Bondi Metric (DeepSeek)

```
/* ======================================================================
   Bondi{Sachs Metric Configuration for Maxima/ctensor
   Coordinates:
```

$$(u, r, x^A) = [u, r, \theta, \phi]$$

```
   Metric Form:
```

$$ds^2 = -(V/r)\ exp(2b)du^2 - 2\exp(2b)dudr + r^2 h_A B(dx^A - U^A du)(dx^B - U^B du)$$

```
   ====================================================================== */
```

```
(%i2) info:build_info()$info@version;
```

(%o2)

5.38.1

```
(%i2)  reset()$kill(all)$
(%i1)  derivabbrev:true$
(%i2)  ratprint:false$
(%i3)  fpprintprec:5$
(%i4)  load(linearalgebra)$
(%i5)  if get('draw,'version)=false then load(draw)$
(%i6)  wxplot_size:[1024,768]$
(%i7)  if get('itensor,'version)=false then load(itensor)$
(%i8)  imetric(g)$
(%i9)  if get('ctensor,'version)=false then load(ctensor)$
(%i10) if get('rkf45,'version)=false then load(rkf45)$
(%i11) declare(trigsimp,evfun)$
(%i12) declare(s,mainvar)$
```

1

```
(%i16) assume(0≤r)$
       assume(0≤θ,θ≤π)$
       assume(0≤sin(θ))$
       assume(0≤ϕ,ϕ≤2*π)$
(%i18) /* 3.  Declare coordinates explicitly (avoid interactive csetup() for batch scripts) */
       ξ:ct_coords:[u,r,θ,ϕ]$
       dim:length(ct_coords)$
```

(%i21)
```
/* 2.  Declare functional dependencies for metric variables All functions depend on (u,
r, θ, φ) in general */
depends([V, b], ξ)$
depends([U_θ, U_φ], ξ)$
depends([h_θθ, h_θφ, h_φφ], ξ)$
```

(%i22)
```
/* 4.  Define the Bondi metric components as matrix lg This is the canonical form with
all off-diagonal terms */
lg:  matrix( /* u-index (0):  u coordinate */ [ - (V/r) * exp(2*b), /* g_uu */ -
exp(2*b), /* g_ur */ r^2 * h_θθ * (-U_θ), /* g_uθ (off-diag) */ r^2 * h_φφ * (-U_φ) ], /*
g_uφ (off-diag) */
/* r-index (1):  r coordinate */ [ - exp(2*b), /* g_ru = g_ur */ 0, /* g_rr = 0 (Bondi
gauge) */ 0, /* g_rθ = 0 (Bondi gauge) */ 0 ], /* g_rφ = 0 (Bondi gauge) */
/* θ-index (2):  θ coordinate */ [ r^2 * h_θθ * (-U_θ), /* g_θu */ 0, /* g_θr = 0 */ r^2 *
h_θθ, /* g_θθ */ r^2 * h_θφ ], /* g_θφ */
/* φ-index (3):  φ coordinate */ [ r^2 * h_φφ * (-U_φ), /* g_φu */ 0, /* g_φr = 0 */ r^2 *
h_θφ, /* g_φθ = g_θφ */ r^2 * h_φφ ] /* g_φφ */ )$
```

-->
```
/* 5.  (Optional) For axisymmetry, simplify dependencies */ /* Uncomment if you want to
assume no φ-dependence:  depends([V, b, U_θ, h_θθ, h_θφ, h_φφ], [u, r, θ])$ */;
```

(%i23)
```
/* 6.  Set the metric and compute inverse, determinant */
cmetric()$
```

(%i25)
```
/* 7.  Optional:  Display the metric to verify */
print("Bondi metric components entered:")$
print(lg)$
```

Bondi metric components entered:

$$\begin{pmatrix} -\frac{V\,\%e^{2b}}{r} & -\%e^{2b} & -U_{\theta} h_{\theta\theta} r^2 & -U_{\phi} h_{\phi\phi} r^2 \\ -\%e^{2b} & 0 & 0 & 0 \\ -U_{\theta} h_{\theta\theta} r^2 & 0 & h_{\theta\theta} r^2 & h_{\theta\phi} r^2 \\ -U_{\phi} h_{\phi\phi} r^2 & 0 & h_{\theta\phi} r^2 & h_{\phi\phi} r^2 \end{pmatrix}$$

