdaPerp2m[-z, -a, -b, -c], 0},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[DRLambdaPerp2p, MakeRule[{DRLambdaPerp2p[-z, a, -a], 0},
    MetricOn → All, ContractMetrics → True]];
DTLambdaPerp0pActivate = MakeRule[{CD[-z][TLambdaPerp0p[]], DTLambdaPerp0p[-z]},
    MetricOn → All, ContractMetrics → True];
DTLambdaPerp1pActivate = MakeRule[{CD[-z][TLambdaPerp1p[-a, -b]],
     DTLambdaPerp1p[-z, -a, -b] + A[i, -a, -z] TLambdaPerp1p[-i, -b] +
      A[i, -b, -z] TLambdaPerp1p[-a, -i]}, MetricOn → All, ContractMetrics → True];
DTLambdaPerp1mActivate = MakeRule[{CD[-z][TLambdaPerp1m[-a]],
     DTLambdaPerp1m[-z, -a] + A[i, -a, -z] TLambdaPerp1m[-i]},
    MetricOn → All, ContractMetrics → True];
DTLambdaPerp2pActivate = MakeRule[{CD[-z][TLambdaPerp2p[-a, -b]],
     DTLambdaPerp2p[-z, -a, -b] + A[i, -a, -z] TLambdaPerp2p[-i, -b] +
      A[i, -b, -z] TLambdaPerp2p[-a, -i]}, MetricOn → All, ContractMetrics → True];
DRLambdaPerp0pActivate = MakeRule[{CD[-z][RLambdaPerp0p[]], DRLambdaPerp0p[-z]},
    MetricOn → All, ContractMetrics → True];
DRLambdaPerp0mActivate = MakeRule[{CD[-z][RLambdaPerp0m[]], DRLambdaPerp0m[-z]},
    MetricOn → All, ContractMetrics → True];
DRLambdaPerp1pActivate = MakeRule[{CD[-z][RLambdaPerp1p[-a, -b]],
```

```
    DRLambdaPerp1p[-z, -a, -b] + A[i, -a, -z] RLambdaPerp1p[-i, -b] +
     A[i, -b, -z] RLambdaPerp1p[-a, -i]}, MetricOn → All, ContractMetrics → True];
DRLambdaPerp1mActivate = MakeRule[{CD[-z][RLambdaPerp1m[-a]],
    DRLambdaPerp1m[-z, -a] + A[i, -a, -z] RLambdaPerp1m[-i]},
   MetricOn → All, ContractMetrics → True];
DRLambdaPerp2pActivate = MakeRule[{CD[-z][RLambdaPerp2p[-a, -b]],
    DRLambdaPerp2p[-z, -a, -b] + A[i, -a, -z] RLambdaPerp2p[-i, -b] +
     A[i, -b, -z] RLambdaPerp2p[-a, -i]}, MetricOn → All, ContractMetrics → True];
DRLambdaPerp2mActivate = MakeRule[{CD[-z][RLambdaPerp2m[-a, -b, -c]],
    DRLambdaPerp2m[-z, -a, -b, -c] + A[i, -a, -z] RLambdaPerp2m[-i, -b, -c] +
     A[i, -b, -z] RLambdaPerp2m[-a, -i, -c] + A[i, -c, -z]
      RLambdaPerp2m[-a, -b, -i]}, MetricOn → All, ContractMetrics → True];
DRLambdaPerpActivate = Join[DTLambdaPerp0pActivate, DTLambdaPerp1pActivate,
   DTLambdaPerp1mActivate, DTLambdaPerp2pActivate, DRLambdaPerp0pActivate,
   DRLambdaPerp0mActivate, DRLambdaPerp1pActivate, DRLambdaPerp1mActivate,
   DRLambdaPerp2pActivate, DRLambdaPerp2mActivate];
ClearBuild[];
```

build

Define $\mathbb{D}H$

build

In[•]:=

```
DefTensor[DHComp[-z], M4, PrintAs → "𝔻H"];
DHCompActivate = MakeRule[
   {CD[-z][HComp[]], DHComp[-z]}, MetricOn → All, ContractMetrics → True];
ClearBuild[];
```

build

Define $\hat{\mathbb{D}}\hat{T}\,J^P$, $\hat{\mathbb{D}}\hat{R}\,J^P$

build

```
DefTensor[DpTP0m[-z], M4,
  PrintAs → SymbolBuild[TPSymb, Spin0m, "Derivative" -> 2], OrthogonalTo → {V[z]}];
DeclareOrder[DpTP0m[-z], 1];
DeclareOrder[DTP0m[-z], 1,
  "approximation" → B[w, -z] DpTP0m[-w] + V[-v] B[v, -z] V[u] H[-u, w] DTP0m[-w]];
DefTensor[DpTP1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TPSymb, Spin1p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpTP1p[-z, -a, -b], 1];
DeclareOrder[DTP1p[-z, -a, -b], 1, "approximation" →
   B[w, -z] DpTP1p[-w, -a, -b] + V[-v] B[v, -z] V[u] H[-u, w] DTP1p[-w, -a, -b]];
DefTensor[DpTP1m[-z, -a], M4, PrintAs → SymbolBuild[TPSymb,
    Spin1m, "Derivative" -> 2], OrthogonalTo → {V[z], V[a]}];
```

```
DeclareOrder[DpTP1m[-z, -a], 1];
DeclareOrder[DTP1m[-z, -a], 1, "approximation" →
    B[w, -z] DpTP1m[-w, -a] + V[-v] B[v, -z] V[u] H[-u, w] DTP1m[-w, -a]];
DefTensor[DpTP2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TPSymb, Spin2m, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b], V[c]}];
DeclareOrder[DpTP2m[-z, -a, -b, -c], 1];
DeclareOrder[DTP2m[-z, -a, -b, -c],
  1, "approximation" → B[w, -z] DpTP2m[-w, -a, -b, -c] +
    V[-v] B[v, -z] V[u] H[-u, w] DTP2m[-w, -a, -b, -c]];
AutomaticRules[DpTP2m, MakeRule[{DpTP2m[-z, a, -b, -a], 0},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpTP2m, MakeRule[{epsilonG[a, b, c, d] DpTP2m[-z, -a, -b, -c], 0},
    MetricOn → All, ContractMetrics → True]];
DefTensor[DpRP0p[-z], M4, PrintAs → SymbolBuild[RPSymb, Spin0p, "Derivative" -> 2],
  OrthogonalTo → {V[z]}];
DeclareOrder[DpRP0p[-z], 1];
DeclareOrder[DRP0p[-z], 1,
  "approximation" → B[w, -z] DpRP0p[-w] + V[-v] B[v, -z] V[u] H[-u, w] DRP0p[-w]];
DefTensor[DpRP0m[-z], M4, PrintAs → SymbolBuild[RPSymb, Spin0m, "Derivative" -> 2],
  OrthogonalTo → {V[z]}];
DeclareOrder[DpRP0m[-z], 1];
DeclareOrder[DRP0m[-z], 1,
  "approximation" → B[w, -z] DpRP0m[-w] + V[-v] B[v, -z] V[u] H[-u, w] DRP0m[-w]];
DefTensor[DpRP1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPSymb, Spin1p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpRP1p[-z, -a, -b], 1];
DeclareOrder[DRP1p[-z, -a, -b], 1, "approximation" →
    B[w, -z] DpRP1p[-w, -a, -b] + V[-v] B[v, -z] V[u] H[-u, w] DRP1p[-w, -a, -b]];
DefTensor[DpRP1m[-z, -a], M4, PrintAs → SymbolBuild[RPSymb,
    Spin1m, "Derivative" -> 2], OrthogonalTo → {V[z], V[a]}];
DeclareOrder[DpRP1m[-z, -a], 1];
DeclareOrder[DRP1m[-z, -a], 1, "approximation" →
    B[w, -z] DpRP1m[-w, -a] + V[-v] B[v, -z] V[u] H[-u, w] DRP1m[-w, -a]];
DefTensor[DpRP2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPSymb, Spin2p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpRP2p[-z, -a, -b], 1];
DeclareOrder[DRP2p[-z, -a, -b], 1, "approximation" →
    B[w, -z] DpRP2p[-w, -a, -b] + V[-v] B[v, -z] V[u] H[-u, w] DRP2p[-w, -a, -b]];
DefTensor[DpRP2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RPSymb, Spin2m, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b], V[c]}];
```

```
DeclareOrder[DpRP2m[-z, -a, -b, -c], 1];
DeclareOrder[DRP2m[-z, -a, -b, -c],
  1, "approximation" → B[w, -z] DpRP2m[-w, -a, -b, -c] +
    V[-v] B[v, -z] V[u] H[-u, w] DRP2m[-w, -a, -b, -c]];
AutomaticRules[DpRP2m, MakeRule[{DpRP2m[-z, a, -b, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpRP2m, MakeRule[{epsilonG[a, b, c, d] DpRP2m[-z, -a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpRP2p, MakeRule[{DpRP2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
DpTP0mActivate = MakeRule[{G3[-y, z] DTP0m[-z], G3[-y, z] B[x, -z] DpTP0m[-x]},
   MetricOn → All, ContractMetrics → True];
DpTP1pActivate = MakeRule[{G3[-y, z] DTP1p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpTP1p[-x, -a, -b] +
       (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j]) G3[-y, z] DTP1p[-z, -i, -j] /.
      PADMActivate]}, MetricOn → All, ContractMetrics → True];
DpTP1mActivate = MakeRule[{G3[-y, z] DTP1m[-z, -a],
    Evaluate[G3[-y, z] B[x, -z] DpTP1m[-x, -a] +
       (G[-a, i] - PPara[-a, i]) G3[-y, z] DTP1m[-z, -i] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpTP2mActivate = MakeRule[{G3[-y, z] DTP2m[-z, -a, -b, -c],
    Evaluate[G3[-y, z] B[x, -z] DpTP2m[-x, -a, -b, -c] +
       (G[-a, i] G[-b, j] G[-c, k] - PPara[-a, i] PPara[-b, j] PPara[-c, k])
        G3[-y, z] DTP2m[-z, -i, -j, -k] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpRP0pActivate = MakeRule[{G3[-y, z] DRP0p[-z], G3[-y, z] B[x, -z] DpRP0p[-x]},
   MetricOn → All, ContractMetrics → True];
DpRP0mActivate = MakeRule[{G3[-y, z] DRP0m[-z], G3[-y, z] B[x, -z] DpRP0m[-x]},
   MetricOn → All, ContractMetrics → True];
DpRP1pActivate = MakeRule[{G3[-y, z] DRP1p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpRP1p[-x, -a, -b] +
       (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j]) G3[-y, z] DRP1p[-z, -i, -j] /.
      PADMActivate]}, MetricOn → All, ContractMetrics → True];
DpRP1mActivate = MakeRule[{G3[-y, z] DRP1m[-z, -a],
    Evaluate[G3[-y, z] B[x, -z] DpRP1m[-x, -a] +
       (G[-a, i] - PPara[-a, i]) G3[-y, z] DRP1m[-z, -i] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpRP2pActivate = MakeRule[{G3[-y, z] DRP2p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpRP2p[-x, -a, -b] +
       (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j]) G3[-y, z] DRP2p[-z, -i, -j] /.
      PADMActivate]}, MetricOn → All, ContractMetrics → True];
DpRP2mActivate = MakeRule[{G3[-y, z] DRP2m[-z, -a, -b, -c],
    Evaluate[G3[-y, z] B[x, -z] DpRP2m[-x, -a, -b, -c] +
```

```
        (G[-a, i] G[-b, j] G[-c, k] - PPara[-a, i] PPara[-b, j] PPara[-c, k])
         G3[-y, z] DRP2m[-z, -i, -j, -k] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True];
DpRPActivate = Join[DpTP0mActivate, DpTP1pActivate, DpTP1mActivate,
    DpTP2mActivate, DpRP0pActivate, DpRP0mActivate, DpRP1pActivate,
    DpRP1mActivate, DpRP2pActivate, DpRP2mActivate];

(*again this should be extended over other derivatives,
multiply the above by PPara[-w,v]H[-v,y]*)
DpTP0mDeactivate =
  MakeRule[{DpTP0m[-w], PPara[-w, v] H[-v, y] G3[-y, z] DTP0m[-z]},
    MetricOn → All, ContractMetrics → True];
DpTP1pDeactivate = MakeRule[{DpTP1p[-w, -a, -b],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DTP1p[-z, -a, -b] -
        PPara[-w, v] H[-v, y] (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
         G3[-y, z] DTP1p[-z, -i, -j] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True];
DpTP1mDeactivate = MakeRule[{DpTP1m[-w, -a],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DTP1m[-z, -a] -
        PPara[-w, v] H[-v, y] (G[-a, i] - PPara[-a, i]) G3[-y, z] DTP1m[-z, -i] /.
       PADMActivate]}, MetricOn → All, ContractMetrics → True];
DpTP2mDeactivate = MakeRule[{DpTP2m[-w, -a, -b, -c], Evaluate[
     PPara[-w, v] H[-v, y] G3[-y, z] DTP2m[-z, -a, -b, -c] - PPara[-w, v] H[-v, y]
         (G[-a, i] G[-b, j] G[-c, k] - PPara[-a, i] PPara[-b, j] PPara[-c, k])
         G3[-y, z] DTP2m[-z, -i, -j, -k] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True];
DpRP0pDeactivate = MakeRule[{DpRP0p[-w], PPara[-w, v] H[-v, y]
     G3[-y, z] DRP0p[-z]}, MetricOn → All, ContractMetrics → True];
DpRP0mDeactivate = MakeRule[{DpRP0m[-w], PPara[-w, v] H[-v, y]
     G3[-y, z] DRP0m[-z]}, MetricOn → All, ContractMetrics → True];
DpRP1pDeactivate = MakeRule[{DpRP1p[-w, -a, -b],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DRP1p[-z, -a, -b] -
        PPara[-w, v] H[-v, y] (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
         G3[-y, z] DRP1p[-z, -i, -j] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True];
DpRP1mDeactivate = MakeRule[{DpRP1m[-w, -a],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DRP1m[-z, -a] -
        PPara[-w, v] H[-v, y] (G[-a, i] - PPara[-a, i]) G3[-y, z] DRP1m[-z, -i] /.
       PADMActivate]}, MetricOn → All, ContractMetrics → True];
DpRP2pDeactivate = MakeRule[{DpRP2p[-w, -a, -b],
    Evaluate[PPara[-w, v] H[-v, y] G3[-y, z] DRP2p[-z, -a, -b] -
        PPara[-w, v] H[-v, y] (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
```

```
        G3[-y, z] DRP2p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpRP2mDeactivate = MakeRule[{DpRP2m[-w, -a, -b, -c], Evaluate[
     PPara[-w, v] H[-v, y] G3[-y, z] DRP2m[-z, -a, -b, -c] - PPara[-w, v] H[-v, y]
        (G[-a, i] G[-b, j] G[-c, k] - PPara[-a, i] PPara[-b, j] PPara[-c, k])
        G3[-y, z] DRP2m[-z, -i, -j, -k] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpRPDeactivate = Join[DpTP0mDeactivate, DpTP1pDeactivate, DpTP1mDeactivate,
   DpTP2mDeactivate, DpRP0pDeactivate, DpRP0mDeactivate, DpRP1pDeactivate,
   DpRP1mDeactivate, DpRP2pDeactivate, DpRP2mDeactivate];
ClearBuild[];
```

build

Define $\hat{\mathbb{D}}\hat{\lambda}J^P$

build

```
DefTensor[DpTLambdaP0m[-z], M4, PrintAs →
   SymbolBuild[TLambdaPSymb, Spin0m, "Derivative" -> 2], OrthogonalTo → {V[z]}];
DeclareOrder[DpTLambdaP0m[-z], 1];
DeclareOrder[DTLambdaP0m[-z], 1, "approximation" →
   B[w, -z] DpTLambdaP0m[-w] + V[-v] B[v, -z] V[u] H[-u, w] DTLambdaP0m[-w]];
DefTensor[DpTLambdaP1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPSymb, Spin1p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpTLambdaP1p[-z, -a, -b], 1];
DeclareOrder[DTLambdaP1p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpTLambdaP1p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DTLambdaP1p[-w, -a, -b]];
DefTensor[DpTLambdaP1m[-z, -a], M4, PrintAs → SymbolBuild[TLambdaPSymb,
    Spin1m, "Derivative" -> 2], OrthogonalTo → {V[z], V[a]}];
DeclareOrder[DpTLambdaP1m[-z, -a], 1];
DeclareOrder[DTLambdaP1m[-z, -a],
  1, "approximation" → B[w, -z] DpTLambdaP1m[-w, -a] +
    V[-v] B[v, -z] V[u] H[-u, w] DTLambdaP1m[-w, -a]];
DefTensor[DpTLambdaP2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPSymb, Spin2m, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b], V[c]}];
DeclareOrder[DpTLambdaP2m[-z, -a, -b, -c], 1];
DeclareOrder[DTLambdaP2m[-z, -a, -b, -c], 1,
  "approximation" → B[w, -z] DpTLambdaP2m[-w, -a, -b, -c] +
    V[-v] B[v, -z] V[u] H[-u, w] DTLambdaP2m[-w, -a, -b, -c]];
AutomaticRules[DpTLambdaP2m, MakeRule[{DpTLambdaP2m[-z, a, -b, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpTLambdaP2m, MakeRule[
```

```
    {epsilonG[a, b, c, d] DpTLambdaP2m[-z, -a, -b, -c], 0},
    MetricOn → All, ContractMetrics → True]];
DefTensor[DpRLambdaP0p[-z], M4, PrintAs →
    SymbolBuild[RLambdaPSymb, Spin0p, "Derivative" -> 2], OrthogonalTo → {V[z]}];
DeclareOrder[DpRLambdaP0p[-z], 1];
DeclareOrder[DRLambdaP0p[-z], 1, "approximation" →
    B[w, -z] DpRLambdaP0p[-w] + V[-v] B[v, -z] V[u] H[-u, w] DRLambdaP0p[-w]];
DefTensor[DpRLambdaP0m[-z], M4, PrintAs →
    SymbolBuild[RLambdaPSymb, Spin0m, "Derivative" -> 2], OrthogonalTo → {V[z]}];
DeclareOrder[DpRLambdaP0m[-z], 1];
DeclareOrder[DRLambdaP0m[-z], 1, "approximation" →
    B[w, -z] DpRLambdaP0m[-w] + V[-v] B[v, -z] V[u] H[-u, w] DRLambdaP0m[-w]];
DefTensor[DpRLambdaP1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPSymb, Spin1p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpRLambdaP1p[-z, -a, -b], 1];
DeclareOrder[DRLambdaP1p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpRLambdaP1p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DRLambdaP1p[-w, -a, -b]];
DefTensor[DpRLambdaP1m[-z, -a], M4, PrintAs → SymbolBuild[RLambdaPSymb,
    Spin1m, "Derivative" -> 2], OrthogonalTo → {V[z], V[a]}];
DeclareOrder[DpRLambdaP1m[-z, -a], 1];
DeclareOrder[DRLambdaP1m[-z, -a],
  1, "approximation" → B[w, -z] DpRLambdaP1m[-w, -a] +
    V[-v] B[v, -z] V[u] H[-u, w] DRLambdaP1m[-w, -a]];
DefTensor[DpRLambdaP2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPSymb, Spin2p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpRLambdaP2p[-z, -a, -b], 1];
DeclareOrder[DRLambdaP2p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpRLambdaP2p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DRLambdaP2p[-w, -a, -b]];
DefTensor[DpRLambdaP2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPSymb, Spin2m, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b], V[c]}];
DeclareOrder[DpRLambdaP2m[-z, -a, -b, -c], 1];
DeclareOrder[DRLambdaP2m[-z, -a, -b, -c], 1,
  "approximation" → B[w, -z] DpRLambdaP2m[-w, -a, -b, -c] +
    V[-v] B[v, -z] V[u] H[-u, w] DRLambdaP2m[-w, -a, -b, -c]];
AutomaticRules[DpRLambdaP2m, MakeRule[{DpRLambdaP2m[-z, a, -b, -a], 0},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpRLambdaP2m, MakeRule[
    {epsilonG[a, b, c, d] DpRLambdaP2m[-z, -a, -b, -c], 0},
    MetricOn → All, ContractMetrics → True]];
```

```
AutomaticRules[DpRLambdaP2p, MakeRule[{DpRLambdaP2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
DpTLambdaP0mActivate = MakeRule[{G3[-y, z] DTLambdaP0m[-z],
    G3[-y, z] B[x, -z] DpTLambdaP0m[-x]}, MetricOn → All, ContractMetrics → True];
DpTLambdaP1pActivate = MakeRule[{G3[-y, z] DTLambdaP1p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpTLambdaP1p[-x, -a, -b] +
       (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
        G3[-y, z] DTLambdaP1p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpTLambdaP1mActivate = MakeRule[{G3[-y, z] DTLambdaP1m[-z, -a],
    Evaluate[G3[-y, z] B[x, -z] DpTLambdaP1m[-x, -a] +
       (G[-a, i] - PPara[-a, i]) G3[-y, z] DTLambdaP1m[-z, -i] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpTLambdaP2mActivate = MakeRule[{G3[-y, z] DTLambdaP2m[-z, -a, -b, -c],
    Evaluate[G3[-y, z] B[x, -z] DpTLambdaP2m[-x, -a, -b, -c] +
       (G[-a, i] G[-b, j] G[-c, k] - PPara[-a, i] PPara[-b, j] PPara[-c, k])
        G3[-y, z] DTLambdaP2m[-z, -i, -j, -k] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpRLambdaP0pActivate = MakeRule[{G3[-y, z] DRLambdaP0p[-z],
    G3[-y, z] B[x, -z] DpRLambdaP0p[-x]}, MetricOn → All, ContractMetrics → True];
DpRLambdaP0mActivate = MakeRule[{G3[-y, z] DRLambdaP0m[-z],
    G3[-y, z] B[x, -z] DpRLambdaP0m[-x]}, MetricOn → All, ContractMetrics → True];
DpRLambdaP1pActivate = MakeRule[{G3[-y, z] DRLambdaP1p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpRLambdaP1p[-x, -a, -b] +
       (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
        G3[-y, z] DRLambdaP1p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpRLambdaP1mActivate = MakeRule[{G3[-y, z] DRLambdaP1m[-z, -a],
    Evaluate[G3[-y, z] B[x, -z] DpRLambdaP1m[-x, -a] +
       (G[-a, i] - PPara[-a, i]) G3[-y, z] DRLambdaP1m[-z, -i] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpRLambdaP2pActivate = MakeRule[{G3[-y, z] DRLambdaP2p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpRLambdaP2p[-x, -a, -b] +
       (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
        G3[-y, z] DRLambdaP2p[-z, -i, -j] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpRLambdaP2mActivate = MakeRule[{G3[-y, z] DRLambdaP2m[-z, -a, -b, -c],
    Evaluate[G3[-y, z] B[x, -z] DpRLambdaP2m[-x, -a, -b, -c] +
       (G[-a, i] G[-b, j] G[-c, k] - PPara[-a, i] PPara[-b, j] PPara[-c, k])
        G3[-y, z] DRLambdaP2m[-z, -i, -j, -k] /. PADMActivate]},
   MetricOn → All, ContractMetrics → True];
DpRLambdaPActivate = Join[DpTLambdaP0mActivate, DpTLambdaP1pActivate,
```

```
    DpTLambdaP1mActivate, DpTLambdaP2mActivate, DpRLambdaP0pActivate,
    DpRLambdaP0mActivate, DpRLambdaP1pActivate, DpRLambdaP1mActivate,
    DpRLambdaP2pActivate, DpRLambdaP2mActivate];
ClearBuild[];
```

build

Define $\hat{\mathbb{D}}\overset{*}{\lambda}J^P$

build

```
DefTensor[DpTLambdaPerp0p[-z], M4,
  PrintAs → SymbolBuild[TLambdaPerpSymb, Spin0p, "Derivative" -> 2],
  OrthogonalTo → {V[z]}];
DeclareOrder[DpTLambdaPerp0p[-z], 1];
DeclareOrder[DTLambdaPerp0p[-z], 1, "approximation" →
   B[w, -z] DpTLambdaPerp0p[-w] + V[-v] B[v, -z] V[u] H[-u, w] DTLambdaPerp0p[-w]];
DefTensor[DpTLambdaPerp1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPerpSymb, Spin1p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpTLambdaPerp1p[-z, -a, -b], 1];
DeclareOrder[DTLambdaPerp1p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpTLambdaPerp1p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DTLambdaPerp1p[-w, -a, -b]];
DefTensor[DpTLambdaPerp1m[-z, -a], M4, PrintAs → SymbolBuild[TLambdaPerpSymb,
    Spin1m, "Derivative" -> 2], OrthogonalTo → {V[z], V[a]}];
DeclareOrder[DpTLambdaPerp1m[-z, -a], 1];
DeclareOrder[DTLambdaPerp1m[-z, -a],
  1, "approximation" → B[w, -z] DpTLambdaPerp1m[-w, -a] +
    V[-v] B[v, -z] V[u] H[-u, w] DTLambdaPerp1m[-w, -a]];
DefTensor[DpTLambdaPerp2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[TLambdaPerpSymb, Spin2p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpTLambdaPerp2p[-z, -a, -b], 1];
DeclareOrder[DTLambdaPerp2p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpTLambdaPerp2p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DTLambdaPerp2p[-w, -a, -b]];
AutomaticRules[DpTLambdaPerp2p, MakeRule[{DpTLambdaPerp2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
DefTensor[DpRLambdaPerp0p[-z], M4, PrintAs → SymbolBuild[
    RLambdaPerpSymb, Spin0p, "Derivative" -> 2], OrthogonalTo → {V[z]}];
DeclareOrder[DpRLambdaPerp0p[-z], 1];
DeclareOrder[DRLambdaPerp0p[-z], 1, "approximation" →
   B[w, -z] DpRLambdaPerp0p[-w] + V[-v] B[v, -z] V[u] H[-u, w] DRLambdaPerp0p[-w]];
DefTensor[DpRLambdaPerp0m[-z], M4, PrintAs → SymbolBuild[
    RLambdaPerpSymb, Spin0m, "Derivative" -> 2], OrthogonalTo → {V[z]}];
```

```
DeclareOrder[DpRLambdaPerp0m[-z], 1];
DeclareOrder[DRLambdaPerp0m[-z], 1, "approximation" →
    B[w, -z] DpRLambdaPerp0m[-w] + V[-v] B[v, -z] V[u] H[-u, w] DRLambdaPerp0m[-w]];
DefTensor[DpRLambdaPerp1p[-z, -a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin1p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpRLambdaPerp1p[-z, -a, -b], 1];
DeclareOrder[DRLambdaPerp1p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpRLambdaPerp1p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DRLambdaPerp1p[-w, -a, -b]];
DefTensor[DpRLambdaPerp1m[-z, -a], M4, PrintAs → SymbolBuild[RLambdaPerpSymb,
    Spin1m, "Derivative" -> 2], OrthogonalTo → {V[z], V[a]}];
DeclareOrder[DpRLambdaPerp1m[-z, -a], 1];
DeclareOrder[DRLambdaPerp1m[-z, -a],
  1, "approximation" → B[w, -z] DpRLambdaPerp1m[-w, -a] +
    V[-v] B[v, -z] V[u] H[-u, w] DRLambdaPerp1m[-w, -a]];
DefTensor[DpRLambdaPerp2p[-z, -a, -b], M4, Symmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin2p, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b]}];
DeclareOrder[DpRLambdaPerp2p[-z, -a, -b], 1];
DeclareOrder[DRLambdaPerp2p[-z, -a, -b],
  1, "approximation" → B[w, -z] DpRLambdaPerp2p[-w, -a, -b] +
    V[-v] B[v, -z] V[u] H[-u, w] DRLambdaPerp2p[-w, -a, -b]];
DefTensor[DpRLambdaPerp2m[-z, -a, -b, -c], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RLambdaPerpSymb, Spin2m, "Derivative" -> 2],
  OrthogonalTo → {V[z], V[a], V[b], V[c]}];
DeclareOrder[DpRLambdaPerp2m[-z, -a, -b, -c], 1];
DeclareOrder[DRLambdaPerp2m[-z, -a, -b, -c], 1,
  "approximation" → B[w, -z] DpRLambdaPerp2m[-w, -a, -b, -c] +
    V[-v] B[v, -z] V[u] H[-u, w] DRLambdaPerp2m[-w, -a, -b, -c]];
AutomaticRules[DpRLambdaPerp2m, MakeRule[{DpRLambdaPerp2m[-z, a, -b, -a], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpRLambdaPerp2m, MakeRule[
   {epsilonG[a, b, c, d] DpRLambdaPerp2m[-z, -a, -b, -c], 0},
   MetricOn → All, ContractMetrics → True]];
AutomaticRules[DpRLambdaPerp2p, MakeRule[{DpRLambdaPerp2p[-z, a, -a], 0},
   MetricOn → All, ContractMetrics → True]];
DpTLambdaPerp0pActivate = MakeRule[{G3[-y, z] DTLambdaPerp0p[-z], G3[-y, z]
     B[x, -z] DpTLambdaPerp0p[-x]}, MetricOn → All, ContractMetrics → True];
DpTLambdaPerp1pActivate = MakeRule[{G3[-y, z] DTLambdaPerp1p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpTLambdaPerp1p[-x, -a, -b] +
       (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
        G3[-y, z] DTLambdaPerp1p[-z, -i, -j] /. PADMActivate]},
```

