Preliminaries

The usual

```
ClearAll["Global`*"]
Needs["xAct`xTensor`"];
Needs["xAct`xCoba`"];
Needs["xAct`TexAct`"];
$DefInfoQ=$UndefInfoQ=False;
DefManifold[M3,3,{α,β,γ,δ}];
```

Definitions

Scalar functions and constants

```
DefConstantSymbol@{m,M,a,σ,k};
DefScalarFunction@{ν,λ,rs,ρc}
```

Abbreviations et cetera

```
minkowskiLimit={ν→0,λ→0};
kerrAbbreviations={Σ→r[]^2+a^2 Cos[θ[]]^2,Δ→r[]^2-2M r[]+a^2,curlyA→Σ Δ+2M r[]
```

```
nBL:=CTensor[{Sqrt[g22[r[],θ[]]/(g11[r[],θ[]](g11[r[],θ[]] (D[rs[θ[]],θ[]])^2+g
nWSC:=CTensor[{Sqrt[g22[r[],θ[]]/(g11[r[],θ[]](g11[r[],θ[]] (D[rs[θ[]],θ[]])^2-
nWCC:=CTensor[{Sqrt[g22[ρ[],z[]]/(g11[ρ[],z[]](g11[ρ[],z[]] (D[ρc[z[]],z[]])^2-
```

Assumptions

```
$Assumptions={ν[r[],θ[]]∈PositiveReals,ν[ρ[],z[]]∈PositiveReals,λ[r[],θ[]]∈Pos
```

The Kerr case

```
DefChart[BL,M3,{1,2,3},{r[],θ[],ϕ[]},ChartColor→Blue];
kerrMetricBL=CTensor[{{Σ/Δ,0,0},{0,Σ,0},{0,0,curlyA/Σ Sin[θ[]]^2}},{-BL,-BL}]/
kerrMetricBL[-μ,-ν]
SetCMetric[kerrMetricBL,BL,SignatureOfMetric→{3,0,0}]
CD3=LC[kerrMetricBL];
```

```
metricCoefficientsBL={g11[r[],θ[]]→kerrMetricBL⟦1,1,1⟧,g22[r[],θ[]]→kerrMetric
metricCoefficientsBLfunction={g11→((a^2 Cos[#2]^2+#1^2)/(a^2-2 M #1+#1^2)&),g2
normalBL[α]=nBL[α]/.metricCoefficientsBL//FullSimplify
```

```
eqnBL=Simplify[-CD3[-α]@normalBL[α]==0,$Assumptions]/.r[]→rs[θ[]]
H=Simplify[1/2* -CD3[-α]@normalBL[α],$Assumptions]/.r[]→rs[θ[]];
```

```
UnsetCMetric[kerrMetricBL]
CD3=.
```

The general Weyl case with spherical Weyl coordinates

```
DefChart[wsc,M3,{1,2,3},{r[],θ[],ϕ[]},ChartColor→Green];
```

```
weylWSC=CTensor[{{Exp[-2*ν]*Exp[2*λ],0,0},{0,r[]^2Exp[-2*ν]*Exp[2*λ],0},{0,0,r
weylWSC[-μ,-ν]
SetCMetric[weylWSC,wsc,SignatureOfMetric→{3,0,0}]
CD3=LC[weylWSC];
```

```
metricCoefficientsWSC={g11[r[],θ[]]→weylWSC⟦1,1,1⟧,g22[r[],θ[]]→weylWSC⟦1,2,2⟧
metricCoefficientsWSCfunction={g11→(Exp[2(λ[#1,#2]-ν[#1,#2])]&),g22→(#1^2Exp[2
normalWSC[α]=nWSC[α]/.metricCoefficientsWSC//FullSimplify
```

```
eqnWSC=Simplify[-CD3[-α]@normalWSC[α]==0,r[]∈PositiveReals]/.r[]→rs[θ[]]
```

```
UnsetCMetric[weylWSC]
CD3=.
```

