KerrGeodesicsC

Generated by Doxygen 1.8.13

Contents

1 korb.h

This software describes the generic bound motion of a test mass around a Kerr black hole. There are also functions for calculating r from the tortoise coordinate via root finding. The main function takes inputs: eccentric (boolean), inclined (boolean), a (double), p (double), e (double), x (double), lambdaSteps (integer), lambdaMax (double), and from those arguments calculates and prints the constants of motion, orbital frequencies, and position at equally spaced Mino times. The original implementation of this code was in support of the following work: https↩://arxiv.org/abs/1905.13237 Please consider citing the above paper if you make use of this code. All descriptions involving numbered equations in this code's documentation are in reference to the above paper.

Author

Thomas Osburn

Version

0.1

Date

2019

Copyright

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Data Structure Documentation

4.1 korb_params Struct Reference

```
#include <korb.h>
```

Data Fields

- int eccentric
- int inclined
- double a
- double p
- double e
- double x
- double thmin
- double E
- double Lz
- double Q
- double B
- double zm
- double zp
- double r1
- double r2
- double r3
- double r4
- double p3
- double p4
- double Vr
- double Vth
- double Yr
- double Yth

- double Yphi
- double Ga
- double wr
- double wth
- double wphi
- double * dladpsiamps
- int dladpsinum
- double * dladchiamps
- int dladchinum
- double * dpsidlaamps
- int dpsidlanum
- double * dchidlaamps
- int dchidlanum
- double * Tramps
- int Trnum
- double * Tthamps
- int Tthnum
- double * Pramps
- int Prnum
- double * Pthamps
- int Pthnum
- double err

4.1.1 Field Documentation

4.1.1.1 eccentric

`int korb_params::eccentric`

Boolean to indicate whether the orbit is eccentric

4.1.1.2 inclined

`int korb_params::inclined`

Boolean to indicate whether the orbit is inclined

4.1.1.3 a

`double korb_params::a`

specific spin angular momentum divided by black hole mass

4.1.1.4 p

`double korb_params::p`

Orbital semilatus rectum

4.1.1.5 e

`double korb_params::e`

Orbital eccentricity

4.1.1.6 x

`double korb_params::x`

x = sin(theta_min)

4.1.1.7 thmin

`double korb_params::thmin`

Minimum polar angle

4.1.1.8 E

`double korb_params::E`

Specific energy

4.1.1.9 Lz

`double korb_params::Lz`

Specific angular momentum

4.1.1.10 Q

`double korb_params::Q`

Carter constant

4.1.1.11 B

`double korb_params::B`

Beta = (1-E^2)*a^2

4.1.1.12 zm

`double korb_params::zm`

z_- = cos^2(theta_min)

4.1.1.13 zp

`double korb_params::zp`

z_+ = other root of polar equation of motion (not directly related to theta_max)

4.1.1.14 r1

```
double korb_params::r1
```

r_1 = largest root of radial equation of motion = r_max

4.1.1.15 r2

```
double korb_params::r2
```

r_2 = 2nd largest root of radial equation of motion = r_min

4.1.1.16 r3

```
double korb_params::r3
```

r_3 = 3rd largest root of radial equation of motion (not directly related to a turning point)

4.1.1.17 r4

```
double korb_params::r4
```

r_4 = smallest root of radial equation of motion (not directly related to a turning point)

4.1.1.18 p3

```
double korb_params::p3
```

p_3 = r_3∗(1-e)

4.1.1.19 p4

```
double korb_params::p4
```

p_4 = r_4∗(1+e)

4.1.1.20 Vr

```
double korb_params::Vr
```

Lambda_r = radial period in Mino time

4.1.1.21 Vth

```
double korb_params::Vth
```

Lambda_theta = polar period in Mino time

4.1.1.22 Yr

```
double korb_params::Yr
```

Upsilon_r = fundamental radial angular frequency in Mino time

4.1.1.23 Yth

```
double korb_params::Yth
```

Upsilon_theta = fundamental polar angular frequency in Mino time

4.1.1.24 Yphi

```
double korb_params::Yphi
```

Upsilon_phi = fundamental azimuthal angular frequency in Mino time

4.1.1.25 Ga

```
double korb_params::Ga
```

Gamma = average rate of t advance in Mino time

4.1.1.26 wr

```
double korb_params::wr
```

Omega_r = fundamental radial angular frequency in t

4.1.1.27 wth

```
double korb_params::wth
```

Omega_theta = fundamental polar angular frequency in t

4.1.1.28 wphi

```
double korb_params::wphi
```

Omega_phi = fundamental azimuthal angular frequency in t

4.1.1.29 dladpsiamps

```
double* korb_params::dladpsiamps
```

Fourier amplitudes of the derivative of Mino time with respect to chi_r

4.1.1.30 dladpsinum

```
int korb_params::dladpsinum
```

number of Fourier amplitudes of the derivative of Mino time with respect to chi_r