```
    MetricOn → All, ContractMetrics → True];
DpTLambdaPerp1mActivate = MakeRule[{G3[-y, z] DTLambdaPerp1m[-z, -a],
    Evaluate[G3[-y, z] B[x, -z] DpTLambdaPerp1m[-x, -a] +
        (G[-a, i] - PPara[-a, i]) G3[-y, z] DTLambdaPerp1m[-z, -i] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True];
DpTLambdaPerp2pActivate = MakeRule[{G3[-y, z] DTLambdaPerp2p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpTLambdaPerp2p[-x, -a, -b] +
        (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
         G3[-y, z] DTLambdaPerp2p[-z, -i, -j] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True];
DpRLambdaPerp0pActivate = MakeRule[{G3[-y, z] DRLambdaPerp0p[-z], G3[-y, z]
     B[x, -z] DpRLambdaPerp0p[-x]}, MetricOn → All, ContractMetrics → True];
DpRLambdaPerp0mActivate = MakeRule[{G3[-y, z] DRLambdaPerp0m[-z], G3[-y, z]
     B[x, -z] DpRLambdaPerp0m[-x]}, MetricOn → All, ContractMetrics → True];
DpRLambdaPerp1pActivate = MakeRule[{G3[-y, z] DRLambdaPerp1p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpRLambdaPerp1p[-x, -a, -b] +
        (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
         G3[-y, z] DRLambdaPerp1p[-z, -i, -j] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True];
DpRLambdaPerp1mActivate = MakeRule[{G3[-y, z] DRLambdaPerp1m[-z, -a],
    Evaluate[G3[-y, z] B[x, -z] DpRLambdaPerp1m[-x, -a] +
        (G[-a, i] - PPara[-a, i]) G3[-y, z] DRLambdaPerp1m[-z, -i] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True];
DpRLambdaPerp2pActivate = MakeRule[{G3[-y, z] DRLambdaPerp2p[-z, -a, -b],
    Evaluate[G3[-y, z] B[x, -z] DpRLambdaPerp2p[-x, -a, -b] +
        (G[-a, i] G[-b, j] - PPara[-a, i] PPara[-b, j])
         G3[-y, z] DRLambdaPerp2p[-z, -i, -j] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True];
DpRLambdaPerp2mActivate = MakeRule[{G3[-y, z] DRLambdaPerp2m[-z, -a, -b, -c],
    Evaluate[G3[-y, z] B[x, -z] DpRLambdaPerp2m[-x, -a, -b, -c] +
        (G[-a, i] G[-b, j] G[-c, k] - PPara[-a, i] PPara[-b, j] PPara[-c, k])
         G3[-y, z] DRLambdaPerp2m[-z, -i, -j, -k] /. PADMActivate]},
    MetricOn → All, ContractMetrics → True];
DpRLambdaPerpActivate = Join[DpTLambdaPerp0pActivate, DpTLambdaPerp1pActivate,
    DpTLambdaPerp1mActivate, DpTLambdaPerp2pActivate, DpRLambdaPerp0pActivate,
    DpRLambdaPerp0mActivate, DpRLambdaPerp1pActivate, DpRLambdaPerp1mActivate,
    DpRLambdaPerp2pActivate, DpRLambdaPerp2mActivate];
ClearBuild[];
```

build

Define $\hat{D}$H

build

```
UnknownSymb = "*";
DefTensor[DpHComp[-z], M4,
  PrintAs → SymbolBuild[UnknownSymb, "Derivative" -> 2], OrthogonalTo → {V[z]}];
DpHCompActivate = MakeRule[{G3[-y, z] DHComp[-z], G3[-y, z] B[x, -z] DpHComp[-x]},
   MetricOn → All, ContractMetrics → True];

DGrandActivate = Join[DPiPActivate, DRPActivate,
   DRLambdaPActivate, DRLambdaPerpActivate, DHCompActivate];
DpGrandActivate = Join[DpPiPActivate, DpRPActivate, DpRLambdaPActivate,
   DpRLambdaPerpActivate, DpJActivate, DpVActivate, DpHCompActivate];
ClearBuild[];
```

build

Define Hamiltonian

build

In[•]:=

```
SuperHamiltonianSymb = "\[ScriptCapitalH]_b";
DefTensor[SuperHamiltonian0p[], M4,
  PrintAs → SymbolBuild[SuperHamiltonianSymb, Spin0p]];
DeclareOrder[SuperHamiltonian0p[], 1];
LinearSuperMomentumSymb = "\[ScriptCapitalH]_b";
DefTensor[LinearSuperMomentum1m[-a], M4,
  PrintAs → SymbolBuild[LinearSuperMomentumSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[LinearSuperMomentum1m[-a], 1];
RotationalSuperMomentumSymb = "\[ScriptCapitalH]_\[CapitalR]";
DefTensor[RotationalSuperMomentum1m[-a], M4, PrintAs →
   SymbolBuild[RotationalSuperMomentumSymb, Spin1m], OrthogonalTo → {V[a]}];
DeclareOrder[RotationalSuperMomentum1m[-a], 1];
DefTensor[RotationalSuperMomentum1p[-a, -b], M4, Antisymmetric[{-a, -b}],
  PrintAs → SymbolBuild[RotationalSuperMomentumSymb, Spin1p],
  OrthogonalTo → {V[a], V[b]}];
DeclareOrder[RotationalSuperMomentum1p[-a, -b], 1];
ClearBuild[];
```

build

Shell rules

build

Constraint Structure

build

```
(*Here are the generalised freedom coefficients*)
```

```
DefNiceConstantSymbol[ShellPara, ToExpression[#]] & /@ ASectorNames;
DefNiceConstantSymbol[ShellOrig, ToExpression[#]] & /@ ASectorNames;
DefNiceConstantSymbol[ShellPerp, ToExpression[#]] & /@ ASectorNames;
DefNiceConstantSymbol[ShellSing, ToExpression[#]] & /@ ASectorNames;
DefNiceConstantSymbol[ShellPrim, ToExpression[#]] & /@ ASectorNames;
DefNiceConstantSymbol[ShellPara, ToExpression[#]] & /@ BSectorNames;
DefNiceConstantSymbol[ShellOrig, ToExpression[#]] & /@ BSectorNames;
DefNiceConstantSymbol[ShellPerp, ToExpression[#]] & /@ BSectorNames;
DefNiceConstantSymbol[ShellSing, ToExpression[#]] & /@ BSectorNames;
DefNiceConstantSymbol[ShellPrim, ToExpression[#]] & /@ BSectorNames;

ComputeShellFreedoms[$ToTheory_, $Theory_] :=
  Module[{KeepOnlyObviousZeros, cAlpPerpPerpTheory, cAlpPerpParaTheory,
    cAlpParaPerpTheory, cAlpParaParaTheory, cAlpDetTheory, AlpPerpPerpTheory,
    AlpPerpParaTheory, AlpParaPerpTheory, AlpParaParaTheory, AlpDetTheory,
    cBetPerpPerpTheory, cBetPerpParaTheory, cBetParaPerpTheory,
    cBetParaParaTheory, cBetDetTheory, BetPerpPerpTheory, BetPerpParaTheory,
    BetParaPerpTheory, BetParaParaTheory, BetDetTheory},
   (*a message*)
   xAct`xTensor`Private`MakeDefInfo[
    DefTheory, $Theory, {"$ToShellFreedoms for the theory", ""}];

   (*We don't want our theory-defining rules to have unintended side-
      effects... so we only keep zeros which pop out of the initial rules.*)
   KeepOnlyObviousZeros[q_] := If[q == 0, 0, 1, 1];

   (*We fix $ToOrderRules according to whether there is an Einstein--Hilbert
    term, recalling that this can change the order of certain constraints*)
   $ToOrderRules = {};
   Switch[KeepOnlyObviousZeros@(Alp0 /. $ToTheory), 0,
    $ToOrderRules = $ToNormalOrderRules, 1, $ToOrderRules = $ToEHOrderRules];

   (*We impose the theory on the coefficients*)
   cAlpPerpPerpTheory =
    KeepOnlyObviousZeros /@ (cAlpPerpPerp /. TocAlp /. $ToTheory);
   cAlpPerpParaTheory = KeepOnlyObviousZeros /@
     (cAlpPerpPara /. TocAlp /. $ToTheory);
   cAlpParaPerpTheory = KeepOnlyObviousZeros /@
     (cAlpParaPerp /. TocAlp /. $ToTheory);
   cAlpParaParaTheory = KeepOnlyObviousZeros /@
     (cAlpParaPara /. TocAlp /. $ToTheory);
   cAlpDetTheory = KeepOnlyObviousZeros /@
     (cAlpDeterminants /. TocAlp /. $ToTheory);
```

```
AlpPerpPerpTheory = KeepOnlyObviousZeros /@
  (AlpPerpPerp /. ToAlp /. $ToTheory);
AlpPerpParaTheory = KeepOnlyObviousZeros /@
  (AlpPerpPara /. ToAlp /. $ToTheory);
AlpParaPerpTheory = KeepOnlyObviousZeros /@
  (AlpParaPerp /. ToAlp /. $ToTheory);
AlpParaParaTheory = KeepOnlyObviousZeros /@
  (AlpParaPara /. ToAlp /. $ToTheory);
AlpDetTheory = KeepOnlyObviousZeros /@ (AlpDeterminants /. ToAlp /. $ToTheory);
cBetPerpPerpTheory =
 KeepOnlyObviousZeros /@ (cBetPerpPerp /. TocBet /. $ToTheory);
cBetPerpParaTheory = KeepOnlyObviousZeros /@
  (cBetPerpPara /. TocBet /. $ToTheory);
cBetParaPerpTheory = KeepOnlyObviousZeros /@
  (cBetParaPerp /. TocBet /. $ToTheory);
cBetParaParaTheory = KeepOnlyObviousZeros /@
  (cBetParaPara /. TocBet /. $ToTheory);
cBetDetTheory = KeepOnlyObviousZeros /@
  (cBetDeterminants /. TocBet /. $ToTheory);
BetPerpPerpTheory = KeepOnlyObviousZeros /@
  (BetPerpPerp /. ToBet /. $ToTheory);
BetPerpParaTheory = KeepOnlyObviousZeros /@
  (BetPerpPara /. ToBet /. $ToTheory);
BetParaPerpTheory = KeepOnlyObviousZeros /@
  (BetParaPerp /. ToBet /. $ToTheory);
BetParaParaTheory = KeepOnlyObviousZeros /@
  (BetParaPara /. ToBet /. $ToTheory);
BetDetTheory = KeepOnlyObviousZeros /@ (BetDeterminants /. ToBet /. $ToTheory);

(*We construct the rule which sends the freedom coefficients to the shell*)
$ToShellFreedoms = {};

(*These versions seem not to be correct*)
(*
For[ii=1,ii<7,ii++,
 If[cAlpPerpPerpTheory[[ii]]cAlpPerpParaTheory[[ii]]
    cAlpParaPerpTheory[[ii]]cAlpParaParaTheory[[ii]]==0,
  {AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
      "ShellPara"<>ToString[ASectorNames[[ii]]]<>"->1"]]],
   AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
      "ShellPerp"<>ToString[ASectorNames[[ii]]]<>"->1"]]],
   AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
      "ShellSing"<>ToString[ASectorNames[[ii]]]<>"->1"]]],
```

```
    If[AlpPerpPerpTheory[[ii]]==0,
     AppendTo[$ToShellFreedoms,Evaluate[
       ToExpression["ShellOrig"<>ToString[ASectorNames[[ii]]]<>"->0"]]],
     AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
        "ShellOrig"<>ToString[ASectorNames[[ii]]]<>"->1"]]]],
   If[cAlpDetTheory[[ii]]==0,
    {AppendTo[$ToShellFreedoms,Evaluate[
       ToExpression["ShellPara"<>ToString[ASectorNames[[ii]]]<>"->1"]]],
     AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
        "ShellSing"<>ToString[ASectorNames[[ii]]]<>"->0"]]],
     If[AlpPerpPerpTheory[[ii]]==0,
      {AppendTo[$ToShellFreedoms,Evaluate[
         ToExpression["ShellOrig"<>ToString[ASectorNames[[ii]]]<>"->0"]]],
       AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellPerp"<>ToString[ASectorNames[[ii]]]<>"->1"]]]},
      {AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellPerp"<>ToString[ASectorNames[[ii]]]<>"->0"]]],
       AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellOrig"<>ToString[ASectorNames[[ii]]]<>"->1"]]]}]},
    {AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
        "ShellPara"<>ToString[ASectorNames[[ii]]]<>"->0"]]],
     AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
        "ShellPerp"<>ToString[ASectorNames[[ii]]]<>"->1"]]],
     If[AlpPerpPerpTheory[[ii]]==0,
      {AppendTo[$ToShellFreedoms,Evaluate[
         ToExpression["ShellOrig"<>ToString[ASectorNames[[ii]]]<>"->0"]]],
       AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellSing"<>ToString[ASectorNames[[ii]]]<>"->1"]]]},
      {AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellSing"<>ToString[ASectorNames[[ii]]]<>"->0"]]],
       AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellOrig"<>ToString[ASectorNames[[ii]]]<>"->1"]]]}]}]]];

 For[ii=1,ii<7,ii++,
  If[cBetPerpPerpTheory[[ii]]cBetPerpParaTheory[[ii]]
     cBetParaPerpTheory[[ii]]cBetParaParaTheory[[ii]]==0,
   {AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
       "ShellPara"<>ToString[BSectorNames[[ii]]]<>"->1"]]],
    AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
       "ShellPerp"<>ToString[BSectorNames[[ii]]]<>"->1"]]],
    AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
       "ShellSing"<>ToString[BSectorNames[[ii]]]<>"->1"]]],
    If[BetPerpPerpTheory[[ii]]==0,
     AppendTo[$ToShellFreedoms,Evaluate[
```

```
        ToExpression["ShellOrig"<>ToString[BSectorNames[[ii]]]<>"->0"]]],
     AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
        "ShellOrig"<>ToString[BSectorNames[[ii]]]<>"->1"]]]],
   If[cBetDetTheory[[ii]]==0,
    {AppendTo[$ToShellFreedoms,Evaluate[
       ToExpression["ShellPara"<>ToString[BSectorNames[[ii]]]<>"->1"]]],
     AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
        "ShellSing"<>ToString[BSectorNames[[ii]]]<>"->0"]]],
     If[BetPerpPerpTheory[[ii]]==0,
      {AppendTo[$ToShellFreedoms,Evaluate[
         ToExpression["ShellOrig"<>ToString[BSectorNames[[ii]]]<>"->0"]]],
       AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellPerp"<>ToString[BSectorNames[[ii]]]<>"->1"]]]},
      {AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellPerp"<>ToString[BSectorNames[[ii]]]<>"->0"]]],
       AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellOrig"<>ToString[BSectorNames[[ii]]]<>"->1"]]]}]},
    {AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
        "ShellPara"<>ToString[BSectorNames[[ii]]]<>"->0"]]],
     AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
        "ShellPerp"<>ToString[BSectorNames[[ii]]]<>"->1"]]],
     If[BetPerpPerpTheory[[ii]]==0,
      {AppendTo[$ToShellFreedoms,Evaluate[
         ToExpression["ShellOrig"<>ToString[BSectorNames[[ii]]]<>"->0"]]],
       AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellSing"<>ToString[BSectorNames[[ii]]]<>"->1"]]]},
      {AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellSing"<>ToString[BSectorNames[[ii]]]<>"->0"]]],
       AppendTo[$ToShellFreedoms,Evaluate[ToExpression[
          "ShellOrig"<>ToString[BSectorNames[[ii]]]<>"->1"]]]}]}]]];
*)

For[ii = 1, ii < 7, ii++,
 If[cAlpPerpPerpTheory[[ii]] cAlpPerpParaTheory[[ii]]
     cAlpParaPerpTheory[[ii]] cAlpParaParaTheory[[ii]] == 0,
   {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
       "ShellPara" <> ToString[ASectorNames[[ii]]] <> "->1"]]],
    AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
       "ShellPerp" <> ToString[ASectorNames[[ii]]] <> "->1"]]],
    AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
       "ShellSing" <> ToString[ASectorNames[[ii]]] <> "->1"]]],
    If[AlpPerpPerpTheory[[ii]] == 0,
     AppendTo[$ToShellFreedoms, Evaluate[
       ToExpression["ShellOrig" <> ToString[ASectorNames[[ii]]] <> "->0"]]],
```

```
      AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
         "ShellOrig" <> ToString[ASectorNames[[ii]]] <> "->1"]]]],
    If[cAlpDetTheory[[ii]] == 0,
      If[AlpPerpPerpTheory[[ii]] == 0,
        {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellOrig" <> ToString[ASectorNames[[ii]]] <> "->0"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellPara" <> ToString[ASectorNames[[ii]]] <> "->1"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellPerp" <> ToString[ASectorNames[[ii]]] <> "->1"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellSing" <> ToString[ASectorNames[[ii]]] <> "->1"]]]},
        {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellOrig" <> ToString[ASectorNames[[ii]]] <> "->1"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellPara" <> ToString[ASectorNames[[ii]]] <> "->1"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellPerp" <> ToString[ASectorNames[[ii]]] <> "->1"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellSing" <> ToString[ASectorNames[[ii]]] <> "->0"]]]}
       ];,
      If[AlpPerpPerpTheory[[ii]] == 0,
        {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellOrig" <> ToString[ASectorNames[[ii]]] <> "->0"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellPara" <> ToString[ASectorNames[[ii]]] <> "->0"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellPerp" <> ToString[ASectorNames[[ii]]] <> "->1"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellSing" <> ToString[ASectorNames[[ii]]] <> "->1"]]]},
        {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellOrig" <> ToString[ASectorNames[[ii]]] <> "->1"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellPara" <> ToString[ASectorNames[[ii]]] <> "->0"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellPerp" <> ToString[ASectorNames[[ii]]] <> "->0"]]],
         AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
            "ShellSing" <> ToString[ASectorNames[[ii]]] <> "->1"]]]}
       ];
     ];
  ];
];

For[ii = 1, ii < 7, ii++,
```

```
If[cBetPerpPerpTheory[[ii]] cBetPerpParaTheory[[ii]]
    cBetParaPerpTheory[[ii]] cBetParaParaTheory[[ii]] == 0,
  If[ii == 2 || ii == 6,
    If[! (cBetParaParaTheory[[ii]] == 0),
      {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
          "ShellPara" <> ToString[BSectorNames[[ii]]] <> "->0"]]],
       AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
          "ShellPerp" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
       AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
          "ShellSing" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
       If[BetPerpPerpTheory[[ii]] == 0,
        AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
           "ShellOrig" <> ToString[BSectorNames[[ii]]] <> "->0"]]],
        AppendTo[$ToShellFreedoms, Evaluate[ToExpression["ShellOrig" <>
            ToString[BSectorNames[[ii]]] <> "->1"]]]]},
      {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
          "ShellPara" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
       AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
          "ShellPerp" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
       AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
          "ShellSing" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
       If[BetPerpPerpTheory[[ii]] == 0,
        AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
           "ShellOrig" <> ToString[BSectorNames[[ii]]] <> "->0"]]],
        AppendTo[$ToShellFreedoms, Evaluate[ToExpression["ShellOrig" <>
            ToString[BSectorNames[[ii]]] <> "->1"]]]]};,
    {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
        "ShellPara" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
     AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
        "ShellPerp" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
     AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
        "ShellSing" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
     If[BetPerpPerpTheory[[ii]] == 0,
      AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
         "ShellOrig" <> ToString[BSectorNames[[ii]]] <> "->0"]]],
      AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
         "ShellOrig" <> ToString[BSectorNames[[ii]]] <> "->1"]]]]};,
  If[cBetDetTheory[[ii]] == 0,
    If[BetPerpPerpTheory[[ii]] == 0,
      {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
          "ShellOrig" <> ToString[BSectorNames[[ii]]] <> "->0"]]],
       AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
          "ShellPara" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
```

```
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellPerp" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellSing" <> ToString[BSectorNames[[ii]]] <> "->1"]]]},
          {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellOrig" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellPara" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellPerp" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellSing" <> ToString[BSectorNames[[ii]]] <> "->0"]]]}
         ];,
        If[BetPerpPerpTheory[[ii]] == 0,
          {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellOrig" <> ToString[BSectorNames[[ii]]] <> "->0"]]],
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellPara" <> ToString[BSectorNames[[ii]]] <> "->0"]]],
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellPerp" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellSing" <> ToString[BSectorNames[[ii]]] <> "->1"]]]},
          {AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellOrig" <> ToString[BSectorNames[[ii]]] <> "->1"]]],
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellPara" <> ToString[BSectorNames[[ii]]] <> "->0"]]],
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellPerp" <> ToString[BSectorNames[[ii]]] <> "->0"]]],
            AppendTo[$ToShellFreedoms, Evaluate[ToExpression[
                "ShellSing" <> ToString[BSectorNames[[ii]]] <> "->1"]]]}
         ];
        ];
      ];
    ];


  ];

ToOrderCanonical[expr_, order_] :=
  "ToOrderCanonical"~TimeWrapper~Module[{res, printer},
    printer = PrintTemporary[" ** ToOrderCanonical: order ", order, "..."];
    res = expr;
    Switch[order, 0, {
```

```
      res = res /. $ToOrderRules;
      res = CollectConstants[res, Prt];
      res = res /. {Prt → 0};
     }, 1, {
      res = res /. $ToOrderRules;
      res = CollectConstants[res, Prt];
      res = res /.
        {Prt^2 → 0, Prt^3 → 0, Prt^4 → 0, Prt^5 → 0, Prt^6 → 0, Prt^7 → 0, Prt^8 → 0,
         Prt^9 → 0, Prt^10 → 0, Prt^11 → 0, Prt^12 → 0, Prt^13 → 0, Prt^14 → 0};
      res = res /. {Prt → 1};
     }, Infinity, {}];
    res = res // ToNewCanonical;
    NotebookDelete[printer];
    res];
ClearBuild[];
```

build

Calculate $\hat{T}, \hat{R}$ shell

build

In[ ]:=

```
DefTensor[RPShellPara[-a, -b, -c, -d], M4,
  {Antisymmetric[{-a, -b}], Antisymmetric[{-c, -d}]},
  OrthogonalTo → {V[a], V[b], V[c], V[d]}];
DefTensor[RPShellPerp[-a, -b, -c], M4, Antisymmetric[{-b, -c}],
  OrthogonalTo → {V[a], V[b], V[c]}];

DefFieldStrengthShell[$ToShellFreedoms_, $Theory_] :=
  Module[{TPShellDefinition, RPShellParaDefinition, RPShellPerpDefinition,
    RPShellDefinition, RPShellParaActivate, RPShellPerpActivate,
    RPShellParaPerpActivate, TPShellActivate, RPShellActivate},
   (*a message*)
   xAct`xTensor`Private`MakeDefInfo[DefTheory,
    $Theory, {"$StrengthPShellToStrengthPO3 for the theory", ""}];

   TPShellDefinition = ShellParaB1p V[-a] TP1p[-b, -c] +
         -(1/6) ShellParaB0m PT0m[-a, -b, -c] TP0m[] +
          ShellParaB1m Antisymmetrize[-PPara[-a, -b] TP1m[-c], {-b, -c}] +
          (4/3) ShellParaB2m TP2m[-b, -c, -a] /. $ToShellFreedoms /.
       PO3TActivate /. PADMActivate // ToCanonical;

   RPShellParaDefinition = -(1/6) ShellParaA0p
      (PPara[-a, -d] PPara[-b, -c] - PPara[-a, -c] PPara[-b, -d]) RP0p[] -
     ShellParaA1p (PPara[-b, -d] RP1p[-a, -c] - PPara[-b, -c] RP1p[-a, -d] -
        PPara[-a, -d] RP1p[-b, -c] + PPara[-a, -c] RP1p[-b, -d]) +
```

```
    ShellParaA2p (PPara[-b, -d] RP2p[-a, -c] - PPara[-b, -c] RP2p[-a, -d] -
       PPara[-a, -d] RP2p[-b, -c] + PPara[-a, -c] RP2p[-b, -d]);
  RPShellPerpDefinition = -(1/6) ShellParaA0m PR0m[-a, -b, -c] RP0m[] +
    ShellParaA1m Antisymmetrize[-PPara[-a, -b] RP1m[-c], {-b, -c}] +
    (4/3) ShellParaA2m RP2m[-b, -c, -a];

  RPShellParaActivate = MakeRule[{RPShellPara[-a, -b, -c, -d],
     Evaluate[RPShellParaDefinition]}, MetricOn → All, ContractMetrics → True];
  RPShellPerpActivate = MakeRule[{RPShellPerp[-a, -b, -c],
     Evaluate[RPShellPerpDefinition]}, MetricOn → All, ContractMetrics → True];
  RPShellParaPerpActivate = Join[RPShellParaActivate, RPShellPerpActivate];

  RPShellDefinition =
   RPShellPara[-a, -b, -c, -d] + 2 Antisymmetrize[V[-a] RPShellPerp[-b, -c, -d],
         {-a, -b}] /. RPShellParaPerpActivate /.
     $ToShellFreedoms /. PO3RActivate /. PADMActivate // ToCanonical;

  TPShellDefinition =
   TPShellDefinition // CollectTensors // ScreenDollarIndices // CollectTensors;
  RPShellDefinition = RPShellDefinition // CollectTensors //
     ScreenDollarIndices // CollectTensors;

  TPShellActivate = MakeRule[{TP[-a, -b, -c], Evaluate[TPShellDefinition]},
    MetricOn → All, ContractMetrics → True];
  RPShellActivate = MakeRule[{RP[-a, -b, -c, -d], Evaluate[RPShellDefinition]},
    MetricOn → All, ContractMetrics → True];
  $StrengthPShellToStrengthPO3 = Join[TPShellActivate, RPShellActivate];
 ];
ClearBuild[];
```

build

Calculate $\hat{\pi}\, bJ^P$, $\hat{\pi}\, AJ^P$ shell

build

In[ ]:=

```
DefTensor[PerpBComplement[-i, -k], M4];
DefTensor[OrigBComplement[-i, -k], M4];
DefTensor[SingBComplement[-i, -k], M4];
DefTensor[PerpAComplement[-i, -j, -k], M4, Antisymmetric[{-i, -j}]];
DefTensor[OrigAComplement[-i, -j, -k], M4, Antisymmetric[{-i, -j}]];
DefTensor[SingAComplement[-i, -k], M4, Antisymmetric[{-i, -j}]];
ClearBuild[];
```

GeneralComplements oT ggle

```
IfBuild["GeneralComplementsToggle",
```

```
HiGGSPrint["OrigBComplementDefinition..."];
OrigBComplementDefinition =
 Evaluate[J[] 4 V[g] B[-k, -o] G3[o, -z] H[h, z] (Bet1 PT1[-i, -g, -h, a, c, d] +
            Bet2 PT2[-i, -g, -h, a, c, d] + Bet3 PT3[-i, -g, -h, a, c, d])
          PPara[-c, x] PPara[-d, y] T[-a, -x, -y] + 2 J[] V[g] B[-k, -o]
          G3[o, -z] H[h, z] (cBet1 PT1[-i, -g, -h, a, c, d] +
            cBet2 PT2[-i, -g, -h, a, c, d] + cBet3 PT3[-i, -g, -h, a, c, d])
          PPara[-c, m] PPara[-d, n] TLambda[-a, -m, -n] +
         2 J[] V[g] B[-k, -o] G3[o, -z] H[h, z] (cBet1 PT1[-i, -g, -h, a, c, d] +
            cBet2 PT2[-i, -g, -h, a, c, d] + cBet3 PT3[-i, -g, -h, a, c, d])
          (PPerp[-c, m] PPara[-d, n] TLambda[-a, -m, -n] +
            PPara[-c, m] PPerp[-d, n] TLambda[-a, -m, -n]) /. PActivate /.
       PADMActivate // ToCanonical // ContractMetric // CollectTensors];
HiGGSPrint["PerpBComplementDefinition..."];
PerpBComplementDefinition =
 Evaluate[2 J[] V[g] B[-k, -o] G3[o, -z] H[h, z] (cBet1 PT1[-i, -g, -h, a, c, d] +
            cBet2 PT2[-i, -g, -h, a, c, d] + cBet3 PT3[-i, -g, -h, a, c, d])
          PPara[-c, m] PPara[-d, n] TLambda[-a, -m, -n] +
         2 J[] V[g] B[-k, -o] G3[o, -z] H[h, z] (cBet1 PT1[-i, -g, -h, a, c, d] +
            cBet2 PT2[-i, -g, -h, a, c, d] + cBet3 PT3[-i, -g, -h, a, c, d])
          (PPerp[-c, m] PPara[-d, n] TLambda[-a, -m, -n] +
            PPara[-c, m] PPerp[-d, n] TLambda[-a, -m, -n]) /. PActivate /.
       PADMActivate // ToCanonical // ContractMetric // CollectTensors];
HiGGSPrint["SingBComplementDefinition..."];
SingBComplementDefinition =
 Evaluate[-J[] 4 V[g] B[-k, -o] G3[o, -z] H[h, z] (cBet1 PT1[-i, -g, -h, a, c, d] +
            cBet2 PT2[-i, -g, -h, a, c, d] + cBet3 PT3[-i, -g, -h, a, c, d])
          PPara[-c, x] PPara[-d, y] T[-a, -x, -y] /. PActivate /. PADMActivate //
      ToCanonical // ContractMetric // CollectTensors];
HiGGSPrint["OrigAComplementDefinition..."];
OrigAComplementDefinition =
 Evaluate[-2 Alp0 J[] Antisymmetrize[V[-i] PPara[-j, -k], {-i, -j}] +
         J[] 8 V[g] B[-k, -o] G3[o, -z] H[h, z]
          (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] + Alp2 PR2[-i, -j, -g, -h,
              a, b, c, d] + Alp3 PR3[-i, -j, -g, -h, a, b, c, d] + Alp4
             PR4[-i, -j, -g, -h, a, b, c, d] + Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
            Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) PPara[-c, x] PPara[-d, y]
          R[-a, -b, -x, -y] + 4 J[] V[g] B[-k, -o] G3[o, -z] H[h, z]
          (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] + cAlp2 PR2[-i, -j, -g,
              -h, a, b, c, d] + cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] +
            cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] + cAlp5 PR5[-i, -j, -g,
              -h, a, b, c, d] + cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
          PPara[-c, m] PPara[-d, n] RLambda[-a, -b, -m, -n] +
```

```
        4 J[] V[g] B[-k, -o] G3[o, -z] H[h, z] (cAlp1 PR1[-i, -j, -g, -h, a, b,
             c, d] + cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] + cAlp3 PR3[-i, -j, -g,
             -h, a, b, c, d] + cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] + cAlp5 PR5[
             -i, -j, -g, -h, a, b, c, d] + cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
         (PPerp[-c, m] PPara[-d, n] RLambda[-a, -b, -m, -n] +
           PPara[-c, m] PPerp[-d, n] RLambda[-a, -b, -m, -n]) /. PActivate /.
      PADMActivate // ToCanonical // ContractMetric // CollectTensors];
HiGGSPrint["PerpAComplementDefinition..."];
PerpAComplementDefinition = Evaluate[
  4 J[] V[g] B[-k, -o] G3[o, -z] H[h, z] (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
          cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] + cAlp3 PR3[-i, -j, -g, -h, a,
            b, c, d] + cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] + cAlp5 PR5[-i,
            -j, -g, -h, a, b, c, d] + cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
         PPara[-c, m] PPara[-d, n] RLambda[-a, -b, -m, -n] +
        4 J[] V[g] B[-k, -o] G3[o, -z] H[h, z] (cAlp1 PR1[-i, -j, -g, -h, a, b,
             c, d] + cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] + cAlp3 PR3[-i, -j, -g,
             -h, a, b, c, d] + cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] + cAlp5 PR5[
             -i, -j, -g, -h, a, b, c, d] + cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
         (PPerp[-c, m] PPara[-d, n] RLambda[-a, -b, -m, -n] +
           PPara[-c, m] PPerp[-d, n] RLambda[-a, -b, -m, -n]) /. PActivate /.
      PADMActivate // ToCanonical // ContractMetric // CollectTensors];
HiGGSPrint["SingAComplementDefinition..."];
SingAComplementDefinition = Evaluate[
  -J[] 8 V[g] B[-k, -o] G3[o, -z] H[h, z] (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
         cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] + cAlp3 PR3[-i, -j, -g,
           -h, a, b, c, d] + cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] + cAlp5
          PR5[-i, -j, -g, -h, a, b, c, d] + cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
        PPara[-c, x] PPara[-d, y] R[-a, -b, -x, -y] /. PActivate /.
      PADMActivate // ToCanonical // ContractMetric // CollectTensors];