The general Weyl case with cylindrical Weyl coordinates

```
DefChart[wcc,M3,{1,2,3},{ρ[],z[],ϕ[]},ChartColor→Red];
```

```
weylWCC=CTensor[{{Exp[-2*ν]*Exp[2*λ],0,0},{0,Exp[-2*ν]*Exp[2*λ],0},{0,0,ρ[]^2*E
weylWCC[-μ,-ν]
SetCMetric[weylWCC,wcc,SignatureOfMetric→{3,0,0}]
CD3=LC[weylWCC];
```

```
metricCoefficientsWCC={g11[ρ[],z[]]→weylWCC⟦1,1,1⟧,g22[ρ[],z[]]→weylWCC⟦1,2,2⟧
metricCoefficientsWCCfunction={g11→(Exp[2(λ[#1,#2]-ν[#1,#2])]&),g22→(Exp[2(λ[#
normalWCC[α]=nWCC[α]/.metricCoefficientsWCC//FullSimplify
```

```
eqnWCC=Simplify[-CD3[-α]@normalWCC[α]==0,$Assumptions]/.ρ[]→ρc[z[]]
```

The Majumdar-Papapetrou disc case

```
νλMajumdarPapapetrouDisc={ν[ρc_[z_],z_]:→(-Log[1+(2*M*EllipticK[Sqrt[((2*a*ρc[z
```

```
eqnMP=eqnWCC/.νλMajumdarPapapetrouDisc;
eqnMP=eqnWCC/.νλMajumdarPapapetrouDisc/.{Derivative[1,0][ν][ρc[z[]],z[]]:→D[νλM
```

```
UnsetCMetric[weylWCC]
CD3=.
```

The Minkowski limit

```
eqnWSCMinkowski=eqnWSC/.minkowskiLimit//FullSimplify
eqnWCCMinkowski= eqnWCC/.minkowskiLimit//FullSimplify
```

```
catenoid[C_,u_,z_]:={C Cosh[z/C]Cos[u],C Cosh[z/C]Sin[u],z}
catenary[C_,z_]:={C Cosh[z/C],0,z}
valuesOfC={0.5,1,1.5,2};
commonPlotRangeCatenoid={{-2.5,2.5},{-2.5,2.5},{-1.5,1.5}};
plotsCatenoid=Table[Show[ParametricPlot3D[catenoid[C,u,z],{u,0,2 π},{z,-1.5,1.5
finalImageCatenoid=GraphicsGrid[{{plotsCatenoid⟦1⟧,plotsCatenoid⟦2⟧},{plotsCate
```

```
RValues={1,4,7};
ϕConstant=7 π/5;
ϕRange={0,2 π};
zRange={-3,3};
commonPlotRange={{-10,10},{-10,10},zRange};
fromNewCoordinates[R_,ϕ_,z_]:={R Cosh[z/R] Cos[ϕ],R Cosh[z/R] Sin[ϕ],z}
catenoid[R_,ϕ_,z_]:={R Cosh[z/R] Cos[ϕ],R Cosh[z/R] Sin[ϕ],z}
catenary[R_,z_]:={R Cosh[z/R] Cos[ϕConstant],R Cosh[z/R] Sin[ϕConstant],z}
cartesianPlotCatenoids=Table[Show[ParametricPlot3D[catenoid[R,ϕ,z],{ϕ,0,2 π},{z
newCoordinatePlotCatenoids=Table[Show[ParametricPlot3D[{R,ϕ,z},{ϕ,0,2 π},{z,zRa
GraphicsGrid[{{Show[{cartesianPlotCatenoids},ImageSize→Large],Show[{newCoordina
```

Cross check against the paper of Krivan & Herold

```
eqnKHWCC=1/2(2g11[ρ[],z[]] g22[ρ[],z[]] g33[ρ[],z[]] D[ρc[z[]],z[],z[]]+(D[ρc[z
(*eqnKHWSC=1/2(2g11[r[],θ[]] g22[r[],θ[]] g33[r[],θ[]] D[rs[θ[]],θ[],θ[]]+(D[rs
(*eqnKHBL=1/2(2g11[r[],θ[]] g22[r[],θ[]] g33[r[],θ[]] D[rs[θ[]],θ[],θ[]]+(D[rs
```

```
eqnKHWCC-eqnWCC//Simplify
(*eqnKHWSC-eqnWSC//Simplify*)
(*eqnKHBL-eqnBL*)
```

Cylindrical solutions: Testing solution, Levi-Civita, Curzon-Chazy

```
DefManifold[M4,4,{b,c,d}];
```

```
DefChart[wcc4,M4,{0,1,2,3},{t[],ρ[],z[],ϕ[]},ChartColor→Red];
```

```
νFunction=Log[#1^2]&;
λFunction=Log[#1^4]&;
weylWCC4=CTensor[{{-Exp[2*ν[ρ[],z[]]],0,0,0},{0,Exp[-2*ν[ρ[],z[]]]*Exp[2*λ[ρ[]
weylWCC4[-b,-c]
SetCMetric[weylWCC4,wcc4,SignatureOfMetric→{3,1,0}]
CD4=LC[weylWCC4];
```

```
Einstein[CD4]==0//FullSimplify
```

```
UnsetCMetric[weylWCC4]
CD4=.
```