4.1.1.31 dladchiamps

```
double* korb_params::dladchiamps
```

Fourier amplitudes of the derivative of Mino time with respect to chi_theta

4.1.1.32 dladchinum

```
int korb_params::dladchinum
```

number of Fourier amplitudes of the derivative of Mino time with respect to chi_theta

4.1.1.33 dpsidlaamps

```
double* korb_params::dpsidlaamps
```

Fourier amplitudes of the derivative of chi_r with respect to Mino time

4.1.1.34 dpsidlanum

```
int korb_params::dpsidlanum
```

number of Fourier amplitudes of the derivative of chi_r with respect to Mino time

4.1.1.35 dchidlaamps

```
double* korb_params::dchidlaamps
```

Fourier amplitudes of the derivative of chi_theta with respect to Mino time

4.1.1.36 dchidlanum

```
int korb_params::dchidlanum
```

number of Fourier amplitudes of the derivative of chi_theta with respect to Mino time

4.1.1.37 Tramps

```
double* korb_params::Tramps
```

Fourier amplitudes of T_r

4.1.1.38 Trnum

```
int korb_params::Trnum
```

number of Fourier amplitudes of T_r

4.1.1.39 Tthamps

```
double* korb_params::Tthamps
```

Fourier amplitudes of T_theta

4.1.1.40 Tthnum

```
int korb_params::Tthnum
```

number of Fourier amplitudes of T_theta

4.1.1.41 Pramps

`double* korb_params::Pramps`

Fourier amplitudes of Psi_r

4.1.1.42 Prnum

`int korb_params::Prnum`

number of Fourier amplitudes of Psi_r

4.1.1.43 Pthamps

`double* korb_params::Pthamps`

Fourier amplitudes of Psi_theta

4.1.1.44 Pthnum

`int korb_params::Pthnum`

number of Fourier amplitudes of Psi_theta

4.1.1.45 err

`double korb_params::err`

relative error tolerance used to determine convergence of each Fourier series

The documentation for this struct was generated from the following file:

- korb.h

5 File Documentation

5.1 korb.h File Reference

Data Structures

- struct korb_params

Typedefs

- typedef struct korb_params korb_params

Functions

- int korb_getparams (int eccentric, int inclined, double a, double p, double e0, double x0, double err, korb_params *orbpar)
- int korb_freepar (korb_params orbpar)
- double korb_zfromchi (double chi, korb_params orbpar)
- double korb_thfromla (double la, korb_params orbpar)
- double korb_rfrompsi (double psi, korb_params orbpar)
- double korb_dladchi (double chi, korb_params orbpar)
- double korb_dladpsi (double psi, korb_params orbpar)
- double korb_dchidla (double chi, korb_params orbpar)
- double korb_dpsidla (double psi, korb_params orbpar)
- double korb_lafrompsi (double psi, korb_params orbpar)
- double korb_lafromchi (double chi, korb_params orbpar)
- double korb_D (double r, korb_params orbpar)
- double korb_Tr (double psi, korb_params orbpar)
- double complex korb_Trint (double psi, int n, korb_params orbpar)
- double korb_Tth (double chi, korb_params orbpar)
- double complex korb_Tthint (double chi, int n, korb_params orbpar)
- double korb_Pr (double psi, korb_params orbpar)
- double complex korb_Print (double psi, int n, korb_params orbpar)
- double korb_Pth (double chi, korb_params orbpar)
- double complex korb_Pthint (double chi, int n, korb_params orbpar)
- double complex korb_dchidlaint (double chi, int n, korb_params orbpar)
- double complex korb_dpsidlaint (double psi, int n, korb_params orbpar)
- double korb_dtrfromla (double la, korb_params orbpar)
- double korb_dtthfromla (double la, korb_params orbpar)
- double korb_psifromla (double la, korb_params orbpar)
- double korb_chifromla (double la, korb_params orbpar)
- double korb_tfromla (double la, korb_params orbpar)
- double korb_dphirfromla (double la, korb_params orbpar)
- double korb_dphithfromla (double la, korb_params orbpar)
- double korb_phifromla (double la, korb_params orbpar)
- int korb_specint (double complex *amp, double large, int n, korb_params orbpar, double complex(*func)(double, int, korb_params))
- int korb_dft (int *N, korb_params orbpar, double complex(*func)(double, int, korb_params), double **amps)
- int korb_dct (int num, double period, korb_params orbpar, double(*func)(double, korb_params), double amps[ ])
- int korb_getamps (int *num, double period, korb_params orbpar, double(*func)(double, korb_params), double **amps)
- double korb_drsdr (double rm, double a)
- double korb_rsfromrsubtrplus (double rsubtrplus, double a)
- double korb_rmf (double rm, void *params)
- double korb_rmdf (double rm, void *params)
- void korb_rmfdf (double rm, void *params, double *y, double *dy)
- double korb_rsubtrplusfromrs (double rs, double a)