PerpBComplementDefinition =
 PerpBComplementDefinition /. HG3BExpandLazy // ToNewCanonical //
  CollectTensors;
OrigBComplementDefinition = OrigBComplementDefinition /. HG3BExpandLazy //
   ToNewCanonical // CollectTensors;
SingBComplementDefinition = SingBComplementDefinition /. HG3BExpandLazy //
   ToNewCanonical // CollectTensors;
PerpBComplementDefinition = PerpBComplementDefinition /. ExpandStrengths //
   ToNewCanonical // CollectTensors;
OrigBComplementDefinition = OrigBComplementDefinition /. ExpandStrengths //
   ToNewCanonical // CollectTensors;
SingBComplementDefinition = SingBComplementDefinition /. ExpandStrengths //
   ToNewCanonical // CollectTensors;
```

```
PerpAComplementDefinition =
 PerpAComplementDefinition /. HG3BExpandLazy // ToNewCanonical //
  CollectTensors;
OrigAComplementDefinition = OrigAComplementDefinition /. HG3BExpandLazy //
   ToNewCanonical // CollectTensors;
SingAComplementDefinition = SingAComplementDefinition /. HG3BExpandLazy //
   ToNewCanonical // CollectTensors;
PerpAComplementDefinition = PerpAComplementDefinition /. ExpandStrengths //
   ToNewCanonical // CollectTensors;
OrigAComplementDefinition = OrigAComplementDefinition /. ExpandStrengths //
   ToNewCanonical // CollectTensors;
SingAComplementDefinition = SingAComplementDefinition /. ExpandStrengths //
   ToNewCanonical // CollectTensors;

RawPerpBComplementActivate =
 MakeRule[{PerpBComplement[-i, -k], Evaluate[PerpBComplementDefinition]},
  MetricOn → All, ContractMetrics → True];
RawOrigBComplementActivate = MakeRule[
  {OrigBComplement[-i, -k], Evaluate[OrigBComplementDefinition]},
  MetricOn → All, ContractMetrics → True];
RawSingBComplementActivate = MakeRule[
  {SingBComplement[-i, -k], Evaluate[SingBComplementDefinition]},
  MetricOn → All, ContractMetrics → True];
RawPerpAComplementActivate = MakeRule[
  {PerpAComplement[-i, -j, -k], Evaluate[PerpAComplementDefinition]},
  MetricOn → All, ContractMetrics → True];
RawOrigAComplementActivate = MakeRule[
  {OrigAComplement[-i, -j, -k], Evaluate[OrigAComplementDefinition]},
  MetricOn → All, ContractMetrics → True];
RawSingAComplementActivate = MakeRule[
  {SingAComplement[-i, -j, -k], Evaluate[SingAComplementDefinition]},
  MetricOn → All, ContractMetrics → True];

RawComplementActivate =
 Join[RawPerpBComplementActivate, RawOrigBComplementActivate,
  RawSingBComplementActivate, RawPerpAComplementActivate,
  RawOrigAComplementActivate, RawSingAComplementActivate];

OnShellBLambdaDefinition = (ShellOrigB0p ShellPerpB0p PB0pT[-n, -m, a, c] +
     ShellOrigB1p ShellPerpB1p ShellSingB1p PB1pT[-n, -m, a, c] +
     ShellOrigB2p ShellPerpB2p PB2pT[-n, -m, a, c] +
     ShellOrigB1m ShellPerpB1m ShellSingB1m PB1mT[-n, -m, a, c]) BPiP[-a, -c] +
```

```
    ((1 - ShellOrigB0p) PB0pT[-n, -m, i, k] +
       (1 - ShellOrigB1p) PB1pT[-n, -m, i, k] +
       (1 - ShellOrigB2p) PB2pT[-n, -m, i, k] +
       (1 - ShellOrigB1m) PB1mT[-n, -m, i, k]) OrigBComplement[-i, -k] +
    ((1 - ShellPerpB0p) PB0pT[-n, -m, i, k] +
       (1 - ShellPerpB1p) PB1pT[-n, -m, i, k] +
       (1 - ShellPerpB2p) PB2pT[-n, -m, i, k] +
       (1 - ShellPerpB1m) PB1mT[-n, -m, i, k]) PerpBComplement[-i, -k] +
    ((1 - ShellSingB1p) PB1pT[-n, -m, i, k] +
       (1 - ShellSingB1m) PB1mT[-n, -m, i, k]) OrigBComplement[-i, -k] +
    ((1 - ShellSingB1p) (BetPerpPerp1p / cBetPerpPerp1p) PB1pT[-n, -m, i, k] +
       (1 - ShellSingB1m) (BetPerpPerp1m / cBetPerpPerp1m) PB1mT[-n, -m, i, k])
     SingBComplement[-i, -k];

OnShellALambdaDefinition =
  (ShellOrigA0p ShellPerpA0p ShellSingA0p PA0pT[-n, -m, -o, a, b, c] +
      ShellOrigA1p ShellPerpA1p ShellSingA1p PA1pT[-n, -m, -o, a, b, c] +
      ShellOrigA2p ShellPerpA2p ShellSingA2p PA2pT[-n, -m, -o, a, b, c] +
      ShellOrigA0m ShellPerpA0m ShellSingA0m PA0mT[-n, -m, -o, a, b, c] +
      ShellOrigA1m ShellPerpA1m ShellSingA1m PA1mT[-n, -m, -o, a, b, c] +
      ShellOrigA2m ShellPerpA2m ShellSingA2m PA2mT[-n, -m, -o, a, b, c])
    APiP[-a, -b, -c] +
   ((1 - ShellOrigA0p) PA0pT[-n, -m, -o, i, j, k] +
      (1 - ShellOrigA1p) PA1pT[-n, -m, -o, i, j, k] +
      (1 - ShellOrigA2p) PA2pT[-n, -m, -o, i, j, k] +
      (1 - ShellOrigA0m) PA0mT[-n, -m, -o, i, j, k] +
      (1 - ShellOrigA1m) PA1mT[-n, -m, -o, i, j, k] +
      (1 - ShellOrigA2m) PA2mT[-n, -m, -o, i, j, k])
    OrigAComplement[-i, -j, -k] +
   ((1 - ShellPerpA0p) PA0pT[-n, -m, -o, i, j, k] +
      (1 - ShellPerpA1p) PA1pT[-n, -m, -o, i, j, k] +
      (1 - ShellPerpA2p) PA2pT[-n, -m, -o, i, j, k] +
      (1 - ShellPerpA0m) PA0mT[-n, -m, -o, i, j, k] +
      (1 - ShellPerpA1m) PA1mT[-n, -m, -o, i, j, k] +
      (1 - ShellPerpA2m) PA2mT[-n, -m, -o, i, j, k])
    PerpAComplement[-i, -j, -k] +
   ((1 - ShellSingA0p) PA0pT[-n, -m, -o, i, j, k] +
      (1 - ShellSingA1p) PA1pT[-n, -m, -o, i, j, k] +
      (1 - ShellSingA2p) PA2pT[-n, -m, -o, i, j, k] +
      (1 - ShellSingA0m) PA0mT[-n, -m, -o, i, j, k] +
      (1 - ShellSingA1m) PA1mT[-n, -m, -o, i, j, k] +
      (1 - ShellSingA2m) PA2mT[-n, -m, -o, i, j, k])
```

```
      OrigAComplement[-i, -j, -k] +
     ((1 - ShellSingA0p) (AlpPerpPerp0p / cAlpPerpPerp0p) PA0pT[-n, -m, -o, i, j, k] +
        (1 - ShellSingA1p)
         (AlpPerpPerp1p / cAlpPerpPerp1p) PA1pT[-n, -m, -o, i, j, k] +
        (1 - ShellSingA2p) (AlpPerpPerp2p / cAlpPerpPerp2p)
         PA2pT[-n, -m, -o, i, j, k] +
        (1 - ShellSingA0m) (AlpPerpPerp0m / cAlpPerpPerp0m)
         PA0mT[-n, -m, -o, i, j, k] +
        (1 - ShellSingA1m) (AlpPerpPerp1m / cAlpPerpPerp1m)
         PA1mT[-n, -m, -o, i, j, k] +
        (1 - ShellSingA2m) (AlpPerpPerp2m / cAlpPerpPerp2m)
         PA2mT[-n, -m, -o, i, j, k]) SingAComplement[-i, -j, -k];

  OnShellALambdaDefinition =
   OnShellALambdaDefinition // ToCanonical // ContractMetric;
  OnShellALambdaDefinition = OnShellALambdaDefinition /. RawComplementActivate //
     ToCanonical // ContractMetric;
  RawOnShellALambdaDefinition = OnShellALambdaDefinition /. NewPO3TActivate //
     ToCanonical // ContractMetric;

  OnShellBLambdaDefinition =
   OnShellBLambdaDefinition // ToCanonical // ContractMetric;
  OnShellBLambdaDefinition = OnShellBLambdaDefinition /. RawComplementActivate //
     ToCanonical // ContractMetric;
  RawOnShellBLambdaDefinition = OnShellBLambdaDefinition /. NewPO3TActivate //
     ToCanonical // ContractMetric;

  DumpSave[BinaryLocation[], {RawComplementActivate,
    RawOnShellALambdaDefinition, RawOnShellBLambdaDefinition,
    OrigBComplementDefinition, PerpBComplementDefinition,
    SingBComplementDefinition, OrigAComplementDefinition,
    PerpAComplementDefinition, SingAComplementDefinition}];
  ClearBuild[];
 ];
```

build

In[•]:=

```
OpenLastCache[];
```

build

In[•]:=

```
DefMomentaShell[$ToShellFreedoms_, $ToTheory_, $Theory_] :=
  Module[{PerpBComplementActivate, OrigBComplementActivate,
    SingBComplementActivate, PerpAComplementActivate, OrigAComplementActivate,
    SingAComplementActivate, OnShellBLambdaDefinition,
    OnShellALambdaDefinition, OnShellBLambdaActivate, OnShellALambdaActivate},
```

```
(*a message*)
xAct`xTensor`Private`MakeDefInfo[DefTheory,
 $Theory, {"$PiPShellToPiPPO3 for the theory", ""}];

OrigBComplementDefinition =
 OrigBComplementDefinition /. $ToTheory // ToNewCanonical // CollectTensors;
PerpBComplementDefinition = PerpBComplementDefinition /. $ToTheory //
    ToNewCanonical // CollectTensors;
SingBComplementDefinition = SingBComplementDefinition /. $ToTheory //
    ToNewCanonical // CollectTensors;
OrigAComplementDefinition = OrigAComplementDefinition /. $ToTheory //
    ToNewCanonical // CollectTensors;
PerpAComplementDefinition = PerpAComplementDefinition /. $ToTheory //
    ToNewCanonical // CollectTensors;
SingAComplementDefinition = SingAComplementDefinition /. $ToTheory //
    ToNewCanonical // CollectTensors;

PerpBComplementActivate =
 MakeRule[{PerpBComplement[-i, -k], Evaluate[PerpBComplementDefinition]},
  MetricOn → All, ContractMetrics → True];
OrigBComplementActivate = MakeRule[{OrigBComplement[-i, -k], Evaluate[
     OrigBComplementDefinition]}, MetricOn → All, ContractMetrics → True];
SingBComplementActivate = MakeRule[{SingBComplement[-i, -k], Evaluate[
     SingBComplementDefinition]}, MetricOn → All, ContractMetrics → True];
PerpAComplementActivate = MakeRule[{PerpAComplement[-i, -j, -k], Evaluate[
     PerpAComplementDefinition]}, MetricOn → All, ContractMetrics → True];
OrigAComplementActivate = MakeRule[{OrigAComplement[-i, -j, -k], Evaluate[
     OrigAComplementDefinition]}, MetricOn → All, ContractMetrics → True];
SingAComplementActivate = MakeRule[{SingAComplement[-i, -j, -k], Evaluate[
     SingAComplementDefinition]}, MetricOn → All, ContractMetrics → True];

ComplementActivate = Join[PerpBComplementActivate,
  OrigBComplementActivate, SingBComplementActivate, PerpAComplementActivate,
  OrigAComplementActivate, SingAComplementActivate];

OnShellALambdaDefinition =
 RawOnShellALambdaDefinition /. $ToShellFreedoms /. ToAlp /. TocAlp /.
   $ToTheory // ToNewCanonical;
OnShellBLambdaDefinition = RawOnShellBLambdaDefinition /. $ToShellFreedoms /.
     ToBet /. TocBet /. $ToTheory // ToNewCanonical;

OnShellBLambdaActivate =
 MakeRule[{BPiP[-n, -m], Evaluate[OnShellBLambdaDefinition]},
  MetricOn → All, ContractMetrics → True];
```

```
    OnShellALambdaActivate = MakeRule[{APiP[-n, -m, -o], Evaluate[
        OnShellALambdaDefinition]}, MetricOn → All, ContractMetrics → True];
    $PiPShellToPiPPO3 = Join[OnShellBLambdaActivate, OnShellALambdaActivate];
   ];
ClearBuild[];
```

build

Special 2^- rules

build

In[•]:=

```
(*
ManualA2m=MakeRule[{PiPA2m[-a,-b,-c],-16Alp6 J[]RP2m[-a,-b,-c]},
    MetricOn→All,ContractMetrics→True];
 (*It is essential that we update this for Lambdas*)
*)(*this version for case 16*)
(**)
ManualA2m =
   MakeRule[{PiPA2m[-a, -b, -c], 0}, MetricOn → All, ContractMetrics → True];
 (*It is essential that we update this for Lambdas*)
(**)(*this version for case 26*)
(*comment out for simple Spin1⁺*)
ClearBuild[];
```

build

Calculate $\pi b, \pi A, \mathbb{D}\pi b, \mathbb{D}\pi A$ shell

build

```
Options[ToO3] = {"ToShell" → True, "Order" → Infinity};
ToO3[x_, OptionsPattern[]] := Module[{res, printer}, res = x;
   printer = PrintTemporary[" ** ToO3..."];
   (*res=res/.CDPiToCDPiP;*)
   (*res=res/.CDPiToCDPiPHard;*)
   (*this and the non-Hard line above are new,
   I'm not sure why I didn't need these before?*)
   res = res // NoScalar /. PiToPiP;
   (*not clear how necessary this is!*)
   res = res /. PiToPiP;
   res = ToOrderCanonical[res, OptionValue["Order"]];
   If[OptionValue["ToShell"], res = res /. $PiPShellToPiPPO3];
   res = res // ToNewCanonical;
   res = res /. ToStrengths;
   res = ToOrderCanonical[res, OptionValue["Order"]];
   res = res /. StrengthDecompose;
   res = res /. StrengthLambdaDecompose;
   res = res // ToNewCanonical;
   If[OptionValue["ToShell"], res = res /. $StrengthPShellToStrengthPO3];
   res = res /. StrengthPToStrengthPO3;
   res = res /. StrengthPerpToStrengthPerpO3;
   res = res /. StrengthLambdaPToStrengthLambdaPO3;
   res = res /. StrengthLambdaPerpToStrengthLambdaPerpO3;
   res = res // ToNewCanonical;
   res = res /. PiPToPiPO3;
   res = res // ToNewCanonical;
   (*If[OptionValue["ToShell"],res=res/.ManualA2m];
   *)res = ToOrderCanonical[res, OptionValue["Order"]];
   NotebookDelete[printer];
   res];
ClearBuild[];
```

CDPiP oT CDPiPO3

In[•]:=
```
IfBuild["CDPiPToCDPiPO3",
  tmp = ToO3[APiP[-a, -b, -c], "ToShell" → False];
  tmp = CD[-z][tmp] // ToNewCanonical;
  CDAPiPToCDAPiPO3 = MakeRule[{CD[-z][APiP[-a, -b, -c]], Evaluate[tmp]},
    MetricOn → All, ContractMetrics → True];
  tmp = ToO3[BPiP[-a, -b], "ToShell" → False];
  tmp = CD[-z][tmp] // ToNewCanonical;
  CDBPiPToCDBPiPO3 = MakeRule[{CD[-z][BPiP[-a, -b]], Evaluate[tmp]},
    MetricOn → All, ContractMetrics → True];
  $CDPiPToCDPiPO3 = Join[CDAPiPToCDAPiPO3, CDBPiPToCDBPiPO3];
  DumpSave[BinaryLocation[], {$CDPiPToCDPiPO3}];
  ClearBuild[];
 ];
```

build

In[•]:=
```
OpenLastCache[];
```

build

In[•]:=
```
DefO3MomentaShell[$Theory_] :=
  Module[{tmp, CDBPiPToCDBPiPO3, CDAPiPToCDAPiPO3, TheoryCDBPiPToCDBPiPO3,
    TheoryBPiPToBPiPO3, TheoryCDAPiPToCDAPiPO3, TheoryAPiPToAPiPO3},
   (*a message*)
   xAct`xTensor`Private`MakeDefInfo[DefTheory, $Theory,
    {"$TheoryCDPiPToCDPiPO3, $TheoryPiPToPiPO3 for the theory", ""}];

   tmp = APiP[-a, -b, -c] // ToO3;
   TheoryAPiPToAPiPO3 = MakeRule[
     {APiP[-a, -b, -c], Evaluate[tmp]}, MetricOn → All, ContractMetrics → True];
   tmp = CD[-z][tmp] // ToNewCanonical;
   TheoryCDAPiPToCDAPiPO3 = MakeRule[{CD[-z][APiP[-a, -b, -c]], Evaluate[tmp]},
     MetricOn → All, ContractMetrics → True];
   tmp = BPiP[-a, -b] // ToO3;
   TheoryBPiPToBPiPO3 = MakeRule[
     {BPiP[-a, -b], Evaluate[tmp]}, MetricOn → All, ContractMetrics → True];
   tmp = CD[-z][tmp] // ToNewCanonical;
   TheoryCDBPiPToCDBPiPO3 = MakeRule[{CD[-z][BPiP[-a, -b]], Evaluate[tmp]},
     MetricOn → All, ContractMetrics → True];
   $TheoryCDPiPToCDPiPO3 = Join[TheoryCDAPiPToCDAPiPO3, TheoryCDBPiPToCDBPiPO3];
   $TheoryPiPToPiPO3 = Join[TheoryAPiPToAPiPO3, TheoryBPiPToBPiPO3];
  ];
ClearBuild[];
```

build

ToNesterForm and ToBasicForm

build

ToNesterForm

build

```
Options[TotalToO3] = {"ToShell" → True, "Order" → Infinity};
TotalToO3[x_, OptionsPattern[]] := Module[{res, printer},
    printer = PrintTemporary[" ** TotalToO3 with ToShell ",
      OptionValue["ToShell"], " and Order ", OptionValue["Order"], "..."];
    res = x;
    (**)res = res /. CDPiToCDPiP; (**)
    (**)res = res /. CDPiToCDPiPHard;
    (**)(*this and the non-Hard line above are new,
    I'm not sure why I didn't need these before?*)
    res = res // NoScalar /. PiToPiP;
    (*not clear how necessary this is!*)
    res = res /. PiToPiP;
    res = res /. PiToPiPHard; (*new in 14/04*)
    res = ToOrderCanonical[res, OptionValue["Order"]];
    If[OptionValue["ToShell"],
     res = res /. $TheoryCDPiPToCDPiPO3, res = res /. $CDPiPToCDPiPO3];
    res = res // ToNewCanonical;
    If[OptionValue["ToShell"],
     res = res /. $TheoryPiPToPiPO3, res = res /. PiPToPiPO3];
    res = res // ToNewCanonical;
    res = ToO3[res,
      "ToShell" → OptionValue["ToShell"], "Order" -> OptionValue["Order"]];
    res = ToOrderCanonical[res, OptionValue["Order"]];
    (*res=res//ToNewCanonical;*)
    NotebookDelete[printer];
    res];

CDBToDJDV[x_] := Module[{res, printer},
    printer = PrintTemporary[" ** CDBToDJDV..."];
    res = x;
    res = res /. G3HCDBToDJ;
    res = res // ToNewCanonical;
    res = res /. G3VCDBToG3DV;
    res = res // ToNewCanonical;
    res = res /. CDBCommute;
    res = res // ToNewCanonical;
```

```
    res = res /. G3HCDBToDJ;
    res = res // ToNewCanonical;
    res = res /. G3VCDBToG3DV;
    res = res // ToNewCanonical;
    res = res /. HExpand;
    res = res // ToNewCanonical;
    res = res /. G3HCDBToDJ;
    res = res // ToNewCanonical;
    res = res /. G3VCDBToG3DV;
    res = res // ToNewCanonical;
    res = res /. CDBCommute;
    res = res // ToNewCanonical;
    res = res /. G3HCDBToDJ;
    res = res // ToNewCanonical;
    res = res /. G3VCDBToG3DV;
    res = res // ToNewCanonical;
    NotebookDelete[printer];
    res];

CDToD[x_] := Module[{res, printer},
    printer = PrintTemporary[" ** CDToD..."];
    res = x;
    res = res /. DGrandActivate;
    res = res /. DpGrandActivate;
    res = res /. DpVExpand; (*this is new!*)
    res = res // ToNewCanonical;
    res = res /. epsilonGVToEps;
    res = res /. epsilonGToEpsV;
    res = res // ToNewCanonical;
    NotebookDelete[printer];
    res];

CollapseA[x_] := Module[{res, printer},
    printer = PrintTemporary[" ** CollapseA..."];
    res = x;
    res = res /. CDAToCDAInert;
    res = res /. AExpand;
    res = res /. G3HExpand;
    res = res // ToNewCanonical;
    res = res /. HG3VCDAToHVCDA;
    res = res // ToNewCanonical;
    res = res /. HG3VAToHVA;
    res = res // ToNewCanonical;
    res = res /. G3HExpand;
```

```
    res = res // ToNewCanonical;
    res = res /. HExpand;
    res = res // ToNewCanonical;
    res = res /. CDAInertToCDA;
    res = res // ToNewCanonical;
    res = res /. HG3BExpand;
    (*to deal with the strange combination of A epsilon which cancels*)
    res = res /. G3HExpand;
    res = res /. HEpsToHG3Eps;
    res = res // ToNewCanonical;
    res = res /. AHEpsExpand;
    res = res // ToNewCanonical;
    res = res /. EpsEpsExpand;
    res = res // ToNewCanonical;
    (*finished dealing with this combination*)
    NotebookDelete[printer];
    res];

Options[PreSimplify] = {"Hard" → False, "Order" → Infinity};
PreSimplify[x_, OptionsPattern[]] := Module[{res, printer},
    printer = PrintTemporary[" ** TotalToO3 with Hard ",
      OptionValue["Hard"], " and Order ", OptionValue["Order"], "..."];
    res = x;
    (*res=res//ToNewCanonical;*)(*should re-test after implementing this*)
    res = ToOrderCanonical[res, OptionValue["Order"]];
    If[OptionValue["Hard"], res = res /. HExpand];
    res = res // ToNewCanonical;
    res = res /. HG3BExpandLazy;
    res = res // ToNewCanonical;
    res = res /. G3HExpand;
    res = ToOrderCanonical[res, OptionValue["Order"]];
    (*res=res//ToNewCanonical;*)
    NotebookDelete[printer];
    res];

Options[ToNesterForm] =
  {"ToShell" → True, "Hard" → False, "Order" → Infinity, "GToFoliG" → True};
ToNesterForm[x_, OptionsPattern[]] := Module[{res, printer},
    printer = PrintTemporary[" ** ToNesterForm with Hard ",
      OptionValue["Hard"], " and Order ", OptionValue["Order"],
      " and GToFoliG ", OptionValue["GToFoliG"], "..."];
    res = x;
    res = res /. PhiActivate // NoScalar;
    res = res /. ChiParaActivate // NoScalar;
```

```
    res = res /. ChiPerpActivate // NoScalar;
    res = res /. ChiSingActivate // NoScalar;
    If[OptionValue["ToShell"], res = res /. $ToTheory];
    res = PreSimplify[res,
      "Hard" → OptionValue["Hard"], "Order" → OptionValue["Order"]];
    res = TotalToO3[res, "ToShell" → OptionValue["ToShell"],
      "Order" → OptionValue["Order"]];
    res = res // CDToD;
    res = TotalToO3[res,
      "ToShell" → OptionValue["ToShell"], "Order" → OptionValue["Order"]];
    res = res // CDBToDJDV;
    res = res // CDToD;
    res = TotalToO3[res,
      "ToShell" → OptionValue["ToShell"], "Order" → OptionValue["Order"]];
    res = res // CollapseA;
    If[OptionValue["GToFoliG"], res = res /. GToFoliG];
    res = res // ToNewCanonical;
    res = res /. CollapseJ;
    (*Adding this*)
    (**)
    res = res /. JiToJ;
    (**)
    res = ToOrderCanonical[res, OptionValue["Order"]];
    (*res=res//ToNewCanonical;*)
    NotebookDelete[printer];
    res];
ClearBuild[];
```

build

ToBasicForm

build

In[ ]:=

```
ChiActivate = {ρρ → 1};(*dummy version until the secondaries are determined!*)
Options[ToBasicForm] = {"Hard" → False, "Order" → Infinity};
ToBasicForm[x_, OptionsPattern[]] := Module[{res, printer},
    printer = PrintTemporary[" ** ToBasicForm..."];
    res = x;
    res = res /. PhiActivate // NoScalar;
    res = res /. ChiActivate // NoScalar;
    res = res /. ChiParaActivate // NoScalar;
    res = res /. ChiPerpActivate // NoScalar;
    res = res /. ChiSingActivate // NoScalar;
    res = ToOrderCanonical[res, OptionValue["Order"]];
    res = res /. DpRPDeactivate // NoScalar;
```

```
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. DRPDeactivate // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. RPO3Activate // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. TPO3Activate // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. StrengthPToStrength // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. StrengthLambdaPToStrengthLambda // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. DpPiPDeactivate // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. DPiPDeactivate // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. PiPO3Activate // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. PO3PiActivate // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. PADMPiActivate // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. PiPToPi // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. PhiActivate // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. $ToTheory // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. ExpandStrengths // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = res /. PADMActivate // NoScalar;
    If[OptionValue["Hard"], res = res // ToNewCanonical];
    res = ToOrderCanonical[res, OptionValue["Order"]];
    res = res // NoScalar;
    res = res // ToNewCanonical;
    res = res // NoScalar;
    NotebookDelete[printer];
    res];
ClearBuild[];
```

build

Constraints

build

If-constraints

NesterFormIfConstraints

```
IfBuild["NesterFormIfConstraints",
  NesterFormPhiB0pDefinition = ToNesterForm[PhiB0p[], "ToShell" → False];
  HiGGSPrint[NesterFormPhiB0pDefinition];
  ToNesterFormPhiB0p =
   MakeRule[{PhiB0p[], Evaluate[NesterFormPhiB0pDefinition]},
    MetricOn → All, ContractMetrics → True];
  NesterFormPhiB1pDefinition = ToNesterForm[PhiB1p[-i, -j], "ToShell" → False];
  HiGGSPrint[NesterFormPhiB1pDefinition];
  ToNesterFormPhiB1p =
   MakeRule[{PhiB1p[-i, -j], Evaluate[NesterFormPhiB1pDefinition]},
    MetricOn → All, ContractMetrics → True];
  NesterFormPhiB1mDefinition = ToNesterForm[PhiB1m[-i], "ToShell" → False];
  HiGGSPrint[NesterFormPhiB1mDefinition];
  ToNesterFormPhiB1m =
   MakeRule[{PhiB1m[-i], Evaluate[NesterFormPhiB1mDefinition]},
    MetricOn → All, ContractMetrics → True];
  NesterFormPhiB2pDefinition = ToNesterForm[PhiB2p[-i, -j], "ToShell" → False];
  HiGGSPrint[NesterFormPhiB2pDefinition];
  ToNesterFormPhiB2p =
   MakeRule[{PhiB2p[-i, -j], Evaluate[NesterFormPhiB2pDefinition]},
    MetricOn → All, ContractMetrics → True];
  NesterFormPhiA0pDefinition = ToNesterForm[PhiA0p[], "ToShell" → False];
  HiGGSPrint[NesterFormPhiA0pDefinition];
  ToNesterFormPhiA0p =
   MakeRule[{PhiA0p[], Evaluate[NesterFormPhiA0pDefinition]},
    MetricOn → All, ContractMetrics → True];
  NesterFormPhiA0mDefinition = ToNesterForm[PhiA0m[], "ToShell" → False];
  HiGGSPrint[NesterFormPhiA0mDefinition];
  ToNesterFormPhiA0m =
   MakeRule[{PhiA0m[], Evaluate[NesterFormPhiA0mDefinition]},
    MetricOn → All, ContractMetrics → True];
  NesterFormPhiA1pDefinition = ToNesterForm[PhiA1p[-i, -j], "ToShell" → False];
  HiGGSPrint[NesterFormPhiA1pDefinition];
  ToNesterFormPhiA1p =
   MakeRule[{PhiA1p[-i, -j], Evaluate[NesterFormPhiA1pDefinition]},
    MetricOn → All, ContractMetrics → True];
  NesterFormPhiA1mDefinition = ToNesterForm[PhiA1m[-i], "ToShell" → False];
  HiGGSPrint[NesterFormPhiA1mDefinition];
  ToNesterFormPhiA1m =
```

```
 MakeRule[{PhiA1m[-i], Evaluate[NesterFormPhiA1mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormPhiA2pDefinition = ToNesterForm[PhiA2p[-i, -j], "ToShell" → False];
HiGGSPrint[NesterFormPhiA2pDefinition];
ToNesterFormPhiA2p =
 MakeRule[{PhiA2p[-i, -j], Evaluate[NesterFormPhiA2pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormPhiA2mDefinition = ToNesterForm[
  PhiA2m[-i, -j, -k], "ToShell" → False];
HiGGSPrint[NesterFormPhiA2mDefinition];
ToNesterFormPhiA2m =
 MakeRule[{PhiA2m[-i, -j, -k], Evaluate[NesterFormPhiA2mDefinition]},
  MetricOn → All, ContractMetrics → True];

PhiToNesterFormPhi = Join[ToNesterFormPhiB0p,
  ToNesterFormPhiB1p, ToNesterFormPhiB1m, ToNesterFormPhiB2p,
  ToNesterFormPhiA0p, ToNesterFormPhiA0m, ToNesterFormPhiA1p,
  ToNesterFormPhiA1m, ToNesterFormPhiA2p, ToNesterFormPhiA2m];