Levi-Civita

```
νFunction=2σ Log[#1]&;
λFunction=4σ^2 Log[#1]+Log[k]&;
weylWCC4=CTensor[{{-Exp[2*ν],0,0,0},{0,Exp[-2*ν]*Exp[2*λ],0,0},{0,0,Exp[-2*ν]*E
weylWCC4[-b,-c]
SetCMetric[weylWCC4,wcc4,SignatureOfMetric→{3,1,0}]
CD4=LC[weylWCC4];
eqnLC=eqnWCC/.{ν→νFunction,λ→λFunction}//Simplify
```

```
Einstein[CD4]==0//FullSimplify
```

```
UnsetCMetric[weylWCC4]
CD4=.
```

Curzon-Chazy

```
νFunction=-M/Sqrt[#1^2+#2^2]&;
λFunction=-M^2 #1^2/(2(#1^2+#2^2)^2)&;
weylWCC4=CTensor[{{-Exp[2*ν],0,0,0},{0,Exp[-2*ν]*Exp[2*λ],0,0},{0,0,Exp[-2*ν]*E
weylWCC4[-b,-c]
SetCMetric[weylWCC4,wcc4,SignatureOfMetric→{3,1,0}];
CD4=LC[weylWCC4];
```

```
Einstein[CD4]==0//FullSimplify
```

```
eqnWCCCC=eqnWCC/.{ν→νFunction,λ→λFunction}//Simplify//FullSimplify
```

```
UnsetCMetric[weylWCC4]
CD4=.
```

Spherical solutions: Curzon-Chazy

```
DefChart[wsc4,M4,{0,1,2,3},{t[],r[],θ[],ϕ[]},ChartColor→Green];
```

```
νFunction=-M/#1&;
λFunction=-M^2Sin[#2]^2/(2#1^2)&;
weylWSC4=CTensor[{{-Exp[2*ν],0,0,0},{0,Exp[-2*ν]*Exp[2*λ],0,0},{0,0,r[]^2Exp[-2
weylWSC4[-μ,-ν]
SetCMetric[weylWSC4,wsc4,SignatureOfMetric→{3,1,0}];
CD4=LC[weylWSC4];
```

```
Einstein[CD4]==0//FullSimplify
```

```
eqnWSCCC=eqnWSC/.{ν→νFunction,λ→λFunction}//Simplify//FullSimplify
```

```
UnsetCMetric[weylWSC4]
CD4=.
```

```
weylWSC4=CTensor[{{-Exp[2*ν[r[],θ[]]],0,0,0},{0,Exp[-2*ν[r[],θ[]]]*Exp[2*λ[r[],
weylWSC4[-μ,-ν]
SetCMetric[weylWSC4,wsc4,SignatureOfMetric→{3,1,0}];
CD4=LC[weylWSC4];
```

```
UnsetCMetric[weylWSC4]
CD4=.
```

Numerical solutions: spherical

Kerr

Reproducing the Krivan & Harold plot

```
M=1;
ϵ=10^(-20);
odeKH=H==-0.15/.{θ[]→θ,a→1};
conditionsKH={rs[ϵ]==5.205,rs'[ϵ]==ϵ};
domain={θ,ϵ,π/2};
```

```
H
```

```
shtSltnKH=NDSolve[{odeKH,conditionsKH},rs,domain,Method→{"Shooting","StartingI
solKH=rs/.shtSltnKH;
plotLeft=Plot[{solKH[θ]},domain,PlotRange→{{0,π/2},{5.16,5.215}},PlotStyle→{Th
plotRight=Plot[H/.{rs[θ[]]→5.190,rs'[θ[]]→0,rs''[θ[]]→0,a→1,M→1}//Simplify,{θ
KHplot=Overlay[{plotLeft, plotRight}]
```

```
conditionsBL={rs[ϵ]==2,rs'[ϵ]==ϵ};
domain={θ,ϵ,π/2};
shtSltnBL=NDSolve[{eqnBL/.{θ[]→θ,a→1},conditionsBL},rs,domain,Method→{"Shootin
solBL=rs/.shtSltnBL;
```

```
plotBL1=ParametricPlot3D[{solBL[θ]*Sin[θ]*Cos[ϕ],solBL[θ]*Sin[θ]*Sin[ϕ],solBL[ϵ
plotBL2=ParametricPlot3D[{2*Sin[θ]*Cos[ϕ],2*Sin[θ]*Sin[ϕ],solBL[θ]*Cos[θ]},{θ,
plotBL=Show[{plotBL1,plotBL2},ViewPoint → {1, 2, 1}]
```