(%i26)
```
/* 8.  Compute Christoffel symbols (first kind if desired) */
/* For null geodesics or field equations */
christof(false)$
```

(%i29)
```
riemann(false)$
lriemann(false)$
uriemann(false)$
```

(%i33)
```
/* 9.  Compute Ricci tensor, Ricci scalar, Einstein tensor */ /* WARNING: This is
extremely algebraically intensive!  */;
ric:zeromatrix(dim,dim)$
ricci(false)$
uric:zeromatrix(dim,dim)$
uricci(false)$
```

(%i37)
```
ein:zeromatrix(dim,dim)$
einstein(false)$
lein:zeromatrix(dim,dim)$
leinstein(false)$
```

(%i38)
```
cgeodesic(false)$
```

Reduce Order

```
(%i40) cv_coords:[T,R,Θ,Φ]$
       depends(cv_coords,s)$
```

```
(%i44) gradef(u,s,T)$
       gradef(r,s,R)$
       gradef(θ,s,Θ)$
       gradef(ϕ,s,Φ)$
```

Geodesic

```
(%i45) cgeodesic(false)$
```

```
(%i46) for i thru dim do geod[i]:fullratsimp(geod[i])$
```

Solve for second derivative of coordinates

```
(%i47) geodsol:linsolve(listarray(geod),diff(ξ,s,2))$
```

```
-->    /* 10.  (Alternative) Work with Taylor series expansion in 1/r for asymptotic analysis.
       Uncomment to use:  ctayswitch:  true$
       ctayvar:  r$
       ctaypov:  2$ // expand to order 1/r^2
       ctaypt:  inf$ // expansion about r = infinity
       csetup(); // re-enter metric in series mode */;
```

```
(%i50) print("Bondi metric configuration complete.")$
       print("NOTE: Full symbolic Ricci computation may be too large.")$
       print("Consider using series expansion (1/r) for asymptotics.")$
```

Bondi metric configuration complete.

NOTE: Full symbolic Ricci computation may be too large.

Consider using series expansion (1/r) for asymptotics.

Numerical solution

```
(%i51) if get('rkf45,'version)=false then load(rkf45)$
```

```
(%i52) params:[b=0.001,V=1.0,U_θ=1.0,U_ϕ=1.0,h_θθ=1.0,h_ϕϕ=1.0,h_θϕ=0.1]$
```

```
(%i59) funcs:append(ct_coords,cv_coords)$ldisplay(funcs)$
       initial:[0,12,π/2,π/4,1.0,-400.0,-0.01,-0.01]$ldisplay(initial)$
       odes:append(cv_coords,map(rhs,geodsol))$
       interval:[s,0,0.05]$ldisplay(interval)$
```

$$funcs = [u, r, \theta, \phi, T, R, \mathit{\Theta}, \mathit{\Phi}] \tag{\%t54}$$

$$initial = \left[0, 12, \frac{\pi}{2}, \frac{\pi}{4}, 1.0, -400.0, -0.01, -0.01\right] \tag{\%t56}$$

$$interval = [s, 0, 0.05] \tag{\%t59}$$

```
(%i60) rksol:rkf45(odes,funcs,initial,interval, absolute_tolerance=1E-12,report=true),params$
```