NesterFormChiPerpB0pDefinition = ToNesterForm[ChiPerpB0p[], "ToShell" → False];
HiGGSPrint[NesterFormChiPerpB0pDefinition];
ToNesterFormChiPerpB0p =
 MakeRule[{ChiPerpB0p[], Evaluate[NesterFormChiPerpB0pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiPerpB1pDefinition = ToNesterForm[
  ChiPerpB1p[-i, -j], "ToShell" → False];
HiGGSPrint[NesterFormChiPerpB1pDefinition];
ToNesterFormChiPerpB1p =
 MakeRule[{ChiPerpB1p[-i, -j], Evaluate[NesterFormChiPerpB1pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiPerpB1mDefinition = ToNesterForm[
  ChiPerpB1m[-i], "ToShell" → False];
HiGGSPrint[NesterFormChiPerpB1mDefinition];
ToNesterFormChiPerpB1m =
 MakeRule[{ChiPerpB1m[-i], Evaluate[NesterFormChiPerpB1mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiPerpB2pDefinition = ToNesterForm[
  ChiPerpB2p[-i, -j], "ToShell" → False];
HiGGSPrint[NesterFormChiPerpB2pDefinition];
ToNesterFormChiPerpB2p =
 MakeRule[{ChiPerpB2p[-i, -j], Evaluate[NesterFormChiPerpB2pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiPerpA0pDefinition = ToNesterForm[ChiPerpA0p[], "ToShell" → False];
HiGGSPrint[NesterFormChiPerpA0pDefinition];
```

```
ToNesterFormChiPerpA0p =
 MakeRule[{ChiPerpA0p[], Evaluate[NesterFormChiPerpA0pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiPerpA0mDefinition = ToNesterForm[ChiPerpA0m[], "ToShell" → False];
HiGGSPrint[NesterFormChiPerpA0mDefinition];
ToNesterFormChiPerpA0m =
 MakeRule[{ChiPerpA0m[], Evaluate[NesterFormChiPerpA0mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiPerpA1pDefinition = ToNesterForm[
  ChiPerpA1p[-i, -j], "ToShell" → False];
HiGGSPrint[NesterFormChiPerpA1pDefinition];
ToNesterFormChiPerpA1p =
 MakeRule[{ChiPerpA1p[-i, -j], Evaluate[NesterFormChiPerpA1pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiPerpA1mDefinition = ToNesterForm[
  ChiPerpA1m[-i], "ToShell" → False];
HiGGSPrint[NesterFormChiPerpA1mDefinition];
ToNesterFormChiPerpA1m =
 MakeRule[{ChiPerpA1m[-i], Evaluate[NesterFormChiPerpA1mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiPerpA2pDefinition = ToNesterForm[
  ChiPerpA2p[-i, -j], "ToShell" → False];
HiGGSPrint[NesterFormChiPerpA2pDefinition];
ToNesterFormChiPerpA2p =
 MakeRule[{ChiPerpA2p[-i, -j], Evaluate[NesterFormChiPerpA2pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiPerpA2mDefinition = ToNesterForm[
  ChiPerpA2m[-i, -j, -k], "ToShell" → False];
HiGGSPrint[NesterFormChiPerpA2mDefinition];
ToNesterFormChiPerpA2m =
 MakeRule[{ChiPerpA2m[-i, -j, -k], Evaluate[NesterFormChiPerpA2mDefinition]},
  MetricOn → All, ContractMetrics → True];

ChiPerpToNesterFormChiPerp = Join[ToNesterFormChiPerpB0p,
  ToNesterFormChiPerpB1p, ToNesterFormChiPerpB1m, ToNesterFormChiPerpB2p,
  ToNesterFormChiPerpA0p, ToNesterFormChiPerpA0m, ToNesterFormChiPerpA1p,
  ToNesterFormChiPerpA1m, ToNesterFormChiPerpA2p, ToNesterFormChiPerpA2m];

NesterFormChiParaB0mDefinition = ToNesterForm[ChiParaB0m[], "ToShell" → False];
HiGGSPrint[NesterFormChiParaB0mDefinition];
ToNesterFormChiParaB0m =
 MakeRule[{ChiParaB0m[], Evaluate[NesterFormChiParaB0mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiParaB1pDefinition = ToNesterForm[
```

```
  ChiParaB1p[-i, -j], "ToShell" → False];
HiGGSPrint[NesterFormChiParaB1pDefinition];
ToNesterFormChiParaB1p =
 MakeRule[{ChiParaB1p[-i, -j], Evaluate[NesterFormChiParaB1pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiParaB1mDefinition = ToNesterForm[
  ChiParaB1m[-i], "ToShell" → False];
HiGGSPrint[NesterFormChiParaB1mDefinition];
ToNesterFormChiParaB1m =
 MakeRule[{ChiParaB1m[-i], Evaluate[NesterFormChiParaB1mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiParaB2mDefinition = ToNesterForm[
  ChiParaB2m[-i, -j, -k], "ToShell" → False];
HiGGSPrint[NesterFormChiParaB2mDefinition];
ToNesterFormChiParaB2m =
 MakeRule[{ChiParaB2m[-i, -j, -k], Evaluate[NesterFormChiParaB2mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiParaA0pDefinition = ToNesterForm[ChiParaA0p[], "ToShell" → False];
HiGGSPrint[NesterFormChiParaA0pDefinition];
ToNesterFormChiParaA0p =
 MakeRule[{ChiParaA0p[], Evaluate[NesterFormChiParaA0pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiParaA0mDefinition = ToNesterForm[ChiParaA0m[], "ToShell" → False];
HiGGSPrint[NesterFormChiParaA0mDefinition];
ToNesterFormChiParaA0m =
 MakeRule[{ChiParaA0m[], Evaluate[NesterFormChiParaA0mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiParaA1pDefinition = ToNesterForm[
  ChiParaA1p[-i, -j], "ToShell" → False];
HiGGSPrint[NesterFormChiParaA1pDefinition];
ToNesterFormChiParaA1p =
 MakeRule[{ChiParaA1p[-i, -j], Evaluate[NesterFormChiParaA1pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiParaA1mDefinition = ToNesterForm[
  ChiParaA1m[-i], "ToShell" → False];
HiGGSPrint[NesterFormChiParaA1mDefinition];
ToNesterFormChiParaA1m =
 MakeRule[{ChiParaA1m[-i], Evaluate[NesterFormChiParaA1mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiParaA2pDefinition = ToNesterForm[
  ChiParaA2p[-i, -j], "ToShell" → False];
HiGGSPrint[NesterFormChiParaA2pDefinition];
ToNesterFormChiParaA2p =
 MakeRule[{ChiParaA2p[-i, -j], Evaluate[NesterFormChiParaA2pDefinition]},
```

```
  MetricOn → All, ContractMetrics → True];
NesterFormChiParaA2mDefinition = ToNesterForm[
  ChiParaA2m[-i, -j, -k], "ToShell" → False];
HiGGSPrint[NesterFormChiParaA2mDefinition];
ToNesterFormChiParaA2m =
 MakeRule[{ChiParaA2m[-i, -j, -k], Evaluate[NesterFormChiParaA2mDefinition]},
  MetricOn → All, ContractMetrics → True];

ChiParaToNesterFormChiPara = Join[ToNesterFormChiParaB0m,
  ToNesterFormChiParaB1p, ToNesterFormChiParaB1m, ToNesterFormChiParaB2m,
  ToNesterFormChiParaA0p, ToNesterFormChiParaA0m, ToNesterFormChiParaA1p,
  ToNesterFormChiParaA1m, ToNesterFormChiParaA2p, ToNesterFormChiParaA2m];

NesterFormChiSingB1pDefinition =
 ToNesterForm[ChiSingB1p[-i, -j], "ToShell" → False];
HiGGSPrint[NesterFormChiSingB1pDefinition];
ToNesterFormChiSingB1p =
 MakeRule[{ChiSingB1p[-i, -j], Evaluate[NesterFormChiSingB1pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiSingB1mDefinition = ToNesterForm[
  ChiSingB1m[-i], "ToShell" → False];
HiGGSPrint[NesterFormChiSingB1mDefinition];
ToNesterFormChiSingB1m =
 MakeRule[{ChiSingB1m[-i], Evaluate[NesterFormChiSingB1mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiSingA0pDefinition = ToNesterForm[ChiSingA0p[], "ToShell" → False];
HiGGSPrint[NesterFormChiSingA0pDefinition];
ToNesterFormChiSingA0p =
 MakeRule[{ChiSingA0p[], Evaluate[NesterFormChiSingA0pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiSingA0mDefinition = ToNesterForm[ChiSingA0m[], "ToShell" → False];
HiGGSPrint[NesterFormChiSingA0mDefinition];
ToNesterFormChiSingA0m =
 MakeRule[{ChiSingA0m[], Evaluate[NesterFormChiSingA0mDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiSingA1pDefinition = ToNesterForm[
  ChiSingA1p[-i, -j], "ToShell" → False];
HiGGSPrint[NesterFormChiSingA1pDefinition];
ToNesterFormChiSingA1p =
 MakeRule[{ChiSingA1p[-i, -j], Evaluate[NesterFormChiSingA1pDefinition]},
  MetricOn → All, ContractMetrics → True];
NesterFormChiSingA1mDefinition = ToNesterForm[
  ChiSingA1m[-i], "ToShell" → False];
HiGGSPrint[NesterFormChiSingA1mDefinition];
```

```
  ToNesterFormChiSingA1m =
   MakeRule[{ChiSingA1m[-i], Evaluate[NesterFormChiSingA1mDefinition]},
    MetricOn → All, ContractMetrics → True];
  NesterFormChiSingA2pDefinition = ToNesterForm[
    ChiSingA2p[-i, -j], "ToShell" → False];
  HiGGSPrint[NesterFormChiSingA2pDefinition];
  ToNesterFormChiSingA2p =
   MakeRule[{ChiSingA2p[-i, -j], Evaluate[NesterFormChiSingA2pDefinition]},
    MetricOn → All, ContractMetrics → True];
  NesterFormChiSingA2mDefinition = ToNesterForm[
    ChiSingA2m[-i, -j, -k], "ToShell" → False];
  HiGGSPrint[NesterFormChiSingA2mDefinition];
  ToNesterFormChiSingA2m =
   MakeRule[{ChiSingA2m[-i, -j, -k], Evaluate[NesterFormChiSingA2mDefinition]},
    MetricOn → All, ContractMetrics → True];

  ChiSingToNesterFormChiSing =
   Join[ToNesterFormChiSingB1p, ToNesterFormChiSingB1m,
    ToNesterFormChiSingA0p, ToNesterFormChiSingA0m, ToNesterFormChiSingA1p,
    ToNesterFormChiSingA1m, ToNesterFormChiSingA2p, ToNesterFormChiSingA2m];

  DumpSave[BinaryLocation[], {PhiToNesterFormPhi, ChiPerpToNesterFormChiPerp,
    ChiParaToNesterFormChiPara, ChiSingToNesterFormChiSing}];
  ClearBuild[];
 ];
```

build

In[•]:=

```
OpenLastCache[];
```

build

```
ImposeTheory[IfConstraint_, $ToTheory_] :=
  Module[{TensorName, tmp, OnShellValue},
   TensorName = ToString@Head@x;
   (*Not actually needing this yet*)
   tmp = IfConstraint /. PhiToNesterFormPhi;
   tmp = tmp /. ChiPerpToNesterFormChiPerp;
   tmp = tmp /. ChiParaToNesterFormChiPara;
   tmp = tmp /. ChiSingToNesterFormChiSing;
   tmp = tmp /. $ToTheory;
   tmp = tmp // ToNewCanonical;
   tmp = tmp // CollectTensors;
   OnShellValue = tmp;
   tmp = MakeRule[{Evaluate@IfConstraint, Evaluate@OnShellValue},
     MetricOn → All, ContractMetrics → True];
```

```
    $IfConstraintToTheoryNesterForm = {};
    $IfConstraintToTheoryNesterForm =
     $IfConstraintToTheoryNesterForm ~ Join ~ tmp;
    OnShellValue];

DefIfConstraintToTheoryNesterForm[$ToShellFreedoms_, $ToTheory_, $Theory_] :=
  Module[{Phis, ChiPerps, ChiParas, ChiSings},
   (*a message*)
   xAct`xTensor`Private`MakeDefInfo[DefTheory, $Theory,
    {"$IfConstraintToTheoryNesterForm for the theory", ""}];
   $IfConstraints = {};
   Phis = DeleteCases[
     Evaluate[{(1 - ShellOrigB0p) PhiB0p[], (1 - ShellOrigB1p) PhiB1p[-i, -j],
        (1 - ShellOrigB1m) PhiB1m[-i], (1 - ShellOrigB2p) PhiB2p[-i, -j],
        (1 - ShellOrigA0p) PhiA0p[], (1 - ShellOrigA0m) PhiA0m[],
        (1 - ShellOrigA1p) PhiA1p[-i, -j], (1 - ShellOrigA1m) PhiA1m[-i],
        (1 - ShellOrigA2p) PhiA2p[-i, -j], (1 - ShellOrigA2m) PhiA2m[-i, -j, -k]} /.
       $ToShellFreedoms], 0, Infinity];
   $IfConstraints = $IfConstraints ~ Join ~ Phis;
   Phis = ({#, ImposeTheory[#, $ToTheory]}) & /@ Phis;
   HiGGSPrint["** DefTheory: Found the following primary if-constraints:"];
   (HiGGSPrint[#[[1]], " ≡ ", #[[2]], " ≈ 0"]) & /@ Phis;
   Phis =
    DeleteCases[Evaluate[{(ShellOrigB0p) PhiB0p[], (ShellOrigB1p) PhiB1p[-i, -j],
       (ShellOrigB1m) PhiB1m[-i], (ShellOrigB2p) PhiB2p[-i, -j], (ShellOrigA0p)
        PhiA0p[], (ShellOrigA0m) PhiA0m[], (ShellOrigA1p) PhiA1p[-i, -j],
       (ShellOrigA1m) PhiA1m[-i], (ShellOrigA2p) PhiA2p[-i, -j],
       (ShellOrigA2m) PhiA2m[-i, -j, -k]} /. $ToShellFreedoms], 0, Infinity];
   Phis = ({#, ImposeTheory[#, $ToTheory]}) & /@ Phis;
   ChiPerps = DeleteCases[Evaluate[
      {(1 - ShellPerpB0p) ChiPerpB0p[], (1 - ShellPerpB1p) ChiPerpB1p[-i, -j],
       (1 - ShellPerpB1m) ChiPerpB1m[-i], (1 - ShellPerpB2p) ChiPerpB2p[-i, -j],
       (1 - ShellPerpA0p) ChiPerpA0p[], (1 - ShellPerpA0m) ChiPerpA0m[],
       (1 - ShellPerpA1p) ChiPerpA1p[-i, -j], (1 - ShellPerpA1m) ChiPerpA1m[-i],
       (1 - ShellPerpA2p) ChiPerpA2p[-i, -j], (1 - ShellPerpA2m)
        ChiPerpA2m[-i, -j, -k]} /. $ToShellFreedoms], 0, Infinity];
   $IfConstraints = $IfConstraints ~ Join ~ ChiPerps;
   ChiPerps = ({#, ImposeTheory[#, $ToTheory]}) & /@ ChiPerps;
   HiGGSPrint["** DefTheory: Found the
      following secondary perpendicular if-constraints:"];
   (HiGGSPrint[#[[1]], " ≡ ", #[[2]], " ≈ 0"]) & /@ ChiPerps;
   ChiParas = DeleteCases[Evaluate[{(1 - ShellParaB0m) ChiParaB0m[],
       (1 - ShellParaB1p) ChiParaB1p[-i, -j], (1 - ShellParaB1m) ChiParaB1m[-i],
```

```
        (1 - ShellParaB2m) ChiParaB2m[-i, -j, -k], (1 - ShellParaA0p) ChiParaA0p[],
        (1 - ShellParaA0m) ChiParaA0m[], (1 - ShellParaA1p) ChiParaA1p[-i, -j],
        (1 - ShellParaA1m) ChiParaA1m[-i], (1 - ShellParaA2p) ChiParaA2p[-i, -j],
        (1 - ShellParaA2m) ChiParaA2m[-i, -j, -k]} /.
      $ToShellFreedoms], 0, Infinity];
  $IfConstraints = $IfConstraints ~Join~ ChiParas;
  ChiParas = ({#, ImposeTheory[#, $ToTheory]}) & /@ ChiParas;
  HiGGSPrint[
   "** DefTheory: Found the following secondary parallel if-constraints:"];
  (HiGGSPrint[#[[1]], " ≡ ", #[[2]], " ≈ 0"]) & /@ ChiParas;
  ChiSings = DeleteCases[Evaluate[
     {(1 - ShellSingB1p) ChiSingB1p[-i, -j], (1 - ShellSingB1m) ChiSingB1m[-i],
      (1 - ShellSingA0p) ChiSingA0p[], (1 - ShellSingA0m) ChiSingA0m[],
      (1 - ShellSingA1p) ChiSingA1p[-i, -j], (1 - ShellSingA1m) ChiSingA1m[-i],
      (1 - ShellSingA2p) ChiSingA2p[-i, -j], (1 - ShellSingA2m)
       ChiSingA2m[-i, -j, -k]} /. $ToShellFreedoms], 0, Infinity];
  $IfConstraints = $IfConstraints ~Join~ ChiSings;
  ChiSings = ({#, ImposeTheory[#, $ToTheory]}) & /@ ChiSings;
  HiGGSPrint[
   "** DefTheory: Found the following secondary singular if-constraints:"];
  (HiGGSPrint[#[[1]], " ≡ ", #[[2]], " ≈ 0"]) & /@ ChiSings;
 ];
ClearBuild[];
```

build

Sure constraints

build

Super-Hamiltonian

build

```
Options[DefSuperHamiltonian] = {"Order" → 1, "ProtectSurface" → False};
```

build

```
DefSuperHamiltonian[$ToShellFreedoms_, $IfConstraintToNesterForm_, $ToTheory_,
  $Theory_, OptionsPattern[]] := Module[{MainPart, GradPart, res},
  (*a message*)
  xAct`xTensor`Private`MakeDefInfo[DefTheory,
   $Theory, {"super-Hamiltonian for the theory", ""}];
  MainPart = J[] ((1/16) (cPerpB0p (1/BetPerpPerp0p)
          ShellOrigB0p PhiB0p[] PhiB0p[] +
        cPerpB1p (1/BetPerpPerp1p) ShellOrigB1p PhiB1p[-a, -b] PhiB1p[a, b] +
        cPerpB1m (1/BetPerpPerp1m) ShellOrigB1m PhiB1m[-a] PhiB1m[a] +
```

```
        cPerpB2p (1 / BetPerpPerp2p) ShellOrigB2p PhiB2p[-a, -b] PhiB2p[a, b] +
        (1 / 4) (cPerpA0p (1 / AlpPerpPerp0p) ShellOrigA0p PhiA0p[] PhiA0p[] +
           cPerpA0m (1 / AlpPerpPerp0m) ShellOrigA0m PhiA0m[] PhiA0m[] +
           cPerpA1p (1 / AlpPerpPerp1p)
            ShellOrigA1p PhiA1p[-a, -b] PhiA1p[a, b] +
           cPerpA1m (1 / AlpPerpPerp1m) ShellOrigA1m PhiA1m[-a] PhiA1m[a] +
           cPerpA2p (1 / AlpPerpPerp2p)
            ShellOrigA2p PhiA2p[-a, -b] PhiA2p[a, b] +
           cPerpA2m (1 / AlpPerpPerp2m) ShellOrigA2m
            PhiA2m[-a, -b, -c] PhiA2m[a, b, c])) -
     (J[] T[i, -m, -n] PPara[m, g] PPara[n, h] (Bet1 PT1[-i, -g, -h, a, c, d] +
         Bet2 PT2[-i, -g, -h, a, c, d] +
         Bet3 PT3[-i, -g, -h, a, c, d]) PPara[-c, e] PPara[-d, f] T[-a, -e, -f] +
       J[] TLambda[i, g, h] (cBet1 PT1[-i, -g, -h, a, c, d] +
         cBet2 PT2[-i, -g, -h, a, c, d] +
         cBet3 PT3[-i, -g, -h, a, c, d])
        PPara[-c, p] PPara[-d, q] T[-a, -p, -q] +
       J[] (R[i, j, -m, -n] PPara[m, g] PPara[n, h]
           (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] +
            Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
            Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
            Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
            Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
            Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) - (1 / 2) Alp0 PPara[a, c]
          PPara[b, d]) PPara[-c, e] PPara[-d, f] R[-a, -b, -e, -f] +
       RLambda[i, j, g, h] (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
         cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
         cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] +
         cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
         cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] +
         cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
        PPara[-c, p] PPara[-d, q] R[-a, -b, -p, -q]);

MainPart = MainPart /. TocPerp;
MainPart = MainPart /. ToAlp;
MainPart = MainPart /. ToBet;
MainPart = MainPart /. PActivate;
MainPart = MainPart // ToNewCanonical;
MainPart = MainPart /. PADMActivate;
(*Remember to enforce zeros above zeros in advance*)
MainPart = MainPart /. $ToShellFreedoms;
MainPart = MainPart // ToNewCanonical;
MainPart = MainPart // CollectTensors;
```

```
    MainPart = MainPart /. $ToTheory;
    MainPart = MainPart // ToNewCanonical;
    MainPart = MainPart // CollectTensors;
    MainPart = MainPart // NoScalar;
    MainPart = MainPart /. $IfConstraintToTheoryNesterForm;
    MainPart = ToNesterForm[MainPart,
      "ToShell" → True, "Hard" → True, "Order" → OptionValue@"Order"];
    MainPart = MainPart // ToNewCanonical;
    MainPart = MainPart // CollectTensors;
    GradPart = -V[k] G3[-b, n] (CD[-n][BPiP[-k, j] H[-j, b]] -
        A[i, -k, -n] BPiP[-i, j] PPara[-j, m] H[-m, b]);
    GradPart = GradPart /. PADMActivate;
    If[! OptionValue@"ProtectSurface",
     GradPart = ToNesterForm[GradPart,
        "ToShell" → True, "Hard" → True, "Order" → OptionValue@"Order"];
    ];
    GradPart = MainPart + GradPart // ToNewCanonical;
    GradPart = GradPart // CollectTensors;
    HiGGSPrint["** DefTheory: The super-Hamiltonian is:"];
    HiGGSPrint[SuperHamiltonian0p[], " ≡ ", GradPart, " ≈ 0"];
   ];
ClearBuild[];
```

build

Linear super-momentum

build

```
Options[DefLinearSuperMomentum] = {"Order" → 1, "ProtectSurface" → False};
```

build

```
DefLinearSuperMomentum[$ToShellFreedoms_, $IfConstraintToNesterForm_, $ToTheory_,
    $Theory_, OptionsPattern[]] := Module[{MainPart, GradPart, res},
    (*a message*)
    xAct`xTensor`Private`MakeDefInfo[DefTheory,
     $Theory, {"linear super-momentum for the theory", ""}];
    MainPart = BPiP[-i, r] PPara[-r, p] PPara[-l, q] T[i, -q, -p] +
      (1/2) APiP[-i, -j, r] PPara[-r, p] PPara[-l, q] R[i, j, -q, -p];
    MainPart = MainPart /. PADMActivate;
    MainPart = ToNesterForm[MainPart,
      "ToShell" → True, "Hard" → True, "Order" → OptionValue@"Order"];
    MainPart = MainPart // ToNewCanonical;
    MainPart = MainPart // CollectTensors;

    GradPart = -PPara[-l, k] G3[-b, n] (CD[-n][BPiP[-k, j] H[-j, b]] +
        A[i, -k, -n] BPiP[-i, j] PPara[-j, m] H[-m, b]);
    GradPart = GradPart /. PADMActivate;
    If[! OptionValue@"ProtectSurface",
     GradPart = ToNesterForm[GradPart,
        "ToShell" → True, "Hard" → True, "Order" → OptionValue@"Order"];
    ];
    GradPart = MainPart + GradPart // ToNewCanonical;
    GradPart = GradPart // CollectTensors;

    HiGGSPrint["** DefTheory: The linear super-momentum is:"];
    HiGGSPrint[LinearSuperMomentum1m[-l], " ≡ ", GradPart, " ≈ 0"];
   ];
ClearBuild[];
```

build

Angular super-momentum

build

```
Options[DefAngularSuperMomentum] = {"Order" → 1, "ProtectSurface" → False};
```

build

```
DefAngularSuperMomentum[$ToShellFreedoms_, $IfConstraintToNesterForm_,
    $ToTheory_, $Theory_, OptionsPattern[]] := Module[{MainPart, GradPart, res},
    (*a message*)
    xAct`xTensor`Private`MakeDefInfo[DefTheory,
     $Theory, {"angular super-momentum for the theory", ""}];
    MainPart = 2 V[k] PPara[-m, l] Antisymmetrize[
       BPi[-k, a] G3[-a, b] B[-l, -b], {-k, -l}];
    MainPart = MainPart /. PADMActivate;
```

```
    MainPart = ToNesterForm[MainPart,
      "ToShell" → True, "Hard" → True, "Order" → OptionValue@"Order"];
    MainPart = MainPart // ToNewCanonical;
    MainPart = MainPart // CollectTensors;

    GradPart = V[k] PPara[-m, l] G3[-b, p] (CD[-p][APiP[-k, -l, j] H[-j, b]]) +
      V[k] PPara[-m, l] G3[-b, p] (-2 Antisymmetrize[
          A[i, -k, -p] APiP[-i, -l, j] PPara[-j, z] H[-z, b], {-k, -l}]);
    GradPart = GradPart /. PADMActivate;
    If[! OptionValue@"ProtectSurface",
     GradPart = ToNesterForm[GradPart,
        "ToShell" → True, "Hard" → True, "Order" → OptionValue@"Order"];
    ];
    GradPart = MainPart + GradPart // ToNewCanonical;
    GradPart = GradPart // CollectTensors;

    HiGGSPrint["** DefTheory: The 1- part of the angular super-momentum is:"];
    HiGGSPrint[RotationalSuperMomentum1m[-m], " ≡ ", GradPart, " ≈ 0"];

    MainPart = 2 PPara[-n, k] PPara[-m, l]
      Antisymmetrize[BPi[-k, a] G3[-a, b] B[-l, -b], {-k, -l}];
    MainPart = MainPart /. PADMActivate;
    MainPart = ToNesterForm[MainPart,
      "ToShell" → True, "Hard" → True, "Order" → OptionValue@"Order"];
    MainPart = MainPart // ToNewCanonical;
    MainPart = MainPart // CollectTensors;

    GradPart =
     PPara[-n, k] PPara[-m, l] G3[-b, p] (CD[-p][APiP[-k, -l, j] H[-j, b]]) +
      PPara[-n, k] PPara[-m, l] G3[-b, p] (-2 Antisymmetrize[
          A[i, -k, -p] APiP[-i, -l, j] PPara[-j, z] H[-z, b], {-k, -l}]);
    GradPart = GradPart /. PADMActivate;
    If[! OptionValue@"ProtectSurface",
     GradPart = ToNesterForm[GradPart,
        "ToShell" → True, "Hard" → True, "Order" → OptionValue@"Order"];
    ];
    GradPart = MainPart + GradPart // ToNewCanonical;
    GradPart = GradPart // CollectTensors;

    HiGGSPrint["** DefTheory: The 1+ part of the angular super-momentum is:"];
    HiGGSPrint[RotationalSuperMomentum1p[-n, -m], " ≡ ", GradPart, " ≈ 0"];
   ];
ClearBuild[];
```

build

PoissonBracket

build

```
DefTensor[KX[-a, -b, -c], M4];
DefTensor[KKX[-a, -b, -c, -d], M4, Antisymmetric[{-b, -c}]];
DefTensor[KXP[-a, -b, -c], M4];
DefTensor[KKXP[-a, -b, -c, -d], M4, Antisymmetric[{-b, -c}]];
InertDerB = MakeRule[{CD[-a][B[-b, -c]], KX[-a, -b, -c]},
    MetricOn → All, ContractMetrics → True];
InertDerA = MakeRule[{CD[-a][A[-b, -c, -d]], KKX[-a, -b, -c, -d]},
    MetricOn → All, ContractMetrics → True];
InertDerBP = MakeRule[{CD[-a][BPi[-b, -c]], KXP[-a, -b, -c]},
    MetricOn → All, ContractMetrics → True];
InertDerAP = MakeRule[{CD[-a][APi[-b, -c, -d]], KKXP[-a, -b, -c, -d]},
    MetricOn → All, ContractMetrics → True];
InertDer = Join[InertDerB, InertDerA, InertDerBP, InertDerAP];
InertDerRevB = MakeRule[{KX[-a, -b, -c], CD[-a][B[-b, -c]]},
    MetricOn → All, ContractMetrics → True];
InertDerRevA = MakeRule[{KKX[-a, -b, -c, -d], CD[-a][A[-b, -c, -d]]},
    MetricOn → All, ContractMetrics → True];
InertDerRevBP = MakeRule[{KXP[-a, -b, -c], CD[-a][BPi[-b, -c]]},
    MetricOn → All, ContractMetrics → True];
InertDerRevAP = MakeRule[{KKXP[-a, -b, -c, -d], CD[-a][APi[-b, -c, -d]]},
    MetricOn → All, ContractMetrics → True];
InertDerRev = Join[InertDerRevB, InertDerRevA, InertDerRevBP, InertDerRevAP];

Derivative3B = MakeRule[{CD[-a][B[b, -c]], G3[-a, d] CD[-d][B[b, -c]]},
    MetricOn → All, ContractMetrics → True];
Derivative3A = MakeRule[{CD[-a][A[b, e, -c]], G3[-a, d] CD[-d][A[b, e, -c]]},
    MetricOn → All, ContractMetrics → True];
Derivative3 = Join[Derivative3B, Derivative3A];

ForceAToZeroExplicit =
  MakeRule[{A[i, j, -k], 0}, MetricOn → All, ContractMetrics → True];
 (*This is included for the surface terms, use very carefully!*)
ForceQToZero = MakeRule[{Q[i, j], 0}, MetricOn → All, ContractMetrics → True];
(*This is included for the surface terms, use very carefully!*)
ForceAToZero = Join[ForceAToZeroExplicit, ForceQToZero];

DefTensor[DummyGradient[-z], M4, PrintAs → "𝔻̂", OrthogonalTo → {V[z]}];
DefTensor[DummyHessian[-z, -w], M4, PrintAs → "𝔻̂𝔻̂", OrthogonalTo → {V[z], V[w]}];
DefTensor[DummyGradientGreek[-z], M4, PrintAs → "𝔻"];
```