Curzon-Chazy

```
eqnWSCCC
```

```
ϵ=10^(-15);
domain={θ,ϵ,π/2-ϵ};
conditionsWSCCC={rs[ϵ]==0.8,rs'[ϵ]==ϵ};
shtSltnWSCCC=NDSolve[{eqnWSCCC/.{θ[]→θ,M→1},conditionsWSCCC},rs,domain,Method→
solWSCCC=rs/.shtSltnWSCCC;
plotCCWSC1=ParametricPlot3D[
  {solWSCCC[θ]*Sin[θ]*Cos[ϕ], solWSCCC[θ]*Sin[θ]*Sin[ϕ], solWSCCC[θ]*Cos[θ]},
  {θ, 0, Pi/2}, {ϕ, 0, 2 Pi},
  PlotRange → All,Mesh → True,Boxed → True,Axes → True];
plotCCWSC2=ParametricPlot3D[
{solWSCCC[θ]*Sin[θ]*Cos[ϕ], solWSCCC[θ]*Sin[θ]*Sin[ϕ], -solWSCCC[θ]*Cos[θ]},
{θ, 0, Pi/2}, {ϕ, 0, 2 Pi},
PlotRange → All,Mesh → True,Boxed → True,Axes → True];
Show[plotCCWSC1,plotCCWSC2]
```

Numerical solutions: cylindrical

Majumdar-Papapetrou

```
ϵ=10^(-20);
conditionsMP={ρc[ϵ]==1.4,ρc'[ϵ]==ϵ};
domain={z,-2,2};
shtSltnMP=NDSolve[{eqnMP/.{z[]→z,a→1,M→1},conditionsMP},ρc,domain,Method→{"Sho
solMP=ρc/.shtSltnMP;
```

```
plotMP = ParametricPlot3D[
  {solMP[z]*Cos[ϕ], solMP[z]*Sin[ϕ], z},
  domain, {ϕ, 0, 2 Pi},
  PlotRange → All, Mesh → None, Boxed → True, Axes → True,
  PlotStyle → Directive[Opacity[0.5]],
  ColorFunction → "Rainbow", ViewPoint → {0, -5, 0}];
disc = ParametricPlot3D[
  {r*Cos[ϕ], r*Sin[ϕ], 0},
  {r, 0, 1}, {ϕ, 0, 2 Pi},
  PlotStyle → {Black, Opacity[1],Thickness[0.05], Specularity[White, 20]},
  Lighting → "Neutral"];
finalMP=Show[plotMP, disc]
```

```
plotMP = ParametricPlot3D[
  {solMP[z]*Cos[ϕ], solMP[z]*Sin[ϕ], z},
  {z, -2, 2}, {ϕ, 0, 2 Pi},
  PlotRange → All, Mesh → None, Boxed → True, Axes → True,
  PlotStyle → Directive[Opacity[0.5]],
  ColorFunction → "Rainbow",ViewPoint → {1, 2, 1}];
disc = ParametricPlot3D[
  {r*Cos[ϕ], r*Sin[ϕ], 0},
  {r, 0, 1}, {ϕ, 0, 2 Pi},
  PlotStyle → {Black, Opacity[1],Thickness[0.05], Specularity[White, 20]},
  Lighting → "Neutral"
];
plotMPevolve=Show[plotMP, disc]
```

```
conditionsMP={ρc[ϵ]==1.1,ρc'[ϵ]==ϵ};
shtSltnMP=NDSolve[{eqnMP/.{z[]→z,a→1,M→1},conditionsMP},ρc,domain,Method→{"Sho
solMP=ρc/.shtSltnMP;
plot1MP=Plot[solMP[z],{z,-2,2},Frame→True,PlotStyle→{Blue,Thin}];

conditionsMP={ρc[ϵ]==1.2,ρc'[ϵ]==ϵ};
shtSltnMP=NDSolve[{eqnMP/.{z[]→z,a→1,M→1},conditionsMP},ρc,domain,Method→{"Sho
solMP=ρc/.shtSltnMP;
plot2MP=Plot[solMP[z],{z,-2,2},Frame→True,PlotStyle→{Purple,Thin}];

conditionsMP={ρc[ϵ]==1.4,ρc'[ϵ]==ϵ};
shtSltnMP=NDSolve[{eqnMP/.{z[]→z,a→1,M→1},conditionsMP},ρc,domain,Method→{"Sho
solMP=ρc/.shtSltnMP;
plot3MP=Plot[solMP[z],{z,-2,2},Frame→True,PlotStyle→{Green,Thin}];

conditionsMP={ρc[ϵ]==1.6,ρc'[ϵ]==ϵ};
shtSltnMP=NDSolve[{eqnMP/.{z[]→z,a→1,M→1},conditionsMP},ρc,domain,Method→{"Sho
solMP=ρc/.shtSltnMP;
plot4MP=Plot[solMP[z],{z,-2,2},Frame→True,PlotStyle→{Red,Thin}];

conditionsMP={ρc[ϵ]==1.8,ρc'[ϵ]==ϵ};
shtSltnMP=NDSolve[{eqnMP/.{z[]→z,a→1,M→1},conditionsMP},ρc,domain,Method→{"Sho
solMP=ρc/.shtSltnMP;
plot5MP=Plot[solMP[z],{z,-2,2},Frame→True,PlotStyle→{Orange,Thin}];

plotMPall=Show[{plot1MP,plot2MP,plot3MP,plot4MP,plot5MP},GridLines→Automatic,F
```