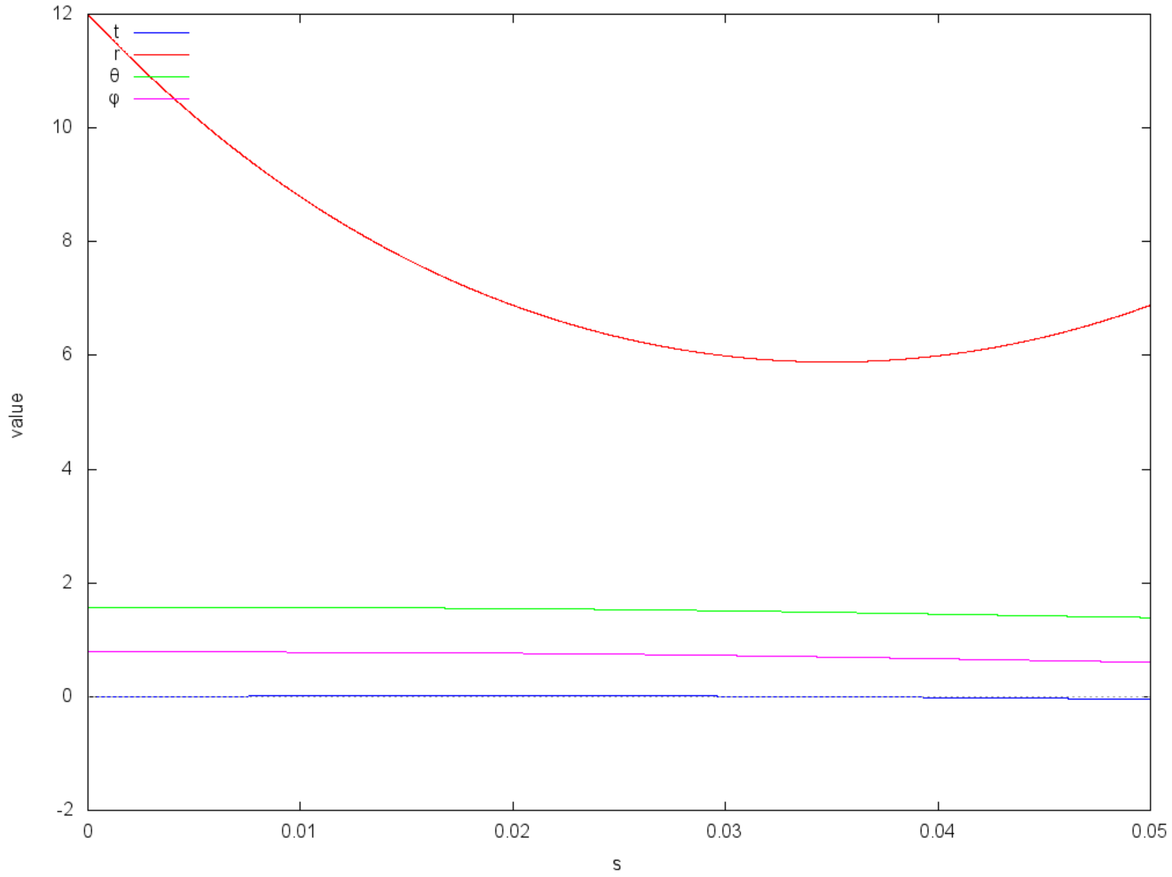
Info: rkf45:
Integration points selected:5220
Total number of iterations:6347
Bad steps corrected:1128
Minimum estimated error:$1.0175\,10^{-15}$
Maximum estimated error:$9.9999\,10^{-13}$
Minimum integration step taken:$3.5811\,10^{-6}$
Maximum integration step taken:$2.2717\,10^{-5}$