5.1.1 Typedef Documentation

5.1.1.1 korb_params

```
typedef struct korb_params korb_params
```

This data structure encodes all information needed to reconstruct the position of a test mass orbiting a Kerr black hole at any time via Fourier series. The pointers within will become lists of Fourier coefficients once they are allocated by the function "korb_getparams"

5.1.2 Function Documentation

5.1.2.1 korb_getparams()

```
int korb_getparams (
            int eccentric,
            int inclined,
            double a,
            double p,
            double e0,
            double x0,
            double err,
            korb_params * orbpar )
```

This function gathers all necessary numerical techniques to determine generic orbital trajectories of a test mass around a Kerr black hole. The general strategy involves spectral calculation of Fourier coefficients describing the rate-of-change of position, then the position itself is determined by integrating the Fourier series term-by-term. All necessary information for reconstructing the position at a certain time is encoded in the "korb_params" data structure. It adaptively handles convergence testing and memory allocation (but a separate function is needed to free the allocated memory).

5.1.2.2 korb_freepar()

```
int korb_freepar (
            korb_params orbpar )
```

This function frees the memory allocated by the function "korb_getparams"

5.1.2.3 korb_zfromchi()

```
double korb_zfromchi (
            double chi,
            korb_params orbpar )
```

Find z = cos^2(theta) from chi_theta (Eq. 2.26)

5.1.2.4 korb_thfromla()

```
double korb_thfromla (
            double la,
            korb_params orbpar )
```

Find theta from Mino time

5.1.2.5 korb_rfrompsi()

```
double korb_rfrompsi (
            double psi,
            korb_params orbpar )
```

Find r from chi_r (Eq. 2.26)

5.1.2.6 korb_dladchi()

```
double korb_dladchi (
            double chi,
            korb_params orbpar )
```

Find the derivative of Mino time with respect to chi_theta (Eq. 2.29)

5.1.2.7 korb_dladpsi()

```
double korb_dladpsi (
            double psi,
            korb_params orbpar )
```

Find the derivative of Mino time with respect to chi_r (Eq. 2.28)

5.1.2.8 korb_dchidla()

```
double korb_dchidla (
            double chi,
            korb_params orbpar )
```

Find the derivative of chi_theta with respect to Mino time from chi_theta

5.1.2.9 korb_dpsidla()

```
double korb_dpsidla (
            double psi,
            korb_params orbpar )
```

Find the derivative of chi_r with respect to Mino time from chi_r

5.1.2.10 korb_lafrompsi()

```
double korb_lafrompsi (
            double psi,
            korb_params orbpar )
```

Find Mino time from chi_r using a Fourier series (Eq. 3.5)

5.1.2.11 korb_lafromchi()

```
double korb_lafromchi (
            double chi,
            korb_params orbpar )
```

Find Mino time from chi_theta using a Fourier series

5.1.2.12 korb_D()

```
double korb_D (
            double r,
            korb_params orbpar )
```

Find Delta from r

5.1.2.13 korb_Tr()

```
double korb_Tr (
            double psi,
            korb_params orbpar )
```

Find T_r, which is the radial part of the derivative of t with respect to Mino time (Eq. 2.19)

5.1.2.14 korb_Trint()

```
double complex korb_Trint (
            double psi,
            int n,
            korb_params orbpar )
```

This function provides the integrand for calculating Mino time Fourier coefficients of T_r

5.1.2.15 korb_Tth()

```
double korb_Tth (
            double chi,
            korb_params orbpar )
```

Find T_th, which is the polar part of the derivative of t with respect to Mino time (Eq. 2.20)

5.1.2.16 korb_Tthint()

```
double complex korb_Tthint (
            double chi,
            int n,
            korb_params orbpar )
```

This function provides the integrand for calculating Mino time Fourier coefficients of T_th

5.1.2.17 korb_Pr()

```
double korb_Pr (
            double psi,
            korb_params orbpar )
```

Find Psi_r, which is the radial part of the derivative of phi with respect to Mino time (Eq. 2.17)

5.1.2.18 korb_Print()

```
double complex korb_Print (
            double psi,
            int n,
            korb_params orbpar )
```

This function provides the integrand for calculating Mino time Fourier coefficients of Psi_r

5.1.2.19 korb_Pth()

```
double korb_Pth (
            double chi,
            korb_params orbpar )
```

Find Psi_theta, which is the polar part of the derivative of phi with respect to Mino time (Eq. 2.18)