Levi-Civita

```
eqnLC
```

```
catenoid[u_,z_]:={Cosh[z]Cos[u],Cosh[z]Sin[u],z}
```

```
σ=0;
ϵ=10^(-7);
conditionsLC={ρc[ϵ]⩵1,ρc'[ϵ]⩵ϵ};
domain={z,-2,2};
shtSltnLC=NDSolve[{eqnLC/.z[]→z,conditionsLC},ρc,domain,Method→{"Shooting","St
solLC=ρc/.shtSltnLC;
plotrhocLC=Plot[solLC[z],domain,PlotStyle→Red];
plotCatenary=Plot[Cosh[z],domain,PlotStyle→{Black,Dashed}];
finalLC2=Show[{plotrhocLC,plotCatenary},Frame→True,FrameLabel → {"z", "ρ(z)"},

plotLC1=ParametricPlot3D[{solLC[z]*Cos[ϕ],solLC[z]*Sin[ϕ],z},domain, {ϕ, 0, 2 P
  PlotRange → All,Mesh → None,Boxed → True,Axes → True,ViewPoint → {1, 2, 1}];
plotLC2=ParametricPlot3D[{solLC[z]*Cos[ϕ],solLC[z]*Sin[ϕ],-z},domain, {ϕ, 0, 2
  PlotRange → All,Mesh → True,Boxed → True,Axes → True,PlotStyle→Orange];
plotCatenoid=ParametricPlot3D[catenoid[u,z],{u,0,2 π},domain,Boxed→True,Axes→T
finalLC1=Show[plotLC1,plotCatenoid];
image=GraphicsGrid[{{finalLC1,finalLC2}}, Spacings → {50, 0.5},ImageSize→1000]
```

Curzon-Chazy

```
ϵ=10^(-15);
conditionsWCCCC={ρc[ϵ]⩵1,ρc'[ϵ]⩵ϵ};
domain={z,-2,2};
shtSltnWCCCC=NDSolve[{eqnWCCCC/.{z[]→z,M→1},conditionsWCCCC},ρc,domain,Method→
solWCCCC=ρc/.shtSltnWCCCC;
plotWCCCC=ParametricPlot3D[{solWCCCC[z]*Cos[ϕ],solWCCCC[z]*Sin[ϕ],z},domain, {ϕ
  PlotRange → All,Mesh → True,Boxed → True,Axes → True,ViewPoint → {1, 2, 1},C
Show[plotWCCCC];
plotCC=Plot[solWCCCC[z],domain,PlotStyle→Red];
plotCatenary=Plot[Cosh[z],domain,PlotStyle→{Black,Dashed}];
finalCC=Show[{plotCC,plotCatenary},Frame→True,FrameLabel → {"z", "ρ(z)"},GridL
image=GraphicsGrid[{{plotWCCCC,finalCC}}, Spacings → {50, 0.5},ImageSize→1000]
```