---

map(ldisp,odes:ev(odes,params))$

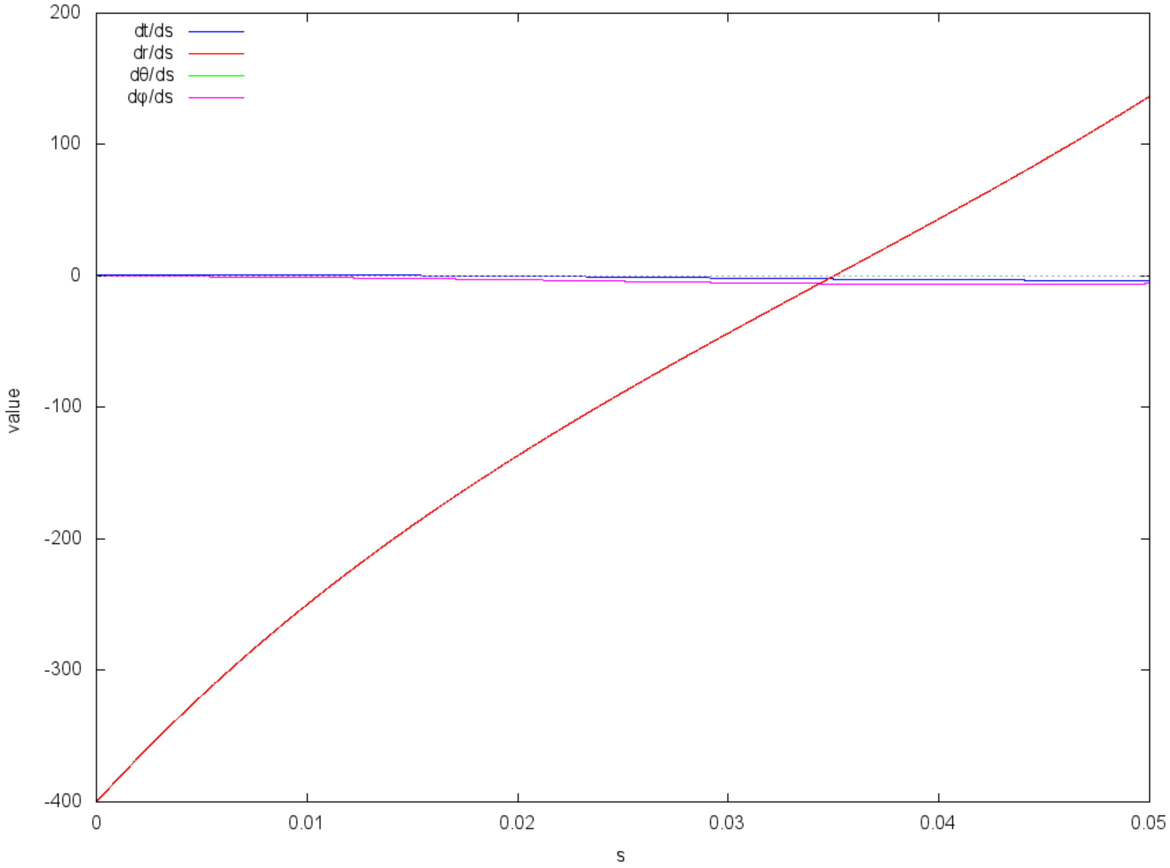
table_form(rksol,column_names=append([s],funcs))$

(%i61)
```
wxplot2d([[discrete,map(lambda([u],part(u,[1,2])),rksol)], [discrete,map(lambda([u],part(u,[1,3]
[discrete,map(lambda([u],part(u,[1,4])),rksol)], [discrete,map(lambda([u],part(u,[1,5])),rksol)]
[style,[lines,1]],[xlabel,"s"],[ylabel,"value"], [legend,"t","r","θ","ϕ"],
[gnuplot_preamble,"set key top left"])$
```