5.1.2.20 korb_Pthint()

```
double complex korb_Pthint (
            double chi,
            int n,
            korb_params orbpar )
```

This function provides the integrand for calculating Mino time Fourier coefficients of Psi_theta

5.1.2.21 korb_dchidlaint()

```
double complex korb_dchidlaint (
            double chi,
            int n,
            korb_params orbpar )
```

This function provides the integrand for calculating Mino time Fourier coefficients of the derivative of chi_theta with respect to Mino time

5.1.2.22 korb_dpsidlaint()

```
double complex korb_dpsidlaint (
            double psi,
            int n,
            korb_params orbpar )
```

This function provides the integrand for calculating Mino time Fourier coefficients of the derivative of chi_r with respect to Mino time

5.1.2.23 korb_dtrfromla()

```
double korb_dtrfromla (
            double la,
            korb_params orbpar )
```

This function returns the integral of T_r with respect to Mino time given the Fourier coefficients of T_r (Eq. 3.19)

5.1.2.24 korb_dtthfromla()

```
double korb_dtthfromla (
            double la,
            korb_params orbpar )
```

This function returns the integral of T_theta with respect to Mino time given the Fourier coefficients of T_theta (Eq. 3.20)

5.1.2.25 korb_psifromla()

```
double korb_psifromla (
            double la,
            korb_params orbpar )
```

Find chi_r from Mino time

5.1.2.26 korb_chifromla()

```
double korb_chifromla (
            double la,
            korb_params orbpar )
```

Find chi_theta from Mino time

5.1.2.27 korb_tfromla()

```
double korb_tfromla (
            double la,
            korb_params orbpar )
```

Find t from Mino time (Eq. 2.36)

5.1.2.28 korb_dphirfromla()

```
double korb_dphirfromla (
            double la,
            korb_params orbpar )
```

This function returns the integral of Psi_r with respect to Mino time given the Fourier coefficients of Psi_r (Eq. 3.21)

5.1.2.29 korb_dphithfromla()

```
double korb_dphithfromla (
            double la,
            korb_params orbpar )
```

This function returns the integral of Psi_theta with respect to Mino time given the Fourier coefficients of Psi_theta (Eq. 3.22)

5.1.2.30 korb_phifromla()

```
double korb_phifromla (
            double la,
            korb_params orbpar )
```

Find phi from Mino time (Eq. 2.37)

5.1.2.31 korb_specint()

```
int korb_specint (
            double complex * amp,
            double large,
            int n,
            korb_params orbpar,
            double complex(*)(double, int, korb_params) func )
```

Takes a function pointer and finds a single Fourier coefficient via DFT

5.1.2.32 korb_dft()

```
int korb_dft (
            int * N,
            korb_params orbpar,
            double complex(*)(double, int, korb_params) func,
            double ** amps )
```

Takes a function pointer and finds a set of Fourier coefficients via DFT (including convergence assessment)

5.1.2.33 korb_dct()

```
int korb_dct (
            int num,
            double period,
            korb_params orbpar,
            double(*)(double, korb_params) func,
            double amps[ ] )
```

Takes a function pointer and samples it for DCT via FFTW

5.1.2.34 korb_getamps()

```
int korb_getamps (
            int * num,
            double period,
            korb_params orbpar,
            double(*)(double, korb_params) func,
            double ** amps )
```

Takes a function pointer and calculates DCT coefficients via FFTW (including convergence assessment)

5.1.2.35 korb_drsdr()

```
double korb_drsdr (
            double rm,
            double a )
```

Gives the derivative of the tortoise coordinate with respect to r (Eq. 2.50)

5.1.2.36 korb_rsfromrsubtrplus()

```
double korb_rsfromrsubtrplus (
            double rsubtrplus,
            double a )
```

Gives the tortoise coordinate from r-r_+ (Eq. 2.49)

5.1.2.37 korb_rmf()

```
double korb_rmf (
            double rm,
            void * params )
```

Residual between guessing the tortoise coordinate value and the actual value (for root finding)

5.1.2.38 korb_rmdf()

```
double korb_rmdf (
            double rm,
            void * params )
```

Gives the derivative of the tortoise coordinate with respect to r (for root finding)

5.1.2.39 korb_rmfdf()

```
void korb_rmfdf (
            double rm,
            void * params,
            double * y,
            double * dy )
```

Gives both the residual between guessing the tortoise coordinate value and the actual value and the derivative of the tortoise coordinate with respect to r (for root finding)

5.1.2.40 korb_rsubtrplusfromrs()

```
double korb_rsubtrplusfromrs (
            double rs,
            double a )
```

Uses various initial guess strategies to efficiently and accurately find r-r_+ from the tortoise coordinate via root finding

Index