```
DefTensor[DummyHessianGreek[-z, -w], M4, PrintAs → "𝔻𝔻"];
DummyGradientGreekActivate =
  MakeRule[{DummyGradientGreek[-b], DummyGradient[-i] B[i, -a] G3[a, -b]},
   MetricOn → All, ContractMetrics → True];
DummyHessianGreekActivate = MakeRule[{DummyHessianGreek[-b, -c],
    DummyHessian[-i, -j] B[i, -a] G3[a, -b] B[j, -d] G3[d, -c]},
   MetricOn → All, ContractMetrics → True];
DummyDerivativeGreekActivate = Join[DummyGradientGreekActivate,
   DummyHessianGreekActivate];

ManualCovariantDerivative[ind_, expr_, greeks_, dummy_] :=
  Module[{res, Inds, UpperInds, LowerInds},
   Inds = Complement[FindFreeIndices[expr], greeks];
   LowerInds = Select[Inds, (Quiet[#[[1]]] == -1) &];
   UpperInds = Complement[Inds, LowerInds];
   res = CD[ind][expr];
   Scan[(res = res - A[dummy, #, ind] ReplaceIndex[Evaluate[expr], # → -dummy]) &,
    LowerInds];
   Scan[(res = res + A[#, -dummy, ind] ReplaceIndex[Evaluate[expr], # → dummy]) &,
    UpperInds];
   res = res // ToNewCanonical;
   res];
```

build

```
(*
PoissonBracket[f1x_,f2x_,options__?
    ((OptionQ@#&&({#}~MemberQ~("Parallel"->True)))&)]:=Catch@Module[{},
    (*Build the HiGGS environment*)
    HiGGSPrint["we got there"];
    BuildHiGGS[];
    (*Define the theory*)
    DefTheory["Import"->$TheoryName];
    HiGGSPrint["fin"];
    (*Evaluate the Poisson bracket*)
    PoissonBracket[f1x,f2x,
     ({options}~Complement~{"Parallel"->True})/.{List->Sequence}]];
*)

PoissonBracketParallel[f1x_, f2x_, theory_String, options___] := Module[{result},
   (*Build the HiGGS environment*)
   (*$Timing=True;*)
   BuildHiGGS[];
   (*import theory names*)
```

```
    Quiet@ToExpression["<<" <> FileNameJoin@
        {$WorkingDirectory, "svy", "node-" <> $Node, "peta4.nom.mx"} <> ";"];
    (*Define the theory*)
    DefTheory["Import" -> theory];
    (*Export to the usual PB function*)
    result = PoissonBracket[f1x, f2x, options];
    ForceTiming[];
    result];

DistributeDefinitions[PoissonBracketParallel];

Options[PoissonBracket] = {"ToShell" → True, "Hard" → False, "Surficial" → False,
    "Order" → Infinity, "GToFoliG" → True, "PreTruncate" → False,
    "NesterForm" → True, "PrintAnswer" -> True, "Parallel" -> False};

PoissonBracket[f1x_, f2x_, OptionsPattern[]] :=
  "PoissonBracket"~TimeWrapper~Catch@Module[
     {sur, sur1, sur2, res, ris, f1, f2, f1a, f2a, f1b, f2b, nf1, nf2, NonVanishing,
      final, failtrue, BracketForm, BracketAnsatzFull, BracketAnsatz,
      BracketSolution, AnsatzSolutions, difference, ret, test, Variationalf1B,
      Variationalf2B, Variationalf1A, Variationalf2A, Variationalf1BPi,
      Variationalf2BPi, Variationalf1APi, Variationalf2APi, Partialf1B,
      Partialf2B, Partialf1A, Partialf2A, Partialf1BPi, Partialf2BPi,
      Partialf1APi, Partialf2APi, Partialf1DBz, Partialf2DBz, Partialf1DAz,
      Partialf2DAz, Partialf1DBPiz, Partialf2DBPiz, Partialf1DAPiz,
      Partialf2DAPiz, Partialf1DBv, Partialf2DBv, Partialf1DAv, Partialf2DAv,
      Partialf1DBPiv, Partialf2DBPiv, Partialf1DAPiv, Partialf2DAPiv,
      BarPartialf1B, BarPartialf2B, BarPartialf1A, BarPartialf2A,
      BarPartialf1BPi, BarPartialf2BPi, BarPartialf1APi, BarPartialf2APi,
      BarVariationalf1B, BarVariationalf2B, BarVariationalf1A, BarVariationalf2A,
      BarVariationalf1BPi, BarVariationalf2BPi, BarVariationalf1APi,
      BarVariationalf2APi, DeltaDelta, DDeltaDelta, DeltaDDelta, DDeltaDDelta,
      return, fieldversion, momentafail, ras, D0Term, D1Term, D2Term,
      D0TermPrimitive, SecondIndices, printer, printer2, printer3},
     (*distributed defs seem not to be working for split def*)
     If[OptionValue["Parallel"],
      (*Build the HiGGS environment*)
      (*$Timing=True;*)
      BuildHiGGS[];
      (*import theory names*)
      Quiet@ToExpression["<<" <> FileNameJoin@
          {$WorkingDirectory, "svy", "node-" <> $Node, "peta4.nom.mx"} <> ";"];
      (*Define the theory*)
      DefTheory["Import" -> $TheoryName];
```

```
];
(*a message*)
printer = {};
printer = printer ~Append~ PrintTemporary[" ** PoissonBracket ",
   {f1x, f2x}, " with options ", Options[PoissonBracket], "..."];
f1 = ToBasicForm[f1x, "Hard" → True];
f1 = f1 // NoScalar;
If[OptionValue["PreTruncate"], f1 = ToOrderCanonical[f1, 1]];
f2 = ToBasicForm[f2x, "Hard" → True];
f2 = f2 // NoScalar;
If[OptionValue["PreTruncate"], f2 = ToOrderCanonical[f2, 1]];
printer = printer ~Append~
  PrintTemporary[" ** PoissonBracket: BasicForm to be evaluated is:"];
printer = printer ~Append~ PrintTemporary[{f1, f2}];
nf1 = Length[FindFreeIndices[f1]];
nf2 = Length[FindFreeIndices[f2]];
f1a = ReplaceDummies[f1];
f2a = ReplaceDummies[f2];
BracketForm = f1x f2x // ToCanonical;
f1a = f1a /. Derivative3;
f2a = f2a /. Derivative3;
f1b = f1a /. InertDer;
f1b = f1b // NoScalar;
f2b = f2a /. InertDer;
f2b = f2b // NoScalar;
Variationalf1B = NewVarAction[f1a, B[q, -r]] + DVDB[-x, -q, r]
    NewVarAction[f1a, V[-x]] + DHDB[-x, y, -q, r] NewVarAction[f1a, H[-x, y]] +
  DJDB[-q, r] NewVarAction[f1a, J[]] + DLapseDB[-q, r]
    NewVarAction[f1a, Lapse[]] + DJiDB[-q, r] NewVarAction[f1a, Ji[]];
Variationalf2B = NewVarAction[f2a, B[q, -r]] + DVDB[-x, -q, r]
    NewVarAction[f2a, V[-x]] + DHDB[-x, y, -q, r] NewVarAction[f2a, H[-x, y]] +
  DJDB[-q, r] NewVarAction[f2a, J[]] + DLapseDB[-q, r]
    NewVarAction[f2a, Lapse[]] + DJiDB[-q, r] NewVarAction[f2a, Ji[]];
Variationalf1A = NewVarAction[f1a, A[q, r, -s]];
Variationalf2A = NewVarAction[f2a, A[q, r, -s]];
Variationalf1BPi = NewVarAction[f1a, BPi[-q, r]];
Variationalf2BPi = NewVarAction[f2a, BPi[-q, r]];
Variationalf1APi = NewVarAction[f1a, APi[-q, -r, s]];
Variationalf2APi = NewVarAction[f2a, APi[-q, -r, s]];
Partialf1B = NewVarAction[f1b, B[q, -r]] + DVDB[-x, -q, r]
    NewVarAction[f1b, V[-x]] + DHDB[-x, y, -q, r] NewVarAction[f1b, H[-x, y]] +
  DJDB[-q, r] NewVarAction[f1b, J[]] + DLapseDB[-q, r]
    NewVarAction[f1b, Lapse[]] + DJiDB[-q, r] NewVarAction[f1b, Ji[]];
Partialf2B = NewVarAction[f2b, B[q, -r]] + DVDB[-x, -q, r]
```

```
    NewVarAction[f2b, V[-x]] + DHDB[-x, y, -q, r] NewVarAction[f2b, H[-x, y]] +
   DJDB[-q, r] NewVarAction[f2b, J[]] + DLapseDB[-q, r]
    NewVarAction[f2b, Lapse[]] + DJiDB[-q, r] NewVarAction[f2b, Ji[]];
Partialf1A = NewVarAction[f1b, A[q, r, -s]];
Partialf2A = NewVarAction[f2b, A[q, r, -s]];
Partialf1BPi = NewVarAction[f1b, BPi[-q, r]];
Partialf2BPi = NewVarAction[f2b, BPi[-q, r]];
Partialf1APi = NewVarAction[f1b, APi[-q, -r, s]];
Partialf2APi = NewVarAction[f2b, APi[-q, -r, s]];
Partialf1DBz = NewVarAction[f1b, KX[-z, q, -r]];
Partialf2DBz = NewVarAction[f2b, KX[-z, q, -r]];
Partialf1DAz = NewVarAction[f1b, KKX[-z, q, r, -s]];
Partialf2DAz = NewVarAction[f2b, KKX[-z, q, r, -s]];
Partialf1DBPiz = NewVarAction[f1b, KXP[-z, -q, r]];
Partialf2DBPiz = NewVarAction[f2b, KXP[-z, -q, r]];
Partialf1DAPiz = NewVarAction[f1b, KKXP[-z, -q, -r, s]];
Partialf2DAPiz = NewVarAction[f2b, KKXP[-z, -q, -r, s]];
Partialf1DBv = NewVarAction[f1b, KX[-v, q, -r]];
Partialf2DBv = NewVarAction[f2b, KX[-v, q, -r]];
Partialf1DAv = NewVarAction[f1b, KKX[-v, q, r, -s]];
Partialf2DAv = NewVarAction[f2b, KKX[-v, q, r, -s]];
Partialf1DBPiv = NewVarAction[f1b, KXP[-v, -q, r]];
Partialf2DBPiv = NewVarAction[f2b, KXP[-v, -q, r]];
Partialf1DAPiv = NewVarAction[f1b, KKXP[-v, -q, -r, s]];
Partialf2DAPiv = NewVarAction[f2b, KKXP[-v, -q, -r, s]];
If[OptionValue["Surficial"], {
  printer2 = {};
  printer2 = printer2 ~Append~
    PrintTemporary[" ** PoissonBracket: Finding barred derivatives..."];
  BarPartialf1B = ReplaceDummies@(Partialf1B -
      ReplaceIndex[Evaluate[Partialf1DBz], -q → -w] A[w, -q, -z]);
  BarPartialf2B = ReplaceDummies@(Partialf2B -
      ReplaceIndex[Evaluate[Partialf2DBz], -q → -w] A[w, -q, -z]);
  BarPartialf1A = ReplaceDummies@(Partialf1A -
      ReplaceIndex[Evaluate[Partialf1DAz], -q → -w] A[w, -q, -z] -
      ReplaceIndex[Evaluate[Partialf1DAz], -r → -w] A[w, -r, -z]);
  BarPartialf2A = ReplaceDummies@(Partialf2A -
      ReplaceIndex[Evaluate[Partialf2DAz], -q → -w] A[w, -q, -z] -
      ReplaceIndex[Evaluate[Partialf2DAz], -r → -w] A[w, -r, -z]);
  BarPartialf1BPi = ReplaceDummies@(Partialf1BPi +
      ReplaceIndex[Evaluate[Partialf1DBPiz], q → w] A[q, -w, -z]);
  BarPartialf2BPi = ReplaceDummies@(Partialf2BPi +
      ReplaceIndex[Evaluate[Partialf2DBPiz], q → w] A[q, -w, -z]);
```

```
BarPartialf1APi = ReplaceDummies@(Partialf1APi +
    ReplaceIndex[Evaluate[Partialf1DAPiz], q → w] A[q, -w, -z] +
    ReplaceIndex[Evaluate[Partialf1DAPiz], r → w] A[r, -w, -z]);
BarPartialf2APi = ReplaceDummies@(Partialf2APi +
    ReplaceIndex[Evaluate[Partialf2DAPiz], q → w] A[q, -w, -z] +
    ReplaceIndex[Evaluate[Partialf2DAPiz], r → w] A[r, -w, -z]);
BarVariationalf1B = ReplaceDummies@(BarPartialf1B -
    ManualCovariantDerivative[-z, Partialf1DBz, IndexList[z, r], w]);
BarVariationalf2B = ReplaceDummies@(BarPartialf2B -
    ManualCovariantDerivative[-z, Partialf2DBz, IndexList[z, r], w]);
BarVariationalf1A = ReplaceDummies@(BarPartialf1A -
    ManualCovariantDerivative[-z, Partialf1DAz, IndexList[z, s], w]);
BarVariationalf2A = ReplaceDummies@(BarPartialf2A -
    ManualCovariantDerivative[-z, Partialf2DAz, IndexList[z, s], w]);
BarVariationalf1BPi = ReplaceDummies@(BarPartialf1BPi -
    ManualCovariantDerivative[-z, Partialf1DBPiz, IndexList[z, -r], w]);
BarVariationalf2BPi = ReplaceDummies@(BarPartialf2BPi -
    ManualCovariantDerivative[-z, Partialf2DBPiz, IndexList[z, -r], w]);
BarVariationalf1APi = ReplaceDummies@(BarPartialf1APi -
    ManualCovariantDerivative[-z, Partialf1DAPiz, IndexList[z, -s], w]);
BarVariationalf2APi = ReplaceDummies@(BarPartialf2APi -
    ManualCovariantDerivative[-z, Partialf2DAPiz, IndexList[z, -s], w]);
printer2 = printer2 ~ Append ~ PrintTemporary[
   " ** PoissonBracket: Finding kernel coefficients..."];
D0Term = BarPartialf1B BarVariationalf2BPi +
  2 BarPartialf1A BarVariationalf2APi -
  BarPartialf1BPi BarVariationalf2B -
  2 BarPartialf1APi BarVariationalf2A +
  ReplaceIndex[Evaluate[Partialf1DBz], z → t]
   ManualCovariantDerivative[-t, BarVariationalf2BPi, IndexList[-r], u] +
  2 ReplaceIndex[Evaluate[Partialf1DAz], z → t]
   ManualCovariantDerivative[-t, BarVariationalf2APi, IndexList[-s], u] -
  ReplaceIndex[Evaluate[Partialf1DBPiz], z → t]
   ManualCovariantDerivative[-t, BarVariationalf2B, IndexList[r], u] -
  2 ReplaceIndex[Evaluate[Partialf1DAPiz], z → t]
   ManualCovariantDerivative[-t, BarVariationalf2A, IndexList[s], u];
D1Term = (Partialf1DBPiz BarVariationalf2B +
    2 Partialf1DAPiz BarVariationalf2A -
    Partialf1DBz BarVariationalf2BPi -
    2 Partialf1DAz BarVariationalf2APi +
    BarPartialf1BPi Partialf2DBz +
    2 BarPartialf1APi Partialf2DAz -
    BarPartialf1B Partialf2DBPiz -
```

```
        2 BarPartialf1A Partialf2DAPiz +
        ReplaceIndex[Evaluate[Partialf1DBPiz], z → w]
         ManualCovariantDerivative[-w, Partialf2DBz, IndexList[z, r], u] +
        2 ReplaceIndex[Evaluate[Partialf1DAPiz], z → w]
         ManualCovariantDerivative[-w, Partialf2DAz, IndexList[z, s], u] -
        ReplaceIndex[Evaluate[Partialf1DBz], z → w]
         ManualCovariantDerivative[-w, Partialf2DBPiz, IndexList[z, -r], u] -
        2 ReplaceIndex[Evaluate[Partialf1DAz], z → w]
         ManualCovariantDerivative[-w, Partialf2DAPiz,
          IndexList[z, -s], u]) CD[-z][HComp[]];
      D2Term = Partialf1DBz ReplaceIndex[Evaluate[Partialf2DBPiz], z → w] +
        2 Partialf1DAz ReplaceIndex[Evaluate[Partialf2DAPiz], z → w] -
        Partialf1DBPiz ReplaceIndex[Evaluate[Partialf2DBz], z → w] -
        2 Partialf1DAPiz ReplaceIndex[Evaluate[Partialf2DAz], z → w];
      res = {D0Term, D1Term, D2Term};
      res = res /. InertDerRev;
      res = res /. Derivative3;
      NotebookDelete[printer2];
      res = ToNesterForm[#, "ToShell" → OptionValue["ToShell"],
          "Hard" → OptionValue["Hard"], "Order" → OptionValue["Order"],
          "GToFoliG" → OptionValue["GToFoliG"]] & /@ res;
      res = CollectTensors /@ res;
     }, {
      printer3 = {};
      printer3 = printer3 ~Append~ PrintTemporary[
         " ** PoissonBracket: Finding (old) kernel coefficients..."];
      DeltaDelta = Variationalf1B Variationalf2BPi +
        2 Variationalf1A Variationalf2APi - Variationalf1BPi Variationalf2B -
        2 Variationalf1APi Variationalf2A;
      DDeltaDelta = -Partialf1DBv Variationalf2BPi -
        2 Partialf1DAv Variationalf2APi + Partialf1DBPiv Variationalf2B +
        2 Partialf1DAPiv Variationalf2A;
      DeltaDDelta = -Variationalf1B Partialf2DBPiv -
        2 Variationalf1A Partialf2DAPiv + Variationalf1BPi Partialf2DBv +
        2 Variationalf1APi Partialf2DAv;
      DDeltaDDelta = Partialf1DBz Partialf2DBPiv + 2 Partialf1DAz Partialf2DAPiv -
        Partialf1DBPiz Partialf2DBv - 2 Partialf1DAPiz Partialf2DAv;
      res = {DeltaDelta, DDeltaDelta, DeltaDDelta, DDeltaDDelta};
      res = res /. InertDerRev;
      res = res /. Derivative3;
      NotebookDelete[printer3];
      If[OptionValue["NesterForm"], {
        res = ToNesterForm[#, "ToShell" → OptionValue["ToShell"],
```

```
                "Hard" → OptionValue["Hard"], "Order" → OptionValue["Order"],
                "GToFoliG" → OptionValue["GToFoliG"]] & /@ res;
          }, {
           res = ToBasicForm[res,
              "Hard" -> OptionValue["Hard"], "Order" → OptionValue["Order"]]];
          }];
         res = CollectTensors /@ res;
        }];
      NotebookDelete[printer];
      If[OptionValue["PrintAnswer"],
       If[OptionValue["ToShell"],
         HiGGSPrint[{f1x, f2x}, " ≈ ", res];,
         HiGGSPrint[{f1x, f2x}, " = ", res];
        ];
      ];
      res];
ClearBuild[];
```

build

Velocity

build

Inert commutators

build

```
(*copies of all the field strength tensors*)
DefTensor[RD[a, b, -d, -e, -x, -y, -z], M4,
  {Antisymmetric[{a, b}], Antisymmetric[{-d, -e}]}, PrintAs → "{ψ,R}δδ"];
DefTensor[RDS1[a, b, -d, -e, -x, -y, -z, v], M4,
  {Antisymmetric[{a, b}], Antisymmetric[{-d, -e}]}, PrintAs → "{ψ,R}∂δδ"];
DefTensor[RDS2[a, b, -d, -e, -x, -y, -z, v], M4,
  {Antisymmetric[{a, b}], Antisymmetric[{-d, -e}]}, PrintAs → "{ψ,R}δ ∂δ"];
DefTensor[RDS3[a, b, -d, -e, -x, -y, -z, v, w], M4,
  {Antisymmetric[{a, b}], Antisymmetric[{-d, -e}]}, PrintAs → "{ψ,R}∂δ ∂δ"];
ClearBuild[];
```

build

```
DefTensor[TD[a, -b, -c, -x, -y, -z],
  M4, Antisymmetric[{-b, -c}], PrintAs → "{ψ,T}δδ"];
DefTensor[TDS1[a, -b, -c, -x, -y, -z, v], M4,
  Antisymmetric[{-b, -c}], PrintAs → "{ψ,T}∂δδ"];
DefTensor[TDS2[a, -b, -c, -x, -y, -z, v], M4,
  Antisymmetric[{-b, -c}], PrintAs → "{ψ,T}δ ∂δ"];
DefTensor[TDS3[a, -b, -c, -x, -y, -z, v, w], M4,
  Antisymmetric[{-b, -c}], PrintAs → "{ψ,T}∂δ ∂δ"];
ClearBuild[];
```

build

```
(*copies of all the constraint functions*)
DefTensor[PhiDB0p[-x, -y, -z], M4, PrintAs → "{ψ,ϕb0⁺}δδ"];
DefTensor[PhiDS1B0p[-x, -y, -z, v], M4, PrintAs → "{ψ,ϕb0⁺}∂δδ"];
DefTensor[PhiDS2B0p[-x, -y, -z, v], M4, PrintAs → "{ψ,ϕb0⁺}δ ∂δ"];
DefTensor[PhiDS3B0p[-x, -y, -z, v, w], M4, PrintAs → "{ψ,ϕb0⁺}∂δ ∂δ"];
ClearBuild[];
```

build

```
DefTensor[PhiDB1p[-a, -b, -x, -y, -z],
  M4, Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕb1⁺}δδ"];
DefTensor[PhiDS1B1p[-a, -b, -x, -y, -z, v], M4,
  Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕb1⁺}∂δδ"];
DefTensor[PhiDS2B1p[-a, -b, -x, -y, -z, v], M4,
  Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕb1⁺}δ ∂δ"];
DefTensor[PhiDS3B1p[-a, -b, -x, -y, -z, v, w], M4,
  Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕb1⁺}∂δ ∂δ"];
ClearBuild[];
```

build

```
DefTensor[PhiDB1m[-a, -x, -y, -z], M4, PrintAs → "{ψ,ϕb1⁻}δδ"];
DefTensor[PhiDS1B1m[-a, -x, -y, -z, v], M4, PrintAs → "{ψ,ϕb1⁻}∂δδ"];
DefTensor[PhiDS2B1m[-a, -x, -y, -z, v], M4, PrintAs → "{ψ,ϕb1⁻}δ ∂δ"];
DefTensor[PhiDS3B1m[-a, -x, -y, -z, v, w], M4, PrintAs → "{ψ,ϕb1⁻}∂δ ∂δ"];
ClearBuild[];
```

build

```
DefTensor[PhiDB2p[-a, -b, -x, -y, -z],
  M4, Symmetric[{-a, -b}], PrintAs → "{ψ,ϕb2⁺}δδ"];
DefTensor[PhiDS1B2p[-a, -b, -x, -y, -z, v], M4,
  Symmetric[{-a, -b}], PrintAs → "{ψ,ϕb2⁺}∂δδ"];
DefTensor[PhiDS2B2p[-a, -b, -x, -y, -z, v], M4,
  Symmetric[{-a, -b}], PrintAs → "{ψ,ϕb2⁺}δ ∂δ"];
DefTensor[PhiDS3B2p[-a, -b, -x, -y, -z, v, w], M4,
  Symmetric[{-a, -b}], PrintAs → "{ψ,ϕb2⁺}∂δ ∂δ"];
ClearBuild[];
```

build

```
DefTensor[PhiDA0p[-x, -y, -z], M4, PrintAs → "{ψ,ϕA0⁺}δδ"];
DefTensor[PhiDS1A0p[-x, -y, -z, v], M4, PrintAs → "{ψ,ϕA0⁺}∂δδ"];
DefTensor[PhiDS2A0p[-x, -y, -z, v], M4, PrintAs → "{ψ,ϕA0⁺}δ ∂δ"];
DefTensor[PhiDS3A0p[-x, -y, -z, v, w], M4, PrintAs → "{ψ,ϕA0⁺}∂δ ∂δ"];
ClearBuild[];
```

build

```
DefTensor[PhiDA0m[-x, -y, -z], M4, PrintAs → "{ψ,ϕA0⁻}δδ"];
DefTensor[PhiDS1A0m[-x, -y, -z, v], M4, PrintAs → "{ψ,ϕA0⁻}∂δδ"];
DefTensor[PhiDS2A0m[-x, -y, -z, v], M4, PrintAs → "{ψ,ϕA0⁻}δ ∂δ"];
DefTensor[PhiDS3A0m[-x, -y, -z, v, w], M4, PrintAs → "{ψ,ϕA0⁻}∂δ ∂δ"];
ClearBuild[];
```

build

```
DefTensor[PhiDA1p[-a, -b, -x, -y, -z],
  M4, Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕA1⁺}δδ"];
DefTensor[PhiDS1A1p[-a, -b, -x, -y, -z, v], M4,
  Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕA1⁺}∂δδ"];
DefTensor[PhiDS2A1p[-a, -b, -x, -y, -z, v], M4,
  Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕA1⁺}δ ∂δ"];
DefTensor[PhiDS3A1p[-a, -b, -x, -y, -z, v, w], M4,
  Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕA1⁺}∂δ ∂δ"];
ClearBuild[];
```

build

```
DefTensor[PhiDA1m[-a, -x, -y, -z], M4, PrintAs → "{ψ,ϕA1⁻}δδ"];
DefTensor[PhiDS1A1m[-a, -x, -y, -z, v], M4, PrintAs → "{ψ,ϕA1⁻}∂δδ"];
DefTensor[PhiDS2A1m[-a, -x, -y, -z, v], M4, PrintAs → "{ψ,ϕA1⁻}δ ∂δ"];
DefTensor[PhiDS3A1m[-a, -x, -y, -z, v, w], M4, PrintAs → "{ψ,ϕA1⁻}∂δ ∂δ"];
ClearBuild[];
```

build

```
DefTensor[PhiDA2p[-a, -b, -x, -y, -z],
  M4, Symmetric[{-a, -b}], PrintAs → "{ψ,ϕA2⁺}δδ"];
DefTensor[PhiDS1A2p[-a, -b, -x, -y, -z, v], M4,
  Symmetric[{-a, -b}], PrintAs → "{ψ,ϕA2⁺}∂δδ"];
DefTensor[PhiDS2A2p[-a, -b, -x, -y, -z, v], M4,
  Symmetric[{-a, -b}], PrintAs → "{ψ,ϕA2⁺}δ ∂δ"];
DefTensor[PhiDS3A2p[-a, -b, -x, -y, -z, v, w], M4,
  Symmetric[{-a, -b}], PrintAs → "{ψ,ϕA2⁺}∂δ ∂δ"];
ClearBuild[];
```

build

```
DefTensor[PhiDA2m[-a, -b, -c, -x, -y, -z],
  M4, Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕA2⁻}δδ"];
DefTensor[PhiDS1A2m[-a, -b, -c, -x, -y, -z, v], M4,
  Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕA2⁻}∂δδ"];
DefTensor[PhiDS2A2m[-a, -b, -c, -x, -y, -z, v], M4,
  Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕA2⁻}δ ∂δ"];
DefTensor[PhiDS3A2m[-a, -b, -c, -x, -y, -z, v, w], M4,
  Antisymmetric[{-a, -b}], PrintAs → "{ψ,ϕA2⁻}∂δ ∂δ"];
ClearBuild[];
```

build

```
(*Apparently A2m is the only sector which
 requires extra attention beyond symmetry declarations*)
AutomaticRules[PhiDA2m, MakeRule[{PhiDA2m[a, -b, -a, -x, -y, -z], 0},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[PhiDS1A2m, MakeRule[{PhiDS1A2m[a, -b, -a, -x, -y, -z, v], 0},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[PhiDS2A2m, MakeRule[{PhiDS2A2m[a, -b, -a, -x, -y, -z, v], 0},
    MetricOn → All, ContractMetrics → True]];
AutomaticRules[PhiDS3A2m, MakeRule[{PhiDS3A2m[a, -b, -a, -x, -y, -z, v, w], 0},
    MetricOn → All, ContractMetrics → True]];
ClearBuild[];
```

build

```
(*This part set up to deal with final surface term*)
DefTensor[QD[-a, -y, -z], M4, PrintAs → "{ψ,-nᵛDα πᵥ^α}δδ"];
DefTensor[QDS1[-a, -y, -z, v], M4, PrintAs → "{ψ,-nᵛDα πᵥ^α}∂δδ"];
DefTensor[QDS2[-a, -y, -z, v], M4, PrintAs → "{ψ,-nᵛDα πᵥ^α}δ ∂δ"];
DefTensor[QDS3[-a, -y, -z, v, w], M4, PrintAs → "{ψ,-nᵛDα πᵥ^α}∂δ ∂δ"];
ClearBuild[];
```

build

```
(*This part to deal with the measure*)
DefTensor[JD[-a, -y, -z], M4, PrintAs → "{ψ,J}δδ"];
DefTensor[JDS1[-a, -y, -z, v], M4, PrintAs → "{ψ,J}∂δδ"];
DefTensor[JDS2[-a, -y, -z, v], M4, PrintAs → "{ψ,J}δ ∂δ"];
DefTensor[JDS3[-a, -y, -z, v, w], M4, PrintAs → "{ψ,J}∂δ ∂δ"];
ClearBuild[];
```

build

```
(*This part to deal with the lapse*)
DefTensor[LapseD[-a, -y, -z], M4, PrintAs → "{ψ,N}δδ"];
DefTensor[LapseDS1[-a, -y, -z, v], M4, PrintAs → "{ψ,N}∂δδ"];
DefTensor[LapseDS2[-a, -y, -z, v], M4, PrintAs → "{ψ,N}δ ∂δ"];
DefTensor[LapseDS3[-a, -y, -z, v, w], M4, PrintAs → "{ψ,N}∂δ ∂δ"];
ClearBuild[];
```

build

Placeholder vectors

build

```
(*The placeholder vectors*)
DefTensor[S1[-a], M4, PrintAs → "σ^1"];
DefTensor[S2[-a], M4, PrintAs → "σ^2"];
DefTensor[S3[-a], M4, PrintAs → "σ^3"];

StripPlaceholderVectors[Psi_, expr_] :=
  Module[{Temp, GradTemp, PsiFreeIndices, PsiFreeIndexList, PhiFreeIndexList,
    PsiFreeIndexListLength, PhiFreeIndexListString, PlaceholderVectors,
    DeltaList, PlaceholderBracketRules, PlaceholdersToDifferentiate,
    return, FreeConstraint, PlaceholderBracketActivate, ii},
   PsiFreeIndices = FindFreeIndices[Psi];
   PsiFreeIndexList =
    Developer`ToList[Delete[Map[ToString[#] &, PsiFreeIndices], 0]];
   PsiFreeIndexListLength = Length[PsiFreeIndexList];
   PlaceholderVectors = {"S1[-k]", "S2[-k]", "S3[-k]"};
   PlaceholdersToDifferentiate = {};
   For[ii = 1, ii < PsiFreeIndexListLength + 1, ii++, PlaceholdersToDifferentiate =
     Append[PlaceholdersToDifferentiate, ToExpression[StringReplace[
        PlaceholderVectors[[ii]], {"-k" → PsiFreeIndexList[[ii]]}]]]];
   HiGGSPrint[PlaceholdersToDifferentiate];
   Temp = expr;
   For[ii = 1, ii < PsiFreeIndexListLength + 1, ii++,
    Temp = NewVarAction[Temp, PlaceholdersToDifferentiate[[ii]]]];
   Temp = Temp // ToNewCanonical;
   HiGGSPrint[Temp];
   Temp];
ClearBuild[];
```

build

Velocity components

eV locity oT ggle