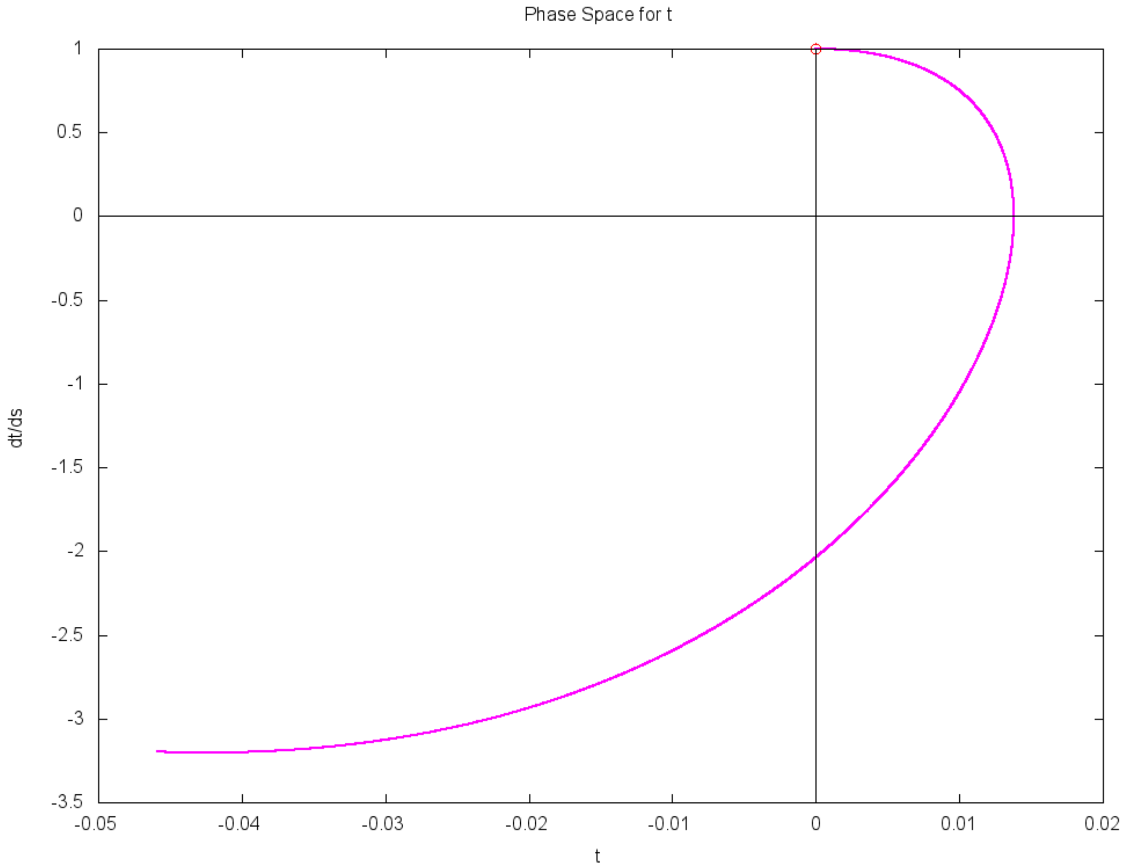
(%t61)

```
(%i62) wxplot2d([[discrete,map(lambda([u],part(u,[1,6])),rksol)], [discrete,map(lambda([u],part(u,[1,7]
       [discrete,map(lambda([u],part(u,[1,8])),rksol)], [discrete,map(lambda([u],part(u,[1,9])),rksol)]
       [style,[lines,1]],[xlabel,"s"],[ylabel,"value"], [legend,"dt/ds","dr/ds","dθ/ds","dϕ/ds"],
       [gnuplot_preamble,"set key top left"])$
```



