Hamiltonian Gauge Gravity Surveyor (HiGGS)

Source notebook for the package file

```
xAct`HiGGS`$Version = {"1.0.0-beta", {2022, 2, 1}};
```

Initialisation

GNU public license

```
(* HiGGS, Hamiltonian analysis of Poincare gauge theory *)

(* Copyright (C) 2022 Will E. V. Barker *)

(* This program is free software; you can redistribute it and/or
  modify it under the terms of the GNU General Public License as
  published by the Free Software Foundation; either version 2 of
  the License,or (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place-Suite 330, Boston, MA 02111-1307,
  USA.
*)
```

Information

```
(* :Title: HiGGS *)

(* :Author: Will E. V. Barker *)

(* :Summary: Hamiltonian analysis of Poincare gauge theory *)

(* :Brief Discussion:
   - tbc
*)

(* :Context: xAct`HiGGS` *)

(* :Package Version: 1.0.0 *)

(* :Copyright: Will E. V. Barker (2022) *)

(* :History: see HiGGS.History *)

(* :Keywords: *)

(* :Source: HiGGS.nb *)

(* :Mathematica Version: 11.3 and later *)

(* :Limitations:
    - many *)
```

Dependencies

Require contexts from the rest of xAct:

In[•]:=

```
BeginPackage["xAct`HiGGS`",
  {"xAct`xTensor`", "xAct`xPerm`", "xAct`xCore`", "xAct`xTras`"}];
```

Require that sub-kernels load HiGGS, too

```
ParallelNeeds["xAct`HiGGS`"];
```

Continually scroll to the last line of the evaluation:

In[•]:=

```
SetOptions[$FrontEndSession, EvaluationCompletionAction → "ScrollToOutput"];
```

Welcome message:

```
Print["Package xAct`HiGGS`  version ", $Version[[1]], ", ", $Version[[2]]];
Print[
  "CopyRight (C) 2022, Will E. V. Barker, under the General Public License."];
Print[xAct`xCore`Private`bars];
Print["This free version of HiGGS is an open source dependent
    of the xAct bundle, but NOT an official part thereof."];
Print["This free version of HiGGS incorporates Cyril Pitrou's code from
    the public repository at https://github.com/xAct-contrib/examples."];
Print[xAct`xCore`Private`bars];
```

Was a node variable defined before the HiGGS package was loaded? If not, set it to the empty string.

```
(*
If[!ValueQ@Global`$Timing,
  Global`$Timing=False;
  Global`$Node="";
 ];
*)
Print["Some hard-to-suppress error messages may appear below..."];
Quiet[
  DistributeDefinitions@$Timing;
  DistributeDefinitions@Global`$Timing;
  If[! ValueQ@$Node,
   $Node = Global`$Node;
   If[! ValueQ@$Node, $Node = ""];
   DistributeDefinitions@$Node;
   DistributeDefinitions@Global`$Node;
  ];
 ];
Print["...and that should be it: no further errors should appear below here."];
Print[xAct`xCore`Private`bars];
(*,Print["issues"],
{$Node::shdw,Global`$Node::shdw,$Timing::shdw,Global`$Timing::shdw}*)
```

Find the install directory:

```
(*Because the developer version of HiGGS is not installed,
and sits locally, we need this*)
(*was Needs called on the HiGGS package from a notebook?*)
If[NotebookDirectory[] == $Failed,
  $WorkingDirectory = Directory[];, $WorkingDirectory = NotebookDirectory[];,
  $WorkingDirectory = NotebookDirectory[];];
Print["The working directory is " <> $WorkingDirectory];
$Path~AppendTo~$WorkingDirectory;
$HiGGSInstallDirectory =
  Select[FileNameJoin[{#, "xAct/HiGGS"}] & /@ $Path, DirectoryQ][[1]];
Print["At least one HiGGS installation directory was found at " <>
   $HiGGSInstallDirectory <> "."];
Print[xAct`xCore`Private`bars];
```

Set up run options:

```
ActiveCellTags = {"build"};
UnitTests = {"CheckOrthogonalityToggle", "ShowIrrepsToggle",
   "ProjectionNormalisationsCheckToggle", "ShowIrrepsToggle", "documentation"};
PrematureCellTags = {"TransferCouplingsPerpPerpToggle",
   "TransferCouplingsPerpParaToggle"};
BinaryNames = {"O13ProjectionsToggle", "CompleteO3ProjectionsToggle",
   "ProjectionNormalisationsToggle", "CanonicalPhiToggle",
   "NonCanonicalPhiToggle", "ChiPerpToggle", "ChiSingToggle",
   "GeneralComplementsToggle", "CDPiPToCDPiPO3",
   "NesterFormIfConstraints", "VelocityToggle"};
BuiltBinaries = BinaryNames~Select~(FileExistsQ@
      FileNameJoin@{$HiGGSInstallDirectory, "bin/build/" <> # <> ".mx"} &);
ActiveCellTags = ActiveCellTags~Join~(BinaryNames~Complement~BuiltBinaries);
```