```
IfBuild["VelocityToggle",
  {$ConstraintHamiltonianBilinearB0p =
    2 (1/16) (cPerpB0p (1/BetPerpPerp0p) ShellOrigB0p
        (Lapse[] J[] PhiB0p[] PhiDB0p[-x, -y, -z] -
          CD[-v][Lapse[] J[] PhiB0p[] PhiDS1B0p[-x, -y, -z, v]] +
          CD[-v][Lapse[] J[] PhiB0p[]] PhiDS2B0p[-x, -y, -z, v] -
          CD[-w][CD[-v][Lapse[] J[] PhiB0p[]] PhiDS3B0p[-x, -y, -z, v, w]]) //
```

```
      ToCanonical // CollectTensors;

$ConstraintHamiltonianBilinearB1p =
    2 (1 / 16) (cPerpB1p (1 / BetPerpPerp1p) ShellOrigB1p
          ( Lapse[] J[] PhiB1p[-a, -b] PhiDB1p[a, b, -x, -y, -z] -
            CD[-v][Lapse[] J[] PhiB1p[-a, -b] PhiDS1B1p[a, b, -x, -y, -z, v]] +
            CD[-v][Lapse[] J[] PhiB1p[-a, -b]] PhiDS2B1p[a, b, -x, -y, -z, v] -
            CD[-w][CD[-v][Lapse[] J[] PhiB1p[-a, -b]] PhiDS3B1p[a, b,
               -x, -y, -z, v, w]])) // ToCanonical // CollectTensors;

$ConstraintHamiltonianBilinearB1m =
    2 (1 / 16) (cPerpB1m (1 / BetPerpPerp1m) ShellOrigB1m
          ( Lapse[] J[] PhiB1m[-a] PhiDB1m[a, -x, -y, -z] -
            CD[-v][Lapse[] J[] PhiB1m[-a] PhiDS1B1m[a, -x, -y, -z, v]] +
            CD[-v][Lapse[] J[] PhiB1m[-a]] PhiDS2B1m[a, -x, -y, -z, v] -
            CD[-w][CD[-v][Lapse[] J[] PhiB1m[-a]]
              PhiDS3B1m[a, -x, -y, -z, v, w]])) // ToCanonical // CollectTensors;

$ConstraintHamiltonianBilinearB2p =
    2 (1 / 16) (cPerpB2p (1 / BetPerpPerp2p) ShellOrigB2p
          ( Lapse[] J[] PhiB2p[-a, -b] PhiDB2p[a, b, -x, -y, -z] -
            CD[-v][Lapse[] J[] PhiB2p[-a, -b] PhiDS1B2p[a, b, -x, -y, -z, v]] +
            CD[-v][Lapse[] J[] PhiB2p[-a, -b]] PhiDS2B2p[a, b, -x, -y, -z, v] -
            CD[-w][CD[-v][Lapse[] J[] PhiB2p[-a, -b]] PhiDS3B2p[a, b,
               -x, -y, -z, v, w]])) // ToCanonical // CollectTensors;

   $ConstraintHamiltonianBilinearA0p =
    2 (1 / 16) ((1 / 4) (cPerpA0p (1 / AlpPerpPerp0p)
          ShellOrigA0p ( Lapse[] J[] PhiA0p[] PhiDA0p[-x, -y, -z] -
            CD[-v][Lapse[] J[] PhiA0p[] PhiDS1A0p[-x, -y, -z, v]] +
            CD[-v][Lapse[] J[] PhiA0p[]] PhiDS2A0p[-x, -y, -z, v] -
            CD[-w][CD[-v][Lapse[] J[] PhiA0p[]] PhiDS3A0p[-x,
               -y, -z, v, w]]))) // ToCanonical // CollectTensors;

   $ConstraintHamiltonianBilinearA0m =
    2 (1 / 16) ((1 / 4) (cPerpA0m (1 / AlpPerpPerp0m)
          ShellOrigA0m ( Lapse[] J[] PhiA0m[] PhiDA0m[-x, -y, -z] -
            CD[-v][Lapse[] J[] PhiA0m[] PhiDS1A0m[-x, -y, -z, v]] +
            CD[-v][Lapse[] J[] PhiA0m[]] PhiDS2A0m[-x, -y, -z, v] -
            CD[-w][CD[-v][Lapse[] J[] PhiA0m[]] PhiDS3A0m[-x,
               -y, -z, v, w]]))) // ToCanonical // CollectTensors;

   $ConstraintHamiltonianBilinearA1p =
```

```
 2 (1/16) ((1/4) (cPerpA1p (1/AlpPerpPerp1p) ShellOrigA1p
        ( Lapse[] J[] PhiA1p[-a, -b] PhiDA1p[a, b, -x, -y, -z] -
          CD[-v][Lapse[] J[] PhiA1p[-a, -b] PhiDS1A1p[a, b, -x, -y, -z, v]] +
          CD[-v][Lapse[] J[] PhiA1p[-a, -b]] PhiDS2A1p[a, b, -x, -y, -z, v] -
          CD[-w][CD[-v][Lapse[] J[] PhiA1p[-a, -b]] PhiDS3A1p[a, b,
             -x, -y, -z, v, w]]))) // ToCanonical // CollectTensors;

$ConstraintHamiltonianBilinearA1m =
 2 (1/16) ((1/4) (cPerpA1m (1/AlpPerpPerp1m) ShellOrigA1m
        ( Lapse[] J[] PhiA1m[-a] PhiDA1m[a, -x, -y, -z] -
          CD[-v][Lapse[] J[] PhiA1m[-a] PhiDS1A1m[a, -x, -y, -z, v]] +
          CD[-v][Lapse[] J[] PhiA1m[-a]] PhiDS2A1m[a, -x, -y, -z, v] -
          CD[-w][CD[-v][Lapse[] J[] PhiA1m[-a]] PhiDS3A1m[a, -x,
             -y, -z, v, w]])) // ToCanonical // CollectTensors;

$ConstraintHamiltonianBilinearA2p =
 2 (1/16) ((1/4) (cPerpA2p (1/AlpPerpPerp2p) ShellOrigA2p
        ( Lapse[] J[] PhiA2p[-a, -b] PhiDA2p[a, b, -x, -y, -z] -
          CD[-v][Lapse[] J[] PhiA2p[-a, -b] PhiDS1A2p[a, b, -x, -y, -z, v]] +
          CD[-v][Lapse[] J[] PhiA2p[-a, -b]] PhiDS2A2p[a, b, -x, -y, -z, v] -
          CD[-w][CD[-v][Lapse[] J[] PhiA2p[-a, -b]] PhiDS3A2p[a, b,
             -x, -y, -z, v, w]])) // ToCanonical // CollectTensors;

$ConstraintHamiltonianBilinearA2m =
 2 (1/16) ((1/4) (cPerpA2m (1/AlpPerpPerp2m) ShellOrigA2m
        ( Lapse[] J[] PhiA2m[-a, -b, -c] PhiDA2m[a, b, c, -x, -y, -z] -
          CD[-v][Lapse[] J[] PhiA2m[-a, -b, -c] PhiDS1A2m[
             a, b, c, -x, -y, -z, v]] +
          CD[-v][Lapse[] J[] PhiA2m[-a, -b, -c]]
           PhiDS2A2m[a, b, c, -x, -y, -z, v] -
          CD[-w][CD[-v][Lapse[] J[] PhiA2m[-a, -b, -c]] PhiDS3A2m[a,
             b, c, -x, -y, -z, v, w]])) // ToCanonical // CollectTensors;

(*PBs for field strength tensors and ADM projectors, remember PBs
 vanish on main field strength projectors as only functions of G*)
$LagrangianHamiltonianBilinearT = -2 ( Lapse[] J[] T[i, -m, -n]
       PPara[m, g] PPara[n, h] (Bet1 PT1[-i, -g, -h, a, c, d] +
         Bet2 PT2[-i, -g, -h, a, c, d] +
         Bet3 PT3[-i, -g, -h, a, c, d]) TD[-a, -c, -d, -x, -y, -z] -
      CD[-v][Lapse[] J[] T[i, -m, -n]
        PPara[m, g] PPara[n, h] (Bet1 PT1[-i, -g, -h, a, c, d] +
          Bet2 PT2[-i, -g, -h, a, c, d] +
          Bet3 PT3[-i, -g, -h, a, c, d]) TDS1[-a, -c, -d, -x, -y, -z, v]] +
```

```
        CD[-v][Lapse[] J[] T[i, -m, -n]
           PPara[m, g] PPara[n, h] (Bet1 PT1[-i, -g, -h, a, c, d] +
             Bet2 PT2[-i, -g, -h, a, c, d] +
             Bet3 PT3[-i, -g, -h, a, c, d])] TDS2[-a, -c, -d, -x, -y, -z, v] -
        CD[-w][CD[-v][Lapse[] J[] T[i, -m, -n]
            PPara[m, g] PPara[n, h] (Bet1 PT1[-i, -g, -h, a, c, d] +
              Bet2 PT2[-i, -g, -h, a, c, d] +
              Bet3 PT3[-i, -g, -h, a, c, d])]
          TDS3[-a, -c, -d, -x, -y, -z, v, w]]) // ToCanonical // CollectTensors;

(*PBs for field strength tensors and ADM projectors, remember PBs
 vanish on main field strength projectors as only functions of G*)
$LagrangianHamiltonianBilinearR =
 -2 ( Lapse[] J[] (R[i, j, -m, -n] PPara[m, g] PPara[n, h]
           (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] +
             Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
             Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
             Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
             Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
             Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) -
          (1 / 4) Alp0 PPara[a, c] PPara[b, d]) RD[-a, -b, -c, -d, -x, -y, -z] -
        CD[-v][Lapse[] J[] (R[i, j, -m, -n] PPara[m, g] PPara[n, h]
             (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] +
               Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
               Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
               Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
               Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
               Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) - (1 / 4) Alp0 PPara[a, c]
             PPara[b, d]) RDS1[-a, -b, -c, -d, -x, -y, -z, v]] +
        CD[-v][Lapse[] J[] (R[i, j, -m, -n] PPara[m, g] PPara[n,
               h] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] +
                Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
                Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
                Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
                Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
                Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) - (1 / 4) Alp0 PPara[a,
               c] PPara[b, d])] RDS2[-a, -b, -c, -d, -x, -y, -z, v] -
        CD[-w][CD[-v][Lapse[] J[] (R[i, j, -m, -n] PPara[m, g]
               PPara[n, h] (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] +
                 Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
                 Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
                 Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
                 Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
```

```
                Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) - (1 / 4)
              Alp0 PPara[a, c] PPara[b, d])] RDS3[-a, -b, -c,
          -d, -x, -y, -z, v, w]]) // ToCanonical // CollectTensors;

$LagrangianHamiltonianBilinearMultiplierT =
 -( Lapse[] J[] TLambda[i, g, h] (cBet1 PT1[-i, -g, -h, a, c, d] +
         cBet2 PT2[-i, -g, -h, a, c, d] +
         cBet3 PT3[-i, -g, -h, a, c, d]) TD[-a, -c, -d, -x, -y, -z] -
       CD[-v][Lapse[] J[] TLambda[i, g, h] (cBet1 PT1[-i, -g, -h, a, c, d] +
          cBet2 PT2[-i, -g, -h, a, c, d] +
          cBet3 PT3[-i, -g, -h, a, c, d]) TDS1[-a, -c, -d, -x, -y, -z, v]] +
       CD[-v][Lapse[] J[] TLambda[i, g, h] (cBet1 PT1[-i, -g, -h, a, c, d] +
           cBet2 PT2[-i, -g, -h, a, c, d] +
           cBet3 PT3[-i, -g, -h, a, c, d])] TDS2[-a, -c, -d, -x, -y, -z, v] -
       CD[-w][CD[-v][Lapse[] J[] TLambda[i, -m, -n]
           PPara[m, g] PPara[n, h] (cBet1 PT1[-i, -g, -h, a, c, d] +
             cBet2 PT2[-i, -g, -h, a, c, d] +
             cBet3 PT3[-i, -g, -h, a, c, d])]
         TDS3[-a, -c, -d, -x, -y, -z, v, w]]) // ToCanonical // CollectTensors;

$LagrangianHamiltonianBilinearMultiplierR =
 -( Lapse[] J[] RLambda[i, j, -m, -n] PPara[m, g]
        PPara[n, h] (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
          cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
          cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] +
          cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
          cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] +
          cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
        RD[-a, -b, -c, -d, -x, -y, -z] -
       CD[-v][Lapse[] J[] RLambda[i, j, -m, -n] PPara[m, g]
         PPara[n, h] (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
           cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
           cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] +
           cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
           cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] +
           cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
         RDS1[-a, -b, -c, -d, -x, -y, -z, v]] +
       CD[-v][Lapse[] J[] RLambda[i, j, -m, -n] PPara[m, g]
          PPara[n, h] (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
            cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
            cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] +
            cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
            cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] +
```

```
            cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])]
        RDS2[-a, -b, -c, -d, -x, -y, -z, v] -
       CD[-w][CD[-v][Lapse[] J[] RLambda[i, j, -m, -n] PPara[m, g]
           PPara[n, h] (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
             cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
             cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] +
             cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
             cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] +
             cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])] RDS3[-a, -b, -c,
          -d, -x, -y, -z, v, w]]) // ToCanonical // CollectTensors;

(*PB on the measure factor in front of constraint and Lagrangian parts*)
$ConstraintLagrangianMeasure1 =
 ((1/16) (cPerpB0p (1/BetPerpPerp0p) ShellOrigB0p PhiB0p[] PhiB0p[] +
       cPerpB1p (1/BetPerpPerp1p) ShellOrigB1p PhiB1p[-a, -b] PhiB1p[a, b] +
       cPerpB1m (1/BetPerpPerp1m) ShellOrigB1m PhiB1m[-a] PhiB1m[a] +
       cPerpB2p (1/BetPerpPerp2p) ShellOrigB2p PhiB2p[-a, -b] PhiB2p[a, b] +
       (1/4) (cPerpA0p (1/AlpPerpPerp0p) ShellOrigA0p PhiA0p[] PhiA0p[] +
          cPerpA0m (1/AlpPerpPerp0m) ShellOrigA0m PhiA0m[] PhiA0m[] +
          cPerpA1p (1/AlpPerpPerp1p)
           ShellOrigA1p PhiA1p[-a, -b] PhiA1p[a, b] +
          cPerpA1m (1/AlpPerpPerp1m) ShellOrigA1m PhiA1m[-a] PhiA1m[a] +
          cPerpA2p (1/AlpPerpPerp2p)
           ShellOrigA2p PhiA2p[-a, -b] PhiA2p[a, b] +
          cPerpA2m (1/AlpPerpPerp2m) ShellOrigA2m
           PhiA2m[-a, -b, -c] PhiA2m[a, b, c])) -
    (T[i, -m, -n] PPara[m, g] PPara[n, h] (Bet1 PT1[-i, -g, -h, a, c, d] +
         Bet2 PT2[-i, -g, -h, a, c, d] +
         Bet3 PT3[-i, -g, -h, a, c, d])
      PPara[-c, p] PPara[-d, q] T[-a, -p, -q] +
     TLambda[i, -m, -n] PPara[m, g] PPara[n, h]
      (cBet1 PT1[-i, -g, -h, a, c, d] +
        cBet2 PT2[-i, -g, -h, a, c, d] +
        cBet3 PT3[-i, -g, -h, a, c, d])
      PPara[-c, p] PPara[-d, q] T[-a, -p, -q] +
     (R[i, j, -m, -n] PPara[m, g] PPara[n, h]
         (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] +
           Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
           Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
           Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
           Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
           Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) - (1/2) Alp0 PPara[a, c]
        PPara[b, d]) PPara[-c, p] PPara[-d, q] R[-a, -b, -p, -q] +
```

```
        RLambda[i, j, -m, -n] PPara[m, g] PPara[n, h]
         (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
           cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
           cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] +
           cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
           cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] +
           cAlp6 PR6[-i, -j, -g, -h, a, b, c, d])
         PPara[-c, p] PPara[-d, q] R[-a, -b, -p, -q])) JD[-x, -y, -z];

$ConstraintLagrangianMeasure2 =
 - CD[-v][((1/16) (cPerpB0p (1/BetPerpPerp0p) ShellOrigB0p PhiB0p[] PhiB0p[] +
          cPerpB1p (1/BetPerpPerp1p) ShellOrigB1p
           PhiB1p[-a, -b] PhiB1p[a, b] +
          cPerpB1m (1/BetPerpPerp1m) ShellOrigB1m PhiB1m[-a] PhiB1m[a] +
          cPerpB2p (1/BetPerpPerp2p)
           ShellOrigB2p PhiB2p[-a, -b] PhiB2p[a, b] +
          (1/4) (cPerpA0p (1/AlpPerpPerp0p) ShellOrigA0p PhiA0p[] PhiA0p[] +
             cPerpA0m (1/AlpPerpPerp0m) ShellOrigA0m PhiA0m[] PhiA0m[] +
             cPerpA1p (1/AlpPerpPerp1p)
              ShellOrigA1p PhiA1p[-a, -b] PhiA1p[a, b] +
             cPerpA1m (1/AlpPerpPerp1m) ShellOrigA1m PhiA1m[-a] PhiA1m[a] +
             cPerpA2p (1/AlpPerpPerp2p)
              ShellOrigA2p PhiA2p[-a, -b] PhiA2p[a, b] +
             cPerpA2m (1/AlpPerpPerp2m) ShellOrigA2m
              PhiA2m[-a, -b, -c] PhiA2m[a, b, c])) -
       (T[i, -m, -n] PPara[m, g] PPara[n, h] (Bet1 PT1[-i, -g, -h, a, c, d] +
            Bet2 PT2[-i, -g, -h, a, c, d] +
            Bet3 PT3[-i, -g, -h, a, c, d])
          PPara[-c, p] PPara[-d, q] T[-a, -p, -q] +
         TLambda[i, -m, -n] PPara[m, g] PPara[n, h]
          (cBet1 PT1[-i, -g, -h, a, c, d] +
            cBet2 PT2[-i, -g, -h, a, c, d] +
            cBet3 PT3[-i, -g, -h, a, c, d])
          PPara[-c, p] PPara[-d, q] T[-a, -p, -q] +
         (R[i, j, -m, -n] PPara[m, g] PPara[n, h]
             (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] +
               Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
               Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
               Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
               Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
               Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) - (1/2) Alp0 PPara[a,
            c] PPara[b, d]) PPara[-c, p] PPara[-d, q] R[-a, -b, -p, -q] +
         RLambda[i, j, -m, -n] PPara[m, g] PPara[n, h]
```

```
          (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
            cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
            cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] +
            cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
            cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] +
            cAlp6 PR6[-i, -j, -g, -h, a, b, c, d]) PPara[-c, p]
           PPara[-d, q] R[-a, -b, -p, -q])) JDS1[-x, -y, -z, v]];

$ConstraintLagrangianMeasure3 =
 CD[-v][((1/16) (cPerpB0p (1/BetPerpPerp0p) ShellOrigB0p PhiB0p[] PhiB0p[] +
         cPerpB1p (1/BetPerpPerp1p) ShellOrigB1p PhiB1p[-a, -b] PhiB1p[a, b] +
         cPerpB1m (1/BetPerpPerp1m) ShellOrigB1m PhiB1m[-a] PhiB1m[a] +
         cPerpB2p (1/BetPerpPerp2p) ShellOrigB2p PhiB2p[-a, -b] PhiB2p[a, b] +
         (1/4) (cPerpA0p (1/AlpPerpPerp0p) ShellOrigA0p PhiA0p[] PhiA0p[] +
            cPerpA0m (1/AlpPerpPerp0m) ShellOrigA0m PhiA0m[] PhiA0m[] +
            cPerpA1p (1/AlpPerpPerp1p)
             ShellOrigA1p PhiA1p[-a, -b] PhiA1p[a, b] +
            cPerpA1m (1/AlpPerpPerp1m) ShellOrigA1m PhiA1m[-a] PhiA1m[a] +
            cPerpA2p (1/AlpPerpPerp2p)
             ShellOrigA2p PhiA2p[-a, -b] PhiA2p[a, b] +
            cPerpA2m (1/AlpPerpPerp2m) ShellOrigA2m PhiA2m[
              -a, -b, -c] PhiA2m[a, b, c])) -
       (T[i, -m, -n] PPara[m, g] PPara[n, h] (Bet1 PT1[-i, -g, -h, a, c, d] +
            Bet2 PT2[-i, -g, -h, a, c, d] +
            Bet3 PT3[-i, -g, -h, a, c, d])
          PPara[-c, p] PPara[-d, q] T[-a, -p, -q] +
         TLambda[i, -m, -n] PPara[m, g] PPara[n, h]
          (cBet1 PT1[-i, -g, -h, a, c, d] +
            cBet2 PT2[-i, -g, -h, a, c, d] +
            cBet3 PT3[-i, -g, -h, a, c, d])
          PPara[-c, p] PPara[-d, q] T[-a, -p, -q] +
         (R[i, j, -m, -n] PPara[m, g] PPara[n, h]
             (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] +
               Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
               Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
               Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
               Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
               Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) - (1/2) Alp0 PPara[a, c]
            PPara[b, d]) PPara[-c, p] PPara[-d, q] R[-a, -b, -p, -q] +
         RLambda[i, j, -m, -n] PPara[m, g] PPara[n, h]
          (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
            cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
            cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] +
```

```
            cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
            cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] +
            cAlp6 PR6[-i, -j, -g, -h, a, b, c, d]) PPara[-c, p]
          PPara[-d, q] R[-a, -b, -p, -q]))] JDS2[-x, -y, -z, v];


$ConstraintLagrangianMeasure4 = - CD[-w][CD[-v][
      ((1/16) (cPerpB0p (1/BetPerpPerp0p) ShellOrigB0p PhiB0p[] PhiB0p[] +
            cPerpB1p (1/BetPerpPerp1p) ShellOrigB1p
             PhiB1p[-a, -b] PhiB1p[a, b] +
            cPerpB1m (1/BetPerpPerp1m) ShellOrigB1m PhiB1m[-a] PhiB1m[a] +
            cPerpB2p (1/BetPerpPerp2p)
             ShellOrigB2p PhiB2p[-a, -b] PhiB2p[a, b] +
            (1/4) (cPerpA0p (1/AlpPerpPerp0p) ShellOrigA0p PhiA0p[] PhiA0p[] +
               cPerpA0m (1/AlpPerpPerp0m) ShellOrigA0m PhiA0m[] PhiA0m[] +
               cPerpA1p (1/AlpPerpPerp1p)
                ShellOrigA1p PhiA1p[-a, -b] PhiA1p[a, b] +
               cPerpA1m (1/AlpPerpPerp1m) ShellOrigA1m PhiA1m[-a] PhiA1m[a] +
               cPerpA2p (1/AlpPerpPerp2p)
                ShellOrigA2p PhiA2p[-a, -b] PhiA2p[a, b] +
               cPerpA2m (1/AlpPerpPerp2m) ShellOrigA2m
                PhiA2m[-a, -b, -c] PhiA2m[a, b, c])) -
         (T[i, -m, -n] PPara[m, g] PPara[n, h] (Bet1 PT1[-i, -g, -h, a, c, d] +
               Bet2 PT2[-i, -g, -h, a, c, d] +
               Bet3 PT3[-i, -g, -h, a, c, d])
            PPara[-c, p] PPara[-d, q] T[-a, -p, -q] +
           TLambda[i, -m, -n] PPara[m, g] PPara[n, h]
            (cBet1 PT1[-i, -g, -h, a, c, d] +
              cBet2 PT2[-i, -g, -h, a, c, d] +
              cBet3 PT3[-i, -g, -h, a, c, d])
            PPara[-c, p] PPara[-d, q] T[-a, -p, -q] +
           (R[i, j, -m, -n] PPara[m, g] PPara[n, h]
               (Alp1 PR1[-i, -j, -g, -h, a, b, c, d] +
                 Alp2 PR2[-i, -j, -g, -h, a, b, c, d] +
                 Alp3 PR3[-i, -j, -g, -h, a, b, c, d] +
                 Alp4 PR4[-i, -j, -g, -h, a, b, c, d] +
                 Alp5 PR5[-i, -j, -g, -h, a, b, c, d] +
                 Alp6 PR6[-i, -j, -g, -h, a, b, c, d]) - (1/2) Alp0 PPara[a, c]
              PPara[b, d]) PPara[-c, p] PPara[-d, q] R[-a, -b, -p, -q] +
          RLambda[i, j, -m, -n] PPara[m, g] PPara[n, h]
           (cAlp1 PR1[-i, -j, -g, -h, a, b, c, d] +
             cAlp2 PR2[-i, -j, -g, -h, a, b, c, d] +
             cAlp3 PR3[-i, -j, -g, -h, a, b, c, d] +
```

```
                cAlp4 PR4[-i, -j, -g, -h, a, b, c, d] +
                cAlp5 PR5[-i, -j, -g, -h, a, b, c, d] +
                cAlp6 PR6[-i, -j, -g, -h, a, b, c, d]) PPara[-c, p]
              PPara[-d, q] R[-a, -b, -p, -q]))] JDS3[-x, -y, -z, v, w];

  (*PB on final surface term*)
  $SurfaceHamiltonian = Lapse[] QD[-x, -y, -z] -
    CD[-v][Lapse[] QDS1[-x, -y, -z, v]] +
    CD[-v][Lapse[]] QDS2[-x, -y, -z, v] -
    CD[-j][CD[-v][Lapse[]] QDS3[-x, -y, -z, v, j]] -
    (V[k] G3[m, -n]
       (CD[-m][BPi[-k, n]] - A[w, -k, -m] BPi[-w, n]) LapseD[-x, -y, -z] -
      CD[-v][V[k] G3[m, -n] (CD[-m][BPi[-k, n]] - A[w, -k, -m] BPi[-w, n])
        LapseDS1[-x, -y, -z, v]] +
      CD[-v][V[k] G3[m, -n] (CD[-m][BPi[-k, n]] - A[w, -k, -m] BPi[-w, n])]
       LapseDS2[-x, -y, -z, v] -
      CD[-j][CD[-v][V[k] G3[m, -n] (CD[-m][BPi[-k, n]] - A[w, -k, -m] BPi[-w, n])]
        LapseDS3[-x, -y, -z, v, j]]);

  $SurfaceHamiltonian = $SurfaceHamiltonian // ToNewCanonical;
  $SurfaceHamiltonian = $SurfaceHamiltonian // ToNewCanonical;
  $SurfaceHamiltonian = $SurfaceHamiltonian /. PActivate // ToNewCanonical;
  $SurfaceHamiltonian = $SurfaceHamiltonian /. PADMActivate // ToNewCanonical;

  DumpSave[BinaryLocation["VelocityToggle"],
   {$ConstraintHamiltonianBilinearB0p, $ConstraintHamiltonianBilinearB1p,
    $ConstraintHamiltonianBilinearB1m, $ConstraintHamiltonianBilinearB2p,
    $ConstraintHamiltonianBilinearA0p, $ConstraintHamiltonianBilinearA0m,
    $ConstraintHamiltonianBilinearA1p, $ConstraintHamiltonianBilinearA1m,
    $ConstraintHamiltonianBilinearA2p, $ConstraintHamiltonianBilinearA2m,
    $LagrangianHamiltonianBilinearT, $LagrangianHamiltonianBilinearR,
    $LagrangianHamiltonianBilinearMultiplierT,
    $LagrangianHamiltonianBilinearMultiplierR, $ConstraintLagrangianMeasure1,
    $ConstraintLagrangianMeasure2, $ConstraintLagrangianMeasure3,
    $ConstraintLagrangianMeasure4, $SurfaceHamiltonian}];
  ClearBuild[];}
];
```

build

In[•]:=

```
OpenLastCache[];
```

build

```
VelSimplifier[xx_] := Module[{res, printer},
  res = xx;
```

```
    (*a message*)
    printer = {};
    printer = printer~Append~PrintTemporary[" ** DefInertVelocity..."];

    res = res // ToNewCanonical;
    res = res /. TocPerp;
    res = res /. ToAlp;
    res = res /. ToBet;
    (*The order here will be very important,
    you must kill off constrained terms with 0 before the
     mu function on the transfer couplings give you 1/0*)
    res = res /. $ToShellFreedoms;
    res = res // ToNewCanonical;
    (*if you simplify, you increase chance of killing bad terms...*)
    res = res /. $ToTheory;
    res = res // ToNewCanonical;
    res = res /. PActivate // ToNewCanonical;
    res = res /. PADMActivate // ToNewCanonical;
    res = ReplaceDummies[res, IndexList[l, n, m, p, q, r, s, t, u, v, w]];
    res = res S1[x] S2[y] S3[z] // ToNewCanonical;
    HiGGSPrint[res];
    NotebookDelete[printer];
    res];

DefInertVelocity[$ToShellFreedoms_, $ToTheory_, $Theory_] :=
  Module[{printer, Jobs, SegmentList},
    (*a message*)
    xAct`xTensor`Private`MakeDefInfo[
     DefTheory, $Theory, {"inert velocity for the theory", ""}];
    printer = {};

    $InertVelocity = {};
    (**)
    SegmentList = {$ConstraintHamiltonianBilinearB0p,
      $ConstraintHamiltonianBilinearB1p, $ConstraintHamiltonianBilinearB1m,
      $ConstraintHamiltonianBilinearB2p, $ConstraintHamiltonianBilinearA0p,
      $ConstraintHamiltonianBilinearA0m, $ConstraintHamiltonianBilinearA1p,
      $ConstraintHamiltonianBilinearA1m, $ConstraintHamiltonianBilinearA2p,
      $ConstraintHamiltonianBilinearA2m, $LagrangianHamiltonianBilinearT,
      $LagrangianHamiltonianBilinearR, $LagrangianHamiltonianBilinearMultiplierT,
      $LagrangianHamiltonianBilinearMultiplierR, $ConstraintLagrangianMeasure1,
```

```
      $ConstraintLagrangianMeasure2, $ConstraintLagrangianMeasure3,
      $ConstraintLagrangianMeasure4, $SurfaceHamiltonian};
    (*
    (*Large batch of jobs for segments of all velocities*)
    Jobs=ParallelSubmit@VelSimplifier/@SegmentList;
    $InertVelocity=WaitAll[Jobs];
    *)
    $InertVelocity = VelSimplifier /@ SegmentList;
    NotebookDelete[printer];
    Print["made it final"];
    (**)
    $InertVelocity];
ClearBuild[];
```

build

Velocity

build