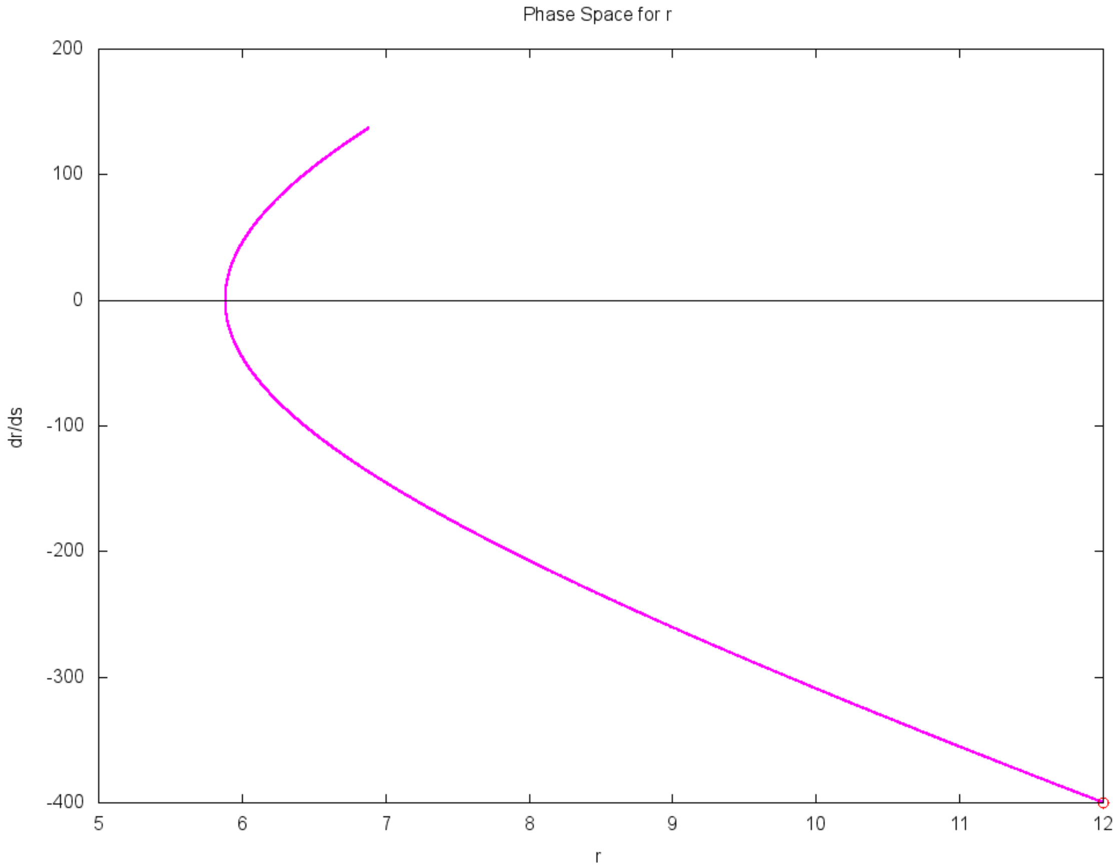
(%t62)

```
(%i63) wxplot2d([[discrete,map(lambda([u],part(u,[2,6])),rksol)], [discrete,[part(initial,[1,5])]]],[ax
       [title,"Phase Space for t"],[point_type,circle], [style,[lines,2],[points,3]],[color,magenta,red]
       [xlabel,"t"],[ylabel,"dt/ds"],[legend,false])$
```



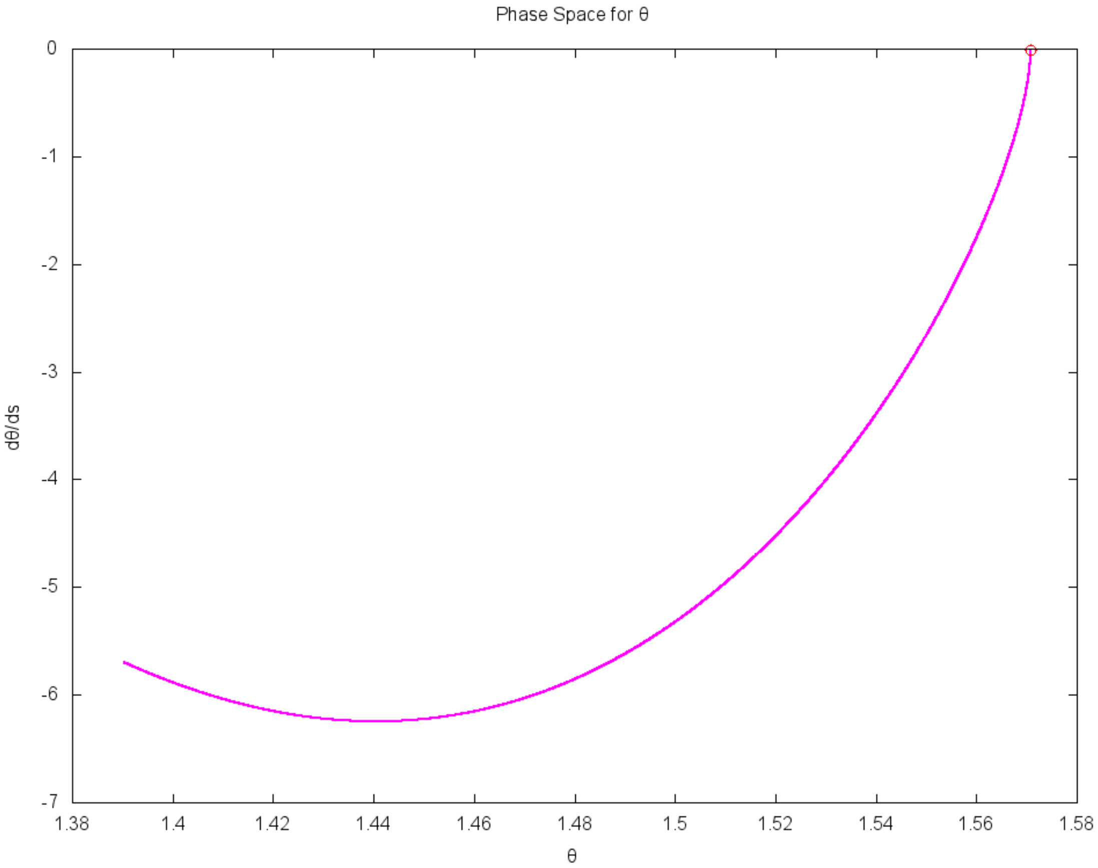
(%t63)

```
(%i64) wxplot2d([[discrete,map(lambda([u],part(u,[3,7])),rksol)], [discrete,[part(initial,[2,6])]]],[ax
       [title,"Phase Space for r"],[point_type,circle], [style,[lines,2],[points,3]],[color,magenta,red]
       [xlabel,"r"],[ylabel,"dr/ds"],[legend,false])$
```



