Package for the calculation of bound time-like geodesics and their properties in Kerr spacetime

Define usage for public functions

```
BeginPackage["KerrGeodesics`",
    {"KerrGeodesics`ConstantsOfMotion`",
     "KerrGeodesics`OrbitalFrequencies`",
     "KerrGeodesics`SpecialOrbits`",
     "KerrGeodesics`KerrGeoOrbit`"}];

$KerrGeodesicsInformation::usage = "$KerrGeodesicsInformation is a list of rules
$KerrGeodesicsInstallationDirectory::usage = "$KerrGeodesicsInstallationDirectory

$KerrGeodesicsVersionNumber::usage = "$KerrGeodesicsVersionNumber is a real numbe
$KerrGeodesicsReleaseNumber::usage = "$KerrGeodesicsReleaseNumber is an integer w
$KerrGeodesicsVersion::usage = "$KerrGeodesicsVersionNumber is a string that give

Begin["`Private`"];

(**********************************************************)
(* Package version information                            *)
(**********************************************************)

$KerrGeodesicsInstallationDirectory = FileNameDrop[FindFile["KerrGeodesics`"], -2

$KerrGeodesicsVersionNumber        = 1.0;
$KerrGeodesicsReleaseNumber        = 0;

$KerrGeodesicsVersion :=
 Module[{path, version, release, buildid, gitrev, gitdir},
  path = $KerrGeodesicsInstallationDirectory;
  version = ToString[NumberForm[$KerrGeodesicsVersionNumber, {Infinity, 1}]];
  release = ToString[$KerrGeodesicsReleaseNumber];
  buildid = Quiet@ReadList[FileNameJoin[{path, "BUILD_ID"}], "String"];
  If[SameQ[buildid, $Failed],
    buildid = "";
  ,
    buildid = " (" <> First[buildid] <> ")";
  ];

  (* First, check for a GIT_REVISION file. If it exists, use its contents as the
  gitrev = Quiet@ReadList[FileNameJoin[{path, "GIT_REVISION"}],"String"];

  (* Otherwise, try to determine the git revision directly *)
  If[SameQ[gitrev, $Failed],
    gitdir = FileNameJoin[{path, ".git"}];
    If[FileType[gitdir] === Directory,
      gitrev = Quiet@ReadList["!git --git-dir "<>gitdir<>" rev-parse HEAD", Strin
      If[gitrev === {}, gitrev = $Failed];
    ];
  ];

  (* If it worked, ReadList returns a list but we just want the first element (li
  If[Head[gitrev] === List, gitrev = First[gitrev]];

  (* Check we have a git revision and otherwise give up trying *)
  If[Head[gitrev] === String && StringMatchQ[gitrev, RegularExpression["[0-9a-f]{

  version <> "." <> release <> buildid <> gitrev
]

$KerrGeodesicsInformation :=
  {"InstallationDirectory" -> $KerrGeodesicsInstallationDirectory,
   "Version" -> $KerrGeodesicsVersion,
   "VersionNumber" -> $KerrGeodesicsVersionNumber,
   "ReleaseNumber" -> $KerrGeodesicsReleaseNumber}
```

Roots of the radial and polar equations

```
(* Returns the roots of the radial equation, as given by Fujita and Hikida *)
KerrGeoRadialRoots[a_, p_, e_, x_, En1_:Null,Q1_:Null] := Module[{M=1,En=En1,Q=Q1
If[En==Null, En = KerrGeoEnergy[a, p, e, x]];
If[Q==Null,  Q = KerrGeoCarterConstant[a, p, e, x]];

r1=p/(1-e);
r2=p/(1+e);
AplusB=(2M)/(1-En^2)-(r1+r2);(*Eq. (11)*)
AB=(a^2 Q)/((1-En^2)r1 r2);(*Eq. (11)*)
r3=(AplusB+Sqrt[(AplusB)^2-4AB])/2;(*Eq. (11)*)
r4=AB/r3;

{r1,r2,r3,r4}

]
```

This code uses the polar equation (z^2-zm^2)(a^2(1-E0^2)z^2-zp^2)==0 as the Polar equation. Hence zp is a*Sqrt[1-E0^2]*zp in other sources.

```
KerrGeoPolarRoots[a_, p_, e_, x_] := Module[{En,L,Q,zm,zp},
  {En,L,Q} = Values[KerrGeoConstantsOfMotion[a, p, e, x]];
  zm = Sqrt[1-x^2];
  zp = (a^2 (1-En^2)+L^2/(1-zm^2))^(1/2);
  {zp,zm}
]
```

Close the package

```
End[];

EndPackage[];
```