```
InertMakeRule[Replacement_List] :=
  MakeRule[Replacement, MetricOn → All, ContractMetrics → True];
```

build

Riemann bracket

build

```
RiemannBracketParallel[Psi_, EH0_,
    PsiFreeIndexListNormal_, $TheoryName_ : $TheoryName] :=
  Module[{Temp, GradTemp, PlaceholderBracketActivate,
    printer, PsiInert, PlaceholderBracketRulesInert,
    EH0Inert, PsiFreeIndexListNormalInert, result},
   (*Build the HiGGS environment*)
   (*$Timing=True;*)
   BuildHiGGS[];
   (*import theory names*)
   Quiet@ToExpression["<<" <> FileNameJoin@
       {$WorkingDirectory, "svy", "node-" <> $Node, "peta4.nom.mx"} <> ";"];
   (*Define the theory*)
   DefTheory["Import" -> $TheoryName];
   PsiInert = ToString@Psi;
   EH0Inert = ToString@EH0;
   PsiFreeIndexListNormalInert = ToString@PsiFreeIndexListNormal;
   result = ToExpression@("RiemannBracket[" <> PsiInert <>
       "," <> EH0Inert <> "," <> PsiFreeIndexListNormalInert <> "]");
   ForceTiming[];
   result];
ClearBuild[];
```

build

```
RiemannBracket[Psi_, EH0_, PsiFreeIndexListNormal_] :=
  Module[{Temp, GradTemp, PlaceholderBracketActivate, printer,
    PsiFreeIndexListD, PsiFreeIndexListDLength, PlaceholderVectors,
    DeltaList, zz, PlaceholderBracketRules, VelocitySegments},
   printer = {};
   PlaceholderBracketActivate = {};

   Print@$InertVelocity;
   Print@"riem";


   PsiFreeIndexListD = Map[ToString[#] &, PsiFreeIndexListNormal];
   PsiFreeIndexListDLength = Length[PsiFreeIndexListD];
   PlaceholderVectors = {"S1[x1]", "S2[y1]", "S3[z1]"};
   DeltaList = {"G[x1,-k]", "G[y1,-k]", "G[z1,-k]"};
   PlaceholderBracketRules = {};
   For[zz = 1, zz < PsiFreeIndexListDLength + 1, zz++, PlaceholderBracketRules =
```

```
  Append[PlaceholderBracketRules, PlaceholderVectors[[zz]] →
    StringReplace[DeltaList[[zz]], {"-k" → PsiFreeIndexListD[[zz]]}]]];

printer =
 printer~Append~PrintTemporary[" ** PoissonBracket: Riemann bracket..."];
Temp = PoissonBracket[Psi, PPara[-i, e] PPara[-j, f] R[-g, -h, -e, -f],
  "ToShell" → True, "Hard" → True, "Surficial" → False, "Order" → EH0,
  "GToFoliG" → False, "NesterForm" → False, "PrintAnswer" -> False];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)δ(x-x2):"];
printer = printer~Append~PrintTemporary[Evaluate[ToExpression[
     StringReplace["RD[-g,-h,-i,-j,-x1,-y1,-z1]S1[x1]S2[y1]S3[z1]",
      PlaceholderBracketRules]]]];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
      StringReplace["RD[-g,-h,-i,-j,-x1,-y1,-z1]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]], Evaluate[Temp[[1]]]}]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)δ(x-x2):"];
printer = printer~Append~PrintTemporary[Evaluate[ToExpression[
     StringReplace["RDS1[-g,-h,-i,-j,-x1,-y1,-z1,z]S1[x1]S2[y1]S3[z1]",
      PlaceholderBracketRules]]]];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
      StringReplace["RDS1[-g,-h,-i,-j,-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[Temp[[2]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)∂δ(x-x2):"];
printer = printer~Append~PrintTemporary[Evaluate[ToExpression[
     StringReplace["RDS2[-g,-h,-i,-j,-x1,-y1,-z1,z]S1[x1]S2[y1]S3[z1]",
      PlaceholderBracketRules]]]];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
      StringReplace["RDS2[-g,-h,-i,-j,-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[Temp[[3]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)∂δ(x-x2):"];
printer = printer~Append~PrintTemporary[Evaluate[ToExpression[
     StringReplace["RDS3[-g,-h,-i,-j,-x1,-y1,-z1,v,z]S1[x1]S2[y1]S3[z1]",
      PlaceholderBracketRules]]]];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
```

```
      StringReplace["RDS3[-g,-h,-i,-j,-x1,-y1,-z1,v,z]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[Temp[[4]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)δ(x-x2):"];
printer = printer~Append~PrintTemporary[Evaluate[ToExpression[StringReplace[
      "CD[-u][RDS1[-g,-h,-i,-j,-x1,-y1,-z1,z]]S1[x1]S2[y1]S3[z1]",
      PlaceholderBracketRules]]]];
GradTemp = CD[-u][Evaluate[Temp[[2]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → EH0];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→EH0,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace[
       "CD[-u][RDS1[-g,-h,-i,-j,-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of δ(x-x1)∂δ(x-x2):"];
printer = printer~Append~PrintTemporary[Evaluate[ToExpression[StringReplace[
      "CD[-u][RDS2[-g,-h,-i,-j,-x1,-y1,-z1,z]]S1[x1]S2[y1]S3[z1]",
      PlaceholderBracketRules]]]];
GradTemp = CD[-u][Evaluate[Temp[[3]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → EH0];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→EH0,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace[
       "CD[-u][RDS2[-g,-h,-i,-j,-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)∂δ(x-x2):"];
printer = printer~Append~PrintTemporary[Evaluate[ToExpression[StringReplace[
      "CD[-u][RDS3[-g,-h,-i,-j,-x1,-y1,-z1,v,z]]S1[x1]S2[y1]S3[z1]",
      PlaceholderBracketRules]]]];
GradTemp = CD[-u][Evaluate[Temp[[4]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → EH0];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→EH0,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
```

```
      MakeRule[{Evaluate[ToExpression[StringReplace[
           "CD[-u][RDS3[-g,-h,-i,-j,-x1,-y1,-z1,v,z]]S1[x1]S2[y1]S3[z1]",
           PlaceholderBracketRules]]],
        Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];

   (*Now we calculate the commutator replacement rule part*)
   VelocitySegments = $InertVelocity[[{12, 14}]];
   VelocitySegments =
    ImposeCommutatorReplacementRules[PlaceholderBracketActivate,
       #, PsiFreeIndexListNormal] & /@ VelocitySegments;

   NotebookDelete[printer];
   VelocitySegments];
DistributeDefinitions@RiemannBracketParallel;
ClearBuild[];
```

build

Torsion bracket

build

```
TorsionBracketParallel[Psi_, EH0_,
   PsiFreeIndexListNormal_, $TheoryName_ : $TheoryName] :=
  Module[{Temp, GradTemp, PlaceholderBracketActivate,
    printer, PsiInert, PlaceholderBracketRulesInert,
    EH0Inert, PsiFreeIndexListNormalInert, result},
   (*Build the HiGGS environment*)
   (*$Timing=True;*)
   BuildHiGGS[];
   (*import theory names*)
   Quiet@ToExpression["<<" <> FileNameJoin@
       {$WorkingDirectory, "svy", "node-" <> $Node, "peta4.nom.mx"} <> ";"];
   (*Define the theory*)
   DefTheory["Import" -> $TheoryName];
   PsiInert = ToString@Psi;
   EH0Inert = ToString@EH0;
   PsiFreeIndexListNormalInert = ToString@PsiFreeIndexListNormal;
   result = ToExpression@("TorsionBracket[" <> PsiInert <>
       "," <> EH0Inert <> "," <> PsiFreeIndexListNormalInert <> "]");
   ForceTiming[];
   result];
ClearBuild[];
```

build

```
TorsionBracket[Psi_, EH0_, PsiFreeIndexListNormal_] :=
```

```
Module[{Temp, GradTemp, PlaceholderBracketActivate, printer,
  PsiFreeIndexListD, PsiFreeIndexListDLength, PlaceholderVectors,
  DeltaList, zz, PlaceholderBracketRules, VelocitySegments},
 printer = {};
 PlaceholderBracketActivate = {};

 Print@$InertVelocity;
 Print@"tors";


 PsiFreeIndexListD = Map[ToString[#] &, PsiFreeIndexListNormal];
 HiGGSPrint[PsiFreeIndexListD];
 PsiFreeIndexListDLength = Length[PsiFreeIndexListD];
 HiGGSPrint[PsiFreeIndexListDLength];
 PlaceholderVectors = {"S1[x1]", "S2[y1]", "S3[z1]"};
 HiGGSPrint[PlaceholderVectors];
 DeltaList = {"G[x1,-k]", "G[y1,-k]", "G[z1,-k]"};
 HiGGSPrint[DeltaList];
 PlaceholderBracketRules = {};
 HiGGSPrint[PlaceholderBracketRules];
 For[zz = 1, zz < PsiFreeIndexListDLength + 1, zz++, PlaceholderBracketRules =
   Append[PlaceholderBracketRules, PlaceholderVectors[[zz]] →
     StringReplace[DeltaList[[zz]], {"-k" → PsiFreeIndexListD[[zz]]}]]];
 HiGGSPrint[PlaceholderBracketRules];

 printer =
  printer~Append~PrintTemporary[" ** PoissonBracket: Torsion bracket..."];
 Temp = PoissonBracket[Psi, PPara[-g, e] PPara[-h, f] T[-d, -e, -f],
   "ToShell" → True, "Hard" → True, "Surficial" → False, "Order" → EH0,
   "GToFoliG" → False, "NesterForm" → False, "PrintAnswer" -> False];
 printer = printer~Append~PrintTemporary[
    " ** PoissonBracket: Rule for coefficient of δ(x-x1)δ(x-x2):"];
 printer = printer~Append~PrintTemporary[Evaluate[
     ToExpression[StringReplace["TD[-d,-g,-h,-x1,-y1,-z1]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]]];
 PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
   MakeRule[{Evaluate[ToExpression[
       StringReplace["TD[-d,-g,-h,-x1,-y1,-z1]S1[x1]S2[y1]S3[z1]",
        PlaceholderBracketRules]]],
     Evaluate[Temp[[1]]]}, MetricOn → All, ContractMetrics → True]];
 printer = printer~Append~PrintTemporary[
    " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)δ(x-x2):"];
 printer = printer~Append~PrintTemporary[Evaluate[ToExpression[
     StringReplace["TDS1[-d,-g,-h,-x1,-y1,-z1,z]S1[x1]S2[y1]S3[z1]",
```

```
      PlaceholderBracketRules]]]];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
       StringReplace["TDS1[-d,-g,-h,-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]",
        PlaceholderBracketRules]]],
    Evaluate[Temp[[2]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)∂δ(x-x2):"];
printer = printer~Append~PrintTemporary[Evaluate[ToExpression[
     StringReplace["TDS2[-d,-g,-h,-x1,-y1,-z1,z]S1[x1]S2[y1]S3[z1]",
      PlaceholderBracketRules]]]];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
       StringReplace["TDS2[-d,-g,-h,-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]",
        PlaceholderBracketRules]]],
    Evaluate[Temp[[3]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)∂δ(x-x2):"];
printer = printer~Append~PrintTemporary[Evaluate[ToExpression[
     StringReplace["TDS3[-d,-g,-h,-x1,-y1,-z1,v,z]S1[x1]S2[y1]S3[z1]",
      PlaceholderBracketRules]]]];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
       StringReplace["TDS3[-d,-g,-h,-x1,-y1,-z1,v,z]S1[x1]S2[y1]S3[z1]",
        PlaceholderBracketRules]]],
    Evaluate[Temp[[4]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[2]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → EH0];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→EH0,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace[
       "CD[-u][TDS1[-d,-g,-h,-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of δ(x-x1)∂δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[3]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → EH0];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
```

```
      "ToShell"→True,"Hard"→True,"Order"→EH0,"GToFoliG"→False];*)
   PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
     MakeRule[{Evaluate[ToExpression[StringReplace[
          "CD[-u][TDS2[-d,-g,-h,-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]",
          PlaceholderBracketRules]]],
       Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
   printer = printer~Append~PrintTemporary[
      " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)∂δ(x-x2):"];
   GradTemp = CD[-u][Evaluate[Temp[[4]]]];
   GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → EH0];
   printer = printer~Append~PrintTemporary[GradTemp];
   (*GradTemp=ToNesterForm[GradTemp,
      "ToShell"→True,"Hard"→True,"Order"→EH0,"GToFoliG"→False];*)
   PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
     MakeRule[{Evaluate[ToExpression[StringReplace[
          "CD[-u][TDS3[-d,-g,-h,-x1,-y1,-z1,v,z]]S1[x1]S2[y1]S3[z1]",
          PlaceholderBracketRules]]],
       Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];

   (*Now we calculate the commutator replacement rule part*)
   VelocitySegments = $InertVelocity[[{11, 13}]];
   VelocitySegments =
    ImposeCommutatorReplacementRules[PlaceholderBracketActivate,
      #, PsiFreeIndexListNormal] & /@ VelocitySegments;

   NotebookDelete[printer];
   VelocitySegments];
DistributeDefinitions@TorsionBracketParallel;
ClearBuild[];
```

build

Surface bracket

build

```
SurfaceBracketParallel[Psi_, EH0_,
    PsiFreeIndexListNormal_, $TheoryName_ : $TheoryName] :=
  Module[{Temp, GradTemp, PlaceholderBracketActivate,
     printer, PsiInert, PlaceholderBracketRulesInert,
     EH0Inert, PsiFreeIndexListNormalInert, result},
    (*Build the HiGGS environment*)
    (*$Timing=True;*)
    BuildHiGGS[];
    (*import theory names*)
    Quiet@ToExpression["<<" <> FileNameJoin@
        {$WorkingDirectory, "svy", "node-" <> $Node, "peta4.nom.mx"} <> ";"];
    (*Define the theory*)
    DefTheory["Import" -> $TheoryName];
    PsiInert = ToString@Psi;
    EH0Inert = ToString@EH0;
    PsiFreeIndexListNormalInert = ToString@PsiFreeIndexListNormal;
    result = ToExpression@("SurfaceBracket[" <> PsiInert <>
        "," <> EH0Inert <> "," <> PsiFreeIndexListNormalInert <> "]");
    ForceTiming[];
    result];
ClearBuild[];
```

build

```
SurfaceBracket[Psi_, EH0_, PsiFreeIndexListNormal_] :=
  Module[{Temp, GradTemp, PlaceholderBracketActivate, printer,
    PsiFreeIndexListD, PsiFreeIndexListDLength, PlaceholderVectors,
    DeltaList, zz, PlaceholderBracketRules, VelocitySegments},
   printer = {};
   PlaceholderBracketActivate = {};

   Print@$InertVelocity;
   Print@"surf";

   PsiFreeIndexListD = Map[ToString[#] &, PsiFreeIndexListNormal];
   PsiFreeIndexListDLength = Length[PsiFreeIndexListD];
   PlaceholderVectors = {"S1[x1]", "S2[y1]", "S3[z1]"};
   DeltaList = {"G[x1,-k]", "G[y1,-k]", "G[z1,-k]"};
   PlaceholderBracketRules = {};
   For[zz = 1, zz < PsiFreeIndexListDLength + 1, zz++, PlaceholderBracketRules =
     Append[PlaceholderBracketRules, PlaceholderVectors[[zz]] →
       StringReplace[DeltaList[[zz]], {"-k" → PsiFreeIndexListD[[zz]]}]]];
```

```
(*this is the `hard'
 segment: the surface quantity that actually gives most of the velocities*)
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Surface bracket..."];
Temp = PoissonBracket[Psi, -V[k] G3[m, -n]
   (CD[-m][BPi[-k, n]] - A[w, -k, -m] BPi[-w, n]), "ToShell" → True,
  "Hard" → True, "Surficial" → False, "Order" → 1, "GToFoliG" → False,
  "NesterForm" → False, "PrintAnswer" -> False];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace[
       "QD[-x1,-y1,-z1]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[1]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace[
       "QDS1[-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[2]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)∂δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace[
       "QDS2[-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[3]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)∂δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[
     ToExpression[StringReplace["QDS3[-x1,-y1,-z1,v,z]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[Temp[[4]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[2]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → 1];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→1,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
      StringReplace["CD[-u][QDS1[-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]",
```

```
        PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of δ(x-x1)∂δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[3]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → 1];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→1,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
      StringReplace["CD[-u][QDS2[-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)∂δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[4]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → 1];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→1,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
      StringReplace["CD[-u][QDS3[-x1,-y1,-z1,v,z]]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];

(*this is the `easier' segment: we look at the Lapse function*)
printer =
 printer~Append~PrintTemporary[" ** PoissonBracket: Lapse bracket..."];
Temp = PoissonBracket[Psi, Lapse[], "ToShell" → True, "Hard" → True,
  "Surficial" → False, "Order" → 1, "GToFoliG" → False,
  "NesterForm" → False, "PrintAnswer" -> False];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace[
       "LapseD[-x1,-y1,-z1]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[1]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[
     ToExpression[StringReplace["LapseDS1[-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]",
```

```
        PlaceholderBracketRules]]],
    Evaluate[Temp[[2]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)∂δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[
     ToExpression[StringReplace["LapseDS2[-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[Temp[[3]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)∂δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
      StringReplace["LapseDS3[-x1,-y1,-z1,v,z]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[Temp[[4]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[2]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → 1];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→1,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
      StringReplace["CD[-u][LapseDS1[-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of δ(x-x1)∂δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[3]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → 1];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→1,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[
      StringReplace["CD[-u][LapseDS2[-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)∂δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[4]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → 1];
```

```
    printer = printer~Append~PrintTemporary[GradTemp];
    (*GradTemp=ToNesterForm[GradTemp,
       "ToShell"→True,"Hard"→True,"Order"→1,"GToFoliG"→False];*)
    PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
      MakeRule[{Evaluate[ToExpression[
          StringReplace["CD[-u][LapseDS3[-x1,-y1,-z1,v,z]]S1[x1]S2[y1]S3[z1]",
           PlaceholderBracketRules]]],
        Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];

    (*Now we calculate the commutator replacement rule part*)
    VelocitySegments = {$InertVelocity[[19]]};
    VelocitySegments =
     ImposeCommutatorReplacementRules[PlaceholderBracketActivate,
        #, PsiFreeIndexListNormal] & /@ VelocitySegments;

    NotebookDelete[printer];
    VelocitySegments];
DistributeDefinitions@SurfaceBracketParallel;
ClearBuild[];
```

build

Measure bracket

build

```
MeasureBracketParallel[Psi_, EH0_,
    PsiFreeIndexListNormal_, $TheoryName_ : $TheoryName] :=
  Module[{Temp, GradTemp, PlaceholderBracketActivate,
     printer, PsiInert, PlaceholderBracketRulesInert,
     EH0Inert, PsiFreeIndexListNormalInert, result},
    (*Build the HiGGS environment*)
    (*$Timing=True;*)
    BuildHiGGS[];
    (*import theory names*)
    Quiet@ToExpression["<<" <> FileNameJoin@
        {$WorkingDirectory, "svy", "node-" <> $Node, "peta4.nom.mx"} <> ";"];
    (*Define the theory*)
    DefTheory["Import" -> $TheoryName];
    PsiInert = ToString@Psi;
    EH0Inert = ToString@EH0;
    PsiFreeIndexListNormalInert = ToString@PsiFreeIndexListNormal;
    result = ToExpression@("MeasureBracket[" <> PsiInert <>
        "," <> EH0Inert <> "," <> PsiFreeIndexListNormalInert <> "]");
    ForceTiming[];
    result];
ClearBuild[];
```

build

```
MeasureBracket[Psi_, EH0_, PsiFreeIndexListNormal_] :=
  Module[{Temp, GradTemp, PlaceholderBracketActivate, printer,
    PsiFreeIndexListD, PsiFreeIndexListDLength, PlaceholderVectors,
    DeltaList, zz, PlaceholderBracketRules, VelocitySegments},
   printer = {};
   PlaceholderBracketActivate = {};

   Print@$InertVelocity;
   Print@"meas";

   PsiFreeIndexListD = Map[ToString[#] &, PsiFreeIndexListNormal];
   HiGGSPrint[PsiFreeIndexListD];
   PsiFreeIndexListDLength = Length[PsiFreeIndexListD];
   HiGGSPrint[PsiFreeIndexListDLength];
   PlaceholderVectors = {"S1[x1]", "S2[y1]", "S3[z1]"};
   HiGGSPrint[PlaceholderVectors];
   DeltaList = {"G[x1,-k]", "G[y1,-k]", "G[z1,-k]"};
   HiGGSPrint[DeltaList];
   PlaceholderBracketRules = {};
```

```
HiGGSPrint[PlaceholderBracketRules];
For[zz = 1, zz < PsiFreeIndexListDLength + 1, zz++, PlaceholderBracketRules =
  Append[PlaceholderBracketRules, PlaceholderVectors[[zz]] →
    StringReplace[DeltaList[[zz]], {"-k" → PsiFreeIndexListD[[zz]]}]]];
HiGGSPrint[PlaceholderBracketRules];

printer =
 printer~Append~PrintTemporary[" ** PoissonBracket: Measure bracket..."];
Temp = PoissonBracket[Psi, Lapse[] J[], "ToShell" → True,
  "Hard" → True, "Surficial" → False, "Order" → 1, "GToFoliG" → False,
  "NesterForm" → False, "PrintAnswer" -> False];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace[
       "JD[-x1,-y1,-z1]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[1]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace[
       "JDS1[-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[2]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)∂δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace[
       "JDS2[-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[3]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)∂δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[
     ToExpression[StringReplace["JDS3[-x1,-y1,-z1,v,z]S1[x1]S2[y1]S3[z1]",
       PlaceholderBracketRules]]],
    Evaluate[Temp[[4]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[2]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → 1];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→1,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
```

```
    MakeRule[{Evaluate[ToExpression[
        StringReplace["CD[-u][JDS1[-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]",
         PlaceholderBracketRules]]],
      Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
  printer = printer~Append~PrintTemporary[
    " ** PoissonBracket: Rule for ∂ coefficient of δ(x-x1)∂δ(x-x2):"];
  GradTemp = CD[-u][Evaluate[Temp[[3]]]];
  GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → 1];
  printer = printer~Append~PrintTemporary[GradTemp];
  (*GradTemp=ToNesterForm[GradTemp,
    "ToShell"→True,"Hard"→True,"Order"→1,"GToFoliG"→False];*)
  PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
    MakeRule[{Evaluate[ToExpression[
        StringReplace["CD[-u][JDS2[-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]",
         PlaceholderBracketRules]]],
      Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
  printer = printer~Append~PrintTemporary[
    " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)∂δ(x-x2):"];
  GradTemp = CD[-u][Evaluate[Temp[[4]]]];
  GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → 1];
  printer = printer~Append~PrintTemporary[GradTemp];
  (*GradTemp=ToNesterForm[GradTemp,
    "ToShell"→True,"Hard"→True,"Order"→1,"GToFoliG"→False];*)
  PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
    MakeRule[{Evaluate[ToExpression[
        StringReplace["CD[-u][JDS3[-x1,-y1,-z1,v,z]]S1[x1]S2[y1]S3[z1]",
         PlaceholderBracketRules]]],
      Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];

  (*Now we calculate the commutator replacement rule part*)

  VelocitySegments = $InertVelocity[[{15, 16, 17, 18}]];
  VelocitySegments =
   ImposeCommutatorReplacementRules[PlaceholderBracketActivate,
    #, PsiFreeIndexListNormal] & /@ VelocitySegments;

  NotebookDelete[printer];
  VelocitySegments];
DistributeDefinitions@MeasureBracketParallel;
ClearBuild[];
```

build

Constraint bracket

build

```
ConstraintBracketParallel[Psi_, EH0_,
   FreeConstraintString_, PhiFreeIndexListNormal_, ii_,
   PsiFreeIndexListNormal_, $TheoryName_ : $TheoryName] :=
  Module[{Temp, GradTemp, PlaceholderBracketActivate, printer, PsiInert,
    PlaceholderBracketRulesInert, EH0Inert, FreeConstraintInert,
    PhiFreeIndexListNormalInert, iiInert, PsiFreeIndexListNormalInert, result},
   (*Build the HiGGS environment*)
   (*$Timing=True;*)
   BuildHiGGS[];
   (*import theory names*)
   Quiet@ToExpression["<<" <> FileNameJoin@
       {$WorkingDirectory, "svy", "node-" <> $Node, "peta4.nom.mx"} <> ";"];
   (*Define the theory*)
   DefTheory["Import" -> $TheoryName];
   PsiInert = ToString@Psi;
   EH0Inert = ToString@EH0;
   PhiFreeIndexListNormalInert = ToString@PhiFreeIndexListNormal;
   iiInert = ToString@ii;
   PsiFreeIndexListNormalInert = ToString@PsiFreeIndexListNormal;
   result = ToExpression@("ConstraintBracket[" <> PsiInert <> "," <> EH0Inert <>
       "," <> FreeConstraintString <> "," <> PhiFreeIndexListNormalInert <>
       "," <> iiInert <> "," <> PsiFreeIndexListNormalInert <> "]");
   ForceTiming[];
   result];
ClearBuild[];
```

build

```
ConstraintBracket[Psi_, EH0_, FreeConstraint_,
   PhiFreeIndexListNormal_, ii_, PsiFreeIndexListNormal_] := Module[
   {Temp, GradTemp, PlaceholderBracketActivate, printer, PhiFreeIndexListString,
    PsiFreeIndexListD, PsiFreeIndexListDLength, PlaceholderVectors,
    DeltaList, zz, PlaceholderBracketRules, VelocitySegments},
   printer = {};
   PlaceholderBracketActivate = {};

   Print@$InertVelocity;
   Print@"cons";

   PsiFreeIndexListD = Map[ToString[#] &, PsiFreeIndexListNormal];
   HiGGSPrint[PsiFreeIndexListD];
   PsiFreeIndexListDLength = Length[PsiFreeIndexListD];
   HiGGSPrint[PsiFreeIndexListDLength];
```

```
PlaceholderVectors = {"S1[x1]", "S2[y1]", "S3[z1]"};
HiGGSPrint[PlaceholderVectors];
DeltaList = {"G[x1,-k]", "G[y1,-k]", "G[z1,-k]"};
HiGGSPrint[DeltaList];
PlaceholderBracketRules = {};
HiGGSPrint[PlaceholderBracketRules];
For[zz = 1, zz < PsiFreeIndexListDLength + 1, zz++, PlaceholderBracketRules =
  Append[PlaceholderBracketRules, PlaceholderVectors[[zz]] →
    StringReplace[DeltaList[[zz]], {"-k" → PsiFreeIndexListD[[zz]]}]]];
HiGGSPrint[PlaceholderBracketRules];

PhiFreeIndexListString = StringDelete[
  StringTrim[ToString[PhiFreeIndexListNormal], ("{" | "}")], " "];
If[Length[ToExpression@PhiFreeIndexListNormal] ≠ 0,
 PhiFreeIndexListString = PhiFreeIndexListString <> ","];

printer = printer ~ Append ~
  PrintTemporary[" ** PoissonBracket: Constraint bracket..."];
printer = printer ~ Append ~ PrintTemporary[FreeConstraint];
Temp = PoissonBracket[Psi, FreeConstraint,
  "ToShell" → True, "Hard" → True, "Surficial" → False, "Order" → EH0,
  "GToFoliG" → False, "NesterForm" → False, "PrintAnswer" -> False];
printer = printer ~ Append ~ PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace["PhiD" <> ToString[
          SectorNames[[ii]]] <> "[" <> ToString[PhiFreeIndexListString] <>
        "-x1,-y1,-z1]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[1]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer ~ Append ~ PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace["PhiDS1" <> ToString[
          SectorNames[[ii]]] <> "[" <> ToString[PhiFreeIndexListString] <>
        "-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[2]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer ~ Append ~ PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of δ(x-x1)∂δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace["PhiDS2" <> ToString[
          SectorNames[[ii]]] <> "[" <> ToString[PhiFreeIndexListString] <>
        "-x1,-y1,-z1,v]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[3]]]}, MetricOn → All, ContractMetrics → True]];
```

```
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for coefficient of ∂δ(x-x1)∂δ(x-x2):"];
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace["PhiDS3" <> ToString[
          SectorNames[[ii]]] <> "[" <> ToString[PhiFreeIndexListString] <>
        "-x1,-y1,-z1,v,z]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[Temp[[4]]]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[2]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → EH0];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→EH0,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace["CD[-u][PhiDS1" <> ToString[
          SectorNames[[ii]]] <> "[" <> ToString[PhiFreeIndexListString] <>
        "-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of δ(x-x1)∂δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[3]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → EH0];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→EH0,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace["CD[-u][PhiDS2" <> ToString[
          SectorNames[[ii]]] <> "[" <> ToString[PhiFreeIndexListString] <>
        "-x1,-y1,-z1,v]]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Rule for ∂ coefficient of ∂δ(x-x1)∂δ(x-x2):"];
GradTemp = CD[-u][Evaluate[Temp[[4]]]];
GradTemp = ToBasicForm[GradTemp, "Hard" → True, "Order" → EH0];
printer = printer~Append~PrintTemporary[GradTemp];
(*GradTemp=ToNesterForm[GradTemp,
   "ToShell"→True,"Hard"→True,"Order"→EH0,"GToFoliG"→False];*)
PlaceholderBracketActivate = Join[PlaceholderBracketActivate,
  MakeRule[{Evaluate[ToExpression[StringReplace["CD[-u][PhiDS3" <> ToString[
          SectorNames[[ii]]] <> "[" <> ToString[PhiFreeIndexListString] <>
        "-x1,-y1,-z1,v,z]]S1[x1]S2[y1]S3[z1]", PlaceholderBracketRules]]],
    Evaluate[GradTemp]}, MetricOn → All, ContractMetrics → True]];
```

```
    (*Now we calculate the commutator replacement rule part*)
    VelocitySegments = {$InertVelocity[[ii]]};
    VelocitySegments =
     ImposeCommutatorReplacementRules[PlaceholderBracketActivate,
        #, PsiFreeIndexListNormal] & /@ VelocitySegments;

    NotebookDelete[printer];
    VelocitySegments];
DistributeDefinitions@ConstraintBracketParallel;
ClearBuild[];
```

build

Velocity

(depreciated)

build

```
Options[Velocity] = {"InertVelocity" → $InertVelocity,
    "Order" → Infinity, "PrintAnswer" -> True, "Parallel" -> False};