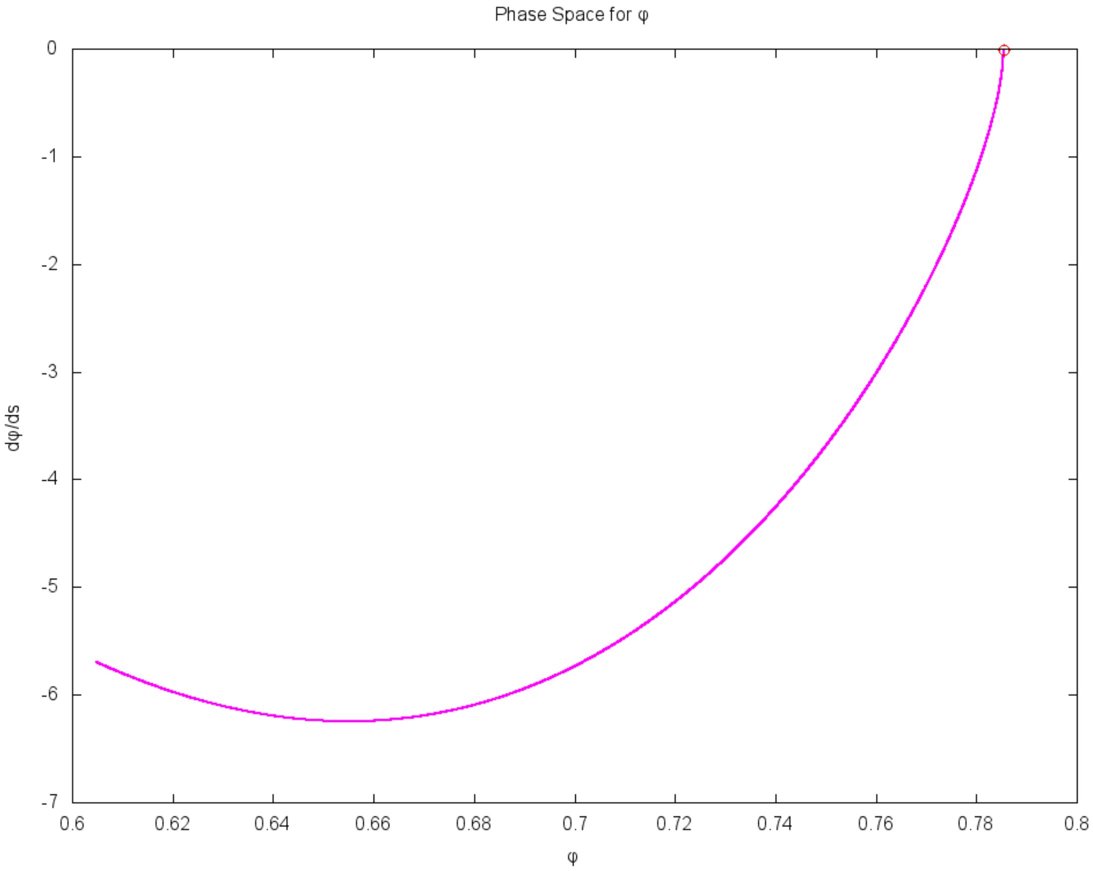
(%t64)

```
(%i65) wxplot2d([[discrete,map(lambda([u],part(u,[4,8])),rksol)], [discrete,[part(initial,[3,7])]]],[axe
       [title,"Phase Space for θ"],[point_type,circle], [style,[lines,2],[points,3]],[color,magenta,red]
       [xlabel,"θ"],[ylabel,"dθ/ds"],[legend,false])$
```