Stack trace

```
(*time when the package is called*)
$HiGGSBuildTime = AbsoluteTime[];
(*set up a file to record the start time of a job*)
$BuildTimeFilename = Quiet@
   FileNameJoin@{$WorkingDirectory, "svy", "node-" <> $Node, "peta4.chr.mx"};
(*is this the first kernel launched in the job? if so,
record start time to file, otherwise import the file*)
Quiet@If[! FileExistsQ@$BuildTimeFilename,
   $BuildTimeFilename~DumpSave~{$HiGGSBuildTime},
   ToExpression@("<<" <> $BuildTimeFilename <> ";");
  ];
(*return time since start time*)
HiGGSAbsoluteTime[] := Module[{}, AbsoluteTime[] - $HiGGSBuildTime];
```

```
(*remember to modify this if you want
 to time another function in HiGGS_sources.nb *)
$TimedFunctionList = {"BuildHiGGS", "DefTheory", "Velocity", "PoissonBracket",
   "DeclareOrder", "ToOrderCanonical", "VarAction", "ToNewCanonical"};
(*initial zeroes, i.e. the default line*)
$HiGGSTimingLine = 0.~ConstantArray~(20 * 2 Length@$TimedFunctionList);
```

```
(*which kernel are we in? This sets the file in which we record stats*)
$HiGGSTimingFile =
  Quiet@FileNameJoin@{$WorkingDirectory, "svy", "node-" <> $Node, "chr",
     "kernel-" <> ToString@$KernelID <> ".chr.csv"};
(*a function which writes all current data to the kernel file*)
WriteHiGGSTimingData[] := Module[{HiGGSOutputStream},
   (*open the stream*)
   HiGGSOutputStream = OpenAppend[$HiGGSTimingFile];
   WriteString[HiGGSOutputStream, ExportString[#, "CSV"]] &@$HiGGSTimingData;
   Close[HiGGSOutputStream];
   (*Zero the data again,
   so that we don't have always to be carrying it around*)
   $HiGGSTimingData = {};
  ];
```

```
(*headers for the timing file*)
$HiGGSTimingData = {};
(*$HiGGSTimingData~AppendTo~
 Flatten@(Flatten@(({#,#})&/@$TimedFunctionList)~ConstantArray~10)*)
$HiGGSTimingData~AppendTo~$HiGGSTimingLine;
(*open the kernel files and write the function headers*)
Quiet[WriteHiGGSTimingData[]];
```

```
(*Try timing, i.e. this only works to print to file once every $PauseSeconds*)
$PauseSeconds = 6;
$LastMultiple = 0;
TryTiming[] := Module[{PrintDamper, HiGGSOutputStream, printer},
   PrintDamper = AbsoluteTime[];
   If[(Ceiling@PrintDamper~Divisible~$PauseSeconds) &&
     ! (Ceiling@PrintDamper / $PauseSeconds == $LastMultiple),
    printer = PrintTemporary[" ** TryTiming: recording timing statistics"];
    (*
    $HiGGSTimingFile~Export~$HiGGSTimingData;
    *)
    (*do all the writing here*)
    WriteHiGGSTimingData[];
    (*log the last multiple of seconds on which we were allowed to print*)
    $LastMultiple = Ceiling@PrintDamper / $PauseSeconds;
    NotebookDelete[printer];
   ];
  ];
```

```
(*This is redefined only when the theory batch is introduced,
but only needed beyond that point anyway*)
Quiet@ToExpression["<<" <> FileNameJoin@
     {$WorkingDirectory, "svy", "node-" <> $Node, "peta4.nom.mx"} <> ";"];
```

```
(*don't try timing until we call the function in expr*)
TimeWrapper~SetAttributes~HoldAll;
(*the actual timing function*)
TimeWrapper[Label_String, expr_] :=
  Module[{res, temp, TimingNowPosition, TimingDurationPosition,
    $HiGGSTimingNow, $HiGGSTimingDuration, NewHiGGSTimingLine, PrintDamper},
   If[Global`$Timing,
    $HiGGSTimingNow = HiGGSAbsoluteTime[];
    (*Label=ToString@Head@expr;*)
    (*nothing wrong with this, but we'll include it later*)
    res = AbsoluteTiming@expr;
    temp = Evaluate@res[[2]];
    $HiGGSTimingDuration = Evaluate@res[[1]];
    If[StringQ@$TheoryName,
     TimingDurationPosition = (2 Length@$TimedFunctionList)
         (($TheoryNames~Position~$TheoryName)[[1]][[1]]) +
        2 ((Flatten@($TimedFunctionList~Position~Label))[[1]]);,
     TimingDurationPosition = 2 ((Flatten@($TimedFunctionList~Position~Label))[[
          1]]);,
     TimingDurationPosition = 2 ((Flatten@($TimedFunctionList~Position~Label))[[
          1]]);];
    TimingNowPosition = TimingDurationPosition - 1;
    NewHiGGSTimingLine = $HiGGSTimingLine~
      ReplacePart~(TimingDurationPosition -> $HiGGSTimingDuration);
    NewHiGGSTimingLine = NewHiGGSTimingLine~ReplacePart~
      (TimingNowPosition -> $HiGGSTimingNow);
    $HiGGSTimingData~AppendTo~NewHiGGSTimingLine;
    (*need to be careful not to spend all our time printing *)
    TryTiming[];,
    temp = Evaluate@expr,
    temp = Evaluate@expr];
   temp];
```

```
ForceTiming[] := WriteHiGGSTimingData[];
```