Velocity[Psi_, OptionsPattern[]] :=
  "Velocity"~TimeWrapper~Catch@Block[{Temp, GradTemp, PsiFreeIndices,
      PsiFreeIndexList, PhiFreeIndexList, PsiFreeIndexListLength,
      PhiFreeIndexListString, PhiFreeIndexListNormal, PlaceholderVectors,
      DeltaList, PlaceholderBracketRules, return, FreeConstraint,
      FreeConstraintString, PlaceholderBracketActivate, ii, KeepOnlyObviousZeros,
      EH0, Phis, printer, Jobs, RuleResults, PsiFreeIndexListNormal,
      PsiFreeIndexListString, PsiFreeIndexListD, PsiFreeIndexListDLength},
     (*a message*)
     printer = {};
     printer = printer~Append~PrintTemporary[" ** Velocity of ",
        Psi, " with options ", Options[Velocity], "..."];
     (*
     (#~ChangeFreeIndices~({-q1,-p1,-v1}~Take~Length@FindFreeIndices@#))&;
     *)
     KeepOnlyObviousZeros[q_] := If[q == 0, 0, 1, 1];
     (*We fix EH0 legacy*)
     Switch[KeepOnlyObviousZeros@(Alp0 /. $ToTheory), 0, EH0 = 0, 1, EH0 = 1];

     printer = printer~Append~
       PrintTemporary[" ** PoissonBracket: Stripping indices..."];

     PsiFreeIndexList = FindFreeIndices[Psi];
     PsiFreeIndexListString = StringDelete[
```

```
  StringTrim[ToString[PsiFreeIndexList], ("IndexList[" | "]")], " "];
PsiFreeIndexListNormal = "{" <> PsiFreeIndexListString <> "}";
DistributeDefinitions@PsiFreeIndexListNormal;

PsiFreeIndices = FindFreeIndices[Psi];
PsiFreeIndexListD =
 Developer`ToList[Delete[Map[ToString[#] &, PsiFreeIndices], 0]];
PsiFreeIndexListDLength = Length[PsiFreeIndexListD];
PlaceholderVectors = {"S1[x1]", "S2[y1]", "S3[z1]"};
DeltaList = {"G[x1,-k]", "G[y1,-k]", "G[z1,-k]"};
PlaceholderBracketRules = {};
For[ii = 1, ii < PsiFreeIndexListDLength + 1, ii++, PlaceholderBracketRules =
  Append[PlaceholderBracketRules, PlaceholderVectors[[ii]] →
    StringReplace[DeltaList[[ii]], {"-k" → PsiFreeIndexListD[[ii]]}]]];

PlaceholderBracketActivate = {};

If[OptionValue["Parallel"],
 DistributeDefinitions@PlaceholderBracketRules;
 DistributeDefinitions@EH0;
 Jobs =
  {ParallelSubmit@RiemannBracketParallel[Psi, EH0, PsiFreeIndexListNormal],
   ParallelSubmit@TorsionBracketParallel[Psi, EH0, PsiFreeIndexListNormal],
   ParallelSubmit@SurfaceBracketParallel[Psi, EH0, PsiFreeIndexListNormal],
   ParallelSubmit@MeasureBracketParallel[Psi, EH0, PsiFreeIndexListNormal],
   ParallelSubmit@LapseBracketParallel[
     Psi, EH0, PsiFreeIndexListNormal]};
 Phis = {PhiB0p[], PhiB1p[-i, -j], PhiB1m[-i], PhiB2p[-i, -j],
   PhiA0p[], PhiA0m[], PhiA1p[-i, -j], PhiA1m[-i],
   PhiA2p[-i, -j], PhiA2m[-i, -j, -k]};
 For[ii = 1, ii < 11, ii++, If[Evaluate[ToExpression["ShellOrig" <>
       ToString[SectorNames[[ii]]]] /. $ToShellFreedoms] == 1, {
    FreeConstraint = Phis[[ii]];
    PhiFreeIndexList = FindFreeIndices[Evaluate[FreeConstraint]];
    PhiFreeIndexListString = StringDelete[
      StringTrim[ToString[PhiFreeIndexList], ("IndexList[" | "]")], " "];
    PhiFreeIndexListNormal = "{" <> PhiFreeIndexListString <> "}";
    FreeConstraintString = ToString@FreeConstraint;
    DistributeDefinitions@FreeConstraintString;
    DistributeDefinitions@PhiFreeIndexListString;
    DistributeDefinitions@PhiFreeIndexListNormal;
    DistributeDefinitions@ii;
    Jobs = Jobs ~ Join ~ {ParallelSubmit@
```

```
          ConstraintBracketParallel[Psi, EH0, FreeConstraintString,
           PhiFreeIndexListNormal, ii, PsiFreeIndexListNormal]}]];
 RuleResults = WaitAll[Jobs];
 (*Quit[];*)
 PlaceholderBracketActivate = Flatten@RuleResults;
 HiGGSPrint[PlaceholderBracketActivate];,
 PlaceholderBracketActivate = PlaceholderBracketActivate~
   Join~RiemannBracket[Psi, EH0, PsiFreeIndexListNormal];
 PlaceholderBracketActivate = PlaceholderBracketActivate~
   Join~TorsionBracket[Psi, EH0, PsiFreeIndexListNormal];
 PlaceholderBracketActivate = PlaceholderBracketActivate~
   Join~SurfaceBracket[Psi, EH0, PsiFreeIndexListNormal];
 PlaceholderBracketActivate = PlaceholderBracketActivate~
   Join~MeasureBracket[Psi, EH0, PsiFreeIndexListNormal];
 PlaceholderBracketActivate = PlaceholderBracketActivate~
   Join~LapseBracket[Psi, EH0, PsiFreeIndexListNormal];
 Phis = {PhiB0p[], PhiB1p[-i, -j], PhiB1m[-i], PhiB2p[-i, -j],
   PhiA0p[], PhiA0m[], PhiA1p[-i, -j], PhiA1m[-i],
   PhiA2p[-i, -j], PhiA2m[-i, -j, -k]};
 For[ii = 1, ii < 11, ii++, If[Evaluate[ToExpression["ShellOrig" <>
        ToString[SectorNames[[ii]]]] /. $ToShellFreedoms] == 1, {
    FreeConstraint = Phis[[ii]];
    PhiFreeIndexList = FindFreeIndices[Evaluate[FreeConstraint]];
    PhiFreeIndexListString = StringDelete[
      StringTrim[ToString[PhiFreeIndexList], ("IndexList[" | "]")], " "];
    PhiFreeIndexListNormal = "{" <> PhiFreeIndexListString <> "}";
    PlaceholderBracketActivate = PlaceholderBracketActivate~
      Join~ConstraintBracket[Psi, EH0, FreeConstraint,
       PhiFreeIndexListNormal, ii, PsiFreeIndexListNormal];
   }]];
];

printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Imposing commutator replacement rules..."];
return = Evaluate@OptionValue["InertVelocity"];
return = return /. PlaceholderBracketActivate;
(*HiGGSPrint[return];*)
return = ToOrderCanonical[return, 1];
printer = printer~Append~
  PrintTemporary[ToBasicForm[return, "Hard" → True, "Order" → 1]];
printer = printer~Append~PrintTemporary[
   " ** PoissonBracket: Imposing Nester form..."];
return = ToNesterForm[return, "ToShell" → True, "Hard" → True, "Order" → 1];
```

```
    printer =
     printer~Append~PrintTemporary[" ** PoissonBracket: Re-expanding η̂
         because answer is a product of Nester forms..."];
    return = return /. FoliGToG;
    return = return // ToNewCanonical;
    return = return /. GToFoliG;
    return = return // ToNewCanonical;
    printer = printer~Append~
      PrintTemporary[" ** PoissonBracket: Final form of linear velocity:"];
    NotebookDelete[printer];
    If[OptionValue["PrintAnswer"], HiGGSPrint["∂/∂t", Psi, " ≈ ", return];];
    return];
ClearBuild[];
```

build

Parallel velocity

build

```
SetupVelocitySegments[Psi_, EH0_, $TheoryName_ : $TheoryName] :=
  Module[{printer, PsiFreeIndexList, PsiFreeIndexListString,
    PsiFreeIndexListNormal, FreeConstraint, PhiFreeIndexList,
    PhiFreeIndexListString, Jobs, ii, Phis, SetupConstraintSegment},

   (*a message*)
   printer = {};
   printer = printer~Append~
     PrintTemporary[" ** SetupVelocitySegments of ", Psi, "..."];

   PsiFreeIndexList = FindFreeIndices[Psi];
   PsiFreeIndexListString = StringDelete[
     StringTrim[ToString[PsiFreeIndexList], ("IndexList[" | "]")], " "];
   PsiFreeIndexListNormal = "{" <> PsiFreeIndexListString <> "}";
   DistributeDefinitions@PsiFreeIndexListNormal;

   Phis = {PhiB0p[], PhiB1p[-i, -j], PhiB1m[-i], PhiB2p[-i, -j], PhiA0p[],
     PhiA0m[], PhiA1p[-i, -j], PhiA1m[-i], PhiA2p[-i, -j], PhiA2m[-i, -j, -k]};

   (*
   Jobs={ParallelSubmit@
       MeasureBracketParallel[Psi,EH0,PsiFreeIndexListNormal,$TheoryName]};
   *)
   (**)
   Jobs = {ParallelSubmit@
```

```
      RiemannBracketParallel[Psi, EH0, PsiFreeIndexListNormal, $TheoryName],
     ParallelSubmit@TorsionBracketParallel[Psi, EH0,
       PsiFreeIndexListNormal, $TheoryName], ParallelSubmit@
      SurfaceBracketParallel[Psi, EH0, PsiFreeIndexListNormal, $TheoryName]};

   (**)
   (*
   ParallelSubmit@
     MeasureBracketParallel[Psi,EH0,PsiFreeIndexListNormal,$TheoryName];
   *)
   (**)
   SetupConstraintSegment[ii_] :=
    Module[{jj, FreeConstraintString, PhiFreeIndexListNormal},
     jj = ii;
     DistributeDefinitions@jj;

     FreeConstraint = Phis[[jj]];
     FreeConstraintString = ToString@FreeConstraint;
     DistributeDefinitions@FreeConstraintString;

     PhiFreeIndexList = FindFreeIndices[Evaluate[FreeConstraint]];
     PhiFreeIndexListString = StringDelete[
       StringTrim[ToString[PhiFreeIndexList], ("IndexList[" | "]")], " "];
     PhiFreeIndexListNormal = "{" <> PhiFreeIndexListString <> "}";
     DistributeDefinitions@PhiFreeIndexListNormal;

     Jobs = Jobs ~ Join ~
       {ParallelSubmit@ConstraintBracketParallel[Psi, EH0, FreeConstraintString,
          PhiFreeIndexListNormal, jj, PsiFreeIndexListNormal, $TheoryName]}
    ];

   For[ii = 1, ii < 11, ii++,
    If[Evaluate[ToExpression["ShellOrig" <> ToString[SectorNames[[ii]]]] /.
         $ToShellFreedoms] == 1, {
       SetupConstraintSegment[ii];
      }
     ];
   ];
   (**)
   NotebookDelete[printer];
   Jobs];
ClearBuild[];
```

build

```
ImposeCommutatorReplacementRules[PlaceholderBracketActivate_,
```

```
    InertVelocityPart_, PsiFreeIndexListNormal_] :=
  Module[{return, printer, PsiFreeIndexListD, PsiFreeIndexListDLength,
    PlaceholderVectors, VectorsToDiff, zz},

   (*a message*)
   printer = {};
   printer =
    printer~Append~PrintTemporary[" ** ImposeCommutatorReplacementRules"];

   (*routine to strip the sigma vectors*)
   PsiFreeIndexListD = Map[ToString[#] &, PsiFreeIndexListNormal];
   PsiFreeIndexListDLength = Length[PsiFreeIndexListD];
   PlaceholderVectors = {"S1[-k]", "S2[-k]", "S3[-k]"};
   VectorsToDiff = {};
   For[zz = 1, zz < PsiFreeIndexListDLength + 1, zz++,
    VectorsToDiff = Append[VectorsToDiff, ToExpression@StringReplace[
        PlaceholderVectors[[zz]], {"-k" → PsiFreeIndexListD[[zz]]}]]];

   (*simplification process*)
   return = Evaluate@InertVelocityPart;
   For[zz = 1, zz < Length@VectorsToDiff + 1,
    zz++, return = return~NewVarAction~VectorsToDiff[[zz]]];
   return = return /. PlaceholderBracketActivate;
   return = ToOrderCanonical[return, 1];
   printer = printer~Append~
     PrintTemporary[ToBasicForm[return, "Hard" → True, "Order" → 1]];
   printer = printer~Append~PrintTemporary[
      " ** PoissonBracket: Imposing Nester form..."];
   return = ToNesterForm[return, "ToShell" → True, "Hard" → True, "Order" → 1];
   printer =
    printer~Append~PrintTemporary[" ** PoissonBracket: Re-expanding η̂ because
        answer is a product of Nester forms..."];
   return = return /. FoliGToG;
   return = return // ToNewCanonical;
   return = return /. GToFoliG;
   return = return // ToNewCanonical;

   NotebookDelete[printer];
   HiGGSPrint["∂/∂t ψ ≈ ", return];
   return];
ClearBuild[];
```

build

```
SecondaryVelocitySimplification[VelocitySegments_List] :=
  Module[{return, printer},

   (*a message*)
   printer = {};
   printer =
    printer~Append~PrintTemporary[" ** SecondaryVelocitySimplification"];

   return = Flatten@VelocitySegments;
   return = Total@return;

   (*simplification process*)
   printer =
    printer~Append~PrintTemporary[" ** PoissonBracket: Re-expanding η̂ because
         answer is a product of Nester forms..."];
   return = return /. FoliGToG;
   return = return // ToNewCanonical;
   return = return /. GToFoliG;
   return = return // ToNewCanonical;

   NotebookDelete[printer];
   HiGGSPrint[" ∂/∂t ψ ≈ ", return];
   return];
ClearBuild[];
```

build

```
Options[VelocityParallel] = {"Order" → Infinity, "PrintAnswer" -> True};
VelocityParallel[BatchPsis_List, OptionsPattern[]] :=
  "Velocity"~TimeWrapper~Catch@Block[{KeepOnlyObviousZeros, printer,
      Jobs, SplitVelocities, Velocities, PrepareVelocitySegments},

     (*a message*)
     printer = {};
     printer = printer~Append~PrintTemporary[" ** VelocityParallel of ",
        BatchPsis, " with options ", Options[VelocityParallel], "..."];

     (*We fix EH0 legacy*)
     KeepOnlyObviousZeros[q_] := If[q == 0, 0, 1, 1];

     PrepareVelocitySegments[theory_String, Psis_List] := Module[{res, EH0},
       DefTheory["Import" → theory];
       Switch[KeepOnlyObviousZeros@(Alp0 /. $ToTheory), 0, EH0 = 0, 1, EH0 = 1];
       DistributeDefinitions@EH0;

       (*Large batch of jobs for segments of all velocities*)
       res = SetupVelocitySegments[#, EH0, theory] & /@ Psis;
       res];

     Jobs = (#1~PrepareVelocitySegments~#2) &@@@ BatchPsis;

     HiGGSPrint[Jobs];

     SplitVelocities = WaitAll[Jobs];

     Print[SplitVelocities];

     Velocities = (SecondaryVelocitySimplification /@ #) & /@ SplitVelocities;

     Print[Velocities];

     NotebookDelete[printer];
     Velocities];
ClearBuild[];
```

build

DefTheory

build

```
Options[DefTheoryParallel] =
  {"Export" → False, "Import" -> False, "Velocities" → True, "Order" → 1};
```

build

```
DefTheoryParallel[InputSystem___ : Null, OptionsPattern[]] := Module[{},
   (*Build the HiGGS environment*)
   (*$Timing=True;*)
   BuildHiGGS[];
   (*import theory names*)
   Quiet@ToExpression["<<" <> FileNameJoin@
       {$WorkingDirectory, "svy", "node-" <> $Node, "peta4.nom.mx"} <> ";"];
   (*Define the theory*)
   DefTheory[InputSystem,
    "Export" -> OptionValue["Export"], "Import" → OptionValue["Import"],
    "Velocities" → OptionValue@"Velocities", "Order" → OptionValue@"Order"];
   ForceTiming[];
  ];
DistributeDefinitions@DefTheoryParallel;
```

build

```
TheoryQ[x_] := Module[{res},
   res = ListQ[x];
   If[res,
    res = Flatten@{{Alp0}, Alp, Bet, cAlp, cBet} ~
       SubsetQ ~ Flatten@(Variables /@ Flatten@((List @@ (#)) & /@ x));
   ];
   res];
DefTheory::nottheory =
  "Argument `1` is not a linear system in Alp0,...,Alp6, Bet1,...,Bet3,
    cAlp1,...,cAlp6 and cBet1,...,cBet3, e.g. {Alp0+Alp1==0,...}.";
DefTheory::nobin = "The binary at `1` cannot be found; quitting.";
Options[DefTheory] = {"Export" → False, "Import" -> False,
   "Velocities" → False, "Order" → 1, "ProtectSurface" → False};
UndefTheory[] := Clear @@ {"$TheoryName", "$Theory", "$ToTheory",
    "$ToShellFreedoms", "$StrengthPShellToStrengthPO3",
    "$PiPShellToPiPPO3", "$TheoryCDPiPToCDPiPO3", "$TheoryPiPToPiPO3",
    "$IfConstraintToTheoryNesterForm", "$IfConstraints",
    "$InertVelocity", "$ToOrderRules", "$PPM", "$Velocities"};
```

build

```
(*removed timing wrapper since it is better to show
```

```
 the internal steps -- these get washed out in the plot*)
(*DefTheory[InputSystem___:Null,OptionsPattern[]]:=
 "DefTheory"~TimeWrapper~Catch@Module[{res},*)
DefTheory[InputSystem___ : Null, OptionsPattern[]] := Catch@Module[{res},
    (*Firstly we remove all definitions
     which might be associated with a theory already*)
    UndefTheory[];
    If[StringQ@OptionValue@"Import",
     HiGGSPrint[" ** DefTheory: Incorporating the binary at " <>
       FileNameJoin@{"svy", OptionValue@"Import" <> ".thr.mx"}];
     $TheoryName = OptionValue@"Import";
     Check[ToExpression["<<" <> FileNameJoin@
         {$WorkingDirectory, "svy", OptionValue@"Import" <> ".thr.mx"} <> ";"],
      Throw@Message[DefTheory::nobin, FileNameJoin@{$WorkingDirectory,
          "svy", ToString@OptionValue@"Import" <> ".thr.mx"}];
      Quit[];
     ];,
     (*check if a real theory was provided*)
     If[! TheoryQ[InputSystem],
      Throw@Message[DefTheory::nottheory, InputSystem]];
     (*define the theory constant in Global`*)
     $Theory = InputSystem;
     $Theory = $Theory~Join~{dummy → 0};
     (*a message*)
     xAct`xTensor`Private`MakeDefInfo[
      DefTheory, $Theory, {"$ToTheory for the theory", ""}];
     (*these are rules we can always use to impose the theory*)
     If[$Theory == {dummy → 0},
      $ToTheory = {dummy → 0};,
      $ToTheory =
        Quiet[Solve[InputSystem, Join[cAlp, cBet, {Alp0}, Alp, Bet]][[1]]];
     ];
     (*append a dummy replacement rule so that an empty *)
     (*these functions do all the hard work*)
     ComputeShellFreedoms[$ToTheory, $Theory];
     Print@$ToShellFreedoms;
     DefFieldStrengthShell[$ToShellFreedoms, $Theory];
     DefMomentaShell[$ToShellFreedoms, $ToTheory, $Theory];
     DefO3MomentaShell[$Theory];
     DefIfConstraintToTheoryNesterForm[$ToShellFreedoms, $ToTheory, $Theory];
     DefSuperHamiltonian[$ToShellFreedoms, $IfConstraintToNesterForm,
      $ToTheory, $Theory, "Order" → OptionValue@"Order",
      "ProtectSurface" → OptionValue@"ProtectSurface"];
```

```
      DefLinearSuperMomentum[$ToShellFreedoms, $IfConstraintToNesterForm,
       $ToTheory, $Theory, "Order" → OptionValue@"Order",
       "ProtectSurface" → OptionValue@"ProtectSurface"];
      DefAngularSuperMomentum[$ToShellFreedoms, $IfConstraintToNesterForm,
       $ToTheory, $Theory, "Order" → OptionValue@"Order",
       "ProtectSurface" → OptionValue@"ProtectSurface"];
      If[OptionValue@"Velocities",
       DefInertVelocity[$ToShellFreedoms, $ToTheory, $Theory];
      ];
     ];
     If[StringQ@OptionValue@"Export",
      HiGGSPrint[" ** DefTheory: Exporting the binary at " <>
        FileNameJoin@{"svy", OptionValue@"Export" <> ".thr.mx"}];
      $TheoryName = OptionValue@"Export";
      Print@$IfConstraints;
      (FileNameJoin@
         {$WorkingDirectory, "svy", ToString@OptionValue@"Export" <> ".thr.mx"}) ~
       DumpSave ~ {$TheoryName, $Theory, $ToTheory, $ToShellFreedoms,
        $StrengthPShellToStrengthPO3, $PiPShellToPiPPO3, $TheoryCDPiPToCDPiPO3,
        $TheoryPiPToPiPO3, $IfConstraintToTheoryNesterForm,
        $IfConstraints, $InertVelocity, $ToOrderRules};
     ];
    ];
(*so that a replacement rule exists, even if no theory is defined*)
dummySymb = "ℒmatter";
DefConstantSymbol[dummy, PrintAs → SymbolBuild[dummySymb]];
$ToTheory = {dummy → 0};
ClearBuild[];
```

build

ViewTheory

build

```
Options[ViewTheory] = {"Literature" -> True, "PPM" -> True, "Velocities" -> True};
ViewTheory[theory_String, OptionsPattern[]] := Module[{IndIfConstraints, ii, jj},
   DefTheory["Import" → theory];
   If[OptionValue["Literature"],
    DefIfConstraintToTheoryNesterForm[$ToShellFreedoms, $ToTheory, $Theory];
   ];
   (*
   Print@MatrixForm@$PPM;
   *)
   If[OptionValue["PPM"],
```

```
    IndIfConstraints =
     (# ~ ChangeFreeIndices ~ ({-l, -m, -n} ~ Take ~ Length@FindFreeIndices@#)) & /@
      $IfConstraints;
    $PPMlabels = Table[{$IfConstraints[[ii]], IndIfConstraints[[jj]]},
       {ii, Length@$IfConstraints}, {jj, ii, Length@$IfConstraints}] ~
      PadLeft ~ {Length@$IfConstraints, Length@$IfConstraints};
    $PPM = $PPM ~ PadLeft ~ {Length@$IfConstraints, Length@$IfConstraints};
    PrintBracket[x_, y_] := Module[{nontrivial},
      nontrivial = ! (x == {0, 0, 0} || x == {0, 0, 0, 0} || y == 0);
      If[nontrivial,
       HiGGSPrint[y, " ≈ ", x], Null;,
       HiGGSPrint[y, " ≈ ", x]];
     ];
    Print@" ** ViewTheory: encountered
       the following nonvanishing Poisson brackets:";
    MapThread[PrintBracket, {$PPM, $PPMlabels}, 2];
    (*Print/@$Velocities;*)
   ];
   If[OptionValue["Velocities"],
    IndVelocities =
     (# ~ ChangeFreeIndices ~ ({-i, -j, -k} ~ Take ~ Length@FindFreeIndices@#)) & /@
      $Velocities;
    PrintVelocity[x_, y_] := Module[{nontrivial},
      nontrivial = ! (x == 0);
      If[nontrivial,
       HiGGSPrint["d/dt", y, " ≈ ", x], Null;,
       HiGGSPrint["d/dt", y, " ≈ ", x]];
     ];
    Print@
     " ** ViewTheory: encountered the following nonvanishing velocities:";
    MapThread[PrintVelocity, {IndVelocities, $IfConstraints}];
    (*Print/@$Velocities;*)
   ];
  ];
```

build

StudyTheory

build

```
Options[StudyTheory] = {"Export" → False, "Import" -> False,
   "DefTheory" → True, "Brackets" → True, "Velocities" → True};
```

```
StudyTheory[InputBatch___ : Null, OptionsPattern[]] :=
  Module[{LaunchSome, DefinedTheories, IndIfConstraints2, Jobs, PreparePPM,
    PPMs, SavePPM, PrepareVelocities, Velocities, SaveVelocity},
   (*We now want to change this module into something
    which studies batches of theories*)

   (*As long as the 2^- sector remains problematic,
   the optimal quotient will be ~1 theory per core*)
   (*sometimes the launching of kernels simply hangs on the
    node: this repeats the process if it lasts more than n seconds*)
   $TryKernels = True;
   If[ValueQ@$Cores,
    While[$TryKernels,
      HiGGSPrint[" ** StudyTheory: Attempting to launch kernels"];
      CloseKernels[];
      (*launch should be 32*)
      TimeConstrained[Check[LaunchKernels[$Cores], $TryKernels = False;];
       $TryKernels = False;,
       10,
       CloseKernels[];
       HiGGSPrint[" ** StudyTheory: Failed to launch kernels, retrying"];
      ];
     ];,
    While[$TryKernels,
      HiGGSPrint[" ** StudyTheory: Attempting to launch kernels"];
      CloseKernels[];
      (*launch should be 32*)
      TimeConstrained[Check[LaunchKernels[], $TryKernels = False;];
       $TryKernels = False;,
       10,
       CloseKernels[];
       HiGGSPrint[" ** StudyTheory: Failed to launch kernels, retrying"];
      ];
     ];];


   If[OptionValue@"DefTheory",
    If[! OptionValue@"Import",
     If[OptionValue@"Velocities",
      Jobs = ParallelSubmit@DefTheoryParallel[#2,
             "Export" → #1, "Velocities" → True] & @@@ InputBatch;,
      Jobs = ParallelSubmit@DefTheoryParallel[#2, "Export" → #1,
             "Velocities" → False] & @@@ InputBatch;,
```

```
    Jobs = ParallelSubmit@DefTheoryParallel[#2, "Export" → #1,
           "Velocities" → False] &@@@ InputBatch;
   ];
   HiGGSPrint[Jobs];
   DefinedTheories = WaitAll[Jobs];
  ];
  (*problems were encountered using DistributeDefinitions on the list
   of theory name strings for use in timing, so we use a binary*)
  Print@InputBatch;
  $TheoryNames = (#[[1]]) & /@ InputBatch;
  (FileNameJoin@{$WorkingDirectory, "svy", "node-" <> $Node, "peta4.nom.mx"}) ~
   DumpSave ~ {$TheoryNames};
 ];

 (**)

 If[OptionValue@"Brackets",
  PreparePPM[theory_String, conds_List] :=
   Module[{res, PPMArguments, IndIfConstraints},
    DefTheory["Import" → theory];
    IndIfConstraints = (# ~ ChangeFreeIndices ~
          ({-l, -m, -n} ~ Take ~ Length@FindFreeIndices@#)) & /@ $IfConstraints;
    (*Evaluate lots of Poisson brackets*)
    PPMArguments =
     Table[{theory, $IfConstraints[[ii]], IndIfConstraints[[jj]]},
      {ii, Length@$IfConstraints}, {jj, ii, Length@$IfConstraints}];
    PPMArguments];
  Jobs = (#1 ~ PreparePPM ~ #2) &@@@ InputBatch;
  Print@Jobs;
  Jobs = Map[(ParallelSubmit@PoissonBracketParallel[
        #[[2]], #[[3]], #[[1]], "Surficial" -> True]) &, Jobs, {3}];
  Print@Jobs;
  PPMs = WaitAll[Jobs];
  PPMs = Riffle[$TheoryNames, PPMs] ~ Partition ~ 2;
  SavePPM[theory_String, PPM_] :=
   Module[{res, PPMArguments, IndIfConstraints},
    DefTheory["Import" → theory];
    $PPM = PPM;
    HiGGSPrint["$PPM value is ", $PPM];
    HiGGSPrint[" ** StudyTheory: Exporting the binary at " <>
      FileNameJoin@{"svy", theory <> ".thr.mx"}];
    (FileNameJoin@{$WorkingDirectory, "svy", theory <> ".thr.mx"}) ~
     DumpSave ~ {$TheoryName, $Theory, $ToTheory, $ToShellFreedoms,
```

```
        $StrengthPShellToStrengthPO3, $PiPShellToPiPPO3, $TheoryCDPiPToCDPiPO3,
        $TheoryPiPToPiPO3, $IfConstraintToTheoryNesterForm,
        $IfConstraints, $InertVelocity, $ToOrderRules, $PPM};
     ];
    HiGGSPrint[PPMs];
    SavePPM[#1, #2] &@@@ PPMs;
   ];

   (**)

   If[OptionValue@"Velocities",
    PrepareVelocities[theory_String, conds_List] :=
     Module[{res, IndIfConstraints},
      DefTheory["Import" → theory];
      IndIfConstraints = (# ~ ChangeFreeIndices ~ ({-q1, -p1, -v1} ~
            Take ~ Length@FindFreeIndices@#)) & /@ $IfConstraints;
      (*IndIfConstraints=IndIfConstraints~Take~-1;*)
      (*IndIfConstraints={IndIfConstraints[[6]]};*)
      (*Evaluate lots of Velocities*)
      {theory, IndIfConstraints}];
    Jobs = (#1 ~ PrepareVelocities ~ #2) &@@@ InputBatch;
    Velocities = VelocityParallel@Jobs;
    Velocities = Riffle[$TheoryNames, Velocities] ~ Partition ~ 2;
    SaveVelocity[theory_String, Velocity_] :=
     Module[{res, PPMArguments, IndIfConstraints},
      DefTheory["Import" → theory];
      $Velocities = Velocity;
      HiGGSPrint["$Velocities value is ", $Velocities];
      HiGGSPrint[" ** StudyTheory: Exporting the binary at " <>
        FileNameJoin@{"svy", theory <> ".thr.mx"}];
      (FileNameJoin@{$WorkingDirectory, "svy", theory <> ".thr.mx"}) ~
       DumpSave ~ {$TheoryName, $Theory, $ToTheory, $ToShellFreedoms,
        $StrengthPShellToStrengthPO3, $PiPShellToPiPPO3, $TheoryCDPiPToCDPiPO3,
        $TheoryPiPToPiPO3, $IfConstraintToTheoryNesterForm,
        $IfConstraints, $InertVelocity, $ToOrderRules, $PPM, $Velocities};
     ];
    HiGGSPrint[Velocities];
    SaveVelocity[#1, #2] &@@@ Velocities;
   ];
  ];
ClearBuild[];
```

documentation

Build documentation

documentation

Export this notebook as the documentation:

documentation

In[1]:=

```
(**)
FrontEndExecute@{FrontEndToken[InputNotebook[], "SelectAll"],
   FrontEndToken[InputNotebook[], "SelectionOpenAllGroups"]};
Export[NotebookDirectory[] <> "Documentation/HiGGS_sources.pdf",
  EvaluationNotebook[]];
(**)
```