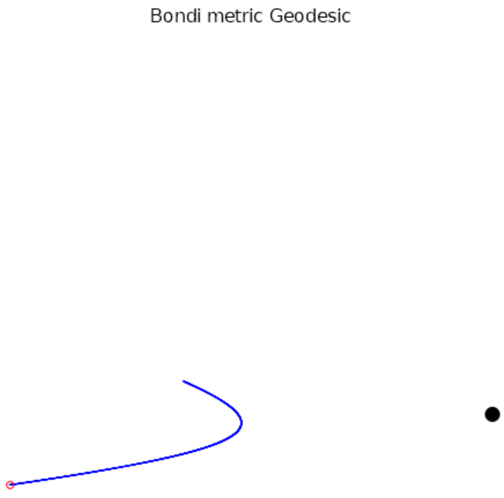
(%t65)

```
(%i66) wxplot2d([[discrete,map(lambda([u],part(u,[5,9])),rksol)], [discrete,[part(initial,[4,8])]]],[axe
       [title,"Phase Space for ϕ"],[point_type,circle], [style,[lines,2],[points,3]],[color,magenta,red]
       [xlabel,"ϕ"],[ylabel,"dϕ/ds"],[legend,false])$
```



(%t66)

```
(%i67) draw3d(title = "Bondi metric Geodesic", proportional_axes = xyz, axis_3d = false, xlabel
       = "", ylabel = "", zlabel = "", dimensions = wxplot_size, view = [80,185], file_name =
       "Bondi_metric_Geodesic", terminal = 'pngcairo,
       transform = [r*sin(θ)*cos(φ),r*sin(θ)*sin(φ),r*cos(θ),r,θ,φ],
       color = blue, point_size = 1, point_type = -1, points_joined = true,
       points(map(lambda([u],part(u,[3,4,5])),rksol)),
       color = red, point_size = 1, point_type = circle, points_joined = false,
       points([part(initial,[2,3,4])]),
       color = black, point_size = 2, point_type = filled_circle, points([[0,0,0]])),params$
(%i68) show_image("Bondi_metric_Geodesic.png")$
```



(%t68)

Minimal Radius

```
(%i69) ldisplay(r_m:lmin(map(lambda([u],part(u,3)),rksol)))$
```

$$r_m = 5.8843$$ (%t69)

at proper time

```
(%i70) ldisplay(s_m:assoc(r_m,map(lambda([u],part(u,[3,1])),rksol)))$
```

$$s_m = 0.03499$$ (%t70)

at coordinate time

```
(%i71) ldisplay(t_m:assoc(r_m,map(lambda([u],part(u,[3,2])),rksol)))$
```

$$t_m = -0.0024632 \tag{\%t71}$$

Clean up

```
(%i75) forget(0≤r)$
       forget(0≤θ,θ≤π)$
       forget(0≤sin(θ))$
       forget(0≤φ,φ≤2*π)$
```