Package

```
BuildHiGGS::usage = "Rebuild the HiGGS session";
ToNesterForm::usage = "Express quantity in terms of human-readable irreps";
ToBasicForm::usage = "Express quantity in terms of basic gauge fields";
PoissonBracket::usage = "Calculate a Poisson bracket between two quantities";
DefTheory::usage = "Define a theory using a system
    of equations to constrain the coupling coefficients";
UndefTheory::usage = "Undefine a theory using a system of
    equations to constrain the coupling coefficients";
StudyTheory::usage = "Calculate the links in the constraint
    chain down do a certain level";
Velocity::usage = "Calculate the velocity of a quantity with
    respect to the Hamiltonian indicated by DefTheory";
```

Global variables:

In[•]:=

```
$Theory::usage = "The gauge theory as defined by a system
    of equations which constrains the coupling coefficients";
```

Private

```
Begin["xAct`HiGGS`Private`"];
```

Build the HiGGS session, which contains all the physics

```
(*HiGGS cannot build itself more than once,
since xAct does not forgive mutability...!*)
$HiGGSBuilt = False;
BuildHiGGS::built = "The HiGGS environment has already been built.";
BuildHiGGS[] :=
  "BuildHiGGS" ~ TimeWrapper ~ Catch@Module[{PriorMemory, UsedMemory},
     (*A message*)
     xAct`xTensor`Private`MakeDefInfo[
      BuildHiGGS, $KernelID, {"HiGGS environment for kernel", ""}];
     (*Check for pre-existing build*)
     If[$HiGGSBuilt, Throw@Message[BuildHiGGS::built]];
     (*List of all print cells
      in front end before this notebook starts to run*)
     $PrintCellsBeforeStartBuildHiGGS = Flatten@
       Cells[SelectedNotebook[], CellStyle → {"Print"}];
     PriorMemory = MemoryInUse[];
     Print[" ** BuildHiGGS: RAM used by kernel ", $KernelID, " is ",
      Dynamic[Refresh[MemoryInUse[], UpdateInterval → 1]], " bytes."];
     Print[" ** BuildHiGGS: Building session from ",
      FileNameJoin@{$HiGGSInstallDirectory, "HiGGS_sources.nb"},
      " with active CellTags ", ActiveCellTags, "."];
     (*NotebookEvaluate[FileNameJoin@{$HiGGSInstallDirectory,
        "HiGGS_sources.nb"},InsertResults→False];*)
     (*NotebookEvaluate[FileNameJoin@{$HiGGSInstallDirectory,
        "HiGGS_sources.nb"},EvaluationElements→
       "Tags"->ActiveCellTags,InsertResults→False];*)
     Get[FileNameJoin@{$HiGGSInstallDirectory, "HiGGS_sources.m"}];
     Print[
      " ** BuildHiGGS: The context on quitting HiGGS.m is ", $Context, "."];
     (*Purge all cells created during build process*)
     Pause[2];
     UsedMemory = MemoryInUse[] - PriorMemory;
     NotebookDelete@(Flatten@Cells[SelectedNotebook[], CellStyle → {"Print"}] ~
        Complement ~ $PrintCellsBeforeStartBuildHiGGS);
     Print[" ** BuildHiGGS: If build was successful, the
        HiGGS environment is now ready to use and is
        occupying ", UsedMemory, " bytes in RAM."];
     $HiGGSBuilt = True;
    ];
```

In[3]:=

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

```
End[];
EndPackage[];
```